The Open Archives Initiative (http://www.openarchives.org/) develops and promotes interoperability standards that aim to facilitate the efficient dissemination of content. One key project is the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH, http://www.openarchives.org/pmh/) which provides “a low-barrier mechanism for repository interoperability” for archives (institutional repositories) containing digital content (digital libraries). OAI-PMH allows people (service providers, such as the ones registered with the OAI listed on http://www.openarchives.org/service/listproviders.html) to harvest metadata (from data providers, such as the ones registered with and validated by the OAI listed on http://www.openarchives.org/Register/BrowseSites/). Data Providers administer systems that support the OAI-PMH as a means of exposing metadata. Service Providers use metadata harvested via the OAI-PMH as a basis for building value-added services.

OAI-PMH, currently in version 2.0, defines a mechanism for data providers to expose their metadata. The protocol mandates that individual archives map their metadata to the Dublin Core (DC, http://dublincore.org/), a simple and common metadata set for cross-domain information resource description. OAI-PMH is a set of six verbs or services that are invoked within HTTP, returning the request results in XML format. The OAI-PMH specification can be found at http://www.openarchives.org/OAI/openarchivesprotocol.html. Here, we summarize the basic facts and terminology.

A harvester is a client application that issues OAI-PMH requests. A harvester is operated by a service provider as a means of collecting metadata from repositories. Repositories are network accessible servers that can process the six OAI-PMH requests, and are managed by a data provider to expose metadata to harvesters. OAI-PMH distinguishes between three distinct entities related to the metadata made accessible by the OAI-PMH:

- **resource**: the object or “stuff” that metadata is “about”. Its nature is outside the scope of the OAI-PMH.
- **item**: a constituent of a repository from which metadata about a resource can be disseminated.
- **record**: metadata in a specific metadata format. A record is returned as an XML-encoded byte stream in response to a protocol request to disseminate a specific metadata format from a constituent item.

For each item there is an unambiguous unique identifier which is used in OAI-PMH requests for extracting metadata from the item. Items may contain metadata in multiple formats; Dublin Core is mandatory.

Selective harvesting allows harvesters to limit harvest requests to portions of the metadata available from a repository. The OAI-PMH supports selective harvesting with two types of harvesting criteria that may be combined in an OAI-PMH request: datestamps and membership in sets, an optional construct for grouping items.

The XML encoding of records is organized into the following parts:

- **header**: contains the unique identifier, a datestamp (the date of creation, modification or deletion of the record), zero or more setSpec elements indicating the set membership of the item, and an optional status attribute for indicating the withdrawal of availability of the specified metadata format for the item, dependent on the repository support for deletions.
metadata a single manifestation (format) of the metadata from an item.

about an optional and repeatable container to hold data about the metadata part of the record.

Contents of the containers must conform to an XML Schema. Common uses of these containers include rights statements and provenance statements.

The OAI-PMH verbs (requests) are as follows.

GetRecord retrieve an individual metadata record from a repository.

Identify retrieve information about a repository.

ListIdentifiers an abbreviated form of ListRecords which retrieves only headers rather than records.

ListMetadataFormats retrieve the metadata formats available from a repository, or optionally the formats available for a specific item.

ListRecords harvest records from a repository.

ListSets retrieve the set structure of a repository.

Optional arguments to ListRecords and ListIdentifiers permit selective harvesting of headers based on set membership and/or datestamp.

Some of these requests return a list of discrete entities: ListRecords returns a list of records, ListIdentifiers returns a list of headers, and ListSets returns a list of sets. These lists may be large, and it may be practical to partition them among a series of requests and responses. Repositories may reply with incomplete results and a resumption token, which the harvester can use to issue an additional request (and repeat until completion).

The R package OAIHarvester provides functions for performing each of the six OAI-PMH requests, using, respectively, packages curl (Ooms (2017)) and xml2 (Wickham, Hester, and Ooms (2017)) for HTTP and XML processing. List requests will automatically be re-issued until complete results are obtained. The names of these verb functions start with ‘oaih’ and follow a “combine words with underscores” scheme (e.g., oaih_list_records, corresponding to the OAI-PMH ListRecords verb, for harvesting records). The functions return the actual (aggregated) result of the repository’s response to the harvester’s request.

In addition to these functions for performing OAI-PMH requests, function oaih_harvest is a high-level harvester which allows specifying several metadata formats or sets, and giving datestamps as Date or POSIXt date/time objects. Finally, function oaih_transform provides functionality for transforming the XML results to “useful” R data structures for further processing or analysis. The results of the verb requests are transformed by default.

