Working with the manifestoR package

Jirka Lewandowski jirka.lewandowski@wzb.eu

03/03/2015

Contents

1 Downloading documents from the Manifesto Corpus

1.1 Loading the package

First of all, load the `manifestoR` package with the usual R syntax:

```
library(manifestoR)
```

```
## Loading required package: NLP
## Loading required package: tm
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
##
## The following object is masked from 'package:stats':
##
##     filter
```

```
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

1.2 Connecting to the Manifesto Project Database API

To access the data in the Manifesto Corpus, an account for the Manifesto Project webpage with an API key is required. If you do not yet have an account, you can create one at https://manifesto-project.wzb.eu/signup. If you have an account, you can create and download the API key on your profile page.

For every R session using manifestoR and connecting to the Manifesto Corpus database, you need to set the API key in your work environment. This can be done by passing either a key or the name of a file containing the key to manifestoR's `mp_setapikey()` function (see documentation `?mp_setapikey` for details). Thus, your R script using manifestoR usually will start like this:

```
library(manifestoR)
mp_setapikey("manifesto_apikey.txt")
```

This code presumes that you have stored and downloaded the API key in a file name `manifesto_apikey.txt` in your current R working directory.

Note that it is a security risk to store the API key file or a script containing the key in public repositories.

1.3 Downloading documents

(Bulk-)Downloading documents works via the function `mp_corpus(...)`. It can be called with a logical expression specifying the subset of the Manifesto Corpus that you want to download:

```
my_corpus <- mp_corpus(countryname == "Austria" & edate > as.Date("2000-01-01"))
```

```
## Connecting to Manifesto Project DB API...
## Connecting to Manifesto Project DB API...
## Connecting to Manifesto Project DB API...
## Connecting to Manifesto Project DB API...
```

```
my_corpus
```

```
## <<ManifestoCorpus (documents: 15, metadata (corpus/indexed): 0/0)>>
```

`mp_corpus` returns a `ManifestoCorpus` object, a subclass of `Corpus` as defined in the natural language processing package `tm`. Following `tm`s logic, a `ManifestoCorpus` consists of `ManifestoDocuments`. For both, corpus and documents, `tm` provides accessor functions to the corpus and documents content and metadata:

```
head(content(my_corpus[[1]]))
```

```
## [1] ""Wir können heute die Existenzgrundlagen"
## [2] "künftiger Generationen zerstören."
## [3] "Oder sie sichern.""
## [4] "Dr. Eva Glawischnig"
## [5] "Österreich braucht jetzt Weitblick."
## [6] "Nachhaltigkeit für zukünftige Generationen"
```

```
meta(my_corpus[[1]])
```

```
## Metadata:
##   manifesto_id               : 1579
##   party                      : 42110
##   date                       : 200211
##   language                   : german
##   is_primary_doc             : TRUE
##   may_contradict_core_dataset: TRUE
##   md5sum_text                : cf902ff2e4e4f9517b6bfc56d3362904
##   url_original               : NA
##   md5sum_original            : NA
##   annotations                : TRUE
##   id                         : 1
```

For more information on the available metadata per document, refer to the section Using the document metadata below. For more information on how to use the text mining functions provided by `tm` for the data from the Manifesto Corpus, refer to the section Processing and analysing the corpus documents below.

The variable names in the logical expression used for querying the corpus database (`countryname` and `edate` in the example above) can be any column names from the Manifesto Project's Main Dataset or your current R environment. The Main Dataset itself is available in manifestoR via the funcion `mp_maindataset()`:

```
mpds <- mp_maindataset()
print(head(names(mpds)))
```

```
## [1] "country"     "countryname" "oecdmember"  "eumember"    "edate"
## [6] "date"
```

```
mp_corpus(rile > 60) ## another example of data set based corpus query
```

```
## Connecting to Manifesto Project DB API...
## Connecting to Manifesto Project DB API...
```

```
## <<ManifestoCorpus (documents: 1, metadata (corpus/indexed): 0/0)>>
```

Alternatively, you can download election programmes on an individual basis by listing combinations of party ids and election dates in a `data.frame` and passing it to `mp_corpus(...)`:

```
wanted <- data.frame(party=c(41220, 41320),
                     date=c(200909, 200909))
mp_corpus(wanted)
```

