Foundation of Financial Data Science (FE 582)

(Homework 2)

Prof. Dragos Bozdog

Student Name: **Paras Garg**

Course Section: **FE 582 A**

Problem –

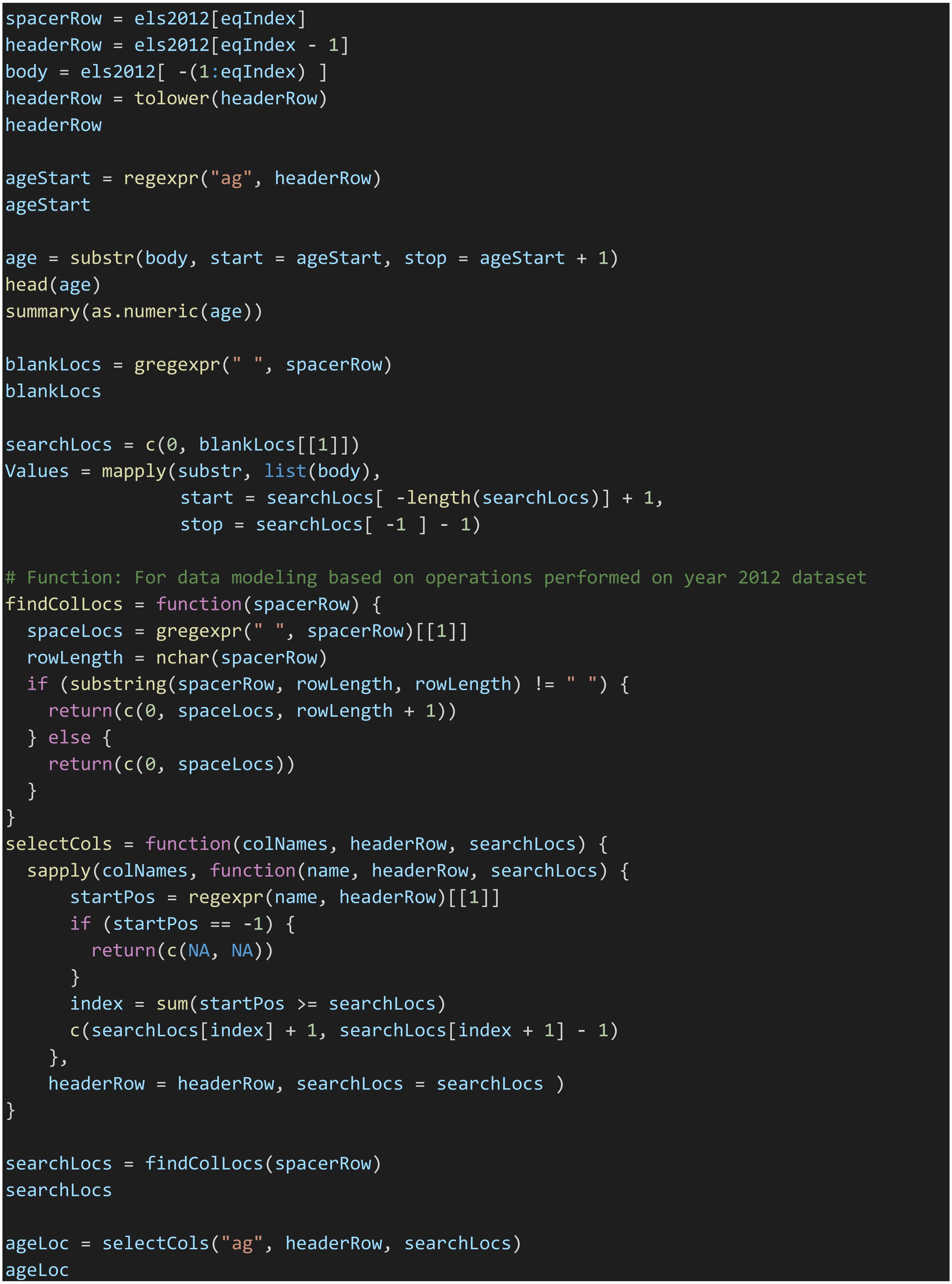
Follow the example in Lecture 3 on Modeling Runners’ Times. Use the Race Result 1999-2012 link (http://cherryblossol.org/aboutus/result_list.php) to extract the Race Result for Order of Finish – Women for a couple of years. Extract and clean the data for as many years as you can. Please do the following:

- Box Plot of Age by Year for Female Runners
- Scatter Plot for Run Times vs. Age for Female Runners
- Fit Models to Average Performance
- Side-by-Side Box plots of Female Runners’ Run Time vs. Age
- Residual Plot from Fitting a Simple Linear Model of Performance to Age
- Piecewise Linear and Loess Curves Fitted to Run Time vs. Age
- Line Plot of the Number of Female Runners by Year
- Density Curves for the Age of Female Runners for 2 years (smallest and largest year that you analyzed)
- Loess Curves Fit to Performance for 2 years (smallest and largest year that you analyzed) Female Runners
- Difference between Loess Curves of the predicted run time for 2 years (smallest and largest year that you analyzed)
- Compare the results of the performance of the male runners (previously analyzed in class) and female runners for the yearly data that you selected to analyze.

