18201912_Data Programming with R STAT40620 Semester Exam

Ashtami Bhuleskar

December 17, 2018

Q1

(a)

```
goodread_data<-load("goodreads.Rdata")
nrow(goodreads)
```

```
## [1] 1048575
```

```
dim(goodreads)
```

```
## [1] 1048575       5
```

```
length(unique(goodreads$user_id))
```

```
## [1] 52927
```

There are 1048575 reviews in the data. There are 52927 different users in the dataset.

(b)

```
length(unique(goodreads$authors))
```

```
## [1] 85
```

```
goodreads$title[which.max(table(goodreads$title))]
```

```
## [1] The Hunger Games (The Hunger Games, #1)
## 110 Levels: 1984 ... Wuthering Heights
```

There are 85 different book authors. "The Hunger Games (The Hunger Games, #1)" book was reviewed most often.

(c)

```
avg_rating <- aggregate(goodreads$rating, list(goodreads$title), mean)
colnames(avg_rating) <- c("Book Title", "Average Rating")

any(avg_rating$`Average Rating`==5) #first check if any book has received average rating of 5. After th
```

```
## [1] FALSE
```

```
review1 <- data.frame(table(avg_rating$`Average Rating`>4)["TRUE"]) #Avg rating of at least 4

length(review1$Var1[which(review1$Freq>=10000)]) #giving list of books having reviewed at least 10000 t
```

```
## [1] 0
```

0 or None of the books got rating equal to 5. 44 books were given an average rating of at least 4 51 Books have been reviewed at least 10000 times

(d)

```
class(goodreads) <- 'bookratings'


# create the S3 summary method
summary.bookratings <- function(goodreads){
# calculate the summary statistics

  avg_rated_authors <- aggregate(goodreads$rating, list(goodreads$authors), mean)
  colnames(avg_rated_authors) <- c("Auhtor", "Average Rating")

  s <- avg_rated_authors[order(-avg_rated_authors$`Average Rating`),][1:10,]


  avg_rated_books <- aggregate(goodreads$rating, list(goodreads$title), mean)
  colnames(avg_rated_books) <- c("Title", "Average Rating")

  l <- avg_rated_books[order(-avg_rated_books$`Average Rating`),][1:10,]

  return(list(s,l))
}
# check the function on the Dublin airport dataset
summary(goodreads)
```

```
## [[1]]
##                        Auhtor Average Rating
## 35               J.K. Rowling       4.391197
## 46           Kathryn Stockett       4.382887
## 31                 Harper Lee       4.329369
## 29         George R.R. Martin       4.323678
## 70           Shel Silverstein       4.323616
## 56               Markus Zusak       4.307004
## 83            William Goldman       4.285644
## 21 Elie Wiesel, Marion Wiesel       4.267553
## 58              Maurice Sendak       4.264177
## 61           Orson Scott Card       4.258221
##
## [[2]]
##                                                          Title
## 34       Harry Potter and the Deathly Hallows (Harry Potter, #7)
## 36     Harry Potter and the Half-Blood Prince (Harry Potter, #6)
## 35        Harry Potter and the Goblet of Fire (Harry Potter, #4)
## 38   Harry Potter and the Prisoner of Azkaban (Harry Potter, #3)
## 82                                                    The Help
## 37  Harry Potter and the Order of the Phoenix (Harry Potter, #5)
## 39      Harry Potter and the Sorcerer's Stone (Harry Potter, #1)
## 3                 A Game of Thrones (A Song of Ice and Fire, #1)
## 78                                                The Giving Tree
## 105                                    To Kill a Mockingbird
##     Average Rating
```

```
## 34        4.525941
## 36        4.443339
## 35        4.430780
## 38        4.418732
## 82        4.382887
## 37        4.358697
## 39        4.351350
## 3         4.339880
## 78        4.338475
## 105       4.329369
```

Top 3 rated authors are:

