

Web Scraping NYC Marathon Results in R

Greg DeVore

March 17th, 2018

Note: The full R script can be found at <https://github.com/gregdevore/Projects/blob/master/MarathonScrapes.R>

The purpose of this write-up is to explore web scraping in R using the *rvest* and *Rselenium* packages. The objective is to compile the top 100 finishers (both men and women) in the NYC marathon since the year 2000 and study the results. Some questions that will be addressed:

- Have the overall finishing times changed by a significant amount over the years?
- Do the distributions of finishing times vary significantly from year to year?
- How do the performances of men and women compare in terms of variance within the top 100 finishers?

The topic itself is merely a byproduct of my interest in running, the main idea is to investigate web scraping with R. The website containing the results is <http://www.marathonguide.com/>, and the particular page of interest is for the NYC marathon:

Initial Page:

[Results Home](#)
New York City Marathon: [Information & Comments](#) | [Press Releases](#) | [News](#) |
New York City Marathon Results: [2017](#) | [2016](#) | [2015](#) | [2014](#) | [2013](#) | [2011](#) | [2010](#) | [2009](#) | [2008](#) |
[2007](#) | [2006](#) | [2005](#) | [2004](#) | [2003](#) | [2002](#) | [2001](#) | [2000](#) |

New York City Marathon - Race Results

New York City Marathon - Results
New York City, NY USA
November 5, 2017
Finishers: 50643, Males - 29583, Females - 21060
Male Winner: 2:10:53 | Female Winner: 2:26:53
Average Finish Time: 4:37:03 | STD: 0:56:32

Use the lists on the left to browse the results for all entries in this marathon. If you are searching for a specific runner in this marathon, enter their name (or the first few letters of their name) in the name box below. Or, enter a time to jump to that time in the overall results.


Browse The Results:
Overall
Overall Results ▾
Men
Men's Results ▾
Women
Women's Results ▾
View

Search The Results:
Search by Name:
First Name:
Last Name:
Max. Results per page: 10 ▾
Search by Time:
Time:

For any given year (2017 is currently shown), to view the top 100 finishers, you simply select the 'Men's Results' or 'Women's Results' dropdown and select '0 - 100', followed by 'View'. That brings up the results shown on the

following page. Obviously, this is very simple to do manually, but we'll need to do it using the R packages listed previously. That can get a little tricky, thankfully both packages are well documented and overall very easy to use.

Results Page:

Results Home New York City Marathon: Information & Comments Press Releases News New York City Marathon Results: 2017 2016 2015 2014 2013 2011 2010 2009 2008 2007 2006 2005 2004 2003 2002 2001 2000							
2017 New York City Marathon, Men's Results 1 - 100							
Have you run this race? Then tell us about it.				More Results: 			
Last Name, First Name (Sex/Age)	OverAll Place	Sex Place / Div Place	DIV	Net Time	City, State, Country	AG Time*	BQ*
Geoffrey Kamworor (M24)	1	1 / 1	M20-24	2:10:53	Kapchorwa District, KEN	2:10:53	BQ
Wilson Kipsang (M35)	2	2 / 1	M35-39	2:10:56	Iten, KEN	2:10:56	BQ
Lelisa Desisa (M27)	3	3 / 1	M25-29	2:11:32	Addis Ababa, ETH	2:11:32	BQ
Lemi Berhanu (M22)	4	4 / 2	M20-24	2:11:52	Addis Ababa, ETH	2:11:48	BQ
Tadesse Abraham (M35)	5	5 / 2	M35-39	2:12:01	Geneva, SUI	2:12:01	BQ
Michel Butter (M32)	6	6 / 1	M30-34	2:12:39	Beverwijk, NED	2:12:39	BQ
Abdi Abdirahman (M40)	7	7 / 1	M40-44	2:12:48	Tucson, AZ, USA	2:09:35	BQ
Koen Naert (M28)	8	8 / 2	M25-29	2:13:21	Oostkamp, BEL	2:13:21	BQ
Fikadu Girma Teferi (M24)	9	9 / 3	M20-24	2:13:58	New York, NY, ETH	2:13:58	BQ
Shadrack Biwott (M32)	10	10 / 2	M30-34	2:14:57	Folsom, CA, USA	2:14:57	BQ
Meb Keflezighi (M42)	11	11 / 2	M40-44	2:15:29	San Diego, CA, USA	2:10:03	BQ

