BibWrangleR

December 23, 2014

This document provides an overview of a set of scripts designed to facilitate bibliometric analyses of a variety of databases that are not readily amenable to analyses due to the shape of the output. The scripts *wrangle* bibliometric data into proper shape using the R programming language – hence, the name, `BibWrangleR` (BWR). Users who are not familiar with the R programming language can use these scripts to process files into a common *.csv format for analysis in common statistical packages, such as Stata, SAS, and SPSS.

Currently, the scripts support the wrangling of three different databases: PubMed, PsychInfo (via EbscoHost), and Citation Reports (via Web of Science). In addition to wrangling the output of search results into a shape for easy analysis, functionality is also provided to link or merge the results across data bases. For example, a highly refined search can first be performed in PubMed or PsychInfo, and those results can then be link with Citation Reports from the Web of Science.

Obtain Data

To functionality of the `BWR` scripts will be demonstrated by examining the growth and impact of qualitative research in social work. For purposes of brevity, this example will focus on growth and impact of a single journal, *Research on Social Work Practice*. It should be noted that the `BWR` functions can process raw data from any number or combination of journals that are indexed by the database from which the data are derived.

PsychInfo (via EbscoHost)

In this example, the advanced search functions in the EbscoHost platform was used to perform a search. This involved setting the `SO` classifier to *"Research on Social Work Practice"* (with quotes), and then using the drop-down to set the methodology classifier to *"Qualitative study"*. Thus, the reader should not make any substantive inferences of the results that are presented. After performing the search, the results from EbscoHost need to be exported. This is done by selecting *Share* and then *Export results*. The is prompted to select one of a number of different formatting options. The required format for the `BWR` fucntions is the *Generic bibliographic management format*. The Appendix shows the format of the raw data in this format.

The search results for this example produced 47 unique article records. It should be noted that no further data processing was involved (e.g., reliability check on the document classification), as the focus of this report is demonstrating functionality of the scripts.

Citation Reports (via Web of Science)

The next part of data collection involves obtaining Citation Reports from the Web of Science. Currently, it is not possible to batch process a set of journal article titles. Therefore, citation reports for every article published in *Research on Social Work Practice* was downloaded. These reports need to be exported and saved as *.xls files. The search that was performed in citation reports for 1,573 articles. However, the web interface limits the user to processing only 500 reports at a time. Thus, four separate reports were exported and saved. The user does not have to perform any merging of files, as this is down automatically by the `BWR` scripts.

Data Wrangling

After the data have been collected and saved in the proper format, the next step is to *wrangle* the data into a shape that can be easily analyzed within the R environment, or saved as a standard *.csv file for analysis using another statistical software package.

Initialize the workspace

The first step of the process involves initializing the R workspace, This involves three basic steps. The following code can be easily adapted by a user who is not familiar with R. This involves changing the respective file paths.

```
# Step 1.  Clear workspace
rm(list=ls())

# Step 2.  Read BWR functions
source("/Users/beperron/Git/BibWrangleR/functions/piWrangleR.R")
source("/Users/beperron/Git/BibWrangleR/functions/packages.R")
source("/Users/beperron/Git/BibWrangleR/functions/wosWrangleR.R")
source("/Users/beperron/Git/BibWrangleR/functions/pi.wos.WrangleR.R")
```

```
# Step 3.  Set the path where original raw data are stored
setwd("/Users/beperron/Git/BibWrangleR/Files2Process")
```

```
# Step 4. Set the working directory to store files created by BWR functions
my.path <- "/Users/beperron/Git/BibWrangleR/Files2Process"
```

Wrangle raw data from PsychInfo

The function `piBWR.f` is used to wrangle raw data from the search results using the PsychoInfo database on the EbscoHost platform. The function has two arguments. The first is `csv`, which gives the user the option of outputting the results as a standard *.csv file. The default of this argument is set to* ***FALSE****, meaning that no* csv file will be outputted. In this example, the argument is set to `TRUE` in order to process the *.csv file. The second argument is the path to the folder that contains the file to be processed. It should be noted that this function expects the raw data to be saved as a* txt file, with that extension. The folder should not contain any other *.txt files other than the* txt files to be processed.

```
piBWR.f(csv=TRUE, path=my.path)
```

```
## Wrangling is complete.  The *.csv file can be found in your working directory.
```

This function produces an R data frame called `pi.df`, which can be accessed in R's global environment. A corresponding file called `pi.csv` is also found in the working directory. Appendix A displays both the raw file and the *csv file processed by the `BWR` scripts.

