Assignment 6

Yuwei Chen 999070073

I did this assignment by myself and developed and wrote the code for each part by myself, drawing only from class, section, Piazza posts and the Web. I did not use code from a fellow student or a tutor or any other individual.

Signature:

Part 1: Scraping the Summaries of the Posts

Username and Reputation

I extracted username and reputation together under the tag . There are more information other than username and reputation. I first extracted value of the second tag under since its always username.

It gets a bit tricky to deal with reputation because anonymous user and community wiki does not have the reputation tag as normal users do, so I have to first specify whether it is a normal user or anonymous/community wiki post. Then I extracted reputation from two places since high reputation and low reputation are put under different path.

```
grab = xpathSApply(html, "//div[@class = 'user-details']", function(x){
    as = xmlSApply(x, xmlValue, trim = TRUE)
    if (length(as) > 4){
      ## Normal users
      name = as[[2]]
      first = xmlChildren(x)$div
      second = xmlChildren(first)$span
      r_1 = xmlValue(second)
      r_2 = xmlGetAttr(second, 'title')
    }
……
```

Date

I found tag under the path.

```
xpathSApply(html, "//div[@class = 'started fr']/./div", function(html){
    as = xmlSApply(html, xmlValue, trim = TRUE)
    if (as[[2]] == 'community wiki'){
      output = NA
    } else {
      a = xmlChildren(html)$div
      b = xmlChildren(a)$span
      output = xmlGetAttr(b, 'title')
……
```

Since community wiki has different format, I first identified them and marked them NA. Then I extracted the span title of the fourth children after div[@class = 'started fr'].

Title

I used

```
path_title  = "//div[@class = 'summary']/h3/a[@class = 'question-hyperlink']/text()"
```

to find the titles of the posts. This worked pretty well on all of the posts.

View

I used

```
  xpathSApply(html, "//div[@class='statscontainer']/div[3]", xmlGetAttr, 'title')
```

to target view counts of the posts under the third tag after div[class='statscontainer']. Then I used regular expression to obtain numeric part.

Answer

I found answer count of a post using path:

```
  nodes_answer = xpathSApply(html, "//div[@class='stats']/div[2]", xmlValue)
```

and used regular expression to extract the number only. The answer counts are in the value of the second after div[class='stats']. This worked for all posts.

Vote

I found vote under the path

```
path = "//div[@class = 'votes']/span[@class = 'vote-count-post ']/strong/text()"
```

There are both positive and negative votes so I used regular expression to extract the numeric part.

URL

I used

```
URL = xpathSApply(html, "//div[@class = 'summary']/h3/a[@class = 'question-hyperlink']", xmlGetAttr, "href")
```

To find partial URL of the post. I worked very well but I have to add the base URL to content I extracted.

ID

I obtained ID of posts by

```
step_1 = xpathSApply(html, "//div[@class = 'question-summary']", xmlGetAttr, 'id')
```

It is in the tag <div[class = 'question-summary']> and is the attribute of id in the tag.

Tag

By extracting the value under "//div[class = 'summary']/div[2]", the second after the tag div[@class = 'summary'], I was able to get tags in an unorganized manner. I then use trimws to get rid of the in between spaces and leading/trailing spaces. Finally, I collapses them in to single string and connected them using '; '.

```
tag = "//div[@class = 'summary']/div[2]"
extract = xpathSApply(html, tag, xmlValue)
```

Construct partial data frame for a page

I used a function that calls all previous function to form a data frame after scraping one page. It has 50 observations and 10 variables. The input of the function are the tag we searched for and the content of the page.

