Uber circuit comparison via Confidence Distribution

Krishnan Raman

11/6/2020

Introduction

A "p-value function" is a special case of a rather useful Fiducial construct devised by Fisher, so-called "Confidence Distribution" aka CD. We use a CD below to tell apart two circuits.

Dr. Tirthankar Dasgupta documents the rise and fall of CD over the years. Initially, CDs fell out of favor since i) a CD conducts a Fisher Sharp Null hypothesis test in a ii) finite sampling context using iii) repeated draws from an assignment distribution. CDs were thus faulted on all 3 counts. On the first count, statisticians were much more interested in the Neyman null, which compares sample means $\bar{y}(1)\ vs\ \bar{y}(0)$ across two populations, whereas Fisher Sharp Null compares individual potential outcomes $Y_i(1)\ vs\ Y_i(0)$ for every observation over a finite sub-population. On the second count, finite sampling is inherently limiting as it is confined to the given small countable sub-population, versus sampling from a potentially infinite super-population. Finally, repeated draws from an assignment distribution were computationally expensive. However, for the very same reasons, Dr. Dasgupta portends a massive resurgence in CD! In this age of precision medicine, one is much more interested in comparing outcomes on an individual level; population averages hold much less appeal. Computational resources are no longer an impediment as social networks routinely handle data in the peta-bytes. Repeatedly sampling from a large finite distribution with combinatorial choice assignments is no longer a show stopper, as Data Scientists in social media companies use tools such as Scalding & Spark to perform this exact task every single day in order to build Who-To_Follow recommender lists for billion-plus users.

An Example

CDs are extremely intuitive and work for a broad class of assignment mechanisms employed everyday by social networks. For example, a network such as Twitter with 400 million users performs over tens of thousands of statistical experiments everyday. In each of these, million-bucket users are allotted to a treatment group vs a million-bucket control. The treatment is a typical UI feature that the Twitter engineers are iterating upon. Rather than double down & build the feature to completion (so-called Waterfall model), most tech giants prefer to take a half-baked feature and assign it to the treatment group of over a million (unsuspecting) users. Only if a significant number of users in the treatment group warm to the feature is a shipit(a go-ahead) given; there is no incentive in shipping features the audience doesn't care about. This model of iterative feature-building combined with experimental design and statistical testing leading to gated shipits naturally lends itself to CD. Using a CD, one obtains a precise "p-value function" that describes, within a finite sub-population, whether the user on an individual level liked the feature or not, aka the Fisher Null. This is precisely what Twitter looks for; a feature that works very well for the niche Japanese Manga sub-population might fail spectacularly for the conservative Wall Street crowd. In the absence of a CD, the average feature response aka Neyman null will most likely lead to a failed experiment and shuttered feature; whereas with a CD, once can identify sub-populations where the feature is a hit & shipit selective features to stratified clusters, boosting revenue for the corp whilst simultaneously increasing user satisfaction; these are not marginal achievements; a 0.1% improvement in revenue or user count translates to a literal increase in a few hundred million dollars in revenue per quarter.

Problem Description

A loop traversed by an Uber driver is termed a k-circuit. A 3-circuit is then denoted by a moniker such as abca, which breaks down into three edges ab, bc and ca, which completes the loop. The Uber driver leaves city block a, where he presumably lives, drives to city block b, then b to c, and then back home from c to a. After analyzing over 30 million Uber trips in the 2014-15 public dataset provided upon FOIA request by Uber to the NYC Transportation Commission, we have determined that a lucrative edge yields a median fare of fifty dollars. Hence, an Uber driver operating a 3-circuit can be expected to take home $150 per day. How best to distinguish among two competing 3-circuits ?

Methodology

To recap, a 3-circuit has 3 edges to form a closed loop. Each edge in a circuit comprises finite samples from a fare distribution. Thus, a 3-circuit take-home is the sum of 3 fares. In the Toy problem below are simulations for two distinct 3-circuits over four weeks (28 samples per circuit) (In reality, we care about distinguishing between k-circuits with k > 10, over a month)

3-Circuit: Circuit C_1: Let ab,bc,ca comprise the first circuit. C_1 = a->b->c->a.
Circuit C_2 = e->f->g->e

Fare Distributions: $ab \sim N(50,1), bc \sim N(51,2), ca \sim N(52,3)$
$ef \sim N(50,2), fg \sim N(52,3), ge \sim N(53,1)$

