loadAllArticles.R source code

```
#Project 1 R program to crawl, parse and extract all articles published in a specific journal
#Group 6:Duyen Ngyen, Bicheng Xiao, Shubham Gulia
#2017 Fall

#Journal Name: BMC Medical Genetics
#Total article number: 1642
#Main page of the journal: https://bmcmedgenet.biomedcentral.com/

source(paste(getwd(),"/R_Scripts/util.R",sep=""))
library(bitops)
library(RCurl)
library(XML)
library(stringr)

#load(readLines) main page & save to (.html) file
site.url = "https://bmcmedgenet.biomedcentral.com/"
site.url.clear = "https://bmcmedgenet.biomedcentral.com"
main.page = readLines(site.url)
options(warn=-1)
dir.create("HTMLs")
#write(main_page, file = "HTMLs/main_page.html")
#print("[I/O]: FILE: main_page.html created.")

#get the link of "Article List Page" URL from main page
page.list.url = paste(site.url,substr(main.page[grep("(<a
class=.navbar__link.+href).+(>Articles<\\/a>)",main.page)],72,79),"/",sep="")
page.list = readLines(page.list.url)
write(page.list, file="HTMLs/article_list_page.html")
print("[I/O]: FILE: article_list_page.html created.")

#get the URL of all "Article list page" url, total 66 pages, each page has maximum 25 articles
#find max page numbers
page_index = str_extract_all(page.list[grep('<span class=\\"Control_name\\" data-
test=\\"pagination-text\\">Page \\d+ of \\d+<\\/span>',page.list)][1],"\\d+")[[1]];
min.page.count = page_index[1];
max.page.count = page_index[2];
page.url = paste(site.url, str_extract(page.list[grep('<a class="Pager Pager--
next".+',page.list)][1],"articles\\?.+page\\="),sep="")
page.url = xpathApply(htmlParse(page.url, asText=TRUE),"//body//text()", xmlValue)[[1]]

#generate article url list
page.url.list = ""
article.data = data.frame(DOI=c(),url=c())
for(i in min.page.count:max.page.count){
  percent = toString(as.integer((i/as.integer(max.page.count))*100))
  cat("\r",paste("[APP]: loading page number: ",i," [", percent, "%]"))
  full.url = paste(page.url,i,sep="")
  articleUrl_and_doi = loadArticleList(full.url)#function call: loadArticleList()
  articleUrl_and_doi$url = paste(site.url.clear, articleUrl_and_doi$url, sep="")#add site url, to fulfill
the url of an article
  article.data = rbind(article.data, articleUrl_and_doi)#rown bind, store all article DOI, urls
  page.url.list = c(page.url.list, full.url)
}
page.url.list = page.url.list[-1]
write(page.url.list, "page.url.list.txt")
print("[I/O]: FILE: page.url.list.txt created.")
write.csv(article.data, "article.DOI.URL.list.csv")
print("[I/O]: FILE: article.DOI.URL.list.csv created.")


#circuly analysis the articles and extract required information, and form all information in a
data.frame
extracted.data = data.frame(DOI=c(),Title=c(),Author=c(), "Author Affiliation"=c(), "Corresponding
Author"=c(), "Corresponding_Author_email"=c(),
                    "Publication Date"=c(), Abstract=c(), Keywords=c(), "Full Text"=c())
total.number = as.integer(length(article.data[,1]))
for(i in 1:total.number){
  extracted.data = rbind(extracted.data, analysisArticle(article.data[i,1], article.data[i,2]))#function
call: analysisArticle
  cat("\r", paste("[I/O]: FILE:", toString(i), article.data[i,1], ".html created.
[",toString(as.integer(i/total.number*100)), "%]"))
}

#Write the final result to BMC Medical Genetics.txt
options(warn=-1)
dir.create("output")
write.table(extracted.data,"output/BMC Medical
Genetics.txt",sep="\t",row.names=FALSE,fileEncoding = "UTF-8")
print("[I/O]: FILE: output/BMC Medical Genetics.txt created.")
