Stat 243 Problem Set 3

Yuchao Guo

SID 26947503

Problem 1

Comments and questions about "Best Practices for scientific computing"
1.Reproducibility
1.1 Comments
Reproducibility is the key idea of the whole paper. As mentioned in the paper, computers are designed to finish those repetitive jobs, we should pay more attention to the reproducibility of our codes. This means we should modularize our codes for the future use. A very good example would be functions. Make those tasks that would appear more than once done by functions would save lots of time from typing same codes over and over again. More importantly, reproducibility would make sure that when multiple users are doing one coding job, others can reproduce the codes and understand what does the person mean by writing down these codes. Therefore, I think I would make more effort in the future to make my own codes more reproducible.
1.2 Questions
I am very concerned about planning how to modularize my own codes. When I am declaring variables, where should I put it? Inside or out side a function? Meanwhile, I also want to make sure what values are good to be inputs of a function and what should be just some variables declared inside the function.
2. Use version control
2.1 Comments
Version control is a very useful method to manage the coding project both for multiple users or single user. The reason is that for multiple users, version control software would make sure the codes will not have any conflicts after editing by different users, as mentioned in the paper, if there exists any conflicts, the software would ask users to make sure which one should be the right version. Users have to edit the file until there is no more conflicts. For a single user, version control system could help with editing on different computers. For example, computers at home, laptop to be carried with, and computers in offices. Version control system could make sure the codes are uniform from version to version.
2.2 Questions
What are the benefits of keeping track of all the history versions?
3. Plan for mistakes
3.1 Comments
It is very important to test the codes for a short time period. Since mistakes are inevitable, it would be better if we plan to see mistakes before it really happens. A practical way is to test the codes often, each time, we could know whether those codes could work. If not, we can edit just those codes that can not work. This could reduce the debugging difficulty.
3.2 Question
For debugging, is there any routine process?

Problem 2

Part a

First, we need to load in some packages using library(), then we can write two functions to select first and all the debates in a time period(1 year). The first function is for the first debate, and the second function is for all debates.

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

```
#This is the funtion to select the first debate
Debate_first<-function(year){
  debateMain<-htmlParse("http://www.debates.org/index.php?page=debate-transcripts")
  debate.transcript<-getNodeSet(debateMain,"//a[@href]")
  matchpoint<-sapply(debate.transcript,xmlValue)
  url<-sapply(debate.transcript, xmlGetAttr, "href")[grep(paste(year,'.*','The First',sep=""),matchpoint
  ##we can grep all the lines with "The First" as the pattern
  return(url)
}


#This is the function to select all debate in a certain year(Most 3 debates, here we select all three d
Debate_all<-function(year){
  debateMain<-htmlParse("http://www.debates.org/index.php?page=debate-transcripts")
  debate.transcript<-getNodeSet(debateMain,"//a[@href]")
  matchpoint<-sapply(debate.transcript,xmlValue)
  url1<-sapply(debate.transcript, xmlGetAttr, "href")[grep(paste(year,'.*','The First',sep=""),matchpoi
  url2<-sapply(debate.transcript, xmlGetAttr, "href")[grep(paste(year,'.*','The Second',sep=""),matchpo
  url3<-sapply(debate.transcript, xmlGetAttr, "href")[grep(paste(year,'.*','The Third',sep=""),matchpoi
  url <- as.list(rbind(url1,url2,url3))
  return(url)
}

url_1996_first <- Debate_first(1996)
url_1996

## Error in eval(expr, envir, enclos):  object 'url_1996' not found
```

