Problem Set 3

Thibault Doutre, ID : 26980469

STAT 243 : Introduction to Statistical Computing

I worked on my own.

1 Problem 1

Wilson et.al

About section 7, Defensive programming :

In R, how do we add assertion to programs ? Should we use the "print" function ? I am afraid that it will slow down the program if I use it in a for loop for example.

About section 10, Collaborate :

Is pair programming widely used in the industry ?

2 Problem 2

2.1 Extract URL

First, I load the necessary databases and store the link provided.

```
library(XML)
library(stringr)
link="http://www.debates.org/index.php?page=debate-transcripts"
```

Then, I create a function which takes the year of the debate and gives us:

- The URL of the first debate
- The names of the candidates

```
extract_URL_debate = function(year){
  #Load HTML as a tree (with nodes)
  doc=htmlTreeParse(link)
  #Extract body of the doc
  root = xmlRoot(doc)
  body= xmlChildren(root)$body
  #Find the four debates of every year : Each element of the list
  #corresponds to different year
  debates_years=xpathApply(body, "//div/..//blockquote")
  #Map the position of the list with the corresponding year
```

```
  code_year=(2012-year)/4+1
  #Find the 4 debates for the required year
  debates_year=xpathApply(debates_years[[code_year]],"//a")
  #Find description of each debate
  description=sapply(debates_year, xmlValue)
  #Find the position of the First debate
  no_debate=grep("First",description)
  #Extract president names from description
  candidates_value=xmlValue(debates_year[[no_debate]])
  duo=str_extract(as.character(candidates_value),
                  "[a-zA-z]+\\-[a-zA-z]+")
  candidates=str_split(duo,"-")
  #Extract and return URL
  return (list(URL=xmlGetAttr(debates_year[[no_debate]],"href"),
               Names=toupper(candidates[[1]])))
}
```

For example, for the debate of 2012, we have :

```
extract_URL_debate(2012)


## $URL
## [1] "http://www.debates.org/index.php?page=october
    -3-2012-debate-transcript"
##
## $Names
## [1] "OBAMA"  "ROMNEY"
```

2.2 Extract content of debate

Now, I create a function which returns the text of the debate of a specified year. More precisely, this function returns:

- The text on the good format
- The lines of the text, which correspond to paragraphs in the website
- The names of the candidates

This functions uses the previous one.

```
extract_text_debate = function (year){
  #Extract url and names of candidates
  URL=extract_URL_debate(year)$URL
  names=extract_URL_debate(year)$Names
  #Exctract body
```

```
  doc=htmlParse(URL,isURL=T)
  root = xmlRoot(doc)
  body= xmlChildren(root)$body
  #Extract lines from body
  lines1=xpathApply(body, "//div[@id='content-sm']/p/text()"
                    , xmlValue)
  ##Cleannig data...
  lines2=unlist(lines1, recursive=FALSE)
  #Get start of speech
  start=which.max(str_detect(lines2,"[A-Z]+: [A-Z][a-z]"))
  #Get end of speech
  end=which.max(str_detect(lines2,"(|END)"))-1
  if (end==0) #If no /END at the end of the text
    end =length(lines2)
  lines3=lines2[start:end]
  return(list(Text=paste(lines3,sep="",collapse = "\n\n"),
              Lines=lines3,Names=names))
}
```

Here, we have to be really careful about what we want to extract from the URL. Indeed, for the year 2008, the debate appears twice on the web page ! So we have to stop when it ends, which is when the FIRST "END" appears. But sometimes - year 2004 for instance - there is no such labels. So I end up the text at the end of the page which is indeed the end of the debate as well.

