Solutions for Laboratory 1
Practical Programming

Biol 535

In this document I provide my initial solutions for the *Practical Programming* assignment that was part of the *Introduction to R Laboratory*. I should emphasize that there are many correct ways to solve these problems. These are just examples.

Problem 1

The first problem focused on using if–then statements. One solution follows.

```
> x.values <- seq(-2,2,by=0.1)
> n <- length(x.values)
> y.values <- rep(0,n)
> # action
> for(i in 1:n){
+   if(x.values[i] <= 0){            # first decision point
+     y.values[i] <- -x.values[i]^3
+   } else {
+     if (x.values[i] <=1){          # second decision point
+       y.values[i] = x.values[i]^2
+     } else {                       # third decision point -- everything else
+       y.values[i] = sqrt(x.values[i])
+     }
+   }
+ }
> show(y.values)
```

```
 [1] 8.000000 6.859000 5.832000 4.913000 4.096000 3.375000 2.744000 2.197000
 [9] 1.728000 1.331000 1.000000 0.729000 0.512000 0.343000 0.216000 0.125000
[17] 0.064000 0.027000 0.008000 0.001000 0.000000 0.010000 0.040000 0.090000
[25] 0.160000 0.250000 0.360000 0.490000 0.640000 0.810000 1.000000 1.048809
[33] 1.095445 1.140175 1.183216 1.224745 1.264911 1.303840 1.341641 1.378405
[41] 1.414214
```

```
> pdf(file="myplot.pdf",height=4,width=5)      # opens PDF file
> plot(x.values,y.values,type="b",col="blue")  # writes the plot to the PDF file
> dev.off()                                    # closes the PDF file
```

```
null device
          1
```

The last command line generates figure 1.

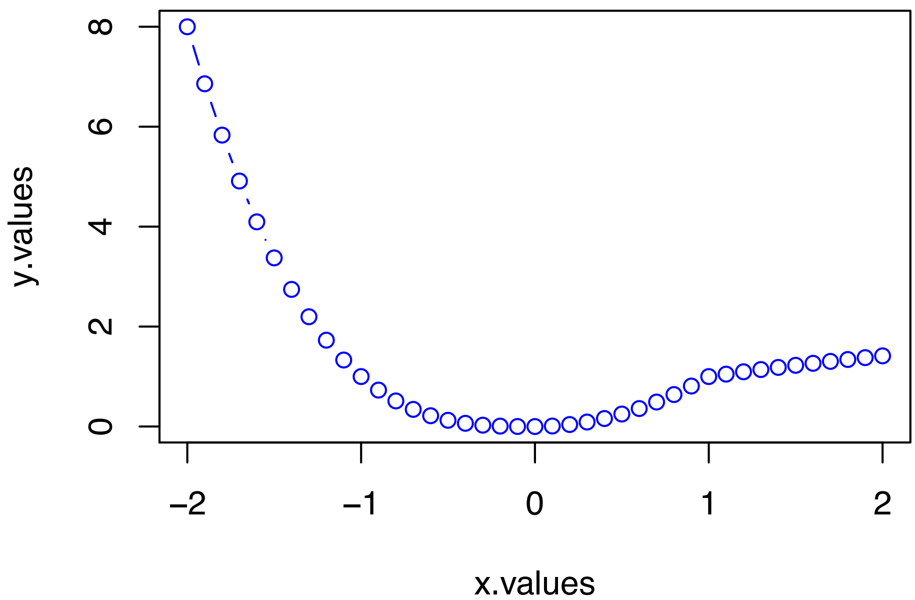

Figure 1: Plot of y-values with respect to x-values for problem 1.

Problem 2

In this problem you were asked to use a for loop to solve

$$h(x,n) = 1 + x + x^2 + \ldots + x^n \quad (1)$$
$$= \sum_{i=0}^{n} x^n \quad (2)$$

My solution was

```
> # Input:  define parameter values
> x = 0.3
> n = 55
> h = 0       # initialize variable
> # Action
> for(i in 0:n){
+   h = h + x^i
+ #  show(c(i,h))
+ }
> show(h)

[1] 1.428571
```

The technique I used here was to keep updating the value of h. Each time through the loop I changed the value of h, using the old value of h. This is a common programming strategy.

Problem 3

In this problem we solve the exact solution to equation 2 when $x = 0.3$ and $n = 55$. The exact solution of the series is given by a known identity.

```
> h.exact = (1-x^(n+1))/(1-x)
> show(h.exact)

[1] 1.428571

> # test if h == h.exact
> h == h.exact

[1] FALSE
```

Notice that when we test to see if the solution from our for-loop calculation is equal to the value from our exact solution the answer is false. Do they look false? Can you explain what is happening here?