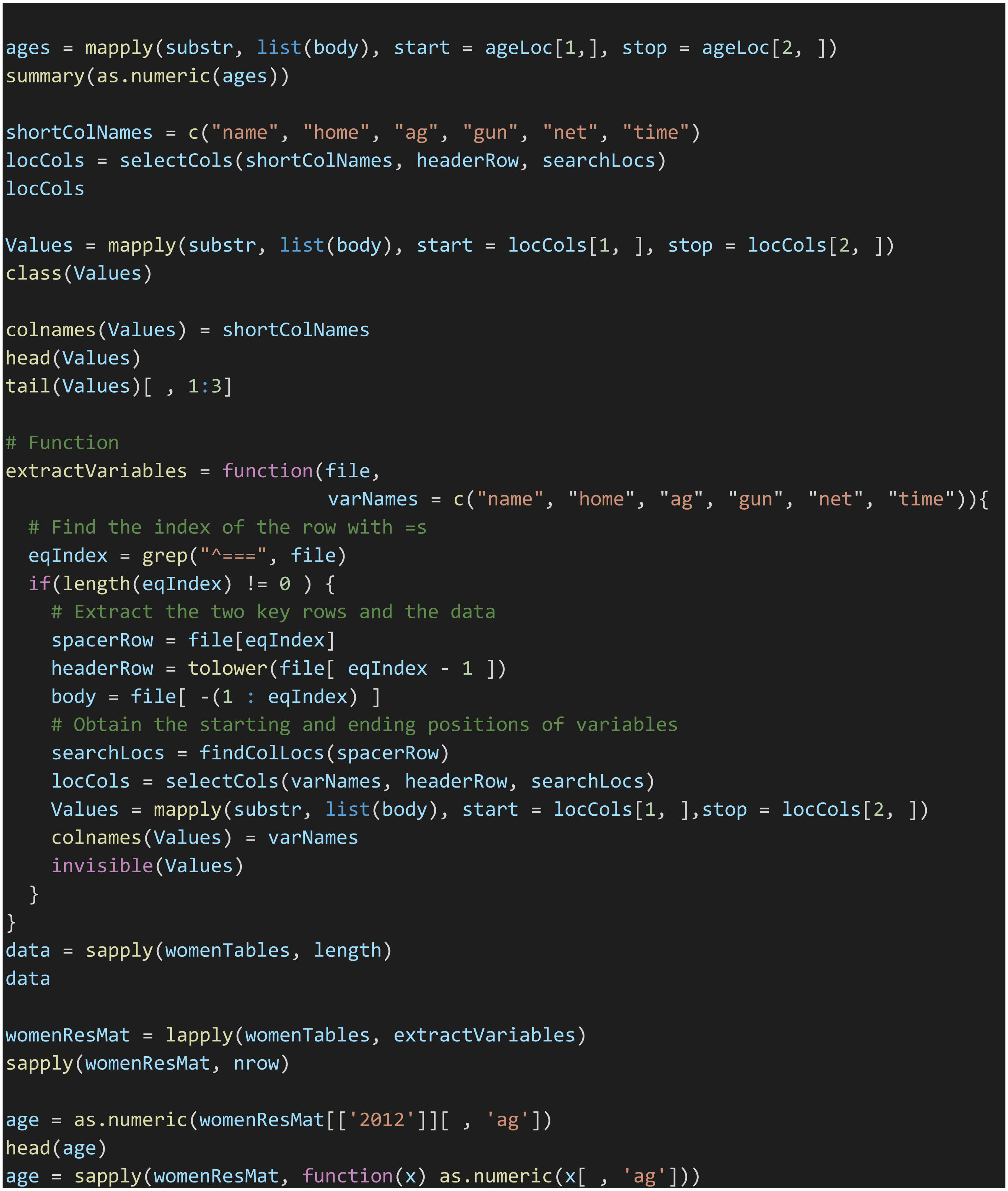
Analysis –

```
# Environment Setup
rm(list = ls())
setwd("C:/Users/Paras Garg/Documents/FE Assignments")
install.packages("XML")
library("XML")

# Scrapping Data
domain = "http://www.cherryblossom.org/"
url = paste(domain, "results/2012/2012cucb10m-f.htm", sep = "")
htmlDoc = htmlParse(url)
preNode = getNodeSet(htmlDoc, "//pre")
txt = xmlValue(preNode[[1]])
nchar(txt)

# Formatting scrapped data
substr(txt, 1, 50)
substr(txt, nchar(txt) - 50, nchar(txt))
els = strsplit(txt, "\\r\\n")[[1]]
length(els)
els[1:3]
els[length(els)]
```

```r
# Function: Retrieves data from website, Find preformatted text, and Return as a
character vector
extractResTable = function(url) {
  htmlDoc = htmlParse(url)
  preNode = getNodeSet(htmlDoc, "//pre")
  txt = xmlValue(preNode[[1]])
  els = strsplit(txt, "\r\n")[[1]]
  if(length(els) == 1) {
    els = strsplit(txt, "\n")[[1]] #If string doesn't have \r eg for year 1999
  }
  return(els)
}
result2012 = extractResTable(url)
identical(result2012, els)

# Women URLs from 1999-2012
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(domain, womenURLs, sep = "")
urls[1:3]

# Women tables
womenTables = lapply(urls, extractResTable)
names(womenTables) = 1999:2012
sapply(womenTables, length)

# Function: Retrieve data from website, Find preformatted text, and Return as a
character vector
extractResTable = function(url, year = 1999) {
  htmlDoc = htmlParse(url)
  if (year == 2000) {
    # Get text from 4th font element
    # File is ill-formed so <pre> search doesn't work.
    fontNode = getNodeSet(htmlDoc, "//font")
    txt = xmlValue(fontNode[[4]])
  } else {
    preNode = getNodeSet(htmlDoc, "//pre")
    txt = xmlValue(preNode[[1]])
  }
  els = strsplit(txt, "\r\n")[[1]]
  if(length(els) == 1) {
    els = strsplit(txt, "\n")[[1]] #If string doesn't have \r eg for year 1999
  }
  return(els)
}
years = 1999:2012
womenTables = mapply(extractResTable, url = urls, year = years)
names(womenTables) = years
sapply(womenTables, length)

# Save File
save(womenTables, file = "CBWomenTextTables.rda")
length(womenTables)

# Checking data for year 1991, 2001, 2012
# Year 1991
womenTables[[1]][[4]]
womenTables[[1]][1:4]
els1991 = womenTables[[1]]
els1991[1:10]

# Year 2001
womenTables[[3]][[4]]
womenTables[[3]][1:4]
womenTables[[3]][[2]] = "PLACE DIV /  NAME                 AG HOMETOWN           TIME
NET "
womenTables[[3]][[3]] = "===== ===== ===================== == ==================
======= ===== "

els2001 = womenTables[[1]]
els2001[1:10]

# Year 2012
womenTables[[14]][[4]]
womenTables[[14]][1:4]
els2012 = womenTables[[14]]
els2012[1:10]

# Data modeling for year 2012 dataset
eqIndex = grep("^===", els2012)
eqIndex

first3 = substr(els2012, 1, 3)
which(first3 == "===")
```



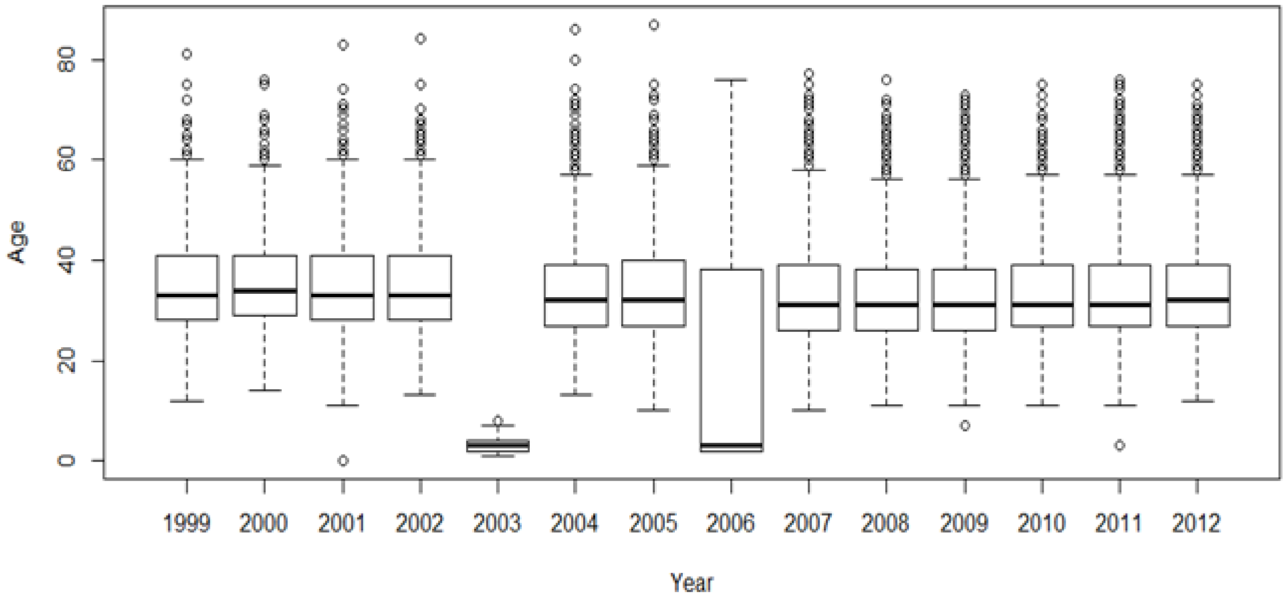
```r
spacerRow = els2012[eqIndex]
headerRow = els2012[eqIndex - 1]
body = els2012[ -(1:eqIndex) ]
headerRow = tolower(headerRow)
headerRow

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)

# Function: For data modeling based on operations performed on year 2012 dataset
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)
searchLocs

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



```
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)
locCols

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

colnames(Values) = shortColNames
head(Values)
tail(Values)[ , 1:3]