To begin, we need to load the required packages. The 'rsDriver' command is from the *Rselenium* package and starts the selenium server and browser (default is Chrome). The 'remoteDriver' function is used to communicate with the server.

```
library(rvest)
library(RSelenium)
library(XML)
library(dplyr)
library(tidyr)
library(ggplot2)
# Start selenium server and browser
rsDriver()
remDr <- remoteDriver(remoteServerAddr = "localhost", port = 4445L, browserName = "chrome")
remDr$open()
```

Next, we define the base URL (for 2017 results), and create an html session using the *rvest* package.

```
# Start at base URL
url <- 'http://www.marathonguide.com/results/browse.cfm?MIDD=472171105'
p <- html_session(url)
```

Before we start collecting results, we need to initialize a collector to store the results from each year. I chose a list, since the number of columns in the results table may differ from year to year, along with the titles for each column. Adding the tables directly into a data frame requires a little more attention, so we'll do that in a separate step.

```
# Store yearly results in list (may have different number of columns depending on the year)
overall_results <- list()
i <- 1
gender <- 'Men'
years <- c(seq(2000,2011),seq(2013,2017))
```

Note that we also need to specify whether we are scraping results for men or women, and specify the years we will be pulling results for (the race was not run in 2012 due to Hurricane Sandy). The following code scrapes finishing times for a single gender, to obtain times for the other gender simply change the variable name from 'Men' to 'Women'.

Now comes the bulk of the program, where we navigate to each year and pull the results. The code is run inside of a loop (over each year), and we'll focus on a few key sections.

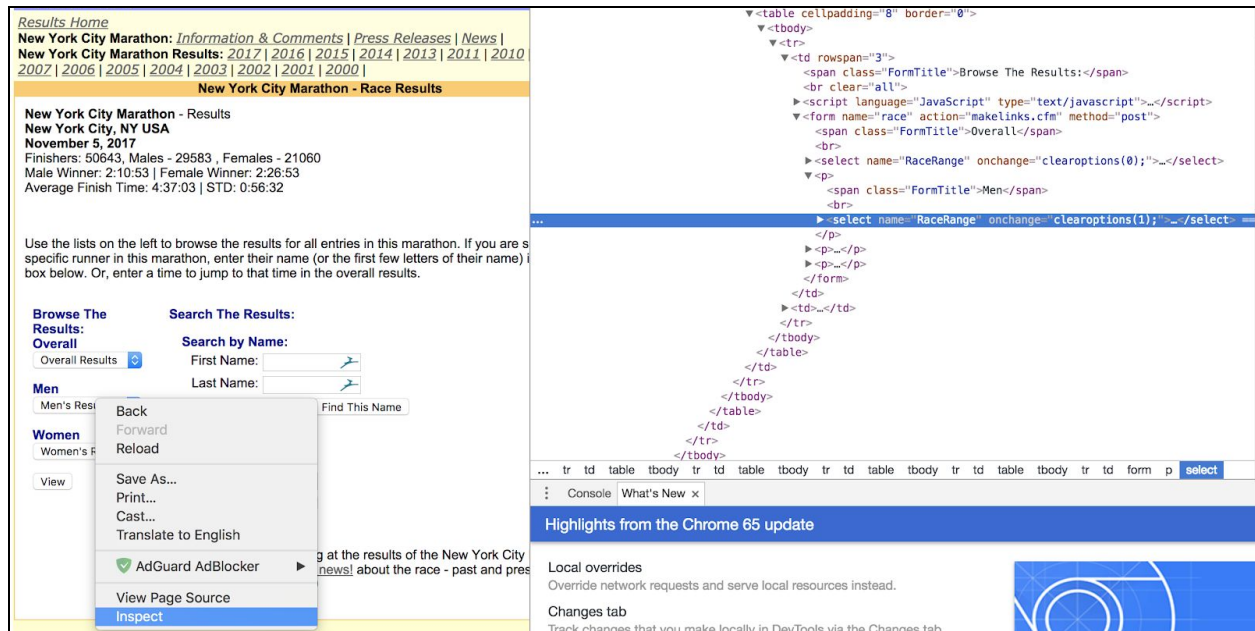
```
pi <- follow_link(p,as.character(year))
# Navigate to page
remDr$navigate(pi$url)
```

