Parse CV

Kevin DelRosso

High level overview

Setting up the workspace

```
rm(list = ls())
setwd("~/Dropbox/GSR/parse_citations/R/")

source("pdf_to_xml.R")
source("parse_citations.R")
source("extract_sections.R")
source("extract_citations.R")
```

Converting pdf to xml

Using the function pdf_to_xml() in pdf_to_xml.R, we can converting all pdf files in the directory to xml.

```
dir = "~/Dropbox/GSR/CV_examples/SampleCVs"
if( FALSE ) pdf_to_xml(dir)
```

Exploring XML

To get a feel for the data, lets look at some of the XML output from the first page of an example CV.

```
# parse xml
filename = "~/Dropbox/GSR/CV_examples/CV_XML/cv_amir.xml"
doc = xmlParse(filename)

# some xml output from page 1
# note, the third entry contains unicode and causes an error with markdown
# which is only reason it is omitted
getNodeSet(doc, "//page[@id=1]//textbox")[c(1:2,4)]
```

```
## [[1]]
## <textbox id="0" bbox="72.000,697.441,368.328,725.972">
##   <textline bbox="72.000,697.441,368.328,725.972" charcterSizes="28.531,28.531,28.531,28.531,,28.531
##     <text font="YGJQUO+CharterBT-Roman" bbox="72.000,697.441,87.205,725.972" size="28.531">Amir Aghal
##   </textline>
## </textbox>
##
## [[2]]
## <textbox id="1" bbox="520.260,636.951,544.867,650.712">
##   <textline bbox="520.260,636.951,544.867,650.712" charcterSizes="13.761,13.761,13.761,13.761,">
##     <text font="YGJQUO+CharterBT-Roman" bbox="520.260,636.951,527.594,650.712" size="13.761">Amir</t
##   </textline>
## </textbox>
##
## [[3]]
## <textbox id="3" bbox="72.000,595.336,282.603,681.326">
```

```
##   <textline bbox="72.000,667.565,168.292,681.326" charcterSizes="13.761,13.761,13.761,13.761,,13.761
##     <text font="YGJQUO+CharterBT-Roman" bbox="72.000,667.565,79.334,681.326" size="13.761">Amir Aghal
##   </textline>
##   <textline bbox="72.000,653.119,282.603,666.880" charcterSizes="13.761,13.761,13.761,13.761,13.761,
##     <text font="YGJQUO+CharterBT-Roman" bbox="72.000,653.119,79.954,666.880" size="13.761">Dept. of
##   </textline>
##   <textline bbox="72.000,638.673,87.735,652.434" charcterSizes="13.761,13.761,13.761,">
##     <text font="YGJQUO+CharterBT-Roman" bbox="72.000,638.673,75.214,652.434" size="13.761">ing</text
##   </textline>
##   <textline bbox="72.000,624.228,220.650,637.989" charcterSizes="13.761,13.761,13.761,13.761,13.761,1
##     <text font="YGJQUO+CharterBT-Roman" bbox="72.000,624.228,79.965,637.989" size="13.761">University
##   </textline>
##   <textline bbox="72.000,609.782,220.202,626.780" charcterSizes="13.761,14.404,13.761,13.761,13.761,1
##     <text font="YGJQUO+CharterBT-Roman" bbox="72.000,609.782,78.611,623.543" size="13.761">E/4130 Eng
##   </textline>
##   <textline bbox="72.000,595.336,157.010,609.097" charcterSizes="13.761,13.761,13.761,13.761,13.761,1
##     <text font="YGJQUO+CharterBT-Roman" bbox="72.000,595.336,75.719,609.097" size="13.761">Irvine, CA
##   </textline>
## </textbox>
```

```
# getting all text locations, font, and size from page 1
xpathSApply(doc, "//page[@id=1]//textbox/textline", xmlGetAttr, "bbox")[1:10]
```

```
##  [1] "72.000,697.441,368.328,725.972"  "520.260,636.951,544.867,650.712"
##  [3] "285.032,666.423,402.327,680.184" "285.032,651.977,407.308,665.738"
##  [5] "285.032,637.531,389.760,651.292" "285.032,623.086,406.460,636.847"
##  [7] "285.032,608.640,477.583,622.401" "72.000,667.565,168.292,681.326"
##  [9] "72.000,653.119,282.603,666.880"  "72.000,638.673,87.735,652.434"
```

```
xpathSApply(doc, "//page[@id=1]//textbox/textline/text", xmlGetAttr, "font")[1:10]
```

```
##  [1] "YGJQUO+CharterBT-Roman" "YGJQUO+CharterBT-Roman"
##  [3] "YGJQUO+CharterBT-Roman" "YGJQUO+CharterBT-Roman"
##  [5] "YGJQUO+CharterBT-Roman" "YGJQUO+CharterBT-Roman"
##  [7] "YGJQUO+CharterBT-Roman" "YGJQUO+CharterBT-Roman"
##  [9] "YGJQUO+CharterBT-Roman" "YGJQUO+CharterBT-Roman"
```

```
xpathSApply(doc, "//page[@id=1]//textbox/textline/text", xmlGetAttr, "size")[1:10]
```

```
##  [1] "28.531" "13.761" "13.761" "13.761" "13.761" "13.761" "13.761"
##  [8] "13.761" "13.761" "13.761"
```

```
# counts for the whole document
table(xpathSApply(doc, "//textbox/textline/text", xmlGetAttr, "font"))
```

```
##
##     HXVHKC+CharterBT-Italic       LBAYIW+CharterBT-Bold
##                          12                          34
##   PWVBBP+BeraSansMono-Roman YEVSYK+CharterBT-BoldItalic
##                           1                          25
##      YGJQUO+CharterBT-Roman
##                         793
```

```
table(xpathSApply(doc, "//textbox/textline/text", xmlGetAttr, "size"))
```

```
##
## 11.773 12.557 13.761 13.853 13.956 16.623 19.816 28.531
##      1      1    810     12     12     13     15      1
```

