Scientific Programming Assignment 1

303034908

October 2018

1 Words [15 marks]

1. The document contains 97723 unique words.
2. 25751 words contain an apostrophe.
3. 159 words show non-ASCII characters.
4. Words with related forms shown in table 1.

″ANALOG″	″ANALOGUE″
″CATALOG″	″CATALOGUE″
″DEMAGOG″	″DEMAGOGUE″
″DIALOG″	″DIALOGUE″
″EPILOG″	″EPILOGUE″
″MONOLOG″	″MONOLOGUE″
″PEDAGOG″	″PEDAGOGUE″
″PROLOG″	″PROLOGUE″
″SYNAGOG″	″SYNAGOGUE″
″TRAVELOG″	″TRAVELOGUE″

Table 1: Question 1.4: related words

5. score

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	3	3	2	1	4	2	4	1	8	5	1	3	1	1	3	10	1	1	1	1	4	4	8	4	10

6. The highest scoring word is ″PIZZAZZ″ with a score of 45. Figure 1 shows a histogram of the word scores across the database.

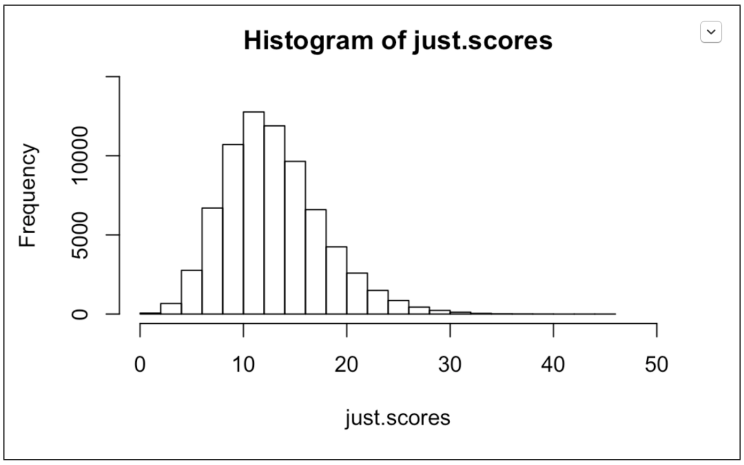

Figure 1: Question 1.6: scrabble score distribution

7. Figure 2 shows the words whose reverse complement as well as themselves are within the database.
8. Given the following nine letters: ″F A L U Y P L N I″, figure 3 shows the words of four or more letters that are in the database and all contain the letter A.

```
  [1] "A"       "AL"      "ARIZ"    "AS"
  [5] "ASP"     "B"       "BEVY"    "BID"
  [9] "BIG"     "BIRD"    "BOIL"    "BOORISH"
 [13] "BOORS"   "BY"      "C"       "CI"
 [17] "CL"      "D"       "E"       "ELM"
 [21] "F"       "FLO"     "FM"      "G"
 [25] "GILL"    "GIRT"    "GO"      "GRID"
 [29] "GRIT"    "H"       "HE"      "HF"
 [33] "HG"      "HI"      "HILLY"   "HO"
 [37] "HOOFS"   "HORN"    "HOVELS"  "HULLS"
 [41] "HZ"      "I"       "IF"      "IN"
 [45] "IO"      "IR"      "J"       "K"
 [49] "KHZ"     "L"       "LN"      "LO"
 [53] "LORN"    "LOU"     "LR"      "LS"
 [57] "LT"      "LYRA"    "M"       "MA"
 [61] "MI"      "MILD"    "MILK"    "MILO"
 [65] "MILS"    "MN"      "MO"      "MR"
 [69] "N"       "ND"      "NOV"     "NU"
 [73] "O"       "OK"      "OORT"    "ORLY"
 [77] "OX"      "OZ"      "P"       "PL"
 [81] "POLK"    "PORN"    "Q"       "R"
 [85] "RH"      "RN"      "RS"      "RX"
 [89] "S"       "SH"      "SHRILLY" "SI"
 [93] "T"       "TRIG"    "TRY"     "TS"
 [97] "U"       "UR"      "US"      "V"
[101] "VELD"    "VOLE"    "VS"      "W"
[105] "WILD"    "WIRY"    "WIZARD"  "WM"
[109] "WORD"    "WORN"    "WOVE"    "WRIT"
[113] "WRY"     "X"       "Y"       "YB"
[117] "Z"       "ZIBO"    "ZN"      "ZOLA"
```