When computing the first and the last four lines for the 2008 debate, we have:

```
lines=extract_text_debate(2008)$Lines
lines[1:3]


## [1] "[*] LEHRER: Good evening from the Ford Center for
     the Performing Arts at the University of Mississippi
    in Oxford. I'm Jim Lehrer of the NewsHour on PBS, and
    I welcome you to the first of the 2008 presidential
    debates between the Republican nominee, Senator John
    McCain of Arizona, and the Democratic nominee, Senator
     Barack Obama of Illinois."
## [2] "The Commission on Presidential Debates is the
    sponsor of this event and the three other presidential
     and vice presidential debates coming in October."
## [3] "Tonight's will primarily be about foreign policy
    and national security, which, by definition, includes
    the global financial crisis. It will be divided
    roughly into nine-minute segments."
```

```
lines[(length(lines)-3):length(lines)]
```

```
## [1] "LEHRER: And that ends this debate tonight."
## [2] "On October 2nd, next Thursday, also at 9:00 p.m.
    Eastern time, the two vice presidential candidates
    will debate at Washington University in St. Louis. My
    PBS colleague, Gwen Ifill, will be the moderator."
## [3] "For now, from Oxford, Mississippi, thank you,
    senators, both. I'm Jim Lehrer. Thank you, and good
    night."
## [4] "(APPLAUSE)"
```

2.3 Extract words and sentences

Sentences I then create a list of sentences using a regular expression which matches sentences without words in capital letters which are not part of spoken sentences.

```
extract_sentences = function(year){
  text=extract_text_debate(year)$Text
  clean_text=gsub('([A-Z]+: |\\([A-Z]+\\))', '',text)
  return(str_extract_all(clean_text, "[A-Z][^.!?]+[.!?]")[[1]])
}
extract_sentences(2008)[1:10]
```

```
##  [1] "Good evening from the Ford Center for the
    Performing Arts at the University of Mississippi in
    Oxford."
##  [2] "I'm Jim Lehrer of the NewsHour on PBS, and I
    welcome you to the first of the 2008 presidential
    debates between the Republican nominee, Senator John
    McCain of Arizona, and the Democratic nominee, Senator
     Barack Obama of Illinois."
##  [3] "The Commission on Presidential Debates is the
    sponsor of this event and the three other presidential
     and vice presidential debates coming in October."
##  [4] "Tonight's will primarily be about foreign policy
     and national security, which, by definition, includes
     the global financial crisis."
##  [5] "It will be divided roughly into nine-minute
    segments."
##  [6] "Direct exchanges between the candidates and
    moderator follow-ups are permitted after each
```

```
    candidate has two minutes to answer the lead question
    in an order determined by a coin toss."
##  [7] "The specific subjects and questions were chosen
    by me."
##  [8] "They have not been shared or cleared with anyone
    ."
##  [9] "The audience here in the hall has promised to
    remain silent, no cheers, no applause, no noise of any
     kind, except right now, as we welcome Senators Obama
    and McCain."
## [10] "Let me begin with something General Eisenhower
    said in his 1952 presidential campaign."
```

Words As for words, I create a similar function with a different regular expression which matches words. Cleanning the text assures me not to extract non verbal words.

```
extract_words = function(year){
  text=extract_text_debate(year)$Text
  clean_text=gsub('([A-Z]+: |\\([A-Z]+\\))', '',text)
  return(str_extract_all(clean_text, "[A-Za-z]+")[[1]])
}
extract_words(2008)[1:10]

##  [1] "Good"       "evening"    "from"       "the"
##  [5] "Ford"       "Center"     "for"        "the"
##  [9] "Performing" "Arts"
```

2.4 Extract speeches

Now, I extract the speeches for every candidates and the speaker, given the year of the debate. The functions returns a list of four elements:

- The speech of candidate1
- The speech of candidate2
- The speech of the moderator
- The names of candidate1, candidate2, moderator

```
extract_speeches = function(year){
  #Extract debate
```

```
  debate=extract_text_debate(year)
  lines=debate$Lines
  names=debate$Names
  #Identify position of speaker indicators
  speaker1=str_detect(lines,names[1])
  speaker2=str_detect(lines,names[2])
  moderator=str_detect(lines,"[A-Z]:")-speaker1-speaker2
  #Assign a unique number per speaker for each line
  state=speaker1*1+speaker2*2+moderator*3
  #Replace the zeros by the labels preceding them
  #in order to have each line labeled by a speaker
  j=state[1]
  for (i in 1:length(lines)){
    if (state[i]==0)
      state[i]=j
    else
      j=state[i]
  }
  ###List of speeches labeled by name of the speakers
  ##For each speaker:
  #Merge consecutive text with same labels
  #Remove names of the speakers
  #Remove non verbal indicators
  list_speech=sapply(c(1,2,3),function(x)
    gsub('([A-Z]+: |\\([A-Z]+\\) )', '',
         paste(lines[state==x],collapse=" ",sep="")))
  names(list_speech)=c(names[1],names[2],"MODERATOR")

  return(list_speech)
}
```

For example, I can display the names of the speakers in 2004:

```
names(extract_speeches(2004))

## [1] "BUSH"      "KERRY"     "MODERATOR"
```

2.5 Extract number of words, characters and average word length

For a single year Once I have the data stored in R, I can process some basic statistics in order to get some information from the text. I create a function which takes a year as an argument and returns a data frame which contains the number of words, the number of characters and the average word length for each

speaker in a given year.// I use stringr library to have the number of words and to have the number of letters of these words, and to compute the total number of characters. Once I have all this information I can easily compute the average word length which is the number of characters used in words divided by the number of words.

```
extract_count_year = function(year){
  speeches= extract_speeches(year)
  #Initialize data frame
  data=data.frame(date=rep(year,3),speaker=names(speeches),
                  n_words=rep(0,3),n_characters=rep(0,3),
                  average_word_length=rep(0,3))
  #Count word characters
  characters_words=rep(0,3)
  for (i in 1:3){
    sp=speeches[[i]]
    #number of words
    data[i,3]=str_count(sp,"[a-zA-Z]+")
    #number of characters
    data[i,4]=str_length(sp)
    #number of word characters
    characters_words[i]=str_count(sp,"[a-zA-Z]")
  }
  #Average length
  data[,5]=(characters_words)/(data[,3])
  return (data)
}
```

For the year 1996, this is the result of this function:

```
extract_count_year(1996)
```

```
##   date   speaker n_words n_characters
    average_word_length
## 1 1996   CLINTON    7577        40702
    4.231358
## 2 1996      DOLE    8445        44414
    4.072232
## 3 1996 MODERATOR     949         5571
    4.642782
```

For every years In order to have the average word length for each candidate, for every years, we have to merge the data frames resulting of the previous

function. To do so, I use the aggregate function and use the fact that the mean of the average word length for every speech in a given year is the average word length of the concatenated speeches for every candidate.

```
#Merge lists like rbind, but in a faster way
extract_count = function(y){
  n=length(y)
  data=vector("list", n)
  for (i in 1:n){
    data[[i]]=extract_count_year(y[i])
  }
  return(Reduce(function(x, y) merge(x, y, all=TRUE), data))
}
#Statistics for every person, by people
y=seq(1996,2012,by=4)
data=extract_count(y)
#Average word length over all speeches
data=aggregate(average_word_length ~ speaker, data = data, mean)
#Ordering the data by average word length
data[order(-data[,2]),]

##      speaker average_word_length
## 3  MODERATOR            4.475649
## 5       GORE            4.280723
## 7     MCCAIN            4.257162
## 1    CLINTON            4.231358
## 8      OBAMA            4.228816
## 4       BUSH            4.157863
## 6      KERRY            4.119239
## 9     ROMNEY            4.110807
## 2       DOLE            4.072232
```

