# Chase Darlington CA4123

*Chase Darlington*

*September 11, 2018*

BSDS 100: Intro to Data Science with R Assignment 4 by Chase Darlington

Directions: For all questions in this assignment, write complete sentences and fully answer any question that is asked. Provide all R code and solutions by knitting your final RStudio file into a single file named [your name] CA4.pdf. This assignment is due next Tuesday by the beginning of class (by 2:39 PM, Tuesday, September 11). The assignment is worth 30 points. Late assignments will automatically have 10 points deducted. Remember to include all R code and results (except when noted) to receive full credit.

1. Create the vector > myAtomicVector <- c(1, 4, 3, 2, NA, 3.22, -44, 2, NA, 0, 22, 34) Now, create code that runs to answer each of the following questions.

```
myAtomicVector <- c(1, 4, 3, 2, NA, 3.22, -44, 2, NA, 0, 22, 34)
myAtomicVector
```

```
##  [1]   1.00   4.00   3.00   2.00     NA   3.22 -44.00   2.00     NA   0.00
## [11]  22.00  34.00
```

```
table(myAtomicVector) []
```

```
## myAtomicVector
##  -44    0    1    2    3 3.22    4   22   34
##    1    1    1    2    1    1    1    1    1
```

(a) (2 points) How many positive numbers (> 0) are there in this vector (do not count NAs)?

```
table1 <- table(myAtomicVector)
table1[names(table1)>0]
```

```
## myAtomicVector
##    1    2    3 3.22    4   22   34
##    1    2    1    1    1    1    1
```

```
sum(table1[names(table1)>0])
```

```
## [1] 8
```

```
table(myAtomicVector>0)
```

```
##
## FALSE  TRUE
##     2     8
```

There are "8" positive numbers (>0) in the "myAtomicVector" vector, not counting NAs.

(b) (2 points) How many negative numbers (< 0) are there in this vector (do not count NAs)?

```
table1 <- table(myAtomicVector)
table1[names(table1)<0]
```

```
## -44
##   1
```

```
sum(table1[names(table1)<0])
```

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

```
table(myAtomicVector<0)
```

```
##
## FALSE  TRUE
##     9     1
```

There is "1" negative number (<0) in the "myAtomicVector" vector, not counting NAs

(c) (2 points) How many 0's are there in this vector (do not count NAs)?

```
table1 <- table(myAtomicVector)
table1[names(table1)==0]
```

```
## 0
## 1
```

```
sum(table1[names(table1)==0])
```

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

```
table(myAtomicVector==0)
```

```
##
## FALSE  TRUE
##     9     1
```

There is "1" zero in the "myAtomicVector" vector, not counting NAs

(d) (2 points) How many NAs are there is this vector?

```
myAtomicVector[myAtomicVector=="N/A"]
```

```
## [1] NA NA
```

```
frequency("N/A"==myAtomicVector)
```

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

```
sapply(myAtomicVector, function(x) sum(is.na(x)))
```

```
##  [1] 0 0 0 0 1 0 0 0 1 0 0 0
```

```
sum(sapply(myAtomicVector, function(x) sum(is.na(x))))
```

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

```
sum(is.na(myAtomicVector))
```

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

(e) (2 points) How many numbers in the vector are non-zero and not NAs?

```
table(myAtomicVector!=0)
```

```
##
## FALSE  TRUE
##     1     9
```

```
myAtomicVector!=0
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE    NA  TRUE  TRUE  TRUE    NA FALSE  TRUE
## [12]  TRUE
```

```
list(table(myAtomicVector)!=0)
```

```
## [[1]]
## myAtomicVector
##  -44    0    1    2    3 3.22    4   22   34
## TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

(f) (2 points) What is the sum of the positive numbers in this vector?

```
subset(myAtomicVector, myAtomicVector>0)
```

```
## [1]  1.00  4.00  3.00  2.00  3.22  2.00 22.00 34.00
```

```
sum(myAtomicVector[which(myAtomicVector>0)])
```

```
## [1] 71.22
```

(g) (2 points) What is the sum of the negative numbers is this vector?

