Tiffany Chen 12/9/15 ID: 998840686

STA 141 Assignment 6

Introduction

Sometimes it's a good idea to extract information from webpages. There is a wide range of uses from research to web data integration. For this assignment, we're going to scrape Stackoverflow webpage to extract information on questions, answers, posts and other information for analysis.

Statement

```
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.
```

Tiffany Chen

Questions

Part 1: scraping the summaries of the posts

For part 1, we are to extract information about who posted it, when it was posted, the title of the post, the reputation level of the poster, current number of views for the post, current number of answers for the post, the vote "score" for the post, the URL for the page with the post, answers and comments, and the id uniquely identifying the post. We also need to get the URL for the "next" page listing the posts. Then, we combine everything into a function that can allow the caller to specify the tag and number of pages to process and it will return a data frame with a row for each post.

To do this, I first got the information needed for each post separately by working on each xpath. Before this, I used RCurl's getURLContent() to download the stackoverflow webpage and parsed the webpage with htmlParse() and set asText = TRUE because doc is the text of the webpage I downloaded. Then, I worked on the individual xpaths. For example, for the user, the xpath was path = "//div[@class = 'user-details']because the username is the text of whats under the a tag which is under the div tag with class with value equal to user-details. I determined the exact path by using inspect element of my browser on the stackoverflow webpage. Then, I used the path to getNodeSet to run the xpath query where it first takes in the parsed document and then the xpath I specified. Then I used sapply() function with the node as the first argument and the second argument as xmlValue to get back all the usernames as a character. Later, I realized from Nick's discussion notes that these two steps can be combined to xpathSApply with the htmlParse doc, path, and xmlValue all in one line. To deal with potential missing values, I added function(x) xmlValue(xmlChildren(x)$a to the xmlSApply based on several posts on Piazza, which returns NA instead. To get the rest of the information, I followed similar procedures where I made the path then used the xpathSApply. These are the xpaths I used for each part of the information we needed to extract:

- For title of post, my xpath was "//div[@class = 'summary']/h3/a/text()". I feel like this strategy is good because it quickly gets to the part of the text we need.
- When it was posted: path = "//div[@class = 'user-action-time']/span/@title"
- The reputation level of the poster: path = "//div[@class = 'user-details']/span[@class = 'reputation-score']/text()"

- Update 12/7/15: apparently overnight, the Stackoverflow changed their HTML of their webpage. So I had to change my path to path = "//div[@class = '-flair']" I added the additional function(x) xmlValue(xmlChildren(x)$span) in the xmlSApply like for user name in case of no reputation level. I know there are other strategies based on several piazza posts, such as sapply() over the the users and applying a function that would check for the length of each post. If the length is less than 1, we would return NA or else just return the reputation. However, I tried many times and could not get it to work. My xpaths work so that it reads all 50 posts at once and I know im supposed to read each post one by one, but when I do that it doesn't show missing values? Somehow, this xmlChildren thing works though! As for the anomaly's, I'm almost sure there are more, but I wasn't aware of them.

- The current number of views for the post: path = "//div[@class = 'views ']/text()"
  - In addition to the xpathSApply, I also did a gsub and regular expression to remove the "views" term and anything else extraneous. Like in "17 views", I did gsub("[a-z\n\r]", "", views) so that I would just get back the number "17". I remember reading somewhere that we didn't need to use regular expressions in part 1, but I can't think of any other way to get rid of the word "views" and this method works well anyways.
- Current number of answers for the post: path = "//div[@class = 'status answered-accepted' or @class = 'status unanswered' or @class = 'status answered']/strong"
  - The key part of this xpath is that there are 3 conditionals for an answer. at first I ran it without the "status answered" part and then later realized it wouldn't get all the posts. So including all the conditionals is key!
- Vote score: path = "//span[@class = 'vote-count-post ']"
  - The tricky part here was the space at the end of post!
- URL for the page with the post: path = "//div[@class = 'summary']/h3/a/@href"
  - Since the URL extracted only gives back the last part of the URL, I used the paste function to add on the beginning of the url so url = "http://stackoverflow.com" and urls = paste(url, urls, sep = ""). I did try the getRelativeURL() command but it gave me so many problems later on in the top-level function so I reverted to using the paste().
- The id for getting the post: path = "//div[@class = 'question-summary']/@id"
  - Then I using gsub and regular expression to remove the extraneous words in before the actual number id to get it from "question-summary-34106898" to "34106898". Once again, im not quite sure how to get just the numerical values without some kind of regular expression.
- Tags: path = "//div[starts-with(@class, 'tags')]"
  - Then used another gsub to add a ; between the tags for each post. I tried to the longest time to use the paste() function with collapse = ; but it wouldn't put the ; between the individual tags, instead it would put the semicolon between each posts' tags. The key part of retrieving the tags was to set trim = TRUE in the xpathSApply parameter to get just the tags without the miscellaneous /n/r.