2.6 Count number of word occurrence

For a single year In order to count the number of word occurrence in the text, for each candidate, for a particular year, I create a function which takes into argument a vector of regular expression and returns the number of matching occurrences. I will not expand myself on this function since it is almost the same as the previous one, excepted the fact that I use a for loop which corresponds to the vector of regular expressions.

```
extract_regexpr_year = function(year,regexpr){
  speeches= extract_speeches(year)
```

```
  data=data.frame(rep(year,3),names(speeches),
                  matrix(rep(0,length(regexpr)*3),nrow=3))
  names(data)=c("date","speaker",regexpr)
  for (i in 1:3){
    sp=speeches[[i]]
    for (r in 1:length(regexpr)){
      expr=regexpr[r]
      data[i,r+2]=str_count(sp,expr)
    }
  }
  return (data)
}
```

Here is the result of the function for the mentioned words:

```
regexpr=c("I","we","America[n?]","democra(cy|tic)","republic",
          "Democrat(ic|[^a-zA-Z])","Republican",
          "free(dom|[^a-zA-Z])","war","God [^bB]","God [Bb]less"
          ,"(Jesus|Christ|Christian)")

extract_regexpr_year(1996,regexpr)

##   date   speaker   I  we America[n?] democra(cy|tic)
## 1 1996   CLINTON 275 174          18               5
## 2 1996      DOLE 346 155          19               0
## 3 1996 MODERATOR  12  16           0               0
##   republic Democrat(ic|[^a-zA-Z]) Republican
## 1        0                      1         10
## 2        0                      7         12
## 3        0                      1          2
##   free(dom|[^a-zA-Z]) war God [^bB] God [Bb]less
## 1                   8  11         0            0
## 2                   1   4         0            1
## 3                   0   2         0            0
##   (Jesus|Christ|Christian)
## 1                        0
## 2                        0
## 3                        0
```

For every years I cannot proceed like the previous section since a candidate like Obama has done 2 elections and a candidate like McCain has only done one. So basically, I store the data corresponding to a speaker at a particular date.

```r
#merge lists in a faster way than rbind
extract_regexpr = function(y,regexpr){
  n=length(y)
  data=vector("list", n)
  for (i in 1:n){
    data[[i]]=extract_regexpr_year(y[i],regexpr)
  }
  return(Reduce(function(x, y) merge(x, y, all=TRUE), data))
}

extract_regexpr(y,regexpr)
```