Figure 2: Question 7: words with reverse complement in database

```
 [1] "AINU"      "FULANI"    "LILA"      "LINA"
 [5] "LULA"      "LUNA"      "PAUL"      "PAULI"
 [9] "PIAF"      "YALU"      "YUAN"      "ALLY"
[13] "FAIL"      "FAIN"      "FALL"      "FAUN"
[17] "FINAL"     "FINALLY"   "FLAIL"     "FLAN"
[21] "FLAP"      "FLAY"      "INLAY"     "LAIN"
[25] "NAIL"      "PAIL"      "PAILFUL"   "PAIN"
[29] "PAINFUL"   "PAINFULLY" "PALL"      "PILAF"
[33] "PILAFF"    "PILAU"     "PLAIN"     "PLAINLY"
[37] "PLAN"      "PLAY"      "PLAYFUL"   "ULNA"
```

Figure 3: Question 8: permutations in database

2 Examination marking [10 points]

(a) Table figure 4 shows the completed markings data frame.

(b) Students cheating would a) choose the same questions to answer and b) have the same answers for those questions.

From the pairwise comparison of questions answered (figure 5) we can see that students 2 and 6 chose the same 27/30 questions, suggesting cheating.

Cheating would further be confirmed if the answers to the 27 mutual questions were the same. This is also the case. Hence, cheating occurred between students 2 and 6.

Student	Score	Grade	Rank
1	19	B	6
2	24	A	2
3	14	D	10
4	17	C	8
5	20	B	5
6	23	A	3
7	29	A	1
8	7	F	12
9	22	A	4
10	17	C	9
11	9	F	11
12	18	B	7

Figure 4: Question 2a: examination markings

	1	2	3	4	5	6	7	8	9	10	11	12
1	30	14	12	7	7	14	9	10	13	11	7	11
2	14	30	12	10	6	27	9	13	12	11	9	11
3	12	12	30	10	6	11	9	9	8	9	6	9
4	7	10	10	30	4	8	8	8	10	6	6	8
5	7	6	6	4	30	7	8	9	6	8	7	9
6	14	27	11	8	7	30	9	13	12	12	10	10
7	9	9	9	8	8	9	30	8	12	7	11	12
8	10	13	9	8	9	13	8	30	11	8	9	7
9	13	12	8	10	6	12	12	11	30	8	12	8
10	11	11	9	6	8	12	7	8	8	30	4	7
11	7	9	6	6	7	10	11	9	12	4	30	11
12	11	11	9	8	9	10	12	7	8	7	11	30

Figure 5: Question 2b: pairwise comparison of questions chosen

3 Appendix

```
---
title: "Scientific programming assignment 1"
output:
  html_document:
    df_print: paged
---
## Words
Read in the file usr-share-dict-words; this is a dictionary that comes with Ubuntu linux. Each line
contains one word. You should ignore the case of the letters, i.e. treat upper and lower case as the
same.

```{r setup}
#setwd(~/Desktop)

urs.dict.words <- read.table("usr-share-dict-words.txt")
```
---
1. Read in the file and convert all words to upper case. Use unique() to keep only one copy of each
word. (e.g. Zest and zest should count as one word ZEST.) How many unique words are there?
[1 mark]
```{r q1}
urs.v1 = unique(data.frame(lapply(urs.dict.words, toupper)))

print(paste("The document contains",  length(urs.v1[,1]), "unique words"))
```

```

2. How many words contain an apostrophe (')? Remove these words from the rest of the analysis.
[1 mark]

```{r}
idx <- c(grep("\'", urs.v1[,]))
length(idx) #number of words with apostrophe
urs.v2 <- urs.v1[-idx,] #remove those words from database
urs.v2<- as.vector(urs.v2)
length(urs.v2)
class(urs.v2)
```
---
3. How many words contain non-ASCII characters? (Remove these words from the rest of the analysis;
the remaining words are called the database in the remainder of the question.) [1 mark]

```{r}
#select words with non-ascii characters
words.w.ascii <- tools::showNonASCII(urs.v2)
#count number of non-ASCII words
length(words.w.ascii)


#find indexes of non ascii words in the database
#grep(words.w.ascii, urs.v2)
#grep(paste0(words.w.ascii, collapse = "|"), urs.v2) #'|' acts as an OR
#use %in% for logical indexing
# remove non ascii words from database
urs.v4 = urs.v2[!urs.v2 %in% grep(paste0(words.w.ascii, collapse = "|"), urs.v2, value = TRUE)]


```


---
4. Find all the words that are in the database as the two related forms XOG and XOGUE.
For example, CATALOG and CATALOGUE have this pattern. [2 mark]

```{r}
o <- grep("OGUE$|OG$", urs.v4, val=TRUE)
og <- grep("OG$", urs.v4, val=TRUE)
sort(o)
```
---

