Homework 4

Statistical Computing, STAT 3675Q

Anthony Anderson

General Instructions

- Answer the questions by inserting R code and necessary comments. Your output must contain the R code (do not use the `echo=FALSE` option).
- After you complete the assignment, save it under the file name `LastName-FirstName-HW4.pdf`
- Then submit the compiled PDF file through HuskyCT by **March 5, 2019, at 11:59 pm**.

Question 1 [15 points]

A Pythagorean triple is a set of three positive integers a, b, and c, such that $a^2 + b^2 = c^2$. For example, (3,4,5) and (5,12,13) are Pythagorean triples.

Write a function that prints all the Pythagorean triples with $a \leq b < c$ and with $c < n$ where n is provided by the user. Your output should not include any duplicates. For example, (4,3,5) should not be included in your output. The function has to print the correct output, **and** it has to return the number of triples which were found.

```
PythTriple <- function(n) {
# Gives all sets of Pythagorean triples less than n
# n is a positive integer
 count <- 0
  for(a in 1:n) {
   for(b in a:n) {
     c=sqrt(a^2+b^2)
     if(c==round(c) & c<n) {
       cat(a,b,c, "\n")
       count <- count+1
     }

   }
 }
  return(count)



}
triplesFound <- PythTriple(141)
```

```
3 4 5
5 12 13
6 8 10
7 24 25
8 15 17
9 12 15
9 40 41
10 24 26
11 60 61
12 16 20
12 35 37
```

13 84 85
14 48 50
15 20 25
15 36 39
15 112 113
16 30 34
16 63 65
18 24 30
18 80 82
20 21 29
20 48 52
20 99 101
21 28 35
21 72 75
22 120 122
24 32 40
24 45 51
24 70 74
25 60 65
27 36 45
27 120 123
28 45 53
28 96 100
30 40 50
30 72 78
32 60 68
32 126 130
33 44 55
33 56 65
35 84 91
35 120 125
36 48 60
36 77 85
36 105 111
39 52 65
39 80 89
40 42 58
40 75 85
40 96 104
42 56 70
44 117 125
45 60 75
45 108 117
48 55 73
48 64 80
48 90 102
50 120 130
51 68 85
54 72 90
56 90 106
56 105 119
57 76 95
60 63 87
60 80 100

```
60 91 109
63 84 105
64 120 136
65 72 97
66 88 110
66 112 130
69 92 115
72 96 120
75 100 125
78 104 130
80 84 116
81 108 135
84 112 140
88 105 137
```

```
# print the number of triples that you found:
triplesFound
```

```
[1] 79
```

Question 2 [15 points]

In an alphabet consisting of n characters, $x_1, \ldots, x_n$, the Shannon entropy of a given text is defined as

$$H = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

where $P(x_i)$ is the probability that letter x_i occurs in the text. Entropy is a measure of unpredictability of information content - the greater the entropy, the less predictable the content.

The file `shortstory.txt` contains the short story "A Perfect Day for Bananafish", by J.D. Salinger.

a) Use the function `readLines` to read the content of the file into a variable in R (you don't have to read the story, but you may enjoy it.)
b) Count the total number of times each letter (a-z) appears in the story.
c) Calculate the probability of occurrence of each letter, and generate a plot using the probabilities. Briefly comment on what you observe in the plot.
d) Calculate the Shannon entropy of the story.
e) Calculate the Shannon entropy of a text which consists of characters drawn uniformly from the English alphabet (a-z). Comment on what you find in (d) and (e).

Notes and hints:

i) Convert all characters to lower case, and ignore any character which is not between a and z.
ii) R contains a built-in vector called **letters** which contains all the letters from a-z. You may find it useful.
iii) The instructions for how to plot the graph in part (c) are deliberately vague. Be creative!

```
story <- readLines("shortstory.txt")
```

```
## Warning in readLines("shortstory.txt"): incomplete final line found on
## 'shortstory.txt'
```

```
story <- paste(story, collapse="")
story <- tolower(story)
story <- strsplit(story,"")
Vectorstory <- unlist(story)
```

