IS 605 - Assignment 6

Dan Fanelli

March 9, 2016

Problem Set 1:

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

```
6 * 6 * 6
```

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

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

```
# a 1 or a 2 on the first die, then the opposite on the next...
(2/6) * (1/6)
```

```
## [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 1=365 each. What happens to this probability when there are 50 people in the room?

Probably that EXACTLY 2 have the same birthday:

- 1 of the 25 can have their birthday any day of the year (prob = 1). This will be our "to be duplicated". They can be in any one of the 25 "positions"
- The duplicate will be our next draw with their birthday on that same day (1/365), but the position of this draw can be in 24 of the remaining "spots", so multiply by 24.
- The next 23 must be unique (prob = 364/365 * 363/364 * 362/363 * ... * 340/341 * 341/342). Their orderings placements don't matter.
- Notice that each numerator (except the last) is the same as the denominator to its right, so everything cancels except the first denominator (365) and the last numerator (341), giving us: (341/365) odds that the last 23 will be on unique days.
- So the final equation is 1 * (23 * (1/365)) * (341/365)

```
1 * (23 * (1/365)) * (341/365)
```

```
## [1] 0.05887033
```

```
# lets put this into a function too and then double check the same values:
p_of_exactly_2 <- function(n) {
  return (1 * ((n-2) * (1/365)) * ((365-n+1)/365))
}

p_of_exactly_2(25)
```

```
## [1] 0.05887033
```

- If there are n people in the room (you can let n = 50), then the equation would be:

```
p_of_exactly_2(50)
```

```
## [1] 0.1138525
```

Probably that AT LEAST 2 have the same birthday:

- This is equal to (1 - P(none have same birthday)), or (1-((365/365)*(364/365)*...*(341/365))
- This can be stated as 1 - (365!/340!)/365^25, but NaN and Inf appear when expressing this as an equation, so we need some code to avoid this:

```
1 - (factorial(365)/factorial(340))/(365^25)
```

```
## [1] NaN
```

```
# the above equation runs out of space, so instead:

to_subtract_from_one <- 1
for (x in 365:341){
  to_subtract_from_one <- (to_subtract_from_one * x / 365)
}
1 - to_subtract_from_one
```

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

```
# lets put this into a function too and then double check the same values:

p_of_at_least_2 <- function(n) {
  to_subtract_from_one <- 1
  for (x in 365:(365-n+1)){
    to_subtract_from_one <- (to_subtract_from_one * x / 365)
  }
  return (1 - to_subtract_from_one)
}

p_of_at_least_2(25)
```

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

- If there are n people in the room (you can let n = 50), then the equation would be:

```
p_of_at_least_2(50)
```

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

Problem Set 2:

Required Functions:

```
# refactored the following to functions since its needed for next part of question:

file_to_clean_string <- function(fileName){
  conn <- file(fileName,open="r")
  lines <- readLines(conn)
  close(conn)
  lines <- paste(lines, collapse=' ')
  lines <- tolower(lines)
  lines <- gsub("[^[:alnum:] ]", "", lines)
  punct <- '[]\\?!\"\'#$%&(){}+*/:;,._`|~\\[<=>@\\^-]'
  # got that regexp online
  lines <- gsub(punct, "", lines)
  # gsub what that expression didn't get
  lines <- gsub("â", "", lines)
  lines <- gsub("  ", " ", lines)
  return (lines)
}

file_to_clean_words_list <- function(fileName){
  all_words_with_repeats <- strsplit(file_to_clean_string(fileName), " ")
  return (all_words_with_repeats)
}

create_word_counts_dataframe <- function(all_words_with_repeats){
  table_with_count <- sort(table(all_words_with_repeats), decreasing = TRUE)
  words <- rownames(table_with_count)
  head(words)
  # data frame with word as col name and count as value
  word_counts_DF <- as.data.frame(table_with_count, words)
  word_counts_DF$the_word <- rownames(word_counts_DF)
  rownames(word_counts_DF) <- 1:nrow(word_counts_DF)
  total_word_count <- sum(word_counts_DF$table_with_count)
  word_counts_DF$word_probability <- word_counts_DF$table_with_count / total_word_count
  head(word_counts_DF, n = 20)
  return (word_counts_DF)
}

file_to_wc_dataframe <- function(fileName){
  all_words_with_repeats <- file_to_clean_words_list(fileName)
  df <- create_word_counts_dataframe(all_words_with_repeats)
  return (df)
}
```

Part A:

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.

