Statistical Methods 201-DDD-05 Final Project Report

Part 3: Simulations in R and Further Explorations

Computationally Simulating the Most Recent Common Ancestor (MRCA) Model and Finding the Estimated Time to MRCA (tMRCA) using R Programming Language

Nguyen Hoang Anh

John Abbott College

Montréal, CANADA.

1. Code

Below is the code for the simulation, it is abridged in order to allow for a concise and more streamlined display (some comments and variable declarations have been omitted). The code is divided up into 2 R script files, *index.r* contains the code for model plus the timestamp outputting portion and *graph.r* contains the code for visualizing the data obtained from running the model. Project files can be accessed via GitHub repository https://github.com/zasshuwu/r-project-2019/.

Alternatively, the scripts can be found and executed on my Kaggle notebook https://www.kaggle.com/nguyenhoanganh/mrca-code, which is a good way to review how it works without ramping your device too much.

Codeblock 1. Content of index.r

```r
# TMRCA Estimation Program using Probability 1/n

# parents choosing function
ChooseParents <- function(n, list_p){
  for (i in 1:n) {
    list_p[[i]] <- sample(1:n, size=2, replace=TRUE)
  }
  return(list_p)
}

# descendants list update function
List_d_update <- function(n, list_d, list_p, list_d_temp) {
  for (i in 1:n) {
    v_c_aux <- IsChildren(i, n, list_p)

    list_d_temp[[i]] <- vector()

    for (j in v_c_aux){
      list_d_temp[[i]] <- union(list_d_temp[[i]], list_d[[j]])
    }
  }
  return(list_d_temp)
}

# checking children function
```

```r
IsChildren <- function(i, n, list_p){
  vector_c <- vector()

  for (j in 1:n) {
    if (is.element(i, list_p[[j]])){
      vector_c <- append(vector_c, j)
    }
  }
  return(vector_c)
}

# checking for max local descendants
check_n_local <- function(list_d, n, n_local) {
  for (i in 1:n) {
    if (n_local < length(list_d[[i]])) {
      n_local <- length(list_d[[i]])
    }
  }
  return(n_local)
}

# MASTER FUNCION
index <- function(n) {

  # INITIALIZATION STEP AT THE BEGINNING
  # ------------------------------------

  # init descendants list and parents list
  list_d <- list()
  list_p <- list()
  # init temp descendants list
  list_d_temp <- list()
  # init descendants
  for (i in 1:n) {
    list_d[[i]] <- c(i)
  }
  # init tmrca
  tmrca <- 0
  # init largest number of descendants variable
  n_local <- 0
  # ------------------------------------

  # PROCESSING STEPS
  # ------------------------------------
  while (n_local != n) {
    list_p <- ChooseParents(n, list_p)
    list_d <- List_d_update(n, list_d, list_p, list_d_temp)
    n_local <- check_n_local(list_d, n, n_local)
    # print(n_local)
    tmrca <- tmrca + 1
  }

  # OUTPUT STEPS
  # ------------------------------------
  return(tmrca)
}

# Warning: Running the codes below may take up to 16 hours of your free time.
# Only run code when free!!!

repetition <- 25
```

```
vector_100 <- vector()
vector_1000 <- vector()
vector_4000 <- vector()
vector_5000 <- vector()
vector_10000 <- vector()


# n = 100
for (i in 1:repetition){
  tmrca <- index(100)
  print(i)
  vector_100[i] <- c(tmrca)
}

# n = 1000
for (i in 1:repetition){
  tmrca <- index(1000)
  print(i)
  vector_1000[i] <- c(tmrca)
}

# n = 4000
for (i in 1:repetition){
  tmrca <- index(4000)
  print(i)
  vector_4000[i] <- c(tmrca)
}

# n = 5000
for (i in 1:repetition){
  tmrca <- index(5000)
  print(i)
  vector_5000[i] <- c(tmrca)
}

# n = 10000
for (i in 1:repetition){
  tmrca <- index(10000)
  print(i)
  vector_10000[i] <- c(tmrca)
}

# (RUN THE MODEL FIRST BEFORE) Saving the simulation results into a .csv file

for (i in 1:25) {
  cat("WARNING: DO NOT RUN FAUX DATA GENERATION IF ALREADY RUN SIMULATION CODE!!!\n")
}

# Faux data generation to test out dataframe export
vector_100 <- c(1:25)
vector_1000 <- c(1:25)
vector_4000 <- c(1:25)
vector_5000 <- c(1:25)
vector_10000 <- c(1:25)

# Export
df <- data.frame(vector_100, vector_1000, vector_4000, vector_5000, vector_10000)
write.csv(df, "results1.csv")
```