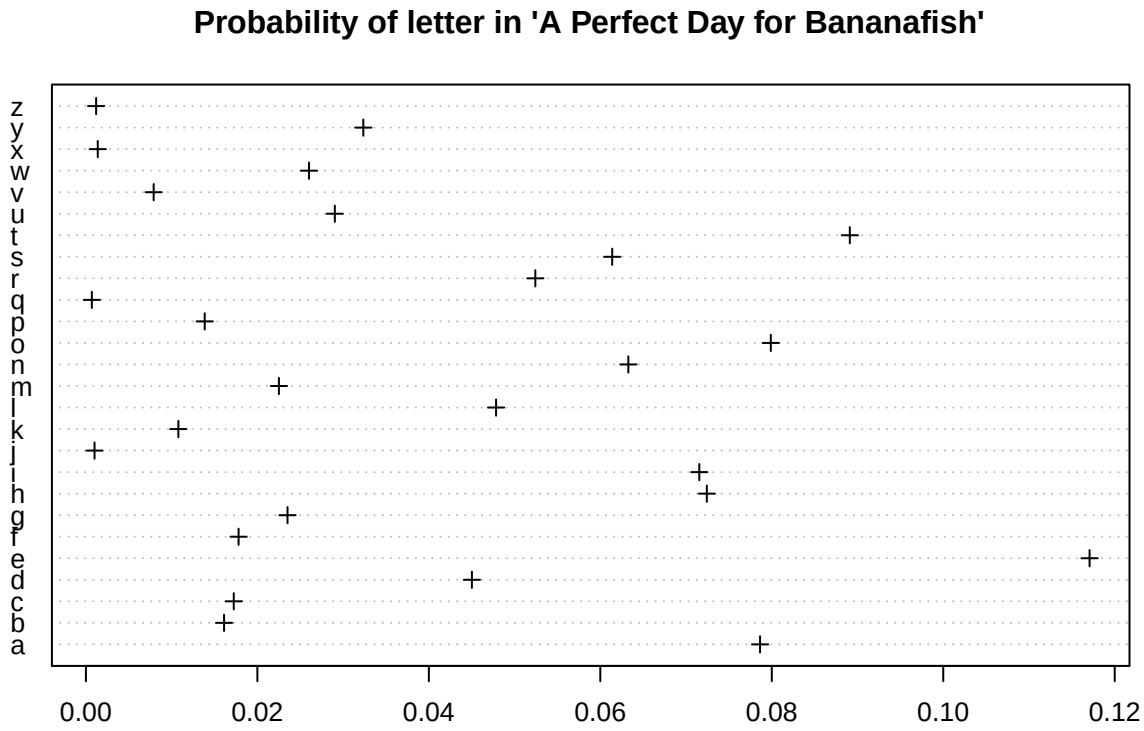
```
countletters <- rep(0,26)
for (i in 1:26) {
  countletters[i]<-  length(grep(letters[i], Vectorstory))

}
countletters
```

```
##  [1] 1254  257  275  718 1867  284  375 1155 1141   16  172  763  359 1009
## [15] 1274  221   11  836  979 1421  463  126  415   22  516   19
```

```
prob<- countletters/sum(countletters)

dotchart(prob, labels=letters, main="Probability of letter in 'A Perfect Day for Bananafish'",
         cex=0.8, pch=3)
```




```
entropy<- -sum(prob* log(prob, base=2))
entropy
```

```
## [1] 4.182034
```

```
unifL <- rep(1/26, 26)
entropyunif <- -sum(unifL*log(unifL,base=2))
entropyunif
```

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

Letter “e” is the most common letter, which isn’t too surprising because it’s a vowel, but “t” comes in second, which I didn’t expect.

There is also a difference between the entropy of the story and a uniform sample, the story is actually less "random" than the uniform sample, it has lower entropy.

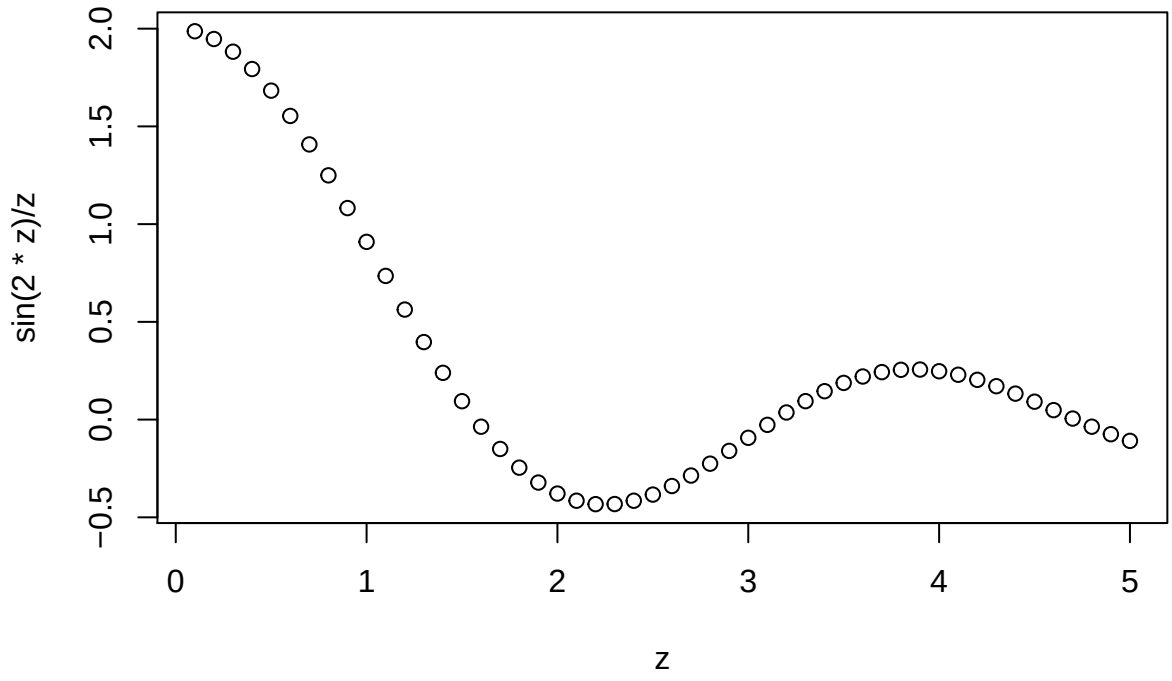
Question 3 [15 points]

