HW1

Yining Qu

2024-03-25

R Markdown

```
# ***** AMAZON REVIEWS

# READ REVIEWS

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

```
## [1] 13319     9
```

```
# 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
# 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)
```

```
## [1] 1125
```

```
#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.
```

```r
# 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)
```

```
## Loading required package: Matrix
```

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

dim(spm)
```

```
## [1] 13319  1125
```

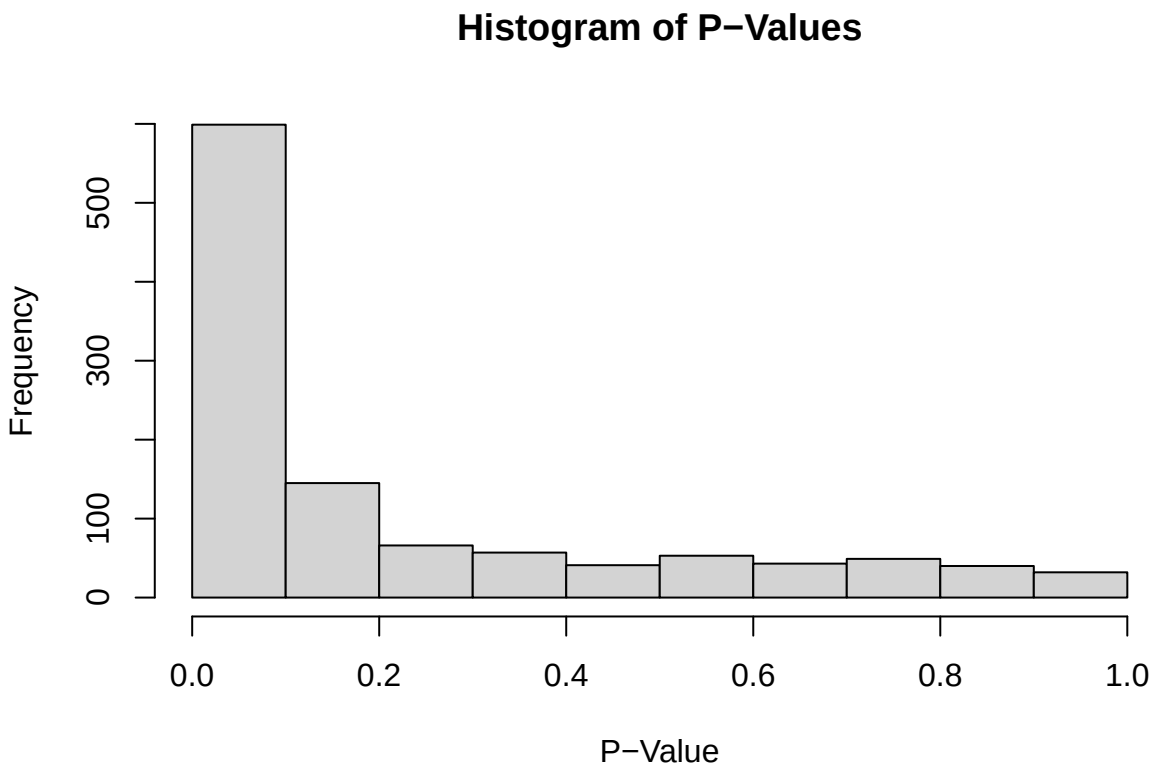
```r
# 13319 reviews using 1125 words

# 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]))
# }
```

```r
# make sure we have names

names(mrgpvals) <- colnames(P)

# The p-values are stored in mrgpvals
```

Q1: Plot the p-values and comment on their distribution.

There is a high frequency of p-values close to 0 (in range 0 - 0.02). This suggests that many of the words are statistically significantly associated with the star ratings. As the p-value increases, the frequency of the p-values seems to even out, showing a roughly uniform distribution. If the null hypothesis is true, which means the word has no association with the star rating, then the p-value is uniformly distributed between 0 and 1.

```r
hist(mrgpvals,
     main="Histogram of P-Values",
     xlab="P-Value",
     ylab="Frequency")
```



Q2: How many tests are significant at the alpha level 0.05 and 0.01?

At alpha level 0.05, 461 tests are significant At alpha level 0.01, 348 tests are significant

```r
# Count the number of tests significant at alpha level 0.05
sum(mrgpvals < 0.05)
```

```
## [1] 461
```

```r
# Count the number of tests significant at alpha level 0.01
sum(mrgpvals < 0.01)
```

```
## [1] 348
```