J.K. Rowling 4.391197 Kathryn Stockett 4.382887 Harper Lee 4.329369 #_____ Top 3 rated books are:

Harry Potter and the Deathly Hallows (Harry Potter, #7) 4.525941

Harry Potter and the Half-Blood Prince (Harry Potter, #6) 4.443339

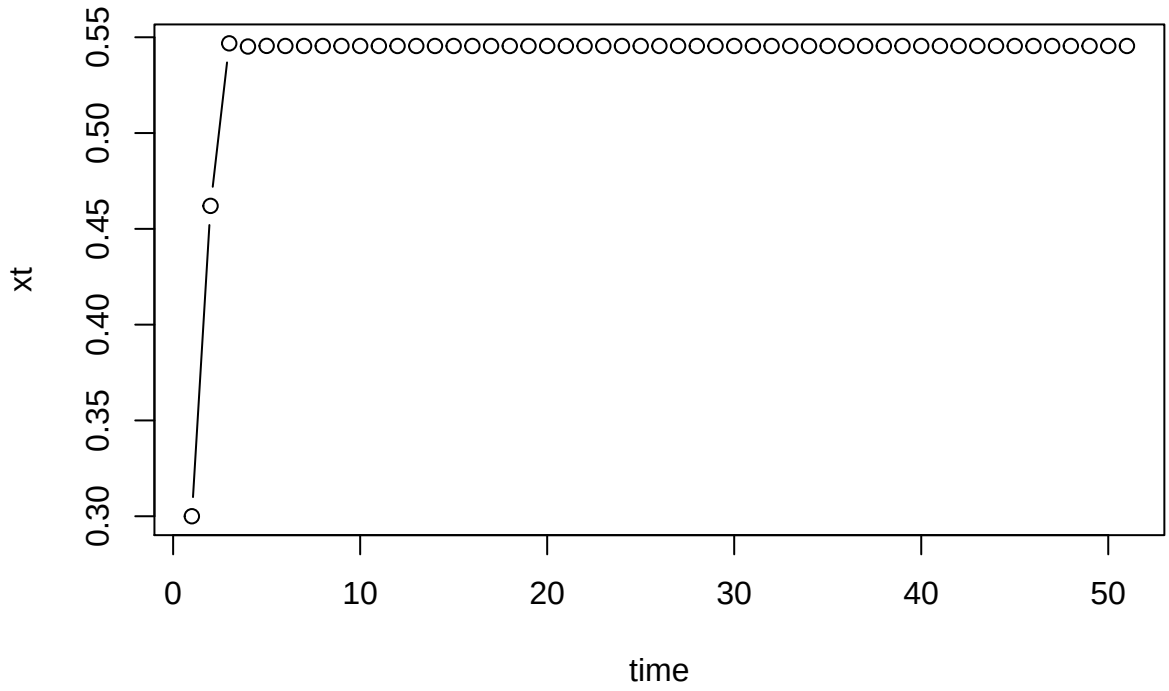
Harry Potter and the Goblet of Fire (Harry Potter, #4) 4.430780

2

(b)

(b) (i)

```r
logistic_map <- function(x0, r, niter = 50){  #function logistic_map is defined which takes 3 arguments
  stopifnot(is.numeric(x0) & length(x0) == 1 & x0 >= 0 & x0 <= 1) #this is optional statement which dea
  xt <- numeric(niter + 1) #xt is initialised having storage capacity equals to niter+1
  xt[1] <- x0 #first value of xt is initialised as equal to x0
  for(i in 1:niter) # for loop runs for niter number of times.
    xt[i + 1] <- r * xt[i] * (1 - xt[i]) #computation of equation step
  plot(xt, type = "b", xlab = "time") #plotting values of xt with label of x axis as "time"
  return(xt) #return xt to the function caller.
}
xt <- logistic_map(x0 = 0.3, r = 2.2)
```




```
xt
```

```
##  [1] 0.3000000 0.4620000 0.5468232 0.5451767 0.5455099 0.5454435 0.5454568
##  [8] 0.5454541 0.5454546 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545
## [15] 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545
## [22] 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545
## [29] 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545
## [36] 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545
## [43] 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545 0.5454545
## [50] 0.5454545 0.5454545
```