```
## Connecting to Manifesto Project DB API...
## Connecting to Manifesto Project DB API...
```

```
## <<ManifestoCorpus (documents: 1, metadata (corpus/indexed): 0/0)>>
```

The party ids (41220 and 41320 in the example) are the ids as in the Manifesto Project's main dataset. They can be found in the current dataset documentation at https://manifesto-project.wzb.eu/datasets or in the main dataset.

Note that we received only 1 document, while querying for two. This is because the party with the id 41220 (KPD) did not run for elections in September 2009. Also, not for every party and election data manifesto documents are availabe in the Manifesto Project Corpus. You can check availability of your query beforehand with the function `mp_availability(...)`:

```
mp_availability(party == 41113)
```

```
## Connecting to Manifesto Project DB API...

##           Queried for       Raw Texts found Coded Documents found
##                     6           5 (83.333%)           5 (83.333%)
##       Originals found             Languages
##               3 (50%)            1 (german)
```

Downloaded documents are automatically cached locally. To learn about the caching mechanism read the section Efficiency and reproducibility: caching and versioning below.

1.4 Viewing original documents

Apart from the machine-readable, annotated documents, the Manifesto Corpus also contains original layouted election programmes in PDF format. If available, they can be viewed via the function `mp_view_originals(...)`, which takes exactly the format of arguments as `mp_corpus(...)` (see above), e.g.:

```
mp_view_originals(party == 41320 & date == 200909)
```

The original documents are shown in you system's web browser. All URLs opened by this function refer only to the Manifesto Project's Website. If you want to open more than 5 PDF documents at once, you have to specify the maximum number of URLs allows to be opened manually via the parameter `maxn`. Since opening URLs in an external browser costs computing resources on your local machine, make sure to use only values for `maxn` that do not slow down or make your computer unresponsive.

```
mp_view_originals(party > 41000 & party < 41999, maxn = 20)
```

2 Processing and analysing the corpus documents

As in `tm`, the textual content of a document is returned by the function `content`:

```
txt <- content(my_corpus[[2]])
class(txt)
```

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

```
head(txt, n = 4)
```

```
## [1] "1 Lebensqualität"
## [2] "1.1 Grüne Energiewende"
## [3] "Lebensqualität bedeutet in einer unversehrten Umwelt zu leben."
## [4] "Die Verantwortung dafür liegt bei uns: Wir alle gestalten Umwelt."
```

2.1 Working with the CMP codings

The central way for accessing the CMP codings is the accessor method `codes(...)`. It can be called on `ManifestoDocuments` and `ManifestoCorpuss` and returns a vector of the CMP codings attached to the quasi-sentences of the document/corpus in a row:

```
doc <- my_corpus[[2]]
head(codes(doc), n = 15)
```

```
##  [1]  NA  NA 501 606 501 501 501 416 416 412 503 411 501 416  NA
```

```
head(codes(my_corpus), n = 15)
```

```
##  [1] 305 305 305  NA  NA  NA 601 416 416 107 107 107 416 416 416
```

Thus you can for example use R's functionality to count the codes or select quasi- sentences (units of texts) based on their code:

```
table(codes(doc))
```

```
##
## 104 105 106 107 108 109 201 202 203 303 305 401 402 403 408 409 411 412
##   3   9   2  52  36  11  36  17   1   3   1   2   6  20   1   1  38  17
## 413 416 501 502 503 504 506 601 604 605 606 607 608 701 703 704 706
##   1  13  62  48  83  24  46  14  20   9  10  15   5  33  13   9  32
```

```
doc_subcodes <- subset(doc, codes(doc) %in% c(202, 503, 607))
length(doc_subcodes)
```

```
## [1] 115
```

```
length(doc_subcodes)/length(doc)
```

```
## [1] 0.149
```

The CMP coding scheme can be found in the online documentation of the Manifesto Project dataset at https://manifesto-project.wzb.eu/coding_schemes/1.

2.2 Text mining tools

Since the Manifesto Corpus uses the infrastructure of the `tm` package, all of `tm`s filtering and transformation functionality can be applied directly to the downloaded `ManifestoCorpus`.

For example, standard natural language processors are available to clean the corpus:

```
head(content(my_corpus[[3]]))
```

```
## [1] "1. SONNE STATT ÖL: WIR  HELFEN  BEIM  SPAREN"
## [2] "Der Umstieg hat begonnen."
## [3] "Die Menschen in Österreich fahren weniger Auto"
## [4] "und mehr mit dem öffentlichen Verkehr"
## [5] "und dem Rad."
## [6] "Sie sanieren Häuser und Wohnungen"
```

```
corpus_cleaned <- tm_map(my_corpus, removePunctuation)
corpus_nostop <- tm_map(corpus_cleaned, removeWords, stopwords("german"))
head(content(corpus_nostop[[3]]))
```

```
## [1] "1 SONNE STATT ÖL WIR  HELFEN  BEIM  SPAREN"
## [2] "Der Umstieg  begonnen"
## [3] "Die Menschen  Österreich fahren weniger Auto"
## [4] " mehr   öffentlichen Verkehr"
## [5] "  Rad"
## [6] "Sie sanieren Häuser  Wohnungen"
```

So is analysis in form of term document matrices:

```
tdm <- TermDocumentMatrix(corpus_nostop)
inspect(tdm[c("menschen", "wahl", "familie"),])
```

```
## <<TermDocumentMatrix (terms: 3, documents: 15)>>
## Non-/sparse entries: 36/9
## Sparsity           : 20%
## Maximal term length: 8
## Weighting          : term frequency (tf)
##
##           Docs
## Terms       1  2  3  4  5  6  7  8 9 10 11 12 13 14 15
##   menschen 65 41 20 15 78 24 50 38 0  6 47 49 27  8  3
##   wahl      2  0  3  3  2  0  0  1 0  0  2  0  1  1  0
##   familie   2  4  2  0  2  3  2 17 3  1 20 20 12  4  6
```

```
findAssocs(tdm, "stadt", 0.97) ## find correlated terms, see ?tm::findAssocs
```

```
##                       stadt
## schrittweise           0.99
## auszubauen             0.98
## erfordert              0.98
## övp                    0.98
```

```
## pflegeberufe             0.98
## denkmalschutz            0.97
## dienstes                 0.97
## geprüft                  0.97
## nonprofitorganisationen  0.97
```

For more information about the functionality provided by the `tm`, please refer to its documentation.

2.3 Selecting relevant parts of text

For applications in which not the entire text of a document is of interest, but rather a subset of the quasi-sentences matching certain criteria, `manifestoR` provides a function `subset(...)` working just like R's internal `subset` function.

It can, for example, be used to filter quasi-sentences based on codes or the text:

```
# subsetting based on codes (as example above)
doc_subcodes <- subset(doc, codes(doc) %in% c(202, 503, 607))
length(doc_subcodes)
```

```
## [1] 115
```

```
# subsetting based on text
doc_subtext <- subset(doc, grepl("Demokratie", content(doc)))
head(content(doc_subtext), n = 3)
```

```
## [1] "Eine Demokratie benötigt auch die Unterstützung von Forschung jenseits wirtschaftlicher Interess
## [2] "In einer Demokratie sollen all jene wählen dürfen, die von den politischen Entscheidungen betro
## [3] "Demokratie braucht die Teilhabe der BürgerInnen."
```

```
head(codes(doc_subtext), n = 10)
```

```
## [1] 506 202 202 201 108  NA 202 107
```

Via `tm_map` the filtering operations can also be applied to an entire corpus:

```
corp_sub <- tm_map(my_corpus, function(doc) {
  subset(doc, codes(doc) %in% c(202, 503, 607))
})
head(content(corp_sub[[3]]))
```

```
## [1] "Das hat einen einzigen Grund: die hohen Öl- und Gaspreise."
## [2] "Immer mehr Menschen können sich Heizung"
## [3] "und Mobilität immer weniger leisten."
## [4] "Ob wir das wollen oder nicht - Erdöl und Erdgas werden weiter teurer."
## [5] "Wir verbrennen Milliarden in unseren Tanks und Öfen,"
## [6] "und: SPAREN STATT VERSCHWENDEN."
```

```
head(codes(corp_sub))
```

```
## [1] 503 202 202 503 503 503
```

For convenience, it is also possible to filter quasi-sentences with specific codes directly when downloading a corpus. For this, the additional argument `codefilter` with a list of CMP codes of interest is passed to `mp_corpus`:

```
corp_sub <- mp_corpus(countryname == "Australia", codefilter = c(103, 104))
```

```
## Connecting to Manifesto Project DB API...
## Connecting to Manifesto Project DB API...
```

```
head(content(corp_sub[[1]]))
```

```
## [1] "The pursuit of military and economic dominance by the United States at the expense of internati
## [2] "and Iraqis allowed their self-determination."
## [3] "while maintaining an adequate defence force"
```

```
head(codes(corp_sub))
```

```
## [1] 103 103 104 104 103 103
```

2.4 Using the document metadata

Each document in the Manifesto Corpus has meta information about itself attached. They can be accessed via the function `meta`:

```
meta(doc)
```

```
## Metadata:
##   manifesto_id               : 1580
##   party                      : 42110
##   date                       : 200610
##   language                   : german
##   is_primary_doc             : TRUE
##   may_contradict_core_dataset: FALSE
##   md5sum_text                : 2ff28d1cc0161b75b0010cf486877f08
##   url_original               : /uploads/attach/file/5238/42110_2006.pdf
##   md5sum_original            : 8fd5726c6363864c3ace6e2d497d647e
##   annotations                : TRUE
##   id                         : 2
```

It is possible to access and also modify specific metadata entries:

```
meta(doc, "party")
```

```
## [1] 42110
```

```
meta(doc, "manual_edits") <- TRUE
meta(doc)
```

```
## Metadata:
##   manifesto_id               : 1580
##   party                      : 42110
##   date                       : 200610
##   language                   : german
##   is_primary_doc             : TRUE
##   may_contradict_core_dataset: FALSE
##   md5sum_text                : 2ff28d1cc0161b75b0010cf486877f08
##   url_original               : /uploads/attach/file/5238/42110_2006.pdf
##   md5sum_original            : 8fd5726c6363864c3ace6e2d497d647e
##   annotations                : TRUE
##   id                         : 2
##   manual_edits               : TRUE
```

Document metadata can also be bulk-downloaded with the function `mp_metadata`, taking the same set of parameters as `mp_corpus`:

```
metas <- mp_metadata(countryname == "Spain")
```

```
## Connecting to Manifesto Project DB API...
```

```
head(metas)
```

```
## Source: local data frame [6 x 10]
##
##   party   date language is_primary_doc may_contradict_core_dataset
## 1 33908 201111 galician           TRUE                        FALSE
## 2 33907 201111  spanish           TRUE                        FALSE
## 3 33905 201111  catalan           TRUE                        FALSE
## 4 33902 201111  spanish           TRUE                        FALSE
## 5 33611 201111  catalan           TRUE                        FALSE
## 6 33610 201111  spanish           TRUE                        FALSE
## Variables not shown: manifesto_id (int), md5sum_text (chr), url_original
##   (chr), md5sum_original (chr), annotations (lgl)
```

The field ...

- ... `party` contains the party id from the Manifesto Project Dataset.
- ... `date` contains the month of the election in the same format as in the Manifesto Project Dataset (`YYYYMM`)
- ... `language` specifies the language of the document as a word.
- ... `is_primary_doc` is FALSE only in cases where for a single party and election date multiple manifestos are available and this is the document not used for coding by the Manifesto Project.
- ... `may_contradict_core_dataset` is TRUE for documents where the CMP codings in the corpus documents might be inconsistent with the coding aggregates in the Manifesto Project's Main Dataset. This applies to manifestos which have been either recoded after they entered the dataset or cases where the dataset entries are derived from hand-written coding sheets used prior to the digitalization of the Manifesto Project's data workflow, but the documents were digitalized and added to the Manifesto Corpus afterwards.

- ... `annotations` is TRUE whenenver there are CMP codings available for the document.

The other metadata entries have primarily technical functions for communication between the `manifestoR` package and the online database.

3 Efficiency and reproducibility: caching and versioning

To save time and network traffic, `manifestoR` caches all downloaded data and documents in your computer's working memory and connects to the online database only when data is required that has not been downloaded before.