Get the URL of the next page

By finding tag a[@rel = 'next'] and extracted its href name, I as able to get the URL for next page. I also need to add 'http://stackoverflow.com' before what I extracted. It will be used to get the content of the next page and continue the loop.

```
  extract = xpathSApply(html, "//a[@rel = 'next']", xmlGetAttr, 'href')
```

Top level loop

In the top level loop, I asked for input of starting page, ending page, and tag that is looked for. The default tag is 'r'. It will first load the content of the starting page and form a data frame of information for that page. Then the loop will load the content in next page and extract information into another data frame. This will be continued to the ending page. All data frames for each page will be combined into a large one and at the end saved as a RDS file called dataframe.rds.

```
data_p1 = main(1, 800, 'r')

head(data_p1, n = 3)
```

```
##         id                 date                                tag
## 1 34169722 2015-12-09 02:23:47Z r; data.frame; data.table; dplyr
## 2 34169588 2015-12-09 02:07:05Z    r; time-series; s4; smoothing
## 3 34169587 2015-12-09 02:07:01Z    r; regression; lapply; sapply
##                                                      title
## 1 Subset a dataframe using 2 columns with a condition in R
## 2                   Error: object used is not an S4 object
## 3                   Regression of variables in a dataframe
##
url
## 1 http://stackoverflow.com/questions/34169722/subset-a-dataframe-using-2-columns
-with-a-condition-in-r
```

```
## 2                    http://stackoverflow.com/questions/34169588/error-object-us
ed-is-not-an-s4-object
## 3                   http://stackoverflow.com/questions/34169587/regression-of-va
riables-in-a-dataframe
##   views votes answers          user reputation
## 1    10     1       0       Sharath        482
## 2     7     0       0 Haroon Rashid        400
## 3     9     0       1           H_A         95
```

```
nrow(data_p1)
```

```
## [1] 40000
```

I scraped 800 pages and obtain 40000 observation to form a data frame. (10 variables) I printed the first three observations to show I get the right format.

Part 3: Analyzing R Questions on StackOverflow

Question 1

What is the distribution of the number of questions each person answered?

In this question, I used the data frame rQAs given in the assignment in the assignment to conduct analysis.

```
head(answer_t)
```

```
##               user Freq   type
## 59           akrun  642 answer
## 1461 Richie Cotton  594 answer
## 981         learnr  581 answer
## 1806    user225056  580 answer
## 606           Frank  167 answer
## 1019      LyzandeR  159 answer
```