(b) (ii)

system.time: It returns CPU (and other) times that the function expression has used.It calls the function proc.time, evaluates expression, and then calls proc.time once more, returning the difference between the two proc.time calls.Timings of evaluations of the same expression can vary considerably depending on whether the evaluation triggers a garbage collection. When gcFirst is TRUE a garbage collection (gc) will be performed immediately before the evaluation of expr. This will usually produce more consistent timings.

Rprof: Enable or disable profiling of the execution of R expressions.Profiling R code gives you the chance to identify bottlenecks and pieces of code that needs to be more efficiently implemented.Rprof works by recording at fixed intervals (by default every 20 msecs) which R function is being used, and recording the results in a file. summaryRprof will give you a list with four elements:

by.self: time spent in function alone. by.total: time spent in function and callees. sample.interval: the

sampling interval, by default every 20 msecs. sampling.time: total time of profiling run. Remember that profiling does impose a small performance penalty.

self.time: how many seconds were spent in that function

self.pct: what percent is this of the overall time. (These numbers should total 100%, modulo rounding and truncation.)

total.time: how many seconds were spent in that function or the functions that it called

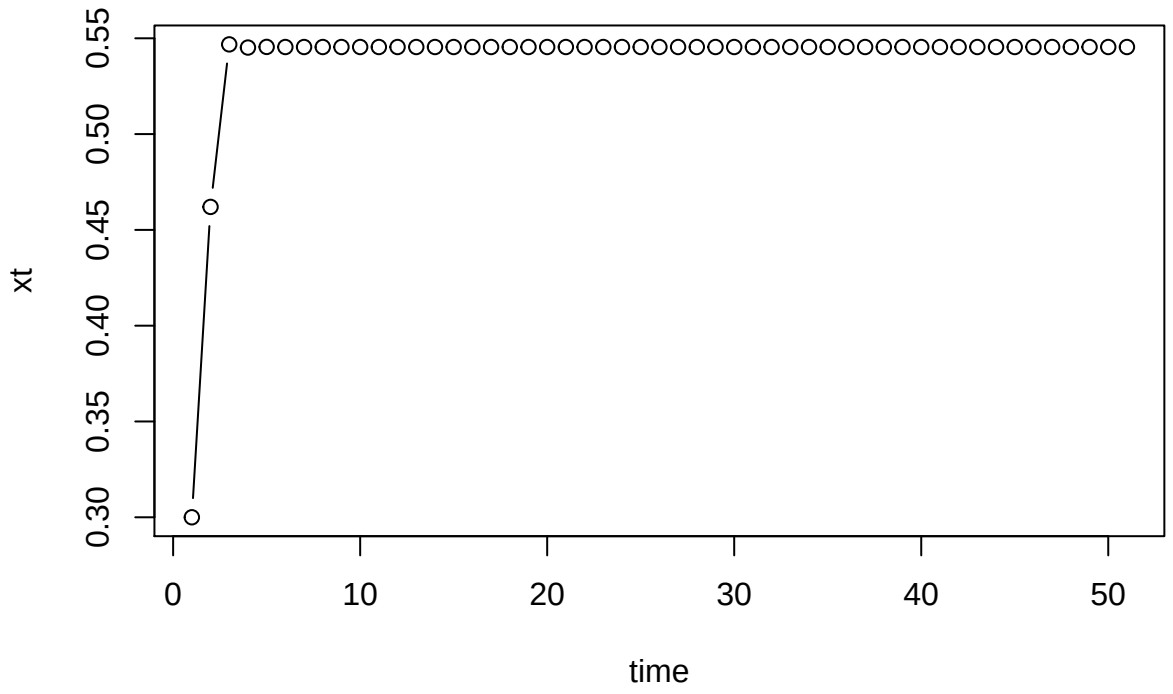
total.pct: what percentage this is of the total. (These numbers will typically total much more than 100%, because functions will be counted for themselves, and for all the functions that call them.)