# Function
extractVariables = function(file,
                            varNames = c("name", "home", "ag", "gun", "net", "time")){
  # Find the index of the row with =s
  eqIndex = grep("^===", file)
  if(length(eqIndex) != 0 ) {
    # 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)
  }
}
data = sapply(womenTables, length)
data

womenResMat = lapply(womenTables, extractVariables)
sapply(womenResMat, nrow)

age = as.numeric(womenResMat[['2012']][ , 'ag'])
head(age)
age = sapply(womenResMat, function(x) as.numeric(x[ , 'ag']))
```

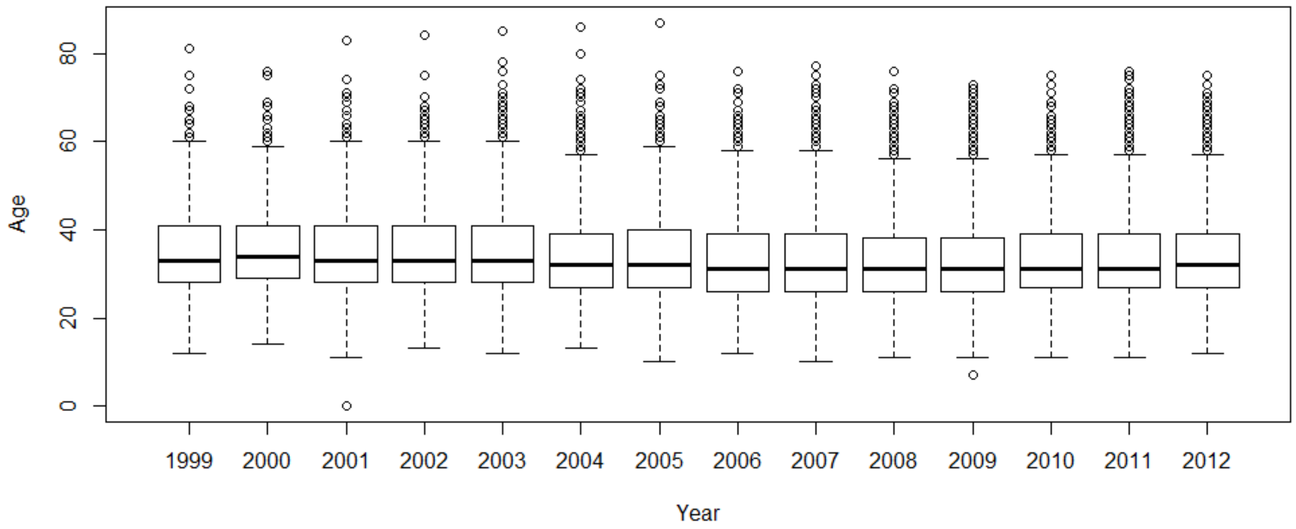
```
# Box Plot of Age by Year for Female Runners
```

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



```
# Need to modify selectCols() by changing the index for end of each variable when we
perform the extraction
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']))
```

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

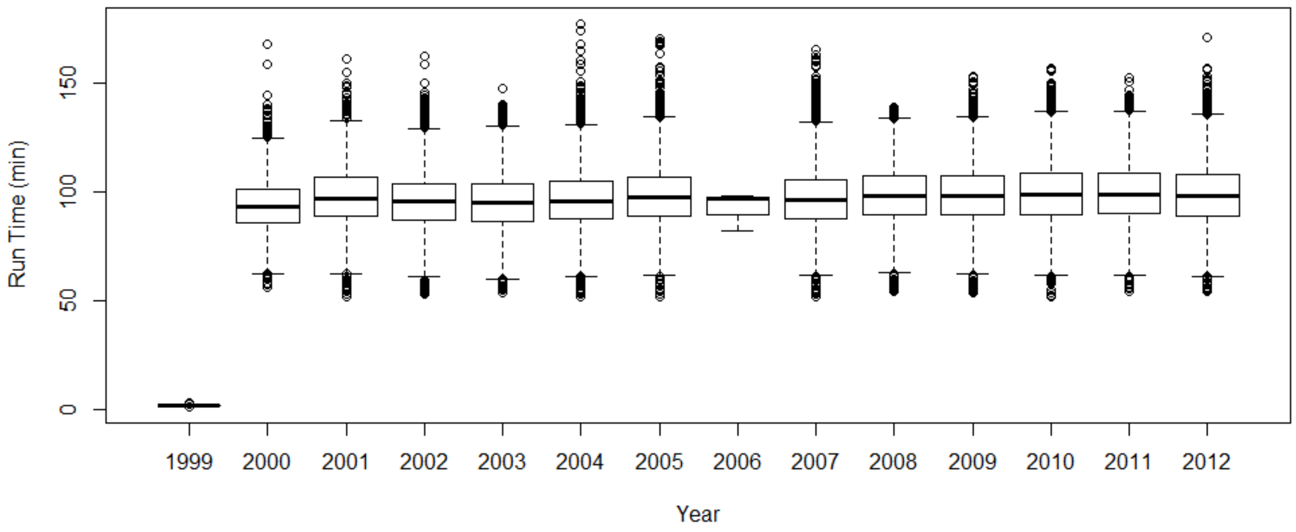


```
sapply(womenResMat, nrow)
sapply(age, function(x) sum(is.na(x)))

age1999 = age[["1999"]]
grep("^===", womenTables[[1]])

womenTables[[1]][1:10]
womenTables[['1999']][1:10]
badAgeIndex = which(is.na(age1999)) + 5
womenTables[['1999']][ badAgeIndex ]

blanks = grep("^[[:blank:]]*$", womenTables[['1999']])
blanks

extractVariables = function(file, varNames =c("name", "home", "ag", "gun", "net",
"time")) {
  # Find the index of the row with =s
  eqIndex = grep("^===", file)
  if(length(eqIndex) != 0 ) {
    # 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
```

```r
    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)
which(age1999 < 5)

womenTables[['1999']][ which(age1999 < 5) + 5 ]

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,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)
  }
womenDF = mapply(createDF,
                 womenResMat,
                 year = 1999:2012,
                 sex = rep("W", 5),
                 SIMPLIFY = FALSE)

warnings()[ c(1:2, 49:50) ]
sapply(womenDF, function(x) sum(is.na(x$runTime)))

createDF = function(Res, year, sex) {
  Res = as.matrix(Res)
  # 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)
}
womenDF = mapply(createDF, womenResMat, year = 1999:2012, sex = rep("W", 5), SIMPLIFY =
FALSE)
sapply(womenDF, function(x) sum(is.na(x$runTime)))

separatorIdx = grep("^===", womenTables[["1999"]])
separatorRow = womenTables[['1999']][separatorIdx]
separatorRowX = paste(substring(separatorRow, 1, 63), " ",
                      substring(separatorRow, 65, nchar(separatorRow)),
                      sep = "")
```

```
womenTables[['1999']][separatorIdx] = separatorRowX
womenResMat = sapply(womenTables, extractVariables)
womenDF = mapply(createDF, womenResMat, year = 1999:2012,
                 sex = rep("W", 5), SIMPLIFY = FALSE)
sapply(womenDF, function(x) sum(is.na(x$runTime)))
```

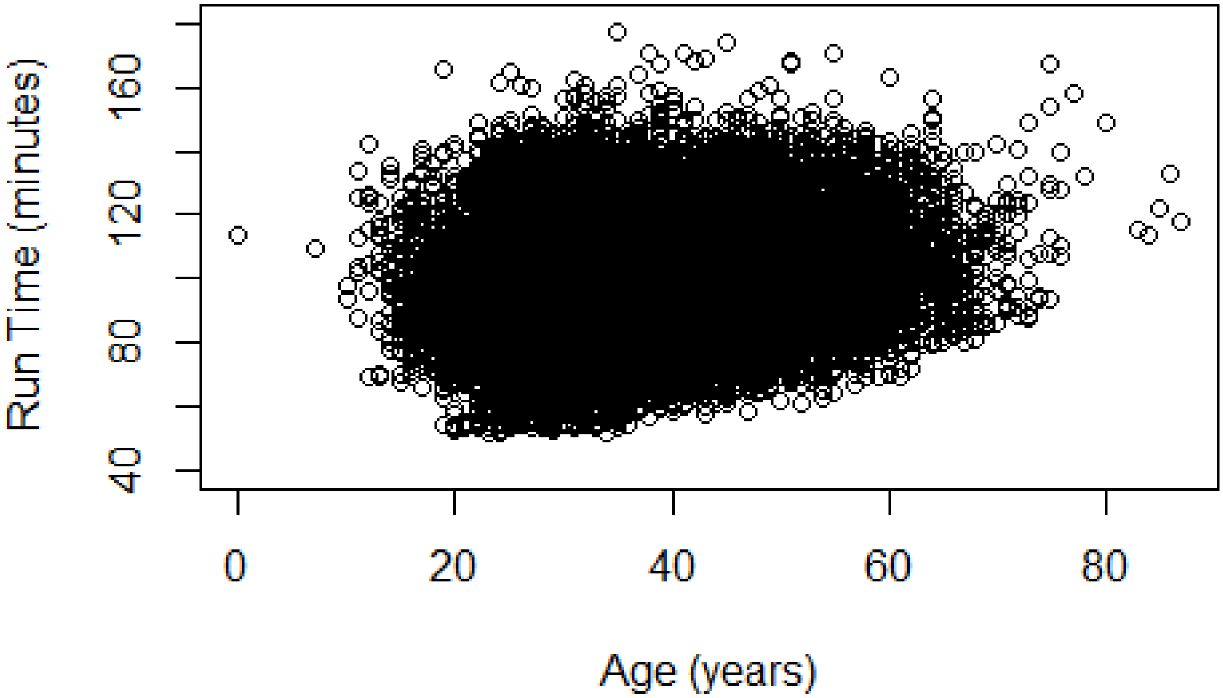
```
boxplot(sapply(womenDF, function(x) x$runTime),
        xlab = "Year", ylab = "Run Time (min)")
```



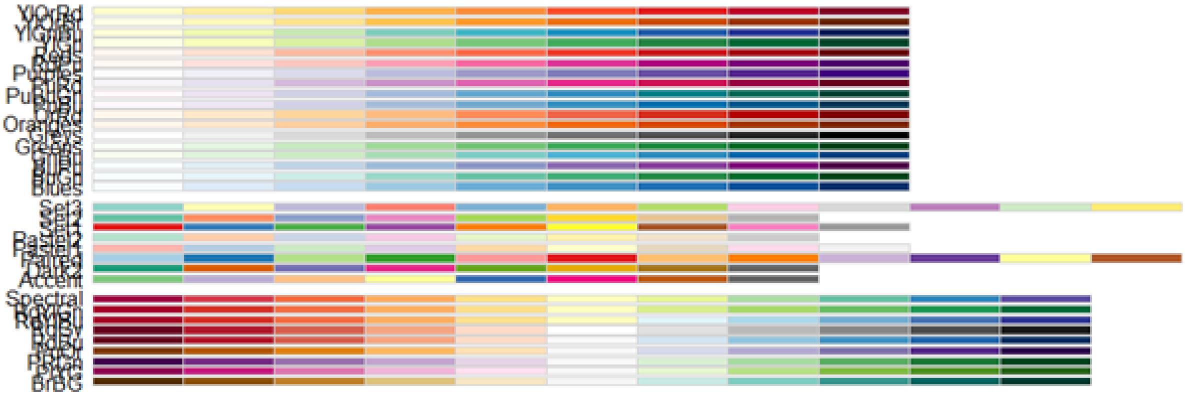
```
# Scatter Plot for Run Times vs. Age for Female Runners
```

