Assignment 1: Simple Computations in R

Garrett Dozier

01/27/2021

Introduction

In this class we will use a programming language called R. R is an open-source language and environment for statistical computing and graphics. Additionally, we will be using a third-party program called RStudio, which is an Integrated Development Environment (IDE). RStudio is a user-friendly program in which we can run R. Before completing this R tutorial, you will need to install R and RStudio on your computer: Instructions to install R and RStudio. If you are unable to install R or RStudio, see instruction in Assignment 0 for setting up RStudio Cloud.

Please complete the following questions in RStudio and save your work as an **R markdown** file (file extension: .Rmd). Here is a Quick introduction to R markdown and a Complete guide to R Markdown.

Instructions

Download the Assignment01.Rmd file (right click on this link, select Save Link As..) and open the it in RStudio. Then complete this assignment by filling in each answer **in the R markdown document**. For full credit, show your R code within the designated R Chunks and print the answers (see example answer to first question below). Run the chunks by clicking on the “Run” button in the code editor menu. You may use the internet to help you solve these problems. Remember to save often.

Deliverables

When finished, click the **Knit** button to generate an HTML document. Click the little arrow next to the knit button and select Knit to PDF to generate a PDF document. If your computer will not generate a PDF, you can Knit to Word instead. Please upload to Canvas each of the following items (submitted as seperate files, not as a single zip file):

1. A completed .Rmd file

2. An HTML file knitted from the completed .Rmd file

3. A PDF (or Word) file knitted from the completed .Rmd file.

Questions

1. Compute 3/5. Set result to the variable A and print A.

```
A = 3/5
print(A)
```

```
## [1] 0.6
```

2. Compute and print e^π.

```
computedequation = exp(pi)
print(computedequation)

## [1] 23.14069
```

3. Make a vector C of length 10, starting with 3 and skipping every 2 integers. Print C.

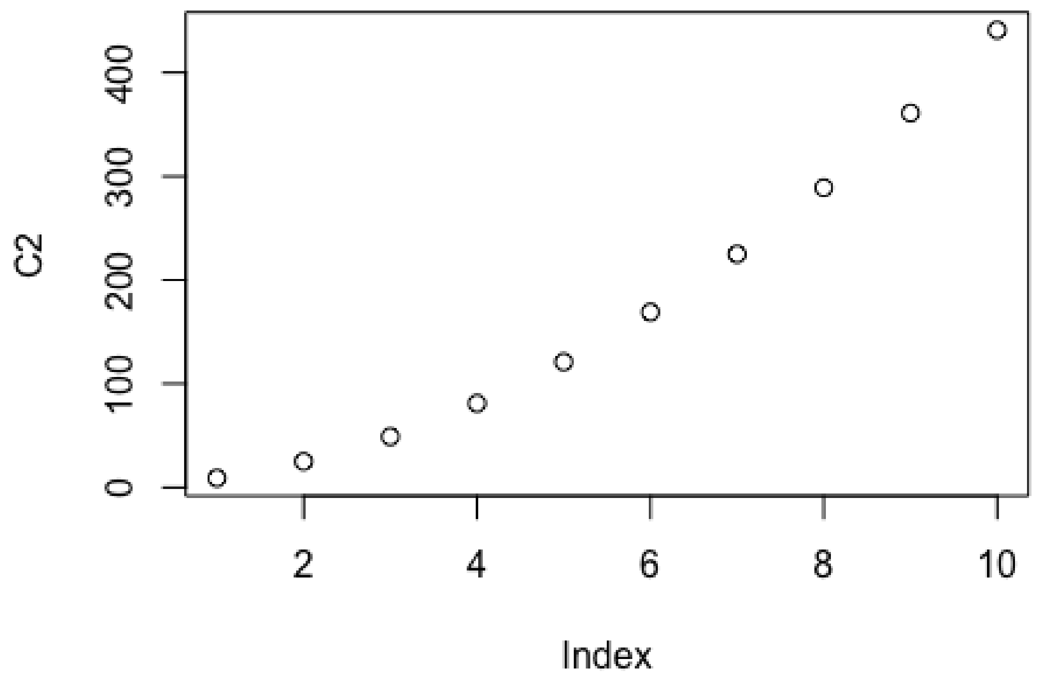
```
C = seq(from = 3, by = 2, length.out = 10)
print(C)

##  [1]  3  5  7  9 11 13 15 17 19 21
```

