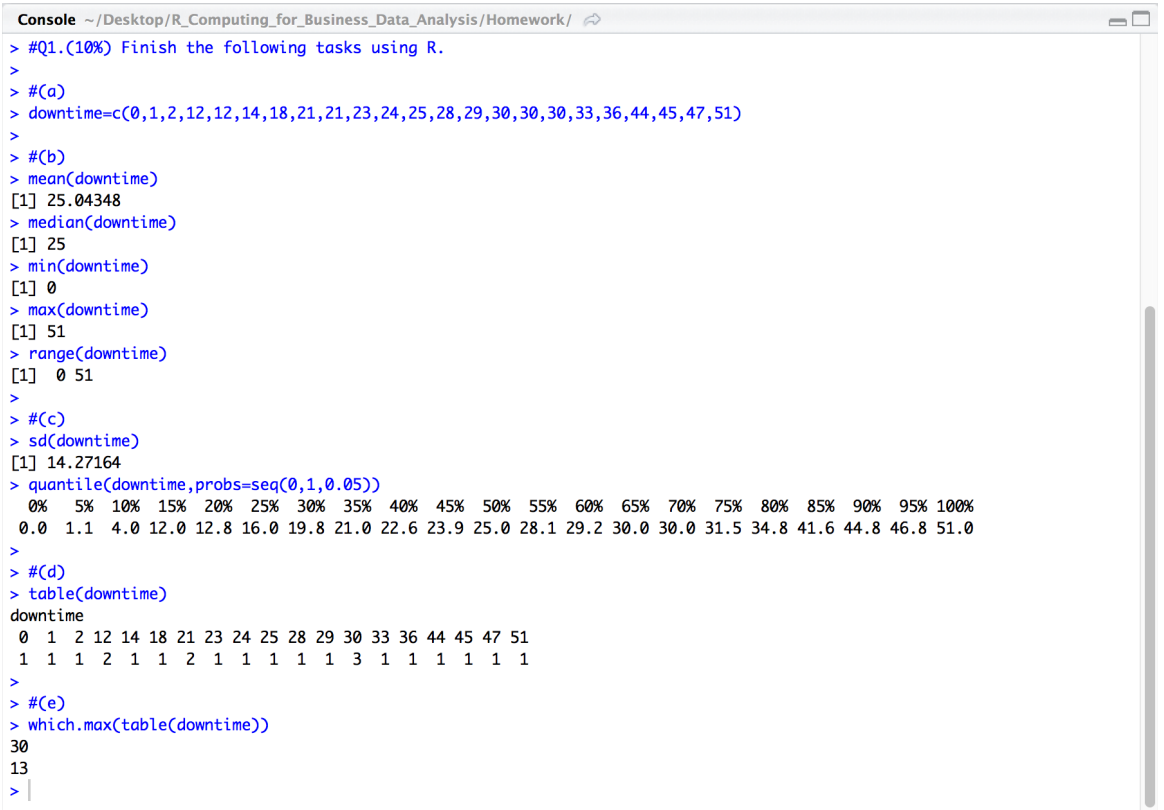
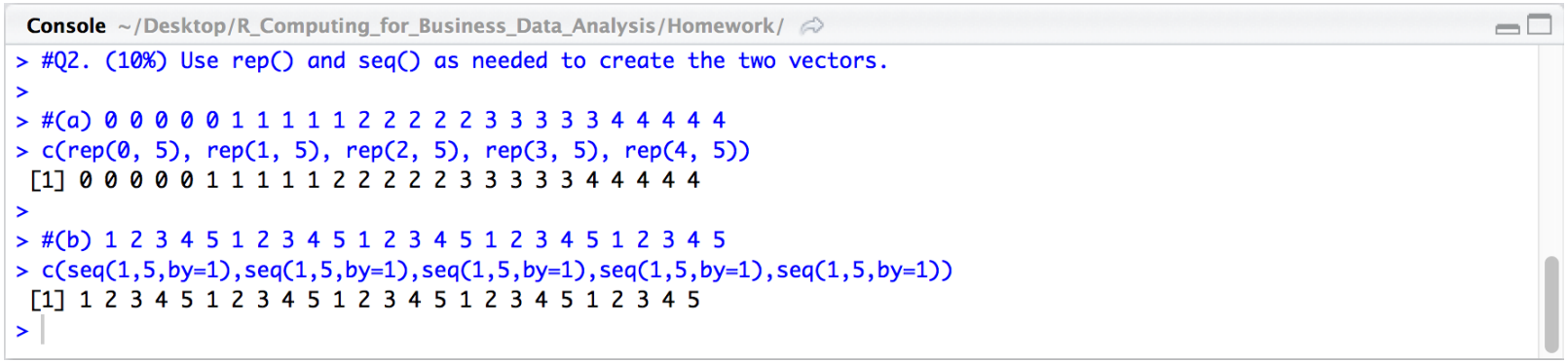
Q1. (10%) Finish the following tasks using *R*.
(a) Create a vector called **downtime**. The vector should contain the following numbers: 0, 1, 2, 12, 12, 14, 18, 21, 21, 23, 24, 25, 28, 29, 30, 30, 30, 33, 36, 44, 45, 47, and 51.
(b) Calculate the mean, median, min, max, and range of **downtime**.
(c) Calculate the standard deviation, 5 percentile, and 95 percentile of downtime (Hint: Use the *quantile* function).
(d) What is the most frequent number? What is the frequency? (Hint: Use table( ))
(e) Use *which*( ) to take the most frequent number from the **downtime** vector.
P.S.: Show all the *R* functions that you use.



