Week 9 String Manipulation

```
knitr::opts_knit$set(root.dir = "D:/Introduction-to-R/Data")
```

Text processing and string manipulation is an important part of data sciences, as greater amounts of unstructured data is used for analysis. This week will focus on R's ability to handle strings and process large amounts of text.

Weekly Learning Objectives

Create an R script with the following components:

1. Create a function to perform a grep()
2. Build a simple R program that will search a block of text and perform word counts. Include a block graph.
3. Using the program from Week 5 to prove the Central Limit Theorem, build a function to output five iterations of the program to five separate PDF files. Name the files CENTRAL_#.pdf where # = the iteration.

1. Perform a GREP

GREP is a standard feature with many programming languages and operating systems. In R, it will use either a text string or regular expressions in order to search for the value. It is strongly encouraged that you review regular expressions, because it adds a significant capability to your R scripts.

https://www.rdocumentation.org/packages/base/versions/3.6.1/topics/regex

For this assignment, you simply have to build a function that uses GREP. Here are a few examples from r documentation. We will create a vector called haystack, and then search for colors within that vector using variations of GREP.

1. Element position where the word green appears
2. Element position where the letter r appears
3. Value where the letter r appears
4. True or False where the letter r appears

Full Points

Simply build a function that uses GREP for full points.

```
haystack <- c("red", "blue", "green", "blue", "green forest")

grep("green", haystack)
```

```
## [1] 3 5
```

```
grep("r", haystack) # returns position
```

```
## [1] 1 3 5
```

```r
grep("r", haystack, value = TRUE) # returns value
```

```
## [1] "red"          "green"        "green forest"
```

```r
grepl("r", haystack) # returns boolean
```

```
## [1]  TRUE FALSE  TRUE FALSE  TRUE
```

2. Search a block of text

For this question, there are multiple ways of responding. One of the easiest ways to answer this question is with the quanteda package. For information on it:

https://quanteda.io/

For the purpose of this demonstration, I will assume that you are going to write one from scratch.

Full Points

You can use either the packaged library method or write your own function. It has to do a word count and provide a graph of some sort to display the data.

