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 *Type 1 Error* occurs when H_0 is actually true but we rejected it. In some cases, we have multiple individual hypotheses which gives us a *Family-wise error rate (FWER)*, the probability of making any Type 1 errors at all.

P(Making an error) $= \alpha$ P(Not making an error) $= 1 - \alpha$ P(Not making an error in k tests) $= (1 - \alpha)^k$ P(Making at least 1 error in k tests) $= 1 - (1 - \alpha)^k$

However, 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.

K means tests

A simple hypothesis test compares the means of $k = 2$ 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 $1 - (1 - \alpha)^k$, which is why as k increases the FWER also increases.

Bonferroni Method

The Bonferroni method guarantees $FWER \leq \alpha$ by decreasing the level for all the individual tests to α/n.

P(any Type 1 error) $\leq \sum_{i=1}^{n}$ P(Type 1 error for test i) $\leq \sum_{i=1}^{n} \frac{\alpha}{n} = \alpha$

- works even if the test statistics are not independent
- very conservative if n is large

Holm-Bonferroni Method

Holm suggests that we sort the tests in order of their obtained p-values. The tests are ordered from the one with the smallest p-value to the one with the largest. The test with the lowest probability is tested first with the Bonferroni method for a family of C tests. If the test is not significant, then the procedure stops. The second test is then tested with the Bonferroni method but for a family of $(C - 1)$ tests. This process continues until a non-significant test is obtained or when all the tests have been performed

- This is a much more powerful approach the Bonferroni

Boole's Inequality

Boole's inquality 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: $\mathrm{P}(E_1) \leq \mathrm{P}(E_1)$

For a collection of n events: $E_1, ..., E_n$: $\mathrm{P}(\cup_{i=1}^{\infty} E_i) \leq \sum_{i=1}^{n}(E_i)$

Recall that $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

We apply it to $A = \cup_{i=1}^{n} E_i$ and $B = E_{n+1}$ and using the associativity of the union $\cup_{i=1}^{n+1} E_i = A \cup B$, we get that:

$\mathrm{P}(\cup_{i=1}^{n+1} E_i) = \mathrm{P}(\cup_{i=1}^{n} E_i) + \mathrm{P}(E_{n+1})$ - $\mathrm{P}(\cup_{i=1}^{n} E_i \cap E_{n+1})$

Since $\mathrm{P}(\cup_{i=1}^{n} E_i \cap E_{n+1}) \geq 0$,

by the first axiom of probability we have, $\mathrm{P}(\cup_{i=1}^{n+1} E_i) \leq \mathrm{P}(\cup_{i=1}^{n} E_i) + \mathrm{P}(E_{n+1})$,

and therefore

$\mathrm{P}(\cup_{i=1}^{n+1} E_i) \leq \sum_{i=1}^{n} P(E_i) + P(E_{n+1}) = \sum_{i=1}^{n+1} P(E_i)$

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, $\bar{y}$. 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 $\hat{\theta}_i = \bar{y} + c(y - \bar{y})$ or as $\hat{\theta}_i - \hat{\beta}\bar{y} + (1 - \hat{\beta})y_i$ for $i = 1, ..k$. y_i is the observed average for an individual, $\bar{y}$ is the average of the y_i's, and $\hat{\beta}$ is a value between 0 and 1 estimated from data. $c = 1 - \hat{\beta}$.

If $\hat{\beta} = 1$ then all of the θ_i's are estimated by their $\bar{y}$ values. This would occur if all of the y_i's were very close to each other.

The quantity $(y - \bar{y})$ is the amount which the player's average differs from the overall average. Therefor this equation says that the estimator (z) differs from the overall average by $c(y - \bar{y})$, where $c = 1 - \frac{(k-3)\sigma^2}{\sum(y-\bar{y})^2}$, is the shrinking factor.

In our example, $\bar{y} = 0.265$, $c = 0.212$ and $\hat{\beta} = 0.7884$. Therefore $\hat{\theta}_i = 0.265 + 0.212(y - 0.265)$ or $\hat{\theta}_i = 0.7884(0.265) + (1 - 0.7884)y_i$. 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 $\hat{\theta}_i = 0.265 + 0.212(0.400 - 0.265) = 0.294$ or $\hat{\theta}_i = 0.788(0.265) + 0.212(0.400) = 0.294$.

The true average of a batter can be designated as θ. To compare our estimators, y and z, we can calculate the total squared error, given by the sum of $(\theta - y_i)^2$ and $(\theta - z_i)^2$ for each player i. In our sample data, the total squared error for $y = 0.077$ whereas Stein's estimate has an error of 0.022. This would imply that the Stein estimate is much more accurate, especially in the case that the true means lie near each other.

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 k averages.

Stein's Paradox Simulation

We will first go through the simulation where $k = 3$ Phil got a ratio of 1.09 with theta = c(0,1,2) meaning the expected sum of squares using the yi's is about 9% larger than using the Stein estimate and shrinking towards mu = 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)
```

Now lets look at the k = 4 example. There is not much improvement in small k problems and there is less improvement for more variable yi's are relative to V and the further the thetai's are from the assumed value mu but it should always show some improvement, no matter the thetai's

For the first example phil got a ratio of 1.07, second 1.1

```r
#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)
```

```r
#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)
```

Sources

https://statweb.stanford.edu/~joftius/slides/testing.pdf Efrom & Morris 1977: Stein's Paradox Everson - Steins Paradox