Wrangle raw data from Web of Science Citation Reports

The process of *wrangling* Citation Reports with the `wosBWR.f` function is exactly the same as the `piBWR.f`function. That is, the user specifies the folder containing the raw data, which is saved in the *.xls format and whether a* .csv file should be outputted. Appendix B displays botht the raw file in the *.xls format and the* .csv file produced by the `wosBWR.f` function.

```
wosBWR.f(csv=TRUE, path=my.path)
```

```
Wrangling is complete.  The *.csv file can be found in your working directory.
```

Merging PsychInfo results with the Citation Reports

So far, two separate data files have been produced, one for PsychInfo and another for Citation Reports. The Citation Reports file contains the entire history of article published in *Research on Social Work Practice*. Thus, a matching function `pi.wos.WrangleR.f` is used to identify only the Citation Reports for the articles contained in the PsychoInfo results. This function uses the processed PsychInfo and Citation Reports data that are saved in R's global environment. Thus, a path argument is not used. The *csv argument is to produced matched data files for both PsychInfo and Citation Reports. The data are not merged with each other because the data are in separate formats (long vs. wide). Thus, shaping and merging of the data ultimately depend on the analytic interests of the user. However, the data are easily linked for any given analysis using the variable `short.title` that is contained in both data files. The output of the function shows the number and proportion of records that were matched and unmatched. In this example, all the article records the PsychInfo search was matched with a Citation Report.

```
pi.wos.WrangleR.f(csv=TRUE)
```

```
Matching is complete and available as pi.match and wos.match in the global environment.
The csv files can be found in the same folder as the original data.

$n.original.records
[1] 47

$n.matched
[1] 47

$prop.match
[1] 1

$n.unmatched
[1] 0

$prop.unmatched
[1] 0
```

Example Analyses With PsychInfo Data

A wide variety of analyses can be performed with these two data files.

