False-Positive Psychology - Simulation

Math 437 Spring 2023

2/6/2023

Table 1 of the "False-Positive Psychology" paper reports the results of simulations when four common decisions are made with respect to "which of these options produces statistical significance."

In this activity, we replicate 2 of the 4 simulations and attempt to verify the results, then partially replicate a third (we need a bit more background for Simulation C and part of D).

```
library(dplyr) # for easy mutate
```

Simulation A

For Simulation A, I have included all the chunks and they should automatically run. I have also indicated what each chunk is intended to do. You can use this as a template if you are struggling to code one of the simulations below.

Consider having two response variables Y_1 and Y_2 that could both be considered to represent the outcome of interest. Therefore, Y_1 and Y_2 are almost certainly positively correlated. Let's let $(Y_1, Y_2) \sim MVN(\vec{0}, \Sigma)$ where $\sigma_1 = \sigma_2 = 1$ and the correlation between the two variables is 0.5. We can use the `mvtnorm` package to simulate from this distribution:

```
library(mvtnorm)
mu0 <- c(0, 0)
Sigma0 <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)

set.seed(105051)
yvalues <- rmvnorm(40, mean = mu0, sigma = Sigma0)
sim1_values <- data.frame(
  condition = rep(c("Low", "High"), each = 20),
  y1 = yvalues[,1],
  y2 = yvalues[,2]
)
```

For each of the response variables Y_1 and Y_2 as well as the average $(Y_1 + Y_2)/2$ of the two response variables, we perform a two-sample t-test with $H_a : \mu_L \neq mu_H$ and report each p-value.

```
sim1_values <- sim1_values %>% mutate(
   y_avg = (y1+y2)/2
 )

t.test(y1 ~ condition, data = sim1_values)$p.value
```

```
## [1] 0.3879776
```

```
t.test(y2 ~ condition, data = sim1_values)$p.value
```

```
## [1] 0.7831837
```

```
t.test(y_avg ~ condition, data = sim1_values)$p.value
```

```
## [1] 0.5018186
```

Now, we will store the set of p-values in a matrix `pvaluesA` of size 15,000 x 3. The for loop in the chunk below runs 15,000 simulations and on each iteration `i` stores the p-values for the i^{th} set of two-sample t-tests in row i of the matrix.

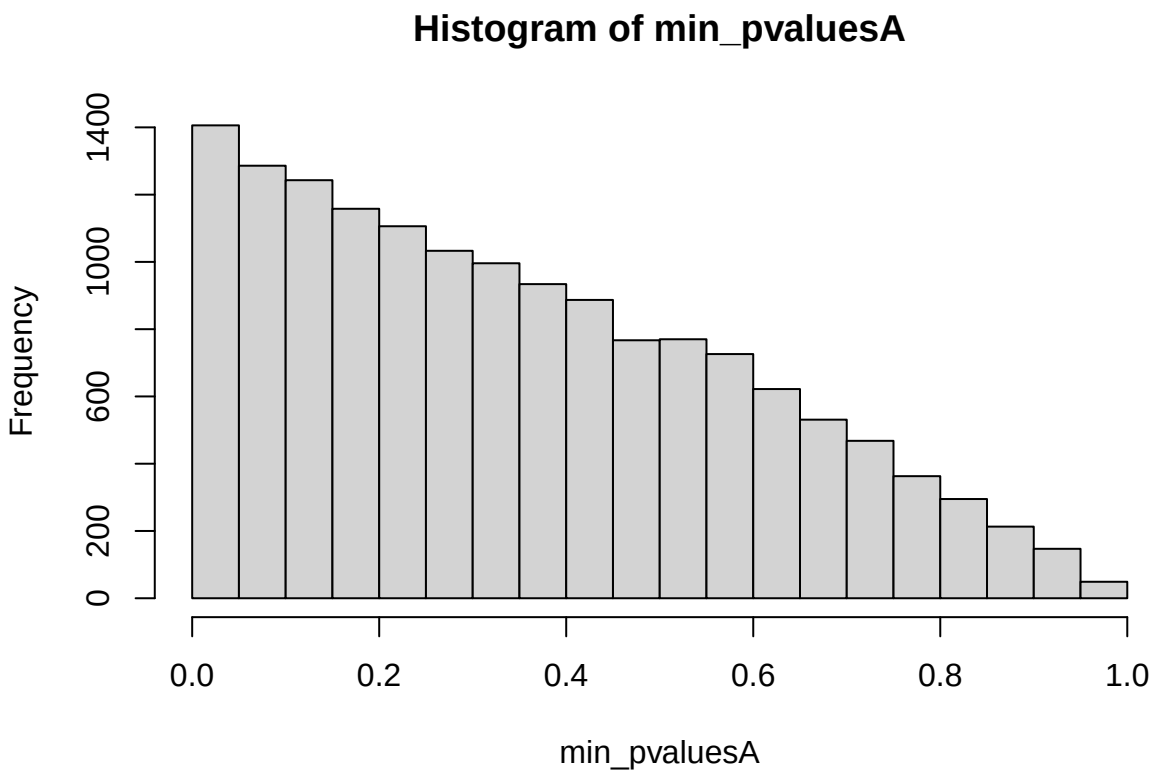
```
pvaluesA <- matrix(0, nrow = 15000, ncol = 3)

set.seed(105051)

for(i in 1:15000){

  yvalues <- rmvnorm(40, mean = mu0, sigma = Sigma0)
  sim1_values <- data.frame(
    condition = rep(c("Low", "High"), each = 20),
    y1 = yvalues[,1],
    y2 = yvalues[,2]
 ) %>% mutate(
   y_avg = (y1+y2)/2
 )

  pvaluesA[i, 1] <- t.test(y1 ~ condition, data = sim1_values)$p.value
  pvaluesA[i, 2] <- t.test(y2 ~ condition, data = sim1_values)$p.value
  pvaluesA[i, 3] <- t.test(y_avg ~ condition, data = sim1_values)$p.value

}
```