Here, year is one of the years from our list (2000, 2001, 2002, etc.), and we use *rvest* to follow the page link to that year (the links are simply the year, see the initial or results page pictures above). Once we have the url for the new page, we point our *remoteDriver* object to the same page.

```
# Select proper results dropdown and top 100 finishers
if (gender == 'Men') {
  css <- 'select+ p select'
} else {
  css <- 'p+ p select'
}
raceRange <- remDr$findElement(using = 'css selector', css)
raceRange$clickElement()
```

Here, we select the relevant results dropdown (Men's or Women's) by using the proper css selector. The selector was obtained using the 'SelectorGadget' extension for Chrome (<http://selectorgadget.com/>). The *findElement* function from *RSelenium* finds a specific part of the webpage for us, and then clicks on it (in this case, the action causes the dropdown menu to expand as if it was clicked).

Once we've expanded the dropdown, we need to select the relevant results (in this case, '0 - 100'). To do this, we'll use the xpath, rather than the css selector. In order to obtain the corresponding xpath, we need to right-click on the web page and select 'Inspect', which will bring up the HTML source code. To locate the relevant part of the source code, we simply right-click on the dropdown of interest and select 'Inspect' (see the picture on the following page). The final step is to right-click on the line in the HTML source code and select 'Copy → Copy XPath'.



```

if (gender == 'Men') {
  xpath <-
'html/body/table[2]/tbody/tr[1]/td[2]/table[3]/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/tr/td/table[2]/tbody/tr/td[1]/form/p[1]/select/option[2]'
} else {
  xpath <-
'html/body/table[2]/tbody/tr[1]/td[2]/table[3]/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/tr/td/table[2]/tbody/tr/td[1]/form/p[2]/select/option[2]'
}
raceRangeValue <- remDr$findElement(using = 'xpath', xpath)
raceRangeValue$clickElement()

```

Here, we set the correct xpath depending on whether men's or women's results are being scraped, and then click the correct option ('0 - 100') to set the value in the dropdown.

```

# Click 'View' to see race results
viewButton <- remDr$findElement(using = 'xpath',
'html/body/table[2]/tbody/tr[1]/td[2]/table[3]/tbody/tr[2]/td/table/tbody/tr/td/table/tbody/tr/td/table[2]/tbody/tr/td[1]/form/p[3]/input[3]')
viewButton$clickElement()

```

Here, we select 'View' to bring up the table of finishing times, once again using the corresponding xpath. Finally, we're ready to extract some results!

```

# Grab HTML table and parse into data frame
table <- remDr$findElement(using = 'xpath', 'html/body/table[2]/tbody/tr[1]/td[2]/table[3]/tbody/tr[2]/td/table')
elemtxt <- table$getAttribute('outerHTML')
elemxml <- htmlTreeParse(elemtxt[[1]], useInternalNodes=T)
results <- readHTMLTable(elemxml)[[1]]

```

Here, we find the results table, extract the HTML source code representation, parse it into XML, and convert the result to a data frame.

```
# Extract row 5 (header data)
col.names <- unlist(results[5,])
names(results) <- col.names
# Remove rows 1 - 5 (not relevant)
results <- results[-c(seq(1,5)), ]
# Store in list
overall_results[[i]] <- results
i <- i + 1
```

The column names of interest are actually in the 5th row, so we extract those and use them to rename the data frame columns. There are some extra rows that aren't relevant to us, so we remove those before storing the results in our list.