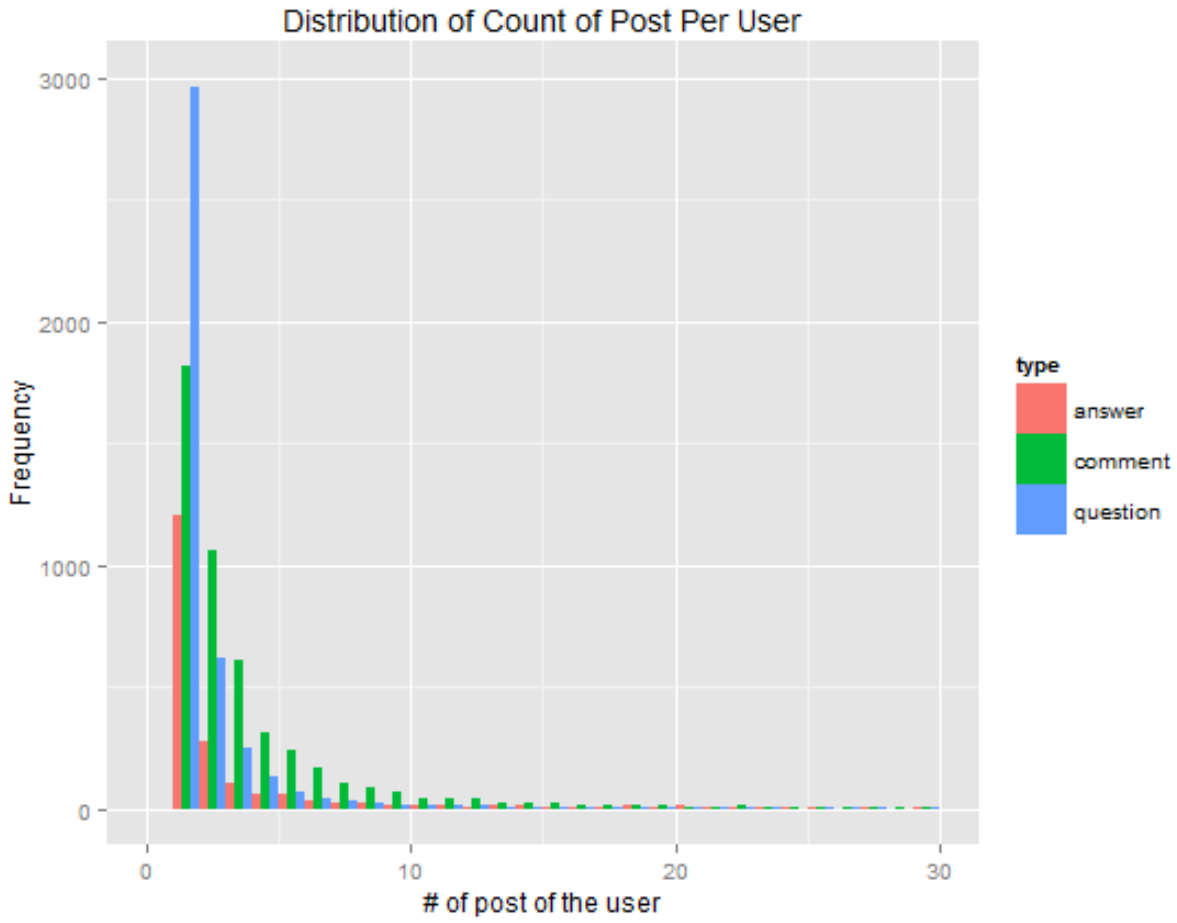
```
head(question_t)
```

```
##                 user Freq     type
## 1058   Eduardo Leoni  580 question
## 2685          Pascal  344 question
## 1537            Jaap  176 question
## 863   David Arenburg  155 question
## 2955 Richard Scriven  121 question
## 1205           Frank  111 question
```

```
head(comment_t)
```

```
##               user Freq    type
## 1223 Eduardo Leoni 1163 comment
## 130          akrun 1066 comment
```

```
## 3132        Pascal  689 comment
## 1395         Frank  663 comment
## 1572        hadley  608 comment
## 3455 Richie Cotton  594 comment
```