Codeblock 2. Content of graph.r

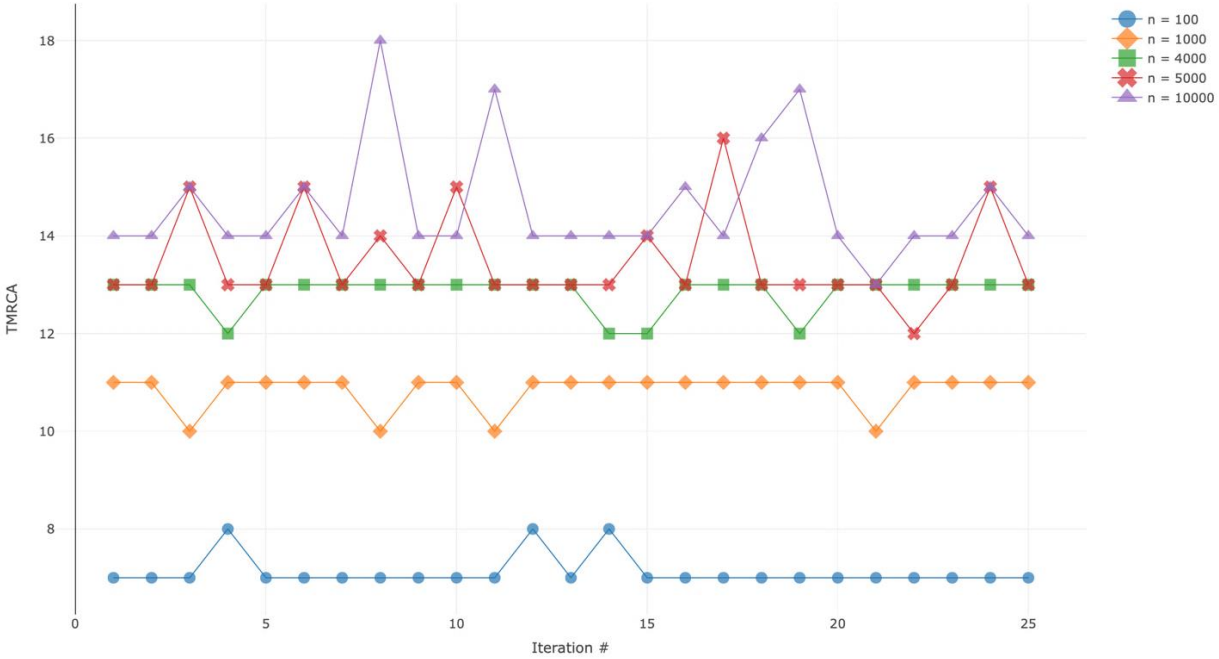
```
# GRAPHING SCRIPTS
# Dependencies: results/results.csv on https://github.com/zasshuwu/r-project-2019

# install.packages(“plotly”)

require(plotly)

df <- read.csv("results.csv")

x <- list(title = "Iteration #")
y <- list(title = "TMRCA")

p <- plot_ly(data = df, x = ~c(1:25)) %>%
    add_trace(y = ~vector_100, name = "n = 100", marker = list(symbol = "circle"), mode="lines
+markers", size = 5, line = list(width=1)) %>%
    add_trace(y = ~vector_1000, name = "n = 1000", marker = list(symbol = "diamond"), mode="li
nes+markers", size = 5, line = list(width=1)) %>%
    add_trace(y = ~vector_4000, name = "n = 4000", marker = list(symbol = "square-
dot"), mode="lines+markers", size = 5, line = list(width=1)) %>%
    add_trace(y = ~vector_5000, name = "n = 5000", marker = list(symbol = "x"), mode="lines+ma
rkers", size = 5, line = list(width=1)) %>%
    add_trace(y = ~vector_10000, name = "n = 10000", marker = list(symbol = "triangle-
up"), mode="lines+markers", size = 5, line = list(width=1)) %>%
    layout(xaxis = x, yaxis = y, title="Estimation of Time to Most Recent Common Ancestor Depe
nding on Population Size")

mean_100 <- mean(df$vector_100)
mean_1000 <- mean(df$vector_1000)
mean_4000 <- mean(df$vector_4000)
mean_5000 <- mean(df$vector_5000)
mean_10000 <- mean(df$vector_10000)

mean_list <- list(mean_100, mean_1000, mean_4000, mean_5000, mean_10000)
p2 <- plot_ly(x = ~c(100, 1000, 4000, 5000, 10000), y = ~mean_list, type="bar") %>%
    layout(title = "Average TMRCA for each n size population", xaxis = list(title="Generations
"), yaxis = list(title="Population size"))
p2
p
```

