Working with the manifestoR package

Jirka Lewandowski jirka.lewandowski@wzb.eu

21/03/2016

Contents

1 Downloading documents from the Manifesto Corpus

When publishing work using the Manifesto Corpus, please make sure to cite it correctly and to give the identification number of the corpus version used for your analysis.

You can print citation and version information with the function `mp_cite()`.

1.1 Loading the package

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

```
library(manifestoR)
```

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 R 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 the Manifesto Project Dataset

You can download the Manifesto Project Dataset (MPDS) with the function `mp_maindataset()`. By default the most recent update is returned, but you can specify older versions to get for reproducibility (type `mp_coreversions()` for a list of version and `?mp_maindataset` for usage information). You can get the Manifesto Project South America Dataset using the function `mp_southamerica_dataset()` that works analogously to `mp_maindataset()`. For analysing the dataset using scaling functions, refer to the section Using manifestoR's scaling functions below.

1.4 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... corpus version: 2016-1
## Connecting to Manifesto Project DB API... corpus version: 2016-1
## Connecting to Manifesto Project DB API... corpus version: 2016-1
## Connecting to Manifesto Project DB API... corpus version: 2016-1
```

```
my_corpus
```

```
## <<ManifestoCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 15
```

`mp_corpus` returns a `ManifestoCorpus` object, a subclass of `Corpus` as defined in the natural language processing package `tm` (Feinerer & Hornik 2015). Following `tm`s logic, a `ManifestoCorpus` consists of `ManifestoDocuments`. Documents in corpus can be indexed via their `manifesto_id` (consisting of the CMP party code, an underscore, and either the election year, if unambigous, or the election year and month) or via their position in the corpus. For both, corpus and documents, `tm` provides accessor functions to the corpus and documents content and metadata:

```
head(content(my_corpus[["42110_2002"]]))
```

```
## [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"
```

```
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[["42110_2002"]])
```

```
##   manifesto_id               : 42110_2002
##   party                      : 42110
##   date                       : 200211
##   language                   : german
##   source                     : MARPOR
##   has_eu_code                : FALSE
##   is_primary_doc             : TRUE
##   may_contradict_core_dataset: TRUE
##   md5sum_text                : 04f07de517283243fdaaf0cbddc2a09e
##   url_original               : NA
##   md5sum_original            : NA
##   annotations                : TRUE
##   id                         : 42110_2002
```

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... corpus version: 2016-1
## Connecting to Manifesto Project DB API... corpus version: 2016-1
```

```
## <<ManifestoCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 5
```

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... corpus version: 2016-1
## Connecting to Manifesto Project DB API... corpus version: 2016-1
```

```
## <<ManifestoCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 1
```

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... corpus version: 2016-1
```

```
##           Queried for        Corpus Version       Documents found
##                     6                2016-1             6 (100%)
## Coded Documents found       Originals found            Languages
##           5 (83.333%)             6 (100%)          1 (german)
```

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

1.5 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[["42110_2006"]])
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[["42110_2006"]]
head(codes(doc), n = 15)
```

```
##  [1] NA    NA    "501" "606" "501" "501" "501" "416" "416" "412" "503"
## [12] "411" "501" "416" NA
```

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

```
##  [1] "305" "305" "305" NA    NA    NA    "601" "416" "416" "107" "107"
## [12] "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] 0.1489637
```

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 Working with additional layers of codings

Besides the main layer of CMP codings, you can create, store and access additional layers of codings in `ManifestoDocuments` by passing a name of the coding layer as additional argument to the function `codes()`:

```
## assigning a dummy code of alternating As and Bs
codes(doc, "my_code") <- rep_len(c("A", "B"), length.out = length(doc))
head(codes(doc, "my_code"))
```

```
## [1] "A" "B" "A" "B" "A" "B"
```

You can view the names of the coding layers stored in a `ManifestoDocument` with the function `code_layers()`:

```
code_layers(doc)
```

```
## [1] "cmp_code" "eu_code"  "my_code"
```

Note that certain documents downloaded from the Manifesto Corpus Database already have a second layer of codes named `eu_code`. These are codes that have been assigned to quasi-sentences by CMP coders additionally to the main CMP code to indicate policy statements that should or should not be implemented on the level of the European union. The documents that were coded in this way are marked in the corpus' metadata with the flag `has_eu_code` (see below Using the document metadata). Note that, since these codes also have been used for computing the `per` and `rile` variables in the Manifesto Project Main Dataset, they are also used in `manifestoRs` `count_codes` and `rile` functions (see below Scaling texts) if the respective metadata flag is present.

2.3 Text mining tools

Since the Manifesto Corpus uses the infrastructure of the `tm` package (Feinerer & Hornik 2015), all of `tms` 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[["42110_2008"]]))
```

```
## [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[["42110_2008"]]))
```

```
## [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      42110_2002 42110_2006 42110_2008 42220_2008 42320_2002
##   menschen         65         41         20         15         78
##   wahl              2          0          3          3          2
##   familie           2          4          2          0          2
##           Docs
## Terms      42320_2006 42320_2008 42420_2002 42420_2006 42420_2008
##   menschen         24         50         38          0          6
##   wahl              0          0          1          0          0
##   familie           3          2         17          3          1
##           Docs
## Terms      42520_2002 42520_2006 42520_2008 42710_2006 42710_2008
##   menschen         47         49         27          8          3
##   wahl              2          0          1          1          0
##   familie          20         20         12          4          6
```

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

```
## $stadt
##            schrittweise              auszubauen               erfordert
##                    0.99                    0.98                    0.98
##                    övp            pflegeberufe           denkmalschutz
##                    0.98                    0.98                    0.97
##               dienstes                geprüft nonprofitorganisationen
##                    0.97                    0.97                    0.97
```

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

2.4 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... corpus version: 2016-1
## Connecting to Manifesto Project DB API... corpus version: 2016-1
```

```
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"
```

2.5 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)
```

```
##   manifesto_id               : 42110_2006
##   party                      : 42110
##   date                       : 200610
##   language                   : german
##   source                     : MARPOR
##   has_eu_code                : FALSE
##   is_primary_doc             : TRUE
##   may_contradict_core_dataset: FALSE
##   md5sum_text                : 37744e88ed32bbf176656883526fc56c
##   url_original               : /down/originals/42110_2006.pdf
##   md5sum_original            : CURRENTLY_UNAVAILABLE
##   annotations                : TRUE
##   id                         : 42110_2006
```

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

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

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

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

```
##   manifesto_id               : 42110_2006
##   party                      : 42110
##   date                       : 200610
##   language                   : german
##   source                     : MARPOR
##   has_eu_code                : FALSE
##   is_primary_doc             : TRUE
##   may_contradict_core_dataset: FALSE
##   md5sum_text                : 37744e88ed32bbf176656883526fc56c
##   url_original               : /down/originals/42110_2006.pdf
##   md5sum_original            : CURRENTLY_UNAVAILABLE
##   annotations                : TRUE
##   id                         : 42110_2006
##   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... corpus version: 2016-1
```

```
head(metas)
```

```
## Source: local data frame [6 x 12]
##
##   party   date language source has_eu_code is_primary_doc
##   (dbl)  (dbl)    (chr)  (chr)       (lgl)          (lgl)
## 1 33908 201111 galician MARPOR       FALSE           TRUE
## 2 33907 201111  spanish MARPOR       FALSE           TRUE
## 3 33905 201111  catalan MARPOR       FALSE           TRUE
## 4 33902 201111  spanish MARPOR       FALSE           TRUE
## 5 33611 201111  catalan MARPOR       FALSE           TRUE
## 6 33610 201111  spanish MARPOR       FALSE           TRUE
## Variables not shown: may_contradict_core_dataset (lgl), manifesto_id
##   (chr), 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.
- ... `has_eu_code` marks document in which the additional coding layer `eu_code` is present. These codes have been assigned to quasi-sentences by CMP coders additionally to the main CMP code to indicate policy statements that should or should not be implemented on the level of the European union.

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... corpus version: 2016-1
## Connecting to Manifesto Project DB API... corpus version: 2016-1
```

```
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 version number, which is stored in the cache together with the data and can be accessed via