```
corpus <- mp_corpus(wanted)
```

```
## Connecting to Manifesto Project DB API...
## Connecting to Manifesto Project DB API...
```

```
subcorpus <- mp_corpus(wanted[3:7,])
```

Note that in the second query no message informing about the connection to the Manifesto Project's Database is printed, since no data is actually downloaded.

This mechanism also ensures **reproducibility** of your scripts, analyses and results: executing your code again will yield the same results, even if the Manifesto Project's Database is updated in the meantime. Since the cache is only stored in the working memory, however, in order to ensure reproducibility across R sessions, it is advisable to **save the cache to the hard drive** at the end of analyses and load it in the beginning:

```
mp_save_cache(file = "manifesto_cache.RData")

## ... start new R session ... then:

library(manifestoR)
mp_setapikey("manifesto_apikey.txt")
mp_load_cache(file = "manifesto_cache.RData")
```

This way `manifestoR` always works with the same snapshot of the Manifesto Project Database and Corpus, saves a lot of unnecessary online traffic and also enables you to continue with your analyses offline.

Each snapshot of the Manifesto Corpus is identified via a timestamp, which is stored in the cache together with the data and can be accessed via

```
mp_which_corpus_version()
```

```
## [1] "20150330165855"
```

When collaborating on a project with other researchers, it is advisable to use the same corpus version for reproducibility of the results. `manifestoR` can be set to use a specific version id with the functions

```
mp_use_corpus_version("20150306152837")
```

```
## Connecting to Manifesto Project DB API...
```

In order to guarantee reproducibility of **published work**, please also mention the corpus version id used for the reported analyses in the publication.

For updating locally cached data to the most recent version of the Manifesto Project Corpus, `manifestoR` provides two functions:

```
mp_check_for_corpus_update()
```

```
## $update_available
## [1] TRUE
##
## $versionid
## [1] "20150330165855"
```

```
mp_update_cache()
```

```
## Connecting to Manifesto Project DB API...

## [1] "20150330165855"
```

```
mp_check_for_corpus_update()
```

```
## $update_available
## [1] FALSE
##
## $versionid
## [1] "20150330165855"
```

For more detailed information on the caching mechanism and on how to use and load specific snapshots of the Manifesto Corpus, refer to the R documentations of the functions mentioned here as well `mp_use_corpus_version`, `mp_corpusversions`, `mp_which_corpus_version`.

4 Exporting documents

If required `ManifestoCorpus` as well as `ManifestoDocument` objects can be converted to R's internal `data.frame` format and processed further:

```
doc_df <- as.data.frame(doc)
head(within(doc_df, {
  ## for pretty printing
  content <- paste0(substr(content, 1, 60), "...")
}))
```

```
##                                                        content code pos
## 1                                          1 Lebensqualität...   NA   1
## 2                                     1.1 Grüne Energiewende...   NA   2
## 3 Lebensqualität bedeutet in einer unversehrten Umwelt zu lebe...  501   3
## 4 Die Verantwortung dafür liegt bei uns: Wir alle gestalten Um...  606   4
## 5 Ein Umdenken in der Energiepolitik ist eine wesentliche Vora...  501   5
## 6 Wir Grüne stehen für eine Energiewende hin zu einem Aufbruch...  501   6
```

The function also provides a parameter to include all available metadata in the export:

```
doc_df_with_meta <- as.data.frame(doc, with.meta = TRUE)
print(names(doc_df_with_meta))
```

```
##  [1] "content"                     "code"
##  [3] "pos"                         "manifesto_id"
##  [5] "party"                       "date"
##  [7] "language"                    "is_primary_doc"
##  [9] "may_contradict_core_dataset" "md5sum_text"
## [11] "url_original"                "md5sum_original"
## [13] "annotations"                 "id"
## [15] "manual_edits"
```

For more information on the available metadata per document, refer to the section Using the document metadata above.

Note again that also all functionality provided by `tm`, such as `writeCorpus` is available on a `ManifestoCorpus`.

5 Additional Information

5.1 Contacting the Manifesto Project team

You can get in touch with the Manifesto Project team by e-mailing to manifesto-communication@wzb.eu. We are happy to receive your feedback and answer questions about the Manifesto Corpus, including errors or obscurities in the corpus documents. In this case please make sure to include the party id, election date and the corpus version you were working with (accessible via `mp_which_corpus_version`). For general questions about the Project and dataset, please check the Frequently Asked Questions section on our website first.

We also welcome bug reports, feature requests or (planned) source code contributions for the `manifestoR` package.