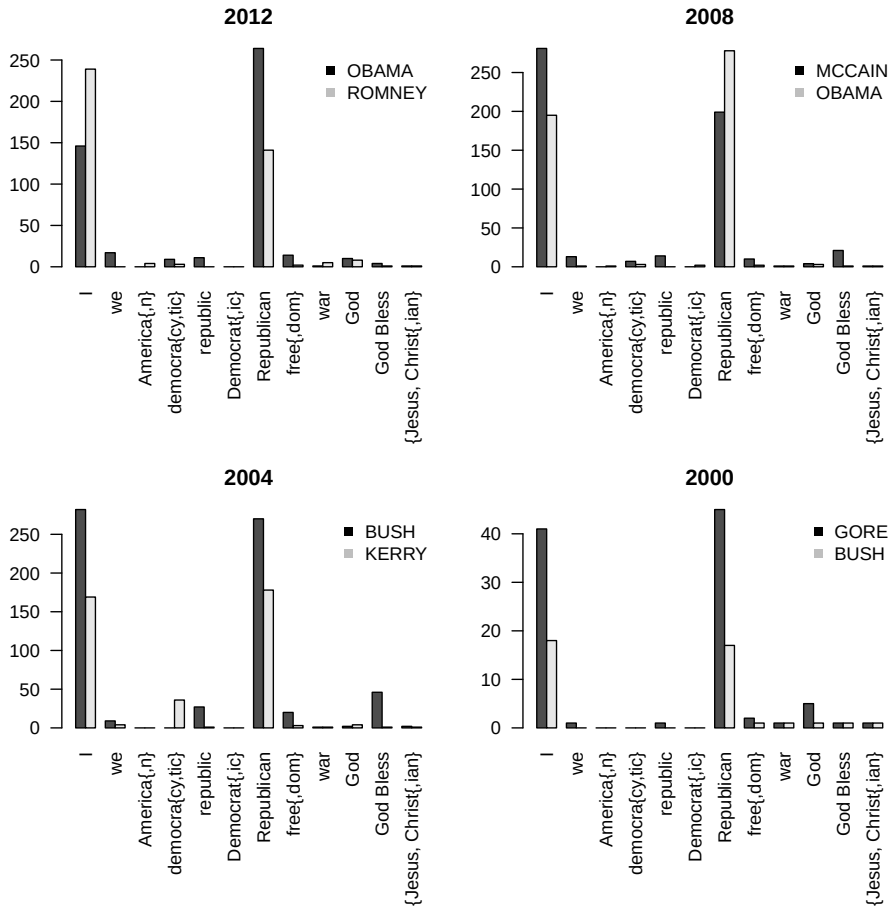
```
##     date    speaker   I  we America[n?] democra(cy|tic)
## 1   1996    CLINTON 275 174          18               5
## 2   1996       DOLE 346 155          19               0
## 3   1996 MODERATOR  12  16           0               0
## 4   2000 MODERATOR   7  10           0               0
## 5   2000       BUSH  44  16           1               0
## 6   2000       GORE  41  18           1               0
## 7   2004 MODERATOR  22  13           3               1
## 8   2004       BUSH 282 169           9               4
## 9   2004      KERRY 269 177          19               2
## 10  2008 MODERATOR  27  18           0               0
## 11  2008     MCCAIN 281 195          13               1
## 12  2008      OBAMA 198 277           9               1
## 13  2012 MODERATOR  27  39           3               0
## 14  2012      OBAMA 146 239          17               0
## 15  2012     ROMNEY 263 140          13               1
##     republic Democrat(ic|[^a-zA-Z]) Republican
## 1          0                      1         10
## 2          0                      7         12
## 3          0                      1          2
## 4          0                      1          1
## 5          0                      0          4
## 6          0                      0          0
## 7          0                      1          1
## 8          0                      0          0
## 9          0                      0          1
## 10         0                      1          1
## 11         0                      1          7
## 12         0                      0          3
## 13         0                      1          1
## 14         0                      4          9
## 15         0                      4          9
##     free(dom|[^a-zA-Z]) war God [^bB] God [Bb]less
```

```
## 1                        8   11           0              0
## 2                        1    4           0              1
## 3                        0    2           0              0
## 4                        0    0           0              0
## 5                        0    0           0              0
## 6                        0    1           0              0
## 7                        0    3           0              0
## 8                       36   27           1              0
## 9                        3   45           0              1
## 10                       0    0           0              0
## 11                       3   14           0              0
## 12                       2   20           0              0
## 13                       0    0           0              0
## 14                       3   11           0              0
## 15                       7    3           0              0
##    (Jesus|Christ|Christian)
## 1                         0
## 2                         0
## 3                         0
## 4                         0
## 5                         0
## 6                         0
## 7                         0
## 8                         0
## 9                         0
## 10                        0
## 11                        2
## 12                        0
## 13                        0
## 14                        0
## 15                        0
```

As an example, I wrote a function which displays the number of occurence for each candidate for every year. I will not explain my code because I wrote it is here as an illustration.

```
plot_speeches= function(year){
  r=extract_regexpr_year(year,regexpr)
  par(mar=c(8,3,3,1))
  barplot(matrix(c(as.numeric(r[1,1:length(regexpr)+2]),
                   as.numeric(r[2,1:length(regexpr)+2]+1)),nrow=2),
          names.arg=c("I", "we", "America{,n}", "democra{cy,tic}",
                      "republic", "Democrat{,ic}", "Republican",
                      "free{,dom}", "war", "God", "God Bless",
                      "{Jesus, Christ{,ian}"),
```

```
          las=2,main=as.character(year),beside=T)
  legend("topright", c(as.vector(r$speaker[[1]]),
                       as.vector(r$speaker[[2]])), pch=15, bty="n",
         col=c("black","gray"))
}
par(mfrow=c(2,2))
for (i in 1:4) plot_speeches(2012-4*(i-1))
```



Looking at the histograms, we can say that Obama uses the word "I" less times than his Republican opponents. But surprisingly, Obama uses the word "Republican" more times than the Republicans themselves. This can highlights the fact that during the debates, he spent a lot of time criticizing his oppenent.

