

# Modeling Runners' Times in the Cherry Blossom Race

Justin Howard, Stuart Miller, Paul Adams

September 22, 2020

## 1 Introduction

The internet is a vast open resource for data, but internet data can be messy and difficult to wrangle. In this case study we collect data on participants of the Cherry Blossom Race from the Cherry Blossom Race website<sup>1</sup>. We find that there are a number of issues present in the data, including inconsistent web addressing formatting, inconsistent formatting, and missing data. We collect and clean this data to provide the Men's and Women's information in a format suitable for analysis.

## 2 Methods

### 2.1 Data

#### 2.1.1 Data Collection

In this case study, we collected data on participants in the Cherry Blossom Race. We scraped the data for Men and Woman from 1999 to 2012. The data was stored under the base address in the following form, where the year is between 1999 and 2012 and page is a specific page name.

`http://www.cherryblossom.org/results/<year>/<page>`

The specific addresses where the data is stored are given in table 1. While the structure follows `results/<year>` consistently, the page naming for each year is not consistent. In some cases, the names for the men's and women's pages are `men` and `women`, respectively. In other cases, a code is used such as `cb99m` and `cb99f` in 1999 for `men` and `women` respectively.

**Table 1. Web Page Pages**

Year	Men's Pages	Women's Pages
1999	<code>results/1999/cb99m.html</code>	<code>results/1999/cb99f.html</code>
2000	<code>results/2000/Cb003m.htm</code>	<code>results/2000/Cb003f.htm</code>

---

<sup>1</sup>See <http://www.cherryblossom.org/>.

Year	Men's Pages	Women's Pages
2001	results/2001/oof_m.html	results/2001/oof_f.html
2002	results/2002/oofm.htm	results/2002/ooff.htm
2003	results/2003/CB03-M.HTM	results/2003/CB03-F.HTM
2004	results/2004/men.htm	results/2004/women.htm
2005	results/2005/CB05-M.htm	results/2005/CB05-F.htm
2006	results/2006/men.htm	results/2006/women.htm
2007	results/2007/men.htm	results/2007/women.ht
2008	results/2008/men.htm	results/2008/women.htm
2009	results/2009/09cucb-M.htm	results/2009/09cucb-F.htm
2010	results/2010/2010cucb10m-m.htm	results/2010/2010cucb10m-F.htm
2011	results/2011/2011cucb10m-m.htm	results/2011/2011cucb10m-F.htm
2012	results/2012/2012cucb10m-m.htm	results/2012/2012cucb10m-F.htm

```
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]
```

```
## [1] "http://www.cherryblossom.org/results/1999/cb99m.html"
## [2] "http://www.cherryblossom.org/results/2000/Cb003m.htm"
## [3] "http://www.cherryblossom.org/results/2001/oof_m.html"
```

```
extractResTable =
  # takes a list of websites from the cherry blossom race
  # a list of years corresponding to the year the result is for
  # and the gender of the participant
  # Retrieve data from web site,
  # find the preformatted text,
  # and write lines or return as a character vector.
  # returns a list of strings corresponding to lines in the web url
```

```

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)
    els = gsub("\r", "", els)
  }
  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
menTables = mapply(extractResTable, url = urls, year = years)
names(menTables) = years
sapply(menTables, length)

```

```
## 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
## 3193 3019 3627 3727 3951 4164 4335 5245 5283 5913 6659 6919 7019 7201
```

```
save(menTables, file = "CBMenTextTables.rda")
```

The data are successfully scraped from the urls that are given and we observe that the number of participants in the Cherry Blossom race has increased steadily from 1999 to 2012. We can use the `extractResTable` function to extract all of the results, or to specify a single set of results.

```
men_2009<-extractResTable(url = urls[11], year = 2009, sex = 'male')
men_2009[1:10]
```

```
## [1] " "
## [2] "          Credit Union Cherry Blossom Ten Mile Run"
## [3] "          Washington, DC          Sunday, April 5, 2009"
## [4] " "
## [5] "          Official Male Results"
## [6] " "
## [7] "Place Div  /Tot  Num  Name                      Ag Hometown          Gun Tim Net Tim  Pace
## [8] "===== ===== = =====
## [9] "      1      1/1420      1 Ridouane Harroufi      27 Morocco          45:56  45:56#  4:36
## [10] "      2      1/62       9 Feyisa Liesa        19 Ethiopia          45:58  45:58#  4:36
```

