3B Assignment Solutions

```
# Assignment 3B

# get the path for the working directory
getwd()
#set the working directory by assigning the path for the file
setwd("C:/Users/Neha/Desktop")
#load the csv file
BirdStrikes<-read.csv("Bird Strikes.csv", header = T, sep = ",",
quote="\"")
Bird.data.new <- read.table("Bird Strikes.csv", header = T, sep = ",",
quote = "\"", fill = T)
```

get the path of the working directory

set the working directory by assigning the path of the file

load the CSV file

Problem 1

```
# Problem 1

# Defining the function UnReportedStrikes: to obtain the bird strikes
not reported
# Input for this function is Bird Strikes data file
UnReportedStrikes <- function (UnReportedStrikes) {
# obtaining the total no of unreported strikes from the appropriate
column
UnReportedStrikes <-sum(UnReportedStrikes$Reported=="")
# To get the output as the sum of total no of unreported bird strikes
cat("The number of bird strikes which were unreported are")
cat("\n")
  print(UnReportedStrikes)
}
UnReportedStrikes(Bird.data.new)
```

Stratergy:

1. *Defining the function UnReportedStrikes: to obtain the bird strikes not reported*
2. *Input for this function is Bird Strikes data file*
3. *obtaining the total no of unreported strikes from the appropriate column*
4. *To get the output as the sum of total no of unreported bird strikes*

Output: Output is the number of bird strikes which were reported

3B Assignment Solutions

```
> UnReportedStrikes(Bird.data.new)
The number of bird strikes which were unreported are
[1] 71537
```

```
# Problem 2

#Defining the function: MostBirdStrikes to obtain the year which had
the most number of bird strikes
# Input for this function is the Bird Strikes data
MostBrirdStrikes<- function(birdstrikescount_year)  {
Bird.Strike.data <- birdstrikescount_year
  # Reading the dates to get the desired output
  birdstrikes_date<-as.Date(Bird.Strike.data$FlightDate,
format="%m/%d/%Y")
# obtaining the desired format for the dates
  birdstrikescount_year<-table(format(birdstrikes_date, "%Y"))
# arranging the table in decending order
birdstrikescount_year<-sort( birdstrikescount_year,dec = T)
# To get the output and printing it out
cat("The year that had the most Bird Strikes was")
cat("\n")
print(birdstrikescount_year[1])
}
 MostBrirdStrikes(Bird.data.new)
```

Stratergy:

1. *Defining the function: MostBirdStrikes to obtain the year which had the most number of bird strikes*
2. *Input for this function is the Bird Strikes data*
3. *Reading the dates to get the desired output*
4. *obtaining the desired format for the dates*
5. *arranging the table in decending order*
6. *To get the output and printing it out*

Output:

The year that had the most Bird Strikes

```
> MostBrirdStrikes(Bird.data.new)
The year that had the most Bird Strikes was
 2010
10923
```

3B Assignment Solutions

```
# Problem 3
# Definig the function: StrikesInAYear to count the total number of
strikes per year
# Input for the function is the Bird Strike data file
StrikesInAYear <- function (BirdStrikesPerYear) {
# to get the desired column from the data set
Strike.year.data <- BirdStrikesPerYear
  BirdStrikesPerYear <- as.Date(Strike.year.data$FlightDate,
format="%m/%d/%Y")
# arranging the obtained data into a tabular column
BirdStrikes.Per.Year <- table(format(BirdStrikesPerYear, "%Y"))
# Placing the results in a Data Frame
BirdStrikesPerYear.data <- as.data.frame(BirdStrikes.Per.Year)
# TO get the output and print results
cat("Number of Bird Strikes for each year in a Data Frame")
cat("\n")
print(BirdStrikesPerYear.data)

}

StrikesInAYear(Bird.data.new)
```

Stratergy:

1. *Definig the function: StrikesInAYear to count the total number of strikes per year*
2. *Input for the function is the Bird Strike data file*
3. *to get the desired column from the data set*
4. *arranging the obtained data into a tabular column*
5. *Placing the results in a Data Frame*
6. *TO get the output and print results*

Output:

The no of strikes for each year in Data Frame

```
> StrikesInAYear(Bird.data.new)
Number of Bird Strikes for each year in a Data Frame
   Var1  Freq
1  2000  6407
2  2001  6287
3  2002  6769
4  2003  6664
5  2004  7667
6  2005  7804
7  2006  8010
8  2007  8746
9  2008  8903
10 2009 10741
11 2010 10923
12 2011 10483
```