4. Raise each element of C by the power of 2 and print the result.

```
C2= (C)^2
print(C2)

##  [1]   9  25  49  81 121 169 225 289 361 441
```

5. Plot the square of vector C versus the index of the elements.

```
plot1 = plot(C2)
```



```
print(plot1)

## NULL
```

6. Sort vector C in reverse order and print the result.

```
C3 = rev(C)
print(C3)

##  [1] 21 19 17 15 13 11  9  7  5  3
```

7. Which elements of C are greater than 10? Set result to the variable B and print B.

```
B = C[C>10]
print(B)

## [1] 11 13 15 17 19 21
```

8. Add all the elements of C together and print the result.

```
C4 = sum(C)
print(C4)
```

```
## [1] 120
```

9. Divide all the elements of C by the sum of C and print the result.

```
C5 = length(C)
C6 = C4/C5
print(C6)

## [1] 12
```

10. Print a sequence of 25 numbers from −π to π.

```
Computedsequence = seq(from = -pi, to = pi, length.out = 25)
print(Computedsequence)

##  [1] -3.1415927 -2.8797933 -2.6179939 -2.3561945 -2.0943951 -1.8325957
##  [7] -1.5707963 -1.3089969 -1.0471976 -0.7853982 -0.5235988 -0.2617994
## [13]  0.0000000  0.2617994  0.5235988  0.7853982  1.0471976  1.3089969
## [19]  1.5707963  1.8325957  2.0943951  2.3561945  2.6179939  2.8797933
## [25]  3.1415927
```

11. Print the first difference of the vector C (difference between each vector element).

```
C7= diff(C, lag = 1)
print(C7)

## [1] 2 2 2 2 2 2 2 2 2
```

12. Print the second difference of the vector C (the difference between every second vector element).

```
C8= diff(C, lag = 2 )
print(C8)

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

13. Get 48 random, uniformly distributed numbers, put in vector L and print L.

```
L = runif(48)
print(L)

##  [1] 0.652773410 0.994218793 0.900326383 0.738893983 0.669099405
0.962865554
##  [7] 0.003686891 0.265741988 0.165501884 0.652329480 0.639699197
0.827112818
## [13] 0.300076495 0.309886977 0.228277181 0.781859671 0.660208025
0.564094203
## [19] 0.370386959 0.415194081 0.714833268 0.805889859 0.199913313
```

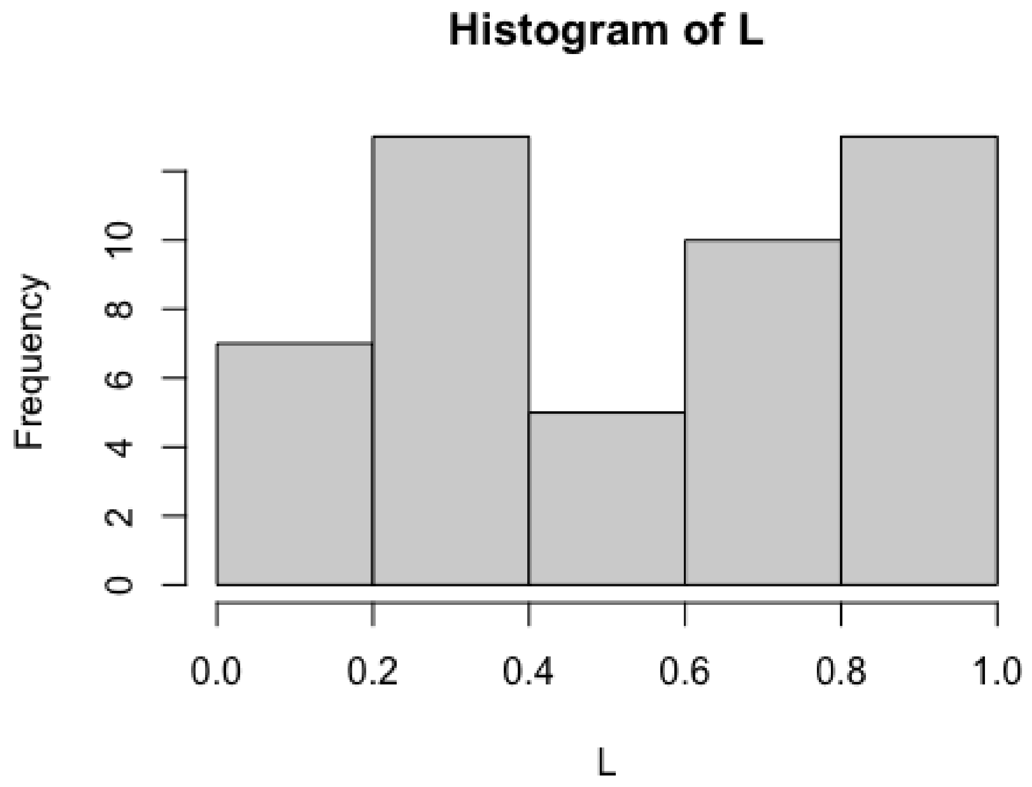
```
0.336465963
## [25] 0.779489944 0.388539814 0.289902258 0.309823335 0.100077131
0.444110490
## [31] 0.853309034 0.855974308 0.354558296 0.447377081 0.013388862
0.994496127
## [37] 0.836871448 0.201375816 0.659939489 0.990985591 0.163770239
0.223871153
## [43] 0.828383479 0.868480975 0.573510242 0.190213253 0.383609225
0.953576955
```

14. Calculate and print the maximum, minimum and median of L.

