Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

MSDS 7333 - Quantifying the World - Case Study 4 (Unit 8)
May 22, 2018

Abstract

It is said about 80% of a data scientist job consists of cleaning data. This is because data sets often have missing or inaccurate data. Furthermore, when scraping data from the web many times the data is in a format that is not uniform and thus unable to use. In this paper, we format web scraped data from the Cherry Blossom Ten Mile Run from 1999 to 2012. We will show the steps taken to organize the data and how to check if the data looks correct so that it can be used for analysis.

Introduction

In the data science community, there is a saying “garbage in garbage out”. This describes how your inferences can only be as good as the data you are inferring about. Good data to analyze means the data set being analyzed has the correct information for the variables and a complete data set with no missing observations. On top of this, there need to be uniform column names as well as observations with the same metrics to be able to draw out any useful insight.

When scraping data from the web we often run into the situation where data is formatted in a way that is not organized or formatted so that it can be analyzed. In this case study, we scraped data from Cherry Blossom Ten Mile Run from 1999-2012. The data came in as a string of characters with different formatting for each year. The data set has the variables, Place, Div/tot, Num, Name, Age, Hometown, 5 Mile Time and Pace. We address the differences in each data set from the respective years and attempt to make the data uniform so a thorough analysis can be completed.

Literature Review

Access to the web, and more over the access to data has been a huge breakthrough for the data science community. This is because this access to data has given us the ability to solve problems we could have never done before. Since being able to extract data from the web is so valuable, there have been many different techniques developed to scrape data from the web successfully.

“One of the most exploited features in Web Data Extraction is the semi-structured nature of Web pages.” [2] In order to access this structure, Tree based techniques have been developed to take advantage of the structure many web pages are set up in. Many times, web pages can be viewed as a tree with roots to specific information that we might want to gather. These roots are labeled, or tagged, with HTML which we can then utilize to easily access this information.

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

Furthermore, by utilizing developed techniques to scrape data, we can better predict the types of errors we may run into when cleaning the data. Being able to know the types of errors we might run into, helps the process of cleaning data by being able to use similar cleaning techniques from past experience to speed up the process of getting our data into a format that can be analyzed.

In this case study, we repeatedly ran into the same issues of making sure all our headers were succinct with one, there were no missing values or why there were missing values, and making sure there were no outliers or each observation was on the same scale as the others.

Background

The data set in this case study contains race times for female runners that participated in the ten-mile Cherry Blossom Run in Washington DC. This data spans a period of 13 years (1999 - 2012) and was obtained from the website (http://www.cherryblossom.org/). The data was scrapped using an R function and placed in data frames separated by year.

Our final data set contains 75971 observations with 6 variables (year, sex, name, home, age, runtime). A major issue encountered was the different format each year's data was in. Each year's data contained different number of variables as well as names so everything had to be standardized before we could proceed with our analysis. Headers of each file can be seen on Table 1 below.

Table 1: Variables (headers) for each Year

Year	**Columns**
1999	PLACE DIV /TOT NAME AG HOMETOWN TIME PACE
2000	PLACE DIV /TOT NUM NAME AG HOMETOWN GUN TIM NET TIM
2001	-
2002	Place Num Name Ag Hometown Net Gun
2003	Place Div /Tot Num Name Ag Hometown Gun Tim Net Tim
2004	Place Div /Tot Num Name Ag Hometown Net Gun
2005	Place Div /Tot Name Ag Hometown Net Gun Pace
2006	Place Div/Tot Num Name Ag Hometown Net Tim Gun Tim Pace S
2007	Place Div /Tot Num Name Ag Hometown Time Pace S Split
2008	Place Div /Tot Num Name Ag Hometown 5 Mi Pace 10 Km Pace Time Pace
2009	Place Div /Tot Num Name Ag Hometown Gun Tim Net Tim Pace S
2010	Place Div /Tot Num Name Ag Hometown 5 Mile Gun Tim Net Tim Pace S
2011	Place Div /Tot Num Name Ag Hometown 5 Mile Time Net Tim Pace S
2012	Place Div /Tot Num Name Ag Hometown 5 Mile Time Pace S

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

Methods

The first thing we did was take a look at the data from each year to see what format that data is in. From here we can see the data does not have uniform variables meaning the column names are not the same, some years have more or less columns, and the columns are not in the same order. Furthermore, we can already start to see there are missing observations if we look at Table 2. Women's 2011 under the "5Mile" column. We can see all of this by looking at just line 6 through 10 for the year 2012 and 2011 respectively. (please see "section 1 in the appendix.)

