Stat243: Problem Set 3

Sicun Huang

September 30, 2015

1. Reading questions

(c) Although workflow management tools are useful in terms of automatically modify related files when one file has been updated, could it potentially create errors if the conditions under which to modify such files are wrongly defined? Would it impose further challenges while locating errors since files are being modified automatically?

Could automated testing be used to resolve the above problem?

Is there a down side for test-driven development?

2.

(a) Use the XML package to scrape info from the provided url. We first parse the url, then get all the a nodes. Then we create a boolean vector with TRUE's on the indices corresponding to the nodes containing "The First" in their names. Take only the relevant nodes by overlaying the list of nodes with the boolean vector and get their "href" attributes, which give the urls for the first debates. Further subset the urls by their years and return the relevant ones.

```r
library(XML)
library(stringr)

theUrl <- "http://www.debates.org/index.php?page=debate-transcripts"

getFirstDebateUrl <- function(url, year){
  #parse url
  doc <- htmlParse(url)
  #get all a nodes
  listOfNodes <- getNodeSet(doc, "//a[@href]")
  #use xmlValue to get the text of the nodes
  #create boolean vector; TRUE if text contains "The First"
  bools <- str_detect( sapply(listOfNodes, xmlValue), "The First" )
  #overlay listOfNodes with boolean vector; only include the nodes for the first debates
  #get herf attributes i.e. urls
  firstDebateUrl <- sapply( listOfNodes[bools], xmlGetAttr, "href" )
  #grep url indices in the year we passed in
  index <- grep(year, firstDebateUrl)
  #return only first debate urls of the relevant year
  firstDebateUrl[index]
}
```

(b) After inspecting the website, notice that all body paragraphs have the xpath "//p/text()." Use such a property to extract the transcripts by paragraphs. Store the transcripts in a list with each element being debate from a year.

```r
extractByP <- function(url){
  doc <- htmlParse(url)
  #find matching nodes with pattern "//p/text()" i.e. body paragraph
  bodyByP <- xpathSApply(doc, '//p/text()', xmlValue)
}

listOfUrl <- list()
listOfTrans <- list()
year <- c("1996", "2000", "2004", "2008", "2012")

for( i in 1:5 ){
  listOfUrl[[i]] <- getFirstDebateUrl(theUrl,year[i])
  listOfTrans[[i]] <- extractByP(listOfUrl[[i]])
}

cat(head(listOfTrans[[2]], sep = "\n"))

##
##  MODERATOR: Good evening from the Clark Athletic Center at the University of Massachusetts in Boston
```