The circuits are shown below:

```
n<-28 # 4 weeks
options(digits=4)
ab <- rnorm(n,50,1)
bc <- rnorm(n,51,2)
ca <- rnorm(n,52,3)
ef <- rnorm(n,50,2)
fg <- rnorm(n,52,3)
ge <- rnorm(n,53,1)
c1<- ab+bc+ca
c2<- ef+fg+ge
df<-data.frame(ab=ab,bc=bc,ca=ca,ef=ef,fg=fg,ge=ge,Ckt_c1=c1,Ckt_c2=c2)
head(df,n)
```

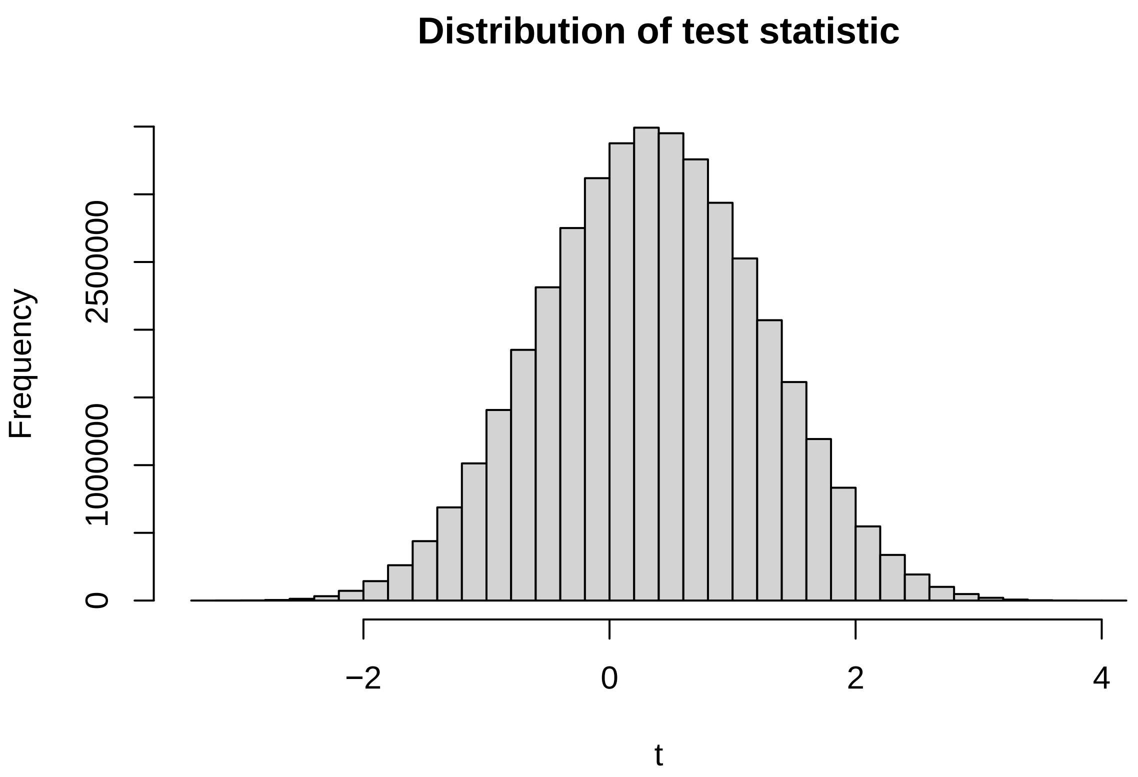
```
##       ab    bc    ca    ef    fg    ge Ckt_c1 Ckt_c2
## 1  50.90 50.59 57.81 48.79 52.43 53.12  159.3  154.3
## 2  51.52 52.44 52.72 47.85 51.77 52.65  156.7  152.3
## 3  49.69 49.91 52.19 50.82 50.38 52.10  151.8  153.3
## 4  50.24 51.35 53.68 49.08 50.32 52.13  155.3  151.5
## 5  51.46 47.43 51.57 48.74 47.39 52.93  150.5  149.1
## 6  51.02 47.16 55.97 46.95 55.61 52.04  154.1  154.6
## 7  49.09 53.27 54.99 50.93 50.80 52.49  157.3  154.2
## 8  49.30 51.97 55.95 48.19 52.93 52.54  157.2  153.7
## 9  49.05 49.70 52.96 49.10 53.77 52.82  151.7  155.7
## 10 48.24 51.30 47.90 50.17 48.18 53.26  147.4  151.6
## 11 48.91 52.98 50.69 52.06 50.51 51.54  152.6  154.1
## 12 50.86 52.53 57.24 54.41 50.76 51.83  160.6  157.0
## 13 48.22 52.44 53.66 50.05 50.48 52.29  154.3  152.8
## 14 50.93 51.95 52.23 50.33 49.17 54.93  155.1  154.4
## 15 49.24 51.31 55.59 55.05 51.57 53.17  156.1  159.8
## 16 50.70 51.73 50.07 49.00 49.77 52.70  152.5  151.5
## 17 49.83 52.73 52.86 49.24 51.87 52.82  155.4  153.9
## 18 49.98 50.50 54.87 51.08 52.29 51.87  155.3  155.2
## 19 49.08 48.63 50.36 51.20 51.10 53.55  148.1  155.8
## 20 50.00 51.76 48.61 49.46 48.91 52.55  150.4  150.9
## 21 50.64 47.25 48.85 54.08 55.04 52.56  146.7  161.7
## 22 50.84 47.88 51.16 49.35 51.71 53.84  149.9  154.9
## 23 50.85 50.08 51.64 49.21 50.86 52.61  152.6  152.7
## 24 49.93 51.65 50.05 47.89 47.73 53.54  151.6  149.2
## 25 51.28 51.13 48.94 50.68 52.66 53.42  151.4  156.8
## 26 50.58 53.25 48.11 51.48 47.88 53.18  151.9  152.5
## 27 51.31 52.08 54.99 47.92 50.67 54.07  158.4  152.7
## 28 50.67 54.41 53.43 50.15 52.65 53.55  158.5  156.4
```