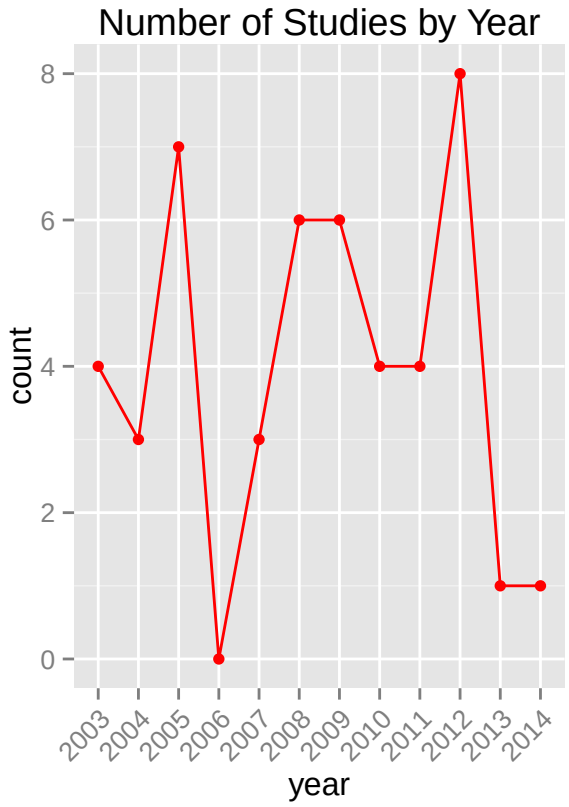
Number of articles over time

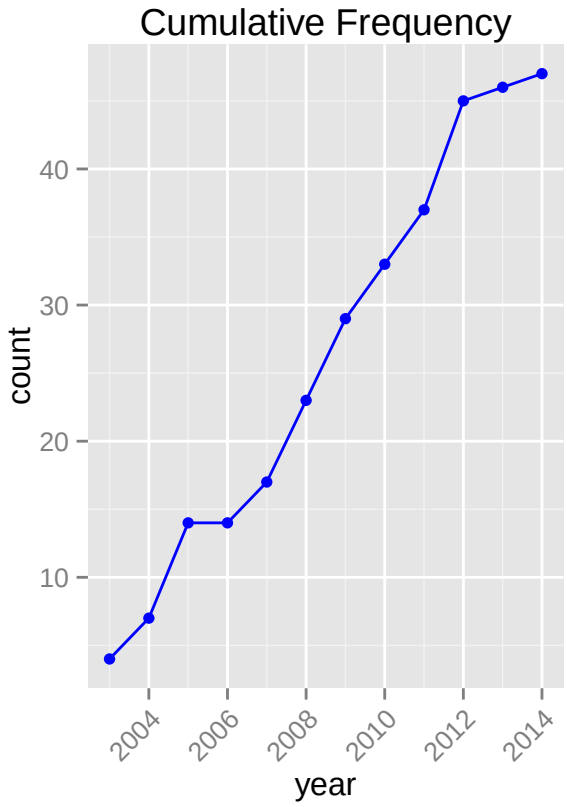
```
colnames(wos.df) <- gsub(" ", "", names(wos.df))
```

```r
library(doBy)
library(ggplot2)
library(gridExtra)
detach(package:plyr)

n.articles.year <- filter(pi.match, attributes == "YR") %>%
    mutate(attributes = as.numeric(attributes))

year.split <- split(n.articles.year, n.articles.year$record)
year.count <- unlist(lapply(year.split, nrow))
year.count[12] <- 0
names(year.count)[12] <- 2006
year.count <- year.count[order(names(year.count))]
years <- (2003:2014)
df <- data.frame(years, year.count)
rownames(df) <- NULL

print.data.frame(df, rownames=FALSE)
```

```
   years year.count
1   2003          4
2   2004          3
3   2005          7
4   2006          0
5   2007          3
6   2008          6
7   2009          6
8   2010          4
9   2011          4
10  2012          8
11  2013          1
12  2014          1
```

```r
plot.article.count <- ggplot(df, aes(as.factor(years), y= year.count, group=1)) +
    geom_line(colour="red") +
    geom_point(colour="red") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    xlab("year") +
    ylab("count") +
    ggtitle("Number of Studies by Year")

plot.article.cumulative <- ggplot(df, aes(x = years, y = cumsum(year.count))) +
    geom_line(colour="blue") +
    geom_point(colour="blue") +
    theme(axis.text.x = element_text(angle=45, hjust=1)) +
    scale_x_continuous(breaks=pretty(df$years)) +
    xlab("year") +
    ylab("count") +
    ggtitle("Cumulative Frequency")

grid.arrange(plot.article.count, plot.article.cumulative, ncol=2)
```





Topic areas (by article keywords)

It is easy to explore some of the different fields within the PsychInfo data frame. For example, each record has one or more subject terms (from the article keywords). The total number, unique number, and most frequently occuring key words can be easily computed. It is, indeed, a bit odd to see *test validity*, *psychometrics*, and *test reliability* as most frequently occuring keywords among articles with "Qualitative Research" as a document classifier. However, be reminded that these data were obtained only for purposes of demonstrating the `BWR` scripts, and no substantive inferences from these data should be made.

```
library(stringr)
library(dplyr)
library(ggplot2)
df.2 <- filter(pi.df, attributes == "SU")
subject.terms <- str_split(df.2$record, pattern = ";")
subject.terms <- unlist(lapply(subject.terms, function(x) gsub(" ", "", x)))
subject.terms.total <- length(unlist(lapply(subject.terms, function(x) gsub(" ", "", x))))
subject.terms.unique <- length(unique(subject.terms))

subject.terms.l <- list(subject.terms.total = subject.terms.total,
                        subject.terms.unique = subject.terms.unique)

most.frequent <- as.data.frame(table(subject.terms))
most.frequent <- arrange(most.frequent, desc(Freq))
most.frequent.t <- head(most.frequent, 10)
```

```
print(subject.terms.l)
```