(c) Write individual functions to: split a transcript by speakers and save it into a data frame; merge the adjacent rows spoken by the same person; extract text by individual speakers and save them as attributes; and count the number of appearance of a certain string. Update data frame with the above functions and add the attributes for text spoken by a certain speaker and number of laughters/applauses he gets.

```r
#initiate a list
dat <- list()

transSplit <- function(transcript){
  #original transcripts were divided up by lines; paste them into one element
  transcript <- paste(transcript, collapse ="\n")
  #split transcript by speaker, which is marked with capital letters followed by colon
  splitBySpeaker <- strsplit(transcript, "[A-Z]+: ")
  #extract speaker names (character vector)
  speakerNames <- str_extract_all(transcript, "[A-Z]+: ")
  data <- data.frame( matrix(0, nrow=length(speakerNames[[1]]), ncol=2) )
  #put speaker names into first column of data frame
  data[,1] <- speakerNames
  #put the rest of the transcript (after stripping out speakers) into second column of
  #data frame; observe that the first element (header) does not correspond to a speaker,
  #exclude it
  data[,2] <- splitBySpeaker[[1]][-1]
  #special case: part of header says "speakers: ..."; exclude that element if applicable
  if(data[1,1]=="SPEAKERS: "){
    data <- data[-1,]
  }
  colnames(data) <- c("speaker", "content")
  return(data)
}

mergeSpeaker <- function(data){
  #initiate
  bools <- rep(TRUE, nrow(data))
  for( i in 1:nrow(data) ){
    #adjacent rows by same speaker
    if( identical(data$speaker[i], data$speaker[i+1]) ){
      #combine text spoken
      data[i,2] <- paste(data[i,2], data[i+1,2])
      #set element in logical vector to FALSE so the repetitive row will be left out when
      #subsetting
      bools[i+1] <- FALSE
    }
  }
  data <- subset(data, bools)
}

chunkBySpeaker <- function(data){
  #get unique speaker names
  uniqueSpeakers <- unique(data[,1])
  for(i in 1:length(uniqueSpeakers) ){
    #assign text spoken by each speaker to the corresponding attribute
    attr(data, uniqueSpeakers[i]) <- data[data$speaker==uniqueSpeakers[i],2]
  }
  return(data)
}

countStr <- function(data, string){
  uniqueSpeakers <- unique(data[,1])
  #set name of the attribute depending on the string passed in
  attrNames <- sapply(uniqueSpeakers, function(x){paste(x, string)})
  for( i in 1:length(uniqueSpeakers) ){
    #get a vector of the locations of the string; save its length i.e. number of appearance
    #to the corresponding attribute
    attr(data, attrNames[i]) <- length(grep(string, attr(data, uniqueSpeakers[i])))
  }
  return(data)
}

#loop through transcripts from 5 years (list of 5 transcripts)
for(i in 1:5){
  dat[[i]] <- transSplit(listOfTrans[[i]])
  dat[[i]] <- mergeSpeaker(dat[[i]])
  #strip out new line characters
  dat[[i]]$content <- sapply(dat[[i]]$content, function(x){str_replace_all(x, "\n", " ")})
  dat[[i]] <- chunkBySpeaker(dat[[i]])
  dat[[i]] <- countStr(dat[[i]], "APPLAUSE")
  dat[[i]] <- countStr(dat[[i]], "LAUGHTER")
  #strip out text not spoken
  dat[[i]]$content <- sapply(dat[[i]]$content, function(x){str_replace_all(x, "\\([a-zA-Z]+\\)", "")})
  #update attributes for text spoken by each speaker
  dat[[i]] <- chunkBySpeaker(dat[[i]])
}

head(dat[[5]])

##     speaker
## 2  LEHRER:
## 5   OBAMA:
## 6  LEHRER:
## 7  ROMNEY:
## 9  LEHRER:
```

```
## 10  OBAMA:
##
## 2
## 5
## 6
## 7                                                                             Thank you, Jim. It's an hon
## 9
## 10 Well, let me talk specifically about what I think we need to do. First, we've got to improve our
```

```
head(attr(dat[[5]], "OBAMA: "))
```

```
## [1] "Well, thank you very much, Jim, for this opportunity. I want to thank Governor Romney and the Un
## [2] "Well, let me talk specifically about what I think we need to do. First, we've got to improve our
## [3] "Well, I think -- let's talk about taxes, because I think it's instructive. Now, four years ago,
## [4] "When you add up all the loopholes and deductions that upper-income individuals can -- are curren
## [5] "Well, for 18 months he's been running on this tax plan. And now, five weeks before the election
## [6] "Jim, I -- you may want to move onto another topic, but I -- I would just say this to the America
```

```
attr(dat[[5]], "OBAMA:  LAUGHTER")
```

```
## [1] 3
```