```
Lmin = min(L)
Lmax = max(L)
Lmean = mean(L)
print("Minimum is,")

## [1] "Minimum is,"

print(Lmin)

## [1] 0.003686891

print("Maximum is,")

## [1] "Maximum is,"

print(Lmax)

## [1] 0.9944961

print("Mean is,")

## [1] "Mean is,"

print(Lmean)

## [1] 0.5388535
```

15. Plot a histogram of L and include the plot below. This is an empirical distribution.

```
Histogram = hist(L)
```



```
print(Histogram)

## $breaks
## [1] 0.0 0.2 0.4 0.6 0.8 1.0
##
## $counts
## [1]  7 13  5 10 13
##
## $density
## [1] 0.7291667 1.3541667 0.5208333 1.0416667 1.3541667
##
## $mids
## [1] 0.1 0.3 0.5 0.7 0.9
##
## $xname
## [1] "L"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

16. Concatenate the 2 vectors, C and C^2, using the ‘c’ function. For example: c(B,L). Print the result.

```
C9 = c(C,C2)
print(C9)

##  [1]   3   5   7   9  11  13  15  17  19  21   9  25  49  81 121 169 225
289 361
## [20] 441
```

17. Form a 6 by 8 matrix of random numbers from vector L in question 13. Set matrix to the variable M and print M.

```
M = matrix(L,nrow = 6, ncol = 8  )
print(M)

##           [,1]        [,2]      [,3]      [,4]      [,5]       [,6]
[,7]
## [1,] 0.6527734 0.003686891 0.3000765 0.3703870 0.7794899 0.85330903
0.8368714
## [2,] 0.9942188 0.265741988 0.3098870 0.4151941 0.3885398 0.85597431
0.2013758
## [3,] 0.9003264 0.165501884 0.2282772 0.7148333 0.2899023 0.35455830
0.6599395
## [4,] 0.7388940 0.652329480 0.7818597 0.8058899 0.3098233 0.44737708
0.9909856
## [5,] 0.6690994 0.639699197 0.6602080 0.1999133 0.1000771 0.01338886
0.1637702
## [6,] 0.9628656 0.827112818 0.5640942 0.3364660 0.4441105 0.99449613
0.2238712
##           [,8]
## [1,] 0.8283835
## [2,] 0.8684810
## [3,] 0.5735102
## [4,] 0.1902133
## [5,] 0.3836092
## [6,] 0.9535770
```

18. Select the 3rd row of M and set it equal to the variable P. Print P.

```
P = M[3,]
print(P)

## [1] 0.9003264 0.1655019 0.2282772 0.7148333 0.2899023 0.3545583 0.6599395
## [8] 0.5735102
```

19. Select the 5th row and set it to Q. Print Q.

```
Q = M[5,]
print(Q)

## [1] 0.66909940 0.63969920 0.66020803 0.19991331 0.10007713 0.01338886
0.16377024
## [8] 0.38360922
```

20. Print the dot product of P and Q.

```
dotproduct = geometry::dot(P,Q)
print(dotproduct)

## [1] 1.363736

#Or a second way to get dot without uploading the geometry package
dotproduct2 = P %*% Q
print(dotproduct2)

##          [,1]
## [1,] 1.363736
```