We observe the formatting of the text data, such as the use of the = character can be used to transform the lines of text into a a matrix. We will use this pattern to identify the boundary between the headering and the body of the data.

```
equals_idx<- grep('^===', men_2009)
equals_idx
```

```
## [1] 8
```

```
equals_idx2<- substr(men_2009, 1,3)
which(equals_idx2 == '===')
```

```
## [1] 8
```

```
spacerRow<- men_2009[equals_idx]
headerRow<- men_2009[equals_idx -1]
body<- men_2009[-(1:equals_idx)] # working with 2009 body
```

```
head(body)
```

```
## [1] "      1      1/1420      1 Ridouane Harroufi      27 Morocco      45:56      45:56#      4:36"
## [2] "      2      1/62      9 Feyisa Liesa      19 Ethiopia      45:58      45:58#      4:36"
## [3] "      3      1/1271     13 Silas Sang      31 Kenya      45:59      45:59#      4:36"
## [4] "      4      2/1420     29 Karim El Mabchour      26 Morocco      46:00      46:00#      4:36"
## [5] "      5      3/1420     81 Stephen Chemlany      26 Kenya      46:06      46:06#      4:37"
## [6] "      6      1/399     83 Cosmas Koech Kimuati      23 Kenya      46:08      46:08#      4:37"
```

We can use the text above the = boundary as a header row to identify the columns names.

```
headerRow<- tolower(headerRow)
headerRow
```

```
## [1] "place div /tot num name ag hometown gun tim net tim pace"
```

Now that we have defined the column names, we are free to use the text structure, such as the spacing of column names and data, to identify datapoints. We will start with `ag`.

```
# looks for first match of given sequence
ageStart<- regexec('ag', headerRow, fixed = TRUE)
ageStart
```

```
## [[1]]
## [1] 49
## attr(,"match.length")
## [1] 2
## attr(,"useBytes")
## [1] TRUE
```

```
" 1 1/1420 1 Ridouane Harroufi 27 Morocco 45:56 45:56# 4:36 "
" 27 " " " " " 26 M" " 26 " " 23 Ken"
```

```
age<- substr(body, start = 49, stop = 50)
head(age)
```

```
## [1] "27" "19" "31" "26" "26" "23"
```

```
summary(as.numeric(age))
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   NA's
##    10.00   29.00   35.00   37.37   44.00   85.00     2
```

We have successfully used the structure of the text data to identify an index location for `ag` data points. We can continue to use the spacing of text data to identify other data points.

```
#gregexpr looks for all matches of a given sequence
```

```
blankLocs<- gregexpr(' ', spacerRow)
blankLocs
```

```
## [[1]]
## [1]  6 18 25 48 51 72 80 89 95
## attr(,"match.length")
## [1]  1  1  1  1  1  1  1  1  1
```

```
searchLocs<- c(0,blankLocs[[1]])
searchLocs
```

```
## [1]  0  6 18 25 48 51 72 80 89 95
```

The spacing indices will now be used in a function that can compile a matrix of data points across the entire list of tables.

```
values<- mapply(substr, list(body), start = searchLocs[-length(searchLocs)] +1, stop = searchLocs[-1] -
length(values))
```

```
## [1] 59859
```

```
head(values)
```

```
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] "  1" "  1/1420" "  1" "Ridouane Harroufi" "27"
## [2,] "  2" "  1/62" "  9" "Feyisa Liesa" "19"
## [3,] "  3" "  1/1271" " 13" "Silas Sang" "31"
## [4,] "  4" "  2/1420" " 29" "Karim El Mabchour" "26"
## [5,] "  5" "  3/1420" " 81" "Stephen Chemlany" "26"
## [6,] "  6" "  1/399" " 83" "Cosmas Koech Kimuati" "23"
##      [,6]      [,7]      [,8]      [,9]
## [1,] "Morocco" " " 45:56" " 45:56#" " 4:36"
## [2,] "Ethiopia" " " 45:58" " 45:58#" " 4:36"
## [3,] "Kenya" " " 45:59" " 45:59#" " 4:36"
## [4,] "Morocco" " " 46:00" " 46:00#" " 4:36"
## [5,] "Kenya" " " 46:06" " 46:06#" " 4:37"
## [6,] "Kenya" " " 46:08" " 46:08#" " 4:37"
```