Part b

We can use the function getNodeSet to get the node we need, and by using some regular expressions, we can get rid of those strange things we don't need. Finally, we can get a list of texts we need.

```
#This is the function to extract the body
extract_body <- function(year){
  url <- Debate_first(year)
  doc <- htmlParse(url) #read in all the xml file
  text <- getNodeSet(doc, "//p/text()")
  list <- unclass(text)
  for (i in 1:length(list)){
    list[[i]] <- toString.XMLNode(list[i])
    list[[i]] <- str_replace_all(list[[i]], "\\[\\[1\\]\\]","") #remove [[1]] in the text
    list[[i]] <- str_replace_all(list[[i]],"\n","") #remove \n
    list[[i]] <- str_replace_all(list[[i]],"\\\"","") #remove\
  }
  list[1] <- NULL
  return(list)
}

# Call the functions to get all debate text
Debate_1996 <- extract_body(1996)
Debate_2000 <- extract_body(2000)
```

```
Debate_2004 <- extract_body(2004)
Debate_2008 <- extract_body(2008)
Debate_2012 <- extract_body(2012)

head(Debate_1996) #For illustration

## [[1]]
## [1] "LEHRER: Good evening from the Bushnell Theatre in Hartford, Connecticut. I'm Jim Lehrer of the
##
## [[2]]
## [1] "There will be two-minute opening and closing statements. In between, a series of questions, eac
##
## [[3]]
## [1] "Under their rules, the candidates are not allowed to question each other directly. I will ask t
##
## [[4]]
## [1] "The order for everything tonight was determined by coin toss. Now, to the opening statements an
##
## [[5]]
## [1] "CLINTON: Thank you, Jim. And thank you to the people of Hartford, our hosts. "
##
## [[6]]
## [1] "I want to begin by saying again how much I respect Senator Dole and his record of public servic
```

Part c

In this part, I would put all the sentences spoken by one person into a chunk. This is good for the future use of these chunks.

```
#Function to make chunks
Debate_in_chunks<-function(year){
Debate_list<- extract_body(year)
mylist <- list()#Declare an empty list for future use
chunk_vector <- grep("^[A-Z]*:",Debate_list) #Find the indice for capital letters(Names)
chunk_vector <- c(chunk_vector, (length(Debate_list)+1))
for(i in 1:(length(chunk_vector)-1)){
  mylist[[i]] <- paste(unlist(Debate_list[chunk_vector[i]:(chunk_vector[i+1]-1)]), collapse = "") #put
  #### names(mylist[[i]]) <- unlist(str_split(mylist[[i]][1], ":"))[1]
}
mylist <- unlist(mylist)
return(mylist)
}
Debate_in_chunks_1996 <- Debate_in_chunks(1996)
Debate_in_chunks_2000 <- Debate_in_chunks(2000)
Debate_in_chunks_2004 <- Debate_in_chunks(2004)
Debate_in_chunks_2008 <- Debate_in_chunks(2008)
Debate_in_chunks_2012 <- Debate_in_chunks(2012)
head(Debata_in_chunks2000)#for illustration

## Error in head(Debata_in_chunks2000):  object 'Debata_in_chunks2000' not found
```

Then, I will count how many applause and how many laughter are in each year's first debate.

```
#This function would tell us how many laughters and how many applauses
COUNTER <- function(year){
  num_APP<-sum(str_count(unlist(Debate_in_chunks(year)), "\\(APPLAUSE\\)"))
  num_app<-sum(str_count(unlist(Debate_in_chunks(year)), "\\(Applause\\)"))
  num_LAU<-sum(str_count(unlist(Debate_in_chunks(year)),"\\(LAUGHTER\\)"))
  print(paste("There are/is",num_APP+num_app,"APPLAUSES in",year,"Debate", collapse = " "))
  print(paste("There are/is",num_LAU,"LAUGHTERS in",year,"Debate", collapse = " "))
}
COUNTER(1996)

## [1] "There are/is 0 APPLAUSES in 1996 Debate"
## [1] "There are/is 0 LAUGHTERS in 1996 Debate"

COUNTER(2000)

## [1] "There are/is 2 APPLAUSES in 2000 Debate"
## [1] "There are/is 0 LAUGHTERS in 2000 Debate"

COUNTER(2004)

## [1] "There are/is 2 APPLAUSES in 2004 Debate"
## [1] "There are/is 3 LAUGHTERS in 2004 Debate"

COUNTER(2008)

## [1] "There are/is 4 APPLAUSES in 2008 Debate"
## [1] "There are/is 4 LAUGHTERS in 2008 Debate"

COUNTER(2012)

## [1] "There are/is 1 APPLAUSES in 2012 Debate"
## [1] "There are/is 4 LAUGHTERS in 2012 Debate"
```