3 Problem 3

3.1 Random Walk

In order to compute the random walk I use the function rbinom which is useful for having a sequence of bernouilli variables 0,1. Then, I map the 0,1 output in order to have -1,1. Since the x and y variables are independent, we can split the walk for the x coordinate and the y coordinate, and then plot the corresponding path.
As an example, I compute a 100 steps Random Walk, return the final node and plot the corresponding path. The function takes into argument :

- n (integer) : number of steps in the random walk
- out (string, default = "") : whether the function returns the path ("path") or only the final node ("last")
- plot (bool, default = TRUE) : whether a plot of the path is displayed or not
- origin (numeric, default = (0,0)) : origin of the random walk
- legend (character, default = "bottomright") : localisation of the legend on the plot

Moreover, I create 3 exceptions to prevent the user that :

- n is not a number
- n is not an integer number
- n is not a positive integer

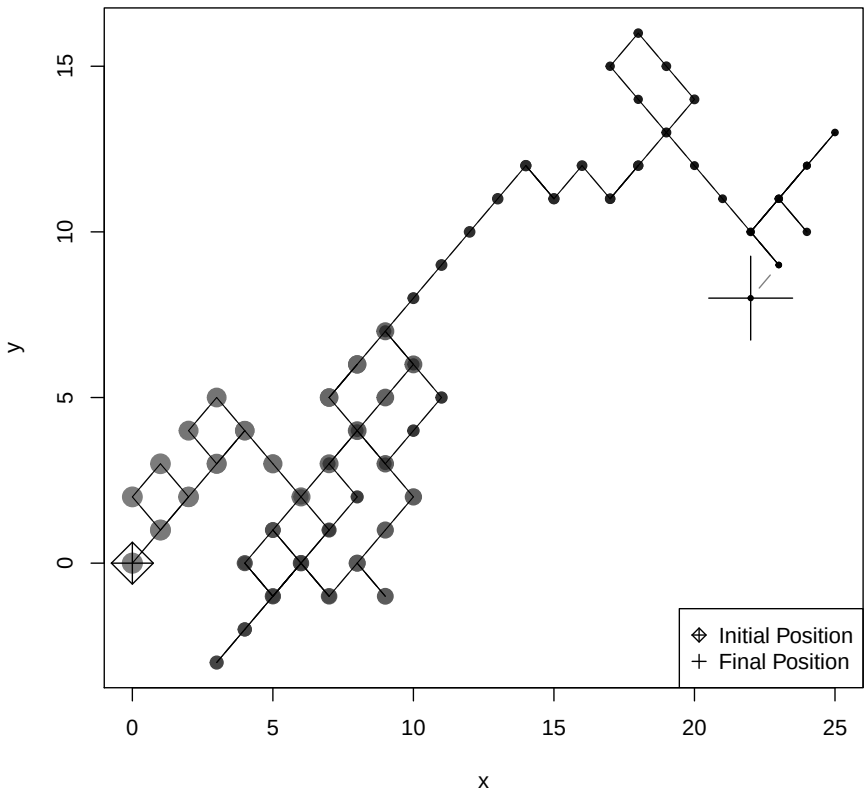
```
random_walk = function (n,out="",plot=T,origin=c(0,0),
                        legend="bottomright"){
  if (!is.numeric(n)){
    stop("\"n\" is not a number")
  }
  else if (!(floor(n)==n)){
    stop("\"n\" is not an integer")
  }
  else if (!(n>0)){
    stop("\"n\" is not positive")
  }
  else{
    x=c(0,cumsum(rbinom(n,1,0.5)-0.5)*2)+origin[1]
    y=c(0,cumsum(rbinom(n,1,0.5)-0.5)*2)+origin[2]
    #Plotting Options
    if (plot){
      ymax=max(y)
      ymin=min(y)
      xmax=max(x)
      xmin=min(x)
```

```
      plot(x,y,ylim=c(ymin,ymax),xlim=c(xmin,xmax),type="b",
           cex=rev(seq(.5,2,1.5/n)),col=gray(rev(seq(0,0.5,0.5/n)))
           ,pch=19,main=cat("Random Walk, ", n, " steps"))
      segments(x[1:(n-1)],y[1:(n-1)],x[2:n],y[2:n],lwd=0.8)
      points(x[1],y[1],pch=9,cex=3)
      points(x[n+1],y[n+1],pch=3,cex=6)
      legend( x=legend,
            legend=c("Initial Position","Final Position"),
            pch=c(9,3) )
    }
    #Return the path of the random walk
    if (out=="path")
      return(cbind(x,y))
    #Return the last position of the random walk olny
    if (out == "last")
      return(c(x[n+1],y[n+1]))
  }

}
```

