Chad Germany, Xinhao Liu, Jerry Liu
Stat 480: Big Data Analytics
September 15, 2023

Homework 1

We would like to pre-screen words that are associated with ratings. To this end, we run a series of (independent) marginal regressions of review Score on word presence in review text for each of 1125 words.

Question 1. Plot the p-values from the marginal screening and comment on their distribution.

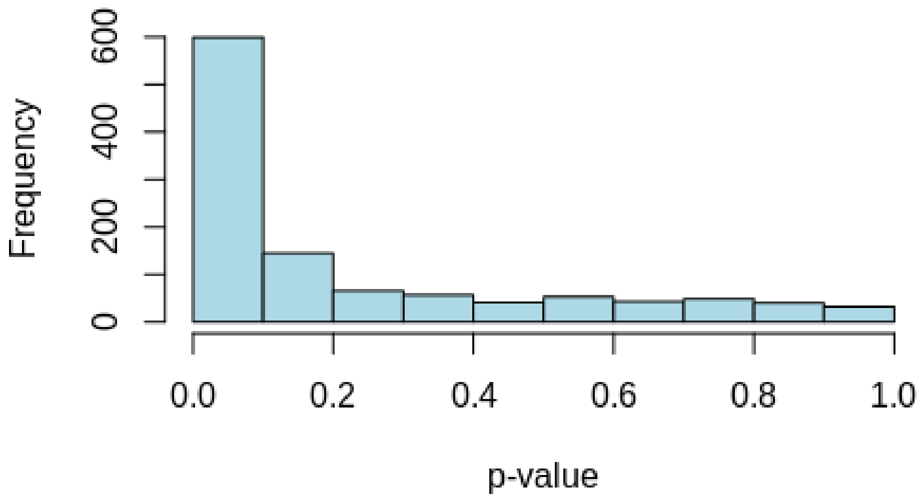

FIGURE 1. The distribution of p-values.

If the null hypothesis was true then this distribution would be uniform. But for our case shown in figure 1 we observe a right-skewed histogram, with many small $p-values$, from this we can conclude that there are words in our dataset that have strong explanatory power in predicting the score.

Question 2. Let's do standard statistical testing. How many tests are significant at the alpha level 0.05 and 0.01?

For $p < \alpha$ we conclude that those alternative hypotheses are significant. So for an α level of .05 we get 461 tests. At an $alpha$ level of .01 we have 348 significant tests.

Question 3. What is the p-value cutoff for .01 FDR? Plot and describe the rejection region.

The rejection region is $p - values <= \alpha^{\star}$. The highest $p - value$ that is $<= q * k/p$ is 0.00241, where q=.01, k is the rank (the p's are in ascending order), p is the total number of $p - values$. When we control the FDR at (say) level q = .01, we are rejecting as many null hypotheses as possible while guaranteeing that no more than .01 of those rejected null hypotheses are false positives, on average. For instance, suppose we control the FDR for p null hypotheses at q = 0.01. This means that if we repeat this experiment a huge number of times, and each time control the FDR at q = 0.01, then we should expect that, on average, .01 of the rejected null hypotheses will be false positives. On a given dataset, the fraction of false positives among the rejected hypotheses may be greater than or less than .01 [1].

