Homework 1

Alexander Hernandez

09/08/2022

1) Calculate the following numerical results to the three decimal places

a)

```
log(3, base=exp(1)) + sqrt(2)*sin(pi) - exp(3)
```

```
## [1] -18.98692
```

```
# The natural log is used here instead of the log function, which requires base e to be used.
```

b)

```
2 * (5+3) - sqrt(6) + 9^2
```

```
## [1] 94.55051
```

```
# Simple functions are automatically done in order of operations by R, resulting in the results
```

c)

```
log(5, base=exp(1)) - exp(2) + 2^3
```

```
## [1] 2.220382
```

```
# Another example of the natural log with an e value used
```

d)

```
(9/2) * 4 - sqrt(10) + log(6, base=exp(1)) - exp(2)
```

```
## [1] 9.240426
```

```
# Similar to the last problem with more complex PEMDAS
```

e)

```
log(14,base=10) + log(14,base=exp(1)) + 47%%5
```

```
## [1] 5.785185
```

```
# log base 10 and natural log are used here, which require specifying in their method
```

2) Create the following vectors using rep function:

v1) V1 = 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

```
V1 = rep(c(1,2,3,4,5), 5)
V1
```

```
##  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

```
# default argument following the vector replicated the vector the specified amount of times
```

v2) V2= 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 6 6 6 6

```
V2 = rep(c(1,2,3,4,5,6), each=4)
V2
```

```
##  [1] 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 6 6 6 6
```

```
# the each function allows the separate values repeated, instead of the entire vector
```

v3) V3= MATH, MATH, STAT, STAT, STAT, STAT, STAT, ECE, ECE, ECE, BIO, BIO

```
V3 = rep(c("MATH", "STAT", "ECE", "BIO"), times=c(2,5,3,2))
V3
```

```
##  [1] "MATH" "MATH" "STAT" "STAT" "STAT" "STAT" "STAT" "ECE"  "ECE"  "ECE"
## [11] "BIO"  "BIO"
```

```
# the times function allows choosing which variables to repeat
```

3) Use data from "What Does it Take to Heat a New Room?"

http://jse.amstat.org/datasets/utility.dat.txt ## a) Import the data in R

```
utility_data = read.table('http://jse.amstat.org/datasets/utility.dat.txt')
# Can import webdata directly into R
```

b) How many variables are included in this dataset?

```
ncol(utility_data)
```

```
## [1] 13
```

```
# Variables are sorted by the number of columns
```

c) The missing values in this dataset are denoted by *. Remove them.

```
new_utility_data = utility_data[
  (utility_data$V1 != '*') &
  (utility_data$V2 != '*') &
  (utility_data$V3 != '*') &
  (utility_data$V4 != '*') &
  (utility_data$V5 != '*') &
  (utility_data$V6 != '*') &
```

```
  (utility_data$V7 != '*') &
  (utility_data$V8 != '*') &
  (utility_data$V9 != '*') &
  (utility_data$V10 != '*') &
  (utility_data$V11 != '*') &
  (utility_data$V12 != '*') &
  (utility_data$V13 != '*'),]

