Retrieving and Processing PubMed Records via easyPubMed (pdf)

Damiano Fantini, Ph.D.

February 25, 2017

PubMed is an online repository of references and abstracts of publications in the fields of medicine and life sciences. PubMed is a free resource that is developed and maintained by the National Center for Biotechnology Information (NCBI), at the U.S. National Library of Medicine (NLM), located at the National Institutes of Health (NIH). PubMed homepage is located at the following URL: https://www.ncbi.nlm.nih.gov/pubmed/. Alternatively, PubMed can be programmatically queried via the NCBI Entrez E-utilities interface.

easyPubMed is an **R interface to the Entrez Programming Utilities** aimed at allowing an easy and smooth programmatic access to PubMed. The package is suitable for batch downloading large volumes of records (via the *batch_pubmed_download()* function) and also comes with a set of functions to perform basic processing of the PubMed query output. *easyPubMed* can request and handle PubMed records in either XML or TXT format. This vignette covers the key functions of the package and provides informative examples.

Retrieving data from PubMed

The first section of this tutorial covers how to use easyPubMed for querying Entrez and how to retrieve or download the desired PubMed records from the Entrez History Server.

A simple PubMed query via easyPubMed

Performing a standard PubMed search via easyPubMed is a two-step process: 1) the *PubMed query* step and 2) the *data retrieval* step. PubMed is queried via the *get_pubmed_ids()* function, that only takes one Query string as argument. The standard PubMed synthax applies, i.e. you can use the same tags-filters as in the This has two results. 1) the query results are posted on the Entrez History Server ready for retrieval and 2) the function returns a list containing all information to access and download these resuts from the server. Data can be retrieved from the History Server via the *fetch_pubmed_data()* function. The records can be requested in either XML or TXT format. Here following you can find a very simple example.

```
my_query <- "Damiano Fantini[AU]"
my_entrez_id <- get_pubmed_ids(my_query)
my_abstracts_txt <- fetch_pubmed_data(my_entrez_id, format = "abstract")
my_abstracts_txt[1:10]
```

```
##  [1] ""
##  [2] "1. Oncotarget. 2017 Dec 16;9(4):4537-4548. doi: 10.18632/oncotarget.23344."
##  [3] "eCollection 2018 Jan 12."
##  [4] ""
##  [5] "APOBEC-mediated mutagenesis in urothelial carcinoma is associated with improved"
##  [6] "survival, mutations in DNA damage response genes, and immune response."
##  [7] ""
##  [8] "Glaser AP(1)(2), Fantini D(1)(2), Wang Y(1)(2), Yu Y(1)(2), Rimar KJ(1)(2),"
##  [9] "Podojil JR(3), Miller SD(3), Meeks JJ(1)(2)."
## [10] ""
```

Here, the PubMed records were retrieved in the *Abstract* format. The formats supported by Entrez and easyPubMed are the following: "asn.1", "xml", "medline", "uilist", "abstract". The following example shows how to retrieve PubMed records in XML format. In this case, the resulting output will be a *XMLInternalDocument* and *XMLAbstractDocument* class object. To access such XML object, we recommend using the functions included in the XML package. For example, it is possible to extract the title of each Article as follows.

```
my_abstracts_xml <- fetch_pubmed_data(my_entrez_id)
class(my_abstracts_xml)
```

```
## [1] "XMLInternalDocument" "XMLAbstractDocument"
```

```
#
# apply "saveXML" to each //ArticleTitle tag via XML::xpathApply()
my_titles <- unlist(xpathApply(my_abstracts_xml, "//ArticleTitle", saveXML))
#
# use gsub to remove the tag, also trim long titles
my_titles <- gsub("(^.{5,10}Title>)|(<\\/.*$)", "", my_titles)
my_titles[nchar(my_titles)>75] <- paste(substr(my_titles[nchar(my_titles)>75], 1, 70),
                                        "...", sep = "")
print(my_titles)
```

```
##  [1] "APOBEC-mediated mutagenesis in urothelial carcinoma is associated with..."
##  [2] "A Carcinogen-induced mouse model recapitulates the molecular alteratio..."
##  [3] "DDB2 Is a Novel Regulator of Wnt Signaling in Colon Cancer."
##  [4] "The evolving genomic landscape of urothelial carcinoma."
##  [5] "Chromatin association of XRCC5/6 in the absence of DNA damage depends ..."
##  [6] "The prion protein is critical for DNA repair and cell survival after g..."
##  [7] "Rapid inactivation and proteasome-mediated degradation of OGG1 contrib..."
##  [8] "Understanding different functions of mammalian AP endonuclease (APE1) ..."
##  [9] "Critical lysine residues within the overlooked N-terminal domain of hu..."
## [10] "APE1/Ref-1 interacts with NPM1 within nucleoli and plays a role in the..."
## [11] "APE1/Ref-1 regulates PTEN expression mediated by Egr-1."
```

