Smith, Catherine BUAD 502B R Programming Module 2 RStudio Assignment

Chapter 7: Built In Functions

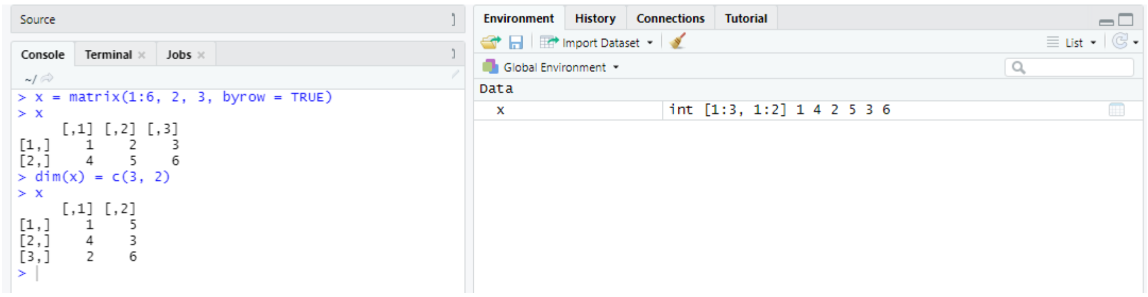
Exercise 7.10: To solve this problem I first created the matrix "x" as taught in chapter 5, ensuring that I used "byrow = TRUE," because R defaults to a column-wise display otherwise. I then used the built in "dim" function to rearrange "x" from two rows by three columns into three rows by two columns using the code displayed below.

```
> x = matrix(1:6, 2, 3, byrow = TRUE)

> x

> dim(x) = c(3,2)

> x
```



```
> x = matrix(1:6, 2, 3, byrow = TRUE)
> x
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> dim(x) = c(3, 2)
> x
     [,1] [,2]
[1,]    1    5
[2,]    4    3
[3,]    2    6
> 
```

```
Data
  x        int [1:3, 1:2] 1 4 2 5 3 6
```

Exercise 7.13: To solve this problem I created a formula applying several functionals. First, I took the cumulative product (cumprod) of all the numerators (3:49) divided by all the denominators (4:50); next, I used the sum function to add those up. Finally, the answer came out to 7.997616.

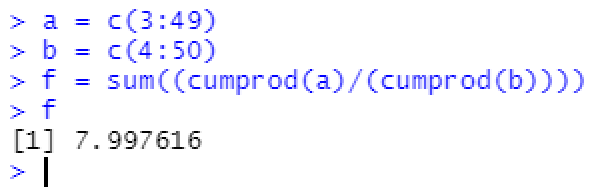
```
> a = c(3:49)

> b = c(4:50)

> f = sum((cumprod(a) / (cumprod(b)))

> f
```

Screenshot from RStudio:

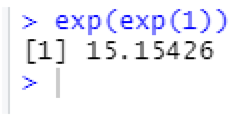


```
> a = c(3:49)
> b = c(4:50)
> f = sum((cumprod(a)/(cumprod(b))))
> f
[1] 7.997616
> 
```

Exercise 7.14: R defaults to e = exp(1), so to write this command my initial instinct was e(exp(1)). However, this will not work because e hasn't been assigned a value. Therefore, since the "exp" function has a base of "e" we can express the same idea using "exp."

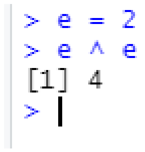
```
> exp(exp(1))
```

Screenshot from RStudio:



```
> exp(exp(1))
[1] 15.15426
> 
```

Proof of concept:



```
> e = 2
> e ^ e
[1] 4
> 
```

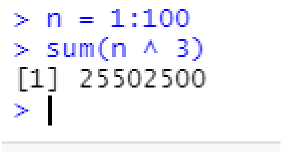
Exercise 7.15: This single command uses the built-in sum function with the first calculation in the series as 1^3 and the final calculation in the series as 100^3, adding each cubed integer including and in between those two numbers. I referenced page 37 of textbook for process.

```
> n = 1:100

> sum(n ^ 3)

> 25502500
```

Screenshot from RStudio:



```
> n = 1:100
> sum(n ^ 3)
[1] 25502500
> 
```

Exercise 7.20: In order to count the number of distinct elements in a vector named "x" in two R functions I first created the vector "x," then made a function that uses the "as.data.frame" and frequency table ideas taught in BUAD 502(A). I made the function name "distinct.elements," and coded a function of "x" where the number of distinct elements appears on the far left hand side of the display (see red box below):