Please read about the "secant method" in order to answer this question:

https://en.wikipedia.org/wiki/Secant_method

a) Plot the function $f(x) = sin(2x)/x$ for $x \in (0, 5)$
b) Implement the secant method in R for the function $f(x)$. Use as many iterations as needed until $|f(x_n)| < 10^{-6}$.
c) Run the secant method for $f(x)$ using the initial values $x_0 = 0.5$ and $x_1 = 0.7$. How many steps were needed? What is the solution obtained by the method?
d) Draw a plot which includes the curve, and implement a neat way to mark the points $(x_n, f(x_n))$ on the graph.
e) Run it again using different initial values to obtain another solution, and include a similar plot.

```
z<- seq(0.1,5, .1)
plot(z, sin(2*z)/z)
```




```
fx<- function(x) {
  sin(2*x)/x
}

Secant <- function(f,x0,x1, tolerance=1e-06, n=100) {
  count <- 0
 for(i in 1:n) {
```

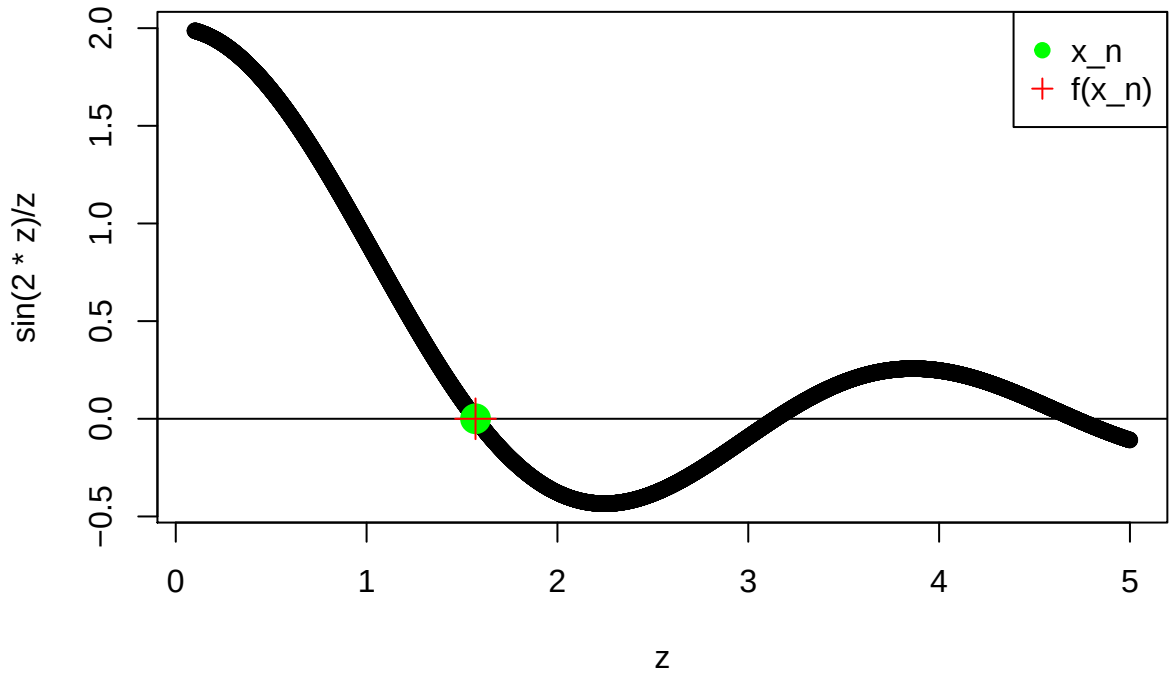
```
   x2 <- x1-fx(x1)*(x1-x0)/(fx(x1)-f(x0))
    if(abs(fx(x2))<tolerance) {
      return(x2)
    }
      x0 <- x1
      x1 <- x2
      count <- count+1
 }
  cat("Unsuccessful, try increasing iterations","\n")
}
Secant(fx,.5,.7, n=4)
```

```
## Unsuccessful, try increasing iterations
```

```
Secant(fx,.5,.7, n=5)
```

```
## [1] 1.570797
```

