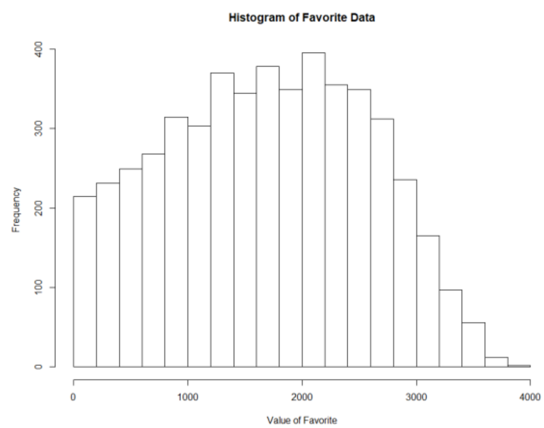
Jared Yu

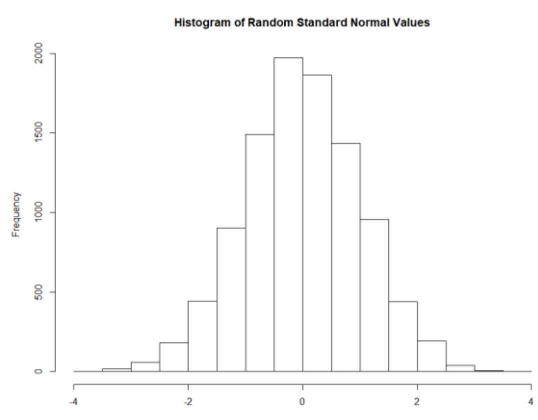
Computational Statistics

Problem Set 1

1. The following are the values for problem 1. The code is in the appendix.
    a. The Mean: 1688.517
    b. The Median: 1706
    c. The Standard Deviation: 833.4741
    d. The minimum value: 2
    e. The maximum value: 3907
    f. A histogram of the data:



2. Using R, 10,000 values were randomly generated from a standard normal distribution with seed set to 1. The code is in the appendix.
    a. A histogram of the data



    b. Mean: -0.006537039, Median: -0.01592883, Standard Deviation: 1.012356
3. Both vectors 'a' and 'b' were generated using the seq() function. The code is in the appendix.
    a. 15^{th} element: 5,475, 16^{th} element: 5,760, 17^{th} element: 6,035

b. The following are the elements of ‘d’ that are greater than 2,000: 2075, 2460, 2835, 3200, 3555, 3900, 4235, 4560, 4875, 5180, 5475, 5760, 6035, 6300, 6555, 6800, 7035, 7260, 7475, 7680, 7875, 8060, 8235, 8400, 8555, 8700, 8835, 8960
c. There are 16 elements in ‘d’ that are larger than 6,000.

4. The function that was created is called ‘sum_perfect_squares’ and it works by first sequencing the numbers from 1 to x. Then it takes the square root of the array, leaving integers and decimals. The decimals are filtered out by using the modulo operator and finally the remaining values are summed. The code is in the appendix.
   a. The sum of all the perfect squares between 1 and $x = 100$: 385
   b. The sum of all the perfect squares between 1 and $x = 100{,}000$: 10,568,146
5. The function that was created is called ‘find_perfect_squares’ and it works like the function in problem 4, except it does not sum the numbers and returns the array first instead. The code is in the appendix.
   a. The following are the numbers from 1 to 500 that are perfect squares: `1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361 400 441 484`
   b. Below is the matrix of the data when the input is set to by.row=FALSE

```
      [,1]  [,2]  [,3]  [,4]
 [1,]    1  6400 25281 56644
 [2,]    4  6561 25600 57121
 [3,]    9  6724 25921 57600
 [4,]   16  6889 26244 58081
 [5,]   25  7056 26569 58564
 [6,]   36  7225 26896 59049
 [7,]   49  7396 27225 59536
 [8,]   64  7569 27556 60025
 [9,]   81  7744 27889 60516
[10,]  100  7921 28224 61009
[11,]  121  8100 28561 61504
[12,]  144  8281 28900 62001
[13,]  169  8464 29241 62500
[14,]  196  8649 29584 63001
[15,]  225  8836 29929 63504
[16,]  256  9025 30276 64009
```