Downloading and saving records as XML or TXT files

Instead of retrieving PubMed records as character- or XML-class objects, it is also possible to download all records of a PubMed query and save them as *txt* or *xml* files on the local computer. Downloaded records will be saved locally as one or more files with the same prefix followed by a sequential number and the *txt* or *xml* extension. If a destination folder is not specified, the current directory will be used as target directory for the download. The *batch_pubmed_download()* function is suitable for downloading very large volumes of PubMed records.

```
new_query <- "(APE1[TI] OR OGG1[TI]) AND (2012[PDAT]:2016[PDAT])"
out.A <- batch_pubmed_download(pubmed_query_string = new_query,
                               format = "xml",
                               batch_size = 150,
                               dest_file_prefix = "easyPM_example")
```

```
## [1] "PubMed data batch 1 / 2 downloaded..."
## [1] "PubMed data batch 2 / 2 downloaded..."
```

```
out.A # this variable stores the name of the output files
```

```
## [1] "easyPM_example01.xml" "easyPM_example02.xml"
```

Manipulating PubMed records

The second section of this tutorial covers those *easyPubMed* functionalities aimed at transforming and analyzing PubMed records. While using the functions from the *XML* package is usually the recommended approach to deal with data stored in XML format, there are some exceptions where it may be convenient to coerce these records to Strings. *easyPubMed* comes with a set of dedicated functions that perform this task and manipulate the results. These functions will be covered in this section.

Extracting Affiliation data from a single PubMed record

TO convert XML PubMed records to strings, the *articles_to_list()* function is used. This function converts an an XML object containing PubMed records (identified by the \\PubmedArticle tag) into a list of individual records from or an XML file obtained as shown above. Each record in the list is a string (character-class vector of length 1) that still includes XML tags.

```
my_PM_list <- articles_to_list(my_abstracts_xml)
class(my_PM_list[[4]])
```

```
## [1] "character"
```

```
cat(substr(my_PM_list[[4]], 1, 984))
```

```
## <PubmedArticle>
##   <MedlineCitation Status="Publisher" Owner="NLM">
##     <PMID Version="1">28169993</PMID>
##     <DateRevised>
##       <Year>2017</Year>
##       <Month>06</Month>
##       <Day>03</Day>
##     </DateRevised>
##     <Article PubModel="Print-Electronic">
##       <Journal>
##         <ISSN IssnType="Electronic">1759-4820</ISSN>
##         <JournalIssue CitedMedium="Internet">
##           <Volume>14</Volume>
##           <Issue>4</Issue>
##           <PubDate>
##             <Year>2017</Year>
##             <Month>Feb</Month>
##             <Day>07</Day>
##           </PubDate>
##         </JournalIssue>
##         <Title>Nature reviews. Urology</Title>
##         <ISOAbbreviation>Nat Rev Urol</ISOAbbreviation>
##       </Journal>
##       <ArticleTitle>The evolving genomic landscape of urothelial carcinoma.</ArticleTitle>
##       <Pagination>
##         <MedlinePgn>215-229</MedlinePgn>
##       </Pagination>
##       <ELocationID EIdType="doi" ValidYN="Y">10.1038/nrurol.2017.11</ELocationID>
##       <Abstract>
##         <AbstractT
```