After counting all these things, we can delelte them to make the text nicer formated.

```
#This funciton would clean all applause and laughters
#We can use str_replace to replace all applause and laughters
Clean<-function(year){
  cleanchunk <- str_replace(Debate_in_chunks(year), "\\(APPLAUSE\\)","")
  cleanchunk <- str_replace(Debate_in_chunks(year), "\\(Applause\\)","")
  cleanchunk <- str_replace(Debate_in_chunks(year), "\\(LAUGHTER\\)","")
  return (cleanchunk)
}
Clean_chunk_1996 <- Clean(1996)
Clean_chunk_2000 <- Clean(2000)
Clean_chunk_2004 <- Clean(2004)
Clean_chunk_2008 <- Clean(2008)
Clean_chunk_2012 <- Clean(2012)
head(Clean_chunk_2008)# illustration

## [1] "SPEAKERS: U.S. SENATOR JOHN MCCAIN (AZ) REPUBLICAN PRESIDENTIAL NOMINEE U. S. SENATOR BARACK OBA
## [2] "OBAMA: Well, thank you very much, Jim, and thanks to the commission and the University of Missis
## [3] "LEHRER: Senator McCain, two minutes. "
## [4] "MCCAIN: Well, thank you, Jim. And thanks to everybody. And I do have a sad note tonight. Senato
## [5] "LEHRER: All right, let's go back to my question. How do you all stand on the recovery plan? And
## [6] "OBAMA: We haven't seen the language yet. And I do think that there's constructive work being do
```

Part d

First, we split all the words, put them into a vector, each element is a word, then we put each sentence into a vector and each element is a sentence.

```
#This is the function to split words
word_split <- function(year) {
  vector <- str_split(Clean(year)," ")#split word by the pattern " "
  vector <- unlist(vector)
  return(vector)
}

word_1996 <- word_split(1996) # a vector, each word an element for 1996
word_2000 <- word_split(2000) # a vector, each word an element for 2000
word_2004 <- word_split(2004) # a vector, each word an element for 2004
word_2008 <- word_split(2008) # a vector, each word an element for 2008
word_2012 <- word_split(2012) # a vector, each word an element for 2012
head(word_2012) # illustration

## [1] "SPEAKERS:" "FORMER"    "GOV."      "MITT"      "ROMNEY,"   "R-MASS."

#This function would combine all the word in one sentence.
sentence_builder<-function(year){
  words <- word_split(year)
  vector1 <- grep((".*[a-z][a-z]\\."), words)#find position for.
  vector2 <- grep((".*[a-z][a-z]\\?"), words)#find postition for ?
  vector3 <- grep((".*[a-z][a-z]\\!"), words)#find position for !
  index <- unique(sort(c(vector1,vector2,vector3)))
  sentence <- c()
  sentence[1] <- paste(words[1:index[1]], collapse = " ")
  for (i in 2:length(index)-1){
    sentence[i] <- paste(words[(index[i]+1):index[i+1]], collapse = " ")
  }
  return(sentence)
}

sentence_1996 <- sentence_builder(1996)
sentence_2000 <- sentence_builder(2000)
sentence_2004 <- sentence_builder(2004)
sentence_2008 <- sentence_builder(2008)
sentence_2012 <- sentence_builder(2012)

head(sentence_2008) #illustration

## [1] "I'm Jim Lehrer of the NewsHour on PBS, and I welcome you to the first of the 2008 presidential
## [2] "The Commission on Presidential Debates is the sponsor of this event and the three other preside
## [3] "Tonight's will primarily be about foreign policy and national security, which, by definition, i
## [4] "It will be divided roughly into nine-minute segments."
## [5] "Direct exchanges between the candidates and moderator follow-ups are permitted after each candi
## [6] "The specific subjects and questions were chosen by me."
```

Part e

First, I claimed a dataframe, and then I wrote two functions to find the number of words for each candidate in each year, another to find the number of characters for each candidates in each year.