(d) Write functions to split the transcripts of each speaker into sentences and words respectively. Clean up the text so that ". " can be used as the delimiter of sentences and " " can be used as that of words. Store them in a list of 5 lists of character vectors.

```
bySentence <- list()
byWord <- list()

extractSentence <- function( data ){
  uniqueSpeakers <- unique(data[,1])
  sentenceSplit <- list()
  for( i in 1:length(uniqueSpeakers) ){
    #get text spoken by ith speaker
    attrSpeaker <- attr(data, uniqueSpeakers[i])
    #checked website: no exclamation points, no Mrs Ms Prof
    #swap Mr./Dr. for Mr/Dr so the period doesn't interfere with other periods when
    #spliting into sentences
    sentenceSplit[[i]] <- str_replace_all(attrSpeaker, "Mr. ", "Mr ")
    sentenceSplit[[i]] <- str_replace_all(sentenceSplit[[i]], "Dr. ", "Dr ")
    #swap ? for . so it can be used to delimit sentences
    sentenceSplit[[i]] <- str_replace_all(sentenceSplit[[i]],  "\\? ", "\\. ")
    #delet ... in the middle of sentences
    sentenceSplit[[i]] <- str_replace_all(sentenceSplit[[i]], "(\\.\\.\\.)+ ", "")
    #split on ". "; unlist to get a character vector with each element being a sentence
    sentenceSplit[[i]] <- unlist(strsplit(sentenceSplit[[i]], "\\. "))
  }
  #write text spoken by each speaker (splited by sentences) into a list
  list <- list(sentenceSplit[1:length(uniqueSpeakers)])
}

extractWord <- function( listSentence ){
  wordSplit <- list()
  for(i in 1:3){
    #delet punctuations
    wordSplit[[i]] <- str_replace_all(listSentence[[i]], ",", "")
    wordSplit[[i]] <- str_replace_all(wordSplit[[i]], "-- ", "")
    wordSplit[[i]] <- str_replace_all(wordSplit[[i]], '\"', "")
    wordSplit[[i]] <- str_replace_all(wordSplit[[i]], ";", "")
    wordSplit[[i]] <- str_replace_all(wordSplit[[i]], ":", "")
    #split on " "; unlist to get a character vector with each element being a word
    wordSplit[[i]] <- unlist(strsplit(wordSplit[[i]], " "))
  }
  #write text spoken by each speaker (splited by words) into a list
  list <- list(wordSplit[1:length(listSentence)])
}

#loop through transcripts from 5 years
for(i in 1:5){
    bySentence[[i]] <- extractSentence( dat[[i]] )
    byWord[[i]] <- extractWord ( bySentence[[i]][[1]] )
}

head(bySentence[[2]][[1]][[2]])

## [1] "Well, Jim, first of all, I would like to thank the sponsors of this debate and the people of Bo
## [2] "I would like to thank Governor Bush for participating, and I would like to say I'm happy to be 
## [3] "I have actually not questioned Governor Bush's experience"
## [4] "I have questioned his proposals"
## [5] "And here is why"
## [6] "I think this is a very important moment for our country"

head(byWord[[2]][[1]][[2]])

## [1] "Well"  "Jim"   "first" "of"    "all"   "I"
```

(e) Calculate number of words, number of characters, and average length of words spoken by each speaker and save them in a matrix. Aggregate all data from 5 years of debates in a list.

```
sumWords <- list()

for( i in 1:5 ){
wordCount <- matrix(0, nrow = 3, ncol=4)
uniqueSpeakers <- unique(dat[[i]][,1])
#get total number of words from a certain speaker
numWords <- sapply(byWord[[i]][[1]], length)
#get number of characters by summing length of individual words
numChar <- sapply(sapply(byWord[[i]][[1]], str_count), sum)
#average length of words spoken
avgLength <- numChar/numWords
#write into a matrix
wordCount[,1] <- uniqueSpeakers
wordCount[,2] <- numWords
wordCount[,3] <- numChar
wordCount[,4] <- avgLength
#assign column names
colnames(wordCount) <- c("Speaker", "numWords", "numChar", "avgLength")
#store matrix as ith element in list
sumWords[[i]] <- wordCount
}

sumWords

## [[1]]
##      Speaker     numWords numChar avgLength
## [1,] "LEHRER: "  "1217"   "5583"  "4.58751027115859"
## [2,] "CLINTON: " "7375"   "32527" "4.4104406779661"
## [3,] "DOLE: "    "8113"   "35157" "4.33341550597806"
##
## [[2]]
##      Speaker       numWords numChar avgLength
## [1,] "MODERATOR: " "1692"   "7842"  "4.63475177304965"
## [2,] "GORE: "      "7194"   "31521" "4.38156797331109"
## [3,] "BUSH: "      "7436"   "32311" "4.34521247982786"
##
## [[3]]
##      Speaker    numWords numChar avgLength
## [1,] "LEHRER: " "1376"   "6591"  "4.78997093023256"
## [2,] "KERRY: "  "7109"   "30666" "4.31368687579125"
## [3,] "BUSH: "   "6318"   "27484" "4.35011079455524"
##
## [[4]]
##      Speaker    numWords numChar avgLength
## [1,] "LEHRER: " "2784"   "12231" "4.39331896551724"
## [2,] "OBAMA: "  "15224"  "66872" "4.39253809774041"
## [3,] "MCCAIN: " "14260"  "63284" "4.43786816269285"
##
## [[5]]
##      Speaker    numWords numChar avgLength
## [1,] "LEHRER: " "1510"   "6680"  "4.42384105960265"
## [2,] "OBAMA: "  "7293"   "32593" "4.46907993966818"
## [3,] "ROMNEY: " "7303"   "31708" "4.34177735177324"
```