Assignments

There are 28 Choose 14 = 40,116,600 possible assignments. An assignment is a boolean vector over two weeks. An assignment simply means on the given day, the Uber driver drove Circuit c2 and obtained wages given in the Ckt_c2 column for that day. A non-assignment means the wages would come from Ckt_c1, as Driver driving Circuit c1 is the Null Hypothesis. For each sample in the assignment distribution, we assign to Y_{obs} the wages from Ckt_c2, and for every non-assignment, the wages from Ckt_c1. Compute the biweekly average and compare with the null hypothesis biweekly average i.e. as if the driver had driven only on Ckt_c1 throughout the 2 weeks. This comparison is a simple difference test statistic.

```r
outersect <- function(x, y) {
  sort(c(x[!x%in%y],
         y[!y%in%x]))
}

 ".combinadic" <- function(n, r, i) {
     # http://msdn.microsoft.com/en-us/library/aa289166(VS.71).aspx
     # http://en.wikipedia.org/wiki/Combinadic

     if(i < 1 | i > choose(n,r)) stop("'i' must be 0 < i <= n!/(n-r)!")

     largestV <- function(n, r, i) {
         #v <- n-1
         v <- n                                  # Adjusted for one-based indexing
         #while(choose(v,r) > i) v <- v-1
         while(choose(v,r) >= i) v <- v-1        # Adjusted for one-based indexing
         return(v)
     }

     res <- rep(NA,r)
     for(j in 1:r) {
         res[j] <- largestV(n,r,i)
         i <- i-choose(res[j],r)
         n <- res[j]
         r <- r-1
     }
     res <- res + 1
```

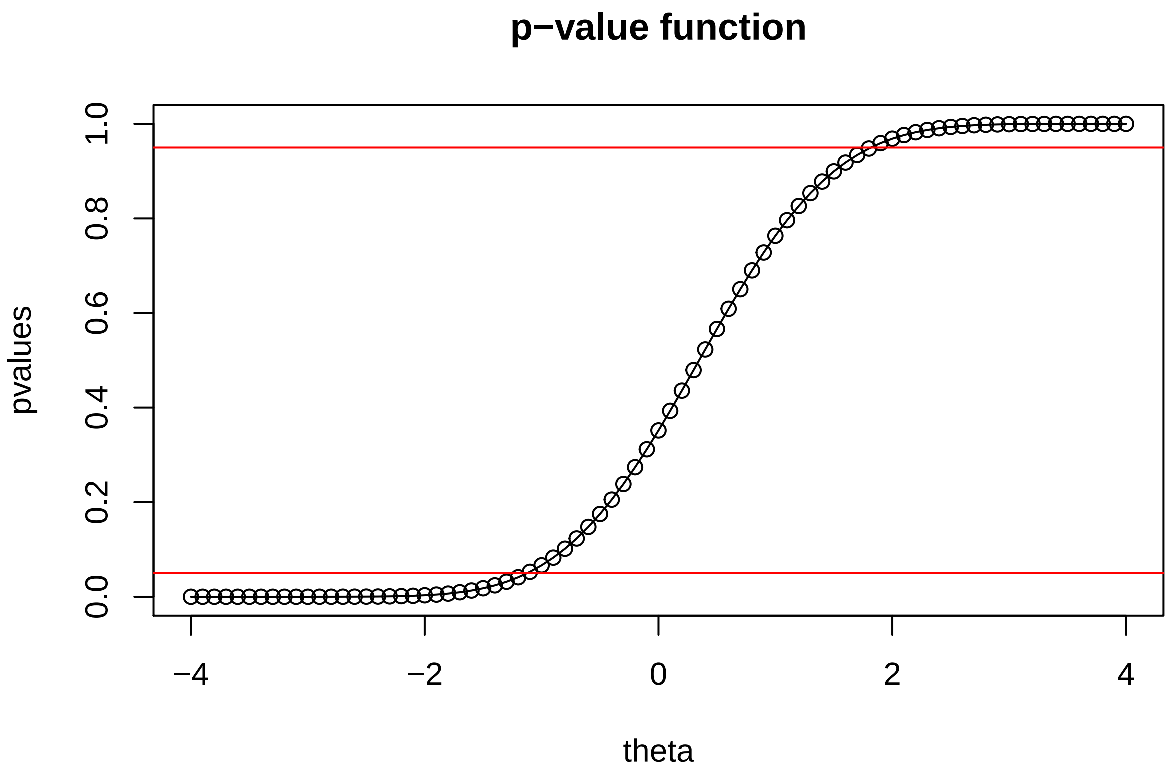
```r
    return(res)
}

r<-n/2
#assignments <- combn(n,r)

len <- choose(n,r)
t <- numeric(len)
for(i in 1:len) {
  #assignment<- assignments[,i]
  assignment<-.combinadic(n,r,i)
  y_obs <- c1
  y_obs[assignment] <- c2[assignment]
  not_assigned <- outersect(1:n, assignment)
  # want Ybar(1) - Ybar(0) on the observed
  diff <- mean(y_obs[assignment]) - mean(y_obs[not_assigned])
  t[i] <- diff
}

hist(t, main="Distribution of test statistic")
```