Affiliations or other fields of interest can be extracted from a specific record using the *custom_grep()* function, that combines regular expressions (*regexpr*, *gsub*) and substring extraction (*substr*). The fields extracted from

the record will be returned as elements of a list or a character vector.

```
curr_PM_record <- my_PM_list[[4]]
custom_grep(curr_PM_record, tag = "DateCompleted")
```

```
## list()
```

```
custom_grep(curr_PM_record, tag = "LastName", format = "char")
```

```
## [1] "Glaser"      "Fantini"     "Shilatifard" "Schaeffer"   "Meeks"
```

easyPubMed implements out of the box a tool for extracting data from a PubMed record: the *article_to_df()* function. This function accepts a string as input (typically, an element of the list outputted by an *articles_to_list()* call) and returns a data.frame. Each row corresponds to a different author; columns include values extracted from the following fields: c("pmid", "doi", "title", "abstract", "year", "month", "day", "jabbrv", "journal", "lastname", "firstname", "address", "email"). One of these fields corresponds to the Article Abstract text (column n. 2). If the full text Abstract is not required, it is possible to limit the number of chars retrieved from this field by setting the max_chars argument to a small integer (>= 1).

```
my.df <- article_to_df(curr_PM_record, max_chars = 18)
#
# Fields extracted from the PubMed record
colnames(my.df)
```

```
##  [1] "pmid"      "doi"       "title"     "abstract"  "year"
##  [6] "month"     "day"       "jabbrv"    "journal"   "lastname"
## [11] "firstname" "address"   "email"
```

```
#
# Trim long strings and then Display some content: each row corresponds to one author
my.df$title <- substr(my.df$title, 1, 15)
my.df$address <- substr(my.df$address, 1, 19)
my.df$jabbrv <- substr(my.df$jabbrv, 1, 10)
my.df[,c("pmid", "title", "jabbrv", "firstname", "address")]
```

```
##       pmid           title     jabbrv   firstname             address
## 1 28169993 The evolving ge Nat Rev Ur Alexander P Northwestern Univer
## 2 28169993 The evolving ge Nat Rev Ur     Damiano Northwestern Univer
## 3 28169993 The evolving ge Nat Rev Ur         Ali Northwestern Univer
## 4 28169993 The evolving ge Nat Rev Ur    Edward M Northwestern Univer
## 5 28169993 The evolving ge Nat Rev Ur    Joshua J Northwestern Univer
```

When affiliation info are identical for multiple authors, they are usually omitted as in the example above. Addresses may be imputed for all authors in the dataframe by setting the "autofill" argument to TRUE.

```
my.df2 <- article_to_df(curr_PM_record, autofill = TRUE)
my.df2$title <- substr(my.df2$title, 1, 15)
my.df2$jabbrv <- substr(my.df2$jabbrv, 1, 10)
my.df2$address <- substr(my.df2$address, 1, 19)
my.df2[,c("pmid", "title", "jabbrv", "firstname", "address")]
```

```
##       pmid           title     jabbrv   firstname             address
## 1 28169993 The evolving ge Nat Rev Ur Alexander P Northwestern Univer
## 2 28169993 The evolving ge Nat Rev Ur     Damiano Northwestern Univer
## 3 28169993 The evolving ge Nat Rev Ur         Ali Northwestern Univer
## 4 28169993 The evolving ge Nat Rev Ur    Edward M Northwestern Univer
## 5 28169993 The evolving ge Nat Rev Ur    Joshua J Northwestern Univer
```