The above results showed that speeches given by both candidates have roughly the same number of words, which is expected since they were given the same amount of time. Also, the average length of words spoken is about 4-5 characters long.

(f) Get the repetitions of each of the 12 words for each speaker and save them in a matrix. Aggregate all data from 5 years of debates in a list.

```
freqWords <- list()

#vectore of words we are intersted in in the form of proper regular expressions
#use \\b to match the letter following it between a word and a non-word character
vecWords <- c("\\bI\\b", "\\bwe\\b", "\\bAmerican?\\b", "\\bdemocra(cy|tic)\\b", "\\brepublic\\b",
              "\\bDemocrat(ic)?\\b", "\\bRepublican\\b", "\\bfree(dom)?\\b", "\\bwar\\b",
              "\\bGod [^b][^l][^e][^s][^s]", "\\bGod Bless\\b", "\\bJesus|Christ|Christian\\b")

for( i in 1:5 ){
  numCount <- matrix(0, nrow = 3, ncol = 13)
  numCount[,1] <- unique(dat[[i]][,1])
  #loop through all words we are interested in
   for(j in 1:length(vecWords)){
     #for each debate and speaker, grep for the locations where a certain word appeared
```

```
      #(returns vector); and compute the length of the vector, which gives the number of appearance
      vecCount <- sapply(sapply(bySentence[[i]][[1]], function(x){grep(vecWords[j], x)}), length)
      #save counts (vector, each entry is from a different speaker) to matrix
      numCount[,j+1] <- vecCount
    }
  colnames(numCount) <- c("Speaker", "I", "we", "America{,n}", "democra{cy,tic}", "republic",
                         "Democrat{,ic}", "Republican", "free{,dom}", "war", "God",
                         "God Bless", "{Jesus, Christ, Christian}")
  #store matrix as ith element in list
  freqWords[[i]] <- numCount
}

freqWords
```

```
## [[1]]
##      Speaker      I     we   America{,n} democra{cy,tic} republic
## [1,] "LEHRER: "   "12"  "7"  "1"         "0"             "0"
## [2,] "CLINTON: "  "171" "84" "29"        "4"             "0"
## [3,] "DOLE: "     "202" "86" "40"        "0"             "0"
##      Democrat{,ic} Republican free{,dom} war God God Bless
## [1,] "1"           "2"        "0"        "0" "0" "0"
## [2,] "1"           "7"        "8"        "2" "0" "0"
## [3,] "6"           "10"       "1"        "1" "0" "0"
##      {Jesus, Christ, Christian}
## [1,] "0"
## [2,] "0"
## [3,] "0"
##
## [[2]]
##      Speaker        I     we   America{,n} democra{cy,tic} republic
## [1,] "MODERATOR: " "13"  "10" "0"         "1"             "0"
## [2,] "GORE: "      "186" "58" "11"        "1"             "0"
## [3,] "BUSH: "      "184" "68" "19"        "1"             "0"
##      Democrat{,ic} Republican free{,dom} war God God Bless
## [1,] "1"           "1"        "0"        "0" "0" "0"
## [2,] "1"           "1"        "1"        "3" "0" "0"
## [3,] "1"           "1"        "2"        "2" "0" "0"
##      {Jesus, Christ, Christian}
## [1,] "0"
## [2,] "0"
## [3,] "0"
##
## [[3]]
##      Speaker    I     we   America{,n} democra{cy,tic} republic
## [1,] "LEHRER: " "7"   "2"  "2"         "1"             "0"
## [2,] "KERRY: "  "154" "95" "42"        "2"             "0"
## [3,] "BUSH: "   "148" "94" "24"        "3"             "0"
##      Democrat{,ic} Republican free{,dom} war  God God Bless
## [1,] "1"           "1"        "0"        "3"  "0" "0"
## [2,] "0"           "1"        "2"        "33" "0" "0"
## [3,] "0"           "0"        "28"       "20" "1" "0"
##      {Jesus, Christ, Christian}
## [1,] "0"
## [2,] "0"
## [3,] "0"
##
## [[4]]
##      Speaker    I     we    America{,n} democra{cy,tic} republic
## [1,] "LEHRER: " "28"  "14"  "0"         "0"             "0"
## [2,] "OBAMA: "  "222" "292" "26"        "2"             "0"
## [3,] "MCCAIN: " "320" "204" "36"        "2"             "0"
##      Democrat{,ic} Republican free{,dom} war  God God Bless
## [1,] "2"           "2"        "0"        "0"  "0" "0"
## [2,] "0"           "4"        "4"        "22" "0" "0"
## [3,] "2"           "4"        "4"        "10" "0" "0"
##      {Jesus, Christ, Christian}
## [1,] "0"
## [2,] "0"
## [3,] "4"
##
## [[5]]
##      Speaker    I     we    America{,n} democra{cy,tic} republic
## [1,] "LEHRER: " "15"  "17"  "1"         "0"             "0"
## [2,] "OBAMA: "  "92"  "109" "17"        "0"             "0"
## [3,] "ROMNEY: " "167" "73"  "27"        "1"             "0"
##      Democrat{,ic} Republican free{,dom} war God God Bless
## [1,] "1"           "1"        "0"        "0" "0" "0"
## [2,] "4"           "5"        "2"        "1" "0" "0"
## [3,] "4"           "5"        "6"        "0" "0" "0"
##      {Jesus, Christ, Christian}
## [1,] "0"
## [2,] "0"
## [3,] "0"
```