Extracting sections and citations

Using the parsed XML file created above, we can create a data frame with features from the XML for each line of text. Using different combinations of features such as capitalization, text size, left indentation, font, bold / italic, and line spacing (currently not used as it's not effective) we can group all lines of text based on common features. We can then identify one grouping as containing sections, or re-group using different features if no groups are found. Once groupings are made, the strings are compared with known section names.

```
output = parse_cv( "~/Dropbox/GSR/CV_examples/CV_XML/cv_amir.xml", short_text = TRUE )
df = output[[1]]
df[c(1:2, 4:20), ]
```

```
##       left  bottom   right     top                 text text_size
## 1   72.000 697.441 368.328 725.972 Amir AghaKouchak, Ph    28.531
## 8   72.000 667.565 168.292 681.326     Amir AghaKouchak    13.761
## 9   72.000 653.119 282.603 666.880 Dept. of Civil & Env    13.761
## 4  285.032 651.977 407.308 665.738 Mobile: (949) 231-89    13.761
## 10  72.000 638.673  87.735 652.434                  ing    13.761
## 5  285.032 637.531 389.760 651.292  Fax: (949) 824-8831    13.761
## 2  520.260 636.951 544.867 650.712                 Amir    13.761
## 11  72.000 624.228 220.650 637.989 University of Califo    13.761
## 6  285.032 623.086 406.460 636.847 Email: amir.a@uci.ed    11.773
## 12  72.000 609.782 220.202 626.780 E/4130 Engineering G    13.761
## 7  285.032 608.640 477.583 622.401 Website: http://amir    11.773
## 13  72.000 595.336 157.010 609.097     Irvine, CA 92697    13.761
## 14  72.000 560.916 189.077 580.732     Current Position    19.816
## 15  72.000 534.915 322.531 548.676 Assistant Professor,    13.761
## 16  72.000 493.457 232.195 513.273 Professional Licensu    19.816
## 17  72.000 467.455 514.691 481.216 Professional License    13.761
## 18  72.000 425.998 144.454 445.814            Education    19.816
## 19  89.216 399.996 501.803 413.757 PhD, Civil and Envir    13.761
## 20 106.431 380.078 463.813 394.103 Dissertation: Simula    13.761
##                        font caps caps_tf   above   below page_num n_char
## 1   YGJQUO+CharterBT-Roman            0      NA  16.115        1     25
## 8   YGJQUO+CharterBT-Roman            0  16.115 -12.619        1     16
## 9   YGJQUO+CharterBT-Roman            0  -0.457 -12.619        1     40
## 4   YGJQUO+CharterBT-Roman            0 -12.619  -0.457        1     22
## 10  YGJQUO+CharterBT-Roman            0  -0.457 -12.619        1      3
## 5   YGJQUO+CharterBT-Roman            0 -12.619 -13.181        1     19
## 2   YGJQUO+CharterBT-Roman            0 -13.181  -1.038        1      4
## 11  YGJQUO+CharterBT-Roman            0  -1.038 -12.619        1     31
## 6   YGJQUO+CharterBT-Roman            0 -12.619  -3.694        1     21
## 12  YGJQUO+CharterBT-Roman            0  -3.694 -12.619        1     26
## 7   YGJQUO+CharterBT-Roman            0 -12.619  -0.457        1     33
## 13  YGJQUO+CharterBT-Roman            0  -0.457  14.604        1     16
## 14  YGJQUO+CharterBT-Roman            0  14.604  12.240        1     16
```

```
## 15  YGJQUO+CharterBT-Roman            0  12.240  21.642        1     53
## 16  YGJQUO+CharterBT-Roman            0  21.642  12.241        1     22
## 17  YGJQUO+CharterBT-Roman            0  12.241  21.641        1     88
## 18  YGJQUO+CharterBT-Roman            0  21.641  12.241        1      9
## 19  YGJQUO+CharterBT-Roman            0  12.241   5.893        1     80
## 20 HXVHKC+CharterBT-Italic            0   5.893   9.026        1     72
##    text_size_norm left_norm right_norm font_bold font_italic space_ab
## 1            28.5        72        368         0           0        1
## 8            13.8        72        168         0           0        0
## 9            13.8        72        283         0           0        0
## 4            13.8       285        407         0           0        0
## 10           13.8        72         88         0           0        0
## 5            13.8       285        390         0           0        0
## 2            13.8       520        545         0           0        0
## 11           13.8        72        221         0           0        0
## 6            11.8       285        406         0           0        0
## 12           13.8        72        220         0           0        0
## 7            11.8       285        478         0           0        0
## 13           13.8        72        157         0           0        0
## 14           19.8        72        189         0           0        1
## 15           13.8        72        323         0           0        1
## 16           19.8        72        232         0           0        1
## 17           13.8        72        515         0           0        1
## 18           19.8        72        144         0           0        1
## 19           13.8        89        502         0           0        1
## 20           13.8       106        464         0           1        1
##    section_tf
## 1           0
## 8           0
## 9           0
## 4           0
## 10          0
## 5           0
## 2           0
## 11          0
## 6           0
## 12          0
## 7           0
## 13          0
## 14          1
## 15          0
## 16          1
## 17          0
## 18          1
## 19          0
## 20          0
```

```
new_sections = output[[2]]
new_sections
```

```
##  [1] "current position"     "professional licensu" "education"
##  [4] "academic experience"  "honors  awards"       "grants awards  pro"
##  [7] "editorial"            "patent"               "publications"
## [10] "service committee"    "graduate and postdoc" "teaching experience"
```

```
## [13] "invited talks invit"  "conference presentat" "professional societi"
```

One strategy is to parse every CV in the database, to build a maximum sized list of sections names. We can then manually determine which sections correspond to education, publications, etc.

