STAT 3675Q Homework 7

Due date: **Thursday, October 16, at noon**

Note:

- Ensure that your code is fully visible in the PDF and not cropped. If needed, break the code into multiple lines to fit.
- It is recommended to write descriptive answers outside of R code chunks (i.e., as text in the main body), while comments within the code chunks can be reserved for brief code annotations.
- In all homework questions, include a written explanation of any output to earn full credit.

Question 1 [30 points]

Reconsider Question 1 in Homework 6.

```
set.seed(1111)
u <- runif(100)
set.seed(1111)
n <- rnorm(100, mean = 1, sd = 0.5)
set.seed(1111)
e <- rexp(100, rate=2)
```

f. For the three random vectors created in parts b–d, can you get their corresponding sample means, medians and standard deviations simultaneously using `apply()`? Note that you should get the same results as in Question 1. Hint: First combine the three vectors into a data frame.

Answer:

```
u <- runif(100)
n <- rnorm(100, mean = 1, sd = 0.5)
e <- rexp(100, rate = 2)
X <- data.frame(uniform = u, normal = n, exponential = e)

means   <- apply(X, 2, mean)
medians <- apply(X, 2, median)
sds     <- apply(X, 2, sd)
```

```
result <- rbind(mean = means, median = medians, sd = sds)
result
```

```
##          uniform    normal exponential
## mean   0.5441035 1.0089944   0.4714926
## median 0.5357662 0.9621732   0.3082433
## sd     0.3101517 0.4733638   0.5118018
```

g. For the three random vectors created in parts b–d, get the results of `summary()` using `apply()`.

Answer:

```
result <- apply(X, 2, summary)
result
```

```
##             uniform      normal exponential
## Min.    0.005239313 -0.01908875 0.008772313
## 1st Qu. 0.304466996  0.71962676 0.141198792
## Median  0.535766245  0.96217324 0.308243253
## Mean    0.544103472  1.00899438 0.471492568
## 3rd Qu. 0.839658774  1.35091997 0.629924896
## Max.    0.999422777  2.21677854 2.557732397
```

h. For parts b–d, how do you think the differences between the theoretical and sample versions of the statistics will change if you increase the sample size? Verify this numerically using the distribution in part c by first creating the following list:

```
set.seed(1234)
list1 <- list(n1 = rnorm(50, mean = 1, sd = 0.5),
              n2 = rnorm(500, mean = 1, sd = 0.5),
              n3 = rnorm(10000, mean = 1, sd = 0.5))
```

Then use `lapply()` or `sapply()` to calculate the corresponding sample means, medians, and standard deviations. Compare the results to the theoretical mean, median, and standard deviation.

Answer:

```
theory <- c(mean = 1, median = 1, sd = 0.5)
stats_mat <- sapply(list1, function(x) c(mean = mean(x), median = median(x), sd = sd(x)
diff_mat <- sweep(stats_mat, 1, theory, FUN = "-")
abs_err   <- abs(diff_mat)

result <- rbind(
  `sample mean`   = stats_mat["mean", ],
  `error mean-1`  = diff_mat["mean", ],
  `sample median` = stats_mat["median", ],
  `error med-1`   = diff_mat["median", ],
```

```
  `sample sd`     = stats_mat["sd", ],
  `error sd-0.5`  = diff_mat["sd", ]
)
result
```

```
##                        n1         n2           n3
## sample mean    0.77347349 1.02400176  1.003505251
## error mean-1  -0.22652651 0.02400176  0.003505251
## sample median  0.73238501 1.01568259  1.003006882
## error med-1   -0.26761499 0.01568259  0.003006882
## sample sd      0.44252176 0.51317652  0.493284459
## error sd-0.5  -0.05747824 0.01317652 -0.006715541
```

Question 2 [20 points]

Reconsider the Forbes Global 2000 data.

a. Import the original data with 2000 observations. For the website addresses, replace "http://www.forbes.com/companies" with "~" (hint: use the function `sub()`) and store the shortened website addresses in a new variable named **newWebsite**. Display the first 6 new addresses.