Then we can calculate the average length of each word. Put all together, we can find some trends.

```
#First I would declare an data.frame to store all the information
info <- as.data.frame(matrix(NA, nrow = 10, ncol =3), row.names = c("1996 CLINTON","1996 DOLE", "2000 BU
                                                                    "2000 GORE", "2004 BUSH", "2004 KERR
                                                                    "2008 OBAMA","2008 MCCAIN", "2012 OB
                                                                    "2012 ROMNEY"))
colnames(info)<-(c("number of words","number of characters","average word length"))

#This
Num_words <- function(name, year){
  chunks <- Clean(year)
  index <- grep(paste0("^",name,collapse = ""), chunks)
  chunk_sub <- unlist(chunks[index])
  chunk_sub <- str_split(chunk_sub, " ")
  num <- length(unlist(chunk_sub))
}

CLINTON_1996_nw <- Num_words("CLINTON", 1996)
DOLE_1996_nw <- Num_words("DOLE", 1996)
BUSH_2000_nw <- Num_words("BUSH", 2000)
GORE_2000_nw <- Num_words("GORE", 2000)
BUSH_2004_nw <- Num_words("BUSH", 2004)
KERRY_2004_nw <- Num_words("KERRY", 2004)
OBAMA_2008_nw <- Num_words("OBAMA", 2008)
MCCAIN_2008_nw <- Num_words("MCCAIN", 2008)
OBAMA_2012_nw <- Num_words("OBAMA", 2012)
ROMNEY_2012_nw <- Num_words("ROMNEY", 2012)

number_of_words <- c(CLINTON_1996_nw, DOLE_1996_nw, BUSH_2000_nw, GORE_2000_nw, BUSH_2004_nw, KERRY_2004
        OBAMA_2008_nw, MCCAIN_2008_nw, OBAMA_2012_nw, ROMNEY_2012_nw)

Num_chara <- function(name, year){
  chunks <- Clean(year)
  index <- grep(name, chunks)
  chunk_sub <- unlist(chunks[index])
  num <- sum(str_count(unlist(chunk_sub)))
}

CLINTON_1996_nc <- Num_chara("CLINTON", 1996)
DOLE_1996_nc <- Num_chara("DOLE", 1996)
BUSH_2000_nc <- Num_chara("BUSH", 2000)
GORE_2000_nc <- Num_chara("GORE", 2000)
BUSH_2004_nc <- Num_chara("BUSH", 2004)
KERRY_2004_nc <- Num_chara("KERRY", 2004)
OBAMA_2008_nc <- Num_chara("OBAMA", 2008)
MCCAIN_2008_nc <- Num_chara("MCCAIN", 2008)
OBAMA_2012_nc <- Num_chara("OBAMA", 2012)
ROMNEY_2012_nc <- Num_chara("ROMNEY", 2012)

number_of_characters <- c(CLINTON_1996_nc, DOLE_1996_nc, BUSH_2000_nc, GORE_2000_nc, BUSH_2004_nc,KERRY_
                          OBAMA_2008_nc, MCCAIN_2008_nc, OBAMA_2012_nc, ROMNEY_2012_nc)

CLINTON_1996_av <- CLINTON_1996_nc/CLINTON_1996_nw
```

```
DOLE_1996_av <- DOLE_1996_nc/DOLE_1996_nw
BUSH_2000_av <- BUSH_2000_nc/BUSH_2000_nw
GORE_2000_av <- GORE_2000_nc/GORE_2000_nw
BUSH_2004_av <- BUSH_2004_nc/BUSH_2004_nw
KERRY_2004_av <- KERRY_2004_nc/KERRY_2004_nw
OBAMA_2008_av <- OBAMA_2008_nc/OBAMA_2008_nw
MCCAIN_2008_av <- MCCAIN_2008_nc/MCCAIN_2008_nw
OBAMA_2012_av <- OBAMA_2012_nc/OBAMA_2012_nw
ROMNEY_2012_av <- ROMNEY_2012_nc/ROMNEY_2012_nw

number_of_average <- c(CLINTON_1996_av, DOLE_1996_av, BUSH_2000_av, GORE_2000_av, BUSH_2004_av, KERRY_2
                       OBAMA_2008_av, MCCAIN_2008_av, OBAMA_2012_av, ROMNEY_2012_av)

info[,1] <- number_of_words
info[,2] <- number_of_characters
info[,3] <- number_of_average

print(info)
```

```
##              number of words number of characters average word length
## 1996 CLINTON            7468                41253            5.523969
## 1996 DOLE               8217                44769            5.448339
## 2000 BUSH               7563                40909            5.409097
## 2000 GORE               7306                39812            5.449220
## 2004 BUSH               6427                35134            5.466625
## 2004 KERRY              6813                39265            5.763247
## 2008 OBAMA             15610                88846            5.691608
## 2008 MCCAIN            14644                84334            5.758946
## 2012 OBAMA              7150                41574            5.814545
## 2012 ROMNEY             8049                43916            5.456082
```