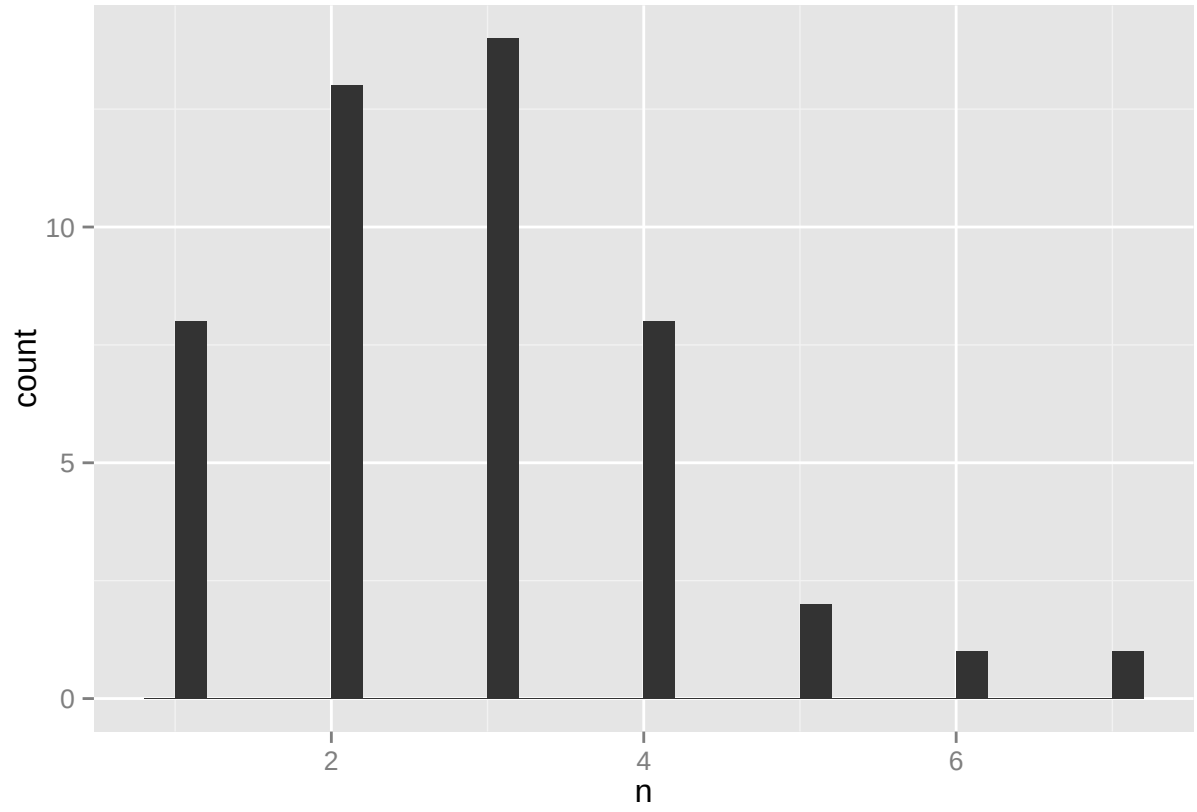
```
$subject.terms.total
[1] 198

$subject.terms.unique
[1] 150
```

```
print(most.frequent.t)
```

```
           subject.terms Freq
1           ChildWelfare    6
2         SocialCasework    6
3  EvidenceBasedPractice    5
4           TestValidity    5
5           Intervention    4
6                 Family    3
7         ParentTraining    3
8          Psychometrics    3
9        TestReliability    3
10       CollegeStudents    2
```

Number of authors

```
n.authors.article <- pi.match %>%
    filter(attributes == "AU") %>%
    select(id = articleID, author= record) %>%
    mutate(id = as.numeric(id))

n_authors <- n.authors.article %>%
        group_by(id) %>%
        summarise(n = n())


ggplot(n_authors, aes(x = n)) + geom_bar()
```




```
summary(n_authors$n)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   2.787   3.500   7.000
```

