Experimentwise error rate in CRDs

In Section 8, we learned how to perform multiple comparisons of mean pairs. Suppose each comparison uses a level of αI for the type I error rate in a hypothesis test. The probability of making AT LEAST one type I error for all multiple comparisons is called the experimentwise type I error rate. In today's lab, we are going to show through simulations what the experimentwise type I error rate will be using the LSD, Bonferroni, and HSD multiple comparison procedures. First, consider the following example:

Example: Fertilizers

Suppose we are conducting an experiment to study the effect of fertilizers on the yield of a particular variety of wheat. There are 12 field locations available for the experiment. Four fields each receive one of the 3 (A, B, or C) fertilizer treatments in a random manner.

|  |  |  |
| --- | --- | --- |
| A | C | A |
| C | B | B |
| B | B | A |
| C | A | C |

Thus, we are interested in testing:

Ho: μA = μB = μC

Ha: At least one pair of means are not equal

To study the type I error rate, we need to set all population means to the same value so that Ho is true. In this example, we let μA = μB = μC = 500 and the population standard deviation  = 50 (you can choose other values but the results should be similar). Because there are three factor levels (treatments), we need to make the following comparisons:

1. μA – μB
2. μB – μC
3. μA – μC

For each factor level (fertilizer type), we simulate 4 observations from a normal distribution with mean 500 and standard deviation 50. Based on this data, we perform multiple comparisons using one of the three procedures (LSD, Bonferroni, and HSD) and calculate the p-values for all mean pairs. This process is repeated 1000 times. In the end, we count the total number of times when at least one of the mean pairs has a p-value less than αI = 0.05 (i.e., making at least one type I error for all multiple comparisons). This number divided by 1000 is then an estimate of the experimentwise type I error rate.

For the LSD procedure, the following code is used to carry out the process described above.

> set.seed(1234)

> numb.data.sets <- 1000

> mu <- 500

> sd <- 50

> alphaI <- 0.05

> y.ij <- matrix(data = rnorm(n = 4\*3\*numb.data.sets, mean = mu, sd = sd), nrow =

 numb.data.sets, ncol = 4\*3)

> factor.levels <- rep(x = c("A","B","C"), each = 4)

> head(y.ij)

 [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]

[1,] 439.6467 439.7333 451.3091 480.7464 451.8563 458.7428 501.1696 512.4940

[2,] 513.8715 515.0733 495.0184 525.7028 459.3144 517.3584 489.7549 524.0960

[3,] 554.2221 423.0427 494.4632 515.4002 490.2744 453.9954 447.5299 470.7975

[4,] 382.7151 531.7685 559.6097 591.9577 595.8006 485.6332 500.8445 552.1356

[5,] 521.4562 535.1476 417.2057 579.6702 535.0885 472.4435 484.6829 569.2272

[6,] 525.3028 404.7059 447.7178 482.0902 548.5275 542.4323 508.0951 491.5868

 [,9] [,10] [,11] [,12]

[1,] 485.0338 525.5266 409.1551 411.3220

[2,] 481.8801 500.2468 531.3583 588.8032

[3,] 453.7564 477.2327 525.9046 501.3697

[4,] 519.7880 446.7974 507.0461 453.1365

[5,] 514.0707 429.9162 572.8636 518.6649

[6,] 540.4705 471.6761 475.3202 398.4487

> factor.levels

 [1] "A" "A" "A" "A" "B" "B" "B" "B" "C" "C" "C" "C"

We have seen the use of matrix function in our previous labs. Now each row of y.ij contains a sample of size n = 12. Next, we wrote the following function LSD.pvalue() to calculate the p-values for each sample using the LSD procedure.

> LSD.pvalue <- function(yield, fertilizer) {

 data1 <- data.frame(yield, fertilizer)

 save.LSD <- pairwise.t.test(x = data1$yield, g = data1$fertilizer,

 p.adjust.method = "none", alternative = "two.sided")

 as.numeric(save.LSD$p.value)

 }

> LSD.pvalue(yield = y.ij[1,], fertilizer = factor.levels)

[1] 0.3337015 0.8630726 NA 0.4205321

Notice how we use pairwise.t.test() inside our function to get the p-values. To show it works, we apply the function to the first row of our generated data y.ij. If we run the pairwise.t.test() function in R directly, we can see how the NA value comes about in the output of our function.

> data1 <- data.frame(yield = y.ij[1,], fertilizer = factor.levels)

> save.LSD <- pairwise.t.test(x = data1$yield, g = data1$fertilizer,

 p.adjust.method = "none", alternative = "two.sided")

> save.LSD$p.value

 A B

B 0.3337015 NA

C 0.8630726 0.4205321

Next we use the apply() function to calculate the p-values for all 1000 samples and save the results in a matrix object p.values. The colSums() function then finds how many of the three p-values is less than 0.05 for *each* column of p.values. Finally, mean(check > 0) will give the proportion of all the samples with at least one p-value less than αI = 0.05.