1.From the result, we can see that in 2008, the number of words are much more than other years, however, by inspect the web page, we can see that in 2008 the transcript was recoded twice for some reason

2.Obama, Kerry, Mccain havae longer average word leghth compared with other candidates.

3.Bush seems to have the shortest average word length.

Part f

In this part, I used a dataframe to capture all the information.

```
#This is the function to translate name into year
name_to_year <- function(name){
  translator <- c("CLINTON 1996"=1996,"DOLE 1996"=1996, "BUSH 2000"=2000, "GORE 2000"=2000, "BUSH 2004"=
                  "OBAMA 2008"=2008, "MCCAIN 2008"=2008,"OBAMA 2012"=2012, "ROMNEY 2012"=2012)
  year <- translator[name]
  return(year)
}

#Here, I declare a new dataframe to store information
name_vector <- c("CLINTON 1996","DOLE 1996", "BUSH 2000", "GORE 2000", "BUSH 2004", "KERRY 2004",
```

```
                "OBAMA 2008", "MCCAIN 2008","OBAMA 2012", "ROMNEY 2012")
exp_vector <- c("I ", "We "," we ","America(n){0,1}"," democra(cy|tic){0,1} "," republic ","Democrat(ic)
               "Republican ", " free(dom){0,1} "," war ","God"," God bless ","(Jesus|Christ|Christian)"
count_table <- as.data.frame(matrix(NA, nrow = length(name_vector), ncol = length(exp_vector)))
rownames(count_table) <- name_vector
colnames(count_table) <-exp_vector

fill_in_table<- function(){
  for(i in 1:length(name_vector)){
    year <- as.numeric(name_to_year(name_vector[i]))
    chunks <- Clean(year)
    index <- grep(unlist(str_split(name_vector[i]," "))[1], chunks)
    chunk_sub <- chunks[index]
    for (j in 1:length(exp_vector)){
      num <- sum(str_count(chunk_sub, exp_vector[j]))
      count_table[i,j] <- num
    }
  }
  return(count_table)
}

count_table <- fill_in_table()
print(count_table)
```

```
##               I  We   we  America(n){0,1}  democra(cy|tic){0,1}
## CLINTON 1996 203  33   77              36                     2
## DOLE 1996    216  38   78              50                     0
## BUSH 2000    171  21   64              26                     1
## GORE 2000    195  19   61              16                     1
## BUSH 2004    151  24   89              24                     1
## KERRY 2004   145  33   92              46                     1
## OBAMA 2008   236  82  340              32                     0
## MCCAIN 2008  336  46  228              48                     0
## OBAMA 2012    92   7  110              24                     0
## ROMNEY 2012  148  19   71              41                     0
##               republic  Democrat(ic){0,1} Republican   free(dom){0,1}
## CLINTON 1996         0                  1          7                2
## DOLE 1996            0                 12         10                0
## BUSH 2000            0                 12          1                3
## GORE 2000            0                  2          1                1
## BUSH 2004            0                  0          0               21
## KERRY 2004           0                  0          1                1
## OBAMA 2008           0                  2          6                4
## MCCAIN 2008          0                  8          6                4
## OBAMA 2012           0                  8          4                2
## ROMNEY 2012          0                  7          5                7
##               war  God  God bless  (Jesus|Christ|Christian)
## CLINTON 1996    2   0           0                         0
## DOLE 1996       0   1           1                         0
## BUSH 2000       4   0           0                         0
## GORE 2000       3   0           0                         0
## BUSH 2004      20   1           0                         0
## KERRY 2004     26   1           1                         0
## OBAMA 2008     20   0           0                         0
```

```
## MCCAIN 2008      6   0          0                       4
## OBAMA 2012       2   0          0                       0
## ROMNEY 2012      0   0          0                       0
```

