Random Variables & Probability – Assignment 6

Sharon Morris

3/6/2017

Problem Set (1) When you roll a fair die 3 times, how many possible outcomes are there?

There are 6 possible outcomes with each roll of a die $= 6^n$

```
die <- 6
outcome <- die^3
outcome
```

```
## [1] 216
```

(2) What is the probability of getting a sum total of 3 when you roll a die two times?

There are 2 successful possibilities $1 + 2$ and $2 + 1$ and 6 possible outcomes $6^2 = 36$

```
library(dice)
```

```
## Loading required package: gtools
```

```
getEventProb(nrolls = 2,
             ndicePerRoll = 1,
             nsidesPerDie = 6,
             eventList = list(1,2),
             orderMatters = FALSE)
```

```
## [1] 0.05555556
```

(3) Assume a room of 25 strangers. What is the probability that two of them have the same birthday? Assume that all birthdays are equally likely and equal to $\frac{1}{365}$ each.

The probability of the first person's= $\frac{365}{365}$ of the 2nd person $\frac{364}{365}$...

```
# 25 people in the room
n <- 25
p <- 365
1 - prod(seq(p - n + 1, p)) / p^n
```

```
## [1] 0.5686997
```

What happens to this probability when there are 50 people in the room?

```
n <- 50
1 - prod(seq(p - n + 1, p)) / p^n
```

```
## [1] 0.9703736
```

Problem Set 2 Sometimes you cannot compute the probability of an outcome by measuring the sample space and examining the symmetries of the underlying physical phenomenon, as you could do when you rolled die or picked a card from a shuffled deck. You have to estimate probabilities by other means. For instance, when you have to compute the probability of various english words, it is not possible to do it by examination of the sample space as it is too large. You have to resort to empirical techniques to get a good enough estimate. One such approach would be to take a large corpus of documents and from those documents, count the number of occurrences of a particular character or word and then base your estimate on that.

Write a program to take a document in English and print out the estimated probabilities for each of the words that occur in that document. Your program should take in a file containing a large document and

write out the probabilities of each of the words that appear in that document. Please remove all punctuation (quotes, commas, hyphens etc) and convert the words to lower case before you perform your calculations.

Extend your program to calculate the probability of two words occurring adjacent to each other. It should take in a document, and two words (say the and for) and compute the probability of each of the words occurring in the document and the joint probability of both of them occurring together. The order of the two words is not important.

Use the accompanying document for your testing purposes. Compare your probabilities of various words with the Time Magazine corpus: http://corpus.byu.edu/time/

The file is from the gutenberg.org which list thousands of free ebooks. The book used for this exercise is Ethics by Aristotle - the text file is 646KB.

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 3.3.2
```

```
## Loading required package: NLP
```

```
library(SnowballC)

#Read text files from gutenberg.org
filepath <- "http://www.gutenberg.org/cache/epub/8438/pg8438.txt"
text <- readLines(filepath)

#Load the data as corpus
docs <- Corpus(VectorSource(text)) #vectorSource creates a corpus of character vectors

# Use tm_map() function to replace "/", "@" and "|" with space
toSpace <- content_transformer(function(x, pattern ) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "/")
docs <- tm_map(docs, toSpace, "@")
docs <- tm_map(docs, toSpace, "\\|")

#Cleaning the text
#Convert the text to lower case
docs <- tm_map(docs, content_transformer(tolower))

# Remove numbers
docs <- tm_map(docs, removeNumbers)

# Remove punctuations
docs <- tm_map(docs, removePunctuation)

#Eliminate extra white spaces
docs <- tm_map(docs, stripWhitespace)

#Count the number of words
library(stringr)
words <- unlist(str_split(docs, "[[:blank:]]+"))
words <- words[nchar(words) != 0]
wordCount <- length(words)
wordCount
```

```
## [1] 116618
```

Probability of a unigram $P(w_1) = c\frac{(w_1 w_2)}{N}$

```
freq <- as.data.frame(table(words))
prob <- round(freq[ ,2] / wordCount, 6)
results <- cbind(freq, prob)
head(results[order(-results[ ,2], results[ ,1]), ])
```

```
##       words Freq     prob
## 10246   the 5759 0.049383
## 7882     of 3984 0.034163
## 3170    and 3294 0.028246
## 10360    to 3108 0.026651
## 6889     is 3052 0.026171
## 6618     in 2327 0.019954
```

Probability of a bigram $P(w_1w_2)$ $P(w_1w_2) = c\frac{(w_1w_2)}{N}$

```
tFreq <- results[results[1] == "the"]
fFreq <- results[results[1] == "for"]
tProb <- as.numeric(tFreq[3])
fProb <- as.numeric(fFreq[3])
iProb <- tProb * fProb; iProb
```

```
## [1] 0.0004624224
```

```
fThe <- unlist(str_match_all(docs, " for the "))
tFor <- unlist(str_match_all(docs, " the for "))
ftFreq <- length(fThe)
tfFreq <- length(tFor)
eProb <- (ftFreq + tfFreq) / wordCount; eProb
```

```
## [1] 0.001071876
```

References

https://cran.r-project.org/web/packages/dice/dice.pdf

https://www.khanacademy.org/math/precalculus/prob-comb/prob-combinatorics-precalc/v/birthday-probability-problem

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/birthday.html

http://www.sthda.com/english/wiki/text-mining-and-word-cloud-fundament

https://english.boisestate.edu/johnfry/files/2013/04/bigram-2x2.pdf