Authors: Justin Beer, Anthony Dalton, Harsha Kommanapalli

Date: March 4, 2018

Cherry Blossom Web-Scrape

And Male Runners Age Analysis

## Abstract

In this case study, we attempt to web-scrape data from <http://www.cherryblossom.org/>. Cherry Blossom is an annual race held in Washington, D.C. The website posts the run times for each participant every year. We will take this data from their webpages and transform it into useful data for analysis. Once we have transformed the data to compare the age distributions of runners in each race and to see how the ages changed from 1999 to 2012.

Through our analysis of the male runners that participated in the Cherry Blossom race, we will find that as the year’s progress from 1999 to 2012 the mean age of the runners’ decreases from approximately 40 to 35 years of age. This could be due to an increase in participation by younger runners while the amount of older runners is unchanged.

## Introduction

Data, data, everywhere. On the internet anyone can post anything that they like, most of the time without repercussion. Have you participated in a public event in the past? Well the event coordinator may have posted statistics about your participation on the internet. This is certainly the case for the coordinators of the Cherry Blossom race in Washington, D.C. The race is held annually and is well known as a practice race for the Boston Marathon. The race coordinators have collected individual statistics, such as run time, age, country, and gender, on every runner since 1999.

The website contains two webpages for every race. One page holds the data for the males and the other for female runners. Now that the data is out there on the internet, it is publically available. The availability is there simply because anyone can visit the site and look at the information or even, as in our case, download the data on the runners and use it for analysis. We will seek to compare the age distributions of the male runners for the Cherry Blossom races held between 1999 and 2012.

## Literature Review

The text we are utilizing, “Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving” (Nolan, Lang 2015), teaches us how to access, download, import, munge, and analyze the male runners results for the annual Cherry Blossom race. The software used is R and the methods are standard for R coding. The implementation of functions for web scraping and data munging are covered in the text and modified by us in order to overcome changes in the websites coding.

## Methods

Since Cherry Blossom organizers have posted the runners results on their webpage we must figure out a way to easily access their page, parse through the code, and extract the data that we want. Thankfully, we can utilize the R package “XML” (Lang 2018) in order to easily access the webpage, identify the node that the data is contained within, and extract the text held within that node. Unfortunately, differences in coding for each years webpages holding the results differs. In order to overcome this, we utilize our extractResTable2 function in order to parse each year differently.

extractResTable2 = 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") {  
 # This was a custom add over what the textbook provided  
 # Get preformatted text from <div class="Section1"> element  
 # Each line of results is in a <pre> element  
 # We then extract the text from the elements and strip it of the character [Â] and remove title headers  
 ff = getNodeSet(doc, "//div[@class='Section1']")  
 ff1 = getNodeSet(ff[[1]], "//pre")  
 txt = sapply(ff1, xmlValue)  
 els = str\_replace\_all(txt, "[Â]", " ")  
 }  
 else if (year == 2009 & sex == "female") {  
 # This was a custom add over what the textbook provided  
 ff = getNodeSet(doc, "//pre")  
 txt = sapply(ff, xmlValue)  
 els = strsplit(txt, "\r\n")[[1]]  
 }  
 else if (year == 1999 & sex == "male") {  
 # This was a custom add over what the textbook provided  
 # Find the <pre> elements  
 # We then extract the text from the elements and split it by \n  
 ff = getNodeSet(doc, "//pre")  
 txt = sapply(ff, xmlValue)  
 els = strsplit(txt, "\n")[[1]]  
 }  
 else if (year == 1999 & sex == "female") {  
 # This was a custom add over what the textbook provided  
 # Find the <pre> elements  
 # We then extract the text from the elements and split it by \n  
 ff = getNodeSet(doc, "//pre")  
 txt = sapply(ff, xmlValue)  
 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)  
}

Now that we have a function to be able to pull out the information we want and write it to an R list. As of now we have a list of lists and each list contains a blob of text. We need to be able to actually transform and get at the data within the list that we want to analyze. In order to do so we create our extractVariable function which will parse through each lists blob text and take out the variables based upon the formatting of the text.

