Summer 2016

Black Cloud Randomization Test

Nicholas S. Vanni
nsv9@zips.uakron.edu

Honors Research Project

Black Cloud Randomization Test

Nicholas Vanni

Summer 2016

Abrstract

The Black Cloud Randomization Test looks at a nontraditional question and attempts to answer the question using unique statistics. The purpose of this paper is to apply what has been learned throughout the years and apply this knowledge to a final project. Data for this project follows an emergency room's on call schedule, as well as the number of traumas that came in during each day shift. The project builds on what has been already learned and helps to open a different way of working with statistics. The project was coded in the R software. With different restrictions, there are attempts to balance the strict definition of a permutation test with the practicality of the dataset involved. Looking at a rather large dataset for a traditional permutation test, the randomization test still nears the strength of the original test. Discussion is open to the accuracy and validity of the modified test.

Introduction

This project looks at a history of an emergency room's residents on call and the number of large traumas that come in on a given day. The idea is to look at each resident on call and determine if their presence on a given day will result in an increased or decreased number of traumas on average. The residents that have statistical significance associated with having more traumas than average are labeled as "black clouds". On the other side of this, residents that have statistical significance associated with having fewer traumas than average are labeled as "white clouds". This could be applicable in an emergency room setting to try optimizing what residents are on call. A randomization test was carried out to determine significance. The randomization test follows the idea behind a permutation test, but is limited by the large number of possible permutations that can be made using the dataset at hand. Instead, a smaller, but more reasonable sample size will be used. In each sample, the number of days with traumas will be randomized, while the schedule for each resident will remain the same. Each randomization will then be matched to the original schedule. The analysis was coded using the R software without the use of any packages.

Explanation of the Data

The dataset used for analysis consists of an emergency room's on call schedule. The schedule spans a full year, 365 days. Each day lists whether or not there was a trauma, listed as a 0 for no trauma and 1 for trauma. Twelve total residents are listed, labeled as "Resident A" through "Resident L". Similar to the trauma indicator, each resident is marked as on call with a 1 or off call with a 0. A numeric representation of the date and a number of consults column are included but are otherwise unused.

Details of the Methods

After reading in and attaching the dataset, the observed average number of traumas for each residents was acquired. First, a simple sum function was used to total the number of days each resident was on call. A self-built function was used in which the desired resident column and the Trauma column were taken as parameters. A loop involving an if-statement was then used to determine both if there was a trauma on a given day and if the resident was on call for that day. A counter variable was used to keep track of how many traumas each resident witnessed. The total trauma count was then divided by the resident's total number of on calls for the average, which was then returned by the function. The average of all residents was also calculated by adding the twelve averages up and dividing by twelve.

Next, the differences for each resident's average from the overall average was calculated and used as observed values. A loop was then used to randomize the Trauma column and record how many times each resident's average number of traumas was greater than or less than the all resident average for each randomization. A separate count for both white cloud and black cloud was kept to make interpretations slightly faster, but is otherwise unnecessary. In each loop, a new variable is created that stores a randomized Trauma column using the sample function. The self-built function is the called to calculate the average for each resident for this randomization, and then an overall mean for all of the residents is calculated. As mentioned before, the difference between each resident average and the overall resident average is calculated. This is then compared to the observed difference for each resident in an if-statement. The decision is made based on whether or not the randomized difference is greater than or less than the observed difference. If greater than or equal to, a "black cloud" counter variable is incremented, otherwise a "white cloud" counter variable is incremented. After the desired number of randomizations

were achieved, the p-values were then calculated by dividing the “black cloud” and “white cloud” counter variables by the number of randomizations. If-statements are then used to determine if the resulting p-values are significant are not. This is purely to help quickly pick out which residents have significance, otherwise this can be done by one’s discretion.

Results

In carrying out the test, 10,000 total randomizations were made. To determine significance, an alpha of 0.05 was used. These results are found without any predetermined seed. The hypotheses are as follows:

H0: The average number of traumas for a given resident is not different from the overall average at all possible outcomes

HA1: The average number of traumas for a given resident is greater than the overall average

HA2: The average number of traumas for a given resident is less than the overall average

The alternative hypothesis is then broken up to consider both “black clouds” and “white clouds”, and looks at each separately.

