APPENDIX

Part I:

```
#top-level function
webScraping = function(url, pageLimit = Inf){
  #create empty list for storing data frames
  storage = list()

  pageCount = 1
  while(TRUE){
    html = read_html(url)

    #store each page data frame into a list
    storage[[pageCount]] = getDataframe(html, url)

    #stop when page limit is reached
    if(pageCount == pageLimit){
      break
    }else{
      #update the url to the next page
      url = nxt_button(html, url)
      pageCount = pageCount + 1
    }
  }
  #combine the list of data frames
  return(ldply(storage, data.frame))
}


########################################

########################################
#gets one large dataframe for username, time posted, etc.
getDataframe = function(html, url){

  urnm = username(html)
  tm = time(html)
  ttl = title(html)
  rep = reput(html)
  vw = views(html)
  ans = answers(html)
  vt = votes(html)
  purl = post_url(html, url)
  pid = post_id(html)
  tgs = tags(html)

  #bind the functions into one data frame
  data = cbind(tm, ttl,vw, ans, vt, purl, pid, tgs, urnm, rep)
  return(data)
}

########################################
```

continuation

```
#####################################
#gets the user ID
username = function(html){
  path = "//div[contains(@class, 'user-info')]"
  user_info = xml_find_all(html, path)

  #find all valid posts
  valid = sapply(user_info, function(node){
    path = "./div[@class = 'user-details']"
    details = xml_find_all(node, path)

    #find children of each node
    children = sapply(details, xml_length)
    if(length(children) > 1){
      #accounts for community posts
      return(FALSE)
    }
    else{
      if(children > '2' & children < '7'){
        #accounts for anonymous posts
        return(TRUE)
      }else{
        return(FALSE)
      }
    }
  })
  #makes path from user-info to users
  path = "./div[@class = 'user-details']/a[contains(@href, 'users')]"

  #make data frame with all NAs
  scrape = data.frame(user_name = rep(NA,length(user_info)))

  #get the username for all valid posts
  scrape$user_name[valid]=
    xml_text(xml_find_one(user_info[valid], path))
  return(scrape)
}
#####################################


#####################################
#gets the time posted
time = function(html){
  path = "//div[contains(@class, 'user-info')]"
  user_info = xml_find_all(html, path)

  #find valid posts (not ommunity, anonymous, or deleted)
  valid = sapply(user_info, function(node){
    path = "./div[@class = 'user-details']"
    details = xml_find_all(node, path)
    i = sapply(details, xml_length)
    if(length(i) > 1){
      #community post
      return(FALSE)
    }
    else{
      #anonymous or deleted poster
      return(TRUE)
    }
  })
  path = "./div[@class = 'user-action-time']/span/@title"
  time = data.frame(times = rep(NA,length(user_info)))

  #get the username for all valid posts
  time$times[valid]=
    xml_text(xml_find_one(user_info[valid], path))
  return(time)
}
#####################################

#####################################
#gets the title of the post
title = function(html){
  path = "//div[@class = 'summary']/h3/a/text()"
  post_title = xml_find_all(html, path)
  post_title = xml_text(post_title)
  data.frame(post_title = post_title)
}
#####################################
```

continuation

```
####################################
#gets the reputation level of poster
reput = function(html){
  path = "//div[contains(@class, 'user-info')]"
  user_info = xml_find_all(html, path)

  #get valid posts (not community, anonymous, or deleted)
  valid = sapply(user_info, function(node){
    path = "./div[@class = 'user-details']"
    details = xml_find_all(node, path)
    i = sapply(details, xml_length)
    if(length(i) > 1){
      #community post
      return(FALSE)
    }
    else{
      if(i > '2' & i < 7){
        return(TRUE)
      }else{
        #anonymous or deleted poster
        return(FALSE)
      }
    }
  })

  reputation = sapply(user_info[valid], function(node){
    #get the reputation score
    path = "./div[@class = 'user-details']/span[@class = 'reputation-score']"
    subset = xml_find_all(node, path)
    rep = xml_text(subset)

    #replace every instance of "k" in text by going to another path to get rep.
    rx = "[k]"
    if(grepl(rx, rep) == TRUE){
      path1 = "./div[@class = 'user-details']/span[contains(@title, 'reputation')]/@title"
      krep = xml_text(xml_find_all(node, path1))
      rep = gsub("[, a-z]+", "", krep)
      return(rep)
    }else{
      #if there is no "k", clean up text
      rep = gsub("[,]", "", rep)
      return(rep)
    }

  })
  scraper = data.frame(repu = rep(NA,length(user_info)))
  scraper$repu[valid] = reputation
  return(scraper)
}
####################################


####################################
#gets view counts of post
views = function(html){
  #accounts for class change when views reach 1k+
  path = "//div[contains(@class, 'views')]/@title"
  views = xml_find_all(html, path)
  views = xml_text(views)

  #removes all instances of commas, letters, space
  views = gsub("[, a-z]+", "", views)
  data.frame(views = views)
}
####################################

####################################
#current number of answers for post
answers = function(html){
  path = "//div[contains(@class, 'status')]/strong/text()"
  answers = xml_find_all(html, path)
  answers = xml_text(answers)
  data.frame(answers = answers)
}
####################################
```

