CSCI 520
Homework 6

Megan Rose Bryant
Department of Mathematics
William and Mary

October 8, 2014

1.) *Identify any typographical errors or suggestions in the first 14 chapters of the R book. Here is a sample format of how to identify the errors. (The header is not included in positive line counts; negative line counts are take from the bottom of the page.)*

- Page vi, line 6, second examples → second example
- Page 14, line 15, four copies → one copy, four → three, four → five
- Page 29, line -4, positive subscript → positive subscripts
- Page 30, line 12, operation is → is
- Page 37, line 19, output should be

```
[1] 22.8 47.5 61.5 91.8 124.8 154.8 184.2 201.0
```

- Page 59, line 5 that → than
- Page 60, line 14. string → string

2.) *Add the following abbreviations to your .vimrc file. Then add two more abbreviations of this nature to your .vimrc file that might be helpful in speeding up your*LaTeX*input. List the extra two commands you choose for the solution to this problem.*

```
set number
set wm=10
set autoindent
set showmatch
ab ci confidence interval
ab PDF probablity density function
ab mrb Megan Rose Bryant
ab mb Megan Bryant
ab wm William \& Mary
ab wrt with respect to
ab lhs left hand side
ab lp linear programming
ab ilp interger linear  program
ab st subject to
ab bs basic solution
ab min minimize
ab max maximize
ab fr feasible region
ab OF objective function
ab rhs right hand side
ab BAS basis
ab CON constraint
ab DV decision variable
ab dlty duality
ab OV objective value
ab SA sensitivity analysis
ab SM simplex method
ab SV slack variable
ab i1n $i = 1,2, \ldots, n$
ab i1m $i = 1,2, \ldots, m$
ab j1n $j = 1,2, \ldots, n$
ab jlm $j = 1,2, \ldots, m$
ab sm \sum\limit_{i=1}^{n}
ab pr \prod\limit_{i=1}^{n}
ab INT \int\limit_{i=1}^{n}
ab Latex \LaTeX
"Comment
map ;bi to\begin{itemize}
\item
\item
\end{itemize}
```

3.) *Write a single R command that calculates n^n for $n = 1, 2, \ldots, 10$.*

```
> n = 1:10; n^n
 n = 1:10; n^n
 [1]           1           4          27         256        3125       46656
 [7]      823543    16777216   387420489 10000000000
```

4.) *A stochastic matrix has elements that are between 0 and 1, row sums that equal 1, and the same number of rows and columns. Write an R command to create the matrix A with elements given by*

$$\mathbf{A} = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix}$$

```
> x = matrix(c(0.8, 0.3, 0.2, 0.7), 2, 2); x
     [,1] [,2]
[1,]  0.8  0.2
[2,]  0.3  0.7
```

5.) *The $3 \times 4 \times 5$ array is filled with numeric values. The R command*

```
 >b = a[c(1,3), -c(1,4), 3:5]
```

sets b to a $2 \times 2 \times 3$array that includes the first and third rows, excludes the first and fourth columns, and includes the third, fourth, and fifth layers of a. Write an R command that accomplishes this same operation using the smallest number of keystrokes.

```
b = a[-2, 2:3, 3:5]
```

6.) *Write an R command that creates a 2×3matrix named x that contains the first six positive integers entered row-wise into the matrix. Display x. Then write a second R command that uses the dim function to change the dimensions of x to a 3×2matrix. Display the updated matrix 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
```

7.) *For the sample data values $x_1, x_2, \ldots, x_n$ and the associated sample order statistics $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$, the sample truncated mean (also known as the sample trimmed mean) is a measure of central tendency defined as*

$$\hat{x} = \frac{x_{(k+1)} + x_{(k+2)} + \cdots + x_{(n-k)}}{n - 2k}$$

This is the arithmetic average of the data values with the k lowest and k highest observations removed. The truncated mean is less sensitive to outliers than the arithmentic mean and is hence known as a robust estimator. This estimator is used in sports that are evaluated by a panel of n judges in which the lowest and highest scores ($k = 1$) are discarded. Likewise, the truncated mean is used in calculating the London Interbank Offered Rate (LIBOR) when $n = 18$ interest rates are collected, and the lowest four and highest four interest rates ($k = 4$) are discarded. Assuming that $k < n/2$, write an R function named tmean with two aruments x and k that calculates the truncated mean of the elements in the vector x discarding the k lowest and the k highest observations.