Observe from the above result that ”I” is usually the most used word for all speakers, followed by ”we”. Also, ”American/American” appeared in moderate amounts. The frequency of the words ”God”, ”God Bless”, ”Jesus/Christ/Christian” is usually quite low, except for in Mccain’s speech, in which they appeared a total of 4 times. In 2004 and 2008, the ongoing war at the time was one of the important topics. In 2004, President Bush’s speech had an emphasize on freedom.
3.

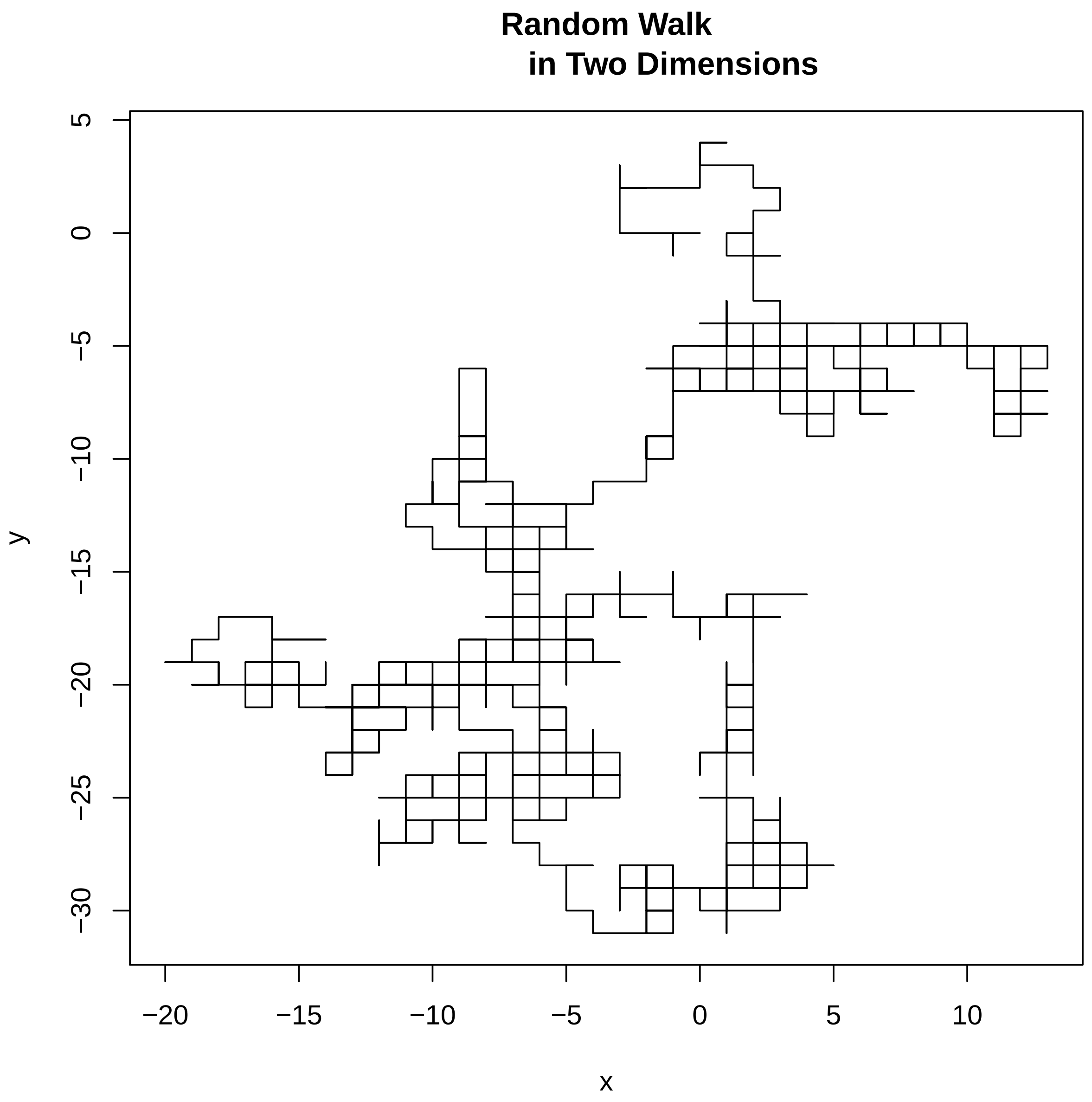
(a)/(b) Function to simulate a random walk in a vectorized fashion. Use multiple assertions to check if the length of walk passed in is valid (i.e. a positive integer). Get 2 samples of n-1-element vectors to denote the direction of the random walk (i.e. N vs S, E vs W). Get x and y locations after each step of the random walk in the form of 2 vectors. If fullpath (optional) argument is set as FALSE, return only the last pair ofcoordinates of the random walk; otherwise, return the whole path.

```
ramWalk <- function( n, fullpath = TRUE ){
  #assertions to make sure the length of walk passed in is a positive integer
  if(n<0){
    stop("Number of steps needs to be positive.")
  }
  if(n==0){
    stop("No steps taken.")
  }
  #check if n is an integer
  #is.integer doesn't always return logical values
  #can also use mod
  if( !isTRUE(n == floor(n)) ){
    stop("Number of steps needs to be an integer")
