Unit 2 Case Study

MSDS Fall ‘19

7333 Quantify the World

*[NOTE: the team solved problems 7, 10, and 11]*

Daniel Serna, Bruce Granger, Brandon de la Houssaye

September 24, 2019

**Abstract**

In this paper we provide a solution to extract, structure, and organize data as it relates to the annual “Cherry Blossom” road race held in Washington, D.C. The road race has a website ([www.cherryblossom.org](http://www.cherryblossom.org)) that provides the race results by year back to 1999. These race results provide, by runner name, the times, positioning, age, hometown, etc. Additionally, race results are separated by sex (i.e., male or female). However, each year (i.e., 1999, 2000, 2001, etc.) the race results are called to a separate html page on the website, and each html page may have a different structure. As a result, this paper provides, for each year between 1999 and 2012, the method to extract the race results for males and females into singular, structured matrix environments allowing for further analysis. In particular, this paper also provides – as a means for demonstrating the effectiveness of the structured data – initial exploratory data analytics on the run race results.

**1 Introduction**

The annual Cherry Blossom road race is considered of material interest in the competitive and hobbyist world of road running. The race is considered a precursor to later races, such as the Boston marathon. The race organizers had the foresight to maintain records of historical race results and post these records on the website: [www.cherryblossom.org](http://www.cherryblossom.org). These records include a number of interesting data attributes by runner, including overall finishing position, age, hometown, and time(s). Given the high participation rates of the road race; historical results, and variance of participants in terms of age, times, and hometown, the data is of interest.

However, the data, as it is made available on the website, is not provided in an easy format for additional analysis. Specifically, the historical race results are separated by males and females as well as by year into separate, linked HTM formats. The data, by sex and year, are also not in similar formats in terms of information provided (i.e., “time”, “gun time”, etc. attributes) or string structure (i.e., header rows). Additionally, the call to the various HTM sites (containing the results) are inconsistent.

As a result, this paper provides the code necessary to do the following:

1. Extract the data for each year and sex from 1999 through 2012 into a singular code environment.
2. Structure the data into a consistent format.
3. Extract the data into a usable format (i.e., from a data string with vectors into a matrix).
4. Cleanse the extracted data to account for missing information and irrational attributes (e.g., an illogical age or time).

Thus, the work contained in this paper provide a reusable tool to any researcher that wishes to further analyze the road race results between 1999 and 2012 for both males and females. For final measure, this paper provides initial exploratory data analytics for the male results in order to confirm that the information was extracted and structured into a usable format, as well as to better understand the relationships between the results by year. What follows is an overview of the data utilized and the methods undertaken.

**2 Data**

Data was provided by [www.cherryblossom.org](http://www.cherryblossom.org). Race results could be gathered by going to each site, by year and sex. The team first gathered the list of sites for each of the 14 years and sex (28 sites in total). The list of the 28 sites is provided for in the attachment to this paper. The data format for each of the years by sex is HTM (i.e., string structure).

As mentioned previously, the information provided by year may vary in terms of attributes assigned to each runner typically by time scores. A list of the typical data attributes provides is provided below.

* Place = order of finish by runner
* Div / total = there are different “divisions” for the race set by the age brackets. Where this was provided (division results) it shows the finishing by runner within their respective division.
* Num = the uniquely assigned number by runner for that given year
* Name = the registered name of the runner
* Ag = the age of the runner
* Hometown = the hometown (or country) of the runner
* 5 mi = the time score of the runner at ‘mile 5’.
* Pace = the minute pace of the runner over the defined course of the race. In some cases, depending upon ordering of the attribute, the pace score refers to overall race, 5 mi, or 10 km.
* 10 km = the time score of the runner at ‘kilometer 10’.
* Time (or Net or Gun) = the final time score of the overall race

The variables of interest (i.e., the data that will be pulled into a usable structure) include year, age, name, hometown, and run time. For runtime, if “net” is available this is used, otherwise we default to “gun” and then “time”.

**3 Data Cleansing and Structuring**

The first step to conduct to define the data source. This was done by providing the list of HTM calls and creating into defined objects.

The next step was to extract the data (from the HTM call objects). Tailoring of the code was required to account for the HTM structure by year and sex. Specific functions were built in order to extract the data (in particular the htmlparse() function as part of the XML library in R). A separate extract data function was built for males versus females due to the difference in year 2009 race results. Another example is that for year 2001, the female results did not contain header rows or data element titles so this information needed to be input.

After the data was extracted and, in some cases, manually fixed, the data needed to be transformed into a dataframe structure. This was the final goal – to create a usable dataframe structure for future analysis.

The conversion from extracted data (which was still in string format) to a dataframe was done by building a number of functions that included extracting the variables (data elements); removing repetitive header rows; converting time scores into a singular format; and singular data elements into individual columns (i.e., so Num and Name was not in the same dataframe column). All of these mentioned functions were built in a dataframe conversion function (so they would be called when that dataframe conversion function was called). The dataframe conversion function also ensured that various variables would be transformed into an appropriate class (e.g., integer or character). The various functions built and utilized to perform this conversion are provided in the attached code.

