CSCI 520
Homework 8

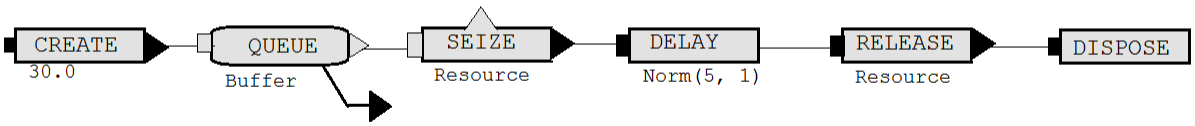
Megan Rose Bryant
Department of Mathematics
William and Mary

October 29, 2014

1. *Identify any typographical errors or suggestions in the first 22 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 taken from the bottom of the page.*

- Page 144, line 6. equal when → equal when the
- Page 148, line 11. set for in monospace
- Page 182, line 7. with out → without
- Page 182, line -11. States nations → States's national
- Page 184, line -13 and line -18. Paragraph indentation issues.
- Page 185, line 3. distance distance → distance
- Page 196. line 13. by inverse → by the inverse
- Page 197, line -15. There is not a trace function → A trace function is not
- Page 200, line -17. which is → which in

2. *Draw the following diagram using xfig and export it as an encapsulated postscript file. The use the LaTeX"includegraphics" command to display it in your homework solution.*



3. *Modify the sieve function to print all of the twin primes between* ***1*** *and* ***N****. The new fuction should be named* ***twin.sieve****. Two adjacent prime numbers are twin primes if they differ by 2.*
R Code:

```
twin.sieve <- function(N){
  prime = c(FALSE, rep(TRUE, N-1))
  for(n in 2:sqrt(N)){
    if (prime[n]) {
      for(s in 2:(N/n)){
        prime[s*n]=FALSE
      }
    }
  }
  x = which(prime[TRUE])
  y={}
  length = length(x)
```

```
  n = 2
  while(n<=length){
    if(x[n]-x[n-1] == 2){
      y = append(y,x[n-1])
      y = append(y,x[n])
    }
    n = n + 1
  }
  return(y)
}
```

R Output:

```
> twin.sieve(100)
 [1]  3  5  5  7 11 13 17 19 29 31 41 43 59 61 71 73
```

4. *Multiply each integer from 1 to 1000 by every integer from 1 to 1000. All one million of these products will have a leading digit that ranges from 1 to 9. Write an R program that calculates the fractions of occurrence of the various leading digits, and displays the results in an appropriate graphic.*

R Code:

```
fractions.occurrence <- function(N){
  x = matrix(1:(N*N), nrow = N, ncol = N)

  digitcount = matrix(c(1:9, rep(0,9)),nrow = 2, ncol = 9, byrow = TRUE)

  i = 1
  j = 1
  while(i <= N){
    j = 1
    while(j <= N){
      digit = benford(x[i,j])
      digitcount[2,digit] = digitcount[2,digit] + (1/1000000)
      j = j + 1
    }
    i = i + 1
  }
  print(sum(digitcount[2,]))
  return(digitcount)
}
```

R Output:

```
> fractions.occurrence(1000)
[1] 1
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]     [,9]
[1,] 1.000000 2.000000 3.000000 4.000000 5.000000 6.000000 7.000000 8.000000 9.000000
[2,] 0.111112 0.111111 0.111111 0.111111 0.111111 0.111111 0.111111 0.111111 0.111111
```