Answer:

```
forbes <- read.csv("Forbes Global 2000.csv")

forbes$newWebsite <- sub(
  "http://www.forbes.com/companies",
  "~",
  forbes$`Forbes.Webpage`
)

head(forbes$newWebsite)
```

```
## [1] "~/icbc/"                       "~/china-construction-bank/"
## [3] "~/agricultural-bank-of-china/" "~/jpmorgan-chase/"
## [5] "~/berkshire-hathaway/"         "~/exxon-mobil/"
```

b. Set the random seed to 1234. Randomly draw a sample of size 10 from the ForbesGlobal2000 data. Extract all columns with character data. Find the number of characters for each entry in all these columns using `apply()`.

Answer:

```
set.seed(1234)
idx  <- sample(nrow(forbes), 10)
samp <- forbes[idx, , drop = FALSE]
is_charlike <- sapply(samp, function(x) is.character(x) || is.factor(x))
```

```
char_df <- data.frame(lapply(samp[, is_charlike, drop = FALSE], as.character),
                      check.names = FALSE, stringsAsFactors = FALSE)
char_len <- apply(as.matrix(char_df), c(1, 2), nchar)
char_len
```

```
##       Company Sector Industry Continent Country Forbes.Webpage newWebsite
##  [1,]      12      9       21         4       8             45         15
##  [2,]      12     10       14         4       5             45         15
##  [3,]      21     26       27         4       8             54         24
##  [4,]       5     10       11         6       6             38          8
##  [5,]      19     10       14        13      13             52         22
##  [6,]       5      9       22         9       9             38          8
##  [7,]      15     10       14         6       6             48         18
##  [8,]       8     26       27         6       7             41         11
##  [9,]       3      9       18         6       6             49         19
## [10,]      28     10       11         6       7             61         31
```

Question 3 [50 points]

Reconsider Question 3 in Homework 6.

```
txt <- readLines("shortstory.txt")
txt <- paste(txt, collapse = "") # line 1
txt <- tolower(txt) # line 2
txt <- unlist(strsplit(txt,"")) # line 3
```

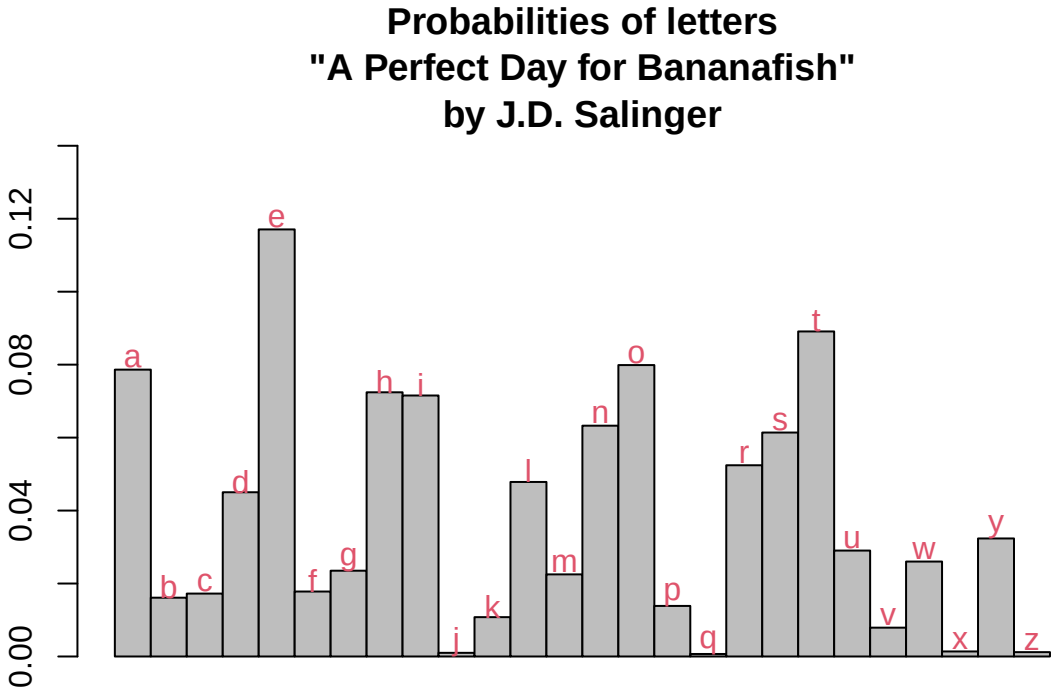
c. Note that R has a built-in vector called **letters** which contains all the letters from a-z (hint: try typing `letters` in the console). What is the goal of the following code?

```
counts <- rep(0,26)
for (i in 1:length(letters)) {
  counts[i] <- length(grep(letters[i], txt))
}
```