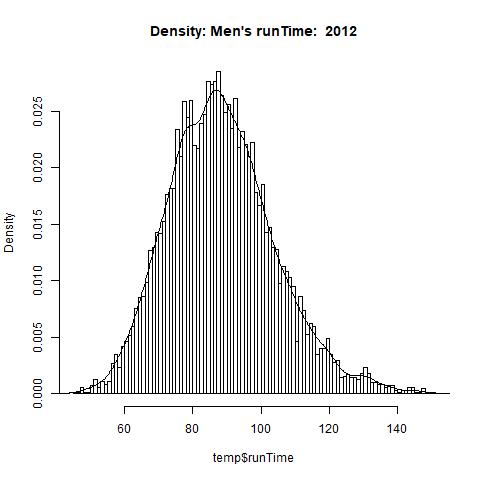
The end result was a dataframe that included the following variables by runner: year, age, name, hometown, and run time. A separate dataframe was created for males and females. After the dataframes were created, “NA” age results were also eliminated (removing the entire runner records where age was not available). This last step, removing the “NA” results by age was done because age was determined to be a key variable of interest for most analyses that would be accomplished (i.e., without this variable, the user would likely not be interested in considering the runner results).

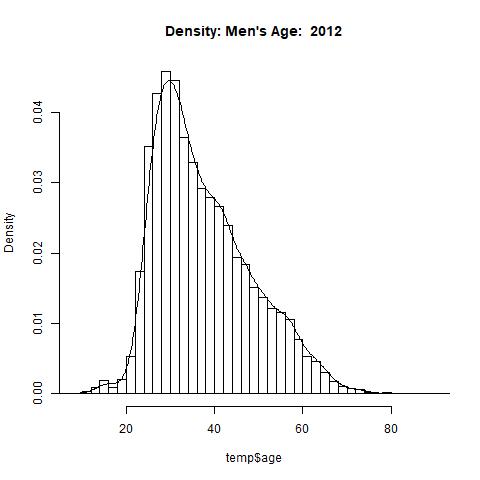
These dataframes were finally written into ‘.csv’ files for user consumption.

**4 EDA**

Once the data was extracted, cleansed and structured, the team created basic dispersion plots of the male race results in order to observe the relationship between certain variables over the years. In particular, QQ plots, boxplots, and density plots were created by year (and overall) for runtime and age for the males. In a simple sense, considering things by year, this means that there are (for three different types of plots and two different variables) 84 plots. However, for boxplots, the years for each of the variables were combined into singular views (which is most helpful for year over year comparison).

All of the plots effectively represent the same concept: the distribution of [variable] over the population in real terms. That is, for all the runners, do the times and ages skew towards the minimum or maximum results? Are they evenly distributed around the median? If we compare the distribution results between the two variables, do we see variance? Consider the density plots for 2012:

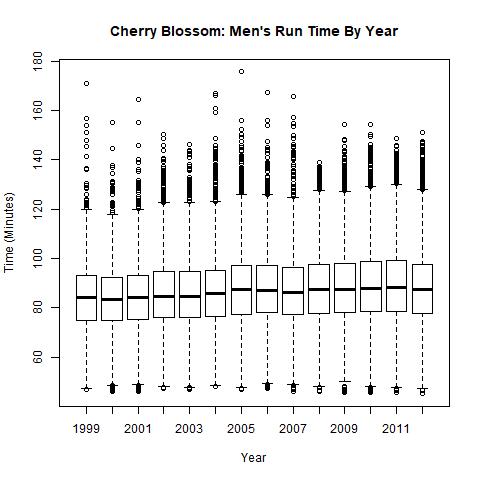


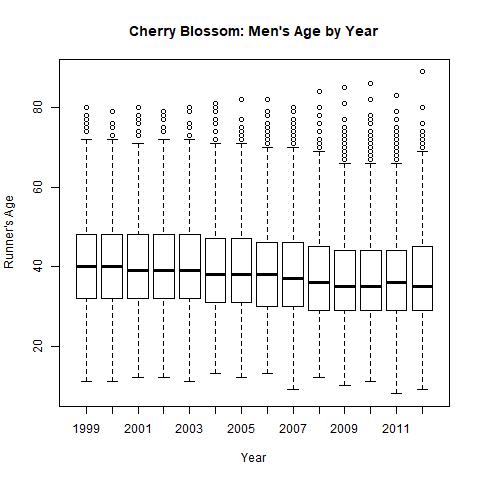


What could we conclude from these two plots? In general, the overall group of runners (male) tended to skew towards the youngest age (i.e., most runners were ‘young’, but there were a handful of very old runners). In other words, people have a tendency to stop running in road races as they get older except for a very small group of fitness fanatics!

When we compare the runner age distribution results against the runtime results, we observe a more normalized distribution around the median scores. However, the tail to the right seems to be longer. This simply implies that the great majority of ‘younger’ runners or overall participants had a more normalized tendency of results. i.e., the skews were not as significant. The time scores that seem to skew towards the longer end (to the right could be attributable to the ‘older’ runners or even less in shape runners closer towards the age median).

Let’s consider the bloxplot results which are provided across all years.