Problem 4

This problem required you to solve equation 2 using a while loop. The programming trick here is to define a counter variable that increments by one each time you pass through the while-loop.

```
> # -- define parameters
> x = 6.6
> n = 8
> i=0       # initialize counter
> h=0       # inital value of h
> # -- while loop
> while(i <=n){
+   h = h + x^i
+   i = i+1  # increment counter
+ }
> show(h)

[1] 4243336
```

Problem 6

Here you were to find the geometric mean for a vector x. Recall that the geometric mean is defined as $(\prod_{i=1}^{n} x_i)^{1/n}$. As x was not specified, you could have used any vector. I used $x = 1 : 100$, but a good starting point would have been to use a vector for which you could calculate the answer by hand to check that your program was working.

My program below also finds the solution without a for-loop.

```
> x = 1:100
> n = length(x)
> # geometric mean
> gm1 = prod(x^(1/n))
> show(gm1)

[1] 37.99269

> gm2=1
> for(i in x){   #walk through values of x
+   gm2 = gm2 *(i)^(1/n)
+ }
> show(gm2)

[1] 37.99269
```

The next challenge was to calculate the harmonic mean $(\sum_{i=1}^{n} 1/x_i)^{-1}$. A solution for this follows.

```
> # Harmonic Mean
> hm1 = (sum(1/x)*1/n)^-1
> hm2 = 0
> for(i in x){
+   hm2 = hm2 + 1/x[i] * 1/n
+ }
> hm2 = 1/hm2
> show(hm2)

[1] 19.27756
```

The arithmetic mean is

```
> mean(x)

[1] 50.5
```

As expected, the arithmetic mean is greater than or equal to the geometric mean, which is in turn greater than or equal to the harmonic mean.

```
> show(c(mean(x),gm2, hm2))

[1] 50.50000 37.99269 19.27756

> (mean(x) >= gm2) && (gm2 >= hm2)

[1] TRUE
```

Checking the expected relationships is a good way to verify that your programs are working correctly.

Problem 7

This problem was to find the sum of every third element of a vector.

```
> x = 1:100
> x.sum=0
> x.vec=c()
> for(i in x){
+   # using modulo math concept.
+   if(i%%3 == 0){
+     x.sum = x.sum + x[i]
+     x.vec=c(x.vec, x[i])
+   }
+ }
> show(x.sum)
```

```
[1] 1683
```

The %% operator returns the modulo or remainder of the division of i by 3. If i is perfectly divisable by 3, then the remainder is 0. This solution uses a math concept you may not have seen before, but illustrates how expanding your knowledge of math may help you solve some of the problems. One of the example problems I gave you used this concept. You can search for help in R on the operator by typing

```
?'%%'
```

An alternative solution suggested by one of your colleagues that does not use a for-loop would be to do the following.

```
> j=seq(3,length(x),by=3)
> sum(x[j])
```

```
[1] 1683
```

Problem 9a

The problem was to create a flow chart for the program provided. The solution is shown in Table 1.

Problem 10

```
> # given vector x, find the minimum values
> n <- 1000
> x <- rnorm(n)  # creates vector with 1000 elements drawn from normal distribution
> x.min <- x[1]
> #
> for(i in 2:n){
+   if(x[i]<x.min){x.min = x[i]}
+ }
```

Table 1: Flow chart for program 'threeplus1array.r'.

Line	x	i	**Comments**
1	3	#N/A	
2	3	1	i is set to 1
3	3	1	x is written to command window
4	3	1	(x[i]%%2 ==0) is false so go to line 7
7	3 10	1	x[2] is set to 10
8	3 10	1	end of else
9	3 10	1	end of for
2	3 10	2	i is set to 2
3	3 10	2	x is written to the window
4	3 10	2	(x[i]%%2 ==0) is true so go to line 5
5	3 10 5	2	x[3] is set to 5, go to line 8
6	3 10 5	2	end of if
9	3 10 5	2	end of for
2	3 10 5	3	i is set to 3
3	3 10 5	3	x is written to the window
4	3 10 5	3	(x[i]%%2 ==0) is false so go to line 7
7	3 10 5 16	3	x[4] is set to 16
8	3 10 5 16	3	end of else
9	3 10 5 16	3	end of for

Domeig Function

The last problem was:

> Write a function "domeig" that takes as input a single vector and returns a list with components "average" (mean of the values of in the vector) and "variance" (the variance of the values in the vector). [DMB]

Lets first define the function.

```
> domeig <- function(x){
+   m <- mean(x)
+   v <- var(x)
+   y=list(mean=m,variance=v)
+   return(y)
+ }
```

Given this function, we can now use it.

```
> x <- rnorm(8)
> domeig(x)

$mean
[1] -0.1757752

$variance
[1] 1.104
```