2. Simulation, Data Processing and Visualization

Using the ideas in the previous part 2, I have written a program with the help of Prof. Takei to simulate the Most Recent Common Ancestor (MRCA) in R programming language. Below is the data collected from running the program.

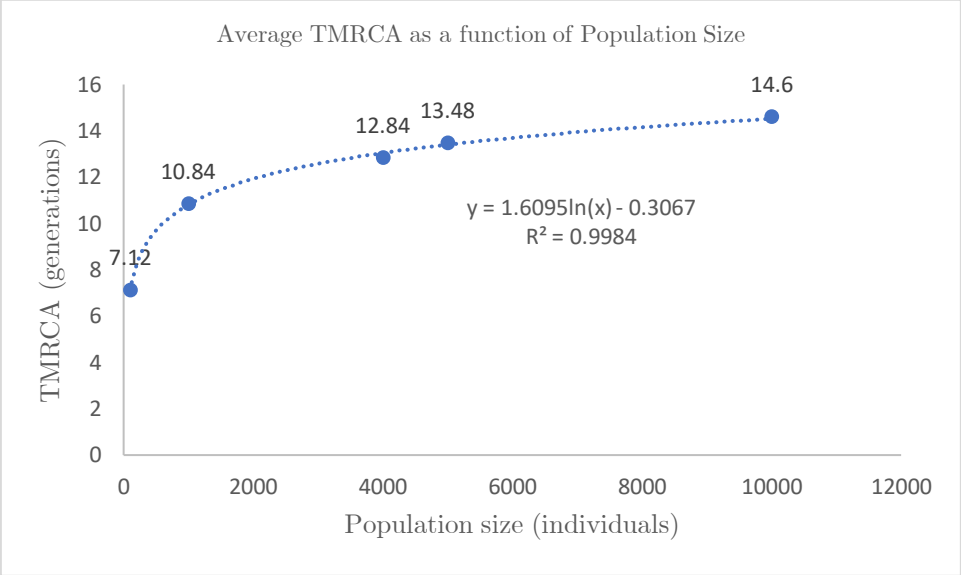
Table 1. Raw data collected from the simulation