5. *The prime number theorem states that*

$$\lim_{n\to\infty} \frac{\pi(n)\ln(n)}{n} = 1$$

where $\pi(n)$ is the prime-counting function which gives a count of the number of prime numbers that are less than or equal to n for some positive integer n. It was first suggested by Carl Friedrich Gauss at the age of about 15 in 1792. Modify the sieve function to return a vector of length $\boldsymbol{N}$ *containing the values of the prime-counting function (that is, the n^{th} value in the vector returned by the new function contains a count of the number of primes that are less than or equal to n, for $n = 1, 2, \ldots, N$). Name the new function count.sieve. Use count.sieve to plot $\pi(n)\ln(n)/n$ for $n = 1, 2, \ldots, 1000000$ to assess the plausibility of the prime number theorem.*
Modified Sieve Function:

```
sieve <- function(N){
  prime = c(FALSE, rep(TRUE, N-1))
  for(n in 2:sqrt(N)){
    if (prime[n]) {
      for(s in 2:(N/n)){
        prime[s*n]=FALSE
      }
    }
  }
  y = which(prime[TRUE])
  return(y)
}
```

Count.Sieve Function (With Internal Plotting):

```
count.sieve <- function(N){
plot.new()

prime = c(FALSE, rep(TRUE, N-1))
for(n in 2:sqrt(N)){
  if (prime[n]) {
    for(s in 2:(N/n)){
      prime[s*n]=FALSE
    }
  }
}

p = which(prime[TRUE])
l = length(p)
reimannpi= {}
i = 1
print(l)

while(i <= N){
```

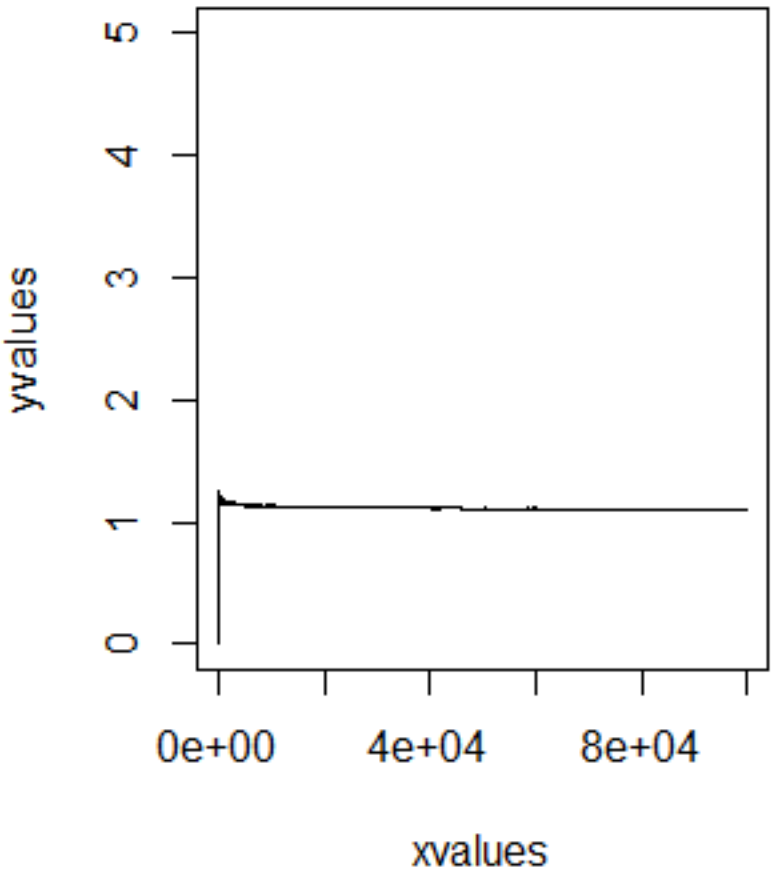
```
  n = 1
  count = 0
  while(n <= l){
    if(p[n] < i){
      count = count + 1
    }
    n = n + 1
  }
  reimannpi[i] = count
  i = i + 1
}

x = 1
xvalues = {}
yvalues = {}
while(x <= N){
  y = (reimannpi[x]*log(x)) / x
  xvalues = append(xvalues, x)
  yvalues = append(yvalues, y)
  x = x + 1
}

plot(xvalues, yvalues, xlim = c(0, N), ylim = c(0,5), type = "l")
}
```

Running the program at n = 100000 obtains the following graph, which supports the theory that the limit of the function is 1.