A function `findColLocs` can be defined that uses these space indices to locate the begin points for data throughout the extracted text data.

```
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))
}
```

```
locs_2012<- findColLocs(spacerRow)
length(locs_2012)
```

```
## [1] 10
```

```
locs_2012
```

```
## [1] 0 6 18 25 48 51 72 80 89 95
```

selectCols is a tool that uses findColLocs to locate the data points for each column throughout the text for all of the race years.

```
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 )
}
```

We can examine the results for validity with the summary function. Our functions have extracted almost all of the age data and may have identified some errors within the dataset.

```
searchLocs = findColLocs(spacerRow)
ageLoc = selectCols("ag", headerRow, searchLocs)
ages = mapply(substr, list(body),
```

```

start = ageLoc[1,], stop = ageLoc[2, ])
summary(as.numeric(ages))

```

```

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   NA's
##    10.00   29.00   35.00   37.37   44.00   85.00     2

```

We can test the functionality of the tools we have assembled so far.

```

shortColNames = c("name", "home", "ag", "gun", "net", "time")
locCols = selectCols(shortColNames, headerRow, searchLocs)

```

*#mapply forms a matrix after completion*

```

Values = mapply(substr, list(body), start = locCols[1, ],
                stop = locCols[2, ])

```

```

class(Values)

```

```

## [1] "matrix" "array"

```

```

colnames(Values) = shortColNames
head(Values)

```

```

##      name                home      ag  gun      net
## [1,] "Ridouane Harroufi" "Morocco" "27" "45:56" "45:56#"
## [2,] "Feyisa Liesa"      "Ethiopia" "19" "45:58" "45:58#"
## [3,] "Silas Sang"        "Kenya"    "31" "45:59" "45:59#"
## [4,] "Karim El Mabchour" "Morocco" "26" "46:00" "46:00#"
## [5,] "Stephen Chemlany"  "Kenya"    "26" "46:06" "46:06#"
## [6,] "Cosmas Koech Kimuati" "Kenya"    "23" "46:08" "46:08#"
##      time
## [1,] NA
## [2,] NA
## [3,] NA
## [4,] NA
## [5,] NA
## [6,] NA

```

```

tail(Values)[ , 1:3]

```

```

##      name                home      ag
## [6646,] "John Sim"         "Fairfax VA"    "34"
## [6647,] "Kerry Kilman"     "Rockville MD" "61"

```



```
## [6648,] "Edward Jefferson      " "Norfolk VA          " "75"
## [6649,] "Robert Smith          " "Comus MD             " "75"
## [6650,] "ine                  " "                     " "
## [6651,] "uideline"            " "                     " "
```

To generalize this method, we define a function that combines the various processes we have used so far in the `extractVariables` function.

```
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)
}
```

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

We now have a list of matrices. The original text data are in a much more machine readable format. Our number of rows for each matrix is equal to the number of original rows, minus the original text header.

```
menResMat = lapply(menTables, extractVariables)
length(menResMat)
```

```
## [1] 14
```

```
sapply(menResMat, nrow)
```

```
## 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012  
## 3190 3017 3622 3724 3948 4156 4327 5237 5276 5905 6651 6911 7011 7193
```

To perform statistical analyses on the numeric data, we must perform a datatype transformation.

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

```
## [1] 41 39 56 35 NA 48
```

```
age = sapply(menResMat,  
             function(x) as.numeric(x[ , 'ag']))
```