```
> tmean(c(9.4, 9.6, 9.1, 9.5, 9.3), 1)
> tmean(1:18, 4)
```

My code:

```
tmean <- function(x,k) {return((sum(x[(k+1):(length(x)-k)]))/(length(x)-2*k))}
> tmean(c(9.4,9.6,9.1,9.5,9.3),1)
[1] 9.4
> tmean(1:18,4)
[1] 9.5
```

8.) *Write R commands that calculate $(1-i)^n$for $n = 4, 8, 16, 20, 24$. Use your results to write a general mathematical expression for $(1-i)^n$ where n is a multiple of 4.*

```
> n = c(4,8,16,20,24);(1-1i)^n
[1]    -4+0i    16+0i   256+0i -1024+0i  4096+0i
```

Based on our findings, we can conclude that

$$(1-i)^n = (-1)^n * 2^{2n} + 0i$$

9.) *Write R commands to create a 5×5matrix named w whose elements are complex numbers. The real part of each element is the row number; the imaginary part of each element is the column number. Compute the matrix that results from multiplying each element of w by its conjugate.*

```
> w = matrix(complex(real = 1:5, imaginary = rep(c(1:5),c(5,5,5,5,5))), 5, 5, )
> w
     [,1] [,2] [,3] [,4] [,5]
[1,] 1+1i 1+2i 1+3i 1+4i 1+5i
[2,] 2+1i 2+2i 2+3i 2+4i 2+5i
[3,] 3+1i 3+2i 3+3i 3+4i 3+5i
[4,] 4+1i 4+2i 4+3i 4+4i 4+5i
[5,] 5+1i 5+2i 5+3i 5+4i 5+5i
> Conj(w)
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,] 1-1i 1-2i 1-3i 1-4i 1-5i
[2,] 2-1i 2-2i 2-3i 2-4i 2-5i
[3,] 3-1i 3-2i 3-3i 3-4i 3-5i
[4,] 4-1i 4-2i 4-3i 4-4i 4-5i
[5,] 5-1i 5-2i 5-3i 5-4i 5-5i
> w*Conj(w)
      [,1]  [,2]  [,3]  [,4]  [,5]
[1,]  2+0i  5+0i 10+0i 17+0i 26+0i
[2,]  5+0i  8+0i 13+0i 20+0i 29+0i
[3,] 10+0i 13+0i 18+0i 25+0i 34+0i
[4,] 17+0i 20+0i 25+0i 32+0i 41+0i
[5,] 26+0i 29+0i 34+0i 41+0i 50+0i
```

10.) *Write a description of the purpose of the xor (exclusive or) function based on results of the R command.*

```
> xor(c(F, F, T, T), c(F, T, F, T))
```

The R output is:

```
> xor(c(F, F, T, T), c(F, T, F, T))
[1] FALSE  TRUE  TRUE FALSE
```

The funciton exclusive or, xor, is returning 'x or y' and not 'x and y'. This is consistent with the output that we saw.

11.) *Write an R function named benford (after Benford's Law) that returns the leading digit of its single numeric argument. The leading digit of 365 is 3; the leading digit of 0.02432 is 2.*
The Benford Function:

```
function(x){
if(x>=1){
as.numeric(head(strsplit(as.character(x),''))[[1]],n=1))
}
else{
n = floor(nchar(x)/2)
floor((x%%1)*10^(n))
}
}
```

R Output:

```
> benford(0.00243)
[1] 2
> benford(365)
[1] 3
```

12.) *Discrete-event simulations involve a simulation clock, entities that pass through a system, attributes associated with the entities, and a calendar of future events. Consider a coffee shop with a single server. Currently, Arthur is being served at simulated time 57 minutes. Waiting in line behind Arthur, are Bert, Charise, and David. The even calendar contains, in order, the arrival of Emma at time 60, and the end of service of Arthur at time 62. The current state of the coffee shop is modeled by the five R commands*

```
> time       = 57
> coffee     = c("Arthur", "Bert", "Charise", "David")
> arrival     = c(50, 52, 55, 56)
> calendar  = c(60,62)
> event      = c("arrival", "end.of.service")
```

The object named time contains the simulated time. The vector named coffee contains the names of customers in the coffee shop as character strings, ordered by their positions in the queue. The vector named arrival contains the arrival times of the customer of the customer currently being served (in this case Arthur), and the customers currently waiting in line. The vector calendar contains the times of future events. The associated vector named event contains the names of the future events.

- *Write the R commands that update the data structures to process the arrival of Emma at time 60, and schedule the arrival of Flip at time 64.*

```
> time = 57
> coffee = c("Arthur", "Bert", "Charise", "David")
> arrival = c(50,52,55,56)
> calendar = c(60,62)
> event = c("arrival", "end.of.service")
> coffee
> time = 60
[1] "Arthur"  "Bert"    "Charise" "David"
> event = append(event, "arrival")
> event
[1] "arrival"        "end.of.service" "arrival"
> calendar = append(calendar, 64)
> calendar
[1] 60 62 64
> coffee = append(coffee, "Emily")
> coffee
[1] "Arthur"  "Bert"    "Charise" "David"   "Emily"
>
```

- *Write the R commands that update the data structures to process the end of service Arthur at time 62, and schedule the end of service of Bert at time 66.*

```
> time = 62
> coffee = coffee[-1]
> coffee
[1] "Bert"    "Charise" "David"   "Emily"
> event = append(event, "arrival")
> event
[1] "arrival"        "end.of.service" "arrival"
> event=event[-1]
> event
[1] "end.of.service" "arrival"
> calendar = calendar[-1]
> calendar
[1] 62 64
> calendar = append(calendar, 66)
> calendar
[1] 62 64 66
```