Working with the manifestoR package

Jirka Lewandowski jirka.lewandowski@wzb.eu

21/03/2016 (updated 16/05/2024)

Contents

Hint: See https://manifesto-project.wzb.eu for additional **tutorials**, **documentation**, **data**, and **election programmes**.

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). 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: 2024-1
## Connecting to Manifesto Project DB API... corpus version: 2024-1
## Connecting to Manifesto Project DB API... corpus version: 2024-1
```

```
## Your query resulted in 33 requested document items containing corpus metadata. Of these
    items 1 could not be retrieved (reasons: 1 having no machine-readable texts).
```

```
my_corpus
```

```
## <<ManifestoCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 32
```

`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_200211"]]))
```

```
## [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_200211"]])
```

```
##   manifesto_id               : 42110_200211
##   party                      : 42110
##   date                       : 200211
##   language                   : german
##   source                     : MARPOR
##   has_eu_code                : FALSE
##   is_primary_doc             : TRUE
##   may_contradict_core_dataset: FALSE
##   md5sum_text                : 4e1877c110c3d01db9eaf864310cc0b0
##   url_original               : NA
##   md5sum_original            : NA
##   annotations                : TRUE
##   handbook                   : 3
##   is_copy_of                 : NA
##   title                      : Österreich braucht jetzt die Grünen. Das Wahlprogramm
##   translation_en             : TRUE
##   id                         : 42110_200211
```

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.

If you want to get your results as a `tibble/data.frame` object instead of a `ManifestoCorpus` object, you can simply set the `as_tibble` parameter of `mp_corpus` to TRUE or use the convenience shorthand function `mp_corpus_df` that does the same. By default it also contains the the main document-level metadata ("manifesto_id", "party", "date", "language", "annotations", "translation_en") but you can also opt for "none" or "all" metadata by specifying the `tibble_metadata` parameter accordingly.

```
my_corpus_df <- mp_corpus(countryname == "Austria" & edate > as.Date("2000-01-01"),
                          as_tibble = TRUE)
```

```
## Your query resulted in 33 requested document items containing corpus metadata. Of these
    items 1 could not be retrieved (reasons: 1 having no machine-readable texts).
```

```
# or alternatively with the shorthand function:
# my_corpus_df <- mp_corpus_df(countryname == "Austria" & edate > as.Date("2000-01-01"))
# and to include all document-level metadata
# my_corpus_df <- mp_corpus_df(countryname == "Austria" & edate > as.Date("2000-01-01"),
#                              tibble_metadata = "all")
my_corpus_df
```

```
## # A tibble: 39,517 x 10
##    text    cmp_code eu_code   pos manifesto_id party   date language annotations
##    <chr>   <chr>    <chr>   <int> <chr>        <dbl>  <dbl> <chr>    <lgl>
##  1 "Wir k~ 305      <NA>        1 42110_200211 42110 200211 german   TRUE
##  2 künfti~ 305      <NA>        2 42110_200211 42110 200211 german   TRUE
##  3 Oder s~ 305      <NA>        3 42110_200211 42110 200211 german   TRUE
##  4 Dr. Ev~ <NA>     <NA>        4 42110_200211 42110 200211 german   TRUE
##  5 Österr~ <NA>     <NA>        5 42110_200211 42110 200211 german   TRUE
##  6 Nachha~ <NA>     <NA>        6 42110_200211 42110 200211 german   TRUE
##  7 Die Zu~ 601      <NA>        7 42110_200211 42110 200211 german   TRUE
##  8 Neue T~ 416      <NA>        8 42110_200211 42110 200211 german   TRUE
##  9 In Ver~ 416      <NA>        9 42110_200211 42110 200211 german   TRUE
## 10 Sorglo~ 107      <NA>       10 42110_200211 42110 200211 german   TRUE
## # i 39,507 more rows
## # i 1 more variable: translation_en <lgl>
```

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 (please keep in mind that this mechanism always uses the most recent version of the Dataset) or your current R environment. The Main Dataset itself is available in manifestoR via the function `mp_maindataset()`:

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

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

```
mp_corpus(rile > 60)
```

```
## Connecting to Manifesto Project DB API... corpus version: 2024-1
## Connecting to Manifesto Project DB API... corpus version: 2024-1

## Your query resulted in 34 requested document items containing corpus metadata. Of these
    items 25 could not be retrieved (reasons: 4 having no real documents coded (see progtype
    3 and 99), 21 having no machine-readable texts).

## <<ManifestoCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 9
```

```
# ... which is the same as
# mp_corpus(mpds %>% filter(rile > 60))
```

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(201709, 201709))
mp_corpus(wanted)
```

```
## Connecting to Manifesto Project DB API... corpus version: 2024-1

## Warning: No document/metadata found with id 41220_201709. Please double check
## your request if it was specified manually.

## Connecting to Manifesto Project DB API... corpus version: 2024-1

## <<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 2017.

Also, not for every party and election observation of the Manifesto Project Dataset manifesto documents are available in the Manifesto Project Corpus and for some only the machine-readable texts are available but not the digitally annotated codes. In case of such missing documents you get an information message containing the number of missing documents and the reasons for the missingness. For details you can also check our corpus information webpage.

You can check the document availability of your query beforehand with the function `mp_availability(...)`:

```
mp_availability(countryname == "Belgium")
```

```
## Connecting to Manifesto Project DB API... corpus version: 2024-1
```

```
##                Queried for             Corpus Version
##                        194                     2024-1
##           Documents found      Coded Documents found
##             166 (85.567%)               48 (24.742%)
##           Originals found English Translations found
##             134 (69.072%)               48 (24.742%)
##                Languages
##          2 (french dutch)
```

And you can even use the result of this ability request for further specification of your corpus query as it is basically a data.frame that contains party-date combinations with the information on the availability of machine-readable text (`manifestos`), machine-readable codings (`annotations`), machine-readable english translations (`translation_en`), PDF originals (`originals`), and `language`:

```
avail <- mp_availability(countryname == "Belgium")
wanted_be <- avail %>% dplyr::filter(language == "dutch" & annotations == TRUE)
mp_corpus(wanted_be)
```

```
## Connecting to Manifesto Project DB API... corpus version: 2024-1
```

```
## <<ManifestoCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 30
```

Since corpus version 2024-1 there are also english translations available for a large number of documents in the corpus. You can can simply request them by setting the `translation`-parameter of the `mp_corpus` function to "en". This returns the english translation instead of the original language text. For documents that are originally in english, it returns the original english text. In case no translation is available for a requested document a not-found warning happens.

```
my_corpus_en = mp_corpus(wanted, translation = "en")
```

```
## Connecting to Manifesto Project DB API... corpus version: 2024-1
## Connecting to Manifesto Project DB API... corpus version: 2024-1
```

```
my_corpus_en
```

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

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

```
## [1] "It is time for more justice!"
## [2] "2017 is a decisive year."
## [3] "The SPD is running to set the course for the future of Germany and Europe
## with Martin Schulz as chancellor."
## [4] "The elections this year are fundamental decisions about what kind of
## society we want to live in."
## [5] "Values that were taken for granted are at stake."
## [6] "We fight for these values, as we have always done in our long history."
```

In case you need parallely both the original language text and the english translations you can make use of our convenience function `mp_corpus_df_bilingual` which returns a tibble/data.frame with the "text" column containing the original language text and a column "text_en" containing the english translation.

```
mp_corpus_df_bilingual(wanted, translation = "en")
```

```
## Connecting to Manifesto Project DB API... corpus version: 2024-1
```

```
## # A tibble: 2,750 x 11
##    text        text_en cmp_code eu_code   pos manifesto_id party   date language
##    <chr>       <chr>   <chr>    <chr>   <int> <chr>        <dbl>  <dbl> <chr>
##  1 Es ist Zei~ It is ~ H        <NA>        1 41320_201709 41320 201709 german
##  2 2017 ist e~ 2017 i~ 000      <NA>        2 41320_201709 41320 201709 german
##  3 Die SPD tr~ The SP~ 305.1    <NA>        3 41320_201709 41320 201709 german
##  4 Die Wahlen~ The el~ 000      <NA>        4 41320_201709 41320 201709 german
##  5 Werte, die~ Values~ 503      <NA>        5 41320_201709 41320 201709 german
##  6 Für diese ~ We fig~ 503      <NA>        6 41320_201709 41320 201709 german
##  7 Für eine G~ For a ~ 503      <NA>        7 41320_201709 41320 201709 german
##  8 Unabhängig~ Regard~ 503      <NA>        8 41320_201709 41320 201709 german
##  9 Unabhängig~ Regard~ 503      <NA>        9 41320_201709 41320 201709 german
## 10 Unabhängig~ Regard~ 503      <NA>       10 41320_201709 41320 201709 german
## # i 2,740 more rows
## # i 2 more variables: annotations <lgl>, translation_en <lgl>
```