We can then work on extracting citation information from each CV. We proceed down two paths, either we successfully grouped the CV and found section names or we didn't. For specific details on how the text grouping works, see the top section of extract_sections.R. In the first case, we'll locate the publication section and extract all text until the next section (which we know). In the other case, we'll walk through the CV looking for the publication section, and then extract all text until we find any previously identified section which isn't publication.

Once we have the citation text, we have several possible methods for grouping the citation text together correctly:

- numbered list
- author's name
- year (used as a list)
- textbox (using the grouping returned from pdfminer, see note below)
- left indentation
- most common starting word

Note: pdfminer at times will group text oddly. For example, if text is in a numbered list, sometimes all the text furthest left will be contained in one group, while all the indented text will be in another. It's also possible that just the numbers (1., 2., . . .) will be on their own lines, even when they appear inline in the text.

Example

The following is an example which finds the sections and then extracts and parses each citation within the publication section.

```
cv_name = "aghakouchak"
cv_filename = "~/Dropbox/GSR/CV_examples/CV_XML/cv_amir.xml"
pub_filename = "~/Dropbox/GSR/parse_citations/text_files/publications.txt"
section_filename = "~/Dropbox/GSR/parse_citations/text_files/section_names.txt"

output = parse_cv( cv_filename, short_text = FALSE )
df = output[[1]]
found_sections = output[[2]]

# contains: "start_index", "start_section", "end_index", "end_section"
name_index = get_section_locations( df$text, found_sections, pub_filename, section_filename )

# print found sections using to extract citations
print( t (as.data.frame(name_index) ) )
```

```
##               [,1]
## start_index   "129"
## start_section "publications"
## end_index     "410"
## end_section   "service committee  panel assignments"
```

```
cat( "\n" )
```

```
# citation text
df_text = df[ name_index$start_index : name_index$end_index, c("left_norm", "right_norm", "text")]
df_text$text[1:10]
```

```
##  [1] "Journal Publications (Students and Postdocs Underlined)"
##  [2] "58. AghaKouchak A., Feldman D., Hoerling M., Huxman T., Lund J., 2015, Recognize An-"
##  [3] "thropogenic Drought, Nature, 524 (7566), 409-4011, doi:10.1038/524409a."
##  [4] "57. Mazdiyasni O., AghaKouchak A., 2015, Substantial Increase in Concurrent Droughts"
##  [5] "and Heatwaves in the United States, Proceedings of the National Academy of Sciences,"
##  [6] "doi: 10.1073/pnas.1422945112."
##  [7] "56. Vahedifard F., AghaKouchak A., Robinson J.D., 2015, Drought threatens California's"
##  [8] "levees, Science, 349 (6250), 799, doi: 10.1126/science.349.6250.799-a."
##  [9] "55. AghaKouchak A., Farahmand A., Teixeira J., Wardlow B.D., Melton F.S., Anderson M.C.,"
## [10] "Hain C.R., 2015, Remote Sensing of Drought: Progress, Challenges and Opportunities,"
```

```
# add auther's name and year
df_text$name = get_name_lines( cv_name, df_text$text )