```
df_1 <- file_to_wc_dataframe("assign6.sample.txt")
head(df_1, n = 20)
```

Estimated probabilities for each of the words that occur in that document

```
##    table_with_count    the_word word_probability
## 1                76         the      0.056047198
## 2                45           a      0.033185841
## 3                38         and      0.028023599
## 4                31         for      0.022861357
## 5                28          in      0.020648968
## 6                28          of      0.020648968
## 7                28          to      0.020648968
## 8                22          is      0.016224189
## 9                22        said      0.016224189
## 10               19        that      0.014011799
## 11               15    tutwiler      0.011061947
## 12               12          at      0.008849558
## 13               12          it      0.008849558
## 14               11      prison      0.008112094
## 15               10 corrections      0.007374631
## 16               10         its      0.007374631
## 17                9         are      0.006637168
## 18                9        been      0.006637168
## 19                9         but      0.006637168
## 20                9        have      0.006637168
```

```
tail(df_1, n = 20)
```

```
##     table_with_count    the_word word_probability
## 560                1      warden     0.0007374631
## 561                1     watched     0.0007374631
## 562                1         way     0.0007374631
## 563                1        week     0.0007374631
## 564                1    weighing     0.0007374631
## 565                1       which     0.0007374631
## 566                1       while     0.0007374631
## 567                1       whose     0.0007374631
## 568                1 wideranging     0.0007374631
## 569                1        will     0.0007374631
## 570                1     without     0.0007374631
## 571                1       woman     0.0007374631
## 572                1        wood     0.0007374631
## 573                1        work     0.0007374631
## 574                1      worked     0.0007374631
## 575                1     working     0.0007374631
## 576                1       worse     0.0007374631
## 577                1       would     0.0007374631
## 578                1         yes     0.0007374631
## 579                1         you     0.0007374631
```

```r
# double check that the sums of all these "word_probability" values equals 1
sum(df_1$word_probability)
```

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

Part B:

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.

```r
library(stringr)

adjoining_words_report <- function(fileName, word1, word2){
  words_as_single_string <- file_to_clean_string(fileName)
  forwards <- paste(word1, word2, sep=" ")
  backwards <- paste(word2, word1, sep=" ")
  word1_occurrences <- str_count(words_as_single_string, word1)
  word2_occurrences <- str_count(words_as_single_string, word2)
  total_num_words <- str_count(words_as_single_string, " ")

  forwards_count <- str_count(words_as_single_string, forwards)
  backwars_count <- str_count(words_as_single_string, backwards)
  forwards_and_backwards_occurrences <- forwards_count + backwars_count
  # the number of spaces is the total number of 2 word combos, or the total_words - 1
  total_2_word_combos <- str_count(words_as_single_string, " ")
  odds_of_2_words <- (forwards_and_backwards_occurrences / total_2_word_combos)
  cat("THE ODDS OF '", word1,"' IS: ", (word1_occurrences / total_num_words),"\n")
  cat("THE ODDS OF '", word2,"' IS: ", (word2_occurrences / total_num_words),"\n")
  cat("THE ODDS OF THE 2 WORDS TOGETHER IS: ", odds_of_2_words,"\n")
}

# There are 10 words in the fanelli txt file, and it ends with 'lazy dog lazy'
# The Quick broum lazy fox jumped over the laZy doG lazy
adjoining_words_report("assign6.sample.fanelli.1.txt", "lazy", "dog")
```

```
## THE ODDS OF ' lazy ' IS:  0.3
## THE ODDS OF ' dog ' IS:  0.1
## THE ODDS OF THE 2 WORDS TOGETHER IS:  0.2
```

```r
adjoining_words_report("assign6.sample.txt", "corrections", "officers")
```

```
## THE ODDS OF ' corrections ' IS:  0.007380074
## THE ODDS OF ' officers ' IS:  0.004428044
## THE ODDS OF THE 2 WORDS TOGETHER IS:  0.002214022
```

Use the accompanying document for your testing purposes. Compare your probabilities of various words with the Time Magazine corpus: http://corpus.byu.edu/time/1