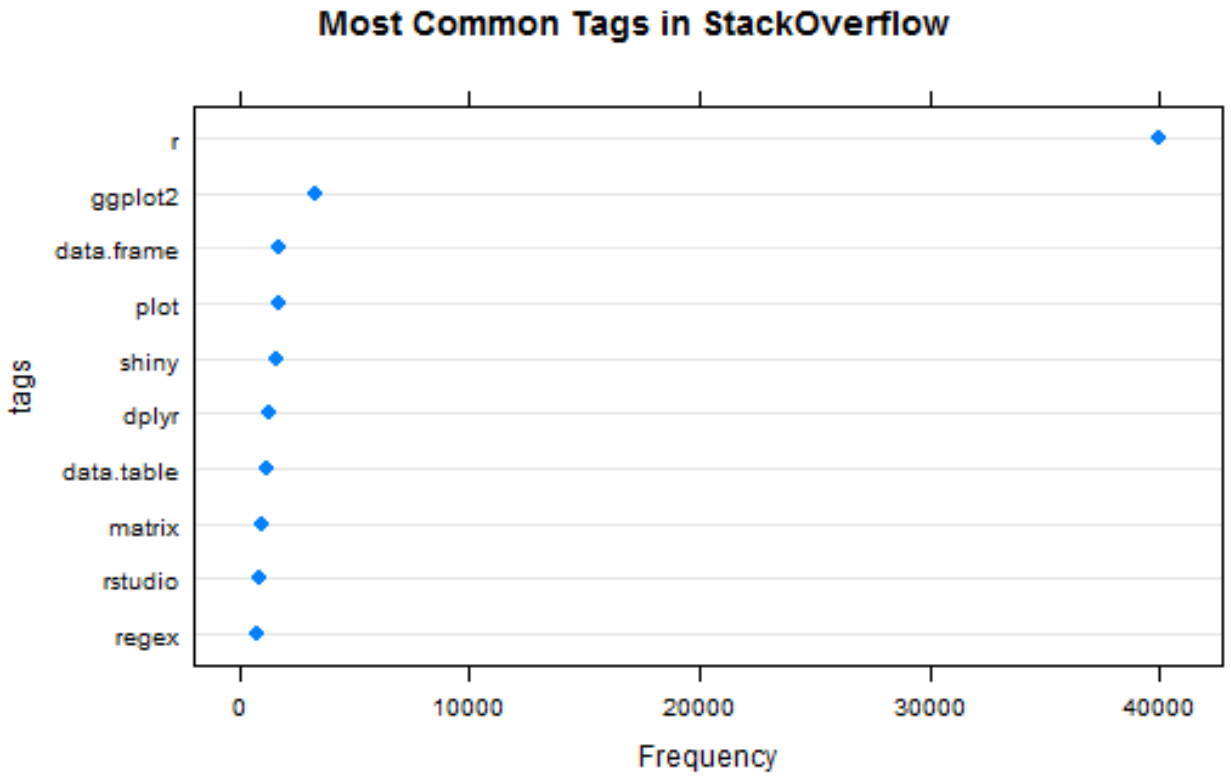
We can see that most of the users only make post with for limited times and then never actively post. Most users seem to sign up an acount and ask one or two questions in the website. On the contrary, there are far less people sign up to answer question on StackOverflow.com. Compared to posting questions and answers, it seems that users post more comments, possibly because it is ealiser to post comments to have conversations with other users. A common point among all three types of posts per user is that the majority of users have limited activities on the websites. Only several users can consistenly keep posting on StackOverflow.com.

In the graph, I omitted the users with high involvement, since there is only a few of them and they are not obvious on a plot. They are shown under the output table.

Question 2

What are the most common tags?

In this question, I used the data frame I formed using codes from part 1. I extracted the tag column from it to do the analysis.



Since we scraped the page with all questions mentioning r, apparently r is the most common tag in this data set. It appears in almost every post I scraped. Besides r, ggplot is a popular tag because its useful and complex. Other popular tags mentioned in the data set includes data.frame, plot, shiny, dplyr, data.table, matrix, rstudio, and regex.

Question 3

How many questions are about ggplot?

In this question, I used the data frame I formed using codes from part 1. I extracted the tag column from it to do the analysis.