We next create a vector `min_pvaluesA` that will store obtain the minimum p-value for each iteration.

```
min_pvaluesA <- apply(pvaluesA, 1, min)
```

To investigate our simulation results, we look at a histogram of `min_pvaluesA` and estimate the true False Positive Rate (assuming a significance level of 5%).

```
hist(min_pvaluesA)
```




```
mean(min_pvaluesA <= 0.05) # FPR at alpha = 5%
```

```
## [1] 0.09373333
```

Notice that we obtain a false positive rate fair close to the 9.5% reported by the authors of the paper.

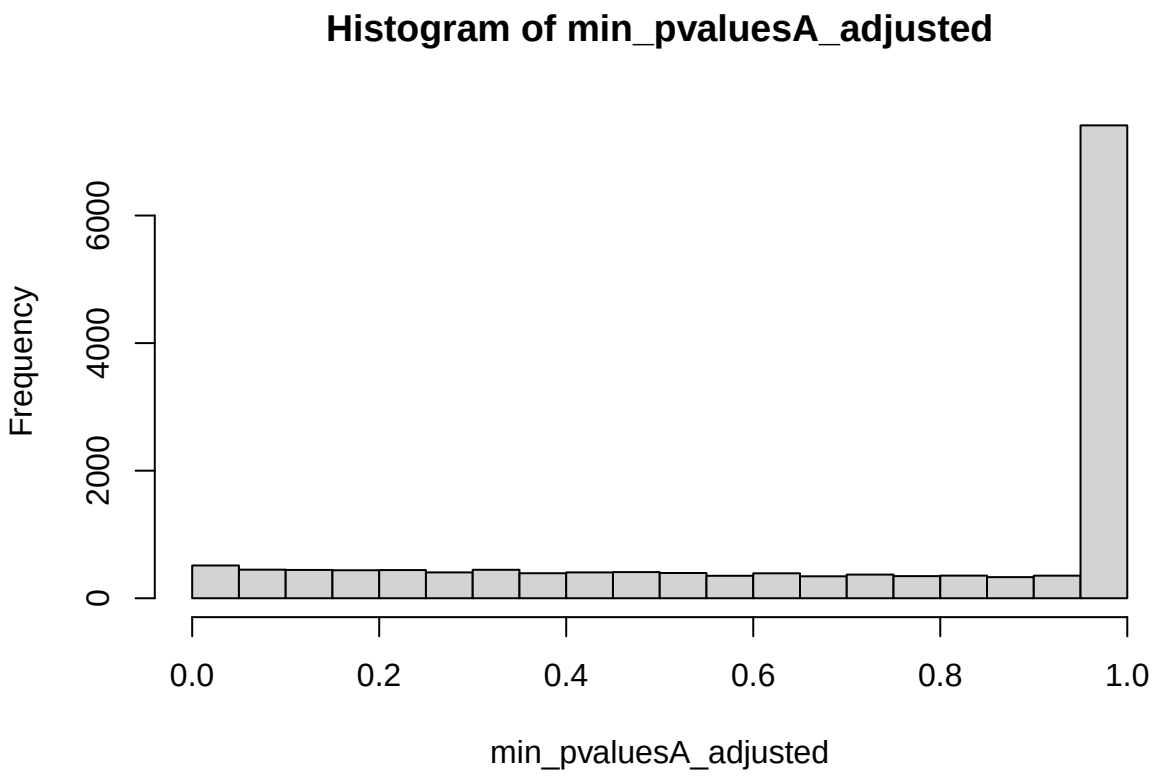
One possible solution to this "researcher degrees of freedom" is to do all possible tests being considered and then adjust the p-values to account for having done multiple hypothesis tests. Let's see if this actually works to reduce the false positive rate.