Table 2. Women's 2012

Place	Div/Tot	Num	Name	Ag	Hometown	5 mile	Time	Pace	s
=	=	=	=	=	=	=	=	=	=
1	1/27/81	2	Jelliah Tinega	26	Kenya	26:48	54:02	5:25	!
2	2/2781	24	Malika Mejdoub	29	Ethiopia	27.09	54:24	5:27	!

Table 3. Women's 2011

Place	Div/Tot	Num	Name	Ag	Hometown	5 mile	Time	Net Time	Pace	S
=	=	=	=	=	=	=	=	=	=	=
1	1/2706	14	Julliah Tinega	25	Kenya		54:02	54:02	5:25	!
2	1/937	16	Risper Gesabwa	22	Kenya	27:17	54:03	54:03	5:25	!

As we can see, in both data sets, line 8 is where there are "=" signs separating the columns from the data. Even though there are similar column headers we can see the 2011 data set contains "Net Time" which is not contained in 2012. Based on this, we can start to pull column headers before row 8 and data after row 8.

Now that we understand our data's format and discrepancies, we begin to make the data uniform. We will do this by focusing on each individual column and then proceed to the next.

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

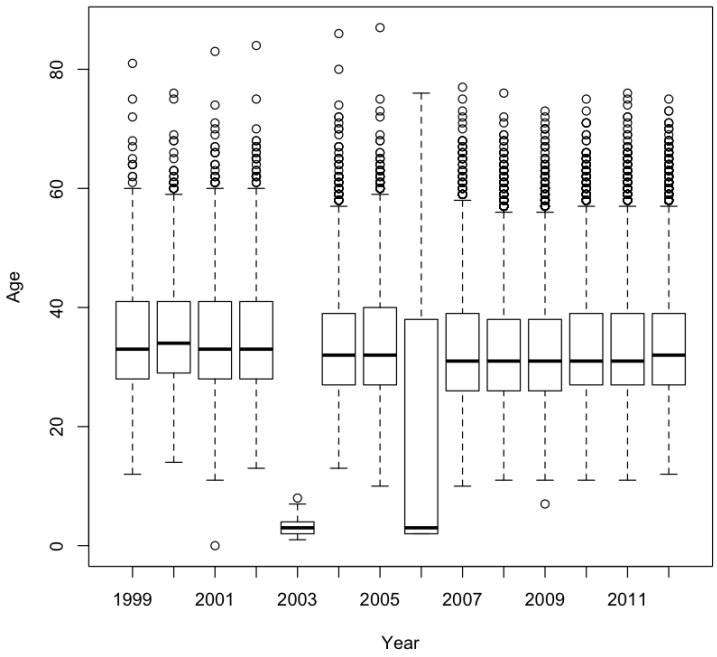
The first variable we focused on was the “age” column. For the “age” column we first have to figure out where the columns start and end. We do this by searching for the “breaks” in the spacer row and append a 0 to signal where these breaks are. (please see “Section 2 in the appendix.) We can then apply this idea to all of the other columns to get the exact values that we need.

Now that we know where the age begins and ends, we need to make uniform column names. (please see “Section 3 in the appendix.) We did this by making a variable that will hold all of the column names that we will use and apply to the data sets for each “year”.

Once all the years have the same column headers we then need to find what we missed and fill in all of the missing values with NAs. (please see “Section 4 in the appendix.) This will specially show what data we need for each column for each year.

Now we start to validate whether our data is correct. We can look at Table 4. Boxplot below to see something is off with our data in year 2003 and 2006 respectively. (please see “Section 5 in the appendix.)