```
subset(myAtomicVector, myAtomicVector<0)
```

```
## [1] -44
```

```
sum(myAtomicVector[which(myAtomicVector<0)])
```

```
## [1] -44
```

2. Consider a vector of length 1000, where Fn is the nth number in the sequence. Then the Fibonacci sequence is the vector where the following recursion holds.

Fn = Fn-1 + Fn-2

That is, the nth number in the sequence will be the sum of the previous two numbers. Create a vector, Fib.vec, that has the first 1000 numbers in the Fibonacci sequence using the following code (which includes a for loop that we'll talk more about later in this course):

```
#initialize the vector for memory
Fib.vec <- rep(0, 1000)
#store the first two entries to be 1
Fib.vec[1] <- 1
Fib.vec[2] <- 1
#iterate to get the remaining values
for(i in 3:1000){
  Fib.vec[i] <- Fib.vec[i-1] + Fib.vec[i-2]
}
```

(a) (2 points) What are the first 8 and last 8 entries of Fib.vec?

```
#initialize the vector for memory
Fib.vec <- rep(0, 1000)
#Fib.vec
#store the first two entries to be 1
Fib.vec[1] <- 1
Fib.vec[2] <- 1
#iterate to get the remaining values
for(i in 3:1000){
  Fib.vec[i] <- Fib.vec[i-1] + Fib.vec[i-2]
}

Fib.vec[1:8]
```

```
## [1]  1  1  2  3  5  8 13 21
```

```
Fib.vec[993:1000]
```

```
## [1] 1.497069e+207 2.422308e+207 3.919377e+207 6.341685e+207 1.026106e+208
## [6] 1.660275e+208 2.686381e+208 4.346656e+208
```

```
#a <- list(Fib.vec[1:8])
#b <- list(Fib.vec[993:1000])
#list(a,b)
#c(a,b)

#rbind(Fib.vec[1:8],Fib.vec[993:1000])
#table3a <- merge(Fib.vec[1:8],Fib.vec[993:1000])
#names(table3a)
#colnames(table3a)[names(table3a) == "x"] <- "First 8"
#colnames(table3a)[names(table3a) == "y"] <- "Last 8"
#table3a
```

(b) (2 points) Using the Fibonacci numbers generated above, generate a vector (of length 999) with values (again, don't print these out) Zn = Fn sub(n+1) / Fn sub(n) For this sequence, one could use a for loop as used in part (a), or better yet note that dividing two vectors of the same length will return a vector whose entries contain the division of entries in each vector. This is very useful for such calculations and this vector operation is fairly unique to R!

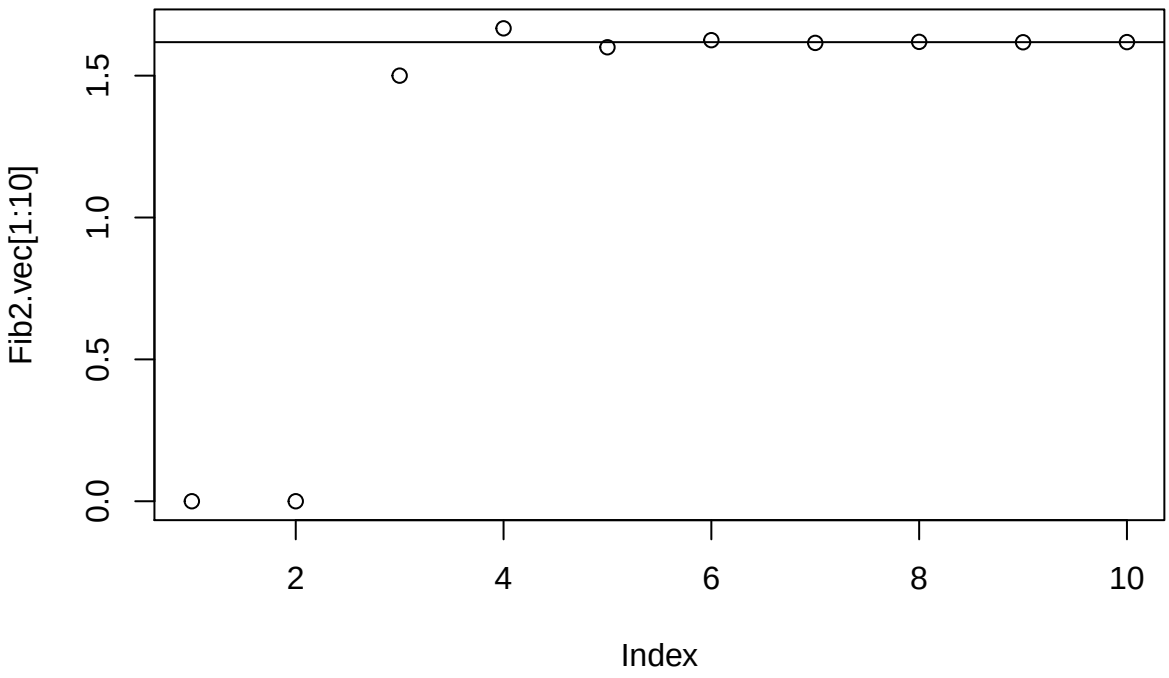
```
#initialize the vector for memory
Fib.vec <- rep(0, 1000)
#store the first two entries to be 1
Fib.vec[1] <- 1
Fib.vec[2] <- 1
#iterate to get the remaining values
for(i in 3:1000){
  Fib.vec[i] <- Fib.vec[i-1] + Fib.vec[i-2]
}