Q2. (10%) Use *rep*( ) and *seq*( ) as needed to create the two vectors.

(a) 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4
(b) 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5



Q3. (10%)
(a) Create a 4x3 matrix that stores the values below. Also name each column correctly (x, y, z)

(b) Display the row 1, column 3 element of the matrix.

Console ~/Desktop/R_Computing_for_Business_Data_Analysis/Homework/

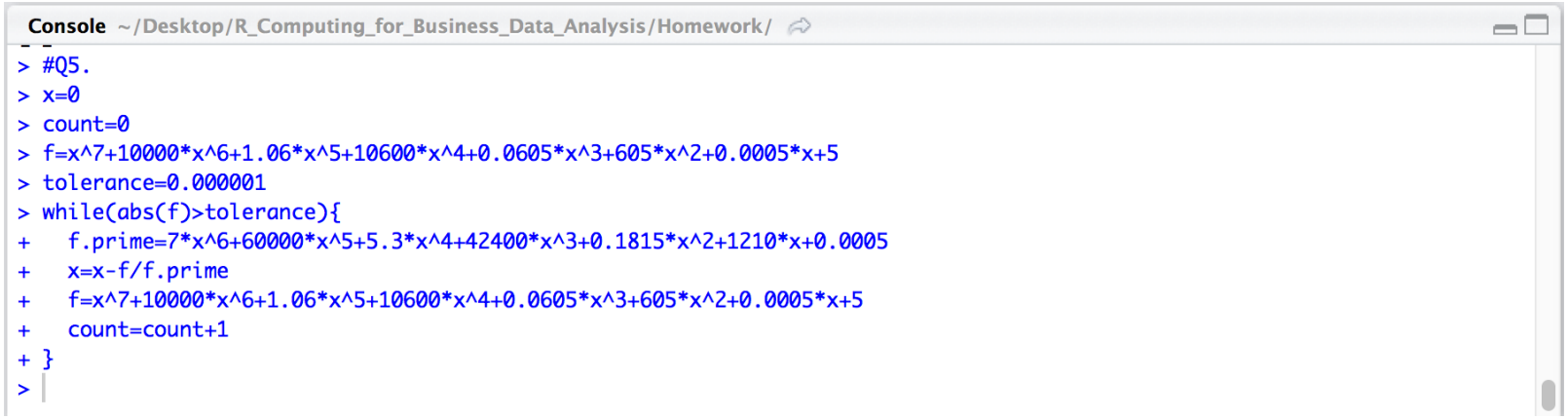
```
> #Q3.
>
> #(a)
> inputdata=c(61,175,111,124,13,21,24,23,4,18,14,18)
> A=matrix(inputdata, ncol = 3,dimnames = list(NULL,c(" x","y","z")))
> A
      x  y  z
[1,]  61 13  4
[2,] 175 21 18
[3,] 111 24 14
[4,] 124 23 18
> #(b)
> A[1,3]
z
4
>
```

Q4. (10%) Calculate $\sum_{i=1}^{N} 1/i$, and compare with $\log(N)+0.6$ for $N = 500, 2000, 8000$.