The boxplots show similar information as the density plots and QQ plots, but, again, what is helpful here is the comparison across all years. So, what does this data show?

First, what this data does not show is statistical significance. That is, we cannot conclude the any year over year variance of population distribution is meaningful (or could be used to conclude on a hypothesis). However, using visual depiction, we can observe an interesting trend. Namely, starting around 2007, middle 50% of participants ages are getting younger while the times are getting higher. It is slight, but it is there. What conclusions can be drawn from this? Simply, the overall ages of participants decreased, but their runtimes increased. The question of “why” will go answered from this data view, but maybe video games are at fault.

**3 Results**

The results of this paper provide (re) usable code to establish Cherry Blossom race results for each year between 1999 and 2012 into structured dataframes for additional analysis. This was done for both men and women. Additionally, the results of this paper show the structured dataframe data in action by considering basic dispersion plots of the race results and considering what those dispersion plots represent.

**4 Conclusion**

This paper can be used to reproduce the Cherry Blossom results in particular (or even just visit the github repository for the created .csv files). However, more importantly, the code and methods contained in this paper can be leveraged to extract, cleanse and structure data from any number of sources available on the internet where the initial information structure is not conducive to further analysis.

**A Code**

Code was done in the “R” language. The workbook utilized (coding environment) was “RMarkdown”. Final files are provided in the following github repository: <https://github.com/G-Bruce/7333-Case-Study-Two>. The repository also contains the data analytics graphical output.

What follows is a summary of the code utilized.