another key part was using starts-with. Nick discussed this in his office hours and I tried using contains() but it got back more than I wanted. This unfortunately took me a while to figure out but piazza posts helped!

Next, I made all these 10 parts into their own function. The input is the htmlParse of the getURLContent of the stackoverflow url which I mentioned I did before finding the xpaths.

```
url = "http://stackoverflow.com/questions/tagged/r?sort=newest&pagesize=50"
u = getURLContent(url)
doc = htmlParse(u, asText = TRUE)
```

The output of the functions is the data frame of the 50 posts I scraped in one column.

Subsequently, I made another function to cbind all these 10 parts by calling all those functions so it would return the results of what I scraped from one page.

I also have the getNextURL function which I mostly copied from Duncan's cybercoders example. However, as mentioned above, I had difficulties with getRelativeURL so I used paste() again.

Finally, I made the top-level function that basically combines everything together. It takes in the tag the user can specify with quotes and the number of pages they wish to scrape. It returns the information requested in the above bullet points for each page according to the tag. Basically I first build the url depending on the tag with a combination of paste() functions. Then, I do the getURLContent and htmlParse. Then, in a for loop I call the function that read all the posts for one page and then the function that gets the next url and bind them together for however many pages necessary. I also made a version that works for until all the pages are processed by using a while loop instead of a for loop. This is quite similar except if the length of the getNextURL function is 0 then it would just break. Creating this top level function was a long trial and error process. Every time I ran it, there would be an error, and when I moved things around, there would be a different error. I tried to debug it many times with the browser() function and ended up realizing the error was with getRelativeURL. I decided to use paste() to make the URLs and this actually solved the problem. Looking back, I think it has to do with how I read in the data. I vaguely remember reading something on piazza where getRelativeURL takes in only the getURLContent or something.

Some of the output looks like this:

```
  id         date                    tags
1 34147406 2015-12-08 03:21:19Z      r; function; ggplot2; axis-labels
2 34147163 2015-12-08 02:53:19Z                              r
3 34147109 2015-12-08 02:46:20Z                              r
4 34146783 2015-12-08 02:10:16Z                              r
5 34146769 2015-12-08 02:07:48Z r; average; numeric; calculated-columns
6 34146635 2015-12-08 01:50:41Z                              r
                                  title
1 R: write function to set title of ggplot based on input
2     How to code for independent 2 sample t-test (x,y)
3           How to extract time form POSIXct and plot?
4               Removing single quote from dataframe
5                 rowMeans if column name is number
6               r barplot: put numbers on top of bar
```

```
                                                                    url
1 http://stackoverflow.com/questions/34147406/r-write-function-to-set-title-of-ggplot-based-
on-input
2      http://stackoverflow.com/questions/34147163/how-to-code-for-independent-2-sample-
t-test-x-y
3           http://stackoverflow.com/questions/34147109/how-to-extract-time-form-posixct-
and-plot
4               http://stackoverflow.com/questions/34146783/removing-single-quote-from-
dataframe
5                http://stackoverflow.com/questions/34146769/rowmeans-if-column-name-is-
number
6               http://stackoverflow.com/questions/34146635/r-barplot-put-numbers-on-top-of-
bar
  views votes answers         user reputation
1     4     0       0         csik          8
2    11    -1       0 Jordan Browne          1
3     6     0       0    Burry  Xie         14
4    23     0       0          MAPK        555
5    20     2       2     Bryan Han         11
6    12    -2       0 Nodir Kodirov          6
```

<u>Part 2: Scraping the posts, answers, and comments</u>

For part 2, we are to write a function that processes the actual page for an individual post. The function would extract and combine information for the post, each answer and comment. I did part of this part, but due to time constraints I wasn't able to finish. The basic methodology is to use the urls from the data frame in part 1, and apply this new function to those urls. I would first save the list of urls from part 1's function by extracting the column. Then, I would imagine it uses something like lapply() to apply the new function to each of the urls. This new function is like the function I made in part 1, but instead it returns different informations such as the type of entry, user, userid, date, user's reputation, score/votes for the entry, the HTML content as a string, the unique id for the overall post, and the identifier for the parent entry. I ended up making some of those smaller functions. I will use bullet point format similar to part 1 to explain the xpaths I used:

- Users: path = "//div[@class = 'user-details']"
- User id: path = "//div[@class = 'user-details']/a"
- Date: path = "//p[@class = 'label-key']/@title"
- Reputation: path = "//div[@class = '-flair']"
  - nodes = getNodeSet(doc, path)
  - rep = xmlSApply(nodes, function(x) xmlValue(xmlChildren(x)$span)) this part is to return NA for missing reputation levels similar to in part 1.
- scores/votes: votes = xpathSApply(doc, "//span[@itemprop = 'upvoteCount']", xmlValue)
- unique id: path = "//h1[@itemprop = 'name']/a/@href"

- then, used gsub and a regular expression to get back just the numerical id values.

Part 3: Analyzing R Questions on Stackoverflow

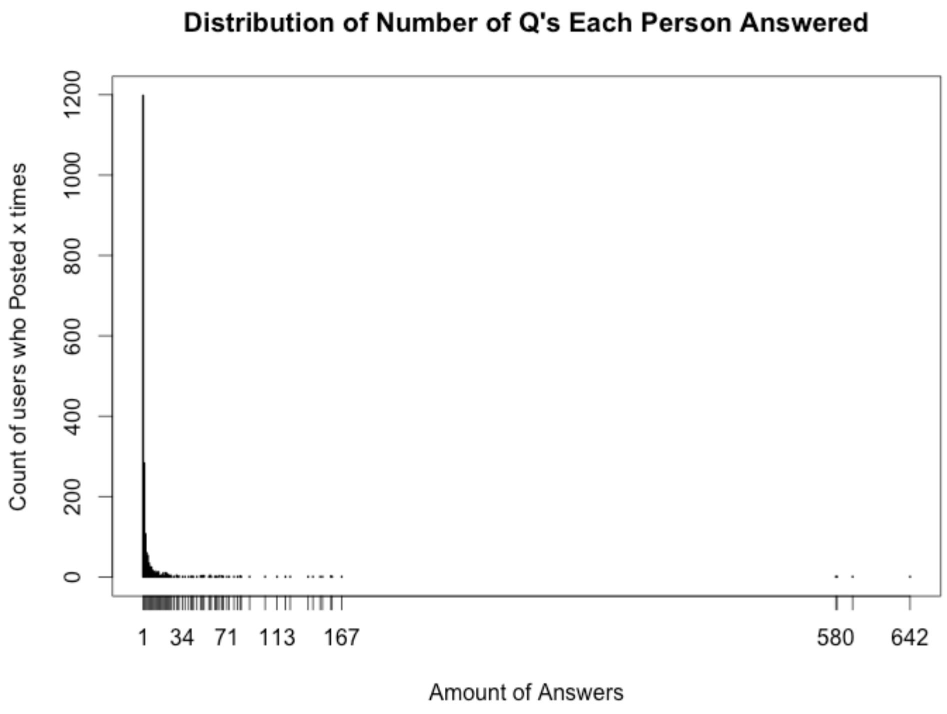
* Note: I used the given rQAs data for some of the questions. I will explicitly state it for each question.

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

First glance, I thought to literally find each user and get the number of questions they answered. However, according to a post on Piazza, someone suggested to have the x axis have the amount of answers and the y axis to have the frequency of users who posted that many times. I couldn't understand what he was trying to say for a long time. Then an example is like if 5 people (y) posted 2 answers(x), that is how we would plot those two numerical values. First off, I used the rQAs data given. I subsetted just the $type by answer. Then, I did a table of those answers$user and then, did a table of that. This way I got back something that looked like