Iteration #	vector_100	vector_1000	vector_4000	vector_5000	vector_10000
1	7	11	13	13	14
2	7	11	13	13	14
3	7	10	13	15	15
4	8	11	12	13	14
5	7	11	13	13	14
6	7	11	13	15	15
7	7	11	13	13	14
8	7	10	13	14	18
9	7	11	13	13	14
10	7	11	13	15	14
11	7	10	13	13	17
12	8	11	13	13	14
13	7	11	13	13	14
14	8	11	12	13	14
15	7	11	12	14	14
16	7	11	13	13	15
17	7	11	13	16	14
18	7	11	13	13	16
19	7	11	12	13	17
20	7	11	13	13	14
21	7	10	13	13	13
22	7	11	13	12	14
23	7	11	13	13	14
24	7	11	13	15	15
25	7	11	13	13	14



Graph 1. Visual Representation of the Collected Data from the MRCA Simulation Program. The graph shows the estimated TMRCA (y-axis) of each of the 25 runs (iterations, x-axis). The total elapsed time for all simulation of population size n = {100, 1000, 4000, 5000, 10000} running concurrently was approximately 16 hours.

Graph 2. Average TMRCA and Population Size Correlation (Generated with Excel)



(Bar from left to right) n = 100, n = 1000, n = 4000, n = 5000, n = 10000

Sys.time() raw output to shell

```
Start time:  2019-12-06 15:27:56 EST
End time for n =  100  is:  2019-12-06 15:27:57 EST
----------------------------------------
Start time:  2019-12-06 15:27:57 EST
End time for n =  1000  is:  2019-12-06 15:28:08 EST
----------------------------------------
Start time:  2019-12-06 15:28:08 EST
End time for n =  4000  is:  2019-12-06 15:31:29 EST
----------------------------------------
Start time:  2019-12-06 15:31:29 EST
End time for n =  5000  is:  2019-12-06 15:36:43 EST
----------------------------------------
Start time:  2019-12-06 15:36:43 EST
End time for n =  10000  is:  2019-12-06 16:02:34 EST
----------------------------------------
```

Let Δ_n be the time in seconds it takes for 1 iteration of the model to run for n population size, t_{start} be the start time of the simulation in HH:MM:SS format and t_{end} be the end time of the simulation in HH:MM:SS format, both of respective n population size. Therefore, Δ_n can be calculated by the equation.