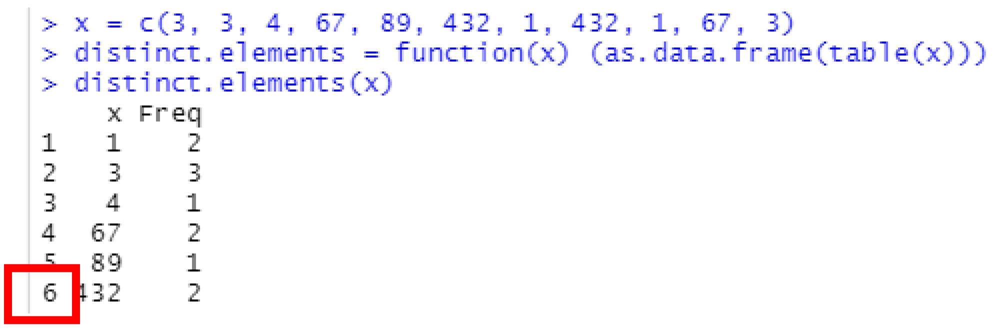
```
> x = c(3, 3, 4, 67, 89, 432, 1, 432, 1, 67, 3)

> distinct.elements = function(x) (as.data.frame(table(x)))

> distinct.elements(x)
```

Screenshot from RStudio:

Smith, Catherine BUAD 502B R Programming Module 2 RStudio Assignment



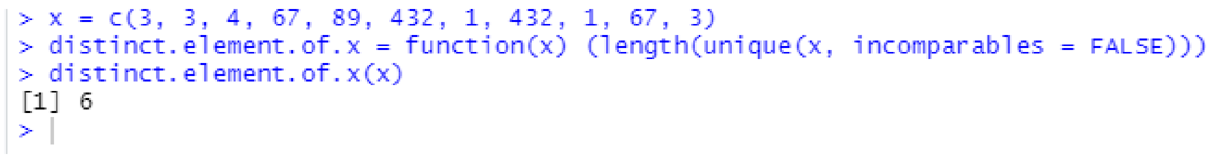
```
> x = c(3, 3, 4, 67, 89, 432, 1, 432, 1, 67, 3)
> distinct.elements = function(x) (as.data.frame(table(x)))
> distinct.elements(x)
    x Freq
1   1    2
2   3    3
3   4    1
4  67    2
5  89    1
6 432    2
```

Another way to solve this is to use both the "length" and "unique" functionals. The "unique" functional removes those numbers which appear more than once, then "length" provides the total elements remaining in that vector.

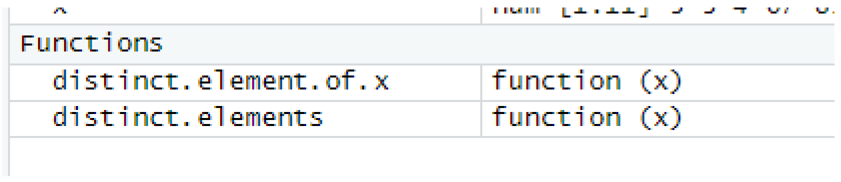
> x = c(3, 3, 4, 67, 89, 432, 1, 432, 1, 67, 3)

> distinct.element.of.x = function(x) (length(unique(x, incomparables = FALSE)))

> distinct.element.of.x(x)



```
> x = c(3, 3, 4, 67, 89, 432, 1, 432, 1, 67, 3)
> distinct.element.of.x = function(x) (length(unique(x, incomparables = FALSE)))
> distinct.element.of.x(x)
[1] 6
> 
```



```
Functions
  distinct.element.of.x    function (x)
  distinct.elements        function (x)
```

Exercise 7.21: To understand how to do this problem, I had to break it into smaller elements, solve each of those individually, then determine how to use multiple concepts in R to combine those to a one line function. My initial steps were:

> x = seq (1:100)