df_text$year = get_year_lines( df_text$text )
```

```
# group text into citations
citations = get_citations( cv_filename, df_text, name_index )
```

```
## [1] "Citation Pass: 1"
```

```
citations = trim( citations )

# extract doi and year (if they appear in citation)
doi = get_doi( citations )
year = get_year( citations )

# make citations begin with a letter (remove numbering)
citations = str_extract( citations, "[[:alpha:]].*" )

citation_id = gsub( ".*/(.*)\\.xml$", "\\1", cv_filename )

n = 10
found_citations = query_crossref( citations[1:n], doi[1:n], year[1:n],
                                  id = citation_id, cv_name = cv_name )

# removing these columns for display only
col_names = c("original_citation", "fullCitation", "authors", "title")
found_citations = found_citations[ found_citations$citation_rank == 1,
                                   !names(found_citations) %in% col_names]
found_citations
```

```
##                                                doi    score year
## 1                http://dx.doi.org/10.1038/524409a 5.734515 2015
## 3         http://dx.doi.org/10.1073/pnas.1422945112 4.139546 2015
## 6  http://dx.doi.org/10.1126/science.349.6250.799-a 7.452650 2015
```

```
## 7            http://dx.doi.org/10.1002/2014rg000456 3.645093 2015
## 11           http://dx.doi.org/10.1002/2015jd023147 3.915291 2015
## 16           http://dx.doi.org/10.1002/2015gl063666 3.599102 2015
## 20        http://dx.doi.org/10.1021/acs.est.5b01635 4.267826 2015
## 23     http://dx.doi.org/10.1016/j.jglr.2014.12.007 4.950608 2015
## 24      http://dx.doi.org/10.1007/s00382-015-2625-y 4.755967 2015
## 28           http://dx.doi.org/10.1002/2014wr016318 4.745144 2015
##    title_score merge_score                                        journal
## 1          0.7    4.014161                                         Nature
## 3          1.0    4.139546 Proceedings of the National Academy of Sciences
## 6          1.0    7.452650                                        Science
## 7          1.0    3.645093                        Reviews of Geophysics
## 11         1.0    3.915291    Journal of Geophysical Research: Atmospheres
## 16         1.0    3.599102                  Geophysical Research Letters
## 20         1.0    4.267826          Environmental Science &amp;amp; Technology
## 23         1.0    4.950608                 Journal of Great Lakes Research
## 24         1.0    4.755967                              Climate Dynamics
## 28         1.0    4.745144                        Water Resources Research
##    cite_count_crossref journal_tf citation_rank      id
## 1                    0          1             1 cv_amir
## 3                    0          1             1 cv_amir
## 6                    0          1             1 cv_amir
## 7                    2          1             1 cv_amir
## 11                   0          0             1 cv_amir
## 16                   5          1             1 cv_amir
## 20                   2          0             1 cv_amir
## 23                   1          1             1 cv_amir
## 24                   0          1             1 cv_amir
## 28                   1          1             1 cv_amir
```