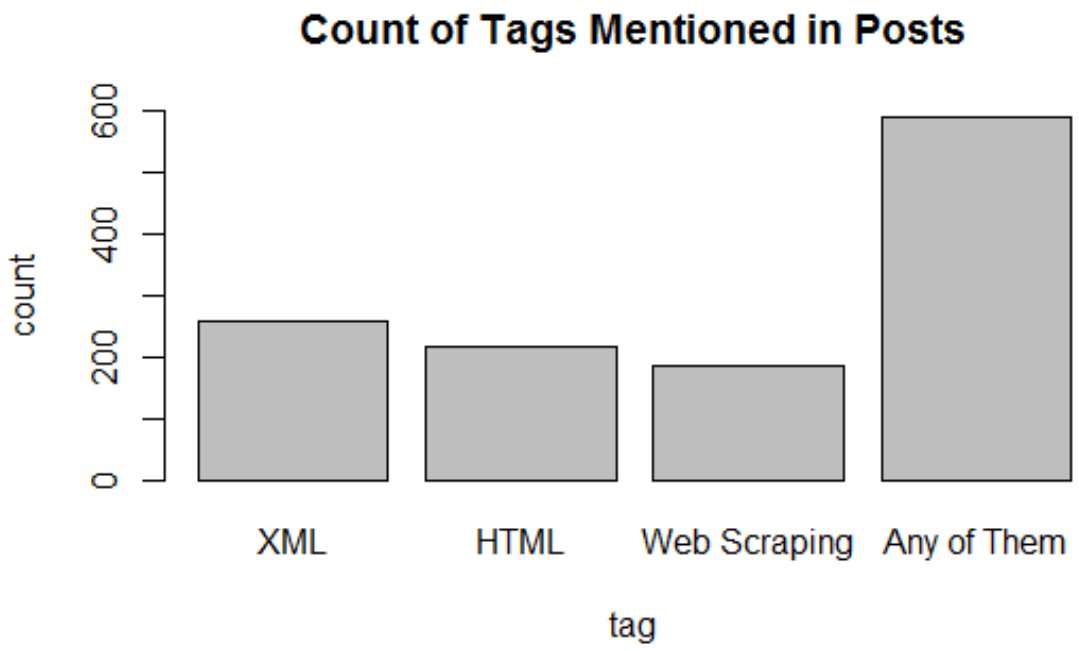
There are 3298 questions mentioning ggplot or similar topics(ie. ggplot2). My strategy is to subset the data frame using the tag column.

Question 4

How many questions involve XML, HTML or Web Scraping?

In this question, I used the data frame I formed in part 1 in the assignment to conduct analysis.

```
count_xml

  ## [1] 257
count_html

  ## [1] 217
count_web_scraping

  ## [1] 185

xml_html_web_scraping

  ## [1] 590
```



People used the tag XML more often than HTML and web scraping. There are some posts mentioning two or more among XML, HTML and web scraping, but not many. Although they seem to be related to each other, generally people don't tag these three together. On the contrary, people tag them individually and ask specific questions about these three tags separately.

Question 5

What are the names of the R functions referenced in the titles of the posts?

In this question, I used the data frame rQAs given in the assignment in the assignment to conduct analysis.

```
head(sort(table(f_list), decreasing = T), n = 50)
```

```
## f_list
##      with      data       for  function        by      plot        is
##      1409      1348       916       532       431       426       397
##     frame     table      list        as      file    matrix      time
##       353       312       282       239       235       232       231
##       row       get    vector       all      find      text       new
##       230       190       186       175       167       153       143
##      date    remove        if      axis     names character    within
##       138       128       124       120       119       118       116
##    subset    factor     apply      line   numeric   replace     split
##       113       111       103       102        86        82        80
##     while     order   missing    return    legend     merge      mean
##        79        76        75        74        72        72        69
##    points    single     which     lines    labels       sum     match
##        69        69        69        68        67        65        64
##     scale
##        64
unique_f = unique(f_list)
length(unique_f)
```

```
## [1] 399
```

At the beginning of the report, I used search() to get a list of preloaded packages in R and got all function names in these packages. They are used to match words appeared in the title to see if they are functions.

In total, there are 399 functions used in the data frame rQAs. I listed the most frequently used functions in the table. However, it should be noted that some top the top tags are not only functions but also important words we used in daily life such as 'with', 'by', 'for', 'is' and so on . Therefore, there exists the case where I identified them as function name while they only appear as an English word in the totle.

Question 6

What are the names of the R functions referenced in the accepted answers and comments of the posts? We can do better than we did in processing the title of the posts as there is HTML markup and we can find content in code blocks.

In this question, I used the data frame rQAs given in the assignment in the assignment to conduct analysis.

I first found accepted answer posts by finding answers for a certain questions and take the answer with highest score. They are considered accepted answer for me. Then I conbined accepted answers and comments into a data frame to explore.