```
cbWomen = do.call(rbind, womenDF)
save(cbWomen, file = "cbWomen.rda")
dim(cbWomen)
#load("cbWomen.rda")
```

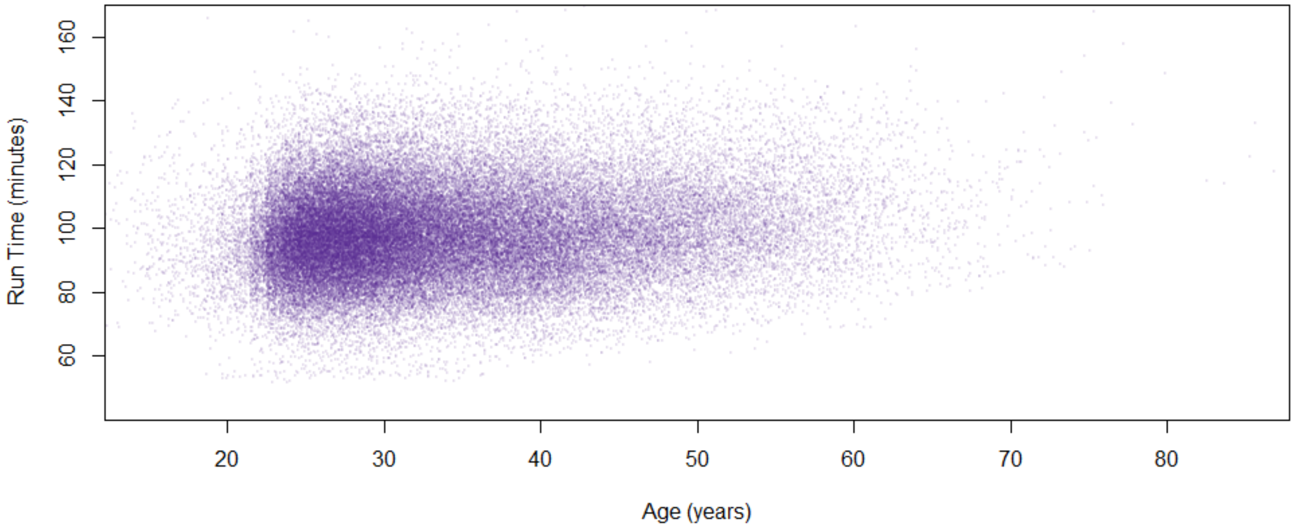
```
plot(runTime ~ age, data = cbWomen, ylim = c(40, 180),
     xlab = "Age (years)", ylab = "Run Time (minutes)")
```



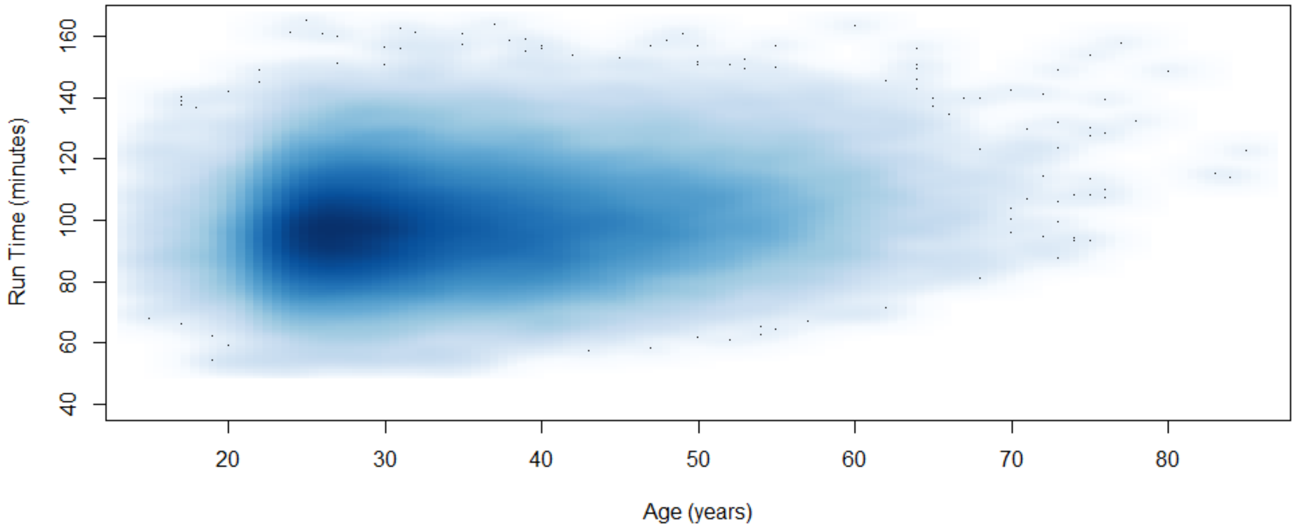
```
library(RColorBrewer)
ls("package:RColorBrewer")
display.brewer.all()
```