5. Read in the file scrabble.txt from which you create a vector called scores
where element i stores the scrabble score of the ith letter of the alphabet. [1 mark]

```{r}
scrabble <- read.delim("scrabble.txt", header = FALSE, sep=" ")
ordered.scrabble <- scrabble[order(scrabble$V1),]
score <- c(ordered.scrabble$V4)
names(score) <- ordered.scrabble$V1

score["P"]
score
```

---
6. Compute the scrabble score for each word in the database. Plot the distribution of scores.
What is the highest-scoring word? [3 marks]

```{r}

score.converter <- function(string, letter.scores=score){
  split.string <- strsplit(string, split="")
  word.score <- 0
  for (n in 1:nchar(string)){
    lscore <- letter.scores[split.string[[1]][n]]
    word.score <- word.score + lscore
  }
  c(string, word.score)
}

scored.words <- sapply(urs.v4, FUN = score.converter)
just.scores <- as.numeric(scored.words[2,])
hist(just.scores, xlim=c(0,50), ylim=c(0,15000))

max(just.scores)

scored.words.df <- data.frame(t(scored.words))
colnames(scored.words.df) <- c("word", "score")
max.word <- scored.words.df[scored.words.df$score==max(just.scores),1]

max.word

```
---

7. The reverse complement of a word is where you reverse the characters in the word,
and then replace A with Z, B with Y, C with X and so on. For example, the reverse complement of HILLY
is BOORS. Find all wordsWwhere bothWand its reverse complement are both in the database.
[2 marks]

```{r}

alphabet <- ordered.scrabble$V1
reverse.ordered.alphabet <- scrabble[order(scrabble$V1, decreasing=TRUE),]$V1
names(alphabet) <-reverse.ordered.alphabet


reverse.complement <- function(string, match.table=alphabet){
  split.string <- strsplit(string, split="")
  rev.string <- rev(split.string[[1]])
  comp.word <- match.table[rev.string]
  comp.word<- as.character(comp.word)
  comp.word <- paste(comp.word, collapse='')
}

comp.rev.dict <- as.character(sapply(urs.v4, FUN=reverse.complement))
comp.rev.dict

words.with.complement <- c()
for (word in urs.v4){
  if (reverse.complement(word) %in% urs.v4){
    words.with.complement <- c(words.with.complement, word)
  }
}
paste0(words.with.complement, collapse=" ")

sort(words.with.complement)
```
---

8. Given the following nine letters:
"F A L U Y P L N I"
how many words of four or more letters can you find that are in the database
AND all contain the letter A? Each letter can be used only once, and you should
be able to find a nine-letter
word. [4 marks]

```{r}

#remove words from database:

bool.size <- sapply(urs.v4, function(x) nchar(x)>=4)
length(urs.v4)
urs.v5 <- urs.v4[bool.size]
length(urs.v5)

bool.A <- sapply(urs.v5, function(x) grepl("A",x))
urs.v6 <- urs.v5[bool.A]
length(urs.v6)

regex <- "^[FALUYPLNI]{4,}$"
urs.v7 <- urs.v6[grepl(regex, urs.v6)]
urs.v7


bool.lengths.L <- lengths(regmatches(urs.v7,gregexpr("L", urs.v7)))<=2
urs.selected <- urs.v7[bool.lengths]
length(urs.selected)

bool.lengths.F <- lengths(regmatches(urs.selected,gregexpr("F", urs.selected)))<=1
urs.selected <- urs.selected[bool.lengths.F]
length(urs.selected)

bool.lengths.A <- lengths(regmatches(urs.selected,gregexpr("A", urs.selected)))<=1
urs.selected <- urs.selected[bool.lengths.A]
length(urs.selected)

bool.lengths.U <- lengths(regmatches(urs.selected,gregexpr("U", urs.selected)))<=1
urs.selected <- urs.selected[bool.lengths.U]
length(urs.selected)

bool.lengths.Y <- lengths(regmatches(urs.selected,gregexpr("Y", urs.selected)))<=1
urs.selected <- urs.selected[bool.lengths.Y]
length(urs.selected)

bool.lengths.P <- lengths(regmatches(urs.selected,gregexpr("P", urs.selected)))<=1
urs.selected <-urs.selected[bool.lengths.P]
length(urs.selected)

bool.lengths.N <- lengths(regmatches(urs.selected,gregexpr("N", urs.selected)))<=1
urs.selected <- urs.selected[bool.lengths.N]
length(urs.selected)

bool.lengths.I <- lengths(regmatches(urs.selected,gregexpr("I", urs.selected)))<=1
urs.selected <-urs.selected[bool.lengths.I]
length(urs.selected)

paste0(urs.selected, collapse=" ")
urs.selected
```

```
#Question 2: Examination marking [10 points]