I first dealt with the text column and extracted contents within a pair of tag "<code" "</code>". Things within them are codes, which contain functions. I string split the content and separate words in order to match the function vector.

```
sort(table(f_list_2), decreasing = T)
```

```
## f_list_2
##  character          c       list    library   function data.frame
##      27186       9995       1305        944        873        629
##     lapply         df         by       data          :        for
##        612        574        487        446        435        397
##       dput         if      names     length        sum     sapply
##        330        328        297        296        295        284
##     matrix      paste       mean      class      apply          {
##        281        267        264        251        250        243
##       with    do.call          -        rep     paste0       nrow
##        234        229        223        219        211        210
## as.numeric          t read.table       plot        seq     ifelse
##        202        190        189        188        188        186
##       text       ncol     factor     sample         is  row.names
##        186        181        180        178        174        174
##       %in%      table       head      merge      cbind     unique
##        171        170        166        164        163        162
##      which        max
##        160        157
```

```
unique_f_2 = unique(f_list_2)
length(unique_f_2)
```

```
## [1] 900
```

In total, there are 900 functions used in the data frame rQAs. I listed the most frequently used functions in the table. Compared to Q5, by extracting from the code blocks in the text, I was able to get more accurate results because the contents will be purely codes instead of sentences. Therefore, I am certain that they are used as functions. 'character' and 'c' functions seemed to be the popular functions mentioned in the accepted answers and comments.

Appendix

```
## Obtain preloaded package names before I load any outside packages
## Code needed for part 3 Q5 and Q6
packages = search()
library(stringr)
packages = unlist(str_extract_all(packages, 'package\\:.*'))
all_function = unlist(lapply(packages, ls))

## Uploading library
library(XML)
library(RCurl)
library(memisc)
library(psych)
library(ggplot2)
library(lattice)
```

Username and Reputation

```
get_info = function(html){
  grab = xpathSApply(html, "//div[@class = 'user-details']", function(x){
    as = xmlSApply(x, xmlValue, trim = TRUE)
    if (length(as) > 4){
      ## Normal users
      name = as[[2]]
      first = xmlChildren(x)$div
      second = xmlChildren(first)$span
      r_1 = xmlValue(second)
      r_2 = xmlGetAttr(second, 'title')
      if (grepl('[0-9,]+', r_2)) {
        ## User with high reputation
        reputation = str_extract(r_2, '[,0-9]+')
      } else {
        ## User with low or moderate reputation
        reputation = r_1
      }
    } else {
      if (length(as) < 4) {
      ## community wiki and anonymous users
      name = NA
      reputation = NA
      } else {
        ## useless empty string
        name = 'drop'
        reputation = 'drop'
      }
    }
    output = cbind(name, reputation)
  })
  ## Form a data frame with two columns
  return_data = data.frame(t(grab))
```

```
  colnames(return_data) = c("user", "reputation")
  return_data_1 = subset(return_data, reputation != 'drop'| is.na(reputation) == TR
UE)
}
```

Date

```
get_time = function(html){
  xpathSApply(html, "//div[@class = 'started fr']/./div", function(html){
    as = xmlSApply(html, xmlValue, trim = TRUE)
    if (as[[2]] == 'community wiki'){
    ## Deal with community wiki where time is not correct
    output = NA
    } else {
      a = xmlChildren(html)$div
      b = xmlChildren(a)$span
      output = xmlGetAttr(b, 'title')
    }
    output
  })
}
```

Title

```
get_title = function(html){
  path_title  = "//div[@class = 'summary']/h3/a[@class = 'question-hyperlink']/text
()"
  nodes_title = xpathSApply(html, path_title, xmlValue)
  nodes_title
}
```

View

```
get_view = function(html){
  nodes_view_1 = xpathSApply(html, "//div[@class='statscontainer']/div[3]", xmlGetA
ttr, 'title')
  view = lapply(nodes_view_1, function(x){
    regex = '[0-9,]+'
    str_extract(x, regex)
  })
  unlist(view)
}
```

Answer

```
get_answer = function(html){
  nodes_answer = xpathSApply(html, "//div[@class='stats']/div[2]", xmlValue)
  answer = str_extract(nodes_answer, '[0-9]+')
}
```