Graph:



6. *Marylou's class has 34 students. Find the probability that there are exactly two birthday matches, with each birthday match consisting of exactly to students. Ignore leap years and assume that all 365 birthdays are equally likely. Support your analytic solution by conducting a Monte Carlo simulation in R.*

Analytics: We know from basic probability that the probability that exactly one pair shares a birthday is

$$P_n(1) = \frac{365 * \binom{n}{1} \times \binom{365-1}{n-2} * (n-2)!}{365^n}$$

Therefore, we can infer thant the probability that there are exactly two pairs sharing a birthday is

$$P_n(2) = \frac{\binom{365}{2} * \binom{n}{2} * \binom{n-2}{2} * \binom{365-2}{n-4}(n-4)!}{365^n}$$

We extrapolate this to a generic formula

$$P_n(k) = \frac{\binom{365}{k} * \binom{n}{2k} * (2k)! * \binom{365-k}{n-2k} * (n-2k)!}{2^k * 365^n}$$

Using the values of 34 and 2 for n and k respectively, we solve for the following

$$P_{34}(2) = \frac{\binom{365}{2} * \binom{34}{2*2} * (2*2)! * \binom{365-2}{34-2*2} * (34-2*2)!}{2^2 * 365^3 4} = .25$$

Therefore, the probability that there are exactly two birthday matches with each bithday match consisting of exactly two students is approximately 25%.
We can solve this analytically using the following R code.
R Code:

```
birthday <- function(n,k){
  (choose(365,k) * perm(n, 2* k) * perm(365-k, n - 2 *k)) / (2^k * 365^n)
}
```

R Output:

```
> birthday(34,2)
[1] 0.2575813
```

R Code:

```
perm <- function(n,k){
  choose(n,k) * factorial(k)
}
birthday <- function(n,k){
  (choose(365,k) * perm(n, 2* k) * perm(365-k, n - 2 *k)) / (2^k * 365^n)
}
```

R Output:

```
> birthday(34,2)
[1] 0.2575813
```

Monte Carlo Simulation Code:

```
birthdaysim <- function(n,k){
  time = 1
  globalcount = 0
  while(time <= k){
    x = sample(1:365, n, replace=T)
    i = 1
    count = 0
    while(i <= n){
      j = i + 1
      while(j <= n){
```

```
        if(x[i] == x[j]){
          count = count + 1
          j = j + 1
        }
      j = j + 1
      }
    i = i + 1
    }
  if(count == 2) globalcount = globalcount + 1
  time = time + 1
  }
  return(globalcount/k)
}
```

Simulation Output:

```
> birthdaysim(34,10000)
[1] 0.2604
```

We see from the Monte Carlo simulation that the proportion of matches was .2604. This supports our analytic solution.

7. *Let A, B, and C be the results of three rolls of a fair die.*

(a) *Write an R program that computes the exact probability that the roots of the random quadratic equation* $Ax^2 + Bx + C = 0$ *are real. (A single root with multiplicity two, for example the root* $x =$ *-1 associated with* $A = 1$, $B = 2$, *and* $C = 1$ *counts as real roots).*
For the roots to be real, we know that the following must hold true

$$B^2 - 4AC \geq 0$$

which implies that

$$B^2 \geq 4AC$$

We can solve this analytically for the following

```
> inoptions = 6 * 6 * 6
> count = 0
> for (A in 1:6)
+   for (B in 1:6)
+     for (C in 1:6)
+       if (B ^ 2 - 4 * A * C >= 0) count = count + 1
> count / inoptions
[1] 0.1990741
```

(b) *Support your analytic solution with a Monte Carlo Simulation.*
Monte Carlo Simulation Code:

```
fairdice <- function(n){
  i = 1
  yes = 0
  while(i <= n){
    rolls = sample(1:6, 3, replace=T)
    A = rolls[1]
    B = rolls[2]
    C = rolls[3]
    if(B^2 >= 4 * A * C) yes = yes + 1
    i = i + 1
  }
  print(yes/n)
}
```

Simulation Results:

```
> fairdice(10000)
[1] 0.2029
```

This supports our analytic results.