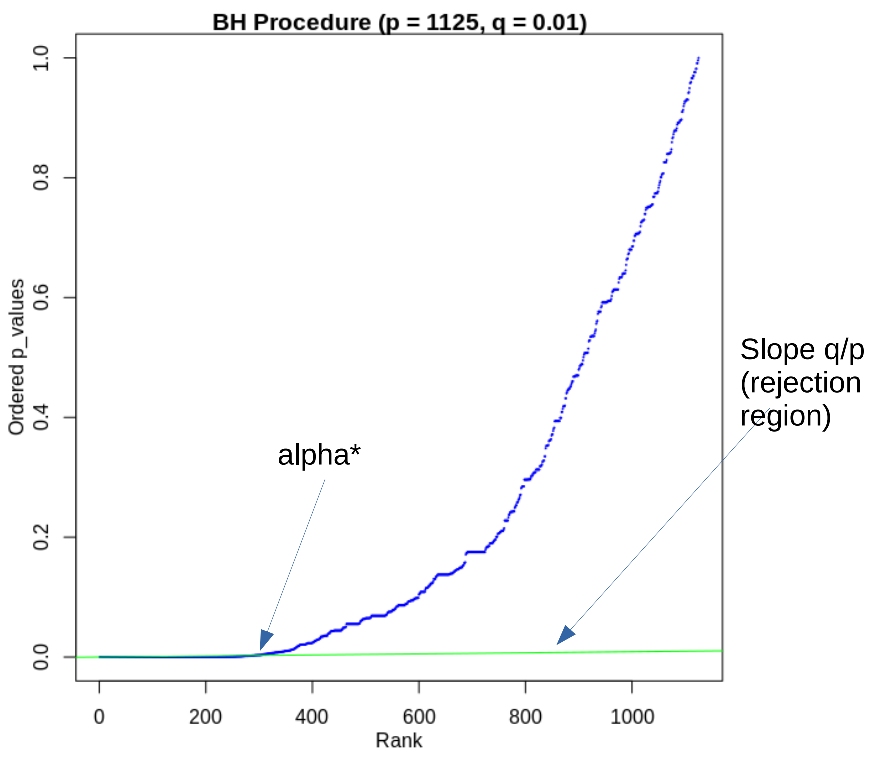

FIGURE 2. The scatterplot of p values using the BH procedure.

Any points below the green slope in figure 2 will be rejected according to the BH algorithm. The slope is q/(total number of p). The alpha star is the highest p value or highest rank p at or below the slope which is k of 290.

Question 4. How many discoveries do you find at q=0.01 and how many do you expect to be false?

For the BH algorithm it ensures that, on average, no more than a fraction q of the rejected null hypotheses are false positives. There are 290 alternate hypotheses discoveries meaning we've rejected 290 null hypotheses. The number of false positives (the number of null hypotheses we rejected by mistake) should be $q * 290$ or ≈ 3 null hypotheses.

Question 5. What are the 10 most significant words? Do these results make sense to you? What are the advantages and disadvantages of our FDR analysis?

The 10 most significant words: ”not”, ”horrible”, ”great”, ”bad”, ”nasty”, ”disappointed”, ”new”, ”but”, ”same”, ”poor”. Let’s look at number 2 which is “horrible”. We see that it shows up in 73 reviews out of 13,319 with varying frequency. Looking at the figure 3 we see that there is definitely a trend between the score given and whether the word was written. Looking at the trendline we see that when the word popped up in written reviews the scores were generally lower. So we can use this information to say that if someone writes the word “horrible” we should expect less stars for that review. Which makes sense because “horrible” has a strong negative connotation attached to it. The same can be said for the other words like “nasty”, “disappointed”, and “poor.” All these words have negative connotations.

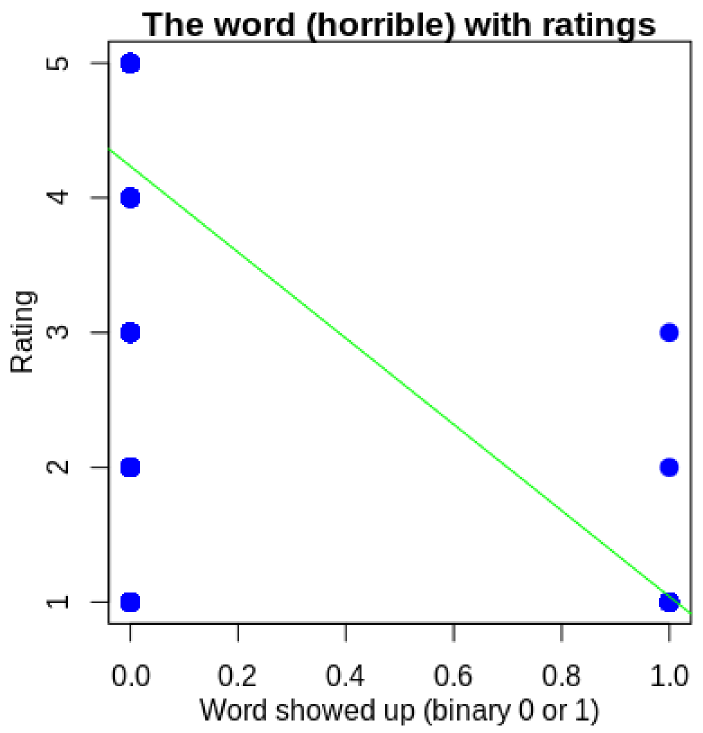

Figure 3. The scatterplot of the number of stars with respect to the whether the word was present in the review or not.