> p.values <- apply(X = y.ij, MARGIN = 1, FUN = LSD.pvalue, fertilizer =

 factor.levels)

> p.values[,1:5]

 [,1] [,2] [,3] [,4] [,5]

[1,] 0.3337015 0.5398331 0.2778690 0.7163789 0.9626153

[2,] 0.8630726 0.5845770 0.7949684 0.4644632 0.9158006

[3,] NA NA NA NA NA

[4,] 0.4205321 0.2592077 0.3980515 0.2841568 0.8787681

> check <- colSums(p.values < alphaI, na.rm = TRUE)

> mean(check > 0)

[1] 0.11

The experimentwise type I error rate αE is estimated to be 0.11, higher than the individual type I error rate αI = 0.05. We can see that the LSD procedure does not have control of the experimentwise type I error rate.

For the Bonferroni procedure, we have (using the same data simulated above)

> Bon.pvalue< - function(yield, fertilizer) {

 data1 <- data.frame(yield, fertilizer)

 save.Bon <- pairwise.t.test(x = data1$yield, g = data1$fertilizer,

 p.adjust.method = "bonferroni",

 alternative = "two.sided")

 as.numeric(save.Bon$p.value)

 }

> p.values <- apply(X = y.ij, MARGIN = 1, FUN = Bon.pvalue, fertilizer =

 factor.levels)

> check <- colSums(p.values<alphaI, na.rm = TRUE)

> mean(check > 0)

[1] 0.035

The experimentwise type I error rate αE is estimated to be 0.035. The Bonferroni procedure was able to control αE less than or equal to the 0.05 level, as we would expect.

For the Tukey’s HSD procedure, we have

> HSD.pvalue <- function(yield, fertilizer) {

 data1 <- data.frame(yield, fertilizer)

 mod.fit <- aov(formula = yield ~ fertilizer, data = data1)

 save.Tukey <- TukeyHSD(x = mod.fit)

 as.numeric(save.Tukey$fertilizer[,4])

 }

> p.values <- apply(X = y.ij, MARGIN = 1, FUN = HSD.pvalue, fertilizer =

 factor.levels)

> check <- colSums(p.values<alphaI, na.rm = TRUE)

> mean(check > 0)

[1] 0.047

We use slightly different coding here. In order to use the TukeyHSD() function, we need to estimate an ANOVA model first. After we save the multiple comparison results in an object called save.Tukey, we can use names()to see that the following object save.Tukey$fertilizer contains all the p-values (the 4th column):

> save.Tukey$fertilizer

 diff lwr upr p adj

B-A 34.91895 -76.8858 146.72371 0.6702203

C-A 11.68956 -100.1152 123.49432 0.9543479

C-B -23.22940 -135.0342 88.57537 0.8338906

Therefore, the statement as.numeric(save.Tukey$fertilizer[,4]) will return all the p-values for each sample in our function.

The experimentwise type I error rate αE is estimated to be 0.047, very close to 0.05. The Tukey’s HSD procedure was also able to control αE well.

So far, we have only considered a very simple example with three factor levels. In practice, multiple comparison procedures are often used when there are *MANY* possible mean pairs. For example, in genetics data, people often want to perform t-tests for *thousands* of genes (one pair of means for each gene). Now, let’s look at another example with 20 factor levels and see how our experimentwise type I error rate αE changes.

> set.seed(2857)

> numb.data.sets <- 1000

> mu <- 500

> sd <- 50

> alphaI <- 0.05

> y.ij <- matrix(data = rnorm(n = 4\*20\*numb.data.sets, mean = mu, sd = sd), nrow =

 numb.data.sets, ncol = 4\*20)

> # R needs to know these are factor levels and not numbers

> factor.levels <- factor(rep(x = 1:20, each = 4))

> p.values <- apply(X = y.ij, MARGIN = 1, FUN = LSD.pvalue, fertilizer =

 factor.levels)

> check <- colSums(p.values < alphaI, na.rm = TRUE)

> mean(check > 0)

[1] 0.905

> p.values <- apply(X = y.ij, MARGIN = 1, FUN = Bon.pvalue, fertilizer =

 factor.levels)

> check < -colSums(p.values < alphaI, na.rm = TRUE)

> mean(check > 0)

[1] 0.033

> p.values <- apply(X = y.ij, MARGIN = 1, FUN = HSD.pvalue, fertilizer =

 factor.levels)

> check <- colSums(p.values < alphaI, na.rm = TRUE)

> mean(check > 0)

[1] 0.052

The results here are quite interesting. Due to the existence of many more factor levels, the experimentwise type I error rate for LSD is now severely inflated to 0.905! The other two procedures are still able to control αE well; but we start to see the clear drawback of the Bonferroni procedure: the actual αE tends to be lower than 0.05, suggesting the procedure is too conservative (rejects H0 at a rate less than what is stated). Generally, as the number of factor levels gets larger, one would expect the estimate for αE to get smaller. This conservativeness often leads to lower power (when there *is* a difference among the pair of means) than other procedures which do reject the null hypothesis at the specified level.

**Exploration**

1. Adjust the sample size (try higher and lower values). What do you find?
2. Increase the number of factor levels. How does this effect the error rate?