Top used libraries and functions for cleaning data revealed from big data analysis of Coursera courses

Siruo Wang

Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health, 615 North Wolfe Street, Baltimore, MD 21205, USA

Introduction

Coursera is an educational online platform which contains thousands of courses taught by instructors from universities and institutions around the world. Components of Coursera courses include but not limited to online recorded video lectures, graded assignments, and discussion forums open to the community. The goal of Coursera is to provide a low cost, open access, and short time training platform to enhance education in specific areas. Among the many courses being taught in the Coursera platform, data analysis is one of the popular areas as data analysis skills are required more often when students are looking for job openings.

In the Coursera platform, tens of thousands of students are taking data analysis online classes. Each of the data analysis class must contain a fundamental but crucial part: learn to get and clean data. Since majority of the data analysis needs to be performed on a tidy dataset, students who are taking these classes are required to submit a final project, Getting and Cleaning Data Project, to their GitHub repository. Although students share the same goal of preparing a tidy dataset that can be used for further data analysis, they have different preferences in providing methods to complete the ultimate goal. Therefore, we want to explore all students' R scripts for completing the goal, and summarize main sources of variations in solution to the completion of the Coursera Project.

In Methods section 1, we outline procedures of retrieving all available repository names for the Getting and Cleaning Data Project from GitHub and each students' R script for performing the analysis under the whole set of repository names. In Methods section 2, we outline the detailed steps of extracting library names and function names, and we fit a linear model to explore the relationship between library usage and number of lines of code for each student's R script. In Results section, we demonstrate the top used libraries and functions in the way of students completing the project, and compare them to the most downloaded libraries in R. We also show trends of the library and function usages over time, and make inferences for which libraries contribute to shorter length of code. In Discussion section, we discuss the limitations of our date collection method and potential further steps to continue the analysis on the Coursera Project.

Methods

1. Retrieving repository names and R scripts

As students are taking the data analysis classes on Coursera and constantly pushing their final projects to GitHub, the number of repository results increases every minute. Updated to October 7, 2017, there are currently over 31K repository results available under the Getting and Cleaning Data project. For each repository result, it is composed of a username and followed by the course title. We develop a system for automatically retrieving a complete set of available repository names on the GitHub website (See Supplemental Code Section 1) [1]. Our data scraping system is able to get around the GitHub API 1K access limit and retrieve repository names sorted by their creation dates in the window of 14 days starting from 2008-01-01 to 2017-09-25 [2].

Note that we use 2008-01-01 as our starting repository creation date to scrape project repository names

from GitHub. According to the instructor of the Coursera data analysis course, Jeffrey Leek, the data analysis Coursera course was created on April 2014 for the first time. So, we believe that majority of students should create the Getting and Cleaning Data Project repository and submit their final project to GitHub after the Coursera course was created and taught online. However, we notice the cases that some students created a repository even before the Coursera course starting date, and later changed the repository name to be the course name after taking the data analysis class. To include such cases, we decide to scrape repositories which were created after 2008-01-01, since we do not find many repositories created before 2008 for this project. It takes our system 4 hours of running to retrieve 20,824 repository names.

After we retrieve the 20,824 repository names from GitHub, we want to look closely to students' R scripts under their repository names. In order to successfully complete the data analysis class, students are required to submit the only final project R file named *run_analysis.R*, which contains their code to prepare a tidy data set. Therefore, we design another system to automatically scrape every single repository name and find student's final project R script under their repository name (See Supplemental Code Section 1) [1]. Besides handling GitHub API access limit issue, our code also provides a solution to the URL path access error caused by special characters existed in the URL to student's raw R file.

2. Performing analysis on R scripts

Since we retrieve the R scripts of the Getting and Cleaning Data project for all 20,824 repositories, we have date from the whole population. We are unable to conduct any statistical tests on the whole population, but we are able to do enough observational analysis on the R scripts to summarize main sources of variation in people completing the project.

Although there are tens of thousands people have taken the data analysis classes on Coursera, we would like to know the trend of when people create the repository on GitHub to take the class and how many of them actually complete the course. We plot the number of repositories created over time to observe the trend and count how many the final R scripts we have retrieved using our system. Since the final R script is required for the course, if we are unable to retrieve it under student's repository name, we consider that student drops out from the course at some point.

We are also interested to know people's preference of using comments and space lines. It is often the case that students tend to use many space lines to make their code look longer, but the pure code may not be as many as it appears. We calculate the number of comment lines and space lines that each student uses and compare the pure code length to the full code length for the whole population.