Now that we have a function for extracting the text from the website and a function for extracting the variables from the text, we can create a useful data frame from this text. To do so we simply apply our two functions:

# Create a menTables list by extracting the text from each male results webpage from 1999 to 2012.  
menTables = mapply(extractResTable2, url = urls, year = years, sex = "male")  
# Name each list within the menTables List with the years the data corresponds to.  
names(menTables) = years  
# Extract the different columns we need from the text in each list.  
menResMat = lapply(menTables, extractVariables)  
# Create the age column.  
age = sapply(menResMat, function(x) as.numeric(x[ , "ag"]))

Up to this point, we have extracted the webpage text and put each year into its own list within a larger list object, 14 total one for each year from 1999 to 2012. We have also looked into each list entries text and pulled out the variables we needed and created a results matrix for each year within the menResMat object.

We still have some residual issues with the age variable for each matrix that we need to resolve. We need to locate the NA’s and fix them. To do so we employ the following code:

# We will determine how many years have N/A's.  
sapply(age, function(x) sum(is.na(x)))  
  
# Here we troubleshoot the N/A Values starting with the year 2001.  
age2001 = age[["2001"]]  
grep("^===", menTables[['2001']])  
badAgeIndex = which(is.na(age2001)) + 5  
menTables[['2001']][ badAgeIndex ]  
badAgeIndex  
  
# Lets locate all rows that are entirely blank.  
blanks = grep("^[[:blank:]]\*$", menTables[['2001']])  
which(age2001 < 5)  
  
# We explore the data to figure out who is less than 5 years old and determine three runners are set to age 0  
menTables[['2001']][ which(age2001 < 5) + 5 ]

We have several years that have erroneous data. Some years have entire rows that are blank, others have runners ages listed as 5 years old and 0 years old. In order to fix all of these problems we employ the following code and functions. After running this code we are left with a usable data frame, cbMen, which we can start to perform our analysis on.

# We need to massage the time into the correct format.  
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  
 })  
}  
# Function for creating our data frame.  
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)  
}  
# Apply the data frame creation function to our menResMat matrix  
menDF = mapply(createDF, menResMat, year = 1999:2012,  
 sex = rep("M", 14), SIMPLIFY = FALSE)  
# Apply our time function to menDF  
sapply(menDF, function(x) sum(is.na(x$runTime)))  
# Bind it all together in a new data frame  
cbMen = do.call(rbind, menDF)

Now that we have accessed the Cherry Blossom race results for males from the years 1999 to 2012, extracted the wanted data, munged and cleaned our data, and finally have a useful data frame, we can start our analysis of the age distributions of the male runners over the years.

## Results

For our case study we are going to answer the question Q.10 from Chapter 2 of our textbook on page 101 (Nolan, 2015). The textbook states “We have seen that the 1999 runners were typically older than the 2012 runners. Compare the age distribution of the runners across all 14 years of the races” (Nolan, 2015). In order to compare the age distribution between the runners for these years we will employ a box plot of the age versus year.

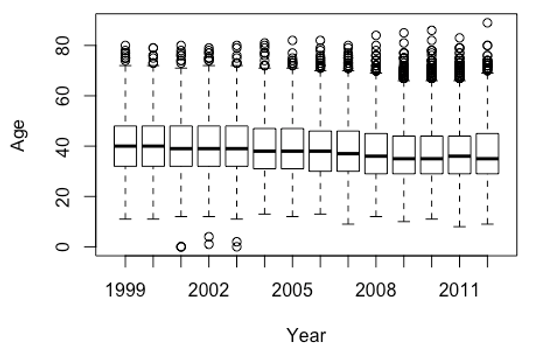


Figure 1. Boxplots of Age Distribution for race years 1999 to 2012.

We can see in Figure 1 above, that the mean age starts in the year 1999 around 40 years old and slowly decreases to around 35 in the year 2012. We can also look at the density plot distribution of the age for each year.

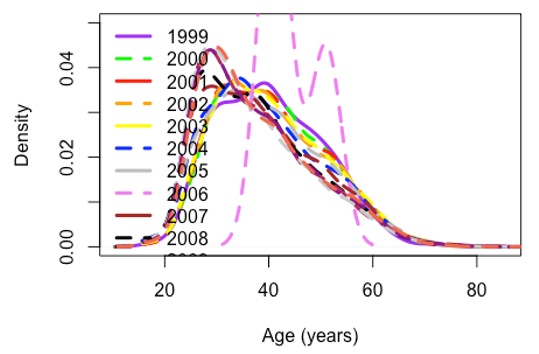


Figure 2. Density Curves for each year of the Cherry Blossom race, 1999 to 2012.