Table 4. Boxplot of Age-Year

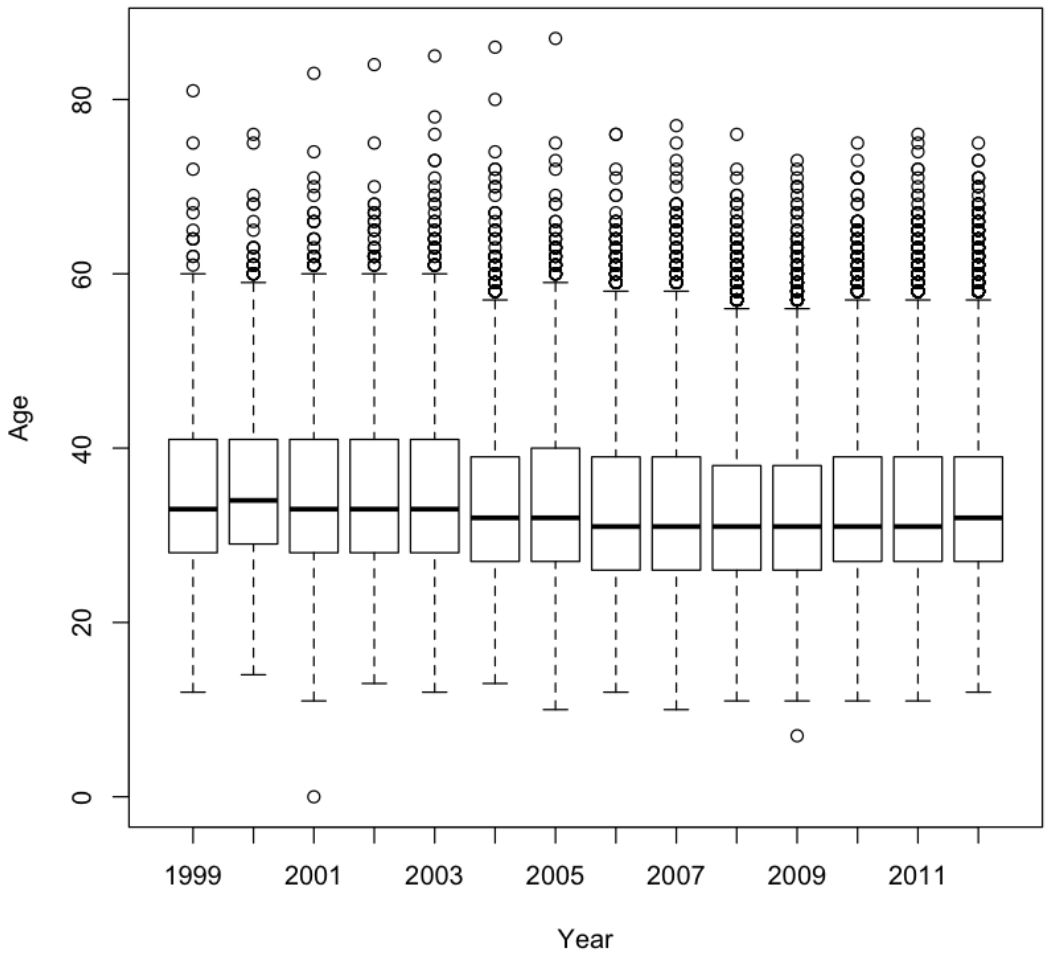


Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

After looking just at the data in these two years, we see the age is right next to Hometown and so for most of the data we are not picking up all of the values. Once we compensate for this and allow our data to account for spaces, our new boxplot shows a more even distribution of ages for these two years. (please see “Section 6 in the appendix.) Please see Table 5. Corrected Boxplot below.

Table 4. Corrected Boxplot of Age-Year



Now that our data looks clean we need to address the NAs observed. We do this by removing all special characters and foot notes. (please see “Section 7 in the appendix.) Finally, we address the age value from the above boxplot that shows a 0. We then replace this with the

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

median value for 2001 and our age column is finally in a working condition ready to be analyzed. We decided on the median because that large sample set makes the median number resilient to outliers which we can clearly see in the right tail from our box plots.