  }

  #random walk of length n has n-1 steps
  #sample n-1 random values with 1=going forward, -1=going backward
  stepDir <- sample( c(-1,1), n-1, replace=TRUE)
  #sample n-1 random values with TRUE=move horizontally, FALSE=move vertically
  horizontal <- sample( c(TRUE, FALSE), n-1, replace=TRUE)
  #get x and y locations as lists of coordinates
  #ifelse takes value of second argument if condition/test(first argument) is TRUE; else
  #takes value of third argument; if ith step is horizontal, add 1/-1 to xlocation,
  #otherwise add 1/-1 to y location
  #cumsum returns cumulative sum of 1's and -1's
  xlocation <- c(0, cumsum( ifelse(horizontal, stepDir, 0) ))
  ylocation <- c(0, cumsum( ifelse(horizontal, 0, stepDir) ))

  #check if second(optional) argument passed in is false
  if( fullpath == FALSE ){
    #if false, return matrix of last elements of x and y locations in long format
    mat <- t(matrix(c(xlocation[n], ylocation[n]), ncol=2))
    rownames(mat) <- c("x", "y")
    return(mat)
  }
  else{
    #otherwise, return matrix of all elements of x and y locations
    mat <- t(matrix(c(xlocation, ylocation), ncol=2))
    rownames(mat) <- c("x", "y")
    return(mat)
  }
}

ramWalk(6)

##   [,1] [,2] [,3] [,4] [,5] [,6]
## x    0    1    1    1    1    1
## y    0    0   -1   -2   -1    0

ramWalk(6, fullpath = FALSE)

##   [,1]
## x   -2
## y   -1
```