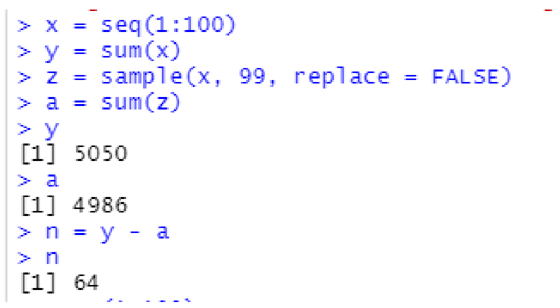
> y = sum(x)

> a = sample(x, 99, replace = FALSE)

> b = sum(a)

> o = y - a

Another method I tried was this:



```
> x = seq(1:100)
> y = sum(x)
> z = sample(x, 99, replace = FALSE)
> a = sum(z)
> y
[1] 5050
> a
[1] 4986
> n = y - a
> n
[1] 64
```

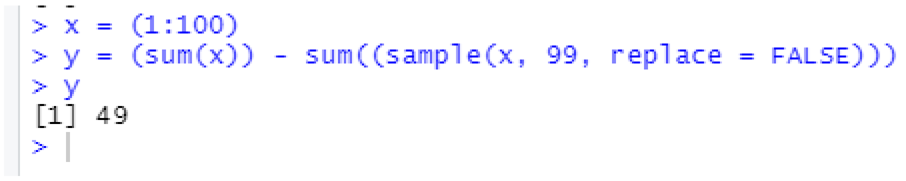
Both of these admittedly similar methods did give me answers but failed to do so in one line as the exercise directions dictate. In order to get it into one line, I had to use more than the vectors and arithmetic capabilities in R. I had to add in built in functions, specifically, a combination of "sum" and "sample."

```
> x = seq(1:100)

> y = (sum(x)) – (sum(sample(x, 99, replace = FALSE)))

> y = 49
```

Screenshot from R:



```
> x = (1:100)
> y = (sum(x)) - sum((sample(x, 99, replace = FALSE)))
> y
[1] 49
> 
```

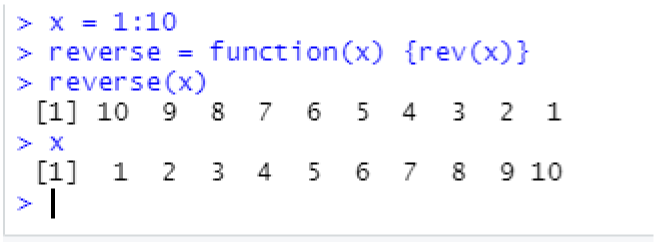
Chapter 8: User Written Functions

Exercise 8.1: To answer this exercise, wrote the function that is named "reverse" and will reverse the order of elements in the vector "x" using the function "rev." I tested this by applying several values to "x" to demonstrate where each was displayed in reverse of its entry.

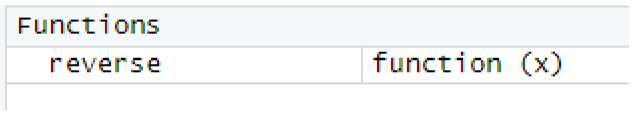
```
> x = 1:10

> reverse = function(x) {rev(x)}
```

Screenshots from RStudio with code:



```
> x = 1:10
> reverse = function(x) {rev(x)}
> reverse(x)
 [1] 10  9  8  7  6  5  4  3  2  1
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> 
```



Functions	
reverse	function (x)

Exercise 8.5: My first try at solving this exercise heavily mirrored the "mean.of.objects" example in the instructional videos for Module 2. However, after building that code [> movave = function(…(return(mean(c(…)))] and assigning values to vector x, I realized that it took the cumulative average not the rolling average and that it didn't account for the integer's division by two. Upon realizing the equation in R would be more complex, I first worked it out algebraically on paper assigning values from 10-15 to the vector "x." After computing that on paper & confirming the results in MS Excel, I conducted significant research across the CRAN website and through trial and error I deduced:

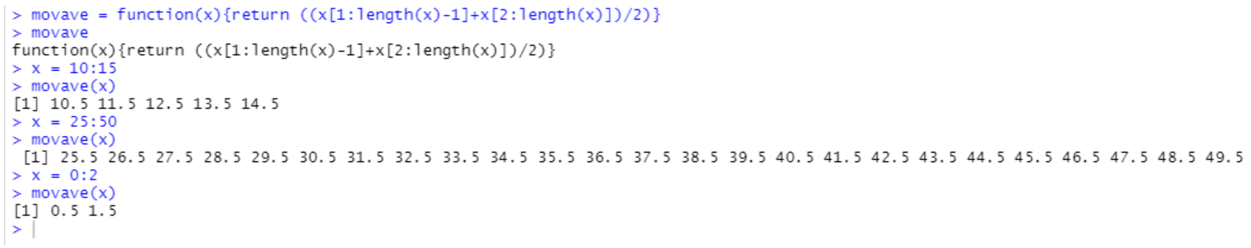
```
> movave = function(x) {return (x[1:length (x-1) + x[2:length (x)])2)}
```