new_utility_data
```

```
##          V1 V2 V3   V4  V5 V6  V7   V8 V9 V10  V11 V12 V13
## 1  Sep-90 30 62  0.8  24 30 432 14.4 30   0  128  48   0
## 2  Oct-90 31 56  2.1  61 29 469 15.6 30   1  299  26   0
## 3  Nov-90 30 45  4.9 159 32 339 10.6 32   0  603   3   0
## 4  Dec-90 31 37  6.1 185 30 408 14.1 29   1  866   0   0
## 5  Jan-91 31 27  8.3 275 33 658 21.9 30   0 1171   0   0
## 6  Feb-91 28 33  8.5 247 29 627 20.2 31   1  889   0   0
## 7  Mar-91 31 39  6.6 186 28 343 11.8 29   0  798   0   0
## 8  Apr-91 30 50  7.0 203 29 399 13.8 29   1  461   0   0
## 9  May-91 31 62  2.5  73 29 503 16.8 30   0  147  64   0
## 10 Jun-91 30 67  0.2   7 32 440 13.8 32   1   69 124   0
## 11 Jul-91 31 72  0.4  14 31 230  7.7 30   0    6 234   0
## 12 Aug-91 31 71  1.2  37 29 374 11.3 33   1   12 208   0
## 14 Oct-91 31 54  2.3  71 30 365 13.0 28   1  333   5   0
## 15 Nov-91 30 42  4.1 119 29 520 15.8 33   0  675   0   0
## 16 Dec-91 31 39  6.2 201 32 524 18.1 29   1 1004   0   0
## 17 Jan-92 31 28  8.8 273 31 675 19.3 35   0 1143   0   0
## 18 Feb-92 28 29 10.5 336 32 469 16.2 29   1 1033   0   0
## 19 Mar-92 31 33  8.7 253 29 490 16.3 30   0  998   0   0
## 20 Apr-92 30 44  6.9 202 29 443 15.3 29   1  624   0   0
## 21 May-92 31 55  3.2 104 32 383 12.0 32   0  333  36   0
## 22 Jun-92 30 65  1.8  54 30 313 10.8 29   1   64  77   0
## 23 Jul-92 31 68  0.8  28 33 331 10.0 33   0   30 123   0
## 24 Aug-92 31 68  1.1  33 29 379 12.6 30   1   29 135   0
## 26 Oct-92 31 49  2.7  88 32 464 14.5 32   1  482   0   0
## 27 Nov-92 30 40  5.2 153 29 181  6.0 30   0  756   0   0
## 28 Dec-92 31 36  6.8 204 30 561 19.3 29   1 1043   0   0
## 29 Jan-93 31 28  8.8 293 33 529 15.1 35   0 1146   0   0
## 30 Feb-93 28 23 11.0 321 29 455 16.3 28   1 1184   0   0
## 31 Mar-93 31 33 10.8 315 29 506 16.9 30   1  996   0   0
## 32 Apr-93 30 46  6.1 179 29 420 13.5 31   1  567   0   0
## 33 May-93 31 59  3.2 104 32 529 17.6 30   0  204  12   0
## 34 Jun-93 30 66  1.7  53 30 311 10.4 30   1   61 108   0
## 36 Aug-93 31 72  0.8  53 62 650 22.4 29   1    2 233   0
## 38 Oct-93 31 50  1.9 123 62 312 10.4 30   1  474   1   0
## 39 Nov-93 30 43  4.6 135 29 544 19.4 28   0  665   0   0
## 40 Dec-93 31 36  6.8 204 30 607 17.9 34   1 1054   0   0
## 41 Jan-94 31 33  8.0 248 31 534 17.2 31   0 1321   0   0
## 42 Feb-94 28 17 12.2 342 28 573 20.5 28   1 1059   0   0
## 43 Mar-94 31 29  8.8 264 30 529 17.6 30   1  827   0   0
## 44 Apr-94 30 44  5.0 163 32 522 16.8 31   1  403   0   0
## 45 May-94 31 51  2.8  82 29 236  7.6 31   0  226   0   0
## 46 Jun-94 30 60  1.6  49 30 403 12.6 32   1    7 222   0
## 48 Aug-94 31 70  0.7  49 62 377 12.6 30   1    3 241   0
```