continuation

```
#####################################
#votes count for the post
votes = function(html){
  path = "//span[contains(@class, 'vote-count-post')]/strong/text()"
  votes = xml_find_all(html, path)
  votes = xml_text(votes)
  data.frame(votes = votes)
}
#####################################

#####################################
#url for the page with the post
post_url = function(html, url){
  path = "//div[@class = 'summary']/h3/a/@href"
  post_urls = xml_find_all(html, path)
  post_urls = xml_text(post_urls)
  post_urls = url_absolute(post_urls, url)
  data.frame(post_urls = post_urls)
}
#####################################

#####################################
#the id uniquely identifying the post
post_id = function(html){
  path = "//div[contains(@id, 'question-summary-')]/@id"
  post_id =  xml_find_all(html, path)
  post_id = xml_text(post_id)

  #remove all instances of letters and dashes
  post_id = gsub("[-a-z]+", "", post_id)
  data.frame(post_id = post_id)
}
#####################################


#####################################
#get the tags for each post
tags = function(html){
  path = "//div[starts-with(@class, 'tags')]"
  tag_nodes = xml_find_all(html, path)

  #get the tag for each post
  all_tags = sapply(tag_nodes, function(node){
    path = "./a[@class = 'post-tag']/text()"
    tags = xml_find_all(node, path)
    tags = xml_text(tags)

    #combine all tags into one string
    #so they can fit in one column
    paste0(tags, collapse = " ")
  })
  data.frame(tags = all_tags)
}
#####################################

#####################################
#the next button
nxt_button = function(html, url){
  path = "//a[@rel = 'next']/@href"
  next_url = xml_text(xml_find_one(html, path))
  return(url_absolute(next_url, url))
}
```

Part 3:

1)

```
#Number 1
load("rQAs.rda")

rqas = rQAs
#subset by type = answer
ans = rqas[rqas$type == "answer",]
times_answered = table(sort(table(ans$userid)))

#plot the distribution of answers
barplot(times_answered, xlab = "Number of Answers",
        ylab = "Freq. of People who Answered",
        main = "Bar Graph of The Number Qs People Answered")
```

2)

```
#NUMBER 2
#Using the data frame created from part 1
ques = first_2000_pages$tags

#split each tag string into words
split_tags = sapply(ques, function(each_post){
  one_post = toString(each_post)
  strsplit(one_post, "[ ]")
})

#unlist all the tags and get the most common
unlst_tags = unlist(split_tags)
sort(table(unlst_tags))
```

3)

```
#NUMBER 3
#Subset by posts that are questions
question = rqas[rqas$type == "question",]

#Find how many times "ggplot" was in the text
table(grepl("ggplot", question$text, ignore.case = TRUE))
```

4)

```
#NUMBER 4
#Subset by posts that are questions
question = rqas[rqas$type == "question",]

#get tags to search for
search_tags = c("XML", "HTML", "Web Scraping")

#find how many times the tags occur in the text
find_tags = lapply(search_tags, function(one_tag){
  table(grepl(one_tag, question$text, ignore.case = TRUE))
})
```

5)

```
#NUMBER 5
URL = rownames(rqas)

#get the title from url
get_title = sapply(strsplit(URL, "/"), tail, 1)
ttl = data.frame(title = get_title)

#separate the words in the title
separate_words = sapply(ttl, function(one_title){
  str_title = toString(one_title)
  remove_dash = strsplit(str_title, "-")
})
tst = unlist(separate_words)

#get the most common word functions
common_func = names(tail(sort(table(tst)), 2000))

#get all base functions
base_functions = ls(getNamespace("base"), all.names=TRUE)

#find how many times base functions occur in the most common words in title
find_func = lapply(base_functions, function(one_func){
  table(grepl(one_func, common_func, ignore.case = TRUE))
})
```