An advantage of the BH algorithm is that the bounds on the FDR set by q holds regardless of how many null hypotheses are true, and regardless of the distribution of the $p-values$ for the null hypotheses that are false. Therefore, the Benjamini–Hochberg procedure gives us a very easy way to determine, given a set of m p-values, which null hypotheses to reject in order to control the FDR at any pre-specified level q [1]. A major disadvantage is that if the test statistics are not independent of each (they are correlated) then the bound on the FDR is no longer valid.

```
# ***** AMAZON REVIEWS

# READ REVIEWS

data<-read.table("Review_subset.csv",header=TRUE)
dim(data)

# 13319 reviews
# ProductID: Amazon ASIN product code
# UserID:  id of the reviewer
# Score: numeric from 1 to 5
# Time: date of the review
# Summary: text review
```

```r
# nrev: number of reviews by this user
# Length: length of the review (number of words)

# READ WORDS

words<-read.table("words.csv")
words<-words[,1]
length(words)
#1125 unique words

# READ text-word pairings file

doc_word<-read.table("word_freq.csv")
names(doc_word)<-c("Review ID","Word ID","Times Word" )
# Review ID: row of the file  Review_subset
# Word ID: index of the word
# Times Word: number of times this word occurred in the text


# We'll do 1125 univariate regressions of
# star rating on word presence, one for each word.
# Each regression will return a p-value, and we can
# use this as an initial screen for useful words.

# Don't worry if you do not understand the code now.
# We will go over similar code in  the class in a few weeks.

# Create a sparse matrix of word presence


library(gamlr)

spm<-sparseMatrix(i=doc_word[,1],
                  j=doc_word[,2],
                  x=doc_word[,3],
                  dimnames=list(id=1:nrow(data),words=words))

dim(spm)
# 13319 reviews using 1125 words

# Create a dense matrix of word presence

P <- as.data.frame(as.matrix(spm>0))

library(parallel)

margreg <- function(p){
  fit <- lm(stars~p)
  sf <- summary(fit)
  return(sf$coef[2,4])
}

# The code below is an example of parallel computing
# No need to understand details now, we will discuss more later

cl <- makeCluster(detectCores())

# Pull out stars and export to cores

stars <- data$Score
```

```

clusterExport(cl,"stars")

# Run the regressions in parallel

mrgpvals <- unlist(parLapply(cl,P,margreg))

# If parallel stuff is not working,
# you can also just do (in serial):
# mrgpvals <- c()
# for(j in 1:1125){
#   print(j)
#   mrgpvals <- c(mrgpvals,margreg(P[,j]))
# }
# make sure we have names

names(mrgpvals) <- colnames(P)

# The p-values are stored in mrgpvals
pvals_ordered<-mrgpvals[order(mrgpvals,decreasing=F)]

plot(pvals_ordered,pch=19)
hist(pvals_ordered, xlab="p-value", main="", col="lightblue")
start <- pvals_ordered[ which(pvals_ordered > .05)]
cutoff_idx = which(start[1]  == pvals_ordered)
cutoff_idx

sig_test <- pvals_ordered[ which(pvals_ordered > .01)]
length(sig_test)

print(pvals_ordered[1])
print(pvals_ordered[20])
len_pval = length(pvals_ordered)

pvals_reject = c()
q = .01
for(j in 1:len_pval){
  #print(q*j/len_pval)
  if( pvals_ordered[j] < q*j/len_pval){
    pvals_reject = append(pvals_reject,pvals_ordered[j])
  }
}

#find the max from this vector
pvals_bh_cutoff = max(pvals_reject)
pvals_bh_accept = pvals_ordered[ which(pvals_ordered > pvals_bh_cutoff)]
pvals_bh_reject = pvals_ordered[which(pvals_ordered <= pvals_bh_cutoff)]
print(length(pvals_bh_accept))

num_false_pos = length(pvals_bh_reject)*q
```

Listing 1. R source code implemented for the homework.

References

[1] Hastie Trevor Tibshirani Robert James Gareth, Witten Daniela. *An introduction to mathematical cryptography*, volume 2.