Sources of Variability in Solutions to Getting and Cleaning Data Coursera Project

Andrew Leroux[1]

[1] *Johns Hopkins University, Baltimore, MD 21205, USA*

October 9, 2017

Abstract

Abstract Text

Keywords:

1 Introduction

2 Methods

Here we discuss our data collection procedure and the statistical methods used in our data analysis.

2.1 Getting the Data

The data was retrieved from Github. The data was retrived over the 72 hour period period 10/02/2017-10/04/2017. All data retreival and analyses were performed in *R* (add pacakge + R citation). At a high level, the retrieval procedure followed the two steps below

- Use the *gh* package available (add pacakge citation) to find all repos associated with the class which were created on or after 12/01/2007 - Use the *gh* package to search the repos discovered in the previous step for a file called "run_analysis.R". Then scrape the data using the url structure implied by the location of the "run_analysis.R" file using the *readlines* function

More detailed information on each of these steps is provided below.

2.1.1 Finding Repo Names

Searching for repos associated with the getting and cleaning data class results in over 30,000 search results. However, the github API will only report up to 1,000 search results. To get around this, we searched repos by date of creation considering periods of two weeks. The creation dates examined spanned the dates 12/01/2007-10/02/2017. The justification for the start date was based on the start of the creation date of the class as well as some manual exploration which suggested the search results turned up zero entries for reasonable creation dates beyond this period. Note that using a 2 week moving window we never hit the 1,000 search result limit, implying that our window was sufficiently high resolution for this data. In another application (or for the class going forward), this window may need to shrink in order to capture all repos created during the search window.

The exact repo search was performed using the query: "GET /search/repositories?q=getting+and+cleaning+data+**repo_date**&per_page=100" where **repo_date** is of the form "created:2007-11-31..2007-12-15" to get repos created anytime during the two week period 12/01/2007-12/14/2007.

We note that one limitation of our procedure is that any repos created prior to 12/01/2007 would not be included in our analysis. However, since this date is prior

to the creation of the class, in order for someone to be missed using this procedure, they would've had to rename or repurpose an existing repo. We believe this to be unlikely, but note that it is a possiblity. Also, note that our code does not allow for repository names to have hypens in them. These individuals were excluded from our analysis. These individuals number **INCLUDE NUMBER**.

2.1.2 Scraping the Data

To scrape the data, we looped over the repos found in the previous step and searched their (entire) repository for a file called "run_analysis.R". To do so we used the following search query in the **gh()** function: "GET /search/code?q=repo:**repo_name**+extension:r" where **repo_name** is the repository name. This will search the entire repository for any .R files.

We then used regular expressions (*gregexpr*) to find whether any of these .R files matched "run_analysis.r" by using the *to.lower* function in R to allow for various capitalizated letters and still match. Finally, we scraped the "run_analysis.R" file using the *readlines* function on the approraite url based on the file name, repository name and branch name.

We note that our code does impose some limitations. First and foremost, there were some files that we attempted to read, but were found to be non- UTF8 files. We did not try to parse these files at all and they were excluded from the analysis. From manual inspection, these tended to be individuals who copied and pasted R console output into a .R script and used that as their final project. So these are users who, sometimes, were able to complete the task. We estimate the number of these individuals to be **INCLUDE NUMBER**.

2.2 Analyzing the Data

3 Results

Feature	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6
# of lines of code	-0.47	0.15	0.02	-0.18	0.15	-0.07
# of lines of comments	-0.35	0.11	-0.27	-0.44	0.21	0.29
# of blank lines	-0.36	0.15	0.06	-0.35	0.2	-0.51
# of characters of code	-0.34	0.31	0.28	0.51	0.02	-0.04
# of packages	-0.1	-0.03	-0.66	0.28	-0.3	-0.57
# of object assignments	-0.31	-0.54	0.21	0.04	0.07	-0.17
# of unique assignment names	-0.2	-0.68	0.13	0.14	0.08	0.01
# of base subsetting used	-0.19	0.01	0.29	-0.32	-0.87	0.02
# of functions called	-0.39	0.23	0.08	0.44	-0.06	0.24
# of unique functions called	-0.28	-0.19	-0.5	0.04	-0.14	0.49
% Variance explained	38.3	15.2	12.7	10.5	8.3	6.8
Cumulative % variance explained	38.3	53.5	66.2	76.7	85	91.9