3B Assignment Solutions

```
# Problem 4

# Defining function AirlineWithMOstStrike
# Input for the function would be Bird Strike data
AirlineWithMostStrike <- function(Maxstrikes){
Maxstrikes.data  <- Maxstrikes
# to get the data from data set
Max.Strikes <- table(Maxstrikes.data[15])
#view(Max.Strikes)
print(Max.Strikes)
# To Storing the function in a variable and printing in the next line
MaxBirdStrikes(Max.Strikes)
}

# defining function MaxBirdStrikes to pass the data frame generated in
the above function
MaxBirdStrikes <- function(Max.Strikes){
  # placing the reults in a data frame
  MaxStrikes.Data.Frame <- Max.Strikes
  MaxStrikesDataFrame <- as.data.frame(MaxStrikes.Data.Frame)
  #To sort the data frame in decreasing order
  MaxNoAirlines<- MaxStrikesDataFrame[with(MaxStrikesDataFrame,
order(-Freq)), ]
  # To get the second max strikes as the first might have missing
values
  SecondMaxStrikes<- MaxNoAirlines[2,]
  # get the output and print the output
  cat("The airline that has the maximum bird strike without missing
values in it")
  cat("\n")
  print(SecondMaxStrikes)
}

AirlineWithMostStrike(Bird.data.new)
```

Stratergy:

1. *Defining function AirlineWithMOstStrike*
2. *Input for the function would be Bird Strike data*
3. *to get the data from data set*
4. *placing the reults in a data frame*
5. *To Storing the function in a variable and printing in the next line*

6. *defining function MaxBirdStrikes to pass the data frame generated in the above function*
7. *placing the reults in a data frame*
8. *To sort the data frame in decreasing order*
9. *To get the second max strikes as the first might have missing values get the output and print the output*

Output:

The airline that has the maximum bird strike without missing values in it

```
> AirlineWithMostStrike(Bird.data.new)
The airline that has the maximum bird strike without missing values in it
        Var1 Freq
241 MILITARY 9193
```

Problem 5

Time Complexity based on graphs and Regression line
The analysis shows that for the input size for different funtions increases linearly, the time complexity would be linearly dependent.

	V1	V2	V3
1	0.03	0.07	0.13
2	1.75	2.52	5.86
3	1.69	2.90	5.88
4	0.08	0.14	0.19

The above matrix shows that al the data obtained is linealy dependent and has been increasing similarly

For problem 1 :
We can see that the values increase linearly with input size. The time complexity would be O(n)

For problem 2:
We can see that the values increase linearly with input size. The time complexity would be O(n)

For problem 3:
We can see that the values increase linearly with input size. The time complexity would be O(n)

For problem 4:

3B Assignment Solutions

We can see that the values increase linearly with input size. The time complexity would be O(n)

As it is increasing linearly for all input sizes it would also increase for 2X, 10X, 100X, 1000X

Space Complexity based on graphs and Regression line:
The analysis shows that for the input size for different funtions increases linearly, the space complexity would be linearly dependent.

For all the problem 1 to 4
The memory size increses linearly, the space complexity would also increase linearly.

```
# Problem 6

# double the input size
double <- rbind(Bird.data.new,Bird.data.new)
#quadraple the input size
quad <- rbind(double,double)
# put all kinds of input sizes into a data object
DataObject <- list(Bird.data.new,double,quad)
#give the matrix of all the outputs from the loop
time <- matrix(nrow=4,ncol =length(DataObject))
for(i in 1: length(DataObject)){
  #calculate elapsed time for different sizes of input for a mentioned function
  time[1,i] <- system.time(UnReportedStrikes(DataObject[[i]]))[3]
  #calculate elapsed time for different sizes of input for a mentioned function
  time[2,i] <- system.time(MostBrirdStrikes(DataObject[[i]]))[3]
  #calculate elapsed time for different sizes of input for a mentioned function
  time[3,i] <- system.time(StrikesInAYear(DataObject[[i]]))[3]
  #calculate elapsed time for different sizes of input for a mentioned function
  time[4,i] <- system.time(AirlineWithMostStrike(DataObject[[i]]))[3]
}

View(time)

#give sizes of input data
DataSize = c(1,2,4)
#plot and check the effect of different input sizes on time complexity for function UnReportedStrikes and then plot a regression line
function1 <- plot(DataSize,time[1,],type="b", main = "UnReportedStrikes", xlim = c(0.5,4.5), ylim = c(0.03,0.13), xlab = "Data size", ylab = "elapsed time")
```

