In this post, we will develop a framework for always-valid inference based on the paper [Always Valid Inference: Continuous Monitoring of A/B Tests](https://arxiv.org/pdf/1512.04922.pdf) (2019 Johari, Pekelis, Walsh). Using an always-valid p-value allows us to continuously monitor A/B tests, and potentially stop the test early in a valid way[1](https://rviews.rstudio.com/2019/08/22/calculating-always-valid-p-values-in-r/#fn1).

In section 5 of the paper, the authors propose their method for calculating always-valid p-values: the mixture sequential ratio probability test (mSPRT), first introduced by Robbins (1970). To keep this post brief, we will not do the paper’s theoretical foundations justice. Instead, we will focus on the most important equations, which we will use to produce always valid p-values in our R code. For those uninterested in the math, feel free to skip ahead.

The paper makes some basic assumptions about the data and its functional form: it is real valued and drawn from a single-parameter exponential family where tests are of the parameter \(\theta\). The mSPRT is parameterized by mixing distribution \(H\) over \(\Theta\), an open interval that contains all \(\theta\). Given an observed sample average \(s\_n\) at time \(n\), the mixture likelihood ratio of \(\theta\) against \(\theta\_0\) with respect to H is defined as:

\(\Lambda\_{n}^{H}(s\_{n}) = \displaystyle\int\_{\Theta}\bigg(\frac{f\_{\theta}(s\_{n})}{f\_{\theta\_{0}}(s\_{n})}\bigg)^{n} dH(\theta)\)

We are told to calculate the p-values as:

\(p\_{0} = 1; p\_{n} = min\{p\_{n-1}, 1/\Lambda\_{n}^{H}\}\)

So, if we have a mixing distribution \(H(\theta)\) and iterate over it, we can calculate \(\Lambda\_{n}^{H}(s\_{n})\) for the sample average we have observed at time \(n\). If \(\Lambda\_{n}^{H}\) is ever greater than \(\alpha\), our tolerance for the false positive rate \(p\_n\) will exceed \(alpha\), and we can stop the test and reject the null hypothesis.

We will implement this for a simple comparison of two binomial distributions first, where some convenient results make \(p\_{n}\) easier to calculate. In section 6.1, the paper describes the deployment for two-stream p-values. In this case, we are looking at two streams of *Bernoulli* data: two streams of ones and zeros describing whether we observed a conversion or an abandonment in each stream.

The [central limit theorem](https://en.wikipedia.org/wiki/Central_limit_theorem) tells us that the distribution of the average number of conversions will be approximately normal after a large number of observations (say, more than 100). What we want to know is: is there a statistically significant difference between the measured averages of the two streams? Put another way: is the difference between the measured averages of the two streams significantly different from zero? It is this second formulation that we will implement.

We will skip the derivation for normal data and say that if we have two streams of Bernoulli data, \(A\_n\) and \(B\_n\), that each yield approximately normal average conversion rates \(\mu\_{A,n}\) and \(\mu\_{B,n}\) after \(n\) observations, we can take \(f\_\theta(s\_n) \approx N(\theta, \sigma^2)\) and use a normal mixing distribution \(H=N(0,\tau^2)\) in \(\Lambda\_n^H\) to decide if \(\theta\_n\) is significantly different from zero. Conveniently, this mixing distribution yields a closed-form formula for \(\Lambda\_n^H\):

\(\Lambda\_n^H = \sqrt{\frac{\sigma\_n^2}{\sigma\_n^2+n\tau^2}} exp\bigg\{\frac{n^2\tau^2(\theta\_n)^2}{2\sigma\_n^2(\sigma\_n^2+n\tau^2)}\bigg\}\)

Let’s look at the variables in the above equation. We can calculate \(\theta\_n = \mu\_{A,n}-\mu\_{B,n}\) from our streams of Bernoulli data by calculating the average conversion rates at \(n\) for each stream. We can also use the fact that the variance \(\nu\) of a binomial distribution with mean \(\mu\) is \(\nu = \mu(1-\mu)\), and that the variance of the sum of two random binomials is the sum of their variances:

\(\nu\_{AB} = \nu\_A + \nu\_B = \mu\_{A,n}(1-\mu\_{A,n}) + \mu\_{B,n}(1-\mu\_{B,n}) = \sigma\_n^2\).

One lingering issue is: how do we define \(\tau^2\)? The answer is that we can choose any \(\tau^2\) we would like, though experience will show it is best if \(\tau^2\) is on the order of \(\sigma^2\). Later we will explore what different \(\tau^2\) values yield, and how there is an optimal \(\tau^2\) to choose.