Once we've repeated the above set of actions for each year, we'll have all of the results compiled into a list of data frames. Now we're done with the main loop.

```
# Parse results, add to single data frame with consistent column names
results.df <- data.frame(Name = c())
for (ind in seq(1,length(overall_results))) {
  df <- overall_results[[ind]]
  names <- grep('name',names(df),ignore.case = TRUE)
  time <- grep('time',names(df),ignore.case = TRUE)
  country <- grep('country',names(df),ignore.case = TRUE)
  colnames(df)[c(names,time[1],country)] <- c('Name','Time','Country')
  df[c(names,time[1],country)] <- lapply(df[c(names,time[1],country)], as.character)
  df$Year <- years[ind]
  results.df <- rbind(results.df, df[, c(names,time[1],country,ncol(df))])
}
```

Here, we initialize a collector data frame and loop over each data frame in our list to find the relevant columns to store. Since the column names aren't guaranteed to be the same for each year, we use the *grep* function to find columns that contain the words 'name', 'time', and 'country' to extract the runner's name, finishing time, and place of origin, respectively. Once we have the indices of the columns, we add them to our collector data frame, and add the year so we can use this for parsing later on.

```
# Split time into hour/minute/second, convert to hours
results.df <- results.df %>% separate(Time, c('Hour','Minute','Second'),':')
results.df[, c('Hour','Minute','Second')] <- lapply(results.df[, c('Hour','Minute','Second')],as.numeric)
results.df <- results.df %>% mutate(TotalTime = Hour + Minute/60 + Second/3600)
```

This section uses the *separate* function from the *tidyr* package to convert the time from the standard form listed in the results table (something like '2:10:53') into separate columns for the hour, minute, and seconds. We convert them to numeric values and use the *mutate* function from the *dplyr* package to create a feature for the total time (in hours).

Now that we have all the data we need, it's time to start computing statistics of interest and generating some plots so we can see the trends from all the data we aggregated.

```
# Compute time stats for each year
results.timeSummary <- results.df %>% group_by(Year) %>%
  summarize(bestTime = min(TotalTime), worstTime = max(TotalTime), meanTime = mean(TotalTime),
    sd = sd(TotalTime))
```

The first area of interest is the range of finishing times over the years, specifically the fastest, slowest, average, and spread of finishing times (in terms of standard deviation). These can be computed using the *summarize* function within the *dplyr* package. The result is a data frame, which is shown below (for both men and women).

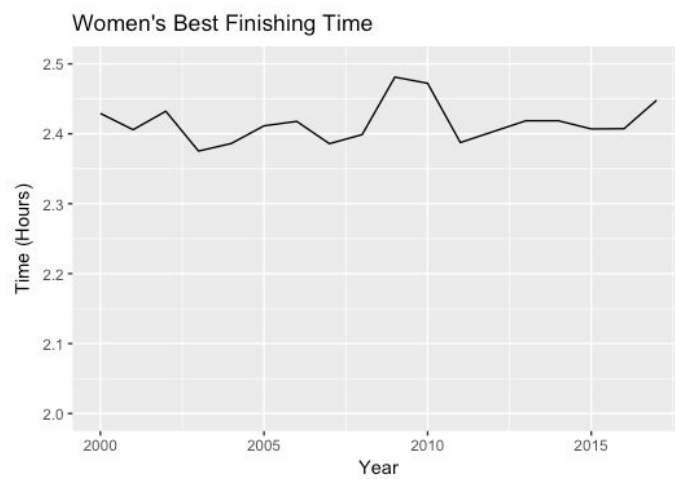
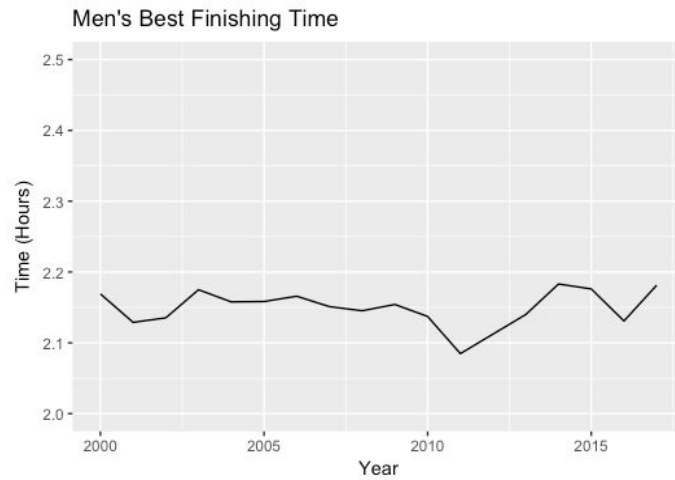
Summary of the Top 100 Finishing Times (Men and Women)