So look at the functions you wrote that score high in total.pct, because they are the ones where optimization will have an impact. You may be able to modify them to avoid calling some of the high cost functions that you didn't write.

[.data.frame is a known slow function. If execution time matters to you, don't use it. Convert data frames to matrices, where indexing is much faster. In your case, this one change could possibly speed up things by a factor of 2, because you're spending half the execution time just indexing into data frames.

One other piece of advice: after you identify a possible change, make it and redo the profiling. [.data.frame calls so many other functions that if you get rid of it you may drastically change the profile.

```
system.time(for(i in 1:1000)logistic_map(x0 = 0.3, r = 2.2))
```




```
##    user  system elapsed
##    1.61    0.00    1.67
```

```
Rprof()
for(i in 1:1000)logistic_map(x0 = 0.3, r = 2.2)
```

```
Rprof(NULL)
summaryRprof()
```

```
## $by.self
##                   self.time self.pct total.time total.pct
## "plot.default"         0.10    11.11       0.82     91.11
## "axis"                 0.10    11.11       0.10     11.11
## ".External2"           0.06     6.67       0.06      6.67
## "plot.window"          0.06     6.67       0.06      6.67
## "plot"                 0.04     4.44       0.86     95.56
## "tryCatch"             0.04     4.44       0.28     31.11
## "dev.list"             0.04     4.44       0.04      4.44
## "structure"            0.04     4.44       0.04      4.44
## "plot.new"             0.02     2.22       0.32     35.56
## "doTryCatch"           0.02     2.22       0.22     24.44
## "plot_snapshot"        0.02     2.22       0.22     24.44
## "Axis"                 0.02     2.22       0.12     13.33
## "localWindow"          0.02     2.22       0.08      8.89
## "lapply"               0.02     2.22       0.06      6.67
## "as.list"              0.02     2.22       0.04      4.44
## "unlist"               0.02     2.22       0.04      4.44
## "-"                    0.02     2.22       0.02      2.22
## "%in%"                 0.02     2.22       0.02      2.22
## "all"                  0.02     2.22       0.02      2.22
## "as.list.default"      0.02     2.22       0.02      2.22
## "as.vector"            0.02     2.22       0.02      2.22
## "grepl"                0.02     2.22       0.02      2.22
## "identical"            0.02     2.22       0.02      2.22
## "match.fun"            0.02     2.22       0.02      2.22
## "parent.frame"         0.02     2.22       0.02      2.22
## "pmatch"               0.02     2.22       0.02      2.22
## "range"                0.02     2.22       0.02      2.22
## "stopifnot"            0.02     2.22       0.02      2.22
## "strsplit"             0.02     2.22       0.02      2.22
##
## $by.total
##                          total.time total.pct self.time self.pct
## "block_exec"                   0.90    100.00      0.00     0.00
## "call_block"                   0.90    100.00      0.00     0.00
## "evaluate"                     0.90    100.00      0.00     0.00
## "evaluate::evaluate"           0.90    100.00      0.00     0.00
## "evaluate_call"                0.90    100.00      0.00     0.00
## "in_dir"                       0.90    100.00      0.00     0.00
## "knitr::knit"                  0.90    100.00      0.00     0.00
## "process_file"                 0.90    100.00      0.00     0.00
## "process_group"                0.90    100.00      0.00     0.00
## "process_group.block"          0.90    100.00      0.00     0.00
## "rmarkdown::render"            0.90    100.00      0.00     0.00
## "withCallingHandlers"          0.90    100.00      0.00     0.00
## "eval"                         0.88     97.78      0.00     0.00
## "handle"                       0.88     97.78      0.00     0.00
## "logistic_map"                 0.88     97.78      0.00     0.00
## "timing_fn"                    0.88     97.78      0.00     0.00
## "withVisible"                  0.88     97.78      0.00     0.00
```