Let’s convert the above into code:

# First we define the function that calculates lambda

# given n, the means at n of A and B, and tau\_squared

lambda <- function(n, mu\_a\_n, mu\_b\_n, tau\_sq){

v\_n <- (mu\_a\_n\*(1-mu\_a\_n) + mu\_b\_n\*(1-mu\_b\_n))

nts <- n\*tau\_sq

if(v\_n == 0){

return(1.0)

}

else {

return(

sqrt((v\_n)/(v\_n+nts))\*

exp(

((n\*nts)\*(mu\_a\_n-mu\_b\_n)^2)/

((2.0\*v\_n)\*(v\_n+nts))

)

)

}

}

# Next we will calculate the always valid p-values at each n

calc\_avpvs <- function(n\_obs, cr\_a\_obs, cr\_b\_obs, tau\_sq = 0.1){

p\_n <- rep(1.0,n\_obs)

for (i in 2:n\_obs) {

mu\_a\_n <- mean(cr\_a\_obs[1:i])

mu\_b\_n <- mean(cr\_b\_obs[1:i])

p\_n[[i]] <- min(p\_n[[i-1]],1/lambda(i,mu\_a\_n, mu\_b\_n, tau\_sq))

}

p\_n

}

Now, let’s create a test example, similar to our previous post, and set an effect size that is twice the minimum detectable effect. In this case, we should expect that after having observed the “correct” number of observations given by the power calculation, there is a very high probability that we’ll see an effect.

library(pwr)

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.5.2

set.seed(5)

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)

# make our "true" effect 1.5x larger than the mde

# this should yield a conclusive test result

effect <- 1.5\*mde

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

observations <- 2\*n\_obs

# two streams of {0,1} conversions

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

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

# now we'll calculate the always-valid p-values

avpvs <- calc\_avpvs(observations, conversions\_a, conversions\_b)

# And we'll calculate "regular" p-values as well

tt <- sapply(10:observations, function(x){

t.test(conversions\_a[1:x],conversions\_b[1:x])$p.value

})

tt <- data.frame(p.value = unlist(tt))

# for plots

conf\_95 <- data.frame( x = c(-Inf, Inf), y = 0.95 )

obs\_limit\_line <- data.frame( x = n\_obs, y = c(-Inf, Inf) )

# plot the evolution of p-values and always-valid p-values

ggplot(tt, aes(x=seq\_along(p.value), y=1-p.value)) +

geom\_line() +

geom\_line(aes(x, y, color="alpha=5%"), linetype=3, conf\_95) +

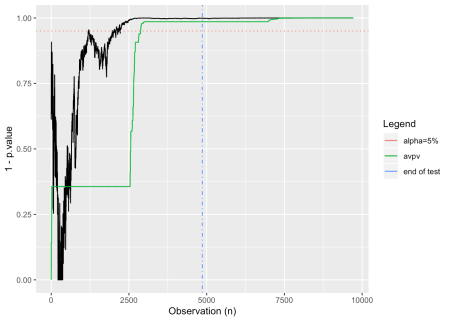
geom\_line(aes(x, y, color="end of test"), linetype=4, obs\_limit\_line) +

geom\_line(data=data.frame(x=seq(1:observations),y=1-avpvs), aes(x=x,y=y, color="avpv")) +

xlab("Observation (n)") +

scale\_color\_discrete(name = "Legend") +

ylim(c(0,1))



In this case, we see that by using the always valid p-value, we would be able to terminate early while still controlling the false positive rate at \(\alpha\). Let’s look at what happens when there is no effect:

effect <- 0

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

observations <- 2\*n\_obs

# two streams of {0,1} conversions

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

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

# now we'll calculate the always-valid p-values

avpvs <- calc\_avpvs(observations, conversions\_a, conversions\_b)

# And we'll calculate "regular" p-values as well

tt <- sapply(10:observations, function(x){

t.test(conversions\_a[1:x],conversions\_b[1:x])$p.value

})

tt <- data.frame(p.value = unlist(tt))

# for plots

conf\_95 <- data.frame( x = c(-Inf, Inf), y = 0.95 )

obs\_limit\_line <- data.frame( x = n\_obs, y = c(-Inf, Inf) )