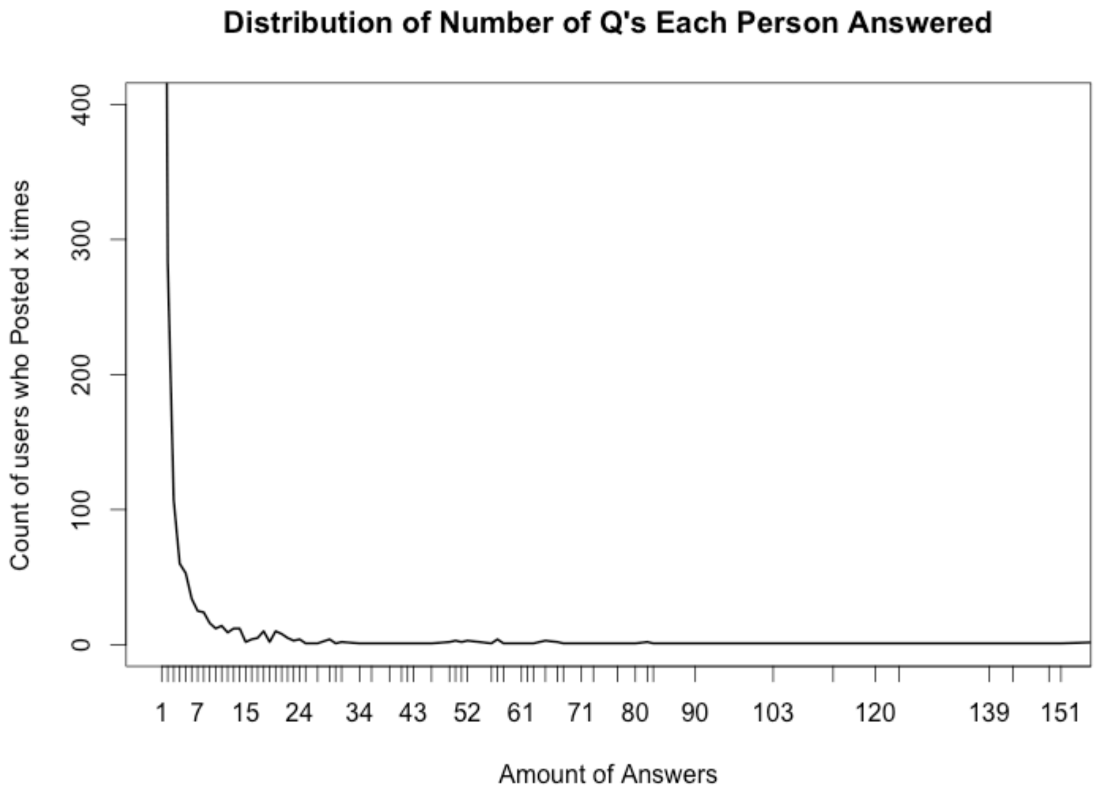
```
113  120  124  139  143  149  151  158  159  167  580  581  594  642
  1    1    1    1    1    1    1    2    1    1    1    1    1    1
```

which means (looking at the last one) that 1 person answered 642 questions! Then, I plotted this table and this is what follows:



From the plot, we can see that majority of people answer 1 question, and very few people answer more than say 34 questions. This makes sense, as most people aren't frequent users. If they are, there are very few of those. Not a lot of people send their time writing hundred of answers on piazza afterall!

I also tried making the graph a little more "readable" but zooming in on the x axis focusing mostly of people who are common and not those extreme stackoverflow users. I also zoomed in on the y axis because from the plot above, it seems like most people answer 1 question and I was interested to see if there is a pattern later on. I also changed the line type to a straight line instead of bars so I can notice any patterns more easily. Now, I see that perhaps people answer mostly 1 question, then to about 7 questions, is where theres a bigger dip and the line just levels off to about 0 users who posted x amount of times.



2. What are the most common tags?

I used the tags I scraped from part 1. I used my function and scraped 100 pages (due to time constraints I did not scrape all the pages and Duncan mentioned to do a couple thousand posts so 50x100 = 5000 posts) and with the tag "r". So, right away I know the most common tag should be "r". in fact, there should be 5000 "r" posts. Then, I got back just the tags by specifying that column. Since the format is like "r; ggplot; shiny", I had to get it back to individual elements. I first made the tags into characters, then unlisted it. Then, it was a matter of figuring out which was most common. I strsplit the tags, then used the table() function, sorted that in decreasing order. Looking at the head(), I got back:

```
     r    ggplot2      plot      shiny data.frame      dplyr
  5000       402       205       203       191       189
```

Since I did scrape for the "r" tag, it is great that I got back that the most common tag is "r"!

