## **Abstract**

Assignment: Case Studies in Data Science With R

Choose one of the following problems from the back of the chapter: **7,** 10, 11, or 17. Write a report on your analysis, including introduction, background, methods, results, and conclusions/discussions. Note that the focus of this study is getting the data into the correct format. The analysis section is going to be relatively simple from a statistical standpoint—graphics, descriptive statistics, etc., to answer the question from the problem that you chose.

Q.7 Follow the approach developed in Section 2.2 to read the files for the female runners and then process them using the functions in Section 2.3 to create a data frame for analysis. You may need to generalize the createDF() and extractVariables() functions to handle additional oddities in the raw text files.

## **Introduction**

One example of the many annual road races is the Cherry Blossom Ten Mile Run

held in Washington D.C. in early April when the cherry trees are typically in bloom.

The Cherry Blossom started in 1973 as a training run for elite runners who were planning to compete in the Boston Marathon. It has since grown in popularity and in 2012 nearly 17,000 runners ranging in age from 9 to 89 participated. The race has become so popular that entrants are chosen via a lottery or they guarantee a spot by raising $500 for an official race charity. After each year’s race, the organizers publish the results at http://www.cherryblossom.org/ (see Figure 2.1). These data offer a tremendous resource for learning about the relationship between age and performance.



Figure 2.1: Screen Shot of Cherry Blossom Run Web site. This page contains links to each year’s race results. The year 1999 is the earliest for which they provide data. Men’s and women’s results are listed separately.

The publicly available race results from the Cherry Blossom Ten Mile Run can be scraped from the Web and read into R [3] for analysis. The currently published results include all years from 1999 to 2012. The task of scraping the Web site and formatting the results in a way that can be analyzed in R is a bit challenging because the information reported and the format of this information changes from year to year. Some simple differences in format occur in the format of the table header and the use of footnotes. The tables also include many mistakes, e.g., values that begin in the wrong column, missing headers, and so on. All in all, the acquisition of the data is quite straightforward, but it is an iterative process as we uncover several small errors. We do this statistically, i.e., we examine summary statistics and plots of the data we have read into R, find anomalies, such as all the runners in 2003 being under 9, cross check sample observations with the original tables, modify our code to handle these problem cases in a way that is as general as possible, recreate our data, and repeat. This is the story of “messy” data. It is the focus of Section 2.2 and Section 2.3 of this chapter. Additionally, Section 2.7 covers the topic of scraping the Web for the race results, for those who are interested in the entire process of data acquisition.

## **Background**

The dataset contains a total of records and ten initial variables. A full list of the initial variable descriptions can be found in Table 1:

Table 1: Variable Descriptions and Types

|  |  |  |
| --- | --- | --- |
| **Variable** | **Description** | **Type** |
| Place |  |  |
| Div / Tot |  |  |
| Num |  |  |
| Name |  |  |
| Ag |  |  |
| Hometown |  |  |
| 5 Mile |  |  |
| Gun Time |  |  |
| Net Tim |  |  |
| Pace |  |  |

**Methods**

## **Results**

## **Future Work, Discussion Conclusions, and Next Steps**

## **References**

## **Appendix - R Code**

setwd("~/EMD/CaseStudies/Runners/WebPages")

m2012 = read.table(file="MenTxt/2012.txt", skip = 8)

els = readLines("MenTxt/2012.txt")

els[1:10]

els2011 = readLines("MenTxt/2011.txt")

els2011[1:10]

eqIndex = grep("^===", els)

eqIndex

first3 = substr(els, 1, 3)

which(first3 == "===")

spacerRow = els[eqIndex]

headerRow = els[eqIndex - 1]

body = els[ -(1:eqIndex) ]

headerRow = tolower(headerRow)

ageStart = regexpr("ag", headerRow)

ageStart

age = substr(body, start = ageStart, stop = ageStart + 1)

head(age)

summary(as.numeric(age))

blankLocs = gregexpr(" ", spacerRow)

blankLocs

searchLocs = c(0, blankLocs[[1]])

Values = mapply(substr, list(body),

start = searchLocs[ -length(searchLocs)] + 1,

stop = searchLocs[ -1 ] - 1)

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(colNames, headerRow, searchLocs)

{

sapply(colNames,

function(name, headerRow, searchLocs)

{

startPos = regexpr(name, headerRow)[[1]]

if (startPos == -1)

return( c(NA, NA) )

index = sum(startPos >= searchLocs)

c(searchLocs[index] + 1, searchLocs[index + 1] - 1)

},

headerRow = headerRow, searchLocs = searchLocs )

}

searchLocs = findColLocs(spacerRow)

ageLoc = selectCols("ag", headerRow, searchLocs)

ages = mapply(substr, list(body),

start = ageLoc[1,], stop = ageLoc[2, ])

summary(as.numeric(ages))

shortColNames = c("name", "home", "ag", "gun", "net", "time")

locCols = selectCols(shortColNames, headerRow, searchLocs)

Values = mapply(substr, list(body), start = locCols[1, ],

stop = locCols[2, ])

class(Values)

colnames(Values) = shortColNames

head(Values)

tail(Values)[ , 1:3]