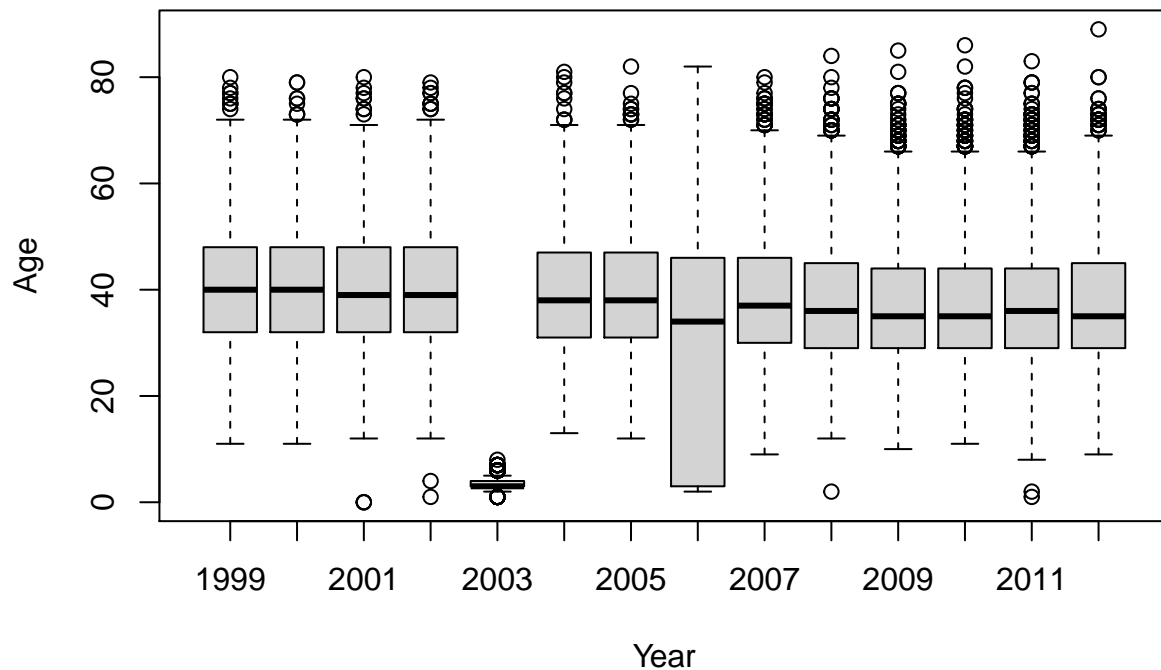
```
## Warning in FUN(X[[i]], ...): NAs introduced by coercion
```

```
## Warning in FUN(X[[i]], ...): NAs introduced by coercion
```

```
## Warning in FUN(X[[i]], ...): NAs introduced by coercion
```

There appears to be an anomaly in the process for the year 2003. After an examination of the original data table, we found that the spacing in the year 2003 data is not the same as the other years. We will adjust our `selectCols` function to capture the data for 2003.

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



```
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 )
}
```

We also observe anomalies in the formatting of the year 2001. We introduce fixes and scrape the original tables once more to form the list of matrices called `menResMat`.

```
age2001 = age[["2001"]]

grep("^===", menResMat['2001'])
```

```
## integer(0)
```

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

```

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(menTables, extractVariables)

```

```

age<- sapply(menResMat, function(x) as.numeric(x[, 'ag']))

```

```

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

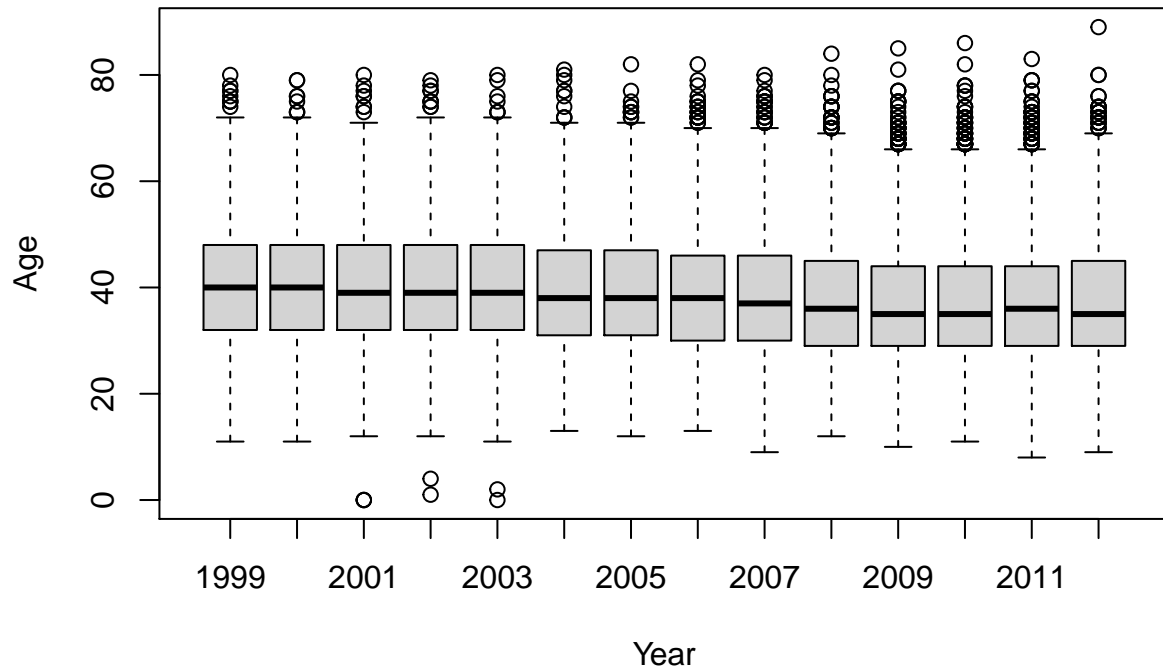
```

```

## 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
##    1    0    0    2    0    0   10    0    3    0    0    4    0    1