3. How many questions are about ggplot?

I answered this problem in two ways using the different data sets. First, I used the tags from the previous question and searched for the term “ggplot” using grep of the rownames of the table I created. I got back 402 questions about ggplot2 out of 1500 tags, which is the same as the output above. In another method, I used the rQAs data. Since the question asks about questions, I subsetted the data type of questions. Then, just specified the column of text because I want to look for the term “ggplot” in the questions. I did the same grep function as my first data and got back 956 questions about ggplot out of 10004 questions.

4. How many questions involve XML, HTML or Web Scraping?
For this problem, I did the same procedures as the previous problem. I used the tags from the data I got from part 1 combined with grep of the key words xml, html, scraping, web [grep("xml|html|web-scraping|scraping"] with ignore.case = TRUE. I got back

```
html    html-parsing     html-table    html-to-text       libxml2
        18            2            1            1            2
      r2html screen-scraping    web-scraping          xml     xml-parsing
          1            3            32            28            5
```

So, 28 questions about xml and 5 on xml-parsing and 2 on libxml2. For HTML, theres 18 on html, 2 on html-parsing, 1 on html-table, and 1 on html-to-text. For web scraping, there are 32 on web-scraping, and 3 on screen-scraping (which I don’t know if it’s the same thing).

I also did this using grep also but looking through the text of the rQAs data and got back 1159 unique questions involving XML, HTML, or web scraping out of 10004 questions of the 58096 total observations from the rQAs data.

5. What are the names of the R functions referenced in the titles of the posts?
To get the R functions, I tried to build a regular expression what would capture a word before a parenthesis. Particularly, just the first parenthesis because there might be some functions that people write ggplot(x, y) which would mess up the pattern. Basically im assuming the function looks like ggplot2(. I googled R functions and noticed some have periods like read.text() or underscore like geom_line(so I also made that into my regular expression. I thought about getting a huge list of R functions and somehow reading that into R to match up with the titles but that seemed unrealistic. Actually, in office hours Nick said we can get R functions using ls(“package:base”) and then checking if it’s a function with is.function(get(symbol)) if symbol is part of a function used in the sapply of all the symbols from the ls(“package:base”). This method seems too complicated. My regular expression became regex = "[A-Za-z_.]+[(]" and I got back about 230 R functions mentioned in the titles of my titles from Part 1 when I ran my function on 100 pages using the “r” tag out of 5000 total titles (since there aren’t titles of the posts in the rQAs data). Some of the R functions are:

```
[1] "png"                 "addADX"              "xy.coords"
 [4] "rfcv"                "GoogleFinanceSource"    "predict"
 [7] "as"                 "fitdistr"             "diff"
[10] ".getReactiveEnvironment" "storage.mode"          "sample"
[13] "arrange"             "scale"                "predict"
[16] "o.init"              "aes_string"            "spread"
```

6. **What are the names of the R functions referenced in the accepted answers and comments of the posts?**

I used the rQAs data and first subsetted the “answer” and “comment” from the type column of the data. Then, I used the same regular expression as I did in the previous problem to match a R function. I then used stringr’s str_extract_all to actually retrieve the matched patterns and then used an sapply to make the ones that didn’t match, as a NA. For the names of the R functions form the 11742 answers of the posts, I got back 10897 functions and some of them include:

```
[9993] "factor"                 "predict"
 [9995] "subset"                 "system"
 [9997] "read.table"              "guide_legend"
 [9999] "library"                "set.seed"
```

and for the 36350 comments of the post, I got back 8735 R functions and some include:

```
 [8729] "gsub"                   "fread"
[8731] "cumsum"                  "as.formula"
[8733] "as.formula"              "str"
[8735] "data"
```

Conclusion

Web scraping is used to extract information from websites and using R, can be done fairly easily. Basically download a webpage, parse it, then identify the xpath to a tag in an XML document combined with maybe a few regular expressions, and you can get back important information!