From the table, we can see that in 2004 and 2008 the word "war" was mentioned multiple times for some reason. And we can tell that Obama and Kerry don't like to use I and we compared with other presidents.

Problem 3

Part a b

I wrote some warning message to tell the user that what problem the input has if it has some. Then, once the user put the right input, we can run into the part to do some random walk. The methods are in the comments inside the function.
Basically, I first get some random indice and use the random indice to get some random step, by these two randomness both equal to 0.5, I can get a random direction with probability equals to 0.25

```
#in part a, I have already used vectorization method

myWalk <- function(n, final){
  if (n < 0){
    print("Can't take negative move")
  }else if(n == 0){
    print("Can't take 0 steps")
  }else if(!is.integer(n)){
    print("Must take an integer value, remember to add an L after the number to tell r this is an integ
  }else if(!is.logical(final)){
    print("final must be TRUE or FALSE")
  }else{
    path <- matrix(0, ncol = 2, nrow = n)
    # generate the indices to set the deltas
    indx <- cbind(seq(n), sample(c(1, 2), n, TRUE))
    # now set the values best on the random index we got
    path[indx] <- sample(c(-1, 1), n, TRUE)
    rwdetail <- matrix(0, ncol=2, nrow = n)
    rwdetail[,1] <- cumsum(path[,1])
    rwdetail[,2] <- cumsum(path[,2])
  #return the value, if final is TRUE, return the final position, otherwise return a matrix containing
  if(final){
    finalposition <- c(sum(path[,1]),sum(path[,2]))
      return(finalposition)
    }else{
      return (rwdetail)
    }
  }
}
RandomWalk <- myWalk(100,FALSE)

## [1] "Must take an integer value, remember to add an L after the number to tell r this is an integer"

RandomWalk2 <- myWalk(-12,FALSE)

## [1] "Can't take negative move"

RandomWalk3 <- myWalk(12L,FALSE)
```

Part c

First, I wrote a constructor which will return a list of a matrix for detail positions and a vector for ending point.

```
#This is the constructor function
myWalk_constructor <- function(n){
    if (n < 0){
      print("Can't take negative move")
    }else if(n == 0){
      print("Can't take 0 steps")
    }else if(!is.integer(n)){
      print("Must take an integer value, remember to add an L after the number to tell r this is an int
    }else{
    rw <- matrix(0, ncol = 2, nrow = n)
    # generate the indices to set the deltas
    indx <- cbind(seq(n), sample(c(1, 2), n, TRUE))
    # now set the values best on the random index we got
    rw[indx] <- sample(c(-1, 1), n, TRUE)
    rwdetail <- matrix(0, ncol=2, nrow = n)
    rwdetail[,1] <- cumsum(rw[,1])
    rwdetail[,2] <- cumsum(rw[,2])
    my_list <- list(rwpath = rwdetail, endpoint=c(sum(rw[,1]),sum(rw[,2])))
    class(my_list) <- 'rw'
    return(my_list) # This my_list contains both the path and ending point
   }
}
random <- myWalk_constructor(20L)
print(random)
```

```
## $rwpath
##       [,1] [,2]
##  [1,]    1    0
##  [2,]    1    1
##  [3,]    2    1
##  [4,]    2    2
##  [5,]    3    2
##  [6,]    2    2
##  [7,]    1    2
##  [8,]    2    2
##  [9,]    1    2
## [10,]    2    2
## [11,]    2    3
## [12,]    3    3
## [13,]    3    2
## [14,]    2    2
## [15,]    1    2
## [16,]    1    3
## [17,]    1    2
## [18,]    1    3
## [19,]    2    3
## [20,]    2    2
##
## $endpoint
## [1] 2 2
```

```
##
## attr(,"class")
## [1] "rw"
```