The elements were deduced as follows:

1) "movave" is the function name
2) "= function(x)" denotes that this will be a function of the vector "x"
3) "{return (x[1:length (x-1) + x[2:length (x)])2)}" denotes that I want the answer returned in such a way that the elements are the first integer in the vector "x" added to the second integer in the vector "x", then divided by two. The length argument tells me that I 1) want to elements in this function's formula to combine and 2) those two elements interact with each other. When I apply a vector with 5 integers, it will return 5 results; when I change the vector to 10 integers, it returns 10 results because the formula computes each element of the vector "x."

```
> movave = function(x) (return((x[1:kength(x)-1] + [2:length(x)]) / 2)
```

Screenshot from RStudio:

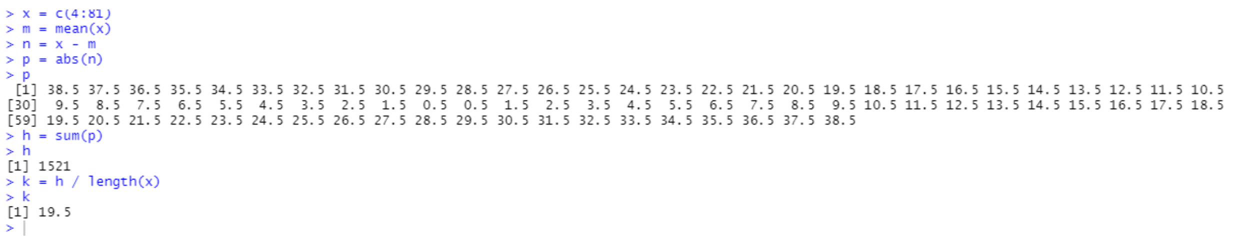


```
> movave = function(x){return ((x[1:length(x)-1]+x[2:length(x)])/2)}
> movave
function(x){return ((x[1:length(x)-1]+x[2:length(x)])/2)}
> x = 10:15
> movave(x)
[1] 10.5 11.5 12.5 13.5 14.5
> x = 25:50
> movave(x)
 [1] 25.5 26.5 27.5 28.5 29.5 30.5 31.5 32.5 33.5 34.5 35.5 36.5 37.5 38.5 39.5 40.5 41.5 42.5 43.5 44.5 45.5 46.5 47.5 48.5 49.5
> x = 0:2
> movave(x)
[1] 0.5 1.5
> 
```

Exercise 8.9: To solve this exercise, I first broke it down into the component parts: 1) create a vector of numbers, 2) find the mean of those numbers, 3) find the difference between each number in the vector and the mean, 4) take the absolute value of those differences, 5) sum the absolute values, 6) divide the sum by total number of elements in the vector, 7) solve.

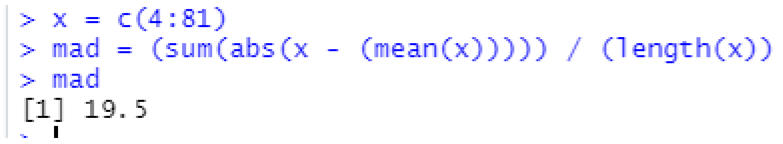
```
> mad = function(x) {(sum(abs(x-(mean(x))))) / length(x))}
```

Screen shot of steps in R:



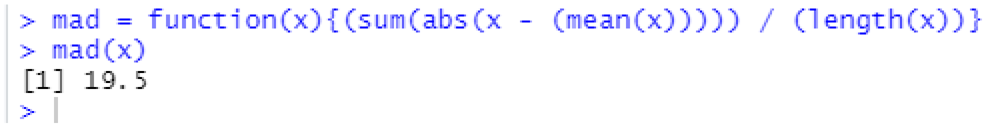
```
> x = c(4:81)
> m = mean(x)
> n = x - m
> p = abs(n)
> p
 [1] 38.5 37.5 36.5 35.5 34.5 33.5 32.5 31.5 30.5 29.5 28.5 27.5 26.5 25.5 24.5 23.5 22.5 21.5 20.5 19.5 18.5 17.5 16.5 15.5 14.5 13.5 12.5 11.5 10.5
[30]  9.5  8.5  7.5  6.5  5.5  4.5  3.5  2.5  1.5  0.5  0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5 14.5 15.5 16.5 17.5 18.5
[59] 19.5 20.5 21.5 22.5 23.5 24.5 25.5 26.5 27.5 28.5 29.5 30.5 31.5 32.5 33.5 34.5 35.5 36.5 37.5 38.5
> h = sum(p)
> h
[1] 1521
> k = h / length(x)
> k
[1] 19.5
> 
```

However, the exercise calls for doing this in a single argument:



```
> x = c(4:81)
> mad = (sum(abs(x - (mean(x))))) / (length(x))
> mad
[1] 19.5
```

Finally, I needed to turn that argument into one function:



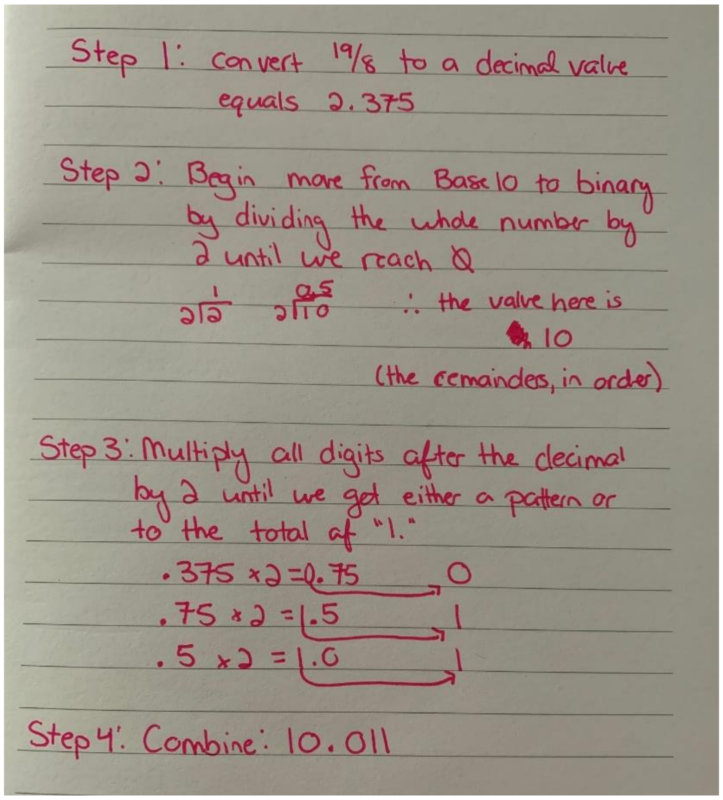
```
> mad = function(x){(sum(abs(x - (mean(x))))) / (length(x))}
> mad(x)
[1] 19.5
> 
```

<u>Chapter 9: Utilities</u>

Exercise 9.4: To solve this exercise, I converted the fraction 19/8 to binary in 4 steps: 1) convert the fraction to a decimal value, 2) begin moving from base 10 to binary by dividing the whole number by 2 until I reached 0, noting the remainder's integer, 3) multiplying the digits to the right of the decimal point by 2 until either the pattern repeats or we reach a total of 1, this time noting the number preceding the decimal; in this case, I reached 1, 4) combining the numbers noted to give the floating binary representation.

In this case, the answer is 10.001.

Work below:



Step 1: convert 19/8 to a decimal value
equals 2.375

Step 2: Begin move from Base 10 to binary by dividing the whole number by 2 until we reach 0

$\overset{1}{2\overline{)2}}$ $\overset{0.5}{2\overline{)1.0}}$ ∴ the value here is 10

(the remainders, in order)

Step 3: Multiply all digits after the decimal by 2 until we get either a pattern or to the total of "1."

$.375 \times 2 = 0.75 \rightarrow 0$

$.75 \times 2 = 1.5 \rightarrow 1$

$.5 \times 2 = 1.0 \rightarrow 1$

Step 4: Combine: 10.011