```

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



```
head(menResMat[['2001']])
```

##	name	home	ag	gun	net
## [1,]	"John KORIR	" "Kenya	" "26	" "46:12#"	" 46:12 "
## [2,]	"Reuben CHERUIYOT	" "Kenya	" "27	" "46:13#"	" 46:11 "
## [3,]	"Evans RUTTO	" "Kenya	" "23	" "46:26#"	" 46:26 "
## [4,]	"Enock MITEI	" "Kenya	" "21	" "46:39#"	" 46:37 "
## [5,]	"Felix LIMO	" "Kenya	" "21	" "46:41#"	" 46:40 "
## [6,]	"El Arbi KHATTABI	" "Morocco	" "34	" "46:49#"	" 46:49 "

##	time
## [1,]	NA
## [2,]	NA
## [3,]	NA
## [4,]	NA
## [5,]	NA
## [6,]	NA

```

young<- menResMat[['2001']][, 'ag'] < 10
menResMat[['2001']]<- menResMat[['2001']] [-c(young),]
y2002<- menResMat[['2002']][, 'ag'] < 10
y2003<- menResMat[['2003']][, 'ag'] < 10
menResMat[['2002']]<- menResMat[['2002']] [-c(y2002),]
menResMat[['2003']]<- menResMat[['2003']] [-c(y2003),]

```

```

y2002<- menResMat[['2002']][, 'ag'] < 10
y2003<- menResMat[['2003']][, 'ag'] < 10

```

```

menResDF = lapply(menResMat, data.frame)

```

```

col_1999 = rep(1999, nrow(menResDF$`1999`))

```

```

col_2000 = rep(2000, nrow(menResDF$`2000`))

```

```

col_2001 = rep(2001, nrow(menResDF$`2001`))

```

```

col_2002 = rep(2002, nrow(menResDF$`2002`))

```

```

col_2003 = rep(2003, nrow(menResDF$`2003`))

```

```

col_2004 = rep(2004, nrow(menResDF$`2004`))

```

```

col_2005 = rep(2005, nrow(menResDF$`2005`))

```

```

col_2006 = rep(2006, nrow(menResDF$`2006`))

```

```

col_2007 = rep(2007, nrow(menResDF$`2007`))

```

```

col_2008 = rep(2008, nrow(menResDF$`2008`))

```

```

col_2009 = rep(2009, nrow(menResDF$`2009`))

```

```

col_2010 = rep(2010, nrow(menResDF$`2010`))

```

```

col_2011 = rep(2011, nrow(menResDF$`2011`))

```

```

col_2012 = rep(2012, nrow(menResDF$`2012`))

```

```

cols = c(col_1999, col_2000, col_2001, col_2002, col_2003, col_2004, col_2005, col_2006, col_2007, col_2008, col_2009, col_2010, col_2011, col_2012)

```

```
menResFinal = data.frame()
for (i in 1:length(names(menResDF))) {

  df<- menResDF[[i]]
  menResFinal = rbind(menResFinal, df)
}
menResFinal$Date<- cols
```

```
menResFinal$ag = as.numeric(menResFinal$ag)
sum(is.na(menResFinal$ag))
```

```
## [1] 21
```

```
timePieces[1:5]
```

```
## [[1]]
## [1] 46 59
##
## [[2]]
## [1] 47 1
##
## [[3]]
## [1] 47 3
##
## [[4]]
## [1] 47 7
##
## [[5]]
## [1] 47 31
```

```
class(timePieces)
```

```
## [1] "list"
```

```
timePieces[1]
```

```
## [[1]]
## [1] 46 59
```

```
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)