Vote

```
get_vote = function(html){
  path_vote  = "//div[@class = 'votes']/span[@class = 'vote-count-post ']/strong/te
xt()"
  vote = xpathSApply(html, path_vote, xmlValue)
}
```

URL

```
get_URL = function(html){
  ## URL
  nodes_URL = xpathSApply(html, "//div[@class = 'summary']/h3/a[@class = 'question-
hyperlink']",    xmlGetAttr, "href")
  ## Add the base URL in the front of the partial URL I extracted
  URL = getRelativeURL(nodes_URL, 'http://stackoverflow.com')
}
```

ID

```
get_id = function(html){
  step_1 = xpathSApply(html, "//div[@class = 'question-summary']", xmlGetAttr, 'id'
)
  regex = '[0-9]+'
  step_2 = str_extract(step_1, regex)
  step_2
}
```

Tag

```
get_tag = function(html){
  tag = "//div[@class = 'summary']/div[2]"
  extract = xpathSApply(html, tag, xmlValue)
  tag = gsub(" ", "; ", trimws(extract))
}
```

Construct partial data frame for a page

```
data_frame = function(html, wanted_tag){
  ## Form a data frame by calling all small functions before
  info = get_info(html)
  date = get_time(html)
  title = get_title(html)
  views = get_view(html)
  answers = get_answer(html)
  votes = get_vote(html)
  url = get_URL(html)
  id = get_id(html)
  tag = get_tag(html)
  df = data.frame(cbind(id, date, tag, title, url, views, votes, answers, info))
```

```
  rownames(df) = NULL
  ## Subset and only get posts with desired tags
  df = subset(df, grepl(wanted_tag, df$tag))
}
```

Get the URL of the next page

```
next_page = function(html){
  extract = xpathSApply(html, "//a[@rel = 'next']", xmlGetAttr, 'href')
  getRelativeURL(extract, 'http://stackoverflow.com')
}
```

Top level loop

```
main = function(start_page = 1, end_page = Inf, wanted_tag = 'r'){
  ## Differentiate starting page
  if (start_page == 1){
    txt = 'http://stackoverflow.com/questions/tagged/r?sort=newest&pagesize=50'
  } else {
    txt = paste0('http://stackoverflow.com/questions/tagged/r?page=', start_page, '
&sort=newest&pagesize=50')
  }

  ## Getting contents form the website
  zz = try(getURLContent(txt, binary = TRUE), silent = TRUE)
  doc = try(rawToChar(zz))
  page_content = try(htmlParse(doc, asText = TRUE), silent = TRUE)
  total = NULL
  partial = NULL

  ## Run a loop From staring page to the ending setted page using
  for (i in start_page : end_page){
    partial = data_frame(page_content, wanted_tag)
    total = rbind(total, partial)
    txt = next_page(page_content)
    doc = try(getURLContent(txt), silent = TRUE)
    page_content = htmlParse(doc, asText = TRUE)
  }

  ## Save the data frame into a file dataframe.rds
  saveRDS(total, "dataframe.rds")
  total
}

data_p1 = main(1, 800, 'r')

data_p1 = readRDS('D:/STA 141/Assignment 6/dataframe.rds')
head(data_p1, n = 3)
```

Part 3:

```
download = load('D:/STA 141/Assignment 6/rQAs.rda')
qa = rQAs

## Subseting the given data frame by the type of the posts
answer = subset(qa, type == 'answer')
question = subset(qa, type == 'question')
comment = subset(qa, type == 'comment')


## Make their count of post per user into data frames
answer_t = data.frame((table(as.matrix(answer$user))))
answer_t = answer_t[order(-answer_t$Freq),]
answer_t$type = 'answer'
head(answer_t)

question_t = data.frame((table(as.matrix(question$user))))
question_t = question_t[order(-question_t$Freq),]
question_t$type = 'question'
head(question_t)

comment_t = data.frame((table(as.matrix(comment$user))))
comment_t = comment_t[order(-comment_t$Freq),]
comment_t$type = 'comment'
head(comment_t)

## Combine them into one for plotting purpose
huge = rbind(answer_t, question_t, comment_t)

ggplot(huge, aes(Freq, fill = type)) +
  geom_bar(breaks = seq(0, 30, by = 1), position = 'dodge')+
  xlab("# of post of the user") + ylab("Frequency") + ylim(0,3000) +
  ggtitle("Distribution of Count of Post Per User")
```