Code Appendix

```
# Assignment 6
# tiffanychen

# PART 1
# GET TAGS so i can use in part 3!
# axis::nametest[predicate]/axis::nametest[predicate]/...
# make the ones that xpath doesn't work as NA

# read in the data from the webpage
library(RCurl); library(XML)
url = "http://stackoverflow.com/questions/tagged/r?sort=newest&pagesize=50"
u = getURLContent(url)
doc = htmlParse(u, asText = TRUE)

# who posted it
path = "//div[@class = 'user-details']/a/text()"
nodes = getNodeSet(doc, path)
users = sapply(nodes, xmlValue)

# this is one line
```

```
users = xpathSApply(doc, path, xmlValue)

# to deal with missing values
path = "//div[@class = 'user-details']"
nodes = getNodeSet(doc, path)
# from piazza to add NA
users = xmlSApply(nodes, function(x) xmlValue(xmlChildren(x)$a))

# the title of the post
path = "//div[@class = 'summary']/h3/a/text()"
titles = getNodeSet(doc, path)
alltitles = sapply(titles, xmlValue)
# this is one line
alltitles = xpathSApply(doc, path, xmlValue)

##  when it was posted,
path = "//div[@class = 'user-action-time']/span/@title"
# when = getNodeSet(doc, path)
# sapply(when, xmlValue)
dates = xpathSApply(doc, path, `[[`, 1)

## the reputation level of the poster,
path = "//div[@class = '-flair']"
# replevel = xpathSApply(doc, path, xmlValue)
nodes = getNodeSet(doc, path)
rep = xmlSApply(nodes, function(x) xmlValue(xmlChildren(x)$span))

# the current number of views for the post,
path = "//div[@class = 'views ']"
nodes = getNodeSet(doc, path)
xmlSApply(nodes, function(x) xmlValue(x))
views = xpathSApply(doc, path, xmlValue)
# to remove the word "views"
views1 = gsub("[a-z\n\r ]", "", views)
views1
length(views1)

# the current number of answers for the post,
path = "//div[@class = 'status answered-accepted' or @class = 'status unanswered' or @class =
'status answered']/strong"
node = getNodeSet(doc, path)
ans = sapply(node, xmlValue)
# one line
ans = xpathSApply(doc, path, xmlValue)
```

```
# the vote "score" for the post,
path = "//span[@class = 'vote-count-post ']" # space!!
node = getNodeSet(doc, path)
score = sapply(node, xmlValue)
# one line
votes = xpathSApply(doc, path, xmlValue)

# the URL for the page with the post, answers and comments,
path = "//div[@class = 'summary']/h3/a/@href"
urls = xpathSApply(doc, path, `[[`, 1)
url = "http://stackoverflow.com" # cus getRelativeURL() didnt work
# so i paste it manually
urls = paste(url, urls, sep = "")
names(urls) = NULL


# the id (a number) uniquely identifying the post.
path = "//div[@class = 'question-summary']/@id"
node = getNodeSet(doc, path)
ids = xpathSApply(doc, path, `[[`, 1)
# regular ex and gsub to get just the numerical id
ids1 = gsub("[[:punct:][:alpha:]]", "", ids)
ids1

# tags
path = "//div[starts-with(@class, 'tags') ]"
node = getNodeSet(doc, path)
tags = xpathSApply(doc, path, xmlValue, trim = TRUE)
all_tags = gsub(" ","; ", tags)

# part 1
# ~3-9 small functions for each item
# 1 function for scraping a page
# 1 function for scraping all pages
# ~300 lines of code
# half an hr to scrape all pages. build url

### SMALL functions ####
users = function(doc){
  path = "//div[@class = 'user-details']"
  nodes = getNodeSet(doc, path)
  users = xmlSApply(nodes, function(x) xmlValue(xmlChildren(x)$a))
  data.frame(user = users)
```

```
}

dates = function(doc){
  dates = xpathSApply(doc, "//div[@class = 'user-action-time']/span/@title", `[[`, 1)
  data.frame(date = dates)
}

titles = function(doc){
  alltitles = xpathSApply(doc, "//div[@class = 'summary']/h3/a/text()", xmlValue)
  data.frame(title = alltitles)
}

rep = function(doc){
  path = "//div[@class = '-flair']"
  nodes = getNodeSet(doc, path)
  replevel = xmlSApply(nodes, function(x) xmlValue(xmlChildren(x)$span))
  data.frame(reputation = replevel)
}


views = function(doc){
  views = gsub("[a-z\n\r ]", "", xpathSApply(doc, "//div[@class = 'views ']/text()", xmlValue))
  data.frame(views = views)
}


ans = function(doc){
  ans = xpathSApply(doc, path = "//div[@class = 'status answered-accepted' or @class = 'status