Console ~/Desktop/R_Computing_for_Business_Data_Analysis/Homework/

```
> #Q4. (10%) Calculate Sigma_(i=1)^N(1/i), and compare with log(N)+0.6 for N = 500, 2000, 8000
>
> sum.frac = function (n) {
+   result = 0
+   for (i in 1:n) {
+     result = result + (1/i)
+   }
+   result
+ }
> log.add06 = function(n) {
+   log(n) + 0.6
+ }
> compare = function(n){
+   sum.frac(n) > log.add06(n)
+ }
> sum.frac(500)
[1] 6.792823
> sum.frac(2000)
[1] 8.178368
> sum.frac(8000)
[1] 9.564475
>
> log.add06(500)
[1] 6.814608
> log.add06(2000)
[1] 8.200902
> log.add06(8000)
[1] 9.587197
>
> compare(500)
[1] FALSE
> compare(2000)
[1] FALSE
> compare(8000)
[1] FALSE
>
```

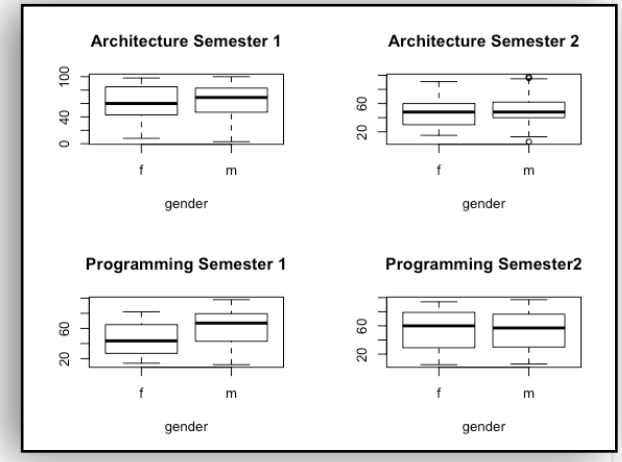
Q5. (10%) The equation $x^7+10000x^6+1.06x^5+10600x^4+0.0605x^3+605x^2+0.0005x+5$ has exactly one real root. Write an *R* program to find the root. What is the root? How many iterations of Newton's method are required to find this root if the initial guess is



Q6. (10%) Based on the *results.txt* file (in Lecture 1), write *R* code to reproduce the graph below.

Console ~/Desktop/R_Computing_for_Business_Data_Analysis/Homework/

```
> #Q6. (10%) Based on the results.txt file (in Lecture 1), write R code to reproduce the graph below.
>
> result = read.table("/Users/flisshou/Desktop/R_Computing_for_Business_Data_Analysis/results.txt", header = T)
> attach(result)
The following objects are masked from result (pos = 3):

    arch1, arch2, gender, prog1, prog2

The following objects are masked from result (pos = 4):

    arch1, arch2, gender, prog1, prog2

The following objects are masked from result (pos = 5):

    arch1, arch2, gender, prog1, prog2

> names(result)
[1] "gender" "arch1"  "prog1"  "arch2"  "prog2"
> par(mfrow = c(2, 2))
>
> #arch1
> boxplot(arch1~gender, xlab = "gender", main = "Architecture Semester 1")
> #arch2
> boxplot(arch2~gender, xlab = "gender", main = "Architecture Semester 2")
> #arch3
> boxplot(prog1~gender, xlab = "gender", main = "Programming Semester 1")
> #arch4
> boxplot(prog2~gender, xlab = "gender", main = "Programming Semester2")
>
```