(c) Create the S3 rw class with a constructor; write the class specific print and plot methods and define the class specific [operator that gives the ith position of the walk. Also, create a replacement method called start that translates the origin of the random walk.

```
#constructor; random walk of the length passed in and default starting point (0,0)
#can access the length, path, and starting point of the walk for objects of rw class
rw <- function( lengthWalk = NA, start = c(0,0) ){
  obj <- list( lengthWalk = lengthWalk, path = ramWalk(lengthWalk), start = start )
  class(obj) <- "rw"
  return(obj)
}
```

```
#define new generic print method
print <- function( obj, ... ){
  UseMethod("print")
}

#rw class specific print method; returns final position, length of walk, and starting
#point of walk
print.rw <- function( obj ){
  print.default(c("The final position is: ", obj$path[,obj$lengthWalk] ))
  print.default(c("The length of the walk is ", obj$lengthWalk))
  print.default(c("The starting point is ", obj$start))
}

x <- rw(6)
x$path

##   [,1] [,2] [,3] [,4] [,5] [,6]
## x    0    1    2    1    2    2
## y    0    0    0    0    0   -1

print(x)

##                                                    x
## "The final position is: "                       "2"
##                          y
##                       "-1"
## [1] "The length of the walk is " "6"
## [1] "The starting point is " "0"
## [3] "0"

plot <- function( obj, ... ){
  UseMethod("plot")
}

#rw class specific plot method; plot the path of the walk; set ranges of the x and y axes
#to be the ranges of the x and y coordinates of the path respectively
plot.rw <- function( obj ){
  plot.default(obj$path[1,], obj$path[2,], type="l", xlab="x", ylab="y", main="Random Walk
               in Two Dimensions", xlim=range(obj$path[1,]),ylim=range(obj$path[2,]))
}

x <- rw(1000)
plot(x)
```



```
#rw class specified operator; returns the (number passed in)th row in the path matrix;
#gives the ith position of the walk
`[.rw` <- function( obj, index ){
  position <- obj$path[,index]
}

x[98]

##   x  y
##   0 -7

'start<-' <- function( obj, ...){
  UseMethod("start<-")
}
```

```r
#replacement method; update starting point of walk to value (numeric vector) passed in;
#shift path to that of the walk starts at new starting point
#by default, replacement function passes in the new value as a parameter named value
'start<-.rw' <- function( obj, value ){
 obj$start <- value
 obj$path <- obj$path+value
 return(obj)
}

x <- rw(6)
x$start

## [1] 0 0

x$path

##   [,1] [,2] [,3] [,4] [,5] [,6]
## x    0   -1   -1   -2   -3   -4
## y    0    0    1    1    1    1

start(x) <- c(2,4)
x$start

## [1] 2 4

x$path

##   [,1] [,2] [,3] [,4] [,5] [,6]
## x    2    1    1    0   -1   -2
## y    4    4    5    5    5    5
```