CrossRef returns the title, full citation (in a consistent format), and a score for the match. We then create a fuzzy match between the title CrossRef returns and the original citation. This gives us a pseudo percentage for the title match. We can then multiply the returned score by the title score to give use a better sense of which citation is correct. This isn't need when the citation is obvious, but many times none of the returned results seem likely based solely on the score. If we get a good title match, however, we can be more confident we've found the citation.

Interestingly, the first citation we had found the doi, which gave us a very high score (~18) returned from CrossRef, however the title_score was only 0.7. Further investigation found that the actual title is

"Water and climate: Recognize anthropogenic drought"

which is different from the citation on the CV:

"AghaKouchak A., Feldman D., Hoerling M., Huxman T., Lund J., 2015, Recognize Anthropogenic Drought, Nature, 524 (7566), 409-4011, doi:10.1038/524409a."

Also notice that there is a typo in the page numbers in the citation (4011 should be 411). All this to be aware of what can go wrong when trying to match, even when we have the doi.

Using with UC Recruit

The main files are:

- pdf_to_xml.R
- parse_citations.R
- extract_sections.R
- extract_citations.R
- parse_ucrecruit.R

We also use the files:

- procMiner.R - called via pdf_to_xml.R
- pdf_to_xml_wrapper.R - called via pdf_to_xml.R
- libraries.R - called by all files

see each file for a full description and the important functions. The most important functions overall are:

- pdf_to_xml()
- parse_cv()
- get_section_locations()
- get_citations()
- query_crossref()

Process all PDFs to XML

In order to process all the PDF CVs to XML, use the function call_pdf_to_xml() in parse_ucrecruit.R. This will run though each folder and convert all the PDFs to XML. If the original folder name is ucrecruit_university, the XMLs will be in the folder ucrecruit_university_xml. Inside this new folder will also be a _error_log.txt file which records all the PDFs which had an error during the conversion process and were not converted successfully. It takes approximately 1 hour to process 1,000 PDFs.

