We will now explore multiple hypothesis testing, or what happens when multiple tests are conducted on the same family of data.

We will set things up as before, with the false positive rate \(\alpha = 0.05\) and false negative rate \(\beta=0.20\).

library(pwr)

library(ggplot2)

set.seed(1)

mde <- 0.1 # minimum detectable effect

cr\_a <- 0.25 # the expected conversion rate for group A

alpha <- 0.05 # the false positive rate

power <- 0.80 # 1-false negative rate

ptpt <- pwr.2p.test(h = ES.h(p1 = cr\_a, p2 = (1+mde)\*cr\_a),

sig.level = alpha,

power = power

)

n\_obs <- ceiling(ptpt$n)

To illustrate the concepts in this article, we are going to use the same monte\_carlo utility function that we used previously:

#

# monte carlo runs n\_simulations and calls the callback function each time with the ... optional args

#

monte\_carlo <- function(n\_simulations, callback, ...){

simulations <- 1:n\_simulations

sapply(1:n\_simulations, function(x){

callback(...)

})

}

We’ll use the monte\_carlo utility function to run 1000 experiments, measuring whether the p.value is less than alpha **after n\_obs observations**. If it is, we reject the null hypothesis. We will set the effect size to 0; we know that there is no effect and that the null hypothesis is globally true. In this case, we expect about 50 rejections and about 950 non-rejections, since 50/1000 would represent our expected maximum false positive rate of 5%.

set.seed(1)

# make our "true" effect zero: the null hypothesis is always true

effect <- 0

cr\_b <- (1+effect)\*cr\_a

observations <- 2\*n\_obs

reject\_at\_i <- function(observations, i){

conversions\_a <- rbinom(observations, 1, cr\_a)

conversions\_b <- rbinom(observations, 1, cr\_b)

( prop.test(c(sum(conversions\_a[1:i]),sum(conversions\_b[1:i])), c(i,i))$p.value ) < alpha

}

# run the simulation

rejected.H0 <- monte\_carlo(1000,

callback=reject\_at\_i,

observations=n\_obs,

i=n\_obs

)

# output the rejection table

table(rejected.H0)

## rejected.H0

## FALSE TRUE

## 939 61

In practice, we don’t usually test the same thing 1000 times; instead, we test it once and state that there is a maximum 5% chance that we have falsely said there was an effect when there wasn’t one[1](https://rviews.rstudio.com/2019/10/02/multiple-hypothesis-testing/#fn1).

**The Family-Wise Error Rate (FWER)**

Now imagine we test two separate statistics using the same source data, with each test constrained by the same \(\alpha\) and \(\beta\) as before. What is the probability that we will detect *at least one* false positive considering the results of both tests? This is known as the family-wise error rate (FWER), and would apply to the case where a researcher claims there is a difference between the populations if any of the tests yields a positive result. It’s clear that this could present issues, as the [family-wise error rate](https://en.wikipedia.org/wiki/Family-wise_error_rate) Wikipedia page illustrates:

Suppose the treatment is a new way of teaching writing to students, and the control is the standard way of teaching writing. Students in the two groups can be compared in terms of grammar, spelling, organization, content, and so on. As more attributes are compared, it becomes increasingly likely that the treatment and control groups will appear to differ on at least one attribute due to random sampling error alone.

What is the FWER for the two tests? To calculate the probability that *at least one* false positive will arise in our two-test example, consider that the probability that one test will not reject the null is \(1-\alpha\). Thus, the probability that both tests will not reject the null is \((1-\alpha)^2)\) and the probability that *at least one* test will reject the null is \(1-(1-\alpha)^2\). For \(m\) tests, this generalizes to \(1-(1-\alpha)^m\). With \(\alpha=0.05\) and \(m=2\), we have:

m<-2

1-(1-alpha)^m

## [1] 0.0975

Let’s see if we can produce the same result with a Monte Carlo simulation. We will run the Monte Carlo for n\_trials and run n\_tests\_per\_trial. For each trial, if *at least one* of the n\_tests\_per\_trial results in a rejection of the null, we consider that the trial rejects the null. We should see that about 1 in 10 trials reject the null. This is implemented below:

set.seed(1)

n\_tests\_per\_trial <- 2

n\_trials <- 1000

rejects <- 0

for(i in 1:n\_trials){

# run the sim

rejected.H0 <- monte\_carlo(n\_tests\_per\_trial,

callback=reject\_at\_i,

observations=n\_obs,

i=n\_obs

)

if(![is.na](http://is.na)(table(rejected.H0)[2])) {

rejects <- rejects + 1

}

}

# Calculate FWER

rejects/n\_trials

## [1] 0.103

Both results show that evaluating two tests on the same family of data will lead to a ~10% chance that a researcher will claim a “significant” result if they look for either test to reject the null. Any claim there is a maximum 5% false positive rate would be mistaken. As an exercise, verify that doing the same on \(m=4\) tests will lead to an ~18% chance!