|  |
| --- |
| title: "Quantifying\_casestudy4" |
|  | author: " De La Houssaye, Brandon; Granger, Bruce Serna, Daniel; Brandon" |
|  | date: "September 14, 2019" |
|  | output: html\_document |
|  | --- |
|  |  |
|  | ```{r setup, include=FALSE} |
|  | knitr::opts\_chunk$set(echo = TRUE) |
|  | ``` |
|  |  |
|  | ```{r setup} |
|  | #URL for MEN and Library |
|  |  |
|  | library(XML) |
|  |  |
|  | library(tidyverse) |
|  |  |
|  |  |
|  | ubase = "http://www.cherryblossom.org/" |
|  |  |
|  |  |
|  |  |
|  | #### Revised URLS |
|  | menURLsV2 = |
|  | c("results/1999/cb99m.html", #"cb99m.htm" |
|  | "results/2000/Cb003m.htm", #"cb003m.htm" |
|  | "results/2001/oof\_m.html", #"results/2001/oof\_m.html" |
|  | "results/2002/oofm.htm", #"results/2002/oofm.htm" |
|  | "results/2003/CB03-M.HTM", #"results/2003/CB03-M.HTM" |
|  | "results/2004/men.htm", #"results/2004/men.htm" |
|  | "results/2005/CB05-M.htm", #"results/2005/CB05-M.htm" |
|  | "results/2006/men.htm", #"results/2006/men.htm" |
|  | "results/2007/men.htm", #"results/2007/men.htm" |
|  | "results/2008/men.htm", #"results/2008/men.htm" |
|  | "results/2009/09cucb-M.htm", #"results/2009/09cucb-M.htm" |
|  | "results/2010/2010cucb10m-m.htm", #"results/2010/2010cucb10m-m.htm" |
|  | "results/2011/2011cucb10m-m.htm", #"results/2011/2011cucb10m-m.htm" |
|  | "results/2012/2012cucb10m-m.htm" #"results/2012/2012cucb10m-m.htm" |
|  | ) |
|  | #### |
|  |  |
|  |  |
|  |  |
|  | #### Revised URLS |
|  | urlsV2 = paste(ubase, menURLsV2, sep="") |
|  | urlsV2[1:4] |
|  |  |
|  | ``` |
|  |  |
|  | ```{r utilityFunctions} |
|  |  |
|  | #FUNCTIONS |
|  |  |
|  | #extractResTablev2 |
|  |  |
|  | extractResTableV2 = |
|  | # |
|  | # Retrieve data from web site, |
|  | # find the preformatted text, |
|  | # and write lines or return as a character vector. |
|  | # |
|  | function(url = "http://www.cherryblossom.org/results/2009/09cucb-F.htm", |
|  | year = 1999, sex = "male", file = NULL) |
|  | { |
|  | #added encoding for windows users who get an "A" symbol |
|  | doc = htmlParse(url, encoding="UTF-8") |
|  |  |
|  | if (year == 2000) { |
|  | # Get preformatted text from 4th font element |
|  | # The top file is ill formed so the <pre> search doesn't work. |
|  | ff = getNodeSet(doc, "//font") |
|  | txt = xmlValue(ff[[4]]) |
|  | els = strsplit(txt, "\r\n")[[1]] |
|  | } |
|  | else if (year == 2009 & sex == "male") { |
|  | # Get preformatted text from <div class="Section1"> element |
|  | # Each line of results is in a <pre> element |
|  | div1 = getNodeSet(doc, "//div[@class='Section1']") |
|  | pres = getNodeSet(div1[[1]], "//pre") |
|  | els = sapply(pres, xmlValue) |
|  | } |
|  | else if (year == 1999 & sex == "male") { # have to add this else if statement |
|  | # Get preformatted text from <pre> elements |
|  | pres = getNodeSet(doc, "//pre") |
|  | txt = xmlValue(pres[[1]]) |
|  | els = strsplit(txt, "\n")[[1]] |
|  | } |
|  | else { |
|  | # Get preformatted text from <pre> elements |
|  | pres = getNodeSet(doc, "//pre") |
|  | txt = xmlValue(pres[[1]]) |
|  | els = strsplit(txt, "\r\n")[[1]] |
|  | } |
|  |  |
|  | if (is.null(file)) return(els) |
|  | # Write the lines as a text file. |
|  | writeLines(els, con = file) |
|  | } |
|  |  |
|  | extractResTableV3 = |
|  | # |
|  | # Retrieve data from web site, |
|  | # find the preformatted text, |
|  | # and write lines or return as a character vector. |
|  | # |
|  | function(url = "http://www.cherryblossom.org/results/2009/09cucb-F.htm", |
|  | year = 1999, sex = "female", file = NULL) |
|  | { |
|  | #added encoding for windows users who get an "A" symbol |
|  | doc = htmlParse(url, encoding="UTF-8") |
|  |  |
|  | if (year == 2000) { |
|  | # Get preformatted text from 4th font element |
|  | # The top file is ill formed so the <pre> search doesn't work. |
|  | ff = getNodeSet(doc, "//font") |
|  | txt = xmlValue(ff[[4]]) |
|  | els = strsplit(txt, "\r\n")[[1]] |
|  | } |
|  | #else if (year == 2009 & sex == "female") { |
|  | # Get preformatted text from <div class="Section1"> element |
|  | # Each line of results is in a <pre> element |
|  | #div1 = getNodeSet(doc, "//div[@class='Section1']") |