The above figure 2 shows us the same trend that we saw looking at the boxplots and the mean age of the male runners. The density plots show us that there is a grouping of years where the age is similar. The density plots are most similar for the years 1999, 2000, 2001, 2003, 2004, 2007, 2009, and 2011. As the years progress we have an interesting phenomenon where the years 2002, 2005, 2006, 2008, and 2010 all become right-skewed with a lower mean age than the other years. Then we have the year 2012 that is a very tall and narrow bi-modal distribution with a larger mean age then all of the others.

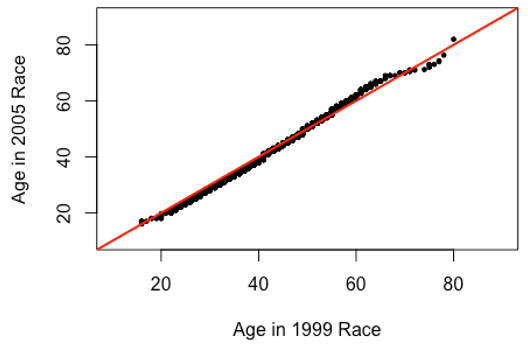


Figure 3. Quantile-Quantile Plot of Age for 1999 & 2005.

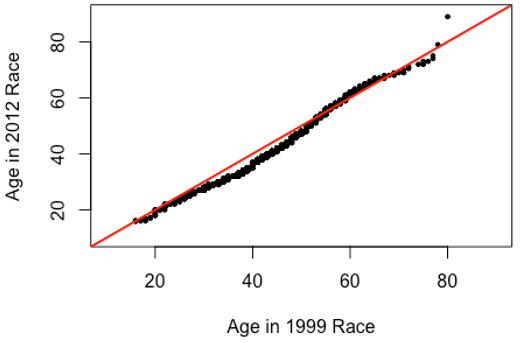


Figure 4. Quantile-Quantile Plot of Age for 1999 & 2012.

The above Figures 3 and 4 are quantile-quantile plots of the years 1999 & 2005 and 1999 & 2012 respectively. We see from these plots that the age distribution of the male runners in the 1999 and 2005 race are very similar with only a small deviation starting for those older than the age of 60. However, when we compare the races in 1999 and 2012 we see that there is a more significant deviation than the previous comparison where the age distribution begins to deviate approximately at the age of 30 and then continues.

## Conclusion

In conclusion, we find that as the years progressed the mean age of the male runners in the Cherry Blossom race decreased. There was one exception for the year 2012 where the mean age was higher than most years. The box plots shows a gradual decreasing of runner age as the years progress along with more outliers to the higher end of the age range than previous years. A possible explanation could be that as the years progressed more young runners joined and participated in the Cherry Blossom race while the same number of older runners participated. This would skew the age downwards as we have seen. Also, the density plots show us the same conclusions aside from the outlier year 2012 where we had a very different distribution. The quantile plots of the year’s show us that the age distributions between the race years of 1999, 2005, and 2012 were very different with younger runners participating in the more recent years.

## Appendix

### Reference

1. Nolan, D., & Lang, D. T. (2015). Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving. (Data science in R.) London: CRC Press.
2. Lang, Duncan Temple. “Package XML.” CRAN, 19 Feb. 2018, cran.r-project.org/web/packages/XML/index.html.