Preparing a tidy data set usually involves importing necessary libraries and applying useful functions, but people have different preferences on their library and function choices. It is interesting to know what libraries and functions are used more often than the others for the whole population. Therefore, we extract distinct library and function names from each students' final R scripts, count the frequency of usages across all students, and plot the usages trend over time. Note that people import the same library differently. For example, we handle the following cases the same as *library(dplyr)*:

library("dplyr"), library('dplyr'), ifelse(library(dplyr)), try(library(dplyr)), if(library(dplyr)), SuppressMessages(dplyr), suppressPackageStartupMessages(dplyr), suppressWarnings(dplyr)

To explore the relationship between code length and library usage, we fit a linear model on the length of pure code and top thirty used libraries. In the model, we use the log value of pure code length for each R script as response, and the appearance of top thirty used libraries in each R script as thirty covariates. We retrieved 19,544 final R scripts so we have 19,544 response values and corresponding covariates to build our model. Note that each covariate should be either 1 or 0 which indicates the library exists in the R script

or not accordingly. Similarly, we build another linear model for pure code length and top thirty used libraries.

Results

We observe that there are less than 20 repositories created before April 2014, which were the cases where students created repositories before the course started on April 2014 and changed the repository name to be the course name afterwards. After April 2014, there are more repositories created between Early-May 2014 to Mid-September 2014 and Early-January 2015 to Late-February 2015 than other months in 2014 and 2015. Thus, we conclude that students are more likely to take Coursera courses during their summer and winter breaks. Majority of repositories were created between April 2014 to the end of 2015. There is a distinct drop drown of the number of new repositories created after the end of 2015 until now (See Supplemental Figure 1 and Supplemental Code section 3).

We calculate the final project submission rate is 94%. There are 1211 out of 20,824 students who do not have the final R script submitted for the Getting and Cleaning Data Project, so there are at least 6% students dropping out from the course (See Supplemental Code section 4).

We compare the length of students' full R scripts to the length of pure text that have comment and space lines removed. The plot shows a clearer difference in mean before and after we remove the comment and space lines (See Supplemental Figure 2 and Supplemental Code section 5).