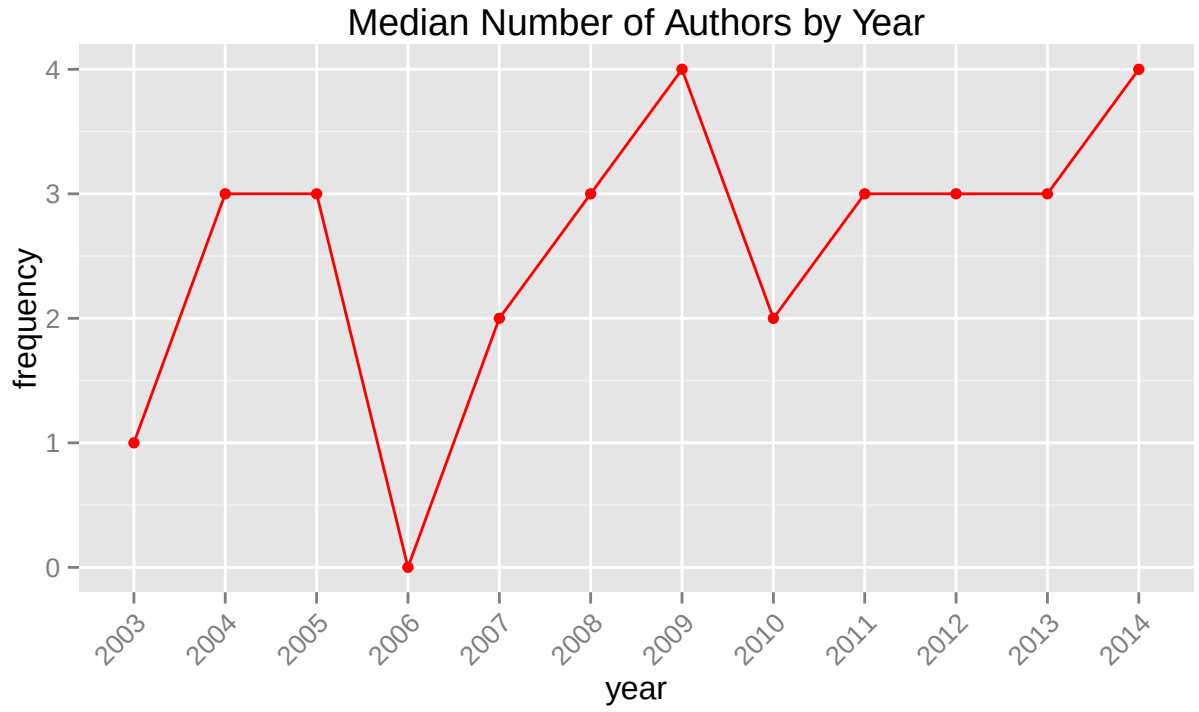
Number of authors over time

```
df.2 <- tbl_df(pi.match)
year <- df.2 %>%
        filter(attributes == "YR") %>%
        select(id = articleID, year = record)

authors <- df.2 %>%
        filter(attributes == "AU") %>%
        select(id = articleID, author = record)

n_authors <- authors %>%
        group_by(id) %>%
        summarise(n=n())

n_authors <- n_authors %>%
        left_join(year) %>%
        group_by(year) %>%
        summarise(median.n = median(n))
```

```
n_authors[12,c(1,2)] <- c(2006, 0)


plot.article.count <- ggplot(n_authors, aes(as.factor(year), y=median.n, group=1)) +
    geom_line(colour="red") +
    geom_point(colour="red") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    xlab("year") +
    ylab("frequency") +
    ggtitle("Median Number of Authors by Year")

print(n_authors)
```

```
Source: local data frame [12 x 2]

   year median.n
1  2003        1
2  2004        3
3  2005        3
4  2007        2
5  2008        3
6  2009        4
7  2010        2
8  2011        3
9  2012        3
10 2013        3
11 2014        4
12 2006        0
```