CD aka p-value function

For each theta, we find the proportion of assignments for which the test statistic does not exceed theta. A plot of p-value versus theta is the p-value function we seek, shown below with 95% Fisherian Interval (NOT Confidence Interval) in red.

```
m<- round(max(abs(min(t)), abs(max(t))))
theta <- seq(-m,m,0.1)
nt <- length(t)
pvalues<-c()
for(mytheta in theta) {
  pval<- length(t[t < mytheta])/nt
  pvalues<-c(pvalues, pval)
}
plot(theta,pvalues, main="p-value function", 'p')
lines(theta,pvalues, main="p-value function", 'l')
abline(h=0.05,col='red')
abline(h=0.95,col='red')
```



The real-life analog:

The median Uber driver makes 10 trips daily. We seek an optimal 10-circuit, i.e. a k-circuit with k=10 which maximizes wages over all possible 10-circuits. Such a k-circuit can be modeled by a Bayesian prior of Gaussian with mean \$500 (since the median lucrative edge fetches \$50). Uber provides fare data over 30 possible locations in NYC, which give us 435 fare distributions for each possible edge between two locations. Of these, only ~20 edges fall in the lucrative \$50 median wage. Constructing a 10-circuit from a choice of 20 edges can be done in 20Choose 9 = about 168,000 ways. For each of these circuits, from the Uber data, we obtain real fares over a given month. That gives us 30 samples per circuit. Using the technique outlined above, we can construct 30C15 = 155 million assignments per circuit, and build p-value functions to tell apart any two circuits. With some clever filtering, it is possible to pick the most lucrative 10-circuit the Uber driver must ply in order to maximize his wages.

Conclusion

Professor Don Rubin aptly summarizes the Fisher Randomization Test as a “stochastic proof by contradiction”. In the present age of democratized Data Science, where executives deal with counter-factual outcomes routinely, the language of FRT becomes a natural fit to scientists seeking to test hypotheses among large finite samples that don’t come from any specific distribution. Most socio-economic data such as fares, retweets, likes etc. tend to belong to fat-tailed multi-modal distributions. Rather than sacrifice accuracy by picking a specific parametric form prior to hypothesis test, FRT deals with data as-is. Sampling from (large) finite datasets to produce a confidence-distribution automates inference across a whole class of survey sampling problems. The results are intuitive to a non-statistical community. The procedure lends itself naturally to automation across data pipelines in social media companies. There do remain challenges on the theoretical front; not every test statistic, in particular normalized t-type statistic, can be employed to build a CD. Since we require a monotonically increasing function, Dr. Dasgupta has demonstrated a sufficiency condition that the test statistic must belong to a family of so-called “Effect Increasing” functions. Moreover, he has devised an upper bound for the number of Monte Carlo samples required to estimate an entire p-value function for a given tolerance. Going forward, we believe Confidence Distributions should see wide usage in disparate domains such as finance, social media & the pharma community.