The ideas underlying these transformations are best illustrated for harvesting records. In a list context, the result is a list of records, each containing the header (with identifier and datestamp and arbitrarily many set specs), metadata in a certain format, and arbitrarily many about entries. Conceptually, we can think of identifier, datestamp, setSpec, metadata and about as variables “observed” for the items in the repository as cases, suggesting the usual rectangular case by variables data organization. When obtaining a single record, it seems natural to transform to a list with these variables. If the rectangular data structure were a data frame, selecting one row (corresponding to a single record) would not straightforwardly yield the single record list transformation (because subscribing list variables in the data frame would give length one sublists rather than the elements). Thus, in the rectangular cases we instead treat rows and columns symmetrically by arranging data in a “list matrix” (a list with a dim attribute, or equivalently, a matrix of list elements). As matrix subscipting drops dimensions when a single row or column is selected, one gets the expected simple list (without a dim attribute) in these cases. (Equivalently, the transformed oaih_list_records result is the same as combining the transformed oaih_get_record results by rows rbind.)
When harvesting records, identifiers and datestamps naturally transform to character strings, and set specs (a header may contain arbitrarily many of these) to character vectors. On the other hand, metadata can be made available in different formats, with different “variables”. We find it more convenient to use a constant set of variables for a single transformation of a certain “kind” of OAI-PMH XML results. Thus, we do not immediately transform the metadata, but instead leave them as lists of XML nodes to be transformed in a second stage (with variables differing according to the metadata format; currently, metadata in the Dublin Core and RFC 1807 (http://www.rfc-editor.org/rfc/rfc1807.txt) formats can be transformed).

These principles (using lists of single observations on variables and possibly arranging them in a rectangular way, and transforming to constant sets of variables) applies for all transformations of OAI-PMH XML results. Transformations can be added by assigning functions in the (currently internal) environment oaih_transform_methods_db.

As an example consider ePubwu, an electronic publication platform for research output provided by WU (Wirtschaftsuniversität Wien), which provides an OAI repository at http://epub.wu.ac.at/cgi/oai2.

```r
> library("OAIHarvester")
> baseurl <- "http://epub.wu.ac.at/cgi/oai2"
```

We can use oaih_identify to retrieve information about the repository.

```r
> x <- oaih_identify(baseurl)
> rbind(x, deparse.level = 0)
```

<table>
<thead>
<tr>
<th>repositoryName</th>
<th>baseURL</th>
<th>protocolVersion</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ePubWU&quot;</td>
<td>&quot;<a href="http://epub.wu.ac.at/cgi/oai2">http://epub.wu.ac.at/cgi/oai2</a>&quot;</td>
<td>&quot;2.0&quot;</td>
</tr>
<tr>
<td>earliestDatestamp</td>
<td>&quot;persistent&quot;</td>
<td>&quot;YYYY-MM-DDThh:mm:ssZ&quot;</td>
</tr>
<tr>
<td>adminEmail</td>
<td>&quot;<a href="mailto:epub@wu.ac.at">epub@wu.ac.at</a>&quot;</td>
<td>Character,0 List,2</td>
</tr>
<tr>
<td>compression</td>
<td>&quot;0&quot;</td>
<td>&quot;description&quot;</td>
</tr>
<tr>
<td>description</td>
<td>&quot;2&quot;</td>
<td>&quot;List&quot;</td>
</tr>
</tbody>
</table>

Here, `rbind` achieves “pretty-printing”: we can see that the repository provides no compression support, and 2 further description entries of kind

```r
> sapply(x$description, xml_name)
```

[1] "oai-identifier" "eprints"

where entry 1 indicates that the repository complies with the OAI format for unique record identifiers:

```r
> oaih_transform(x$description[[1L]])
```

```
$scheme
[1] "oai"

$repositoryIdentifier
[1] "epub.wu-wien.ac.at"

$delimiter
[1] ":" 

$sampleIdentifier
[1] "oai:epub.wu-wien.ac.at:1604"
```

We can use oaih_list_metadata_formats and oaih_list_sets to find out about available metadata formats and sets:
> oaih_list_metadata_formats(baseurl)

    metadataPrefix schema
[1,] "oai_dc"  "http://www.openarchives.org/OAI/2.0/oai_dc.xsd"
    metadataNamespace
[1,] "http://www.openarchives.org/OAI/2.0/oai_dc/"

> sets <- oaih_list_sets(baseurl)
> rbind(head(sets, 3L), tail(sets, 3L))

```
    setSpec
    "796561723D32303136"
    "796561723D32303031"
    "796561723D32303135"
[42,] "74797065733D7465616368696E675F7265736F75726365"
[43,] "74797065733D736F667477617265"
[44,] "openaire"
```

setName setDescription
"Year = 2016" List,0
"Year = 2001" List,0
"Year = 2015" List,0
[42,] "Type = Teaching Resource" List,0
[43,] "Type = Software" List,0
[44,] "OpenAIRE" List,0