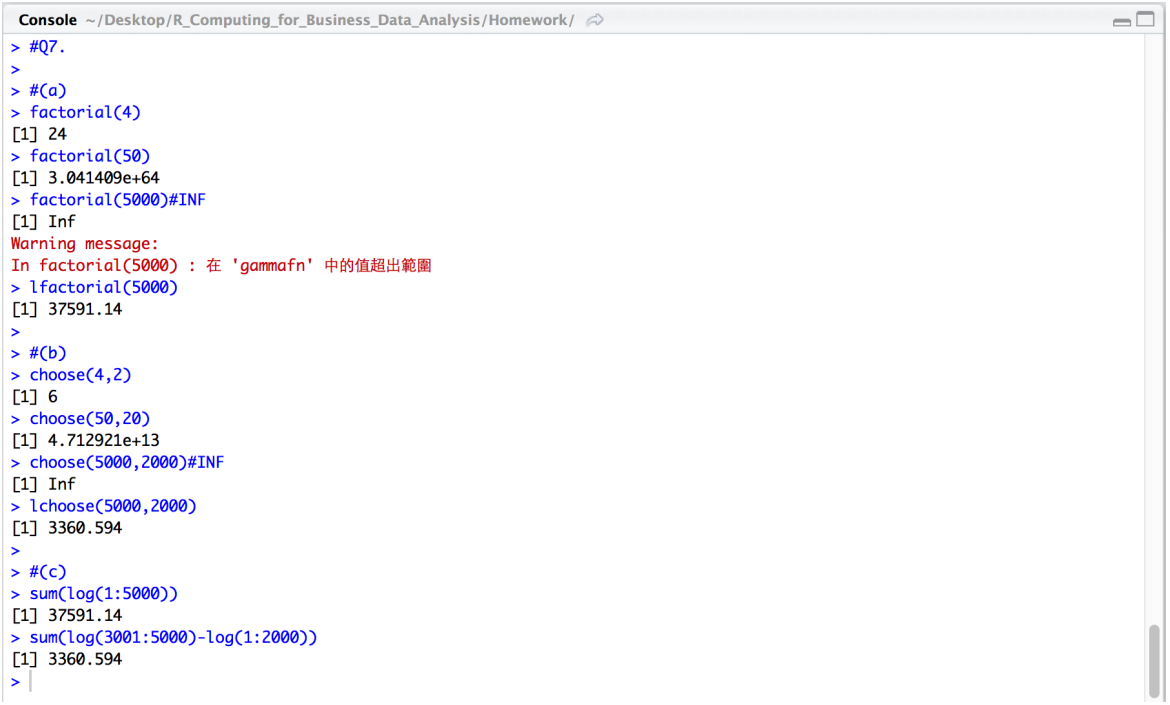


Q7. (10%)

(a) Compute 4!, 50!, and 5000! (Hint: Use the *factorial* function)

(b) Compute $\binom{4}{2}, \binom{50}{20}$, and $\binom{5000}{2000}$

(c) The *factorial* function tends to return Infinity when its argument is large. To tackle this, apply *log*( ) and *sum*( ) to compute 5000! and $\binom{5000}{2000}$. Express your answers in terms of $e^{?}$.