Fib2.vec <- rep(0, 999)
#store the first entry to be 1
#Fib.vec[1] <- 1
#iterate to get the remaining values
for(i in 3:999){
  Fib2.vec[i] <- (Fib.vec[i+1] / Fib.vec[i])
}
Fib2.vec[1:8]
```

```
## [1] 0.000000 0.000000 1.500000 1.666667 1.600000 1.625000 1.615385 1.619048
```

(c) (2 points) Plot the first 10 entries of the vector Zn using the command plot(Zn[1 : 10]). Then add a line to the plot using the following command abline(h = (1 + sqrt(5))/2). This value is known as the golden ratio in mathematics.

```
plot(Fib2.vec[1:10])
abline(h = (1 + sqrt(5))/2)
```




(d) (2 points) Comment on the plot that you obtain. What do you observe?

```
#2 slope of line h=0 and Fib2.vec ~= expression(H) except for 2 outliers: values x=1,2 are outliers
```

(e) (2 points) What is wrong with typing the following code? x <- Fib.vec(1:5)

```
# Fib.vec is interpreted as a function because of the parenthesis, and therefore produces an error beca
```

3. Using the Fibonacci vector above, create the following data structures. Remember that using the ?foo will provide documentation on the function foo as needed. And be careful about the use of arguments here to get the data structure you want. • A matrix of size 100×10 named Fib.matrix1 whose columns, when stacked on top of one another will return the original vector. • A matrix of size 100 × 10 named Fib.matrix2 whose rows, when stacked side by side will return the original vector. • An array of dimension 10 × 10 × 10 names Fib.array where each 10 × 10 matrix in the array is such that when its columns are stacked on top of one another would generate a Fibbonaci vector of length 100. Answer the following questions

```
Fib.matrix1 <- matrix(Fib.vec, nrow=100, ncol=10)
Fib.matrix2 <- matrix(Fib.vec, nrow=100, ncol=10, byrow=T)
#Fib.matrix2
#Fib.matrix1
Fib.matrix1[18,]
```

```
##  [1]  2.584000e+03  2.046711e+24  1.621140e+45  1.284058e+66  1.017065e+87
##  [6] 8.055874e+107 6.380823e+128 5.054064e+149 4.003176e+170 3.170799e+191
```

```
mean(Fib.matrix1[18,])
```

```
## [1] 3.170799e+190
```

```r
sd(Fib.matrix2[,8])
```

```
## [1] Inf
```

```r
Fib.array <- array(Fib.vec, c(10,10,10))
Fib.array[5,2,8]
```

```
## [1] 1.193103e+149
```

(a) (2 points) What is the mean of the 18th row of Fib.matrix1?

```r
#Irrational number: 3.17...
```

(b) (2 points) What is the standard deviation of the 8th column of Fib.matrix2?

```r
#Infinity
```

(c) (2 points) What is the entry in the 5th row of the 2nd column of the 8th matrix in Fib.array?

```r
#The entry in the 5th row of the 2nd column of the 8th matrix in Fib.array is 1.1931... (refer above)
```