Year	Best (M)	Worst (M)	Average (M)	SD (M)	Best (F)	Worst (F)	Average (F)	SD (F)
2000	2.17	2.65	2.50	0.12	2.43	3.29	3.00	0.25
2001	2.13	2.63	2.45	0.13	2.41	3.15	2.89	0.21
2002	2.14	2.69	2.50	0.16	2.43	3.22	2.97	0.23
2003	2.18	2.73	2.51	0.16	2.38	3.27	2.99	0.26
2004	2.16	2.69	2.51	0.15	2.39	3.25	2.99	0.25
2005	2.16	2.69	2.51	0.16	2.41	3.21	2.96	0.25
2006	2.17	2.64	2.46	0.13	2.42	3.17	2.95	0.21
2007	2.15	2.67	2.54	0.14	2.39	3.14	2.94	0.18
2008	2.15	2.64	2.49	0.13	2.40	3.12	2.89	0.19
2009	2.15	2.61	2.45	0.12	2.48	3.16	2.96	0.18
2010	2.14	2.65	2.48	0.13	2.47	3.12	2.87	0.22
2011	2.08	2.62	2.48	0.14	2.39	3.09	2.89	0.21
2013	2.14	2.70	2.52	0.16	2.42	3.17	2.94	0.22
2014	2.18	2.66	2.51	0.14	2.42	3.12	2.93	0.20
2015	2.18	2.69	2.54	0.14	2.41	3.13	2.93	0.20
2016	2.13	2.68	2.51	0.15	2.41	3.12	2.92	0.17
2017	2.18	2.64	2.51	0.13	2.45	3.08	2.87	0.19

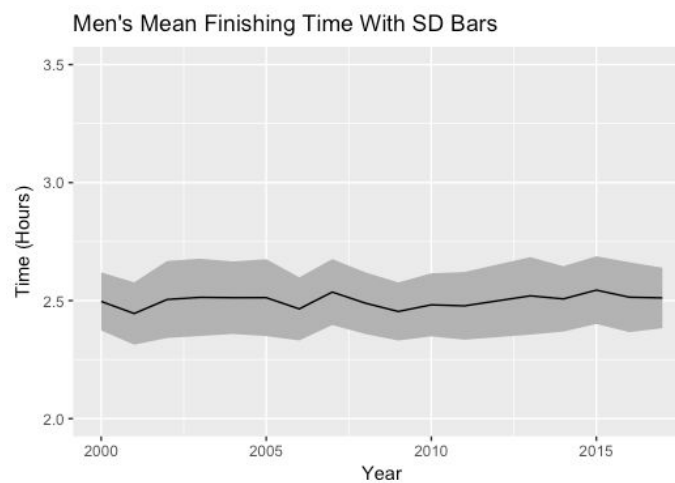
Some observations about the data:

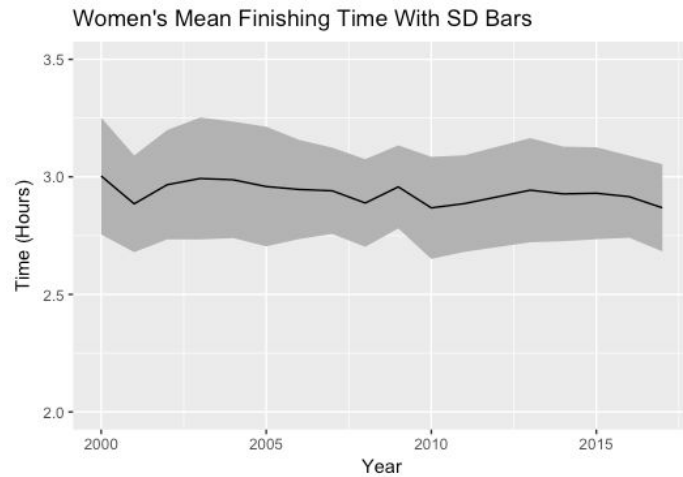
- The fastest times for men and women have not significantly improved since the year 2000.
- The average men's finishing time is consistently around 2.5 hours, and the average women's finishing time is consistently around 3 hours.
- The standard deviation of men's times is smaller than that of the women's times.
- The range of men's finishing times is smaller (typically ~0.6 hours) than women's (typically ~0.8 hours). In other words, there is more spread in the women's finishing times. This helps explain the observation about the standard deviation above.
- Although the fastest times haven't improved, the rest of the women's times have improved slightly over the last decade.