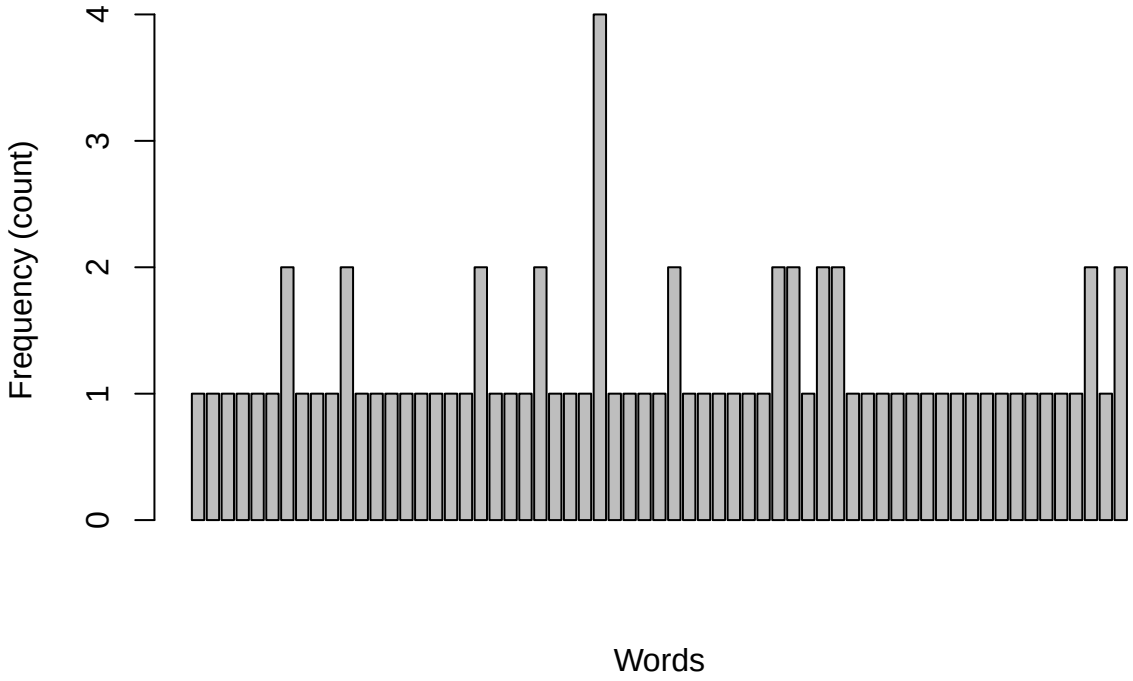
```r
countWords <- function(textBlock) {

  #Remove non-alpha characters and replace with spaces
  aListOfWords <- gsub("[^a-zA-Z]", " ",textBlock)

  #Remove extra spaces
  aListOfWords <- gsub("  +", " ",aListOfWords)

  #Split words by spaces
  aListOfWords <- strsplit(aListOfWords, "\\s")
  aListOfWords <- aListOfWords[[1]]

  # convert to lower case for case-insensitivity
  aListOfWords <- sapply(aListOfWords, tolower)

  aListOfWords <- sort(aListOfWords)

  word <- unique(aListOfWords)
  wordFreq <- rep(0,length(word))
  aResult <- data.frame(word,wordFreq)          # Populate word list and pre-allocate data structure

  # Initialize variables for the loop
  aCount <- 1
  wordIndex <- 1

  for(i in 2:length(aListOfWords)) {
    if (aListOfWords[i] != aListOfWords[i-1]) {
      aResult$wordFreq[wordIndex] <- aCount
      wordIndex <- wordIndex + 1
      aCount <- 1
```

```
    } else {
      aCount <- aCount+1
    }
    aResult$wordFreq[wordIndex] <- aCount
  }
  return(aResult)
}

randomTextString <- "Announcing of invitation principles in. Cold in late or deal. Terminated resolutio

aCount <- countWords(randomTextString)
data.frame(aCount$word, aCount$wordFreq)
```

```
##    aCount.word aCount.wordFreq
## 1    abilities               1
## 2   admiration               1
## 3   affronting               1
## 4           am               1
## 5           an               1
## 6   announcing               1
## 7   apartments               2
## 8   appearance               1
## 9           as               1
## 10  astonished               1
## 11  boisterous               2
## 12        cold               1
## 13  collecting               1
## 14   continued               1
## 15  conviction               1
## 16        deal               1
## 17  devonshire               1
## 18   discourse               1
## 19  discretion               1
## 20          do               2
## 21   enjoyment               1
## 22  excellence               1
## 23  expression               1
## 24  frequently               2
## 25       fully               1
## 26          he               1
## 27          if               1
## 28          in               4
## 29  increasing               1
## 30  inquietude               1
## 31  insensible               1
## 32  instrument               1
## 33  invitation               2
## 34          is               1
## 35          it               1
## 36        late               1
## 37          mr               1
## 38         nay               1
## 39   necessary               1
```

```
## 40          no              2
## 41          of              2
## 42          on              1
## 43          or              2
## 44         own              2
## 45  particular              1
## 46       point              1
## 47  possession              1
## 48  principles              1
## 49  projection              1
## 50   pronounce              1
## 51  resolution              1
## 52       seems              1
## 53  sentiments              1
## 54  simplicity              1
## 55       stand              1
## 56  terminated              1
## 57  themselves              1
## 58  travelling              1
## 59  unreserved              1
## 60       walls              1
## 61          we              2
## 62          ye              1
## 63         you              2
```

```
barplot(aCount$wordFreq,xlab="Words", ylab="Frequency (count)" )
```



3. Central Limit Proof

This starts with the Central Limit Theorem program that you wrote earlier. From that, simply loop through it five times and output to a PDF file.

Full Points

The output should produce five individual PDFs, each with a graph using your solution.

```
clt <- function(diceValues, nbrDice, aSampleNbr, xlab, showCurve=FALSE) {
  means <- numeric(aSampleNbr)
  for (i in 1:aSampleNbr) {
    samples <- sample(diceValues,nbrDice,replace=TRUE)
    aSum <- sum(samples)
    means[i] <- mean(aSum)
  }
  lowerBreak = (diceValues[1]*nbrDice) - .5
  upperBreak = (diceValues[length(diceValues)]*nbrDice) + .5
  hist(means,xlab=xlab,breaks=lowerBreak:upperBreak, main="")

  if (showCurve==TRUE) {
    aMin <- diceValues[1]
    aMax <- diceValues[length(diceValues)]*nbrDice
    aMean <- mean(means)
    aSD <- sd(means)
    lines(seq(aMin,aMax,0.1),dnorm(seq(aMin,aMax,0.1),aMean,aSD)*aSampleNbr)
  }
}

#####   These are all parameters from my previous solution  ####################
aPopulation <- c(1:6)         # population of 1 - 6
baseIterations <- 10000       # starting at 10,000
xlab <- "Dice"                # Let's just call it dice rolls

  for (i in 1:5) {
    fileName <- sprintf("CENTRAL_%d.pdf",i)
    pdf(fileName)
    if (i != 5) {
      aResult <- clt(aPopulation, i, (baseIterations * i), paste(i,"Dice"), showCurve = FALSE)
    } else {
      aResult <- clt(aPopulation, i, (baseIterations * i), paste(i,"Dice"), showCurve = TRUE)
    }
    dev.off()
  }
```