|  | #pres = getNodeSet(div1[[1]], "//pre") |
|  | #els = sapply(pres, xmlValue) |
|  | #} |
|  | else if (year == 1999 & sex == "female") { # have to add this else if statement |
|  | # Get preformatted text from <pre> elements |
|  | pres = getNodeSet(doc, "//pre") |
|  | txt = xmlValue(pres[[1]]) |
|  | els = strsplit(txt, "\n")[[1]] |
|  | } |
|  | else { |
|  | # Get preformatted text from <pre> elements |
|  | pres = getNodeSet(doc, "//pre") |
|  | txt = xmlValue(pres[[1]]) |
|  | els = strsplit(txt, "\r\n")[[1]] |
|  | } |
|  |  |
|  | if (is.null(file)) return(els) |
|  | # Write the lines as a text file. |
|  | writeLines(els, con = file) |
|  | } |
|  |  |
|  |  |
|  | #findColLocs |
|  |  |
|  | findColLocs = function(spacerRow) { |
|  |  |
|  | spaceLocs = gregexpr(" ", spacerRow)[[1]] |
|  | rowLength = nchar(spacerRow) |
|  |  |
|  | if (substring(spacerRow, rowLength, rowLength) != " ") |
|  | return( c(0, spaceLocs, rowLength + 1)) |
|  | else return(c(0, spaceLocs)) |
|  | } |
|  |  |
|  | selectCols = function(shortColNames, headerRow, searchLocs) { |
|  | sapply(shortColNames, function(shortName, headerRow, searchLocs){ |
|  | startPos = regexpr(shortName, headerRow)[[1]] |
|  | if (startPos == -1) return( c(NA, NA) ) |
|  | index = sum(startPos >= searchLocs) |
|  | c(searchLocs[index] + 1, searchLocs[index + 1]) |
|  | }, headerRow = headerRow, searchLocs = searchLocs ) |
|  | } |
|  |  |
|  |  |
|  | #extractVariables |
|  |  |
|  | extractVariables = |
|  | function(file, varNames =c("name", "home", "ag", "gun", |
|  | "net", "time")) |
|  | { |
|  |  |
|  | # Find the index of the row with =s |
|  | eqIndex = grep("^===", file) |
|  | # Extract the two key rows and the data |
|  | spacerRow = file[eqIndex] |
|  | headerRow = tolower(file[ eqIndex - 1 ]) |
|  | body = file[ -(1 : eqIndex) ] |
|  | # Remove footnotes and blank rows |
|  | footnotes = grep("^[[:blank:]]\*(\\\*|\\#)", body) |
|  | if ( length(footnotes) > 0 ) body = body[ -footnotes ] |
|  | blanks = grep("^[[:blank:]]\*$", body) |
|  | if (length(blanks) > 0 ) body = body[ -blanks ] |
|  |  |
|  |  |
|  | # Obtain the starting and ending positions of variables |
|  | searchLocs = findColLocs(spacerRow) |
|  | locCols = selectCols(varNames, headerRow, searchLocs) |
|  |  |
|  | Values = mapply(substr, list(body), start = locCols[1, ], |
|  | stop = locCols[2, ]) |
|  | colnames(Values) = varNames |
|  |  |
|  | return(Values) |
|  | } |
|  |  |
|  |  |
|  | selectCols = function(shortColNames, headerRow, searchLocs) { |
|  | sapply(shortColNames, function(shortName, headerRow, searchLocs){ |
|  | startPos = regexpr(shortName, headerRow)[[1]] |
|  | if (startPos == -1) return( c(NA, NA) ) |
|  | index = sum(startPos >= searchLocs) |
|  | c(searchLocs[index] + 1, searchLocs[index + 1]) |
|  | }, headerRow = headerRow, searchLocs = searchLocs ) |
|  | } |
|  |  |
|  | convertTime = function(time) { |
|  | timePieces = strsplit(time, ":") |
|  | timePieces = sapply(timePieces, as.numeric) |
|  | sapply(timePieces, function(x) { |
|  | if (length(x) == 2) x[1] + x[2]/60 |
|  | else 60\*x[1] + x[2] + x[3]/60 |
|  | }) |
|  | } |
|  | createDF = function(Res, year, sex) |
|  | { |
|  | # Determine which time to use |
|  | if ( !is.na(Res[1, 'net']) ) useTime = Res[ , 'net'] |
|  | else if ( !is.na(Res[1, 'gun']) ) useTime = Res[ , 'gun'] |
|  | else useTime = Res[ , 'time'] |
|  |  |
|  | # Remove # and \* and blanks from time |
|  | useTime = gsub("[#\\\*[:blank:]]", "", useTime) |
|  | runTime = convertTime(useTime[ useTime != "" ]) |
|  |  |
|  | # Drop rows with no time |
|  | Res = Res[ useTime != "", ] |
|  |  |
|  | Results = data.frame(year = rep(year, nrow(Res)), |
|  | sex = rep(sex, nrow(Res)), |
|  | name = Res[ , 'name'], home = Res[ , 'home'], |
|  | age = as.numeric(Res[, 'ag']), |
|  | runTime = runTime, |
|  | stringsAsFactors = FALSE) |
|  | invisible(Results) |
|  | } |
|  |  |
|  |  |
|  | ``` |
|  |  |
|  |  |
|  | ```{r notNeeded} |
|  |  |
|  | urlsV2[1] |
|  | ``` |
|  |  |
|  |  |
|  | ```{r retrieveMenData} |
|  | years = 1999:2012 |
|  |  |