A bad testing platform would be one that claims a maximum 5% false positive rate when any one of multiple tests on the same family of data show significance at the 5% level. Clearly, if a researcher is going to claim that the FWER is no more than \(\alpha\), then they must control for the FWER and carefully consider how individual tests reject the null.

**Controlling the FWER**

There are many ways to control for the FWER, and the most conservative is the [Bonferroni correction](https://en.wikipedia.org/wiki/Bonferroni_correction). The “Bonferroni method” will reject null hypotheses if \(p\_i \le \frac{\alpha}{m}\). Let’s switch our reject\_at\_i function for a p\_value\_at\_i function, and then add in the Bonferroni correction:

set.seed(1)

p\_value\_at\_i <- function(observations, i){

conversions\_a <- rbinom(observations, 1, cr\_a)

conversions\_b <- rbinom(observations, 1, cr\_b)

prop.test(c(sum(conversions\_a[1:i]),sum(conversions\_b[1:i])), c(i,i))$p.value

}

n\_tests\_per\_trial <- 2

n\_trials <- 1000

rejects <- 0

for(i in 1:n\_trials){

# run n\_tests\_per\_trial

p\_values <- monte\_carlo(n\_tests\_per\_trial,

callback=p\_value\_at\_i,

observations=n\_obs,

i=n\_obs

)

# Bonferroni: adjust the p-values and reject any cases with p-values <= alpha

rejects <- rejects + sum(any(p\_values\*n\_tests\_per\_trial <= alpha))

}

# Calculate FWER

rejects/n\_trials

## [1] 0.055

With the Bonferroni correction, we see that the realized false positive rate is back near the 5% level. Note that we use any(...) to add 1 if any hypothesis is rejected.

Until now, we have only shown that the Bonferroni correction controls the FWER for the case that all null hypotheses are actually true: the effect is set to zero. This is called controlling in the *weak sense*. Next, let’s use R’s p.adjust function to illustrate the Bonferroni and [Holm](https://en.wikipedia.org/wiki/Holm%E2%80%93Bonferroni_method) adjustments to the p-values:

set.seed(1)

n\_tests\_per\_trial <- 2

n\_trials <- 1000

res\_bf <- c()

res\_holm <- c()

for(i in 1:n\_trials){

# run n\_tests\_per\_trial

p\_values <- monte\_carlo(n\_tests\_per\_trial,

callback=p\_value\_at\_i,

observations=n\_obs,

i=n\_obs

)

# Bonferroni: adjust the p-values and reject/accept

bf\_reject <- p.adjust(p\_values, "bonferroni") <= alpha

res\_bf <- c(res\_bf, sum(any(bf\_reject)))

# Holm: adjust the p-values and reject/accept

holm\_reject <- p.adjust(sort(p\_values), "holm") <= alpha

res\_holm <- c(res\_holm, sum(any(holm\_reject)))

}

# Calculate FWER

sum(res\_bf)/length(res\_bf)

## [1] 0.055

sum(res\_holm)/length(res\_holm)

## [1] 0.055

We see that the Holm correction is very similar to the Bonferroni correction in the case that the null hypothesis is always true.

**Strongly controlling the FWER**

Both the Bonferroni and Holm corrections guarantee that the FWER is controlled in the *strong sense*, in which we have any configuration of true and non-true null hypothesis. This is ideal, because in reality, we do not know if there is an effect or not.

The Holm correction is uniformly more powerful than the Bonferroni correction, meaning that in the case that there *is* an effect and the null is false, using the Holm correction will be more likely to detect positives.

Let’s test this by randomly setting the effect size to the minimum detectable effect in about half the cases. Note the slightly modified p\_value\_at\_i function as well as the null\_true variable, which will randomly decide if there is a minimum detectable effect size or not for that particular trial.