Automatic Data Extraction from XML PubMed Records

To retrieve author information and publication data from multiple XML records at once, it is possible to use the *table_articles_byAuth()* function. This function relies on the funcions discussed above and returns a dataframe including all the fields extracted in the previous example. The function accepts five arguments. * **pubmed_data**: an XML file or an XML object with PubMed records
* **max_chars** and **autofill**: same as discussed in the previous example * **included_authors**: one of the following options c("first", "last", "all"). The function can return data corresponding to the first, the last or all the authors for each PubMed record. * **dest_file**: if not NULL, the function attempts writing its output to the selected file. Existing files will be overwritten.

```
new_PM_query <- "(APEX1[TI] OR OGG1[TI]) AND (2010[PDAT]:2013[PDAT])"
out.B <- batch_pubmed_download(pubmed_query_string = new_PM_query, dest_file_prefix = "apex1_sample")
```

```
## [1] "PubMed data batch 1 / 1 downloaded..."
```

```
# Retrieve the full name of the XML file downloaded in the previous step
new_PM_file <- out.B[1]
new_PM_df <- table_articles_byAuth(pubmed_data = new_PM_file, included_authors = "first", max_chars = 0
# Alternatively, the output of a fetch_pubmed_data() could have been used
#
# Printing a sample of the resulting data frame
new_PM_df$address <- substr(new_PM_df$address, 1, 28)
new_PM_df$jabbrv <- substr(new_PM_df$jabbrv, 1, 9)
print(new_PM_df[1:10, c("pmid", "year", "jabbrv", "lastname", "address")])
```

```
##        pmid year    jabbrv    lastname                      address
## 1  24190502 2015 Arch. Tox        Bach Grup de Mutagènesi, Departam
## 2  24186001 2014 Tumour Bi         Yan Department of Clinical Labor
## 3  24175791 2014 Asian Pac          Li Cancer Center, Daping Hospit
## 4  24121118 2014 Mech. Age    Lillenes Centre for Molecular Biology
## 5  24101388 2014 J. Physio Antushevich The Kielanowski Institute of
## 6  24075420 2014 DNA Repai          Gu State Key Laboratory of Repr
## 7  23999824 2014 Tumour Bi        Chen Department of Hepatobiliary
## 8  23959014 2014 Biol. Res     Alanazi Genome Research Chair, Depar
## 9  23909557 2014 Genet Tes        Wang Department of Oncology, Shan
## 10 23892003 2014 Exp. Cell         Yan State Key Laboratory of Repr
```

References

- **Sayers, E. A General Introduction to the E-utilities (NCBI)** https://www.ncbi.nlm.nih.gov/books/NBK25497/
- **PubMed Help (NCBI)** https://www.ncbi.nlm.nih.gov/books/NBK3827/
- **Howto: basic usage of easyPubMed - an example** Tutorial/Blog Post
- **Howto: using easyPubMed for a targeting campaign** Tutorial/Blog Post

Feedback and Citation

Thank you very much for using easyPubMed and/or reading this vignette. Please, feel free to contact me (author/maintainer) for feedback, questions and suggestions: my email is <damiano.fantini(at)gmail(dot)com>.

either version 2 of the License, or any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

If you are using *easyPubMed* for a scientific publication, please name the package in the Materials and Methods section of the paper. Thanks! Also, I am always open to collaborations. If you have an idea you would like to discuss or develop based on what you read in this Vignette, feel free to contact me via email. Thank you.

SessionInfo

```
sessionInfo()
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.3 LTS
##
## Matrix products: default
## BLAS: /usr/lib/libblas/libblas.so.3.6.0
## LAPACK: /usr/lib/lapack/liblapack.so.3.6.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] easyPubMed_2.5 XML_3.98-1.9
##
## loaded via a namespace (and not attached):
##  [1] compiler_3.4.3  backports_1.1.2 magrittr_1.5    rprojroot_1.3-2
##  [5] htmltools_0.3.6 tools_3.4.3     yaml_2.1.16     Rcpp_0.12.15
##  [9] stringi_1.1.7   rmarkdown_1.8   knitr_1.19      stringr_1.3.0
## [13] digest_0.6.15   evaluate_0.10.1
```