Q8. (10%)
(a) Use *R* to create a *vector* that contains all integers (整數) from 1 to 100 that are NOT divisible by 2, 3, or 7.
Do NOT use loops (Hint: *which*( ) will help).
(b) Create a 10by10 *identity matrix*. That is, all diagonal (對角) elements are 1 and all remaining elements are 0 (Hint: *diag*( ) will help).
Then use two different ways to make all the non-zero elements 5 (Hint: the first can be *diag*( ) and the second can be *which*( ) + *logical comparisons*).

```
Console ~/Desktop/R_Computing_for_Business_Data_Analysis/Homework/
> #Q8. (10%)
>
> #(a)
> Q8a=c(1:100)
> Q8a[which(Q8a%%2!=0 & Q8a%%3!=0 & Q8a%%7!=0)]
 [1]  1  5 11 13 17 19 23 25 29 31 37 41 43 47 53 55 59 61 65 67 71 73 79 83 85 89 95 97
>
> #(b)
> Q8b=diag(10)
> Q8b
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    0    0    0    0    0    0    0    0     0
 [2,]    0    1    0    0    0    0    0    0    0     0
 [3,]    0    0    1    0    0    0    0    0    0     0
 [4,]    0    0    0    1    0    0    0    0    0     0
 [5,]    0    0    0    0    1    0    0    0    0     0
 [6,]    0    0    0    0    0    1    0    0    0     0
 [7,]    0    0    0    0    0    0    1    0    0     0
 [8,]    0    0    0    0    0    0    0    1    0     0
 [9,]    0    0    0    0    0    0    0    0    1     0
[10,]    0    0    0    0    0    0    0    0    0     1
> #Then use two different ways to make all the non-zero elements 5
> #(Hint: the first can be diag( ) and the second can be which( ) + logical comparisons).
>
> #SOLUTION1:
> Q8b1=diag(5,10,10)
> Q8b1
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    5    0    0    0    0    0    0    0    0     0
 [2,]    0    5    0    0    0    0    0    0    0     0
 [3,]    0    0    5    0    0    0    0    0    0     0
 [4,]    0    0    0    5    0    0    0    0    0     0
 [5,]    0    0    0    0    5    0    0    0    0     0
 [6,]    0    0    0    0    0    5    0    0    0     0
 [7,]    0    0    0    0    0    0    5    0    0     0
 [8,]    0    0    0    0    0    0    0    5    0     0
 [9,]    0    0    0    0    0    0    0    0    5     0
[10,]    0    0    0    0    0    0    0    0    0     5
>
```

```
> #SOLUTION2:
> Q8b[which(Q8b==1)]=5
> Q8b
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    5    0    0    0    0    0    0    0    0     0
 [2,]    0    5    0    0    0    0    0    0    0     0
 [3,]    0    0    5    0    0    0    0    0    0     0
 [4,]    0    0    0    5    0    0    0    0    0     0
 [5,]    0    0    0    0    5    0    0    0    0     0
 [6,]    0    0    0    0    0    5    0    0    0     0
 [7,]    0    0    0    0    0    0    5    0    0     0
 [8,]    0    0    0    0    0    0    0    5    0     0
 [9,]    0    0    0    0    0    0    0    0    5     0
[10,]    0    0    0    0    0    0    0    0    0     5
>
```

Q9. (10%) Use *while*( ) to write an *R* function that prints out all prime numbers <= *n* (where *n* is an integer). After writing the function, set *n*=100 and show me the results.

Console ~/Desktop/R_Computing_for_Business_Data_Analysis/Homework/

```
> #Q9. (10%) Use while( ) to write an R function that prints out all prime numbers <= n(where n
is an integer).
> #After writing the function, set n=100 and show me the results.
>
> primes=function(n) {
+   result=NULL
+   i=2
+   while( i <= n){
+     if(!any(i%%result==0)){
+       result=c(result,i)
+     }
+     i=i+1
+   }
+   result
+ }
> primes(100)
 [1]  2  3  5  7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Q10. (10%) Consider the function $y=f(x)$ defined by $y = f(x) = \begin{cases} -x^3, \forall x \le 0 \\ x^2, \forall x \in (0,1] \\ \sqrt{x}, \forall x > 1 \end{cases}$.

Write an *R* function to calculate *y* using *if* statements.

Generate the following plot for x=seq(-2, 2, 0.1).

Console ~/Desktop/R_Computing_for_Business_Data_Analysis/Homework/

```
> #Q10. Consider the function y=f(x) defined by
> #y=f(x)= <1> -x^3, for all x<=0; <2> x^2, for all x belongs to (0,1]; <3> x^(1/2), for all x>1.
> #Write an R function to calculate y using if-statements.
> #Generate the following plot for x=swq(-2, 2, 0.1).
>
> f = function(x) {
+   y = 0
+
+   if (x > 1) {
+     y = sqrt(x)
+   } else if (x <= 0) {
+     y = (-1) * x^3
+   } else{
+     y = x^2
+   }
+
+   y
+ }
>
> datas = seq(-2, 2, 0.1)
> results = rep(0, length(datas))
> for (i in 1:length(datas)){
+   results[i] = f(datas[i])
+ }
>
> plot(datas, results, type = 'l', xlab = 'x', ylab = 'f(x)')
>
```