Note that in the below example, we will not calculate the FWER using the same any(...) construct from the previous code segments. If we were to do this, we would see that the both corrections have the same FWER *and* the same power (since the outcome of the trial is then decided by whether at least one of the hypotheses was rejected for the trial). Instead, we will tabulate the result for each of the hypotheses. We should see the same false positive rate[4](https://rviews.rstudio.com/2019/10/02/multiple-hypothesis-testing/#fn4), but great power for the Holm method.

set.seed(1)

p\_value\_at\_i <- function(observations, i, cr\_a, cr\_b){

conversions\_a <- rbinom(observations, 1, cr\_a)

conversions\_b <- rbinom(observations, 1, cr\_b)

prop.test(c(sum(conversions\_a[1:i]),sum(conversions\_b[1:i])), c(i,i))$p.value

}

n\_tests\_per\_trial <- 2

n\_trials <- 1000

res\_bf <- c()

res\_holm <- c()

for(i in 1:n\_trials){

null\_true <- rbinom(1,1,prob = 0.5)

effect <- mde \* null\_true

cr\_b <- (1+effect)\*cr\_a

# run n\_tests\_per\_trial

p\_values <- monte\_carlo(n\_tests\_per\_trial,

callback=p\_value\_at\_i,

observations=n\_obs,

i=n\_obs,

cr\_a=cr\_a,

cr\_b=cr\_b

)

# Bonferroni: adjust the p-values and reject/accept

reject\_bf <- p.adjust(p\_values, "bonferroni") <= alpha

for(r in reject\_bf){

res\_bf <- rbind(res\_bf, c(r, null\_true))

}

# Holm: adjust the p-values and reject/accept

reject\_holm <- p.adjust(sort(p\_values), "holm") <= alpha

for(r in reject\_holm){

res\_holm <- rbind(res\_holm, c(r, null\_true))

}

}

# the rows of the table represent the test result

# while the columns represent the null truth

table\_bf <- table(Test=res\_bf[,1], Null=res\_bf[,2])

table\_holm <- table(Test=res\_holm[,1], Null=res\_holm[,2])

# False positive rate

fpr\_bf <- table\_bf['1','0']/sum(table\_bf[,'0'])

fpr\_holm <- table\_holm['1','0']/sum(table\_holm[,'0'])

print(paste0("FPR Bonferroni: ", round(fpr\_bf,3), " FPR Holm: ", round(fpr\_holm,3)))

## [1] "FPR Bonferroni: 0.029 FPR Holm: 0.029"

# Power

power\_bf <- table\_bf['1','1']/sum(table\_bf[,'1'])

power\_holm <- table\_holm['1','1']/sum(table\_holm[,'1'])

print(paste0("Power Bonferroni: ", round(power\_bf,3), " Power Holm: ", round(power\_holm,3)))

## [1] "Power Bonferroni: 0.713 Power Holm: 0.774"

# Comparing the Power of Holm vs. Bonferroni

print(paste0("Power Holm/Power Bonferroni: ", round(power\_holm/power\_bf,3)))

## [1] "Power Holm/Power Bonferroni: 1.084"

Indeed, we observe that while the realized false positive rates of both the Bonferroni and Holm methods are very similar, the Holm method has greater power. These corrections essentially reduce our threshold for each test so that across the family of tests, we produce false positives with a probability of no more than \(\alpha\). This comes at the expense of a reduction in power from the optimal power (\(1-\beta\)).

We have illustrated two methods for deciding what null hypotheses in a family of tests to reject. The Bonferroni method rejects hypotheses at the \(\alpha/m\) level. The Holm method has a more involved algorithm for which hypotheses to reject. The Bonferroni and Holm methods have the property that they *do* control the FWER at \(\alpha\), and Holm is uniformly more powerful than Bonferroni.

This raises an interesting question: What if we are not concerned about controlling the probability of detecting at least one false positive, but something else? We might be more interested in controlling the expected proportion of false discoveries amongst all discoveries, known as the false discovery rate. As a quick preview, let’s calculate the false discovery rate for our two cases:

table\_holm

## Null

## Test 0 1

## 0 959 229

## 1 29 783

table\_bf

## Null

## Test 0 1

## 0 959 290

## 1 29 722

fdr\_holm <- table\_holm['1','0']/sum(table\_holm['1',])

fdr\_holm

## [1] 0.03571

fdr\_bf <- table\_bf['1','0']/sum(table\_bf['1',])

fdr\_bf

## [1] 0.03862

By choosing to control for a metric other than the FWER, we may be able to produce results with power closer to the optimal power (\(1-\beta\)). We will look at the false discovery rate and other measures in a future article.