3B Assignment Solutions

```
abline(lm(time[1,]~DataSize), col="blue")
#plot and check the effect of different input sizes on time complexity
for function MostBrirdStrikes and then plot a regression line
function2 <- plot(DataSize,time[2,],type="b",, main =
"MostBrirdStrikes", xlim = c(0.5,4.5), ylim = c(1.64,5.8), xlab =
"Data size", ylab = "elapsed time")
abline(lm(time[2,]~DataSize), col="blue")
#plot and check the effect of different input sizes on time complexity
for function StrikesInAYear and then plot a regression line
function3 <- plot(DataSize,time[3,],type="b", main = "StrikesInAYear",
xlim = c(0.5,4.5), ylim = c(1.5,5.8), xlab = "Data size", ylab =
"elapsed time")
abline(lm(time[3,]~DataSize), col="blue")
#plot and check the effect of different input sizes on time complexity
for function AirlineWithMostStrike and then plot a regression line
function4 <- plot(DataSize,time[4,],type="b", main =
"AirlineWithMostStrike", xlim = c(0.5,4.5), ylim = c(0.08,0.20), xlab
= "Data size", ylab = "elapsed time")
abline(lm(time[4,]~DataSize), col="blue")
```

Stratergy:

1. *double the input size*
2. *quadraple the input size*
3. *put all kinds of input sizes into a data object*
4. *give the matrix of all the outputs from the loop*
5. *calculate elapsed time for different sizes of input for a mentioned function*

6. *calculate elapsed time for different sizes of input for a mentioned function*

7. *calculate elapsed time for different sizes of input for a mentioned function*
8. *calculate elapsed time for different sizes of input for a mentioned function*

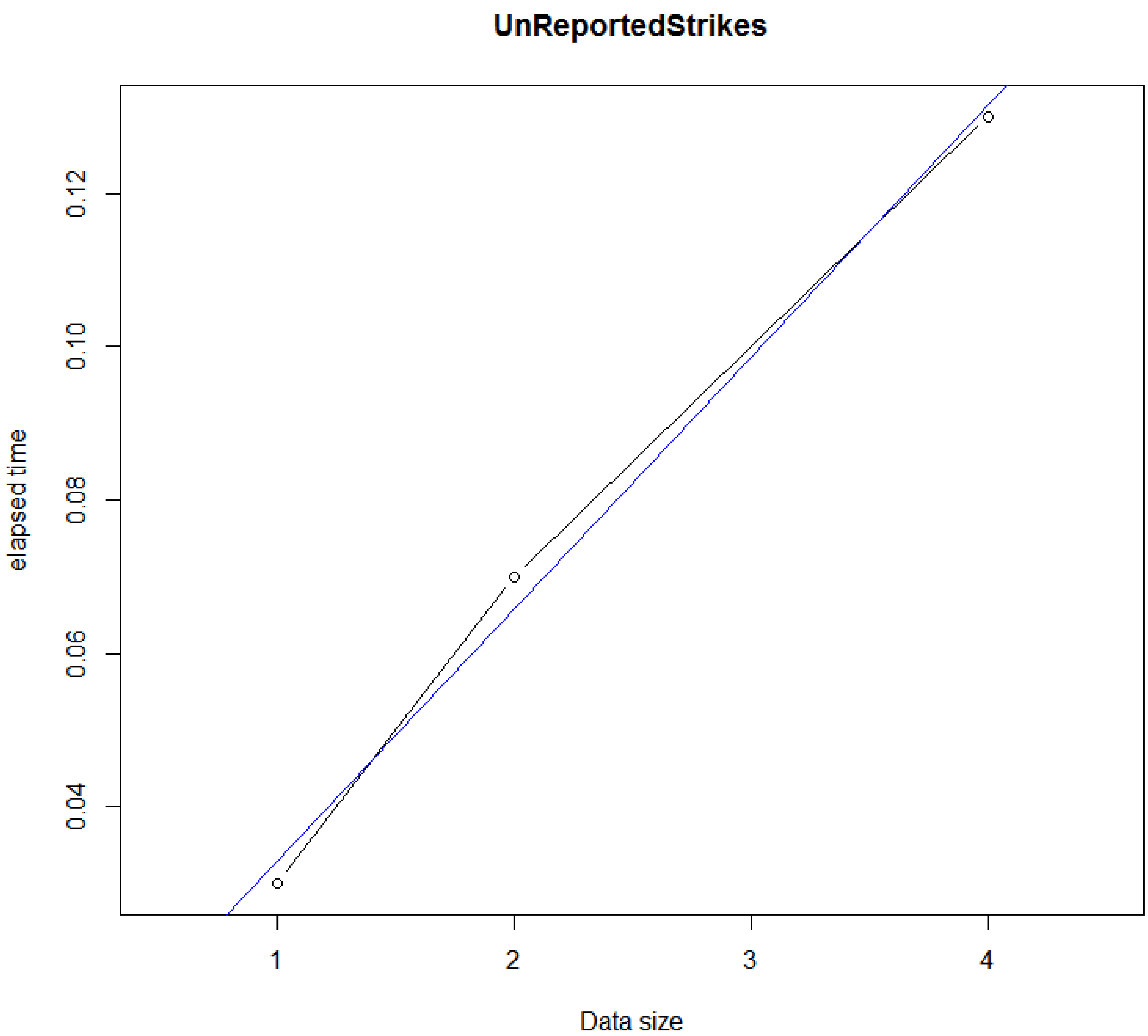
9. *give sizes of input data*
10. *plot and check the effect of different input sizes on time complexity for function UnReportedStrikes and then plot a regression line*
11. *plot and check the effect of different input sizes on time complexity for function MostBrirdStrikes and then plot a regression line*
12. *plot and check the effect of different input sizes on time complexity for function StrikesInAYear and then plot a regression line*