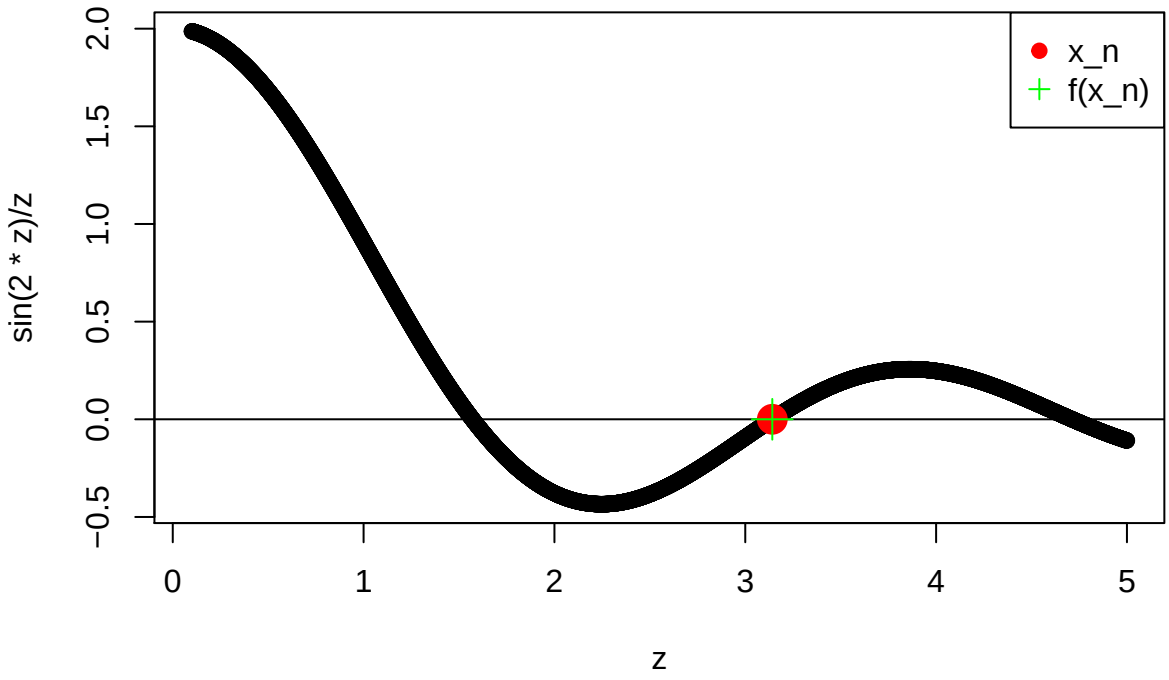
```
z<- seq(.1,5, .001)
plot(z,sin(2*z)/z)
abline(h=0)
points(Secant(fx,.5,.7),0,col="green", cex=2, pch=19)
points(Secant(fx,.5,.7),fx(Secant(fx,.5,.7)),col="red", cex=2, pch=3)
legend("topright", c("x_n","f(x_n)"), pch=c(19,3), col=c("green","red"))
```




```
Secant(fx,.1,.5)
```

```
## [1] 3.141594
```

```
z<- seq(.1,5,.001)
plot(z,sin(2*z)/z)
abline(h=0)
points(Secant(fx,.1,.5),0,col="red", cex=2, pch=19)
points(Secant(fx,.1,.5),fx(Secant(fx,.1,.5)),col="green", cex=2, pch=3)
legend("topright", c("x_n","f(x_n)"), pch=c(19,3), col=c("red","green"))
```



At 4 iterations, the root is not found, but at a minimum of 5 is it. The output is the found root, in this case 1.57

The plots show that for the given initial values and number of iterations in this case, the calculated root via the Secant Method is the same as the real root. This is not always true, depending on the initial values the calculated root can be very off. Here, I used .1, .5 as the initial values in the second plot, which are not close to the calculated root of 3.14 (pi) but it still was correct.