unanswered' or @class = 'status answered']/strong", xmlValue)
  data.frame(answers = ans)
}


votes = function(doc){
  votes = xpathSApply(doc, "//span[@class = 'vote-count-post ']", xmlValue)
  data.frame(votes = votes)
}

url = function(doc){
  path = "//div[@class = 'summary']/h3/a/@href"
  urls = xpathSApply(doc, path, `[[`, 1)
  urls = paste("http://stackoverflow.com", urls, sep = "")
  data.frame(url = urls)
}
```

```
id = function(doc){
  ids1 = gsub("[[:punct:][:alpha:]]", "", xpathSApply(doc, "//div[@class = 'question-
summary']/@id", `[[`, 1))
  data.frame(id = ids1)
}

tags = function(doc){
  path = "//div[starts-with(@class, 'tags') ]"
  node = getNodeSet(doc, path)
  tags = xpathSApply(doc, path, xmlValue, trim = TRUE)
  all_tags = gsub(" ","; ", tags)
  data.frame(tags = all_tags)
}

# function that does one page
# calls the SMALL functions!
one_page = function(i){
  cbind(id(i), dates(i), tags(i),
        titles(i), url(i), views(i),
        votes(i), ans(i), users(i), rep(i))
}

getNextURL =
  function(doc)
  {
    nxt = unique(unlist(getNodeSet(doc, "//a[@rel = 'next']/@href")))
    # getRelativeURL(nxt, url)
    paste("http://stackoverflow.com", nxt, sep = "")
  }

## function
# takes in tag in quotes and
# number of pages you want to process
stackoverflow = function(tag, n){

  # build the url depending on the tag
  base_url = "http://stackoverflow.com/questions/tagged/"
  url = paste(base_url, tag, sep = "")
  url = paste(url, "?sort=newest&pagesize=50", sep = "")
  u = getURLContent(url)
  doc = htmlParse(u, asText = TRUE)

  ans = NULL
  # from page 1 to n (specified by user)
  for(i in 1:n){
    # calls function one_page()
    onepage = one_page(doc)

    # since it goes onto next page
    # by calling getNextURL()
    # redo the reading in url process
    u = getNextURL(doc)
    u = getURLContent(u)
    doc = htmlParse(u, asText = TRUE)

    # rbind each page's results to eachother
    ans = rbind(ans, onepage)
  }
  return(ans)
}

y = stackoverflow("r", 100)

## all the pages
stackoverflow = function(tag){
  base_url = "http://stackoverflow.com/questions/tagged/"
  url = paste(base_url, tag, sep = "")
  url = paste(url, "?sort=newest&pagesize=50", sep = "")
  u = getURLContent(url)
  doc = htmlParse(u, asText = TRUE)

  ans = NULL
  while(TRUE){
    onepage = one_page(doc)
    # browser()
    u = getNextURL(doc)
    if(length(u) == 0)
      break
    u = getURLContent(u)
    doc = htmlParse(u, asText = TRUE)

    ans = rbind(ans, onepage)
  }
  return(ans)
}

# =======================================================
## small things we need to take care of!!!! ##
# deleted users, community wiki posts, using contains() for "status" "views"

# if user is deleted
# just set reputation as NA
library(curl)
library(xml2)

xpath = "//div[@class = 'user-details'"
url = "http://stackoverflow.com/questions/tagged/"
html = read_html(url)
user_details = xml_find_all(html, xpath)

# xml_length()
n_child = sapply(user_details, xml_length)
n_child == 6

## use starts with views  "views hot"
# to deal with views that are high like 13k
# and status/number of answers changing class
# theres 2 usernames per person

# ====================================================
# PART 2
lapply(y$url, stackoverflow("r", 10))

url = "http://stackoverflow.com/questions/34162997/how-to-add-tool-tip-and-animations-in-
bubbles-chart-in-r-shiny"
u = getURLContent(url)
doc = htmlParse(u, asText = TRUE)

# user
users = function(doc){
  path = "//div[@class = 'user-details']"
  nodes = getNodeSet(doc, path)
  users = xmlSApply(nodes, function(x) xmlValue(xmlChildren(x)$a))
  data.frame(user = users)
}

# userid
usersid = function(doc){
  path = "//div[@class = 'user-details']/a"
  nodes = getNodeSet(doc, path)
  usersid = xmlSApply(nodes, function(x) xmlValue(xmlChildren(x)$href))
  data.frame(userid = usersid)
}

# date
date = function(doc){
  path = "//p[@class = 'label-key']/@title"
  dates = xpathSApply(doc, path, `[[`, 1)
  data.frame(date = dates)
}

# users rep
rep = function(doc){
  path = "//div[@class = '-flair']"