```
Purples8 = brewer.pal(9, "Purples")[8]
Purples8
Purples8A = paste(Purples8, "14", sep = "")
plot(runTime ~ jitter(age, amount = 0.5),
     data = cbWomen,
     pch = 19,cex = 0.2, col = Purples8A,
     ylim = c(45, 165), xlim = c(15, 85),
     xlab = "Age (years)", ylab = "Run Time (minutes)")
```

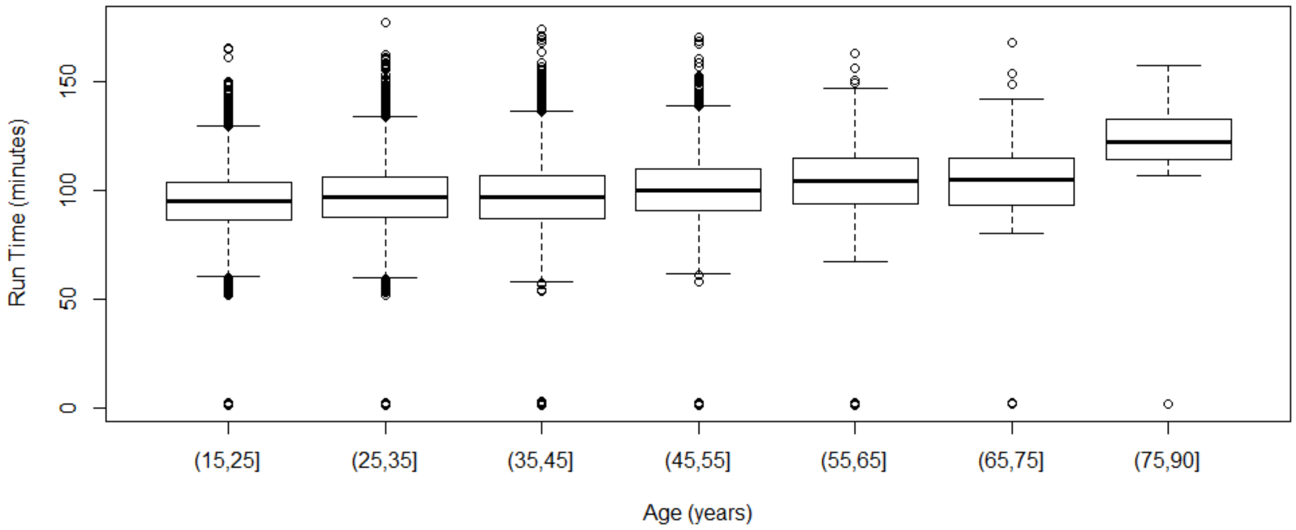


```
smoothScatter(y = cbWomen$runTime, x = cbWomen$age,
              ylim = c(40, 165), xlim = c(15, 85),
              xlab = "Age (years)", ylab = "Run Time (minutes)")
```



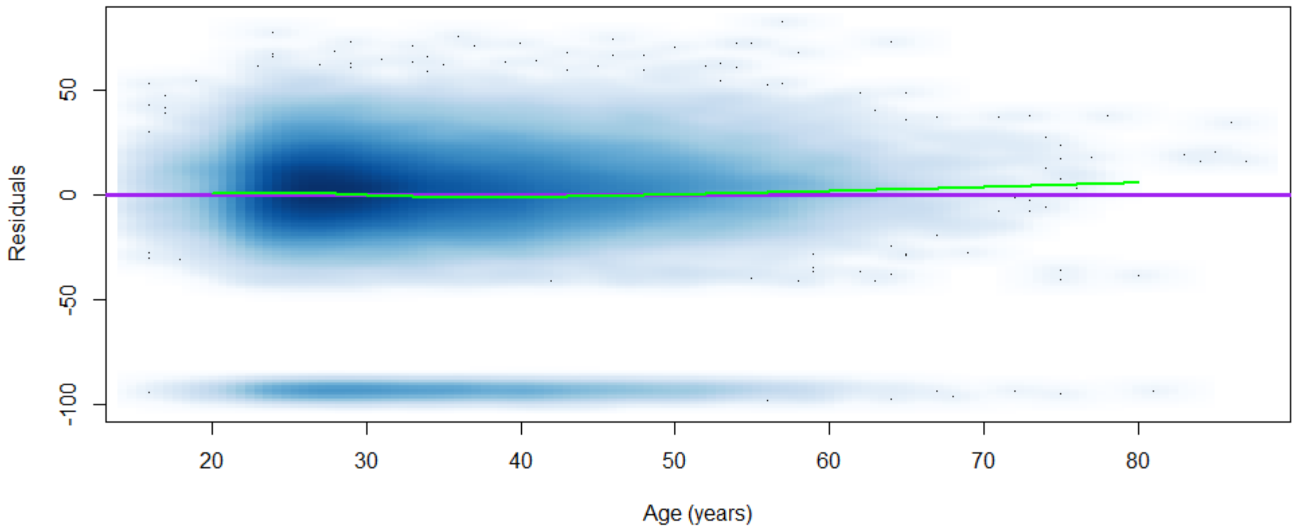
```
# Side-by-Side Box Plot of Female Runners’ Run Time vs. Ags
```

```
cbWomenSub = cbWomen[cbWomen$runTime > 30 & !is.na(cbWomen$age) & cbWomen$age > 15,]
ageCat = cut(cbWomenSub$age, breaks = c(seq(15, 75, 10), 90))
table(ageCat)
plot(cbWomenSub$runTime ~ ageCat, xlab = "Age (years)", ylab = "Run Time (minutes)")
```



# Residual Plot from Fitting a Simple Linear Model of Performance to Age

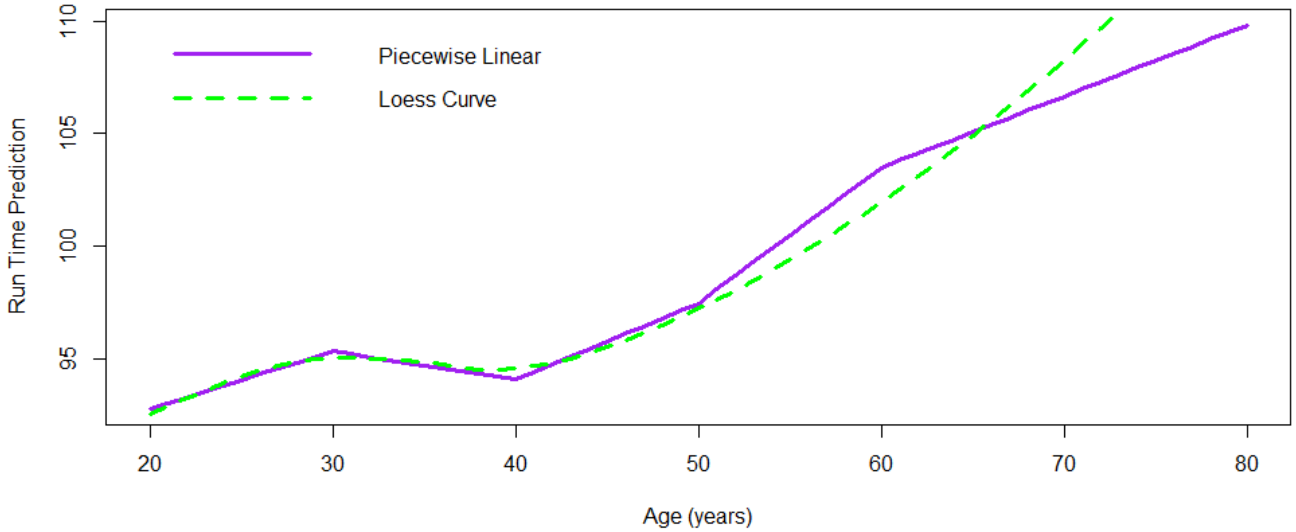
```
lmAge = lm(runTime ~ age, data = cbWomenSub)
lmAge$coefficients
summary(lmAge)
class(lmAge)

cbWomenSubAge = cbWomenSub$age[1:length(lmAge$residuals)] # length(cbWomenSubAge) = length(lmAge$residuals)
smoothScatter(x = cbWomenSubAge, y = lmAge$residuals, xlab = "Age (years)", ylab = "Residuals")
abline(h = 0, col = "purple", lwd = 3)