First, adjust the p-values using Holm's step-down procedure:

```
pvaluesA_adjusted <- t(apply(pvaluesA, 1, p.adjust, method = "holm"))
# the t is to transpose back to the original 15000 x 3 size
```

Create a vector `min_pvaluesA_adjusted` and obtain the minimum adjusted p-value for each iteration. Obtain a histogram of `min_pvaluesA_adjusted` and estimate the true False Positive Rate (assuming a significance level of 5%) when the researcher gets to decide which of the three linear combinations of the response variables are tested, but applies a correction afterwards.

```
min_pvaluesA_adjusted <- apply(pvaluesA_adjusted, 1, min)
hist(min_pvaluesA_adjusted)
```




```
mean(min_pvaluesA_adjusted <= 0.05) # FPR at alpha = 5%
```

```
## [1] 0.03426667
```

Did using Holm's step-down procedure help to appropriately control the False Positive Rate in this scenario?

Simulation B

Now consider the researcher testing for significance after collecting 20 observations per condition, but then deciding to go up to 30 observations per condition if significance is not achieved initially.

Randomly generate 40 values from $Y \sim N(0, 1)$ and randomly assign 20 of them to the Low condition and 20 of them to the High condition. Perform a two-sample t-test with $H_a : \mu_L \neq mu_H$ and report the p-value. If the p-value is not significant at 5%, generate another 20 values from Y and randomly assign 10 of them to each condition, (It may be easiest to use either `rbind` or `bind_rows` to combine the data.) then perform another two-sample t-test with the same H_a and report the new p-value.

Now, create a vector `pvaluesB` of length 15,000. Copy your code into a for loop that will run 15,000 simulations. On each iteration `i`, if the p-value for the first t-test (with 20 data points) is significant, store it in the i^{th} entry of the vector; otherwise, run the second t-test (with 60 data points) and store *that* p-value in the i^{th} entry of the vector.

Obtain a histogram of `pvaluesB` and estimate the true False Positive Rate (assuming a significance level of 5%) when the researcher decides whether to collect more data based on the initial test of significance. How close were we to the supposed 5% significance level? How close were we to the 7.7% reported by the authors?

Can these p-values be automatically adjusted? Why or why not?

Simulation D (Sort of)

Now suppose that there are *three* conditions, Low, Medium, and High.

Randomly generate 60 values from $Y \sim N(0, 1)$ and randomly assign 20 of them to each condition. Perform a two-sample t-test with $H_a : \mu_1 \neq mu_2$ for each of the three possible comparisons (HINT: look up the `subset` argument to `t.test`).

Now, create a matrix `pvaluesD` of size 15,000 x 3. Copy your code into a for loop that will run 15,000 simulations and on each iteration `i` store the p-values for the i^{th} set of tests in row i of the matrix.

Create a vector `min_pvaluesD` and obtain the minimum p-value for each iteration.

Obtain a histogram of `min_pvaluesD` and estimate the true False Positive Rate (assuming a significance level of 5%) when the researcher gets to decide which of the four ways to test the effect of condition.

Adjust the p-values using Holm's step-down procedure.

Create a vector `min_pvaluesD_adjusted` and obtain the minimum adjusted p-value for each iteration. Obtain a histogram of `min_pvaluesD_adjusted` and estimate the true False Positive Rate (assuming a significance level of 5%) when the researcher gets to decide which of the three group comparisons are tested, but applies a correction afterwards.

Did using Holm's step-down procedure help to appropriately control the False Positive Rate in this scenario?

Finally, create a vector `pvaluesD_anova` of length 15,000. Copy your earlier simulations, but instead of doing three t-tests on each iteration of the `for` loop, do a single one-way ANOVA (it is easiest to use `oneway.test` as the inputs and outputs are similar to `t.test`).

Obtain a histogram of the p-values and estimate the true False Positive Rate (assuming a significance level of 5%). Did we obtain a 5% False Positive Rate as expected?

What is the advantage of doing a one-way ANOVA followed by *post hoc* procedures, compared to doing a whole bunch of t-tests and then adjusting the resulting p-values?

Conclusion

In real study design and data analysis, we face many decision points where there is no single mathematically justified solution. Briefly explain the idea of "researcher degrees of freedom" and what your simulations suggest about the arbitrariness of these decisions.