Table 1: *First Six Principal Components*

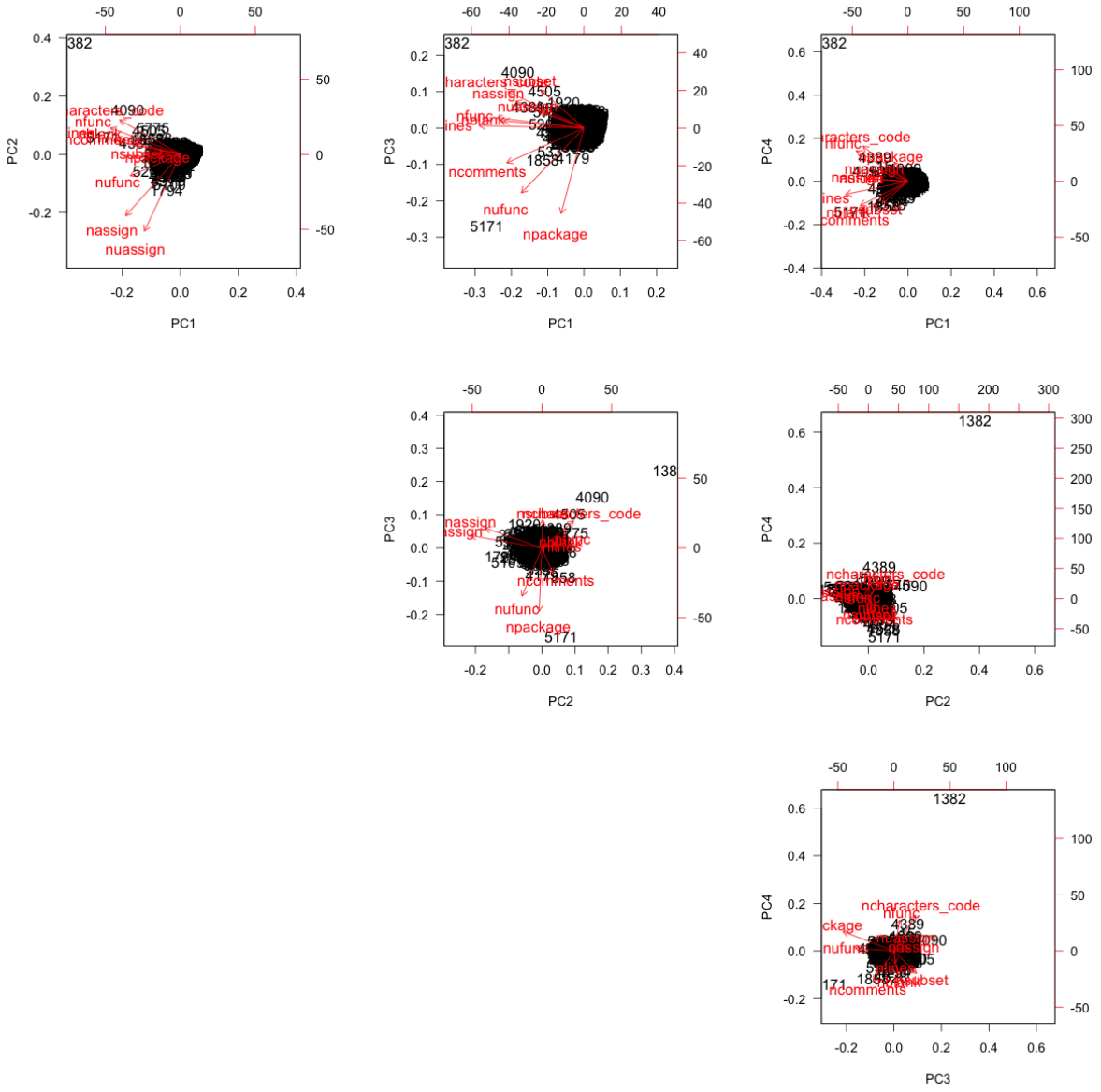

Figure 1: *Biplots for the first 4 principal components*