Now that we have completed the process for the 'age' column we will work on the 'time' column. The time column is fairly clean except for some runners that finished the race with times longer than an hour thus making the times for some runs in hours: minutes: second and other just minutes: seconds. To correct for this, we just convert all of our time data into minutes and seconds. Once we have all of our run times in the correct format we have to go back through again and correct for all NAs and special characters. (please see "Section 8" in the appendix.)

Next, we create a data frame that has the "year" and "sex" column added to it. We did this because these pieces of data will be useful for analysis later on. (please see "Section 9" in the appendix.) By doing this we can see there are many NAs and so we must correct for this. We fix this again by removing NAs and any special characters. This will lead us to having all columns formatted the same way, with the same column names, and clean data. Once we have this we can combine all of the data into one complete data set.

Results

The results from method section above show that we have a complete data set ready for analysis after scraping the data from the web. Our new data set contains 75,981 observations, 6 variables (year, sex, name, home, age, and runtime), and no missing or incomplete data.

The main issues we face, which we will discuss further in our conclusion, were addressing: NAs/missing values, special characters, headers/footers, spacing, and outliers.

Future Work, Discussion Conclusions, and Next Steps

The results show that we have a completed data set ready for analysis. It is said about 80% of a data scientist job consists of cleaning data. For data scientist, we don't necessarily need a rocket ship to get to the next frontier, but rather a washing machine. That is, if we had a machine that could consistently output clean data we could spend the majority of our time analyzing data instead of cleaning it.

In this case study we scraped data from the web for the Cherry Blossom Ten Mile Run from 1999-2012. The data came in as a string of characters with different formatting for each year. This data set had the following variables: Place, Div/tot, Num, Name, Age, Hometown, 5 Mile Time and Pace. In these data sets we experienced problems such as: the data not being formatted uniformly and having missing values and different column names. The common issues we ran into were removing missing values, then formatting all of the values the same way, then checking the format and finally if there were more missing values, repeating this process until the data was clean. On top of this, finding where the data started and ended and removing headers/footers also needed to be taken into consideration when cleaning the data.

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

After taking into the time and research it took to complete this case study there is something to be said in regards to the washing machine analogy. I could imagine running simple regressions and summary statistics to find interesting information to take maybe a tenth of the time it took to prepare the data. On top of having access to the internet and fast computers, the addition of consistent clean data could be a revolution in the data science community.

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

References

1. Deborah Nolan Duncan Tmple Lang “Data Science in R A Case Studies Approach to Computational Reasoning and Problem Solving”
2. Emilio Ferraraa,∗ , Pasquale De Meob , Giacomo Fiumarac , Robert Baumgartnerd http://www.emilio.ferrara.name/wp-content/uploads/2011/07/survey-csur.pdf, Web Data Extraction, Applications and Techniques: A Survey

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

Appendix

Code:

```
library(XML)
ubase = "http://www.cherryblossom.org/"

menURLs =
  c("results/1999/cb99m.html", "results/2000/Cb003m.htm", "results/2001/oof_m.html",
    "results/2002/oofm.htm", "results/2003/CB03-M.HTM",
    "results/2004/men.htm", "results/2005/CB05-M.htm",
    "results/2006/men.htm", "results/2007/men.htm",
    "results/2008/men.htm", "results/2009/09cucb-M.htm",
    "results/2010/2010cucb10m-m.htm",
    "results/2011/2011cucb10m-m.htm",
    "results/2012/2012cucb10m-m.htm")

urls = paste(ubase, menURLs, sep = "")

urls[1:3]

extractResTable =
  #
  # 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)
  {
    doc = htmlParse(url)

    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) {
             # Get preformatted text from <pre> elements
      pres = getNodeSet(doc, "//pre")
      txt = xmlValue(pres[[1]])
      els = strsplit(txt, "\n")[[1]]
     }
```