|  | ###Correct function for 1999 normalization |
|  | #### Corrected function to pull down Male tables with consistent format |
|  | mensTables = lapply (urlsV2, extractResTableV2) |
|  |  |
|  | menTablesV2 = mapply(extractResTableV2, url = urlsV2, year = years) |
|  | names(menTablesV2) = years |
|  | sapply(menTablesV2, length) |
|  |  |
|  | #### Confirmation that the 1999 and other years have consistent formatting |
|  | #####menTablesV2$'1999'[1:10] |
|  | ####menTablesV2[[2]][1:10] |
|  | menTablesV2[[3]][1:10] |
|  | #####menTablesV2[[4]][1:10] |
|  | #####menTablesV2[[5]][1:10] |
|  | #####menTablesV2[[6]][1:10] |
|  | #####menTablesV2[[7]][1:10] |
|  | menTablesV2[[8]][1:10] |
|  | #### Save the outputs |
|  | save(menTablesV2, file = "CBMenTextTables\_Houssaye.rda") |
|  |  |
|  | ``` |
|  |  |
|  |  |
|  | ```{r cleanseMenData} |
|  |  |
|  |  |
|  | separatorIdx = grep("^===", menTablesV2[["2006"]]) |
|  | separatorRow = menTablesV2[['2006']][separatorIdx] |
|  | separatorRowX = paste(substring(separatorRow, 1, 63), " ", |
|  | substring(separatorRow, 65, nchar(separatorRow)), |
|  | sep = "") |
|  | menTablesV2[['2006']][separatorIdx] = separatorRowX |
|  |  |
|  |  |
|  |  |
|  |  |
|  | ``` |
|  |  |
|  |  |
|  | ```{r createMenMatrix} |
|  |  |
|  |  |
|  |  |
|  | menResMat = lapply(menTablesV2, extractVariables) |
|  |  |
|  |  |
|  | ## looking at results and it appears we have peeled out headers. Now time to explore individual years |
|  |  |
|  | ``` |
|  |  |
|  | ```{r createMenDF} |
|  | menDF = mapply(createDF, menResMat, year = 1999:2012, |
|  | sex = rep("M", 14), SIMPLIFY = FALSE) |
|  | sapply(menDF, nrow) |
|  | #remove records with NA for age |
|  | menDF$`1999` = menDF$`1999`[-which(is.na(menDF$`1999`$age)),] |
|  | menDF$`2002` = menDF$`2002`[-which(is.na(menDF$`2002`$age)),] |
|  | menDF$`2005` = menDF$`2005`[-which(is.na(menDF$`2005`$age)),] |
|  | menDF$`2007` = menDF$`2007`[-which(is.na(menDF$`2007`$age)),] |
|  | menDF$`2009` = menDF$`2009`[-which(is.na(menDF$`2009`$age)),] |
|  | menDF$`2010` = menDF$`2010`[-which(is.na(menDF$`2010`$age)),] |
|  | menDF$`2012` = menDF$`2012`[-which(is.na(menDF$`2012`$age)),] |
|  |  |
|  |  |
|  | sapply(menDF, nrow) |
|  | cbMen = do.call(rbind, menDF) |
|  |  |
|  |  |
|  |  |
|  | ``` |
|  |  |
|  | ```{r Injest-WriteCSV} |
|  | library(tidyverse) |
|  |  |
|  |  |
|  | # READ-WRITE CSV |
|  | csv\_path <- "/Users/bgranger/Documents/SMU/7333\_Quantifying\_The\_World/Unit\_4/Unit\_4\_Repo/7333-Case-Study-Two/" |
|  |  |
|  | read\_or\_write = "read" |
|  |  |
|  | if (read\_or\_write == 'read'){ |
|  |  |
|  | cbMen <- read\_csv(paste(csv\_path, "cbMen.csv", sep=""), |
|  | col\_names = TRUE, |
|  | col\_types = cols(year = col\_double(), |
|  | sex = col\_character(), |
|  | name = col\_character(), |
|  | home = col\_character(), |
|  | age = col\_double(), |
|  | runTime = col\_double() |
|  | ) |
|  | ) |
|  |  |
|  | cbWomen <- read\_csv(paste(csv\_path, "cbWomen.csv", sep=""), |
|  | col\_names = TRUE, |
|  | col\_types = cols(year = col\_double(), |
|  | sex = col\_character(), |
|  | name = col\_character(), |
|  | home = col\_character(), |
|  | age = col\_double(), |
|  | runTime = col\_double() |
|  | ) |
|  | ) |
|  |  |
|  | } else if(read\_or\_write == 'write'){ |
|  |  |
|  | write\_csv(cbMen, paste(csv\_path, "cbMen.csv")) |
|  | write\_csv(cbWomen, paste(csv\_path, "cbWomen.csv")) |
|  |  |
|  |  |
|  | } |
|  |  |
|  |  |
|  |  |
|  | ``` |
|  |  |
|  | ```{r ProcessCSVs} |
|  |  |
|  | # MEN -------------------------------------------------------- |
|  |  |
|  | # MEN STARTING RECORD COUNT 70047 |
|  |  |
|  | # DROP THE ERROR FROM THE DATASET |
|  | cbMen <- filter(cbMen, !(year==2001 & runTime == min(runTime) & age == 70)) |
|  |  |
|  | # MEN RECORD COUNT: 70046 |
|  | # LOCATE AND REMOVE AGE OUTLIERS |
|  | cbMen <- filter(cbMen, !(year %in% c(2001:2003) & age %in% c(0,1, 2, 4))) |
|  |  |
|  | # MEN FINAL RECORD COUNT: 70039 |
|  |  |
|  | # WOMEN ------------------------------------------------------ |
|  |  |
|  | # WOMEN STARTING RECORD COUNT 75971 |
|  | # age = 55 name == "Loretta CUCE" & year == 2001 |
|  |  |