resid.lo = loess(resids ~ age, data = data.frame(resids = residuals(lmAge), age = cbWomenSubAge))
age20to80 = 20:80
age20to80
resid.lo.pr = predict(resid.lo, newdata = data.frame(age = age20to80))
lines(x = age20to80, y = resid.lo.pr, col = "green", lwd = 2)
```

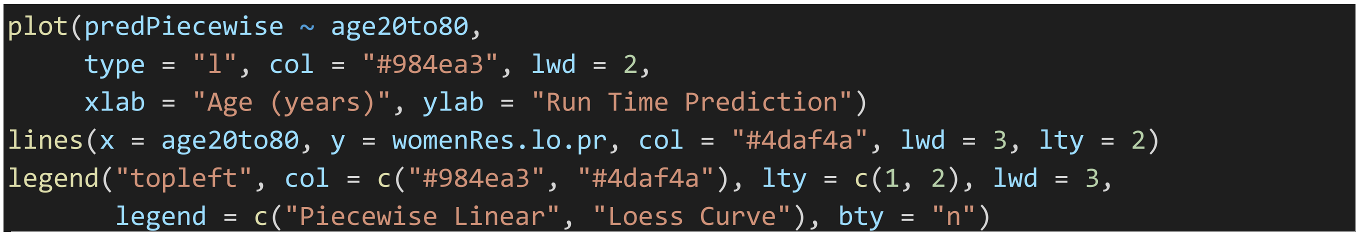


Piecewise Linear and Loess Curves Fitted to Run Time vs. Age

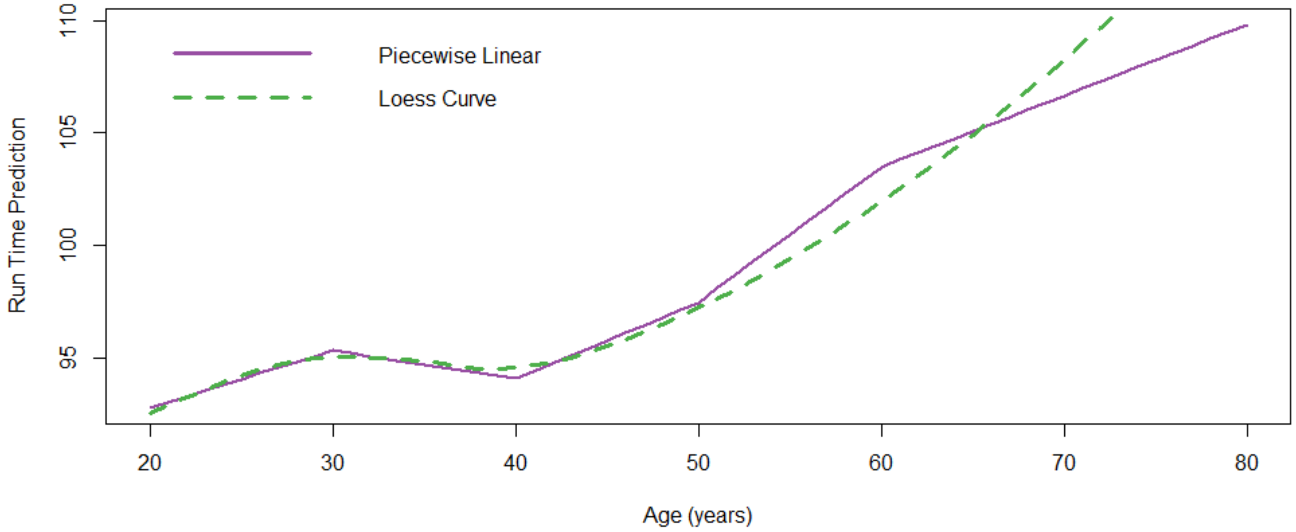
```
womenRes.lo = loess(runTime ~ age, cbWomenSub)
womenRes.lo.pr = predict(womenRes.lo, data.frame(age = age20to80))
over50 = pmax(0, cbWomenSub$age - 50)
lmOver50 = lm(runTime ~ age + over50, data = cbWomenSub)
summary(lmOver50)

decades = seq(30, 60, by = 10)
overAge = lapply(decades, function(x) pmax(0, (cbWomenSub$age - x)))
names(overAge) = paste("over", decades, sep = "")
overAge = as.data.frame(overAge)
tail(overAge)

lmPiecewise = lm(runTime ~ . , data = cbind(cbWomenSub[, c("runTime", "age")],
overAge))
summary(lmPiecewise)

overAge20 = lapply(decades, function(x) pmax(0, (age20to80 - x)))
names(overAge20) = paste("over", decades, sep = "")
overAgeDF = cbind(age = data.frame(age = age20to80), overAge20)
tail(overAgeDF)
predPiecewise = predict(lmPiecewise, overAgeDF)
```

```
plot(predPiecewise ~ age20to80,
     type = "l", col = "purple", lwd = 3,
     xlab = "Age (years)", ylab = "Run Time Prediction")
lines(x = age20to80, y = womenRes.lo.pr,
      col = "green", lty = 2, lwd = 3)
legend("topleft", col = c("purple", "green"),
       lty = c(1, 2), lwd= 3,
       legend = c("Piecewise Linear", "Loess Curve"), bty = "n")
```



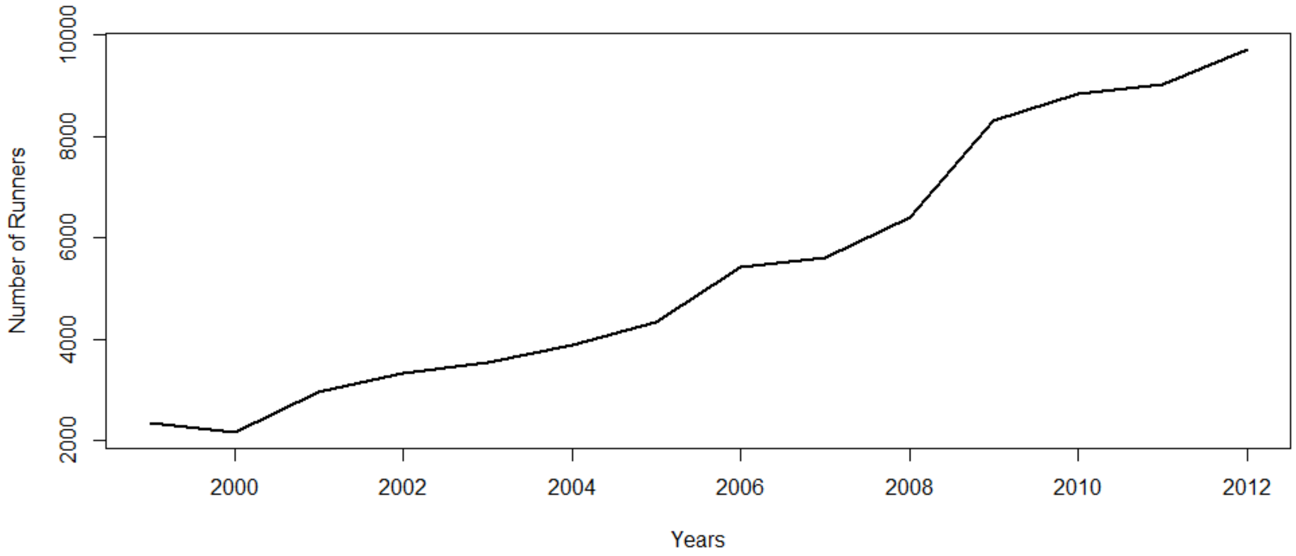


```
plot(predPiecewise ~ age20to80,
     type = "l", col = "#984ea3", lwd = 2,
     xlab = "Age (years)", ylab = "Run Time Prediction")
lines(x = age20to80, y = womenRes.lo.pr, col = "#4daf4a", lwd = 3, lty = 2)
legend("topleft", col = c("#984ea3", "#4daf4a"), lty = c(1, 2), lwd = 3,
       legend = c("Piecewise Linear", "Loess Curve"), bty = "n")
```

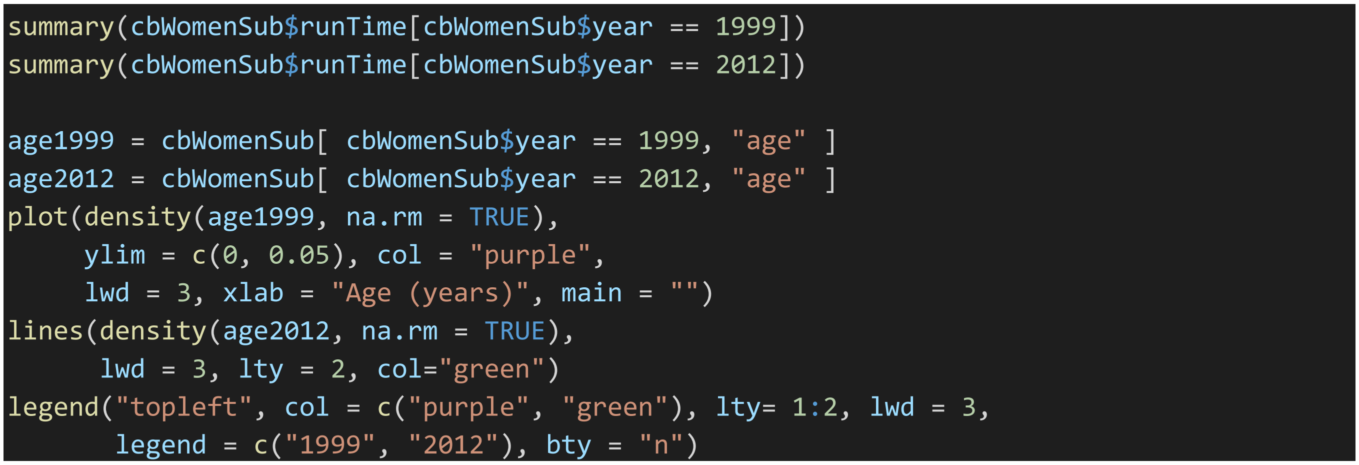