```
mp_which_corpus_version()
```

```
## [1] "2016-1"
```

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 with the functions

```
mp_use_corpus_version("2015-3")
```

```
## Connecting to Manifesto Project DB API... corpus version: 2015-3
```

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] "2016-1"
```

```
mp_update_cache()
```

```
## Connecting to Manifesto Project DB API... corpus version: 2016-1

## [1] "2016-1"
```

```
mp_check_for_corpus_update()
```

```
## $update_available
## [1] FALSE
##
## $versionid
## [1] "2016-1"
```

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
  text <- paste0(substr(text, 1, 60), "...")
}))
```

```
##                                                                text cmp_code
## 1                                                1 Lebensqualität...     <NA>
## 2                                           1.1 Grüne Energiewende...     <NA>
## 3 Lebensqualität bedeutet in einer unversehrten Umwelt zu lebe...      501
## 4 Die Verantwortung dafür liegt bei uns: Wir alle gestalten Um...      606
## 5 Ein Umdenken in der Energiepolitik ist eine wesentliche Vora...      501
## 6 Wir Grüne stehen für eine Energiewende hin zu einem Aufbruch...      501
##   eu_code my_code pos
## 1      NA       A   1
## 2      NA       B   2
## 3      NA       A   3
## 4      NA       B   4
## 5      NA       A   5
## 6      NA       B   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] "text"                        "cmp_code"
##  [3] "eu_code"                     "my_code"
##  [5] "pos"                         "manifesto_id"
##  [7] "party"                       "date"
##  [9] "language"                    "source"
## [11] "has_eu_code"                 "is_primary_doc"
## [13] "may_contradict_core_dataset" "md5sum_text"
```

```
## [15] "url_original"                "md5sum_original"
## [17] "annotations"                 "id"
## [19] "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 Scaling texts

Scaling of political content refers to the estimation of its location in a policy space (Grimmer & Stewart 2013). `manifestoR` provides several functions to scale coded documents by known routines such as the RILE measure (see sections Using manifestoR's scaling functions), as well as infrastructure to create new scales (see section Writing custom scaling functions) and statistical analysis routines for the distributions of scaling functions (see section Bootstrapping scaling function distributions and standard errors).

5.1 Using manifestoR's scaling functions

Implementationwise, a scaling function in `manifestoR` takes a data.frame of cases and outputs a position value for each case. The Manifesto Project Dataset (MPDS) can be downloaded in `manifestoR` using the function `mp_maindataset()` (see section Downloading the Manifesto Project Dataset above). Then you can e.g. compute the RILE scores of cases from the main dataset by calling:

```
mpds <- mp_maindataset()
```

```
## Connecting to Manifesto Project DB API...
## Connecting to Manifesto Project DB API... corpus version: 2016-1
```

```
rile(subset(mpds, countryname == "Albania"))
```

```
##  [1]  1.592900e+01 -1.146300e+01  1.027400e+01  1.111100e+01  7.176000e+00
##  [6]  1.792300e+01  5.405000e+00  5.882000e+00 -7.298000e+00 -1.354000e+01
## [11]  6.012000e+00  4.232200e+01  1.431200e+01  2.220446e-16 -9.090000e+00
## [16] -9.350000e-01 -2.187000e+00 -9.180000e-01  5.596200e+01 -1.304900e+01
## [21]  8.059000e+00  9.919000e+00 -4.166000e+00  7.760000e-01  5.710000e-01
## [26] -2.187000e+00 -9.180000e-01  5.596200e+01  2.718500e+01  2.428600e+01
## [31]  9.919000e+00 -4.166000e+00  2.247200e+01  5.710000e-01 -2.187000e+00
## [36] -9.180000e-01 -4.166000e+00  2.247200e+01
```

What variables are used from the input data depends on the scaling function. All currently implemented functions use only the percentages of coded categories, in the form of variables starting with "per" as in the Manifesto Project Dataset. The following functions are currently available:

- RILE according to Laver & Budge (1992): `rile`
- logit rile according to Lowe et al. (2011): `logit_rile`
- Vanilla scaling according to Gabel & Huber (2000): `vanilla`
- Franzmann & Kaiser (2009): `franzmann_kaiser`
- Issue attention diversity according to Greene (2015): `issue_attention_diversity`

- Programmatic clarity measure according to Giebeler et al. (2015): `mp_clarity`
- Nicheness measures according to Bischof (2015) or Meyer and Miller (2013): `mp_nicheness`
- ... (more scaling functions are planned and contributions are welcome, see Contributing to manifestoR).

To apply scaling functions directly to coded documents or corpora you can use the function `mp_scale`. It takes a `ManifestoCorpus` or `ManifestoDocument` and returns the scaled positions for each document:

```
corpus <- mp_corpus(countryname == "Romania")
```

```
## Connecting to Manifesto Project DB API... corpus version: 2016-1
## Connecting to Manifesto Project DB API... corpus version: 2016-1
## Connecting to Manifesto Project DB API... corpus version: 2016-1
```

```
mp_scale(corpus, scalingfun = logit_rile)
```

```
##   party   date logit_rile
## 1 93031 201212  0.7176668
## 2 93061 201212 -0.4813032
## 3 93981 201212 -0.7845814
```

5.2 Writing custom scaling functions

Writing custom scaling functions for texts in `manifestoR` is easy, since it requires nothing more than writing a function that takes a `data.frame` of cases as input and returns a vector of values. `mp_scale` provides the mechanism that converts a coded text to a `data.frame` with "per" variables such that your function can handle it:

```
custom_scale <- function(data) {
  data$per402 - data$per401
}
mp_scale(corpus, scalingfun = custom_scale)
```

```
##   party   date custom_scale
## 1 93031 201212     1.558442
## 2 93061 201212     3.296703
## 3 93981 201212     2.962963
```

In addition, `manifestoR` provides several function templates you can use for creating scales, e.g. a weighted sum of per variables (`scale_weighted`), the logit ratio of category counts (`scale_logit`) or ratio scaling as suggested by Kim and Fording (1998) and by Laver & Garry (2000) (`scale_ratio`). For example, the ratio equivalent to the simple function above can be implemented as:

```
custom_scale <- function(data) {
    scale_ratio(data, pos = c("per402"), neg = c("per401"))
}
mp_scale(corpus, scalingfun = custom_scale)
```

```
##   party   date custom_scale
## 1 93031 201212     1.857143
## 2 93061 201212     4.000000
## 3 93981 201212     2.333333
```

For details on these template functions, their parameters and how to use them, see the R documentation `?scale`.

5.3 Bootstrapping scaling function distributions and standard errors

In order to better evaluate the significance of analysis results based on scaled coded texts, Benoit, Mikhaylov, and Laver (2009) proposed to approximate the standard errors of the scale variable by bootstrapping its distribution. This procedure is available via the function `mp_bootstrap`:

```
data <- subset(mpds, countryname == "Albania")
mp_bootstrap(data, fun = rile)
```

```
## Source: local data frame [38 x 2]
##
##       rile       sd
##      (dbl)    (dbl)
## 1   15.929 7.318014
## 2  -11.463 4.185576
## 3   10.274 4.570369
## 4   11.111 8.175354
## 5    7.176 4.250592
## 6   17.923 5.777985
## 7    5.405 4.241471
## 8    5.882 6.826547
## 9   -7.298 4.095668
## 10 -13.540 7.145683
## ..     ...      ...
```