### Complete Code Base

#########################################################  
# Unit 8 Case Study  
#########################################################  
# To web scrape the data you only need to run the code from lines 6 to 89.  
# This is to clean up your memory  
#########################################################  
# Q.10 We have seen that the 1999 runners were typically older than the 2012 runners. Compare the age distribution of the runners across all 14 years   
# of the races. Use quantile–quantile plots, boxplots, and density curves to make your comparisons. How do the distributions change over the years?  
# Was it a gradual change?  
rm(list = ls())  
# Garbage collect memory  
gc()  
par(mfrow=c(1,1))  
# Set the working directory  
setwd("~/Desktop")  
library(XML)  
library(stringr)  
urlbase = "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(urlbase, menURLs, sep = "")  
years = 1999:2012  
# We are able to gather the data for every year except 1999 and 2009  
# Create new function to handle the years 1999 and 2000  
extractResTable2 = 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") {  
 # This was a custom add over what the textbook provided  
 # Get preformatted text from <div class="Section1"> element  
 # Each line of results is in a <pre> element  
 # We then extract the text from the elements and strip it of the character [Â] and remove title headers  
 ff = getNodeSet(doc, "//div[@class='Section1']")  
 ff1 = getNodeSet(ff[[1]], "//pre")  
 txt = sapply(ff1, xmlValue)  
 els = str\_replace\_all(txt, "[Â]", " ")  
 }  
 else if (year == 2009 & sex == "female") {  
 # This was a custom add over what the textbook provided  
 ff = getNodeSet(doc, "//pre")  
 txt = sapply(ff, xmlValue)  
 els = strsplit(txt, "\r\n")[[1]]  
 }  
 else if (year == 1999 & sex == "male") {  
 # This was a custom add over what the textbook provided  
 # Find the <pre> elements  
 # We then extract the text from the elements and split it by \n  
 ff = getNodeSet(doc, "//pre")  
 txt = sapply(ff, xmlValue)  
 els = strsplit(txt, "\n")[[1]]  
 }  
 else if (year == 1999 & sex == "female") {  
 # This was a custom add over what the textbook provided  
 # Find the <pre> elements  
 # We then extract the text from the elements and split it by \n  
 ff = getNodeSet(doc, "//pre")  
 txt = sapply(ff, xmlValue)  
 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 = 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 )  
}  
shortColNames = c("name", "home", "ag", "gun", "net", "time")  
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 = 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)  
 }  
# Importing and cleaning data  
menTables = mapply(extractResTable2, url = urls, year = years, sex = "male")  
names(menTables) = years  
menResMat = lapply(menTables, extractVariables)  
length(menResMat)  
sapply(menResMat, nrow)  
  
age = sapply(menResMat, function(x) as.numeric(x[ , "ag"]))  
  
  
#determine how many years have N/A's  
sapply(age, function(x) sum(is.na(x)))  
#61 has the most values  
  
#Troubleshoot N/A Values starting with 2001  
age2001 = age[["2001"]]  
grep("^===", menTables[['2001']])  
badAgeIndex = which(is.na(age2001)) + 5  
menTables[['2001']][ badAgeIndex ]  
badAgeIndex  
  
#locate all rows that are entirely blank  
blanks = grep("^[[:blank:]]\*$", menTables[['2001']])  
  
which(age2001 < 5)  
#We explore the data to figure out who is less than 5 years old and determine three runners are set to age 0  
menTables[['2001']][ which(age2001 < 5) + 5 ]  
  
#First Box Plot  
boxplot(age, ylab = "Age", xlab = "Year")  
  
  
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)  
}  
  
menDF = mapply(createDF, menResMat, year = 1999:2012,  
 sex = rep("M", 14), SIMPLIFY = FALSE)  
  
sapply(menDF, function(x) sum(is.na(x$runTime)))  
  
cbMen = do.call(rbind, menDF)  
save(cbMen, file = "cbMen.rda")  
  
dim(cbMen)  
  
#Plot run times  
plot(runTime ~ age, data = cbMen, ylim = c(40, 180),  
 xlab = "Age (years)", ylab = "Run Time (minutes)")  
  
library(RColorBrewer)  
ls("package:RColorBrewer")  
  
#Smooth scatter plot of run times - experimenting with color brewer  
smoothScatter(y = cbMen$runTime, x = cbMen$age,  
 ylim = c(40, 165), xlim = c(15, 85),  
 xlab = "Age (years)", ylab = "Run Time (minutes)")  
  
#Remove 15 or under runners  
cbMenSub = cbMen[cbMen$runTime > 30 &  
 !is.na(cbMen$age) & cbMen$age > 15, ]  
  
#Create age categories  
ageCat = cut(cbMenSub$age, breaks = c(seq(15, 75, 10), 90))  
table(ageCat)  
  