Note. We ran into an issue with R running out of memory and crashing while processing the thousands of UC Recruit PDFs. A solution that looks promising uses the file pdf_to_xml_wrapper.R. The idea is the function pdf_to_xml() makes a system call to create a new R session. We can then loop over all the files and periodically launch new R sessions, re-allocating memory once an old session closes and a new one begins. We needed to update one function in Duncan's procMiner.R, and updated version is shown below:

```
convertPDF =
    #
    # If given a pdf, call pdfminer's pdf2txt to create the XML file.
    #
    #XXX Need to locate the pdfminer/tools/pdf2txt.py script
    #
function(filename, pdfminer = getOption("PDF2TXT", "pdf2txt.py"))
{
    # update path to pdf2txt.py if not option given above
    if( pdfminer == "pdf2txt.py" ) {
        pdfminer = "~/anaconda/bin/pdf2txt.py"
    }

    cmd = sprintf("%s -t xml -F 1.0 %s", pdfminer, filename)
    system(cmd, intern = TRUE)
}
```

Typically we'll just use the line

```
options(PDF2TXT = "~/anaconda/bin/pdf2txt.py")
```

to add the full path to the file pdf2txt.py. However, this method caused the shell terminal to complain about not finding that file, so we've hard coded the path "~/anaconda/bin/pdf2txt.py". This may need to be updated if working on a different system which doesn't use anaconda.

Process all sections from XML

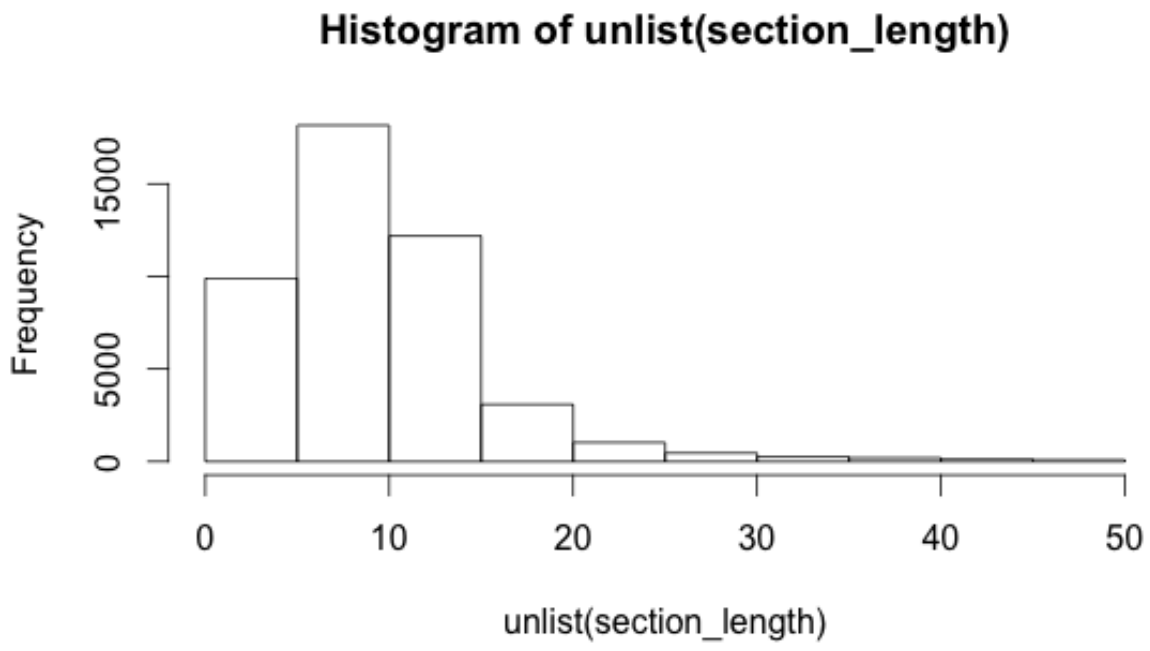
Using the function call_parse_cv() in parse_ucrecruit.R, we can run through all the parsed XML files from above and extract all the sections. Sometimes we'll get errors in the process, notably an error with the XML not being recongized as XML and getting an error with xmlParse(). The files with errors are saved in a log file "~/Dropbox/GSR/parse_citations/text_files/section_errors.txt" and can be further explored to improve the results using explore_section_errors().