```

util.R source code

```
#Project 1 R program to crawl, parse and extract all articles published in a specific journal
#Group 6:Duyen Ngyen, Bicheng Xiao, Shubham Gulia
#2017 Fall

#Journal Name: BMC Medical Genetics
#Total article number: 1642
#Main page of the journal: https://bmcmedgenet.biomedcentral.com/

library(bitops)
library(RCurl)
library(XML)
library(stringr)

#FUNCTION NAME: loadArticleList
#FUNCTION:extract DOI and url of article lists in the specified page
#INPUT:[page_url:the url of article list page]
#OUTPUT:[data.frame(DOI,URL:url of article full text)]
loadArticleList = function(page_url){
  html = getURL(page_url, followlocation=TRUE)
  doc = htmlParse(html, asText=TRUE)
  article.list = xpathSApply(doc,"//div[@class='ResultsList_group']/h3/a",xmlGetAttr,"href")
  DOI.list = xpathSApply(doc,"//div[@class='ResultsList_group']/h3/a",xmlGetAttr,"data-event-
label")
  return(data.frame(DOI=DOI.list,url=article.list))
}


#FUNCTION NAME: analysisArticle
#FUNCTION:extract all required field from the specified article full text page
#INPUT:[DOI,URL:url of article full text]
#OUTPUT:[data.frame(DOI, title, author, authorAffiliation, correspondingAuthor,
#                correspondingAuthorEmail, publicationDate, abstract, keywords, fullText)], single row
analysisArticle = function(DOI, url){
  html = getURL(url, followlocation=TRUE)
  doc = htmlParse(html, asText=TRUE)
  options(warn=-1)
  dir.create("HTMLs")
  article.filename = paste("HTMLs/", gsub("\\/","",DOI),".html",sep="")#generate article filename:
DOI.html
  saveXML(doc, article.filename)#save to file

  author.info = extractAuthors(doc)#function call:extractAuthors()
  DOI = DOI;
  title = extracAttribute(doc, "//meta[@name='citation_title']","content")
  author = author.info[1]
  authorAffiliation = extracAttribute(doc, "//meta[@name='citation_author_institution']","content")
  correspondingAuthor = author.info[2]
  correspondingAuthorEmail = author.info[3]
  publicationDate = extracAttribute(doc, "//meta[@name='prism.publicationDate']","content")
  abstract = extract(doc, "//div[@class='AbstractSection']")
  keywords = extract(doc, "//span[@class='Keyword']")
  fullText = extract(doc, "//div[@id='Test-ImgSrc']")

  extracted.data.row = data.frame(DOI, title, author, authorAffiliation, correspondingAuthor,
                         correspondingAuthorEmail, publicationDate, abstract, keywords, fullText)
  return(extracted.data.row)
}


#FUNCTION NAME: extract
#FUNCTION:extract xmlValue from specified XML tag/node
#INPUT:[parsedHtml, pattern]
#OUTPUT:[string of xmlValue embedded in the tag/node]
extract = function(parsedHtml, pattern){
  resultList = xpathSApply(parsedHtml, pattern, xmlValue)
  if(length(resultList) == 0) return("NULL")
  result = ""
  for(str in resultList){
    if(str == ""|str ==" ") next
    if(result == ""){
      result = str
    } else {
      result = paste(result, ";", str, sep="")
    }
    result = str_replace(result,"\\n","")
  }
  return(result)
}


#FUNCTION NAME: extracAttribute
#FUNCTION:extract XML attribute value from specified XML tag/node and attribute name
#INPUT:[parsedHtml, pattern, attribute]
#OUTPUT:[string of attribute value]
extracAttribute = function(parsedHtml, pattern, attribute){
  resultList = xpathSApply(parsedHtml, pattern, xmlGetAttr, attribute)
  if(length(resultList) == 0) return("NULL")
  result = ""
  for(str in resultList){
    if(str == ""|str ==" ") next
    if(result == ""){
      result = str
    } else {
      result = paste(result, ";", str, sep="")
    }
    result = str_replace(result,"\\n","")
  }
  return(result)
}

#FUNCTION NAME: extractAuthors
#FUNCTION:extract author, corresponding author, corresponding author's email
#INPUT:[parsedHtml]
#OUTPUT:[vector(author, corresponding.author, email)]
extractAuthors = function(parsedHtml){
  #data.frame : (author, affliationId, isCorresponding, email)
  author.data = data.frame(author=c(), affliationId=c(), isCorresponding=c(), email=c())
  author.list = xpathSApply(parsedHtml, "//li[@class='Author']", xmlDoc)
  for(authorHTML in author.list){
    author = extract(authorHTML, "//span[@class='AuthorName']")
    affliationId = extract(authorHTML, "//a[@class='AffiliationID']")
    email = str_replace_all(extracAttribute(authorHTML, "//a[@class='EmailAuthor']",
"href"),"mailto:","")
    isCorresponding = !(email == "NULL")
    author.data.row = data.frame(author, affliationId, isCorresponding, email)
    author.data = rbind(author.data, author.data.row)
  }
  author.filtered.list = author.data[which(author.data$isCorresponding == FALSE),]
  author = ""
  for(i in 1:length(author.filtered.list[,1])){
    if(i == 1){
      seperator = ""
    } else {
      seperator = ";"
    }
    author = paste(author, seperator, author.filtered.list[i,1],sep="")
  }

  corresponding.author.list = author.data[which(author.data$isCorresponding == TRUE),]
  corresponding.author = ""
  email = ""
  for(i in 1:length(corresponding.author.list[,1])){
    if(i == 1){
      seperator = ""
    } else {
      seperator = ";"
    }
    corresponding.author = paste(corresponding.author, seperator,
corresponding.author.list[i,1],sep="")
    email = paste(email, seperator, corresponding.author.list[i,4],sep="")
  }
  return(c(author, corresponding.author, email))
}

#FUNCTION NAME: extractAffliation
#FUNCTION:extract affliationID and affliationText
#INPUT:[parsedHtml]
#OUTPUT:[data.frame(affliationID, affliationText)]
#WARNING: it is not being used in this project, but if you need to comply the author and their
affliations, this dunction will be helpful
extractAffliation = function(parsedHtml){
  #data.frame : (affliationId, affliationText)
  affliation.data = data.frame(affliationId=c(), affliationText=c())
  affliation.list = xpathSApply(parsedHtml, "//div[@class='Affiliation']", xmlDoc)
  for(affliatioHTML in affliation.list){
    affliationId =gsub("\\(|\\)", "", extract(affliatioHTML, "//span[@class='AffiliationNumber']"))
    affliationText = extract(affliatioHTML, "//div[@class='AffiliationText']")
    affliation.data.row = data.frame(affliationId, affliationText)
    affliation.data = rbind(affliation.data, affliation.data.row)
  }
  return(affliation.data)
}
```