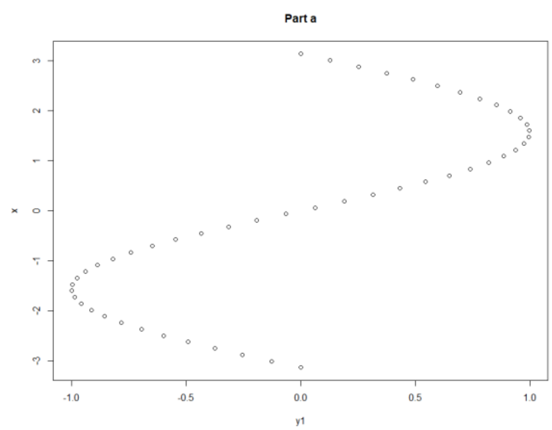
```
[17,]  289  9216 30625 64516
[18,]  324  9409 30976 65025
[19,]  361  9604 31329 65536
[20,]  400  9801 31684 66049
[21,]  441 10000 32041 66564
[22,]  484 10201 32400 67081
[23,]  529 10404 32761 67600
[24,]  576 10609 33124 68121
[25,]  625 10816 33489 68644
[26,]  676 11025 33856 69169
[27,]  729 11236 34225 69696
[28,]  784 11449 34596 70225
[29,]  841 11664 34969 70756
[30,]  900 11881 35344 71289
[31,]  961 12100 35721 71824
[32,] 1024 12321 36100 72361
[33,] 1089 12544 36481 72900
[34,] 1156 12769 36864 73441
[35,] 1225 12996 37249 73984
[36,] 1296 13225 37636 74529
[37,] 1369 13456 38025 75076
[38,] 1444 13689 38416 75625
[39,] 1521 13924 38809 76176
[40,] 1600 14161 39204 76729
[41,] 1681 14400 39601 77284
[42,] 1764 14641 40000 77841
[43,] 1849 14884 40401 78400
[44,] 1936 15129 40804 78961
```

```
[45,] 2025 15376 41209 79524
[46,] 2116 15625 41616 80089
[47,] 2209 15876 42025 80656
[48,] 2304 16129 42436 81225
[49,] 2401 16384 42849 81796
[50,] 2500 16641 43264 82369
[51,] 2601 16900 43681 82944
[52,] 2704 17161 44100 83521
[53,] 2809 17424 44521 84100
[54,] 2916 17689 44944 84681
[55,] 3025 17956 45369 85264
[56,] 3136 18225 45796 85849
[57,] 3249 18496 46225 86436
[58,] 3364 18769 46656 87025
[59,] 3481 19044 47089 87616
[60,] 3600 19321 47524 88209
[61,] 3721 19600 47961 88804
[62,] 3844 19881 48400 89401
[63,] 3969 20164 48841 90000
[64,] 4096 20449 49284 90601
[65,] 4225 20736 49729 91204
[66,] 4356 21025 50176 91809
[67,] 4489 21316 50625 92416
[68,] 4624 21609 51076 93025
[69,] 4761 21904 51529 93636
[70,] 4900 22201 51984 94249
[71,] 5041 22500 52441 94864
[72,] 5184 22801 52900 95481
```