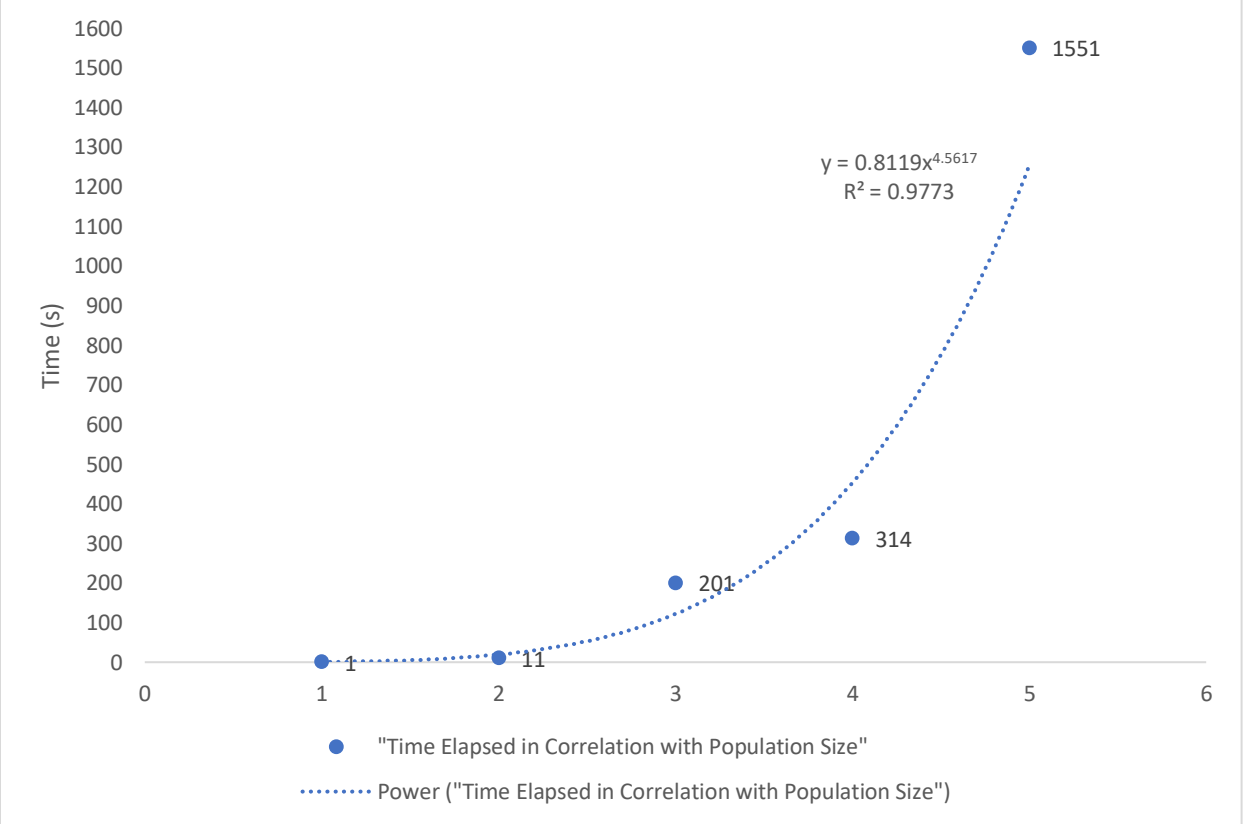
$$\Delta_n = t_{start} - t_{finish}$$

Apply the equation to find the elapsed time for one iteration of the simulation for n population size, we have the following processed data table:

Table 2. Time elapsed for each iteration of each population size n

	Population size n				
	100	1000	4000	5000	10000
Time elapsed $\boldsymbol{\Delta_n}$ (s)	1	11	201	314	1551
Time elapsed for 25 iterations (s)	25	275	5,025	7,850	38,775
25 iterations in hours (h)	0.007	0.076	1.396	2.181	10.771

Graph 3. Bar graph of time elapsed (Generated with Excel)



3. Conclusion

On average, the TMRCA and population size n has a logarithmic correlation as shown in Graph 1. For this observation, it is interesting as it coincides with the MRCA paper by Dr. Chang[1]. And we can see from *Graph 2* that the correlation between the time it takes for an iteration of population size n and the population size n is a power function. Respectively, the fitting of the two functions yield R^2 value of 0.9984 and 0.9773.

As Howard Stark said in Iron Man 2 movie (2010): "I am limited by the technology of my time...", the simulation could have been done better in terms of the number of iterations and the number of population size simulated if I had had a proper knowledge of how to enable multi-threading with R and utilize the calculation power of my GPU. With that in mind, the time to simulate the MRCA would have been shorter; however, it is to be noted that the correlation between simulation runtime and population size is likely to still be a power function.

The limitations of the computational prowess aside, I think that the program serves its purpose of simply running the MRCA simulation and output the TMRCA with a probability of 1/n. The model could be improved in terms of realisticity by factoring other parameters of an individual, such as hereditary diseases that prevent reproduction and/or partner preferences (this was eliminated by using the couple model). The pros of using the probabilistic/statistical methods to find the TMRCA are that it is not difficult to get a program up and running, and we get to eliminate the social complexities that exist in real life in order to get a rough estimation. Also, with a simple model, the computational power required to run the model is achievable for a non-professional individual.

From this part of the project, I learned that even though the concept of the model is easy to understand and fairly simple, the programming part was rather challenging as I am not that familiar with R data types and data structures, as well as many functions of R. However, in the aftermath, R is a convenient language for this very use case because of those built-in functionalities and data structures.

Acknowledgement

Great thanks to Professor Takei for providing me with a guiding frame for the programming part and valuable guidances along the way.

[1] CHANG, J. T. (1999). Recent common ancestors of all present-day individuals. Advances in Applied Probability, 31(4), 1002-1026. doi:10.1239/aap/1029955256.