Often plots tell a better story than tables like this, so let's look at some of the data in a different form. The best times for men and women are shown below:

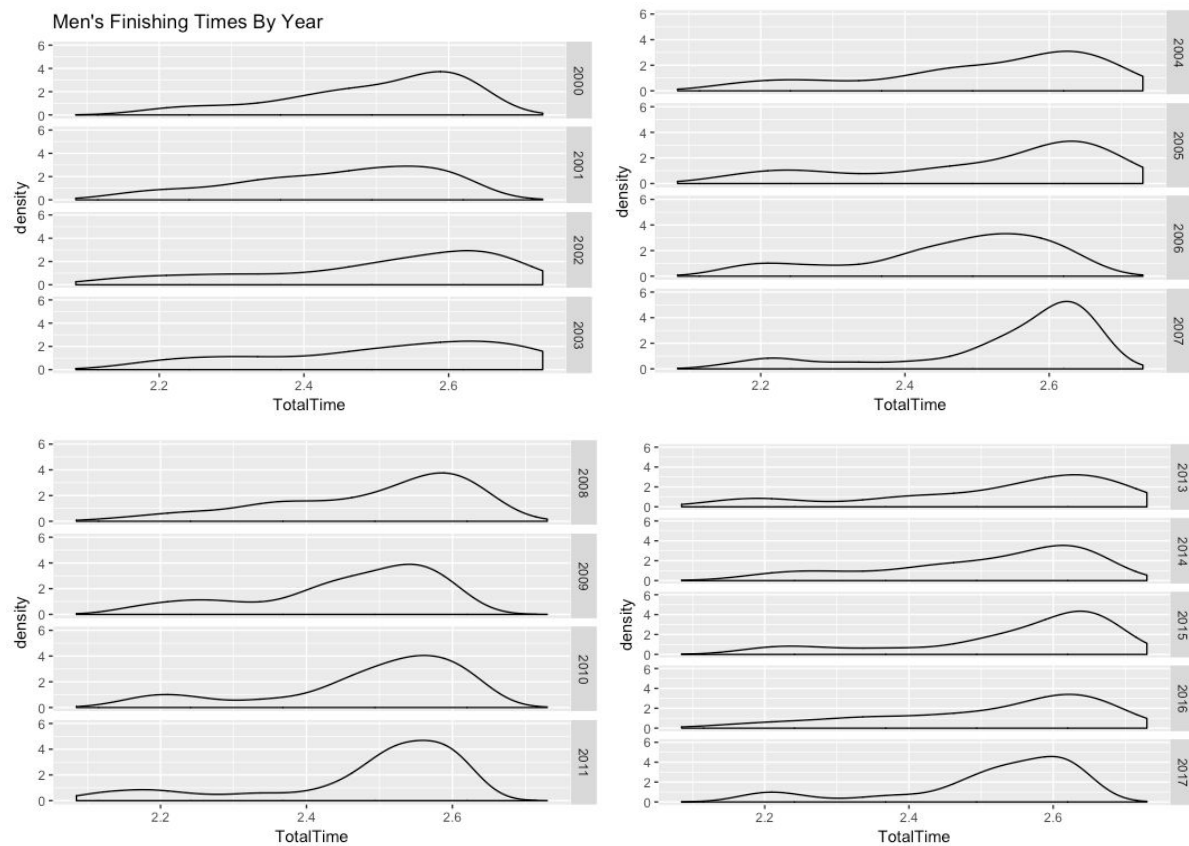


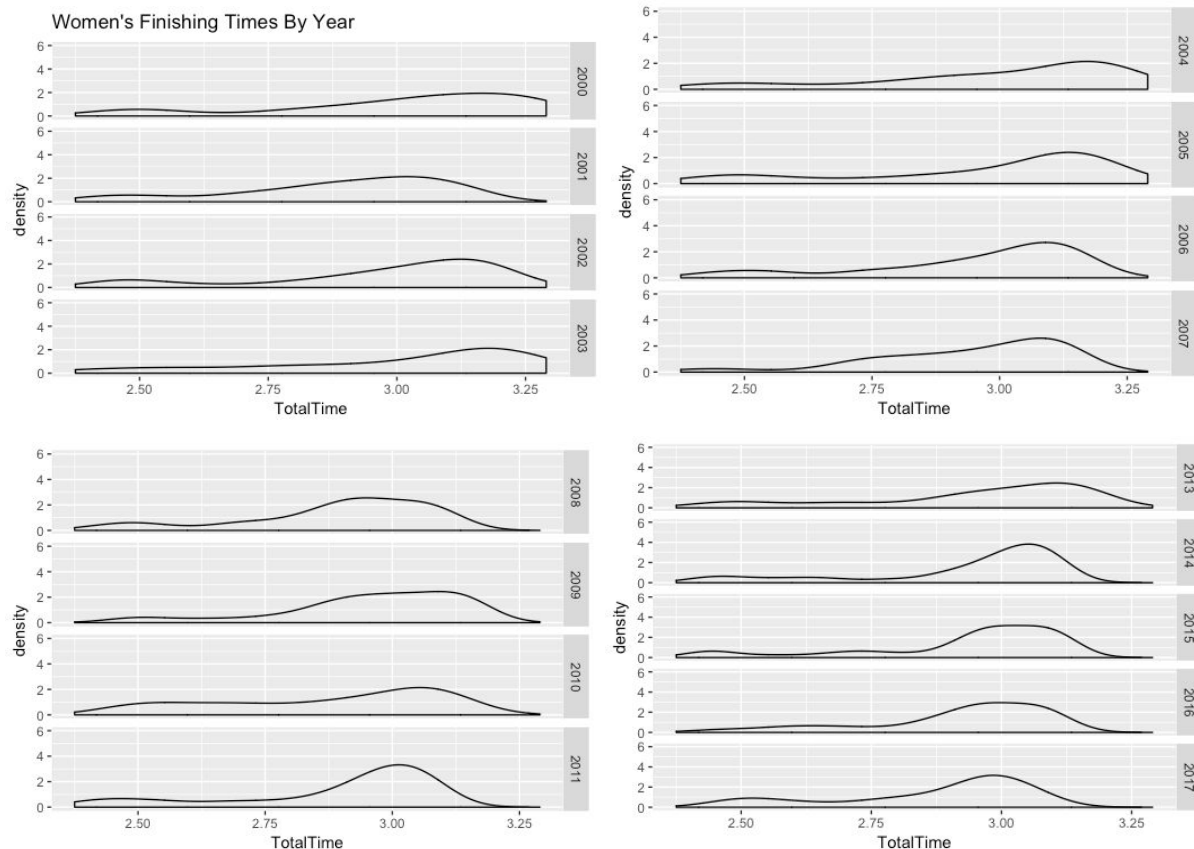
As seen in the table, the winning times for men and women are relatively consistent from year to year, with no real trend up or down. Let's look at the average finishing times with standard deviation bands:





As noted earlier, women's average finishing times have a larger spread compared to men's. These graphs do a good job of illustrating the difference. The last set of plots shows the distribution of finishing times year over year, something we can't get from a simple table.





These plots show an interesting trend that wasn't clear from the table data. The distribution for most years is left skewed, with a small number of the top 100 finishers up front, and the bulk of them near the back. This trend has only gotten more serious over time, especially for women. Note that in the early 2000's, the finishing times were more evenly distributed for men and women, but in the last 10 years, the distributions have become more skewed. Even though the best times have not improved significantly year over year, the distribution of finishers has shifted so that the bulk of the top 100 finish in a tighter pack.

Also, it was noted earlier that women's times have been getting slightly faster over the last few years. This is clearly shown in the plots above, note that the peak in the distribution of times is moving to the left from 2013 to 2017.

Overall, this write-up was meant to serve as an introduction to web scraping with R, in particular using the *rvest* and *selenium* packages. They both make what could be a very cumbersome task (parsing html and xml, navigating between pages) relatively painless. As a bonus, along the way we learned a few things about the trends among the top finishers in the NYC marathon.