Q 2:

```
## Find all tags have appeared
column = unlist(as.character(data_p1$tag))
tags_all = unlist(strsplit(column, "; "))

## Count and sort them and pick the top 10 most common tags
x = head(sort(table(tags_all), decreasing = T), n =10)
tag_df = as.data.frame(table(as.matrix(tags_all)))
colnames(tag_df) = c('tag', 'Frequency')
tag_df = tag_df[order(-tag_df$Frequency),]
tag_df_1 = head(tag_df, n = 10)
df = droplevels(tag_df_1)
dotplot(x = sort(x), ylab = 'tags', xlab = 'Frequency', cex = 1,
        main = "Most Common Tags in StackOverflow")
```

Q 3:

```
count = nrow(subset(data_p1, grepl('ggplot', data_p1$tag)))
```

Q4:

```
## Count how many time does each of XML, HTML and web scraping appear in the tag
count_xml = nrow(subset(data_p1, grepl('xml', data_p1$tag, ignore.case = TRUE)))

count_html = nrow(subset(data_p1, grepl('html', data_p1$tag, ignore.case = TRUE)))

count_web_scraping = nrow(subset(data_p1, grepl('web[-_ ]?scraping', data_p1$tag,
                          ignore.case = TRUE)))


## Count how many time does a tag include any of XML, HTML or web scraping
xml_html_web_scraping = nrow(subset(data_p1, grepl('xml|html|(web[-_ ]?scraping)',
                             data_p1$tag, ignore.case = TRUE)))


tp_plot = c(count_xml, count_html, count_web_scraping, xml_html_web_scraping)

barplot(tp_plot, xlab = 'tag', ylab = 'count', ylim = c(0, 600)
        main = 'Count of Tags Mentioned in Posts',
        names.arg=c("XML", "HTML", "Web Scraping", 'Any of Them'))
```

Q5:

```
q_title = rownames(question)
q_title = str_extract(q_title, '[^//]+$')

## String split titles by '-' into separate words and do matching
f_list = lapply(q_title, function(y){
  words = unlist(str_split(y, '-'))
  appeared = intersect(words, all_function)
})

## Find unique functions in the vector
f_list = unlist(f_list)
head(sort(table(f_list), decreasing = T), n = 50)
unique_f = unique(f_list)
length(unique_f)
```

Question 6

```
## Find accepted answer posts
accepted_answer = split(answer, answer$qid)
```

```r
accepted_answer = lapply(accepted_answer, function(post){
  post[which.max(post$score),]
})

df_accepted = do.call('rbind', accepted_answer)

## Form a data frame containing accepted answers and comments
answer_comment = rbind(df_accepted, comment)

## Extract codes from the text column
codes = str_extract_all(answer_comment$text,
        regex("<code>[[a-z][0-9][:punct:]\\s=]+</code>", ignore_case = TRUE))
codes = gsub('<code>|</code>', '', codes)

## String Split
to_match = lapply(codes, function(z) {
  list = str_split(z, '\n|<.*>| |=|\\(|\\)|"')
  prep = unlist(list)
  prep = prep[prep != ""]
})


## Match functions in the function list
f_list_2 = lapply(to_match, function(a_post) {
  appeared = intersect(a_post, all_function)
})

## Find unique functions in the vector
f_list_2 = unlist(f_list_2)
sort(table(f_list_2), decreasing = T)

unique_f_2 = unique(f_list_2)
length(unique_f_2)
```