|  | filter(cbWomen, (year==2001 & runTime == min(runTime))) |
|  |  |
|  | ``` |
|  |  |
|  | ```{r ProducePlots} |
|  |  |
|  | # BOXPLOTS #### |
|  |  |
|  | # MENS ------------------------------------------------------ |
|  | # BOXPLOT: MENS runTime by Year |
|  |  |
|  | image\_location = "../7333-Case-Study-Two/Images/Boxplot/" |
|  |  |
|  | par(mfrow=c(1,1)) |
|  | jpeg(paste(image\_location, "mens\_runTime\_by\_Year.jpeg")) |
|  | boxplot(runTime~year,data=cbMen, main="Cherry Blossom: Men's Run Time By Year", xlab="Year", ylab="Time (Minutes)") |
|  | dev.off() |
|  |  |
|  |  |
|  | # BOXPLOT: MENS Age by Year |
|  | par(mfrow=c(1,1)) |
|  | jpeg(paste(image\_location, "mens\_age\_by\_Year.jpeg")) |
|  | boxplot(age~year,data=cbMen, main="Cherry Blossom: Men's Age by Year", xlab="Year", ylab="Runner's Age") |
|  | dev.off() |
|  |  |
|  |  |
|  | # WoMENS ------------------------------------------------------ |
|  |  |
|  |  |
|  | # QQ PLOTS #### |
|  |  |
|  | image\_location = "../7333-Case-Study-Two/Images/QQ\_Plot/" |
|  |  |
|  | # QQ PLOT: MENS AGE - ALL YEARS |
|  | par(mfrow=c(1,1)) |
|  | jpeg(paste(image\_location, "mens\_age\_qq\_plot.jpeg")) |
|  | qqnorm(cbMen$age, main = "Q-Q Plot: Men's Age") |
|  | qqline(cbMen$age) |
|  | dev.off() |
|  |  |
|  |  |
|  | # QQ PLOT: MENS AGE - BY YEARS |
|  |  |
|  | par(mar=c(1,1,1,1)) |
|  |  |
|  | par(mfrow=c(1,1)) |
|  | for (i in seq(from=1999, to=2012, by=1)){ |
|  | temp <- filter(cbMen, year == i) |
|  | jpeg(paste(image\_location, "mens\_age\_by\_", i, "\_qq\_plot.jpeg")) |
|  | qqnorm(temp$age, main = paste("Q-Q Plot: Men's Age: ", i, sep=" ")) |
|  | qqline(temp$age) |
|  | Sys.sleep(1) |
|  | dev.off() |
|  | } |
|  |  |
|  |  |
|  | # QQ PLOT: MENS RUNTIME - ALL YEARS |
|  | par("mar") |
|  | par(mar=c(1,1,1,1)) |
|  | par(mfrow=c(1,1)) |
|  | jpeg(paste(image\_location, "mens\_runtime\_qq\_plot.jpeg")) |
|  | qqnorm(cbMen$runTime, main = "Q-Q Plot: Men's runTime") |
|  | qqline(cbMen$runTime) |
|  |  |
|  | # QQ PLOT: MENS RUNTIME BY YEAR |
|  | par("mar") |
|  | par(mar=c(1,1,1,1)) |
|  |  |
|  | par(mfrow=c(1,1)) |
|  |  |
|  | for (i in seq(from=1999, to=2012, by=1)){ |
|  | temp <- filter(cbMen, year == i) |
|  | jpeg(paste(image\_location, "mens\_runtime\_by\_", i, "\_qq\_plot.jpeg")) |
|  | qqnorm(temp$runTime, main = paste("Q-Q Plot: Men's Runtime: ", i, sep=" ")) |
|  | qqline(temp$runTime) |
|  | Sys.sleep(1) |
|  | dev.off() |
|  | } |
|  |  |
|  | # DENSITY PLOTS #### |
|  |  |
|  | image\_location = "../7333-Case-Study-Two/Images/Density/" |
|  |  |
|  | # DENSITY CURVE - MENS AGE BY YEAR |
|  | for (i in seq(from=1999, to=2012, by=1)){ |
|  | temp <- filter(cbMen, year == i) |
|  | jpeg(paste(image\_location, "mens\_age\_by\_", i, "\_density.jpeg")) |
|  | hist(temp$age, mean(temp$age), prob=T, main = paste("Density: Men's Age: ", i, sep=" ")) |
|  | lines(density(temp$age)) |
|  | Sys.sleep(1) |
|  | dev.off() |
|  | } |
|  |  |
|  | # DENSITY CURVE - MENS RUNTIME BY YEAR |
|  |  |
|  | for (i in seq(from=1999, to=2012, by=1)){ |
|  | temp <- filter(cbMen, year == i) |
|  | jpeg(paste(image\_location, "mens\_runtime\_by\_", i, "\_density.jpeg")) |
|  | hist(temp$runTime, mean(temp$runTime), prob=T, main = paste("Density: Men's runTime: ", i, sep=" ")) |
|  | lines(density(temp$runTime)) |
|  | Sys.sleep(1) |
|  | dev.off() |
|  | } |
|  |  |
|  | ``` |
|  |  |
|  |  |
|  | ```{r Testcode} |
|  |  |
|  |  |
|  | par(mfrow=c(1,1)) |
|  | boxplot(age~year,data=cbWomen, main="Cherry Blossom: Women's Age By Year", xlab="Year", ylab="Age") |
|  |  |
|  |  |
|  |  |
|  | # QQ PLOT: AGE BY YEAR |
|  | #par("mar") |
|  | #par(mar=c(1,1,1,1)) |
|  | #jpeg(paste(image\_location, "mens\_age\_by\_year\_qq\_plot.jpeg")) |
|  | #par(mfrow=c(8,2)) |
|  | #for (i in seq(from=1999, to=2012, by=1)){ |
|  | # temp <- filter(cbMen, year == i) |
|  |  |
|  | # qqnorm(temp$age, main = paste("Q-Q Plot: Men's Age: ", i, sep=" ")) |
|  | # qqline(temp$age) |
|  | #} |
|  | #dev.off() |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | dev.set(dev.next()) |
|  | temp <- filter(cbMen, year == 1999) |
|  | hist(temp$age, 20, prob=T) |