```
# Line Plot of the Number of Female Runners by Year
```

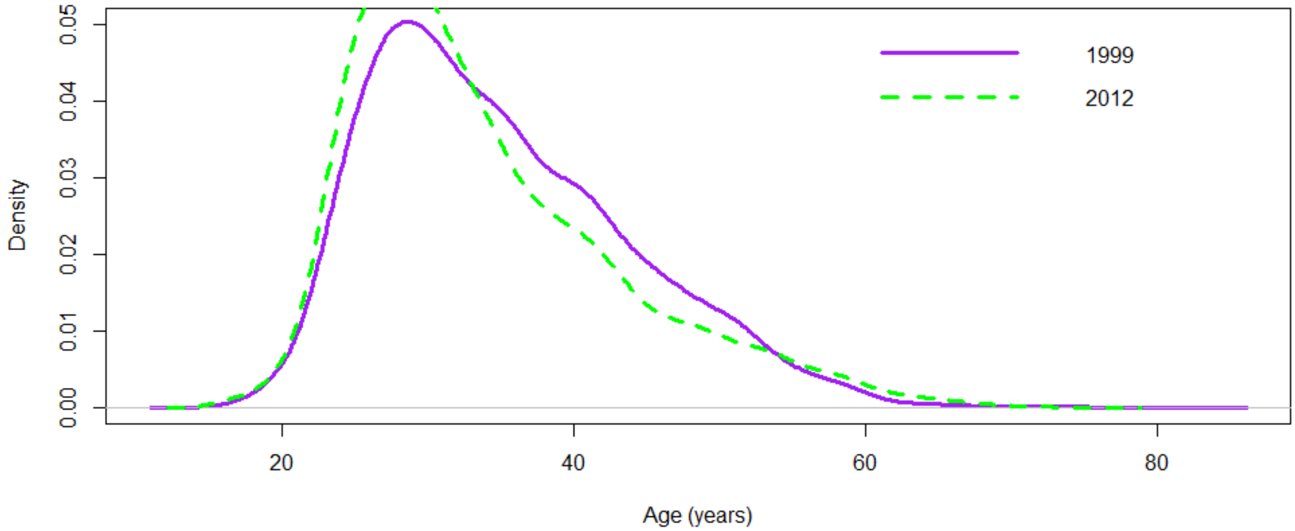
```
numRunners = with(cbWomen, tapply(runTime, year, length))
plot(numRunners ~ names(numRunners), type="l", lwd = 2,
     xlab = "Years", ylab = "Number of Runners")
```



# Density Curves for the Age of Female Runners for 2 years (smallest and largest year that has been analyzed)

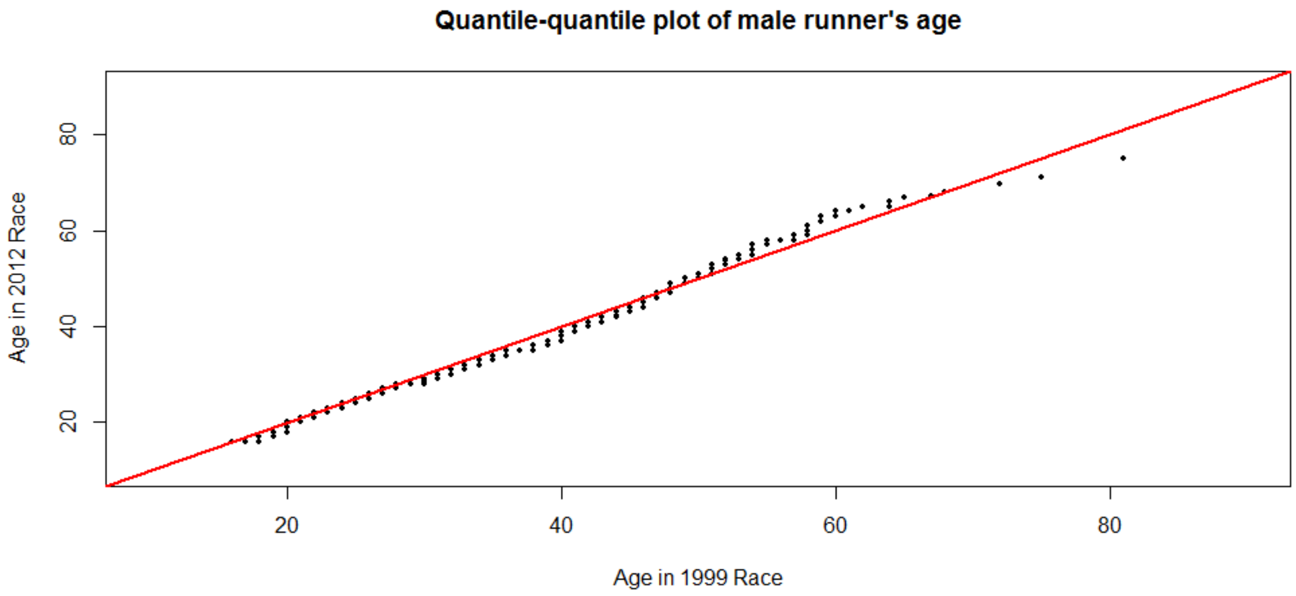


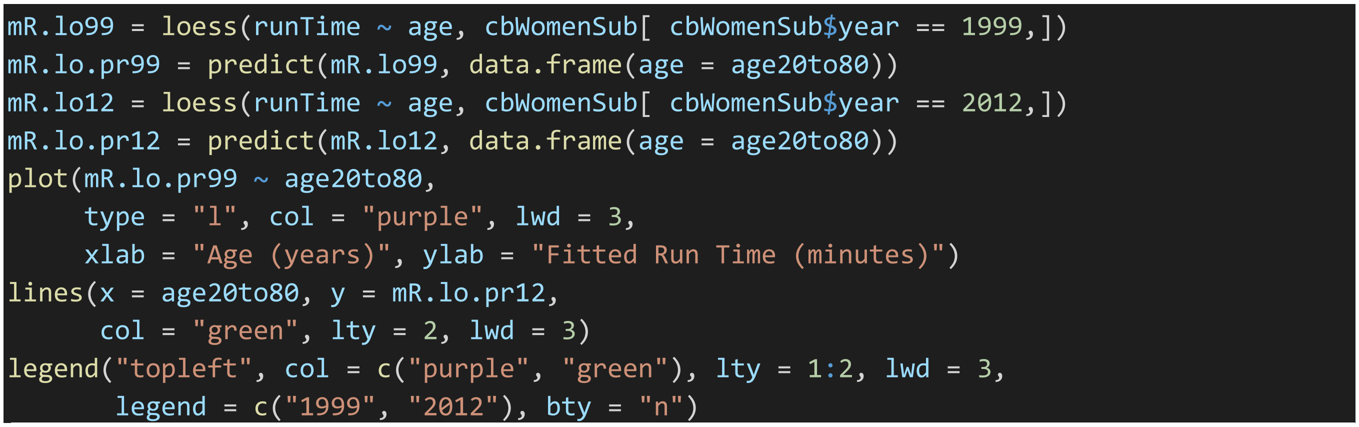
```
summary(cbWomenSub$runTime[cbWomenSub$year == 1999])
summary(cbWomenSub$runTime[cbWomenSub$year == 2012])