# plot the evolution of p-values and always-valid p-values

ggplot(tt, aes(x=seq\_along(p.value), y=1-p.value)) +

geom\_line() +

geom\_line(aes(x, y, color="alpha=5%"), linetype=3, conf\_95) +

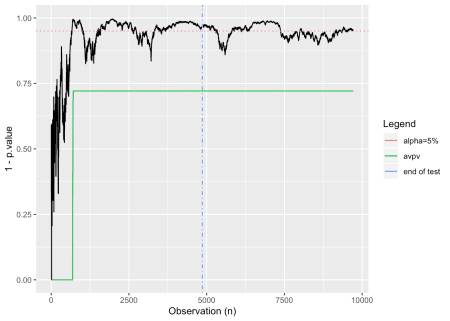
geom\_line(aes(x, y, color="end of test"), linetype=4, obs\_limit\_line) +

geom\_line(data=data.frame(x=seq(1:observations),y=1-avpvs), aes(x=x,y=y, color="avpv")) +

xlab("Observation (n)") +

scale\_color\_discrete(name = "Legend") +

ylim(c(0,1))



In this case, we see a test in which the p-values oscillate near alpha. A “peeker” looking at this might have incorrectly called it a winner early on, however our always-valid p-values maintain the correct outcome.

Now set things up to have an effect 1.5 times the size of the minimum detectable effect and see what effect varying \(\tau^2\) has. We’ll choose five different values at different orders of magnitude between 0.0001 and 1:

library(pwr)

library(ggplot2)

set.seed(5)

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)

n\_obs

## [1] 4860

# make our "true" effect 1.5x larger than the mde

# this should yield a conclusive test result

effect <- 1.5\*mde

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

observations <- 2\*n\_obs

# two streams of {0,1} conversions

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

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

# now we'll calculate the always-valid p-values

avpvs\_tsa <- calc\_avpvs(observations, conversions\_a, conversions\_b, tau\_sq = 0.0001)

avpvs\_tsb <- calc\_avpvs(observations, conversions\_a, conversions\_b, tau\_sq = 0.001)

avpvs\_tsc <- calc\_avpvs(observations, conversions\_a, conversions\_b, tau\_sq = 0.01)

avpvs\_tsd <- calc\_avpvs(observations, conversions\_a, conversions\_b, tau\_sq = 0.1)

avpvs\_tse <- calc\_avpvs(observations, conversions\_a, conversions\_b, tau\_sq = 1)

# And we'll calculate "regular" p-values as well

tt <- sapply(10:observations, function(x){

t.test(conversions\_a[1:x],conversions\_b[1:x])$p.value

})

tt <- data.frame(p.value = unlist(tt))

# for plots

conf\_95 <- data.frame( x = c(-Inf, Inf), y = 0.95 )

obs\_limit\_line <- data.frame( x = n\_obs, y = c(-Inf, Inf) )

# plot the evolution of p-values and always-valid p-values

ggplot(tt, aes(x=seq\_along(p.value), y=1-p.value)) +

geom\_line() +

geom\_line(aes(x, y, color="alpha=5%"), linetype=3, conf\_95) +

geom\_line(aes(x, y, color="end of test"), linetype=4, obs\_limit\_line) +

geom\_line(data=data.frame(x=seq(1:observations),y=1-avpvs\_tsa), aes(x=x,y=y, color="avpv\_0.0001")) +

geom\_line(data=data.frame(x=seq(1:observations),y=1-avpvs\_tsb), aes(x=x,y=y, color="avpv\_0.001")) +

geom\_line(data=data.frame(x=seq(1:observations),y=1-avpvs\_tsc), aes(x=x,y=y, color="avpv\_0.01")) +

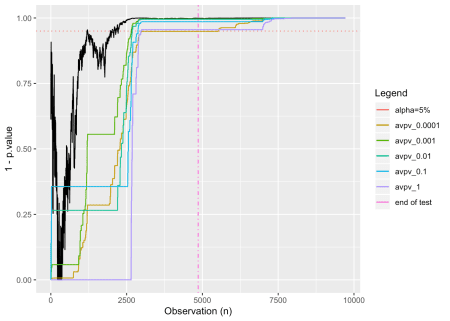
geom\_line(data=data.frame(x=seq(1:observations),y=1-avpvs\_tsd), aes(x=x,y=y, color="avpv\_0.1")) +

geom\_line(data=data.frame(x=seq(1:observations),y=1-avpvs\_tse), aes(x=x,y=y, color="avpv\_1")) +

xlab("Observation (n)") +

scale\_color\_discrete(name = "Legend") +

ylim(c(0,1))



It looks like we might be able to stop early if we use anything but the lowest and highest values for \(\tau^2\).

Using always-valid p-values, particularly for small- or zero-sized effects, can be helpful in avoiding false positives. I’ve glossed over some important details that should be considered when working with non-normal data, but I hope this provides an introduction to an alternative way of evaluating binomial A/B tests. See [Always Valid Inference: Continuous Monitoring of A/B Tests](https://arxiv.org/pdf/1512.04922.pdf) for more details.