3B Assignment Solutions

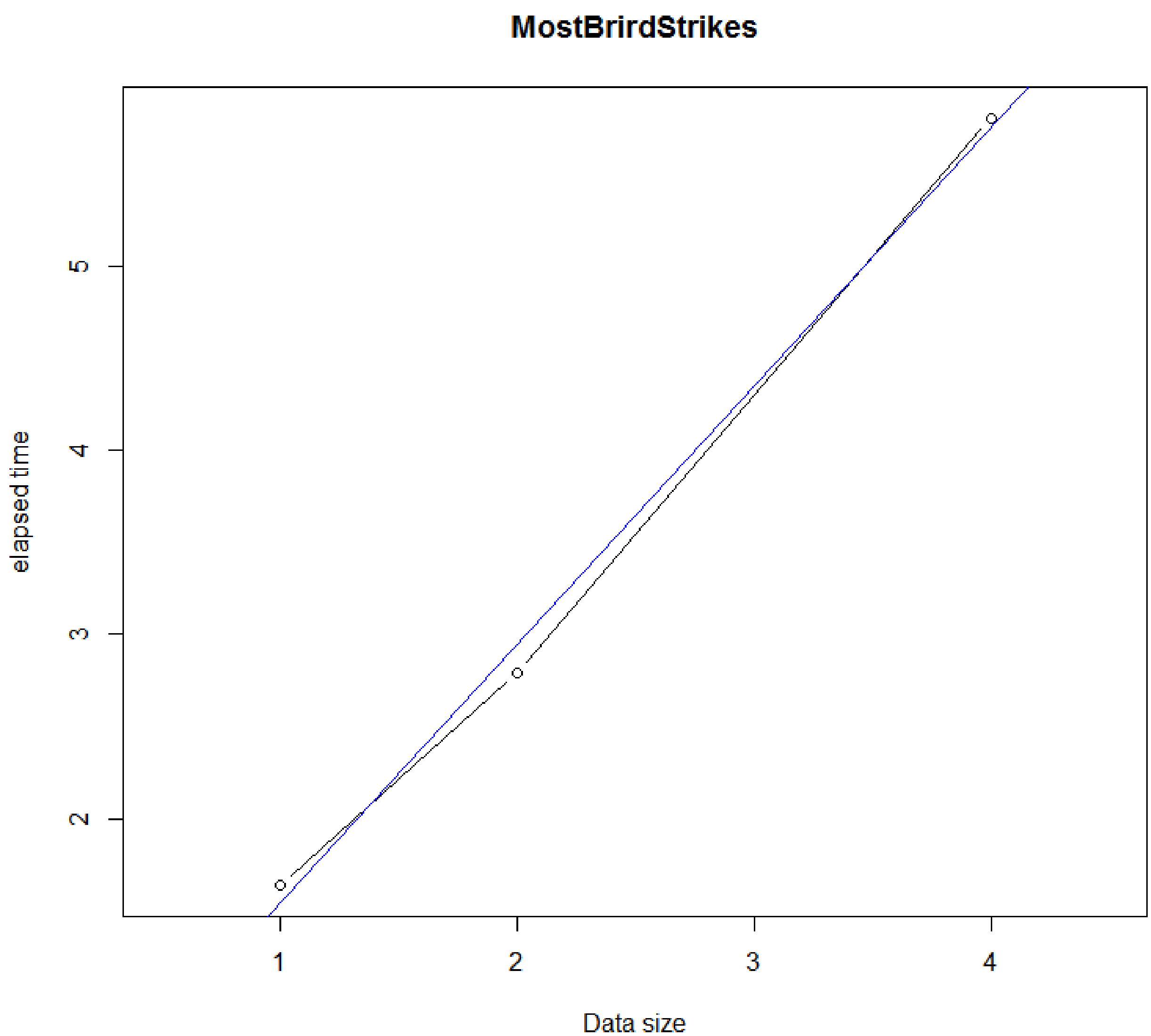
13. *plot and check the effect of different input sizes on time complexity for function AirlineWithMostStrike and then plot a regression line*