```
## "plot"                        0.86     95.56      0.04     4.44
## "plot.default"                0.82     91.11      0.10    11.11
## "plot.new"                    0.32     35.56      0.02     2.22
## "tryCatch"                    0.28     31.11      0.04     4.44
## "try"                         0.28     31.11      0.00     0.00
## "doTryCatch"                  0.22     24.44      0.02     2.22
## "plot_snapshot"               0.22     24.44      0.02     2.22
## "handle_output"               0.22     24.44      0.00     0.00
## "tryCatchList"                0.22     24.44      0.00     0.00
## "tryCatchOne"                 0.22     24.44      0.00     0.00
## "w$get_new"                   0.22     24.44      0.00     0.00
## "fun"                         0.20     22.22      0.00     0.00
## "recordPlot"                  0.14     15.56      0.00     0.00
## "Axis"                        0.12     13.33      0.02     2.22
## ".make_numeric_version"       0.12     13.33      0.00     0.00
## "getRversion"                 0.12     13.33      0.00     0.00
## "localAxis"                   0.12     13.33      0.00     0.00
## "package_version"             0.12     13.33      0.00     0.00
## "par"                         0.12     13.33      0.00     0.00
## "plot.xy"                     0.12     13.33      0.00     0.00
## "R_system_version"            0.12     13.33      0.00     0.00
## "axis"                        0.10     11.11      0.10    11.11
## "Axis.default"                0.10     11.11      0.00     0.00
## "localWindow"                 0.08      8.89      0.02     2.22
## ".External2"                  0.06      6.67      0.06     6.67
## "plot.window"                 0.06      6.67      0.06     6.67
## "lapply"                      0.06      6.67      0.02     2.22
## "paste"                       0.06      6.67      0.00     0.00
## "dev.list"                    0.04      4.44      0.04     4.44
## "structure"                   0.04      4.44      0.04     4.44
## "as.list"                     0.04      4.44      0.02     2.22
## "unlist"                      0.04      4.44      0.02     2.22
## "["                           0.04      4.44      0.00     0.00
## "[.simple.list"               0.04      4.44      0.00     0.00
## "deparse"                     0.04      4.44      0.00     0.00
## "-"                           0.02      2.22      0.02     2.22
## "%in%"                        0.02      2.22      0.02     2.22
## "all"                         0.02      2.22      0.02     2.22
## "as.list.default"             0.02      2.22      0.02     2.22
## "as.vector"                   0.02      2.22      0.02     2.22
## "grepl"                       0.02      2.22      0.02     2.22
## "identical"                   0.02      2.22      0.02     2.22
## "match.fun"                   0.02      2.22      0.02     2.22
## "parent.frame"                0.02      2.22      0.02     2.22
## "pmatch"                      0.02      2.22      0.02     2.22
## "range"                       0.02      2.22      0.02     2.22
## "stopifnot"                   0.02      2.22      0.02     2.22
## "strsplit"                    0.02      2.22      0.02     2.22
## "as.character"                0.02      2.22      0.00     0.00
## "as.character.default"        0.02      2.22      0.00     0.00
## "box"                         0.02      2.22      0.00     0.00
## "localBox"                    0.02      2.22      0.00     0.00
##
## $sample.interval
```

```
## [1] 0.02
##
## $sampling.time
## [1] 0.9
```

Interpretation of the Output: By using System.time: The elapsed time required is 20.12