```

```

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.     NA's
##  45.25   77.21   86.63   87.55   96.75  170.83   48610

```

```

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
  })
}

```

```

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

```

```

## 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
##    0    0    0    0    0    0    0 5232    0    0    0    0    0    0

```

```

separatorIdx = grep("^===", menTables[["2006"]])
separatorRow = menTables[["2006"]][separatorIdx]
separatorRowX = paste(substring(separatorRow, 1, 63), " ",
                      substring(separatorRow, 65, nchar(separatorRow)),
                      sep = "")
menTables[["2006"]][separatorIdx] = separatorRowX

menResMat = sapply(menTables, extractVariables)
menDF = mapapply(createDF, menResMat, year = 1999:2012,
                 sex = rep("M", 14), SIMPLIFY = FALSE)

```

```

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")

```



```
years = 1999:2012
urls = paste(ubase, womenURLs, sep = "")
urls[1:3]
```

```
## [1] "http://www.cherryblossom.org/results/1999/cb99f.html"
## [2] "http://www.cherryblossom.org/results/2000/Cb003f.htm"
## [3] "http://www.cherryblossom.org/results/2001/oof_f.html"
```

```
womenTables = mapply(extractResTable, url = urls, year = years, sex='female')
names(womenTables) = years
sapply(womenTables, length)
```

```
## 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
## 2359 2169 2976 3338 3547 3907 4342 5445 5699 6405 8333 8863 9038 9737
```

```
separatorIdx = grep("^===", womenTables[["2006"]])
separatorRow = womenTables[["2006"]][separatorIdx]
separatorRowX = paste(substring(separatorRow, 1, 63), " ",
                      substring(separatorRow, 65, nchar(separatorRow)),
                      sep = "")
womenTables[["2006"]][separatorIdx] = separatorRowX

womenTables[["2001"]][2:3] = womenTables[["2002"]][2:3]

womenResMat = sapply(womenTables, extractVariables)
womenDF = mapply(createdDF, womenResMat, year = 1999:2012,
                  sex = rep("W", 14), SIMPLIFY = FALSE)
```

```
## Warning in data.frame(year = rep(year, nrow(Res)), sex = rep(sex, nrow(Res)), :
## NAs introduced by coercion
```

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

```
## 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
##    4    0    0    4    0    0    8    1    2    0    2    0    0    0
```

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

```
## 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

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

```
## 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
##    1    0    0    2    0    0   10    0    3    0    0    4    0    1
```

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

```
## 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

```
menDF_complete = na.omit(menDF)
womenDF_complete = na.omit(womenDF)
```

```
cherryblossom = data.frame()
for (i in 1:14) {

  df<- menDF_complete[[i]]
  cherryblossom = rbind(cherryblossom, df)
}
head(cherryblossom)
```

```
##   year sex          name      home age  runTime
## 1 1999   M Worku Bikila    Ethiopia    28 46.98333
## 2 1999   M Lazarus Nyakeraka    Kenya    24 47.01667
## 3 1999   M James Kariuki    Kenya    27 47.05000
## 4 1999   M William Kiptum    Kenya    28 47.11667
## 5 1999   M Joseph Kimani    Kenya    26 47.51667
## 6 1999   M Josphat Machuka    Kenya    25 47.55000
```

```
for (i in 1:14) {

  df<- womenDF_complete[[i]]
  cherryblossom = rbind(cherryblossom, df)
}
tail(cherryblossom)
```

```
##      year sex          name      home age  runTime
## 146036 2012   W Effie Harary    Long Branch NJ    39 152.1333
## 146037 2012   W Khristina Nava    Fort Meade MD    40 153.1833
## 146038 2012   W Geneva Dixon    Manassas Park VA    31 156.0500
## 146039 2012   W Veronica Eligan    Mitchellville MD    55 156.7500
## 146040 2012   W Denise Bobba    Herndon VA    40 156.9000
## 146041 2012   W Rashonna Waples    District Heights MD    38 170.9667
```

```
sum(is.na(cherryblossom))
```

```
## [1] 42
```

```
cherryblossom = na.omit(cherryblossom)  
sum(is.na(cherryblossom))
```

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

### 3 Conclusion

#### A Code

The code is coooool