```
## 50 Oct-94 31 57  1.7 108 62  443 14.8 30   1  288   0   0
## 51 Nov-94 30 51  2.8  83 29  397 14.2 28   0  478   0   0
## 52 Dec-94 31 35  6.5 209 32  606 18.9 32   1  814   0   0
## 53 Jan-95 31 18 11.4 379 33  587 20.2 29   0  932   0   0
## 54 Feb-95 28 22 10.7 301 28  567 19.6 29   1 1016   0   0
## 55 Mar-95 31 32  8.5 273 32  563 18.2 31   0  805   0   0
## 56 Apr-95 30 40  5.9 178 30  486 16.8 29   1  561   0   0
## 57 May-95 31 51  2.9  85 29  554 18.5 30   0  258  25   0
## 58 Jun-95 30 69  1.3  44 32  294 11.9 25   1   30 147   0
## 60 Aug-95 31 73  0.7  47 60  453 15.6 29   1    2 252   0
## 62 Oct-95 31 59  1.5  96 61  557 18.0 31   1  214  15   0
## 63 Nov-95 30 46  4.5 132 29  417 14.9 28   0  685   0   0
## 64 Dec-95 31 29  8.9 285 32  579 18.1 32   1 1023   0   1
## 65 Jan-96 31 30 11.6 361 31  543 18.1 30   1 1074   0   1
## 66 Feb-96 29 31 10.7 311 29  546 18.8 29   1  981   0   1
## 67 Mar-96 31 37 11.6 372 32 1248 37.8 33   0  875   0   1
## 68 Apr-96 30 48  7.5 226 30  494 17.6 28   1  510   3   1
## 69 May-96 31 57  3.5 104 29  520 17.9 29   0  264  30   1
## 70 Jun-96 30 68  1.5  48 32  443 13.0 34   1   20 121   1
## 72 Aug-96 31 71  0.8  50 62  521 17.4 30   1    6 196   1
## 74 Oct-96 31 53  1.9 116 60  512 17.7 29   1  358   0   1
## 75 Nov-96 30 40  5.0 158 31  736 22.3 33   0  739   1   1
## 76 Dec-96 31 39  7.3 219 30  600 20.7 29   1  792   0   1
## 77 Jan-97 31 29  9.3 307 33 1115 32.8 34   0 1104   0   1
## 78 Feb-97 28 36  9.7 283 29  853 30.5 28   1  806   0   1
## 79 Mar-97 31 37  7.9 230 29  713 24.6 29   0  868   0   1
## 80 Apr-97 30 46  5.8 171 29  498 17.2 29   1  551   0   1
## 81 May-97 31 56  3.2 104 32  838 26.2 32   0  269   0   1
```

```
# This looks cumbersome, but it goes through every variable and, if it sees an 'asterisk/*',
# removes that line from the data.
```

4) Extract 2004 (2nd tab) from the CPSS.XLS data and determine dimension

```
library('readxl')
pop_survey_file = file.choose()
pop_survey = read_excel(pop_survey_file, sheet='CPSSW4')
```

```
## New names:
## * `` -> `...1`
```

```
dim(pop_survey)
```

```
## [1] 7986    5
```

```
# since there are two cases, the sheet must be specified in the read_excel function after
# specifying path
```

5) Use R to solve the following system of equations:

```
C = matrix(c(2,1,2,1,1, 1,-1,1,-3,2, 1,2,-1,1,-1, -3,1,2,2,3, 1,-1,1,-1,-1), nrow=5, ncol=5)
Y = matrix(c(12,1,-2,-9,0), nrow=5, ncol=1)
```

```r
C
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    1    1   -3    1
## [2,]    1   -1    2    1   -1
## [3,]    2    1   -1    2    1
## [4,]    1   -3    1    2   -1
## [5,]    1    2   -1    3   -1
```

```r
Y
```

```
##      [,1]
## [1,]   12
## [2,]    1
## [3,]   -2
## [4,]   -9
## [5,]    0
```

```r
D = solve(C,Y)
D
```

```
##      [,1]
## [1,]    1
## [2,]    3
## [3,]    2
## [4,]   -2
## [5,]   -1
```

```r
# Taking the values of the variables in the system of equations forms a matrix.
# Matrices are created in a top to bottom, left to right fashion, with rows and cols specified.
# Another matrix takes the right side of the system and solve takes both to solve it.
```

6) Print the first 50 numbers of the fibbonochi sequence

```r
Fibonacci <- numeric(50)
Fibonacci[1] <- Fibonacci[2] <- 1
for (i in 3:length(Fibonacci)) Fibonacci[i] <- Fibonacci[i-2] + Fibonacci[i-1]
Fibonacci
```

```
##  [1]           1           1           2           3           5           8
##  [7]          13          21          34          55          89         144
## [13]         233         377         610         987        1597        2584
## [19]        4181        6765       10946       17711       28657       46368
## [25]       75025      121393      196418      317811      514229      832040
## [31]     1346269     2178309     3524578     5702887     9227465    14930352
## [37]    24157817    39088169    63245986   102334155   165580141   267914296
## [43]   433494437   701408733  1134903170  1836311903  2971215073  4807526976
## [49]  7778742049 12586269025
```

```r
# The function used was provided in the hint to question 6. Simple changing the vector to 50
# length and repeating the process until 50 gets all 50 numbers.
```

7) Test scores of Fifteen students in Test 1 and Test 2:

```
sn = c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)
test1 = c(56,78,87,89,95,98,NA,78,87,98,54,89,78,98,97)
test2 = c(86,67,78,89,87,67,94,78,81,83,78,NA,93,98,100)

df1 = data.frame(sn, test1,test2)
df1
```

```
##    sn test1 test2
## 1   1    56    86
## 2   2    78    67
## 3   3    87    78
## 4   4    89    89
## 5   5    95    87
## 6   6    98    67
## 7   7    NA    94
## 8   8    78    78
## 9   9    87    81
## 10 10    98    83
## 11 11    54    78
## 12 12    89    NA
## 13 13    78    93
## 14 14    98    98
## 15 15    97   100
```

a) How many students have their test 1 score greater than 80 ?

```
nrow(df1[df1$test1>80 & !is.na(df1$test1),])
```

```
## [1] 9
```

```
# This function gets all test1 scores above 80 and removes instances of 'NA'.
# nrow counts how many students
```

b) How many students have their test 2 score greater than 85 ?

```
nrow(df1[df1$test2>85 & !is.na(df1$test2),])
```

```
## [1] 7
```

```
# This function gets all test2 scores above 85 and removes instances of 'NA'.
# nrow counts how many students
```

c) Did all fifteen students take both tests?

```
print("Which student(s) did not take test 1:")
```

```
## [1] "Which student(s) did not take test 1:"
```

```
which(is.na(df1$test1))
```

```
## [1] 7
```

```
print("Which student(s) did not take test 2:")
```

```
## [1] "Which student(s) did not take test 2:"
```

```
which(is.na(df1$test2))
```

```
## [1] 12
```

```
# student 7 did not take test 1 and student 12 did not take test 2
```

d) How many students did better in the second test than the first test?

```
nrow(df1[(df1$test2 > df1$test1) & !is.na(df1$test2) ,])
```

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

```
# Conditional that selects results with greater test2 than test 1 and
# remove 'NA' results of test2
# Any score is higher than not taking test1 so student 7 is included.
```

e) How many students have the same score in the first and second test?

```
nrow(df1[(df1$test2==df1$test1) & !is.na(df1$test1) & !is.na(df1$test2) ,])
```

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

```
# Checks if the first and second scores are the same, while ignoring 'NA'
```

8) Create the following matrix with column and row names

```
M = matrix(c(1:20), nrow=4, ncol=5)
rownames(M) = c('Experiment.1','Experiment.2','Experiment.3','Experiment.4')
colnames(M) = c('column-1','column-2','column-3','column-4','column-5')
M
```

```
##              column-1 column-2 column-3 column-4 column-5
## Experiment.1        1        5        9       13       17
## Experiment.2        2        6       10       14       18
## Experiment.3        3        7       11       15       19
## Experiment.4        4        8       12       16       20
```

```
# A spread of numbers 1:20 works can make this matrix. Rownames and
# colnames can be used to set them
```

a) Determine the dimension of the matrix M

```
dim(M)
```

```
## [1] 4 5
```

```
# dim function tells the number of rows and cols
```

b) Select the first two row of the matrix M

```
M[0:2,]
```

```
##              column-1 column-2 column-3 column-4 column-5
## Experiment.1        1        5        9       13       17
```

```
## Experiment.2        2        6       10       14       18
# Slice takes the entered number of rows first (and an optional columns)
```

c) Calculate the sum of all columns of the matrix M

```
colSums(M)
```

```
## column-1 column-2 column-3 column-4 column-5
##       10       26       42       58       74
```

```
# colSums sums the vertical numbers of a column
```

d) Calculate the sum of all rows of the matrix M

```
rowSums(M)
```

```
## Experiment.1 Experiment.2 Experiment.3 Experiment.4
##           45           50           55           60
```

```
# rowSums calculates the horizontal sum of a row
```

e) Use "sample" to shuffle the elements of each row of the matrix M

```
shuffled_M = apply(M, 1, sample)
shuffled_M
```

```
##      Experiment.1 Experiment.2 Experiment.3 Experiment.4
## [1,]           17            2           15           12
## [2,]           13           18           19           16
## [3,]            9           10           11            8
## [4,]            1           14            3           20
## [5,]            5            6            7            4
```

```
#This successfully applies the shuffling to the rows in the matrix and
# is correct, but apply rotates the matrix for some reason
```