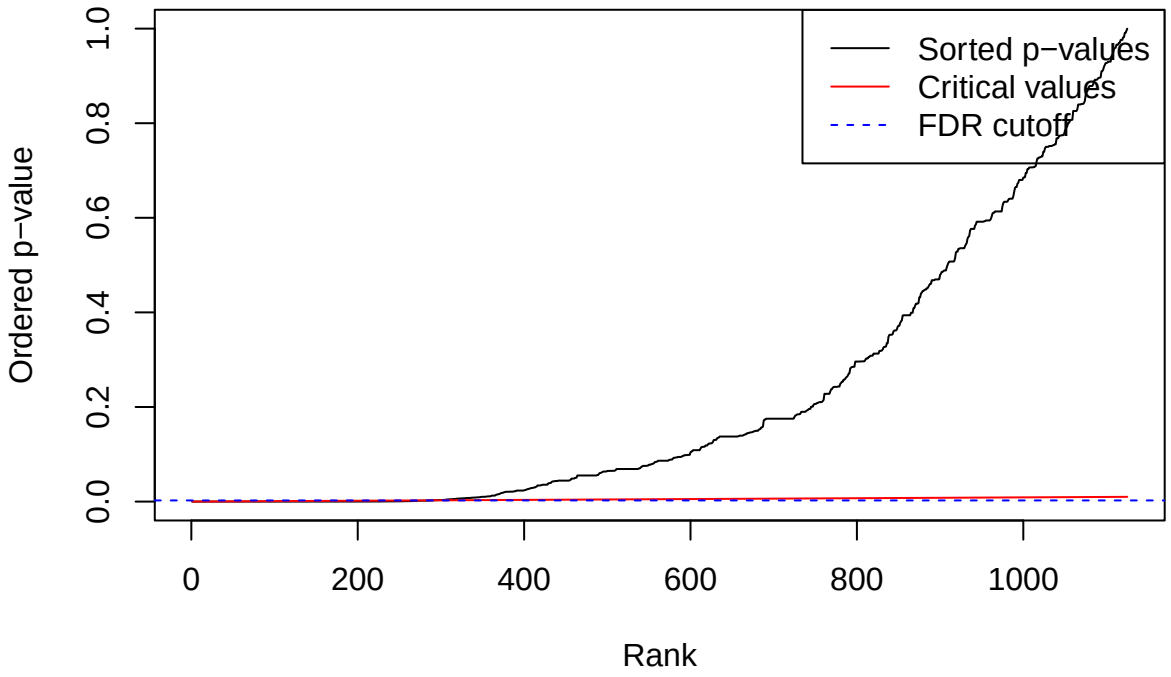
Q3: What is the p-value cutoff for 1% FDR? Plot and describe the rejection region.

p-value cutoff for 1% FDR is 0.002413248819859

```
# Sort p-values in ascending order
p_sorted <- sort(mrgpvals)

# Number of tests
n <- length(p_sorted)

# Desired FDR level
FDR <- 0.01

# Calculate p-value cut-off
bh_critical_values <- (1:n) / n * FDR

# Find the largest p-value where the p-value is less than the critical value
cutoff_index <- max(which(p_sorted < bh_critical_values))
fdr_cutoff <- p_sorted[cutoff_index]
print(paste("FDR 1% cutoff p-value:", fdr_cutoff))
```

```
## [1] "FDR 1% cutoff p-value: 0.002413248819859"
```

```
# Plot the sorted p-values
plot(p_sorted, type = "l", xlab = "Rank", ylab = "Ordered p-value")
# Plot the BH critical values
lines(bh_critical_values, col = "red")
# Add a horizontal line at the FDR cutoff
abline(h = fdr_cutoff, col = "blue", lty = 2)
# Add a legend
legend("topright", legend = c("Sorted p-values", "Critical values", "FDR cutoff"),
       col = c("black", "red", "blue"), lty = c(1, 1, 2))
```



The rejection region is the region below the blue dashed line (or region that the black line falls below the red line). These represent the words most significantly correlated with the star ratings. The number of words that fall into this region is the number of discoveries at a 1% FDR.

Q4: How many discoveries do you find at q=0.01 and how many do you expect to be false?

I find 290 discoveries at q=0.01, 2.9 tests are expected to be false

```
num_reject <- sum(p_sorted <= fdr_cutoff)
print(num_reject)
```

```
## [1] 290
```

```
print(num_reject * 0.01)
```

```
## [1] 2.9
```

Q5: What are the 10 most significant words? Do these results make sense to you? What are the advantages and disadvantages of our FDR anaysis?

10 most significant words are the ones with lowest p-values, which are “not”, “horrible”, “great”, “bad”, “nasty”, “disappointed”, “new”, “but”, “same”, “poor”.

These 10 words make sense, because they are emotionally charged or convey clear opinions. Words like “horrible”, “bad”, “nasty”, “disappointed”, and “poor” are typically used in negative reviews, while “great” suggests positive feedback. “Not” could be part of a negative phrase, such as “not good” or “not worth the money”. “New” might be associated with expectations or novelty. “But” and “same” could be part of a more nuanced review, where the reviewer might be comparing aspects of the product or setting up a contrast.

Advantages: 1. It strikes a balance between identifying significant results and controlling the rate of false positives, which is especially important when running many tests.FDR analysis controls the expected proportion of false positives among the rejected hypotheses, which thereby controls the risk of False discoveries in rejections. 2. FDR typically allows for more discoveries because it’s less conservative. 3. FDR control can be more appropriate for exploratory analysis where we’re willing to accept a higher risk of false discoveries

in order not to miss potentially important findings.FDR is flexible in terms of application, since it can be applied to any model that involves Hypothesis Testing. 4. FDR provides a simple summary of the risk of False discoveries in rejections.

Disadvantages: 1. Even at a 1% FDR, there's a chance that some of the words deemed significant might be false positives. 2. The original BH procedure assumes that tests are independent, which might not be true in all datasets, potentially leading to an incorrect number of false discoveries. 3. Understanding what FDR control actually means in practice can be complex; it controls the expected proportion of false positives, not the actual number.

```
p_sorted[1:10]
```

```
##           not      horrible         great           bad         nasty
## 1.568776e-165 3.012007e-100  2.556997e-74  2.876555e-58  1.121720e-50
##  disappointed           new           but          same          poor
##  2.148354e-48  9.285538e-46  2.703840e-45  4.454271e-40  2.299803e-39
```