Result:

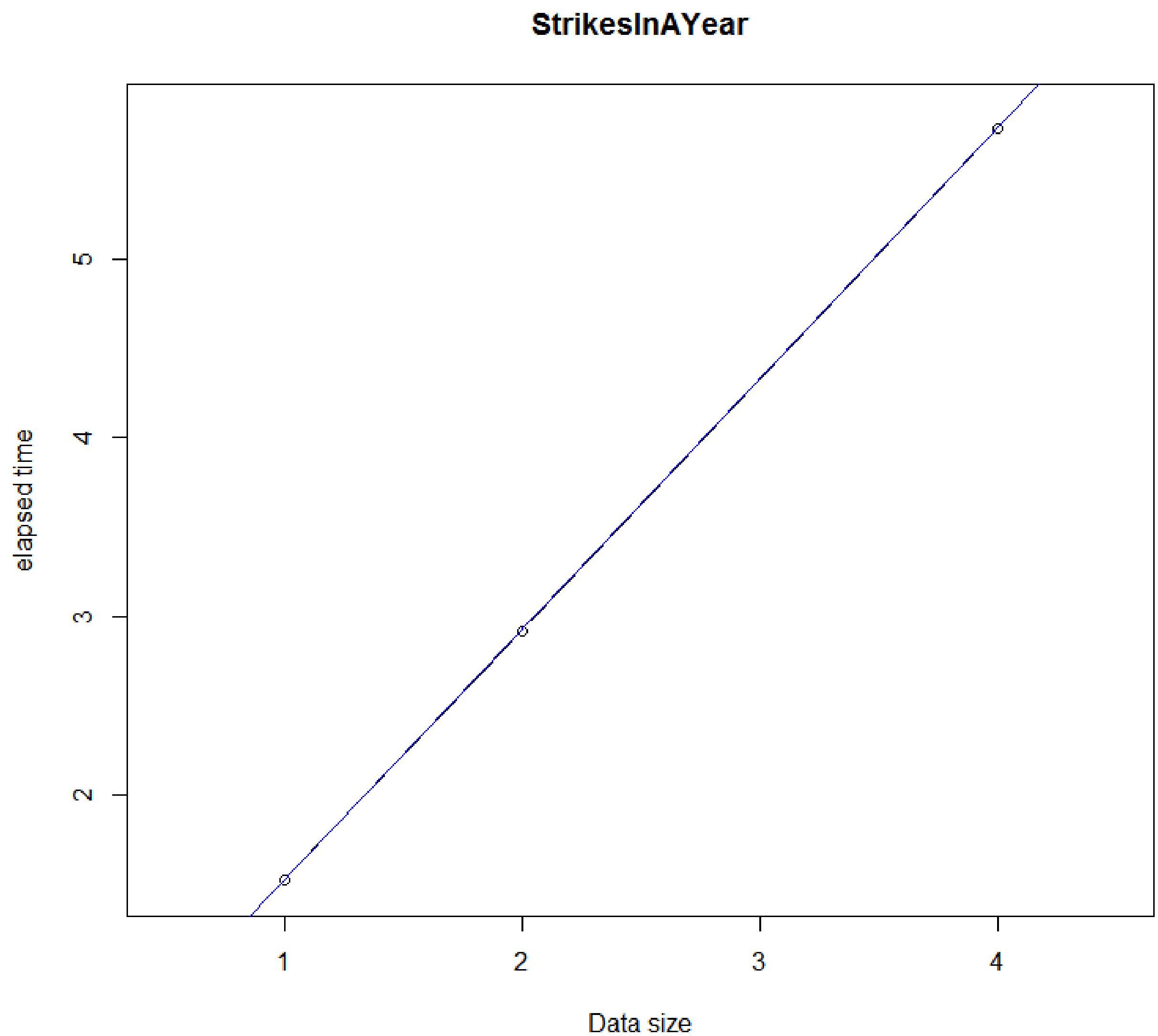
The following graphs were obtained



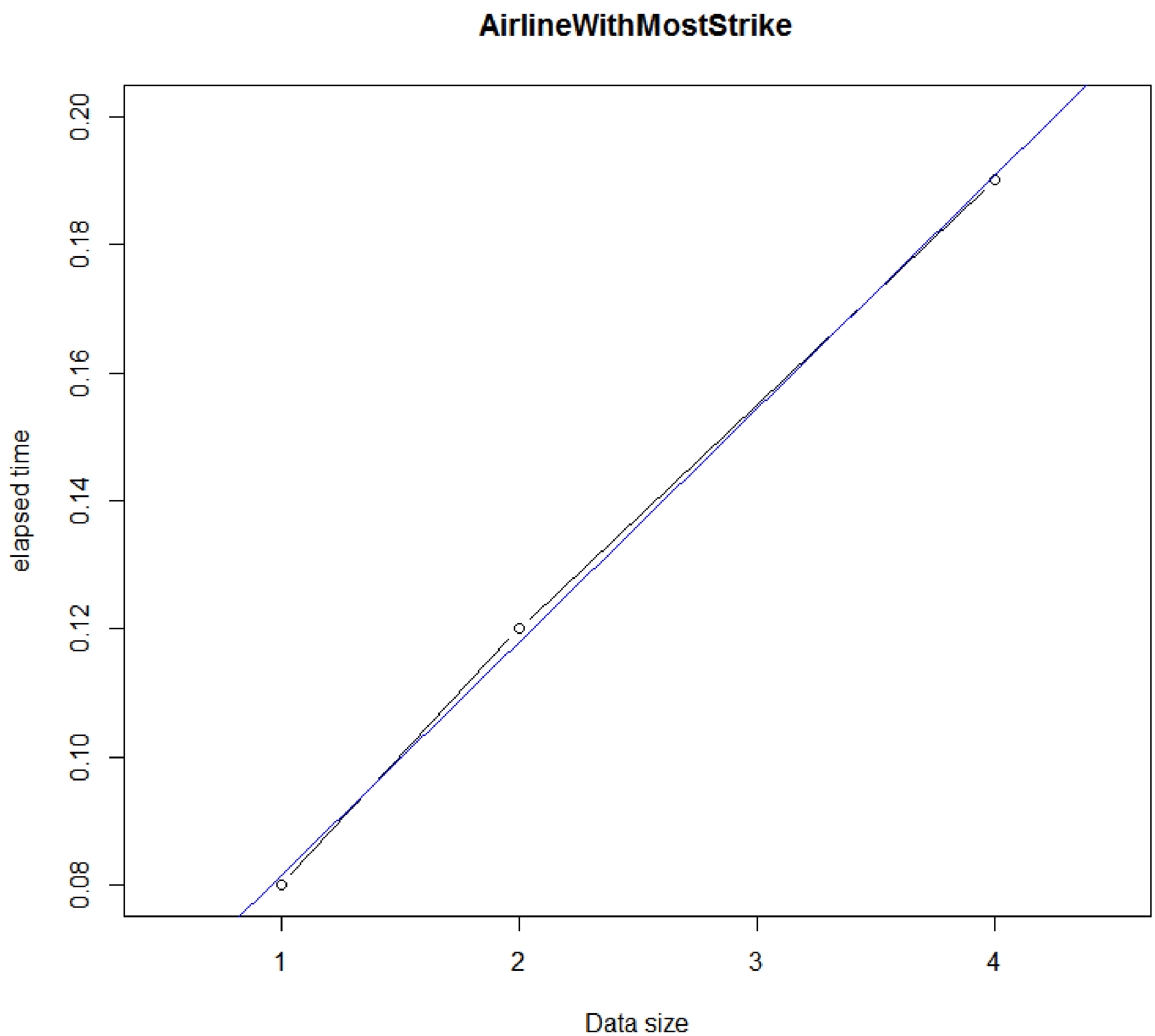
3B Assignment Solutions



3B Assignment Solutions



3B Assignment Solutions



From the graphs above we can summarize that the growth og the function is linearly dependent