As a test, we extracted sections from UCB, 45,606 PDF files from "~/Documents/cv/ucrecruit_ucb_xml/". The results are:

- total files: 45606
- found section count: 38880 (85.3%)
- error count: 1173 (2.6%)
- time: ~2.7 hours

and a rough histogram of the number of sections extracted from each CV.

```
image = "~/Dropbox/GSR/parse_citations/images/number_of_sections.png"
grid.raster( readPNG(image) )
```



Process all citations

The following example is a small version of what the final UC Recruit parsing process will look like. We'll run a loop (mapply) using the XML files and CV author's names. With each pair we'll call main() which runs the whole process from start to finish. If we get an error we'll write that file to an error log, otherwise we'll save the results to a data frame (as a .RData).

```
# load in the XML
xml_files = list.files("~/Dropbox/GSR/CV_examples/CV_XML", full.names = TRUE)
cv_names = tolower( readLines( "~/Dropbox/GSR/parse_citations/text_files/cv_names.txt" ) )

# took ~36 minutes for 65
start = proc.time()

# use sink to capture output rather than printing to the console
sink( file("~/Dropbox/GSR/parse_citations/text_files/citation_output.txt", open = "wt") )

found_citations = mapply( function(file, cv_name) {
    results_df = try( main( file, cv_name, pub_filename, section_filename ), silent = TRUE )

    # if we get an error, print the file name and continue
    if( class(results_df) == "try-error" ) {
        print( paste( "Error with file:", file ) )
        results_df = NULL
    }
    cat( "\n\n" )
    results_df
    }, xml_files, cv_names)

sink()
proc.time() - start

# save the results with today's date, this is the final output
save_filename = sprintf( "~/Dropbox/GSR/parse_citations/saved_results/found_citations_%s.RData",
                         Sys.Date() )
save( found_citations, file = save_filename )
```

This example ran through 65 CVs in ~36 minutes. It returned 1,402 total results and 514 unique citations. The mean and median for title_score and score are shown below. Note, a score above 1 appears to be good (though this value is returned from crossref so it's unknown exactly how it's computed).

```
# loads found_citations object into workspace
load( "~/Dropbox/GSR/parse_citations/saved_results/found_citations_2015-11-28.RData" )

# we don't get back results from every CV
table( sapply( found_citations, class ) )
```

```
##
## data.frame       list       NULL
##         56          5          4
```

```
# combine all the data.frames
mask = which( sapply( found_citations, class ) == "data.frame" )
found_citations = do.call( rbind, found_citations[mask] )
```

```r
# all citations
dim( found_citations )
```

```
## [1] 1402   14
```

```r
# keeping only those with rank = 1
mask = found_citations$citation_rank == 1
found_citations = found_citations[ mask, ]

dim( found_citations )
```

```
## [1] 514  14
```

```r
head( table( found_citations$id ) )
```

```
##
##                          Abrutyn_CV                   Allyson_Stokes_CV
##                                  25                                   3
##                          almquist-cv                   Anderson__KF_-_CV
##                                   8                                   5
##               AnderssonCV-Septb2015 Apesoa-Varano_CV_AUGUST_2015_082815
##                                   9                                  46
```

```r
# some statistics
title_median = median( found_citations$title_score )
title_mean = mean( found_citations$title_score )

score_median = median( found_citations$score )
score_mean = mean( found_citations$score )

data.frame( title_score = c(title_median, title_mean),
            score = c(score_median, score_mean), row.names = c("median", "mean") )
```

```
##        title_score    score
## median   0.9000000 2.838550
## mean     0.7154669 2.916935
```