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

```
    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)
  }

years = 1999:2012
menTables = mapply(extractResTable, url = urls, year = years)
names(menTables) = years
sapply(menTables, length)

menTables$'2001'[0:10]

womenURLs =
  c("results/1999/cb99f.html", "results/2000/Cb003f.htm", "results/2001/oof_f.html",
    "results/2002/ooff.htm", "results/2003/CB03-F.HTM",
    "results/2004/women.htm", "results/2005/CB05-F.htm",
    "results/2006/women.htm", "results/2007/women.htm",
    "results/2008/women.htm", "results/2009/09cucb-F.htm",
    "results/2010/2010cucb10m-f.htm",
    "results/2011/2011cucb10m-f.htm",
    "results/2012/2012cucb10m-f.htm")

urls = paste(ubase, womenURLs, sep = "")

urls[1:3]

extractResTable =
  #
  # 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)
  {
    doc = htmlParse(url)

    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
```

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

```
    # 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) {
           # 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)
 }

years = 1999:2012
womenTables = mapply(extractResTable, url = urls, year = years)
names(womenTables) = years
sapply(womenTables, length)

#section 1
womenTables$'2012'[6:10]
womenTables$'2011'[6:10]

wmn2012 = womenTables$'2012'
eqIndex = grep("^===", wmn2012)
eqIndex

spacerRow = wmn2012[eqIndex]
headerRow = wmn2012[eqIndex - 1]
body = wmn2012[ -(1:eqIndex) ]

headerRow = tolower(headerRow)

ageStart = regexpr("ag", headerRow)
ageStart

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

summary(as.numeric(age))

#section 2
blankLocs = gregexpr(" ", spacerRow)
blankLocs
```

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

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

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))

#section 3
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)

#Section 4

colnames(Values) = shortColNames
head(Values)
#section 7
extractVariables =
```

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

```
 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)
}

names(womenTables) = 1999:2012

womenTables$'2001'[0:10]

womenTables$'2001'[1:3] = menTables$'2001'[3:5]
womenTables$'2001'[1:10]

womenResMat = lapply(womenTables, extractVariables)
length(womenResMat)

age = sapply(womenResMat,
          function(x) as.numeric(x[ , 'ag']))
#section5
boxplot(age, ylab = "Age", xlab = "Year")

womenTables[['2003']][1:15]
womenTables[['2006']][1:15]

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 )
}
womenResMat = lapply(womenTables, extractVariables)
age = sapply(womenResMat,
          function(x) as.numeric(x[ , 'ag']))
#section 6
boxplot(age, ylab = "Age", xlab = "Year")

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

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

```
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)
}
womenResMat = lapply(womenTables, extractVariables)
age = sapply(womenResMat,
             function(x) as.numeric(x[ , 'ag']))

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

sapply(age, function(x) which(x < 5))

age$`2001`[2611]
age$`2001`[2611] = median(age$`2001`)
age$`2001`[2611]

charTime = womenResMat[['2012']][, 'time']
head(charTime, 5)

tail(charTime, 5)

timePieces = strsplit(charTime, ":")
timePieces[[1]]

tail(timePieces, 1)

timePieces = sapply(timePieces, as.numeric)
runTime = sapply(timePieces,
```

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

```
            function(x) {
              if (length(x) == 2) x[1] + x[2]/60
              else 60*x[1] + x[2] + x[3]/60
            })
summary(runTime)

#section 8
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
                  })
}
# Section 9
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)
}

womenDF = mapply(createDF, womenResMat, year = 1999:2012,
            sex = rep("W", 14), SIMPLIFY = FALSE)

sapply(womenDF, 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 != "" ])
```

Unit 8 Case Study – Web Scrapped Data Cleaning

Evan Giakoumakis

```
  # 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)
}

womenDF = mapply(createDF, womenResMat, year = 1999:2012,
                 sex = rep("W", 14), SIMPLIFY = FALSE)

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

cbWomen = do.call(rbind, womenDF)
save(cbWomen, file = "cbWomen.rda")

dim(cbWomen)

summary(cbWomen)
```