As an example, I plot a random walk with 100 steps and origin (0,0).

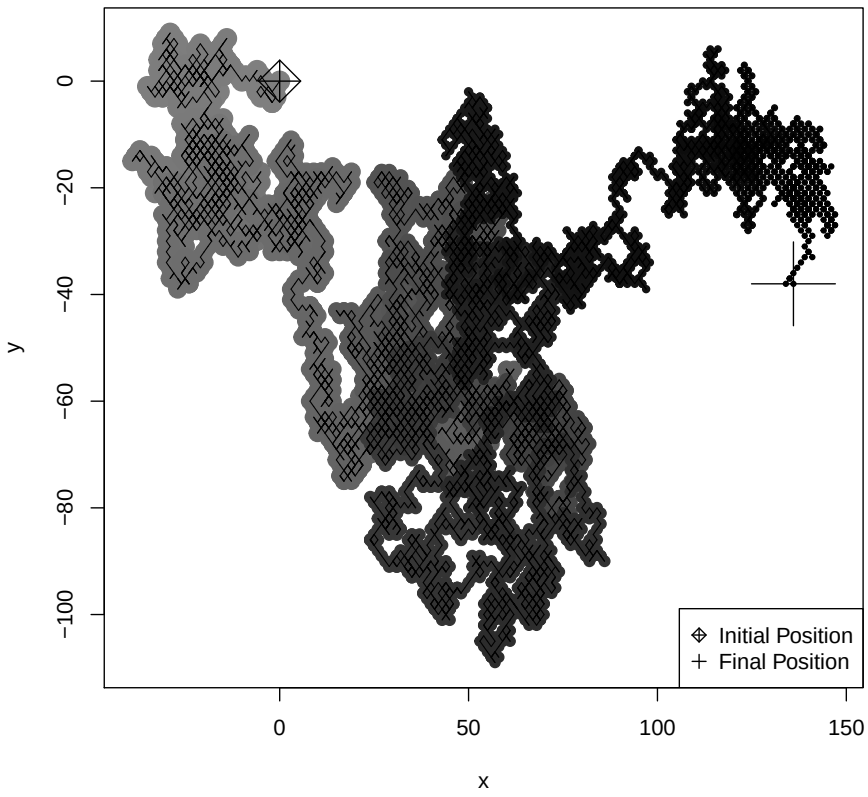
```
set.seed(15)
random_walk(100)
```




```
## Random Walk,  100  steps
```

For 10,000 steps, it looks like this :

```
set.seed(11)
random_walk(10000)
```




```
## Random Walk,  10000  steps
```

I can plot the path of the walk for an origin (4,5) :

```
set.seed(1)
random_walk(10,out="path",origin=c(4,5),plot=F)
```

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

```
##   [7,] 4 3
##   [8,] 5 4
##   [9,] 6 5
##  [10,] 7 4
##  [11,] 6 5
```