Only the mandatory Dublin Core format is offered, and there is a fairly refined set hierarchy for selective harvesting. If we only want the doctoral theses, we need the spec of the set corresponding to type theses:

> spec <- unlist(sets[sets[, "setName"] == "Type = Thesis", "setSpec"])

To obtain all records for the theses:

> x <- oaih_list_records(baseurl, set = spec)

This gives a “list matrix” with observations of 5 variables on 185 items:

> dim(x)

```
[1] 185  5
```

> colnames(x)

```
[1] "identifier" "datestamp" "setSpec" "metadata" "about"
```

Transforming the Dublin Core metadata is achieved by calling `oaih_transform` on the metadata column, after first removing empty metadata (from deleted records):

> m <- x[, "metadata"]
> m <- oaih_transform(m[lengths(m) > 0L])
> dim(m)

```
[1] 184 15
```

giving observations on the 15 (simple) Dublin Core elements:

> colnames(m)
Finally, the topics of the theses are available in the ‘subject’ DC variable, with comment
“Typically, the subject will be represented using keywords, key phrases, or classification codes.
Recommended best practice is to use a controlled vocabulary.” (see [http://dublincore.org/
documents/dcmi-terms/#terms-subject](http://dublincore.org/documents/dcmi-terms/#terms-subject)), but without a more detailed syntactic or semantic specification.

Inspecting the output of \texttt{m[, "subject"], e.g.,}

\begin{verbatim}
> m[c(1L, 6L, 7L), "subject"]
\end{verbatim}

\begin{verbatim}
[[1]]
[1] "RVK QP 530"
[[2]]
[1] "Stock management / marketing / risk behavior / stochastic model "
[[3]]
[1] "TETRA <telecommunication> / acceptance"
\end{verbatim}

\begin{verbatim}
[[4]]
[1] "LINDA <Programmiersprache> / Rechnernetz /"
\end{verbatim}
suggests that “keywords” are separated by slashes, so we can obtain all keywords via

\begin{verbatim}
> sep <- "[[:space:]]*/[[:space:]]*"
> keywords_by_thesis <-
+ strsplit(unlist(lapply(m[, "subject"], paste, collapse = " / ")),
+ sep)
> keywords <- unlist(keywords_by_thesis)
\end{verbatim}
giving a total of 1209 keywords. Most of these only occur once:

\begin{verbatim}
> counts <- table(keywords)
> table(counts)
\end{verbatim}

\begin{verbatim}
counts
1 2 3 4 5 26
985 68 13 2 3 1
\end{verbatim}

(re-iterating the above comment on using controlled vocabularies). The most frequently used keywords are

\begin{verbatim}
> sort(counts[counts >= 3L], decreasing = TRUE)
\end{verbatim}

\textbf{keywords}

\begin{verbatim}
Österreich Umfrage Unternehmen
26 5 5
economic development Austria Internet
5 4 4
Arbeitsmarkt Armut China
3 3 3
Entrepreneurship Hochschule RVK QP 530
3 3 3
RVK QV 578 Wettbewerb Wirtschaftsentwicklung
3 3 3
XML organizational culture supply chain management
3 3 3
Ökonometrie
3
\end{verbatim}
which unfortunately does not include ‘R’: but there is at least one such entry

> counts["R"]

R 2

from David Meyer’s 2003 thesis

> lapply(m[sapply(keywords_by_thesis, function(kw) any(kw == "R")),
+   c("title", "creator")],
+   strwrap)

[[1]]
[1] "A generic simulation environment for heterogeneous agents. With"
[2] "applications in marketing and technological choice."

[[2]]
[1] "Recursive Partitioning of Models of a Generalized Linear Model Type"

[[3]]
[1] "Meyer, David"

[[4]]
[1] "Rusch, Thomas"

whereas Ingo Feinerer’s thesis on text mining (describing in particular the R text mining infrastructure provided by package \texttt{tm} \cite{feinerer08b} has

> m[grep("Feinerer", unlist(m[, "creator"])),
+   c("title", "creator", "subject")]

$title
[1] "A text mining framework in R and its applications"

$creator
[1] "Feinerer, Ingo"

$subject
[1] "RVK ST 304, ST 600"

(again re-iterating the above comment about controlled vocabularies).

Note that OAI-PMH objects obtained by OAI-PMH requests and subsequent transformations are made up of both character vectors and XML nodes from package \texttt{xml2}, with the latter lists of external pointers. Thus, some extra effort is necessary to save OAI-PMH objects to a file or to restore these from a file: see \texttt{?oaih\_save\_RDS} for more information.

\textbf{References}

I. Feinerer. \textit{tm}: Text Mining Package, 2008. \url{http://CRAN.R-project.org/package=tm} R package version 0.3-3.


6