Note that the argument `fun` can be any scaling function and the returned `data.frame` containes the scaled position as well as the bootstrapped standard deviations. Also, with the additional parameters `statistics`, you can compute arbitrary statistics from the bootstrapped distribution, such as variance or quantiles:

```
custom_scale <- function(data) {
  data$per402 - data$per401
}
mp_bootstrap(data,
             fun = custom_scale,
             statistics = list(var, 0.025, 0.975))
```

```
## Source: local data frame [38 x 4]
##
##    custom_scale        var     q0.025   q0.975
##           (dbl)      (dbl)      (dbl)    (dbl)
## 1         0.885  2.3248842  -1.769912 3.539823
## 2        -0.395  0.1525735  -1.185771 0.000000
## 3         0.685  1.4881418  -1.369863 3.424658
## 4        -1.111  1.2721660  -3.333333 0.000000
## 5         0.000  2.7733106  -3.349282 3.349282
## 6         0.000  1.9598238  -2.830189 2.830189
## 7        -1.081  1.2012103  -3.243243 1.081081
## 8        -5.882 16.6861585 -14.705882 0.000000
## 9         0.000  0.5205435  -1.459854 1.459854
## 10        3.125  2.8468127   0.000000 7.291667
## ..          ...        ...        ...      ...
```

6 Additional Information

For a more detailed reference and complete list of the functions provided by `manifestoR`, see the R package reference manual on CRAN: http://cran.r-project.org/web/packages/manifestoR/manifestoR.pdf

6.1 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.

6.2 Contributing to manifestoR

We welcome bug reports, feature requests or (planned) source code contributions for the `manifestoR` package. For all of these, best refer to our repository on github: https://github.com/ManifestoProject/manifestoR. For more information, please refer to the Section "Developing" in the README file of the github repository.

7 References

Benoit, K., Laver, M., & Mikhaylov, S. (2009). Treating Words as Data with Error: Uncertainty in Text Statements of Policy Positions. American Journal of Political Science, 53(2), 495-513. http://doi.org/10.1111/j.1540-5907.2009.00383.x

Bischof, D. (2015). Towards a Renewal of the Niche Party Concept Parties, Market Shares and Condensed Offers. Party Politics.

Feinerer, I., & Hornik, K. (2015). Tm: Text Mining Package. http://cran.r-project.org/web/packages/tm/index.html

Franzmann, S., & Kaiser, A. (2006): Locating Political Parties in Policy Space. A Reanalysis of Party Manifesto Data, Party Politics, 12:2, 163-188

Gabel, M. J., & Huber, J. D. (2000). Putting Parties in Their Place: Inferring Party Left-Right Ideological Positions from Party Manifestos Data. American Journal of Political Science, 44(1), 94-103.

Giebler, H., Lacewell, O.P., Regel, S., and Werner, A. (2015). Niedergang oder Wandel? Parteitypen und die Krise der repraesentativen Demokratie. In Steckt die Demokratie in der Krise?, ed. Wolfgang Merkel, 181-219. Wiesbaden: Springer VS

Greene, Z. (2015). Competing on the Issues How Experience in Government and Economic Conditions Influence the Scope of Parties' Policy Messages. Party Politics.

Grimmer, J., & Stewart, B.. 2013. Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts. Political Analysis 21(3): 267–97.

Kim, H., & Fording, R. C. (1998). Voter ideology in western democracies, 1946-1989. European Journal of Political Research, 33(1), 73-97.

Laver, M. & Budge, I., eds. (1992). Party Policy and Government Coalitions, Houndmills, Basingstoke, Hampshire: The MacMillan Press 1992

Laver, M., & Garry, J. (2000). Estimating Policy Positions from Political Texts. American Journal of Political Science, 44(3), 619-634.

Lehmann, P., Matthieß, T., Merz, N., Regel, S., & Werner, A. (2015): Manifesto Corpus. Version: 2015-4. Berlin: WZB Berlin Social Science Center.

Lowe, W., Benoit, K., Mikhaylov, S., & Laver, M. (2011). Scaling Policy Preferences from Coded Political Texts. Legislative Studies Quarterly, 36(1), 123-155.

Meyer, T.M., & Miller, B. (2013). The Niche Party Concept and Its Measurement. Party Politics 21(2): 259-271.

Volkens, A., Lehmann, P., Matthieß, T., Merz, N., Regel, S., & Werner, A (2015): The Manifesto Data Collection. Manifesto Project (MRG/CMP/MARPOR). Version 2015a. Berlin: Wissenschaftszentrum Berlin für Sozialforschung (WZB)