Then, I use S3 class to finish the following work.
First, it is the print method.

```
print.rw <- function(rwdetail,option=FALSE){
  if (!is.logical(option)){
    print("Option can only be TRUE or FALSE")
  }else{
    if(!option){
      my_matrix <- rwdetail[[1]]
      colnames(my_matrix) <- c("x","y")
      print(my_matrix)
      print("This is a matrix for position after each move")
    }else{
      print(c(rwdetail[[2]]))
      print("This is the ending point")
    }
    }
}
print(random)
```

```
##       x y
##  [1,] 1 0
##  [2,] 1 1
##  [3,] 2 1
##  [4,] 2 2
##  [5,] 3 2
##  [6,] 2 2
##  [7,] 1 2
##  [8,] 2 2
##  [9,] 1 2
## [10,] 2 2
## [11,] 2 3
## [12,] 3 3
## [13,] 3 2
## [14,] 2 2
## [15,] 1 2
## [16,] 1 3
## [17,] 1 2
## [18,] 1 3
## [19,] 2 3
## [20,] 2 2
## [1] "This is a matrix for position after each move"
```

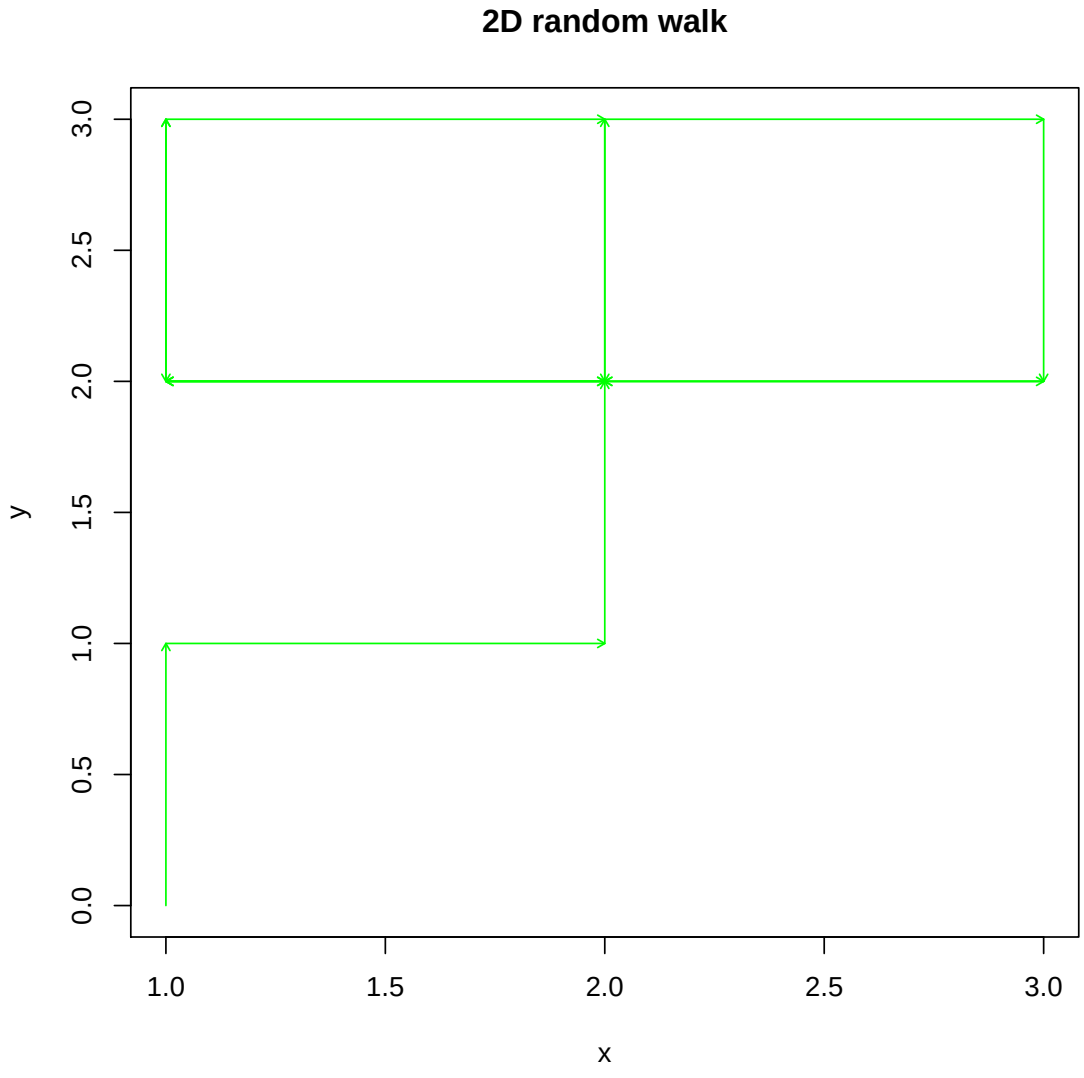
```
print(random, TRUE)
```

```
## [1] 2 2
## [1] "This is the ending point"
```