The results of the final run of the test show that Resident E is statistically significant in the “black cloud” area. As a result, the null hypothesis can be rejected and, in this case, it can be concluded that the average number of traumas for Resident E is greater than the overall average, and can be considered a black cloud. It is also found that Resident B is statistically significant in the “white cloud” area. In this case, the null hypothesis can be rejected and it can be concluded that the average number of traumas for Resident B is less than the overall average, and can be considered a white cloud.

```
> (cbind(Bpvals,Bsigs))
                Bsigs
BpvalA "0.5509" "0"
```

```
BpvalB "0.9812" "0"

BpvalC "0.0593" "0"

BpvalD "0.1596" "0"

BpvalE "0.0292" "Significant"

BpvalF "0.8897" "0"

BpvalG "0.8102" "0"

BpvalH "0.1236" "0"

BpvalI "0.8838" "0"

BpvalJ "0.273"  "0"

BpvalK "0.8846" "0"

BpvalL "0.4249" "0"

> (cbind(Wpvals,Wsigs))

               Wsigs

WpvalA "0.4491" "0"

WpvalB "0.0188" "Significant"

WpvalC "0.9407" "0"

WpvalD "0.8404" "0"

WpvalE "0.9708" "0"

WpvalF "0.1103" "0"

WpvalG "0.1898" "0"

WpvalH "0.8764" "0"

WpvalI "0.1162" "0"

WpvalJ "0.727"  "0"

WpvalK "0.1154" "0"

WpvalL "0.5751" "0"
```

Discussions and Future Uses

The main point of discussion comes from the method of randomization test and how closely it follows a permutation test. A permutation test involves pooling together two samples and then taking all possible permutations of the observed data while keeping the original two respective sample sizes. Each permutation is then used to calculate the intended statistic and then compared to the original, observed statistic. In the case of this project, some problems are ran into when considering this method. To begin with, 365 days are dealt with, which, by itself, would create a very large sample size required to meet all possible permutations. Another issue comes from dealing with twelve total residents, as well as overlapping days that residents are on call and differing numbers of residents on call each day. In sacrificing an arguably unreasonably high permutation size, a smaller, more reasonable sample size is used. Rather than acquire all possible permutations, a randomized sample that is controllable by seed, but otherwise random is used. The large sample size and randomness still keep the fundamentals of the permutation test, but all possible outcomes are sacrificed for both time and computing power. The idea is that at a high enough sample size, it is possible to achieve the true results of a distribution.

One use for this project is that it could help set a precedent for similar situations in which traditional statistical tests do not quite fit. This project also helps to show applications of randomization tests and helps contribute to the growing perceived practicality of the tests. Another use could include a similar study in which this one is used as groundwork. This study could take this work further and create an experimental study based on previous standing data based on which resident is a white or black cloud and how the two interact. Another route that could be taken is to consider synergy, or how pairing or grouping different residents could be an indicator of trauma chances, looking at both black and white clouds. A different approach than

traditional significance testing could be considered, such as a “black-white cloud spectrum”, in which residents are comparable to one another. Above all else, this project helps to open up multiple possible new directions, as well as offers a different perspective on statistical testing using randomization tests.