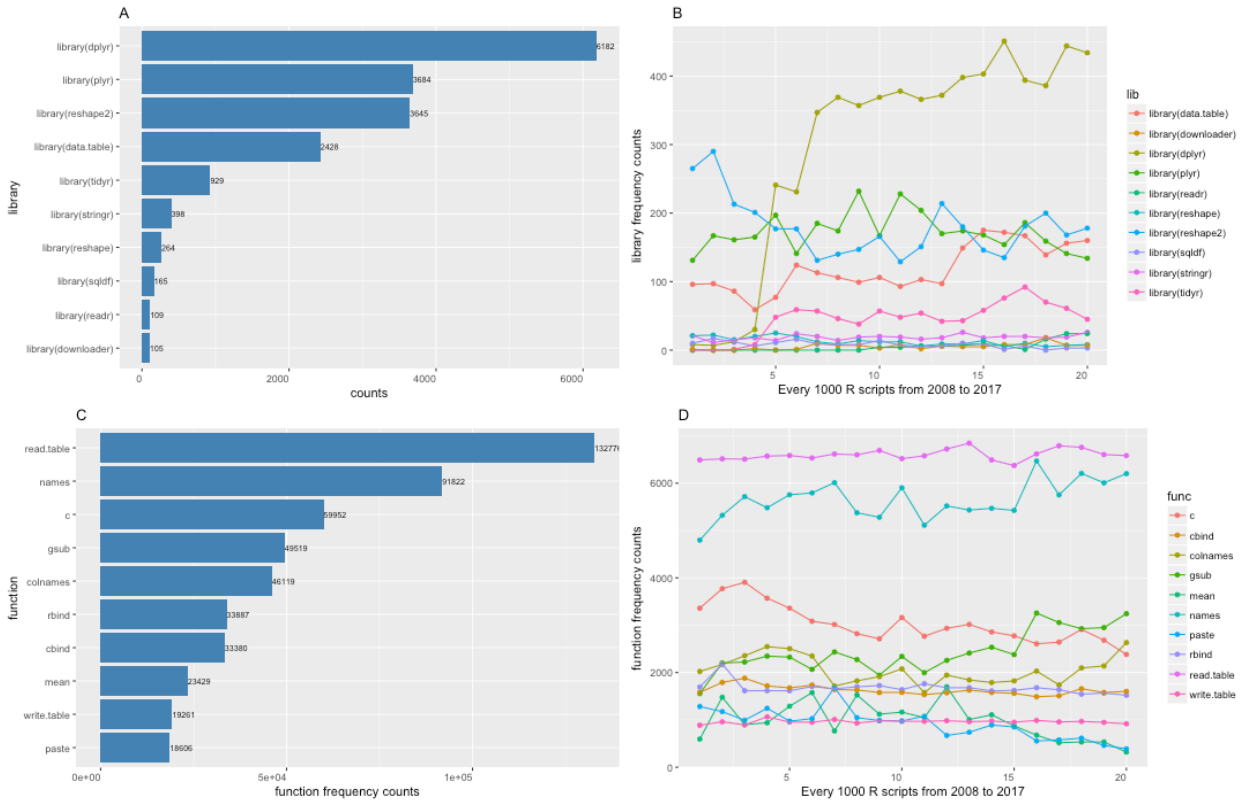

Figure 1 A and C shows the usage for top ten used libraries and functions accordingly. Figure 1 B and D shows the top ten used library usage trend and the top ten used function usage trend for every 1K students' R files from 2008 to 2017 accordingly.

We extract the total 156 unique libraries that are being used in students' final R scripts to complete the

Getting and Cleaning Data Project, and the top 10 used libraries are *dplyr, plyr, reshape2, data.table, tidyr, stringr, reshape, sqldf, readr, downloadr* (See Figure 1 A and Supplemental Code section 6). Six among those ten libraries, *stringr, dplyr, plyr, reshape2, data.table, readr*, are listed in the top 50 most downloaded R packages [3]. From the tread of library usage in Figure 1 B, for every one thousand R scripts, we observe that the usages of library *dplyr* and *data.table* increase over time while the usage of library *reshape2* slightly decreases over time compared to other top 10 used libraries (See Figure 1 B and Supplemental Code section 7).

The top ten used functions out of 10,006 unique functions across the whole population are *read.table, names, c, gsub, colnames, rbind, cbind, mean, write.table, paste* (See Figure 1 C and Supplemental Code section 8). Since all ten top used functions are base R functions instead of functions from particular package, we conclude that students are more likely to use base R functions to manipulate data rather than using R packages even though they import various libraries in their code. In addition, we observe that the top used functions are used consistently over time since the tread of library usage in Figure 1 D indicates no dramatic increase or decrease (See Figure 1 D and Supplemental Code section 9).