|  | lines(density(temp$age)) |
|  | ``` |
|  |  |
|  | ```{r womenSetup} |
|  | #URL for WOMEN and Library |
|  |  |
|  | library(XML) |
|  | ubase = "http://www.cherryblossom.org/" |
|  |  |
|  | #### Revised URLS |
|  | womenURLs = |
|  | c("results/1999/cb99f.html", #"cb99f.htm" |
|  | "results/2000/Cb003f.htm", #"cb003f.htm" |
|  | "results/2001/oof\_f.html", #"results/2001/oof\_f.html" |
|  | "results/2002/ooff.htm", #"results/2002/ooff.htm" |
|  | "results/2003/CB03-F.HTM", #"results/2003/CB03-F.HTM" |
|  | "results/2004/women.htm", #"results/2004/women.htm" |
|  | "results/2005/CB05-F.htm", #"results/2005/CB05-F.htm" |
|  | "results/2006/women.htm", #"results/2006/women.htm" |
|  | "results/2007/women.htm", #"results/2007/women.htm" |
|  | "results/2008/women.htm", #"results/2008/women.htm" |
|  | "results/2009/09cucb-F.htm", #"results/2009/09cucb-F.htm" |
|  | "results/2010/2010cucb10m-f.htm", #"results/2010/2010cucb10m-f.htm" |
|  | "results/2011/2011cucb10m-f.htm", #"results/2011/2011cucb10m-f.htm" |
|  | "results/2012/2012cucb10m-f.htm" #"results/2012/2012cucb10m-f.htm" |
|  | ) |
|  | #### |
|  |  |
|  |  |
|  | #### women URLS |
|  | urlsV3 = paste(ubase, womenURLs, sep="") |
|  | urlsV3[1:4] |
|  |  |
|  | urlsV3[1] |
|  |  |
|  | ``` |
|  |  |
|  | ```{r retrieveWomenData} |
|  |  |
|  | womenTables = lapply(urlsV3, extractResTableV3) |
|  |  |
|  | #womenTables[[3]][1:3] = mensTables[[3]][1:3] |
|  | #womenTables[[3]] = append(x=womenTables[[3]], values=mensTables[[3]][4:5], after = 3) |
|  |  |
|  | years1 = 1999:2012 |
|  |  |
|  |  |
|  |  |
|  | womenTablesv2 <- mapply(extractResTableV3, url = urlsV3, year = years1) |
|  | names(womenTablesv2) = years1 |
|  | sapply(womenTablesv2, length) |
|  |  |
|  | ###womenTablesv2[[1]][1:10] |
|  | ###womenTablesv2[[2]][1:10] |
|  | ###womenTablesv2[[3]][1:10] |
|  | ###womenTablesv2[[4]][1:10] |
|  | ###womenTablesv2[[5]][1:10] |
|  | ###womenTablesv2[[6]][1:10] |
|  | ###womenTablesv2[[7]][1:10] |
|  | ###womenTablesv2[[8]][1:10] |
|  |  |
|  | save(womenTablesv2, file = "CBWoMenTextTables\_Houssaye.rda") |
|  | ``` |
|  |  |
|  | ```{r fix2001Women} |
|  | womenTablesv2[["2001"]][1:3] <- menTablesV2[["2001"]][1:3] |
|  | womenTablesv2[["2001"]] <- append(x=womenTablesv2[["2001"]], values=menTablesV2[["2001"]][4:5], after = 3) |
|  | ``` |
|  |  |
|  | ```{r parseWomenData} |
|  | separatorIdxWomen = grep("^===", womenTablesv2[["2006"]]) |
|  | separatorRowWomen = womenTablesv2[['2006']][separatorIdxWomen] |
|  | separatorRowXWomen = paste(substring(separatorRowWomen, 1, 63), " ", |
|  | substring(separatorRowWomen, 65, nchar(separatorRowWomen)), |
|  | sep = "") |
|  | womenTablesv2[['2006']][separatorIdxWomen] = separatorRowXWomen |
|  |  |
|  | womenResMat = sapply(womenTablesv2, extractVariables) |
|  | womenDF = mapply(createDF, womenResMat, year = 1999:2012, |
|  | sex = rep("F", 14), SIMPLIFY = FALSE) |
|  | sapply(womenDF, nrow) |
|  | sapply(womenDF, nrow) |
|  |  |
|  | ``` |
|  | ```{r removeWomenNA} |
|  | womenDF$`1999` = womenDF$`1999`[-which(is.na(womenDF$`1999`$age)),] |
|  | womenDF$`2002` = womenDF$`2002`[-which(is.na(womenDF$`2002`$age)),] |
|  | womenDF$`2005` = womenDF$`2005`[-which(is.na(womenDF$`2005`$age)),] |
|  | womenDF$`2006` = womenDF$`2006`[-which(is.na(womenDF$`2006`$age)),] |
|  | womenDF$`2007` = womenDF$`2007`[-which(is.na(womenDF$`2007`$age)),] |
|  | womenDF$`2009` = womenDF$`2009`[-which(is.na(womenDF$`2009`$age)),] |
|  | cbWomen = do.call(rbind, womenDF) |
|  | ``` |
|  |  |
|  | ```{r writeWomenCSV} |
|  | # READ-WRITE CSV |
|  | cbWomen\_path <- "../7333-Case-Study-Two/cbWomen.csv" |
|  |  |
|  | read\_or\_write = "read" |
|  |  |
|  | if (read\_or\_write == 'read'){ |
|  |  |
|  | cbWomen <- read\_csv(cbWomen\_path, col\_names = TRUE, |
|  | col\_types = cols(year = |
|  | col\_double(), |
|  | sex = col\_character(), |
|  | name = col\_character(), |
|  | home = col\_character(), |
|  | age = col\_double(), |
|  | runTime = col\_double() |
|  | ) |
|  | ) |
|  |  |
|  | } else if(read\_or\_write == 'write'){ |
|  |  |
|  | write\_csv(cbWomen, cbWomen\_path) |
|  | } |
|  | ``` |