Answer: The goal of this code is to count how many times each letter (a–z) appears in the text. It creates a vector counts of length 26 and, for each letter in the built-in vector letters, uses grep() to find all occurrences of that letter in the text txt. The resulting counts vector therefore contains the frequency of each letter in the story.

d. Briefly comment on what line 1 of the following code does and what you observe in the plot below.

```
probs <- counts/sum(counts) # line 1
barplot(probs, ylim=c(0,0.14), space=0)
title("Probabilities of letters\n\"A Perfect Day for Bananafish\"\nby J.D. Salinger")
text(1:26-0.5,probs+0.003,letters, col=2)
```



Answer: Line 1 converts the raw letter counts into probabilities by dividing each count by the total number of letters, producing the relative frequency of each letter in the text. The bar plot shows that common English letters such as e, t, a, o, and n appear most frequently, while q, x, and z are rare. This pattern matches general English letter frequency distributions.

e. In an alphabet consisting of n characters, $x_1, \ldots, x_n$, whose probabilities of occurrence are $p_1, \ldots, p_n$, respectively, the Shannon entropy is defined as

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

Entropy is a measure of unpredictability of information content - the greater the entropy, the less predictable the content. Write a function called `entropy` that computes the Shannon entropy given the probabilities. The input argument is a vector called `probs` whose ith element is the probability (i.e., relative frequency) that the ith letter occurs in the text.

Answer:

```
entropy <- function(probs) {
  probs <- probs[probs > 0]
  H <- -sum(probs * log2(probs))
  return(H)
}
```

f. Calculate the Shannon entropy of the short story in this question. Hint: Use what you created in parts d and e.

Answer:

```r
H <- entropy(probs)
H_max <- log2(26)
H_rel <- H / H_max
c(H = H, H_max = H_max, Relative_Entropy = H_rel)
```

```
##                 H             H_max Relative_Entropy
##         4.1820336         4.7004397         0.8897111
```

g. Suppose that a text consists of 26 characters drawn **uniformly** from the English alphabet (a-z), i.e., the `probs` vector is `rep(1,26)/26`. Calculate its Shannon entropy. Comparing it to the result in part f, what can you say about the short story in this question?

Note: This question does not ask you to conduct the random sampling. Instead, you only need to apply the `entropy` function to the `probs` vector and interpret the results.

Answer:

```r
probs_uniform <- rep(1, 26) / 26
H_uniform <- entropy(probs_uniform)
H_uniform
```

```
## [1] 4.70044
```

"The Shannon entropy of a uniform distribution over 26 letters is 4.70 bits. In comparison, the short story's entropy from part (f) is slightly lower. This means the story's letter distribution is not perfectly uniform—some letters (like e, t, a, o) appear much more frequently than others (q, x, z). Therefore, the text of A Perfect Day for Bananafish contains less uncertainty and more structure than a completely random sequence of letters, which is characteristic of natural English language."