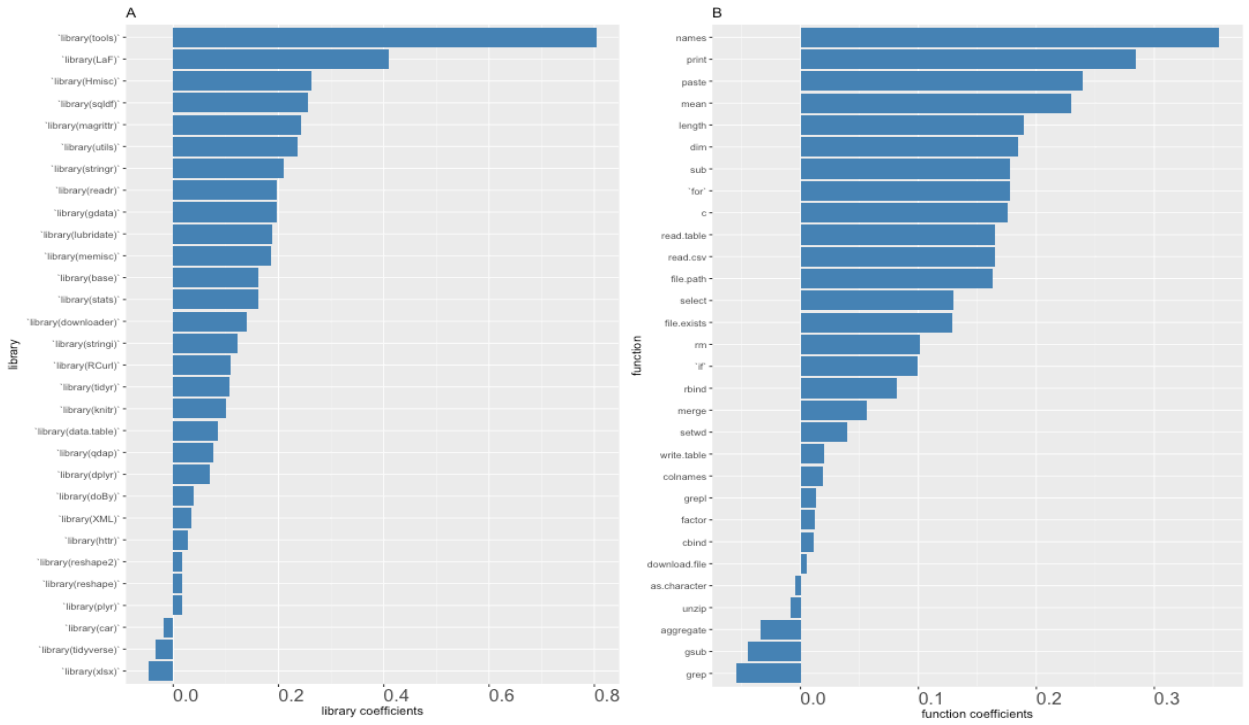

Figure 2 A and B plot the coefficients of top ten used libraries in the linear model and the coefficients of top ten used functions in the linear model separately on the left, and show the coefficients and p values table accordingly on the right.

Based on the statistically significant coefficients in the two linear models (See Supplemental Figure 3 and Supplemental Figure 4), we infer that students who tend to write longer code use libraries such as *tools, LaF, Hmisc, sqldf, magrittr, utils, stringr, readr, gdata, lubridate, downloader, tidyr, knitr* (See Figure 2 A and Supplemental Code Section 10), and functions such as *names, print, mean, length, dim, sub, for, c, read.table, read.csv, file.path, file.exists, select, rm, if, setwd* (See Figure 2 B and Supplemental Code Section 11). Most of the functions here that contribute to longer code length are designed for data exploration rather than tidy dataset preparation. Thus, if students include such data exploration code in their final R script to prepare a tidy data set, it is reasonable to cause an increase in code length. The usages of *reshape, reshape2, plyr* libraries do not seem to have great impact on students' code lengths (See Figure 2 A and Supplemental Code Section 10), while the usages of *grep, gsub, aggregate* functions lead to shorter

code length (See Figure 2 A and Supplemental Code Section 11). Since *grep, gsub, aggregate* functions are top used data cleaning functions in general when people clean data, it is reasonable that appropriate application of such functions shortens student's code length.

Discussion

In Methods section 1, we demonstrate that our automatic system is designed to pull repositories created from 2008-01-01 to 2019-09-25. By doing so, we retrieve 20,824 repository names rather than all 31K available repositories for the Getting and Cleaning Data Project, which is a limitation of our program. If we are willing to sacrifice more time in running code to search for repositories created before 2008, we are likely to retrieve more repository names than what we currently have. There are also two limitations in our code to automatically retrieve URL path to student's final R script under their repository name. One limitation is that we have only handled majority of the special characters within the URLs to R scripts but not all of them. It is possible that we save some students' R scripts as an empty string because we are unable to catch the special characters in their URL paths to R scripts. Another limitation is that we only retrieve R scripts from master branch and discard rare cases where some students' R scripts are stored under other branches.

In Methods section 2, we build two linear models to discuss why R packages are relevant in getting and cleaning data. Because of the limitation of our data, we can only explore how the top used libraries and functions relate to students' code lengths. If we not only have students' R scripts but also obtain other information such as students' project scores, gender, and degrees when they take the class etc., we can conduct more interesting analysis to study the relationship between such information and students' code contents.

In Results section, we report the proportion of students who do not complete the course for sure because we are unable to retrieve their final R script on GitHub. However, limitations of our data retrieving system discussed in Methods section 1 influences our calculation accuracy. Also, existence of the final R script does not imply the student has completed the course. Checking whether their final R script runs properly could be our next step of analysis. In addition, we notice that 7,926 out of 10,006 distinct functions are only used once or twice across the whole population, which are very likely to be self-named functions. To conduct further analysis in people using functions to prepare a tidy data set, we can develop a system to automatically detect function names from base R, R packages, or self-named categories.

Reference

[1] Personal communication with Stephen Cristiano on writing code to scrape GitHub
[2] Personal communication with Andrew Leroux on writing code to set up date periods
[3] Leaderboard. RDocumentation.
https://www.rdocumentation.org/trends?page1=5&sort1=total&page2=1&sort2=total&page3=1&page4=1 [accessed October 24, 2017]