```{r}
correct.answers <- read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/crib.dat", header

sd.1 <- read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/student1.dat", header=T)
sd.2 <- read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/student2.dat", header=T)
sd.3 <- read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/student3.dat", header=T)
sd.4 <- read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/student4.dat", header=T)
sd.5 <- read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/student5.dat", header=T)
sd.6 <- read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/student6.dat", header=T)
sd.7 <- read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/student7.dat", header=T)
sd.8 <- read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/student8.dat", header=T)
sd.9 <- read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/student9.dat", header=T)
sd.10 <-read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/student10.dat", header=T)
sd.11 <- read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/student11.dat", header=T)
sd.12 <- read.delim("/Users/[redacted]/Desktop/rpc2018-master/a1/grading/student12.dat", header=T)

students <- list(sd.1,sd.2,sd.3,sd.4,sd.5,sd.6,sd.7,sd.8,sd.9,sd.10,sd.11, sd.12)

students

find.score.grade <- function(student, right.answers=correct.answers){

  student.raw.mark <- sum(correct.answers[student$qn,]==student$response)
  student.percentage <- floor(student.raw.mark/30*100)

  if (student.percentage <=39){
    grade <- "F"
  }
  if (40 < student.percentage && student.percentage <= 49){
    grade <- "D"
  }
  if (50 <=student.percentage && student.percentage <= 59){
    grade <- "C"
  }
  if(60 <= student.percentage && student.percentage<= 69){
    grade <- "B"
  }
  if(70 <= student.percentage && student.percentage<= 100){
    grade <- "A"
  }
  c(student.raw.mark, grade)
}

results <- matrix(ncol=3, nrow=12)
n<-1
for (sdnt in students){
  sdnt.results <- find.score.grade(sdnt)
  results[n,] <- c(n, sdnt.results[1] ,sdnt.results[2])
  n<-n+1
}
results.df <- as.data.frame(results)

results.df$V2 <- as.numeric(as.character(results.df$V2))

ranking <- order(results.df$V2, decreasing =T)

results.df$rank[ranking] <- 1:nrow(results.df)
```
```

```
colnames(results.df) <- c("Student", "Score", "Grade", "Rank")
results.df
write.csv(results.df, file = "Examination_markings.csv")

```

Cheating?

```{r}
#if cheating distribution of answers chosen will not be ?

querstions.answered <- vector()
for (stdn in students){
  querstions.answered<-c(querstions.answered, stdn$qn)
}
querstions.answered

counts.questions.answerd <- count(querstions.answered)
mean.counts <- mean(counts.questions.answerd$freq)
sd.counts <- sqrt(var(counts.questions.answerd$freq))

words.norm = rnorm(360, mean=3.6)
plot(density(words.norm));plot(density(counts.questions.answerd$freq))

shapiro.test(words.norm); shapiro.test(counts.questions.answerd$freq)

qqnorm(words.norm);qqline(words.norm, col = 2)
qqnorm(counts.questions.answerd$freq);qqline(counts.questions.answerd$freq, col = 2)

hist(querstions.answered, breaks = 100)

save(hist.answers, "hist.answers.png")


#but a non uniform distribution could be due to some questions being
#easier than others so more people answer the independently


#to check would need to do a pairwise comparison of qurstions chosen
#as well as answers chosen. thre will be a staetistical cut off point
#at which ther number of questions/answers being the same will count as cheating

get.same.questions <- function(student.a,student.b){
  sum(student.a$qn %in% student.b$qn)}

same.questions <-outer(students,students,Vectorize(get.same.questions))

same.questions.df
same.questions.df <- data.frame(same.questions)
colnames(same.questions.df) <- c(1:12)

write.csv(same.questions.df, "same.questions.df.csv")
class(same.questions)

##### CHECK ANSWERS
# need to check if answers of student 2 and student 6 were the same

sd.2.common.questions <- sd.2[sd.2$qn %in% sd.6$qn,]
```

```
sd.6.common.question <- sd.6[sd.6$qn %in% sd.2$qn,]
#select all the questions which were answered by both student 2 and 6 and only keep those questions

#order each dataframe by mutual questions -> should result in the same order of questions
ordered.sd.2.common.questions <- sd.2.common.questions[order(sd.2.common.questions$qn),]

ordered.sd.6.common.questions <- sd.6.common.question[order(sd.6.common.question$qn),]
ordered.sd.6.common.questions

#check if the order of the questions is the same
print(paste(sum(ordered.sd.6.common.questions$qn==ordered.sd.2.common.questions$qn),"/27 questions are

#now that questions are in order check if results match:
common.results <- ordered.sd.6.common.questions$response == ordered.sd.2.common.questions$response
print(paste(sum(common.results), "/27 answers are matching"))


```