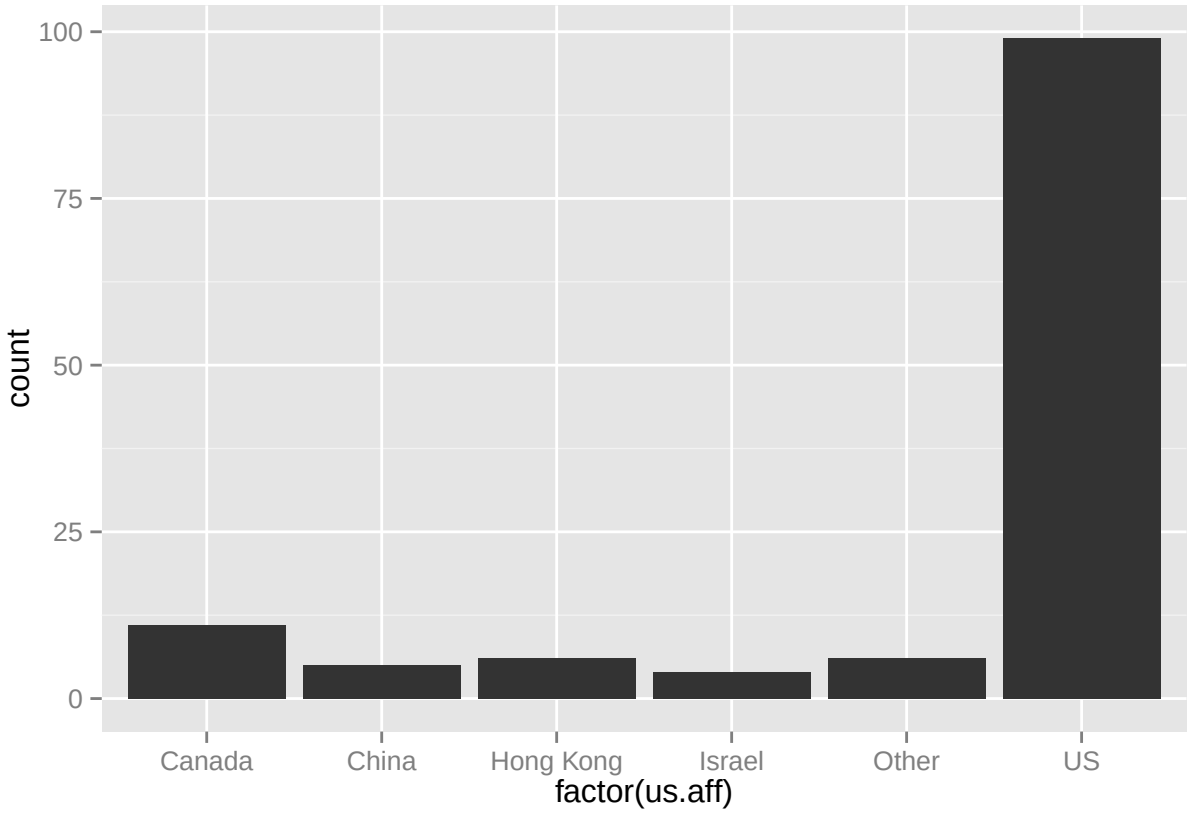
```
plot.article.count
```



How Many International Contributors?

We can look at affiliation and determine how many international contributors. That is, the author affiliations `AF` in the database can be searched using a *regular expression*. The regular expression searches for the characters `US` without any characters immediately preceding or proceding. A quick review of the author affiliations suggests this is a satisfactory search strategy.

```
df.affiliations <- pi.match %>%
                filter(attributes == "AF")

us.aff <- ifelse(grepl("US", df.affiliations$record, perl=TRUE) == TRUE, "US",
          ifelse(grepl("Canada", df.affiliations$record, perl=TRUE ) == TRUE, "Canada",
          ifelse(grepl("Kong", df.affiliations$record, perl=TRUE ) == TRUE, "Hong Kong",
          ifelse(grepl("China", df.affiliations$record, perl=TRUE) == TRUE, "China",
          ifelse(grepl("Israel", df.affiliations$record, perl=TRUE) == TRUE, "Israel", "Other" )))))

affiliations <- data.frame(cbind(df.affiliations,us.aff))

ggplot(data=df.affiliations, aes(x = factor(us.aff))) + geom_bar()
```


100
75
50
25
0
count
Canada
China
Hong Kong
Israel
Other
US
factor(us.aff)