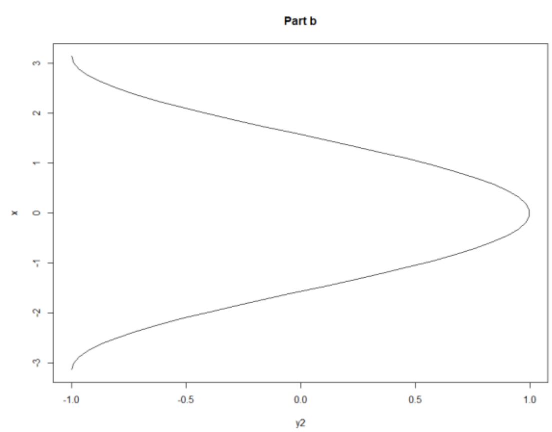
```
[73,] 5329 23104 53361 96100
[74,] 5476 23409 53824 96721
[75,] 5625 23716 54289 97344
[76,] 5776 24025 54756 97969
[77,] 5929 24336 55225 98596
[78,] 6084 24649 55696 99225
[79,] 6241 24964 56169 99856
```

c. The entry of the above matrix for the 15th row and 3rd column is 29,929.

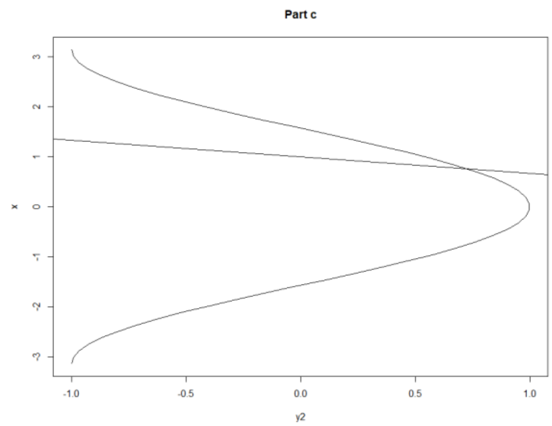
6. The variables 'x', 'y1', and 'y2' were created using the seq(), sin(), and cos() functions. The code is in the appendix.
   a. Below is a plot of y1 vs. x:



   b. Below is a plot of y2 vs. x:



   c. Below is a plot of y2 vs. x with an intercept line:



Code Appendix:

```
# Problem 1
favorite <- read.table(file = 'favorite.data', header = FALSE)
# part a
mean(favorite[,1])

# part b
median(favorite[,1])

# part c
sd(favorite[,1])

# part d
min(favorite[,1])

# part e
max(favorite[,1])

# part f
hist(favorite[,1], main = 'Histogram of Favorite Data', xlab = 'Value of Favorite')

# Problem 2
set.seed(1)
random_values <- rnorm(n = 10000, mean = 0, sd = 1)

# part a
hist(random_values, main = 'Histogram of Random Standard Normal Values', xlab = 'Valu
e of Random Data')

# part b
mean(random_values); median(random_values); sd(random_values)

# Problem 3
a <- seq(from = 5, to = 160, by = 5); b <- seq(from = 87, to = 56); d <- a * b

# part a
d[15:17]

# part b
```

```r
d[d>2000]

# part c
length(d[d>6000])

# Problem 4
sum_perfect_squares <- function(x) {
  sum(which(sqrt(1:x) %% 1 == 0))
}
# part a
sum_perfect_squares(100)

# part b
sum_perfect_squares(100000)

# Problem 5
find_perfect_squares <- function(x) {
  which(sqrt(1:x) %% 1 == 0)
}

# part a
find_perfect_squares(500)

# part b
perfect_squares_matrix <- matrix(find_perfect_squares(100000), ncol = 4)
sink('perfect_squares_matrix.txt')
perfect_squares_matrix
sink()

# part c
perfect_squares_matrix[15, 3]

# Problem 6
x <- seq(from = -pi, to = pi, length.out = 50)
y1 <- sin(x)
y2 <- cos(x)

# part a
plot(y1, x, main = 'Part a')

# part b
plot(y2, x, type = 'l', main = 'Part b')

# part c
plot(y2, x, type = 'l', main = 'Part c')
abline(a = 1, b = -1/3)
```