#plot off new age category  
plot(cbMenSub$runTime ~ ageCat,  
 xlab = "Age (years)", ylab = "Run Time (minutes)")  
  
#summary of run times by year  
cbMenSub = cbMen[cbMen$runTime > 30 & !is.na(cbMen$age) & cbMen$age > 15, ]  
age1999 = cbMenSub[ cbMenSub$year == 1999, "age" ]  
age2000 = cbMenSub[ cbMenSub$year == 2000, "age" ]  
age2001 = cbMenSub[ cbMenSub$year == 2001, "age" ]  
age2002 = cbMenSub[ cbMenSub$year == 2002, "age" ]  
age2003 = cbMenSub[ cbMenSub$year == 2003, "age" ]  
age2004 = cbMenSub[ cbMenSub$year == 2004, "age" ]  
age2005 = cbMenSub[ cbMenSub$year == 2005, "age" ]  
age2006 = cbMenSub[ cbMenSub$year == 2006, "age" ]  
age2007 = cbMenSub[ cbMenSub$year == 2007, "age" ]  
age2008 = cbMenSub[ cbMenSub$year == 2008, "age" ]  
age2009 = cbMenSub[ cbMenSub$year == 2009, "age" ]  
age2010 = cbMenSub[ cbMenSub$year == 2010, "age" ]  
age2011 = cbMenSub[ cbMenSub$year == 2011, "age" ]  
age2012 = cbMenSub[ cbMenSub$year == 2012, "age" ]  
  
#Plot density of ages by year  
plot(density(age1999, na.rm = TRUE), ylim = c(0, 0.05), col = "purple", lwd = 3, xlab = "Age (years)", main = "")  
lines(density(age2000, na.rm = TRUE), lwd = 3, lty = 2, col="green")  
lines(density(age2001, na.rm = TRUE), lwd = 3, lty = 2, col="red")  
lines(density(age2002, na.rm = TRUE), lwd = 3, lty = 2, col="orange")  
lines(density(age2003, na.rm = TRUE), lwd = 3, lty = 2, col="yellow")  
lines(density(age2004, na.rm = TRUE), lwd = 3, lty = 2, col="blue")  
lines(density(age2005, na.rm = TRUE), lwd = 3, lty = 2, col="grey")  
lines(density(age2006, na.rm = TRUE), lwd = 3, lty = 2, col="violet")  
lines(density(age2007, na.rm = TRUE), lwd = 3, lty = 2, col="brown")  
lines(density(age2008, na.rm = TRUE), lwd = 3, lty = 2, col="black")  
lines(density(age2009, na.rm = TRUE), lwd = 3, lty = 2, col="maroon")  
lines(density(age2010, na.rm = TRUE), lwd = 3, lty = 2, col="#CACACA")  
lines(density(age2011, na.rm = TRUE), lwd = 3, lty = 2, col="#8e0085")  
lines(density(age2012, na.rm = TRUE), lwd = 3, lty = 2, col="#ee6a50")  
legend("topleft", col = c("purple", "green","red","orange","yellow","blue","grey", "violet","brown","black","maroon","#CACACA","#8e0085","#ee6a50"),   
 lty= 1:2, lwd = 3, legend = c("1999","2000","2001","2002","2003","2004","2005","2006","2007", "2008", "2009","2010","2011","2012"), bty = "n")  
  
#Plot density of ages by year  
plot(density(age1999, na.rm = TRUE),  
 ylim = c(0, 0.05), col = "purple",  
 lwd = 3, xlab = "Age (years)", main = "")  
lines(density(age2012, na.rm = TRUE),  
 lwd = 3, lty = 2, col="green")  
legend("topleft", col = c("purple", "green"), lty= 1:2, lwd = 3,  
 legend = c("1999", "2012"), bty = "n")  
  
#qq plot of age differences  
qqplot(age1999, age2012, pch = 19, cex = 0.5,   
 ylim = c(10,90), xlim = c(10,90),   
 xlab = "Age in 1999 Race",  
 ylab = "Age in 2012 Race",   
 main = "Quantile-quantile plot of male runner's age")  
abline(a =0, b = 1, col="red", lwd = 2)