Then, the plot method.

```r
#plot method
plot <- function(rwdetail, ...){
  UseMethod('plot')
}
plot.rw <- function(rwdetail){
  matrix_path <- rwdetail[[1]]
  plength=length(matrix_path[,1])
  plot(matrix_path[,1],matrix_path[,2],xlab='x', ylab='y',type='n',
       main="2D random walk")
  arrows(matrix_path[1:(plength-1),1],matrix_path[1:(plength-1),2],
         matrix_path[2:(plength),1],matrix_path[2:(plength),2],
         col = "green",length=0.05)
}

plot(random)
```



"[" operator method

```
######[ operator
`[.rw` <- function(rwdetail, ...) UseMethod('[.rw')
`[.rw` <- function(rwdetail,n){
  matrix_path <- rwdetail[[1]]
  print(paste0("the ",n,"th step is at"))
  print(unique(matrix_path[n,]))
}
random[5]

## [1] "the 5th step is at"
## [1] 3 2
## [1] 3 2
```

Lastly, replace method.

```
#####replace
`origin<-` <- function(rwdetial, ...) UseMethod("origin<-")
`origin<-.rw` <- function(rwdetail, value){
  matrix_path <- rwdetail[[1]]
  matrix_path[,1]=matrix_path[,1]+value[1]
  matrix_path[,2]=matrix_path[,2]+value[2]
  print("here is the path matrix for your random walk after changing the origin point")
  print(matrix_path)
}

origin(random)<-c(1,2)

## [1] "here is the path matrix for your random walk after changing the origin point"
##       [,1] [,2]
##  [1,]    2    2
##  [2,]    2    3
##  [3,]    3    3
##  [4,]    3    4
##  [5,]    4    4
##  [6,]    3    4
##  [7,]    2    4
##  [8,]    3    4
##  [9,]    2    4
## [10,]    3    4
## [11,]    3    5
## [12,]    4    5
## [13,]    4    4
## [14,]    3    4
## [15,]    2    4
## [16,]    2    5
## [17,]    2    4
## [18,]    2    5
## [19,]    3    5
## [20,]    3    4
```