8. *The cubic equation*

$$ax^3 + bx^2 + cx + d = 0$$

with $a \neq 0$ *having discriminant*

$$\delta = 18abcd - 4b^3d + b^2c^2 - 4ac^3 - 27a^2d^2$$

has three distinct real roots if $\delta > 0$*, has one real root with multiplicity one and another real root with multiplicity two if* $\delta = 0$*, and has one real root with multiplicity one and two complex conjugate roots when* $\delta < 0$*. If the coefficients a, b, c, and d are mutually independent U(0,3) random variables, use Monte Carlo Simulation to estimate the probability that all of the roots are real to three digits.*
R Code, Analytic Solution:

```
noptions = 4 * 4 * 4 * 4
count = 0
for (A in 0:3)
  for (B in 0:3)
    for (C in 0:3)
      for (D in 0:3)
      if (18 * A * B * C * D - 4 * B^3 * D + B^2 * C^2 -4 * A * C^3 - 27 * A^2 * D^2 >= 0) cou
count / noptions
```

Analytic Output:

```
[1] 0.1914062
```

R Code, Monte Carlo Simulation:

```
nrep = 400000
count = 0
for (i in 1:nrep) {
  A = sample(0:3, 1)
  B = sample(0:3, 1)
  C = sample(0:3, 1)
  D = sample(0:3, 1)
  if(18 * A * B * C * D - 4 * B^3 * D + B^2 * C^2 -4 * A * C^3 - 27 * A^2 * D^2 >= 0) count = c
}
count / nrep
```

Simulation Output:

```
[1] 0.1926925
```

The analytic and the Monte Carlo Simulations concur that the probability that are roots are real is approximately 19.3%.

9. *Write a function named antiidentity with a single integer-valued argument n that returns an $n \times n$ matrix with diagonal elements of 0 and off-diagonal elements of 1.*
R Code:

```
antiidentity <- function(n){
  x = matrix(nrow = n, ncol = n, 1)
  i = 1
  j = 1
  while(i <= n){
    while(j <= n){
      x[i,j] = 0
      i = i + 1
      j = j + 1
    }
  }
  return(x)
}
```

Test Output:

```
> antiidentity(2)
     [,1] [,2]
[1,]    0    1
[2,]    1    0
> antiidentity(3)
     [,1] [,2] [,3]
[1,]    0    1    1
[2,]    1    0    1
```

```
[3,]    1    1    0
> antiidentity(4)
     [,1] [,2] [,3] [,4]
[1,]    0    1    1    1
[2,]    1    0    1    1
[3,]    1    1    0    1
[4,]    1    1    1    0
```

10. *Write an R command that creates a 5×5 matrix with elements that are the row number raised to the column number power, that is, the (i, j) element is i^j, for $i = 1, 2, \ldots 5$ and $j = 1, 2, \ldots, 5$.*
R Code:

```
x = matrix(nrow = 5, ncol = 5)
i = 1
while(i <= 5){
  j = 1
  while(j <= 5){
    x[i,j] = i^j
    j = j + 1
  }
  i = i + 1
}
```

R Output:

```
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    1    1    1    1
[2,]    2    4    8   16   32
[3,]    3    9   27   81  243
[4,]    4   16   64  256 1024
[5,]    5   25  125  625 3125
```

11. *Write an R function named sum.list with a single argument x, which is a list consisting of entirely atomic components, that returns the sum of all elements in each component that are numeric.*
R Function Code:

```
sumlist <- function(list){
  n = length(list)
  i = 1
  output = 0
  while(i <= n){
    if(is.numeric(x[[i]])){
      output = output + sum(x[[i]])
    }
    i = i + 1
```

```
  }
  return(output)
}
```

Test Output:

```
> x
[[1]]
[1] 2 3 5

[[2]]
[1] "aa" "bb" "cc" "dd" "ee"

[[3]]
[1]  TRUE FALSE  TRUE FALSE FALSE

[[4]]
[1] 3

> sumlist(x)
[1] 13
```