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) ]

# 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

invisible(Values)

}

mfilenames = paste("MenTxt/", 1999:2012, ".txt", sep = "")

menFiles = lapply(mfilenames, readLines)

names(menFiles) = 1999:2012

menResMat = lapply(menFiles, extractVariables)

length(menResMat)

sapply(menResMat, nrow)

### The 2001 results for women are missing the === and the column names.

### Can we pick it up from the 2001 men? YES! Make an exercise

#wfilenames = paste("WomenTxt/", 1999:2012, ".txt", sep = "")

#womenTables = lapply(wfilenames, readLines)

#womenTables[[3]][1:5]

#names(womenTables) = 1999:2012

#womenResMat = lapply(womenTables, extractVariables)

#head(womenResMat[[3]], 10)

#tail(womenResMat[[3]], 10)

age = as.numeric(menResMat[['2012']][ , 'ag'])

tail(age)

age = sapply(menResMat,

function(x) as.numeric(x[ , 'ag']))

pdf("CB\_BoxplotAgeByYr.pdf", width = 8, height = 5)

oldPar = par(mar = c(4.1, 4.1, 1, 1))

boxplot(age, ylab = "Age", xlab = "Year")

par(oldPar)

dev.off()

head(menFiles[['2003']])

menFiles[['2006']][2200:2205]

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 )

}

menResMat = lapply(menFiles, extractVariables)

#womenResMat = lapply(womenFiles, extractVariables)

age = sapply(menResMat,

function(x) as.numeric(x[ , 'ag']))

pdf("CB\_BoxplotAgeByYrRevised.pdf", width = 8, height = 5)

oldPar = par(mar = c(4.1, 4.1, 1, 1))

boxplot(age, ylab = "Age", xlab = "Year")

par(oldPar)

dev.off()

sapply(age, function(x) sum(is.na(x)))

age2001 = age[["2001"]]

grep("^===", menFiles[['2001']])

badAgeIndex = which(is.na(age2001)) + 5

menFiles[['2001']][ badAgeIndex ]

badAgeIndex

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)

}

menResMat = lapply(menFiles, extractVariables)

#womenResMat = lapply(womenFiles, extractVariables)

which(age2001 < 5)

menFiles[['2001']][ which(age2001 < 5) + 5 ]

charTime = menResMat[['2012']][, 'time']

head(charTime, 5)

tail(charTime, 5)

timePieces = strsplit(charTime, ":")

timePieces[[1]]

tail(timePieces, 1)

timePieces = sapply(timePieces, as.numeric)

runTime = sapply(timePieces,

function(x) {

if (length(x) == 2) x[1] + x[2]/60

else 60\*x[1] + x[2] + x[3]/60

})

summary(runTime)

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

useTime = if( !is.na(Res[1, 'net']) )

Res[ , 'net']

else if( !is.na(Res[1, 'gun']) )

Res[ , 'gun']

else

Res[ , 'time']

runTime = convertTime(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)

}

menDF = mapply(createDF, menResMat, year = 1999:2012,

sex = rep("M", 14), SIMPLIFY = FALSE)

warnings()[ c(1:2, 49:50) ]

sapply(menDF, function(x) sum(is.na(x$runTime)))

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)

}

menDF = mapply(createDF, menResMat, year = 1999:2012,

sex = rep("M", 14), SIMPLIFY = FALSE)

sapply(menDF, function(x) sum(is.na(x$runTime)))

separatorIdx = grep("^===", menFiles[["2006"]])

separatorRow = menFiles[['2006']][separatorIdx]

separatorRowX = paste(substring(separatorRow, 1, 63), " ",

substring(separatorRow, 65, nchar(separatorRow)),

sep = "")

menFiles[['2006']][separatorIdx] = separatorRowX

menResMat = sapply(menFiles, extractVariables)

menDF = mapply(createDF, menResMat, year = 1999:2012,

sex = rep("M", 14), SIMPLIFY = FALSE)

#separatorIdx = grep("^===", womenFiles[["2006"]])

#separatorRow = womenFiles[['2006']][separatorIdx]

#separatorRowX = paste(substring(separatorRow, 1, 63), " ",

# substring(separatorRow, 65, nchar(separatorRow)),

# sep = "")

#womenFiles[['2006']][separatorIdx] = separatorRowX

#womenResMat = sapply(womenFiles, extractVariables)

#womenDF = mapply(createDF, womenResMat, year = 1999:2012,

# sex = rep("W", 14), SIMPLIFY = FALSE)

pdf("CB\_BoxplotTimeByYr.pdf", width = 8, height = 5)

boxplot(sapply(menDF, function(x) x$runTime),

xlab = "Year", ylab = "Run Time (min)")

dev.off()

cbMen = do.call(rbind, menDF)

save(cbMen, file = "cbMen.rda")

dim(cbMen)

load("cbMen.rda")

pdf("CB\_Overplot.pdf", width = 8, height = 6)

oldPar = par(mar = c(4.1, 4.1, 1, 1))