Appendix: Code

```
setwd("F:\\Honors Project")

data=read.csv("BlackCloudResidentsDataforRandTest.csv")

attach(data)

head(data)


avgTs = function(resident, traumas) {
        Tcount=0
        for(i in 1:365) {
        if (resident[i]==1) (Tcount = Tcount + traumas[i]) }
        avg = Tcount/sum(resident)
        return(avg) }


# Total days each resident worked (sample size)


Adays = sum(ResA)

Bdays = sum(ResB)

Cdays = sum(ResC)

Ddays = sum(ResD)

Edays = sum(ResE)

Fdays = sum(ResF)

Gdays = sum(ResG)

Hdays = sum(ResH)
```

```
Idays = sum(ResI)

Jdays = sum(ResJ)

Kdays = sum(ResK)

Ldays = sum(ResL)


# Average number of days each resident had a trauma come in


avgA = avgTs(ResA, Trauma)

avgB = avgTs(ResB, Trauma)

avgC = avgTs(ResC, Trauma)

avgD = avgTs(ResD, Trauma)

avgE = avgTs(ResE, Trauma)

avgF = avgTs(ResF, Trauma)

avgG = avgTs(ResG, Trauma)

avgH = avgTs(ResH, Trauma)

avgI = avgTs(ResI, Trauma)

avgJ = avgTs(ResJ, Trauma)

avgK = avgTs(ResK, Trauma)

avgL = avgTs(ResL, Trauma)

rbind(avgA, avgB, avgC, avgD, avgE, avgF, avgG, avgH, avgI, avgJ, avgK, avgL)

mean = (avgA+avgB+avgC+avgD+avgE+avgF+avgG+avgH+avgI+avgJ+avgK+avgL)/12

mean


#The "observed" difference in overall mean and resident mean
```

```
obsdiffA = avgA - mean

obsdiffB = avgB - mean

obsdiffC = avgC - mean

obsdiffD = avgD - mean

obsdiffE = avgE - mean

obsdiffF = avgF - mean

obsdiffG = avgG - mean

obsdiffH = avgH - mean

obsdiffI = avgI - mean

obsdiffJ = avgJ - mean

obsdiffK = avgK - mean

obsdiffL = avgL - mean

obsdiffs = rbind(obsdiffA,obsdiffB,obsdiffC,obsdiffD,obsdiffE,obsdiffF,obsdiffG,obsdiffH,
        obsdiffI,obsdiffJ,obsdiffK,obsdiffL)


#Initializes significance counters


BsigsA = 0

BsigsB = 0

BsigsC = 0

BsigsD = 0

BsigsE = 0

BsigsF = 0
```

```
BsigsG = 0

BsigsH = 0

BsigsI = 0

BsigsJ = 0

BsigsK = 0

BsigsL = 0


WsigsA = 0

WsigsB = 0

WsigsC = 0

WsigsD = 0

WsigsE = 0

WsigsF = 0

WsigsG = 0

WsigsH = 0

WsigsI = 0

WsigsJ = 0

WsigsK = 0

WsigsL = 0


perms = 10000



for (n in 1:perms) {
```

```
        ptrauma = sample(Trauma)
        pavgA = avgTs(ResA, ptrauma)
        pavgB = avgTs(ResB, ptrauma)
        pavgC = avgTs(ResC, ptrauma)
        pavgD = avgTs(ResD, ptrauma)
        pavgE = avgTs(ResE, ptrauma)
        pavgF = avgTs(ResF, ptrauma)
        pavgG = avgTs(ResG, ptrauma)
        pavgH = avgTs(ResH, ptrauma)
        pavgI = avgTs(ResI, ptrauma)
        pavgJ = avgTs(ResJ, ptrauma)
        pavgK = avgTs(ResK, ptrauma)
        pavgL = avgTs(ResL, ptrauma)
        pmean =
(pavgA+pavgB+pavgC+pavgD+pavgE+pavgF+pavgG+pavgH+pavgI+pavgJ+pavgK+pavgL)/12

        if((pavgA - pmean) >= obsdiffA) {BsigsA = BsigsA + 1} else {WsigsA = WsigsA + 1}
        if((pavgB - pmean) >= obsdiffB) {BsigsB = BsigsB + 1} else {WsigsB = WsigsB + 1}
        if((pavgC - pmean) >= obsdiffC) {BsigsC = BsigsC + 1} else {WsigsC = WsigsC + 1}
        if((pavgD - pmean) >= obsdiffD) {BsigsD = BsigsD + 1} else {WsigsD = WsigsD + 1}
        if((pavgE - pmean) >= obsdiffE) {BsigsE = BsigsE + 1} else {WsigsE = WsigsE + 1}
        if((pavgF - pmean) >= obsdiffF) {BsigsF = BsigsF + 1} else {WsigsF = WsigsF + 1}
        if((pavgG - pmean) >= obsdiffG) {BsigsG = BsigsG + 1} else {WsigsG = WsigsG + 1}
        if((pavgH - pmean) >= obsdiffH) {BsigsH = BsigsH + 1} else {WsigsH = WsigsH + 1}
```

```
    if((pavgI - pmean) >= obsdiffI) {BsigsI = BsigsI + 1} else {WsigsI = WsigsI + 1}
    if((pavgJ - pmean) >= obsdiffJ) {BsigsJ = BsigsJ + 1} else {WsigsJ = WsigsJ + 1}
    if((pavgK - pmean) >= obsdiffK) {BsigsK = BsigsK + 1} else {WsigsK = WsigsK + 1}
    if((pavgL - pmean) >= obsdiffL) {BsigsL = BsigsL + 1} else {WsigsL = WsigsL + 1} }

#Provides the p-value for each resident

BpvalA = BsigsA / perms
BpvalB = BsigsB / perms
BpvalC = BsigsC / perms
BpvalD = BsigsD / perms
BpvalE = BsigsE / perms
BpvalF = BsigsF / perms
BpvalG = BsigsG / perms
BpvalH = BsigsH / perms
BpvalI = BsigsI / perms
BpvalJ = BsigsJ / perms
BpvalK = BsigsK / perms
BpvalL = BsigsL / perms

WpvalA = WsigsA / perms
WpvalB = WsigsB / perms
WpvalC = WsigsC / perms
WpvalD = WsigsD / perms
```

```
WpvalE = WsigsE / perms

WpvalF = WsigsF / perms

WpvalG = WsigsG / perms

WpvalH = WsigsH / perms

WpvalI = WsigsI / perms

WpvalJ = WsigsJ / perms

WpvalK = WsigsK / perms

WpvalL = WsigsL / perms


(Bpvals =

rbind(BpvalA,BpvalB,BpvalC,BpvalD,BpvalE,BpvalF,BpvalG,BpvalH,BpvalI,BpvalJ,BpvalK,B

pvalL))

(Wpvals =

rbind(WpvalA,WpvalB,WpvalC,WpvalD,WpvalE,WpvalF,WpvalG,WpvalH,WpvalI,WpvalJ,W

pvalK,WpvalL))


#Sets the significance value and initializes significance vectors

alpha = 0.05

Bsigs = numeric(length=12)

Wsigs = numeric(length=12)


for (m in 1:12) {

        if (Bpvals[m] < alpha) (Bsigs[m]="Significant")

        if (Wpvals[m] < alpha) (Wsigs[m]="Significant") }
```

(cbind(Bpvals,Bsigs))

(cbind(Wpvals,Wsigs))