3.2 S3 Classes

I construct my rw class and the requested methods. In the constructor, I define two parameters : the number of steps of the random walk and the origin of it with a default value (0,0)

```
## @knitr constructor
rw <- function(steps=NA,origin=c(0,0)){
  # constructor for 'indiv' class
  obj <- list(steps=steps,origin=origin)
  class(obj) <- 'rw'
  return(obj)
}
```

```
## @knitr methods
print.rw <- function(object)
  return(random_walk(object$steps,plot=F,out="last",
                     origin=object$origin))
plot.rw <- function(object,legend="bottomright")
  return(random_walk(object$steps,plot=T,out="",origin=object$origin,
                     legend=legend))
summarize.rw <- function(object)
  return(with(object, cat("Random walk with ", steps,
                          " steps and origin (", origin[1],",",
                          origin[2],").\n",sep = "")))
```

```
## @knitr class-operators
`[.rw` <- function(object,i) {
  r=random_walk(object$steps,origin=object$origin,plot=F,out="path")
  return(r[i+1,])
}
```

```
## @knitr replacement
`start<-` <- function(x, ...) UseMethod("start<-")
```

```
`start<-.rw` <- function(object, value){
  object$origin <- value
  return(object)
}
```

Once I have defined my rw class, I can :

- Set a new "rw" variable called "walk".

```
set.seed(1)
walk <- rw(20)
```

- Redefine the start of the random walk.

```
start(walk) = c(5, 7)
```

- Print the summary of the walk object.

```
summarize.rw(walk)

## Random walk with 20 steps and origin (5,7).
```

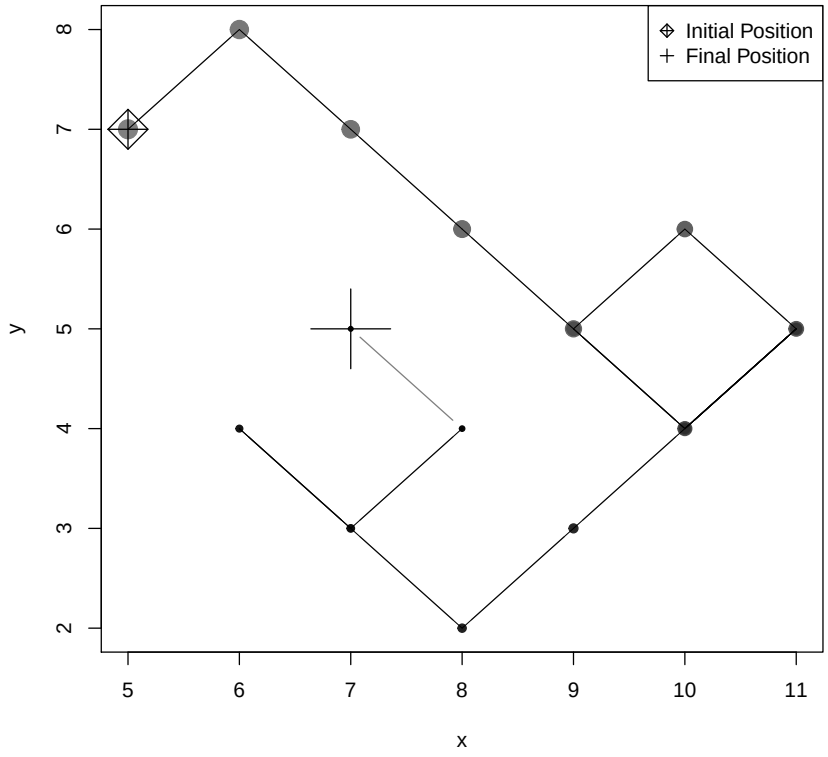
- Return the third position of the random walk.

```
`[`(walk,3)

##  x  y
##  4  8
```

- Plot the walk using the new plot method for rw objects.

```
plot(walk,legend="topright")
```




```
## Random Walk,  20  steps
```