age1999 = cbWomenSub[ cbWomenSub$year == 1999, "age" ]
age2012 = cbWomenSub[ cbWomenSub$year == 2012, "age" ]
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")
```



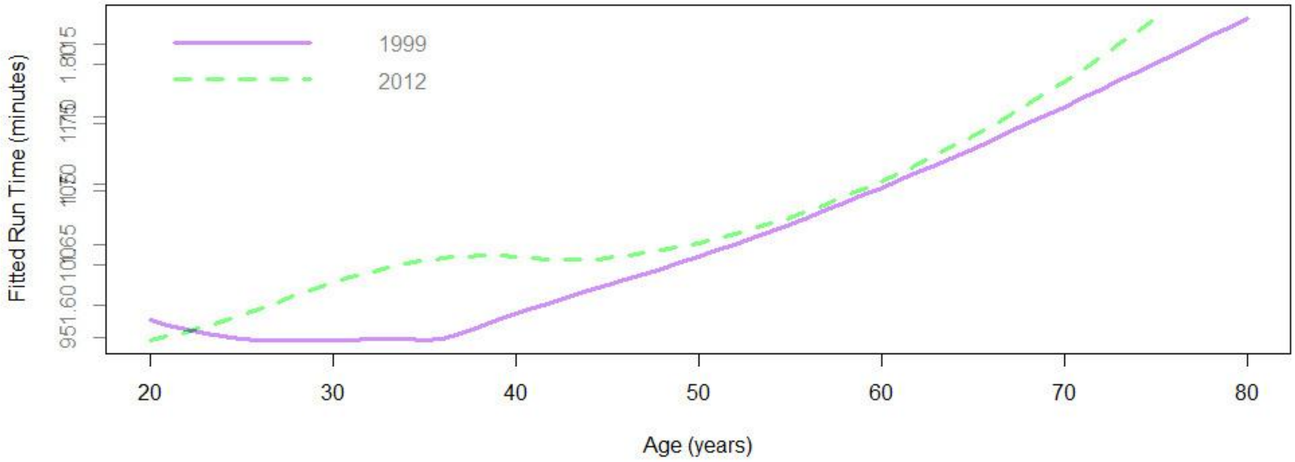
Loess Curves Fit to Performance for 2 years (smallest and largest year that you analyzed) Female Runners

```
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)
```



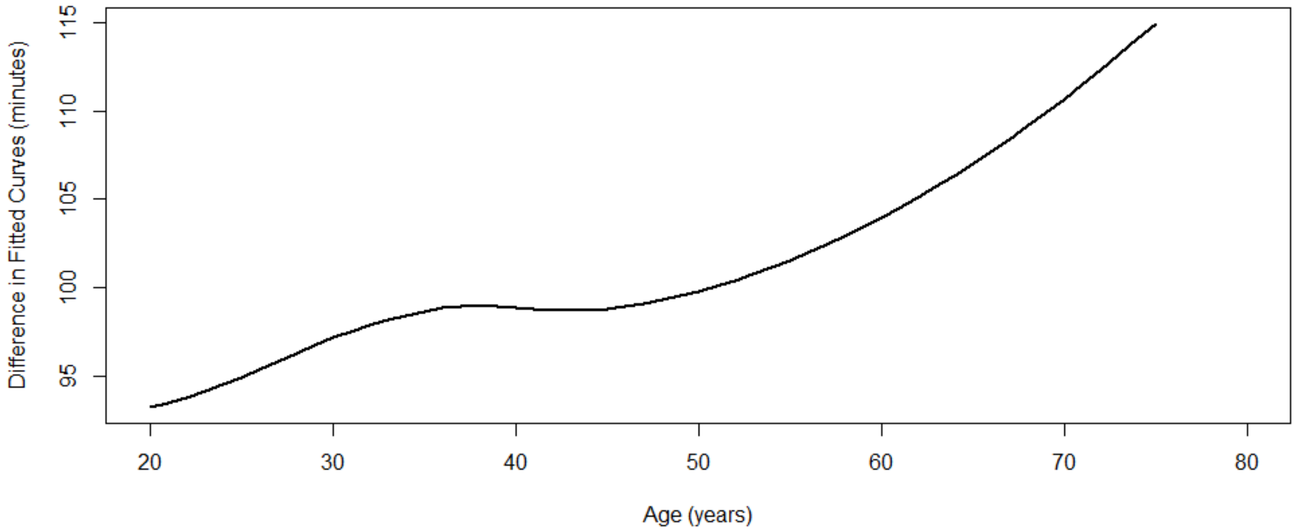


```
mR.lo99 = loess(runTime ~ age, cbWomenSub[ cbWomenSub$year == 1999,])
mR.lo.pr99 = predict(mR.lo99, data.frame(age = age20to80))
mR.lo12 = loess(runTime ~ age, cbWomenSub[ cbWomenSub$year == 2012,])
mR.lo.pr12 = predict(mR.lo12, data.frame(age = age20to80))
plot(mR.lo.pr99 ~ age20to80,
     type = "l", col = "purple", lwd = 3,
     xlab = "Age (years)", ylab = "Fitted Run Time (minutes)")
lines(x = age20to80, y = mR.lo.pr12,
      col = "green", lty = 2, lwd = 3)
legend("topleft", col = c("purple", "green"), lty = 1:2, lwd = 3,
       legend = c("1999", "2012"), bty = "n")
```



# Difference between Loess Curves of the predicted run time for 2 years (smallest and largest year that has been analyzed)

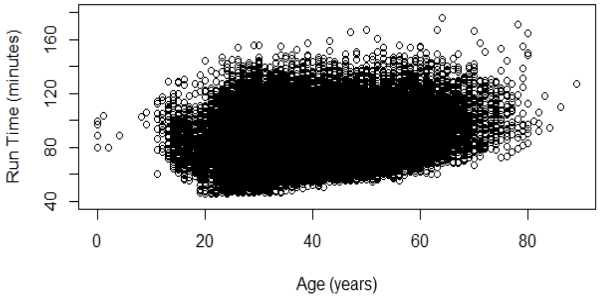
```
gap14 = mR.lo.pr12 - mR.lo.pr99
plot(gap14 ~ age20to80, type = "l" , xlab = "Age (years)",
     ylab = "Difference in Fitted Curves (minutes)", lwd = 2)
```

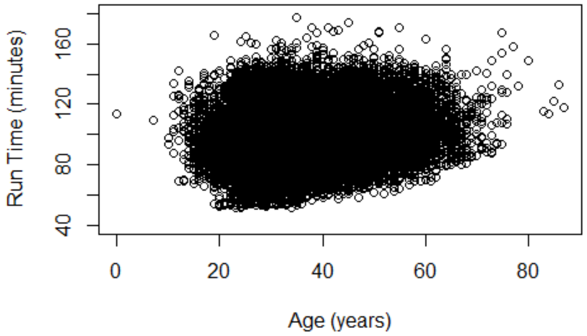


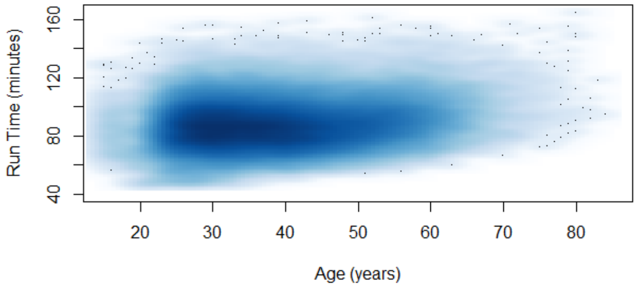
# Compare the results of the performance of the male runners (previously analyzed) and female runners for the yearly data that you selected to analyze.

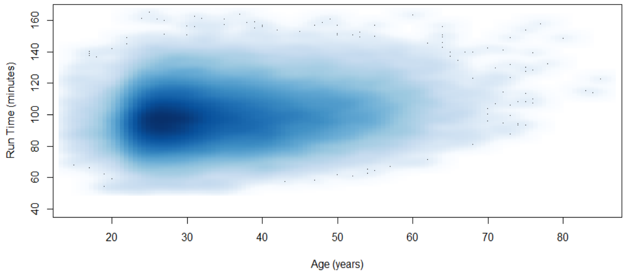
Male

Female









As per the graphical analysis done we can conclude that female runners are more prominent in the age span between 20 and 30 whereas on the other hand male runners are prominent in the age span between 20 and 40.