  nodes = getNodeSet(doc, path)
  rep = xmlSApply(nodes, function(x) xmlValue(xmlChildren(x)$span))
  data.frame(rep = rep)
}


# the score/votes for this entry
votes = function(doc){
  votes = xpathSApply(doc, "//span[@itemprop = 'upvoteCount']", xmlValue)
  data.frame(votes = votes)
}

# the HTML content as a string for this entry
doc = htmlParse(u, asText = TRUE)

# the identifier of the "parent" entry,
# i.e., which entry this is a response to - a comment to an answer, or an answer to a post,


# the unique identifier for the overall post.
postid = function(doc){
  path = "//h1[@itemprop = 'name']/a/@href"
  id = xpathSApply(doc, path, `[[`, 1)
  library(stringr)
  actualid = str_extract_all(id, "[0-9]*")

}

# the type of entry (post, answer, comment)

### ===============================================
#### PART 3
# rQAs data
load("/Users/tiffanychen/Downloads/rQAs.rda")
rqas = rQAs
rownames(rqas)
# user: username
# userid: unique user id
# date: date
# reputation: user reputation
# score: number of votes
# text: message they posted
# type: question, comment, or answer
# parent: unique id of parent post
# id: unique id of this post
# qid: unique id of related question

# Q1
# What is the distribution of the number of questions each person answered?
# from piazza..
# try to have the x axis contain amount of answers
# y axis the frequency of users who posted that many times
# 5 people (y) posted 2 answers (x)
answers = rqas[rqas$type == "answer", ]
count = table(answers$user)
doublecount = table(count)
plot(doublecount, main = "Distribution of Number of Q's Each Person Answered", xlab =
"Amount of Answers", ylab = "Count of users who Posted x times")

# lets make it look a little better
plot(doublecount, main = "Distribution of Number of Q's Each Person Answered", xlab =
"Amount of Answers", ylab = "Count of users who Posted x times",  xlim = c(1, 150), type = "l",
ylim = c(0, 400))

## Q2
# What are the most common tags?
# using scraped 100 pages, with 50 posts per page
y = stackoverflow("r", 100) # use my data from part 1
tags = y$tags

# convert into character
# since strsplit only works with character
ctags = lapply(tags, as.character)
z = unlist(ctags)
```

```
# want the sorted table
# of individual tags
# in decreasing order
tagstable = head(sort(table(unlist(strsplit(z, "; "))), decreasing = TRUE))

tagstableall = table(unlist(strsplit(z, "; ")))

## Q3
# How many questions are about ggplot?
tagstableall[grep("ggplot", rownames(tagstableall)) ]

# look at the questions
q = rqas[rqas$type == "question", ]
# text with word ggplot
text = q$text
length(unique(grep("ggplot", text))) #  956

## Q4
# How many questions involve XML, HTML or Web Scraping?
tagstableall[grep("xml|html|web-scraping|scraping", rownames(tagstableall), ignore.case =
TRUE) ]

length(unique(grep("xml|html|web scraping|scraping", text)))
# 1159

## Q5
# What are the names of the R functions
# referenced in the titles of the posts?
titles = y$title
library(stringr)
text = "func.tion() this is a reg_ex()" # testing

regex = "[A-Za-z_.]+[(]"
result = str_extract_all(titles, regex)
result
result = sapply(result, function(t) {
  if (length(t) >0)t[1]
  else NA
})

funcs = result[!is.na(result)] # names of the R functions
substr(funcs, 1, nchar(funcs)-1)

titlecount = length(result) - sum(is.na(result))
```

```
titlecount

## Q6
# What are the names of the R functions referenced
# in the accepted answers and comments of the posts?
# use rQAs data

answers = rqas[rqas$type == "answer", ]
comments = rqas[rqas$type == "comment", ]

regex = "[A-Za-z_.]+[(]"
result = str_extract_all(answers$text, regex)
result
result = sapply(result, function(t) {
  if (length(t) >0)t[1]
  else NA
})

a_funcs = result[!is.na(result)]
substr(a_funcs, 1, nchar(a_funcs)-1)


regex = "[A-Za-z_.]+[(]"
result = str_extract_all(comments$text, regex)
result
result = sapply(result, function(t) {
  if (length(t) >0)t[1]
  else NA
})

comment_func = result[!is.na(result)]
substr(comment_func, 1, nchar(comment_func)-1)
```