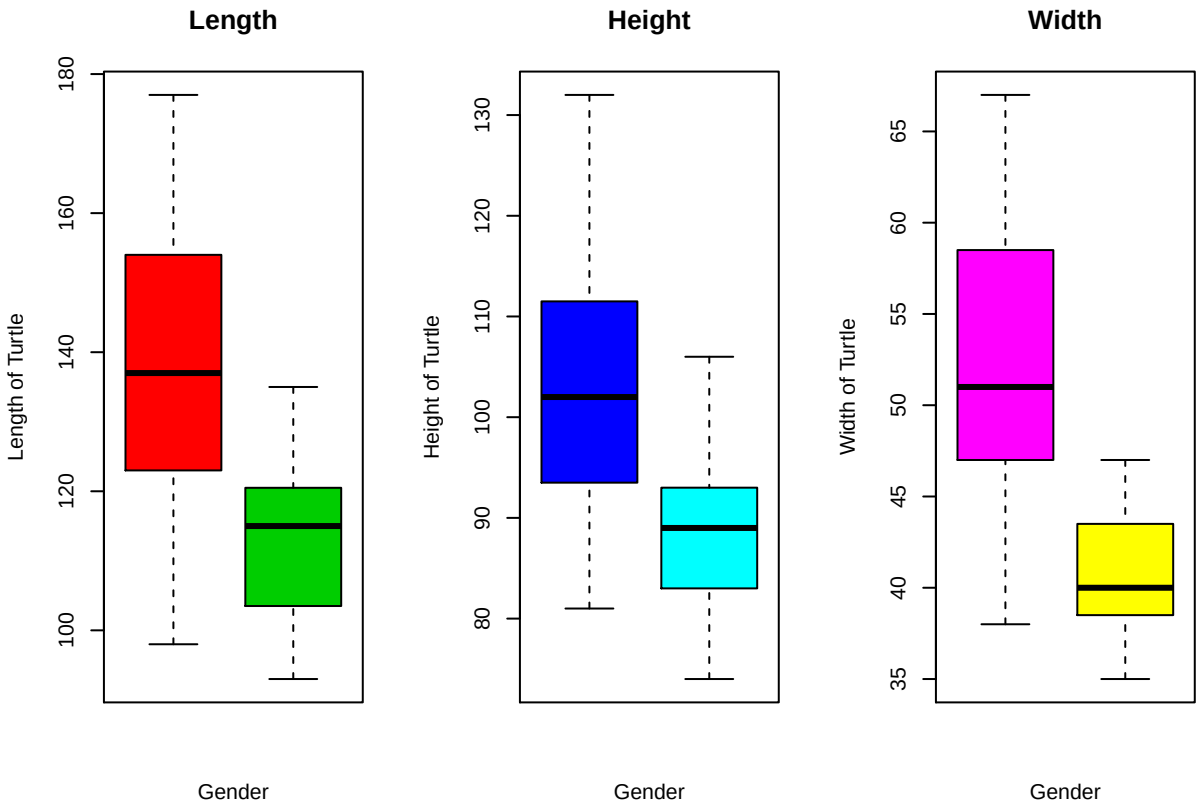
By using Rprof: sampling time is 14.04

Q2 (a)

(i)

```
turtle_data <- read.csv(file = "turtle.csv", head=TRUE,sep=",")

par(mfrow = c(1, 3))
boxplot(turtle_data$length ~ turtle_data$gender, xaxt = "n", main = "Length", col = 2:3, ylab = 'Length
boxplot(turtle_data$height ~ turtle_data$gender, xaxt = "n", main = "Height", col = 4:5, ylab = 'Height
boxplot(turtle_data$width ~ turtle_data$gender, xaxt = "n", main = "Width", col = 6:7, ylab = 'Width of
```



In all three plots, the values of length, Height and Width are significantly high for Female turtles than Male turtles. It would be even rightful to say that the lowest value in all 3 plots is greater than the respective largest value. The Length and Height plot of Female turtle is fairly normally distributed with median at the centre.For the males, length and height shows left skewness whereas width shows right skewness.