You can get all available documents with english translations by querying first the metadata and then using the `translation_en` metadata value. To learn more about querying metadata check the section: Using the document metadata.

```
wanted_en = mp_metadata(TRUE) %>%
  dplyr::filter(translation_en == TRUE)
```

```
## Connecting to Manifesto Project DB API... corpus version: 2024-1
```

```
# ... or if you also want to include also the original english documents
wanted_en = mp_metadata(TRUE) %>%
  dplyr::filter(translation_en == TRUE | language == "english")

wanted_en %>%
  dplyr::select(party, date, manifesto_id, language, annotations, translation_en)
```

```
## # A tibble: 1,878 x 6
##    party   date manifesto_id language annotations translation_en
##    <dbl>  <dbl> <chr>        <chr>    <lgl>       <lgl>
##  1 11110 200609 11110_200609 swedish  TRUE        TRUE
##  2 11220 200609 11220_200609 swedish  TRUE        TRUE
##  3 11320 200609 11320_200609 swedish  TRUE        TRUE
##  4 11420 200609 11420_200609 swedish  TRUE        TRUE
##  5 11520 200609 11520_200609 swedish  TRUE        TRUE
##  6 11620 200609 11620_200609 swedish  TRUE        TRUE
##  7 11810 200609 11810_200609 swedish  TRUE        TRUE
##  8 11110 201009 11110_201009 swedish  TRUE        TRUE
##  9 11220 201009 11220_201009 swedish  TRUE        TRUE
## 10 11320 201009 11320_201009 swedish  TRUE        TRUE
## # i 1,868 more rows
```

```
# ... and to query their english text
# corpus <- mp_corpus(wanted_en, translation = "en")
# ... their english text directly as tibble/data.frame
# corpusdf <- mp_corpus_df(wanted_en, translation = "en")
```

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)
```

1.6 Accessing the category scheme and category descriptions

The main dataset and the corpus are using the alphanumerical category codes of the manifesto project category scheme. To get a data.frame of these codes together with their domains, variable names, titles, descriptions, labels etc. you can use the function `mp_codebook()` (or `mp_describe_code(...)` to show/use the information for the code(s) you are interested in).

```
mp_codebook()
```

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

## # A tibble: 143 x 8
##    type  domain_code domain_name  code  variable_name title description_md label
##    <chr> <chr>       <chr>        <chr> <chr>         <chr> <chr>          <chr>
##  1 main  0           NA           000   peruncod      No o~ "Share of unc~ perc~
##  2 main  1           External Re~ 101   per101        Fore~ "Favourable m~ fore~
##  3 main  1           External Re~ 102   per102        Fore~ "Negative men~ fore~
##  4 main  1           External Re~ 103   per103        Anti~ "Negative ref~ anti~
##  5 main  1           External Re~ 104   per104        Mili~ "The importan~ mili~
##  6 main  1           External Re~ 105   per105        Mili~ "Negative ref~ mili~
##  7 main  1           External Re~ 106   per106        Peace "Any declarat~ peace
##  8 main  1           External Re~ 107   per107        Inte~ "Need for int~ inte~
##  9 main  1           External Re~ 108   per108        Euro~ "Favourable m~ euro~
## 10 main  1           External Re~ 109   per109        Inte~ "Negative ref~ inte~
## # i 133 more rows
```

```
mp_describe_code("504")
```

```
## code: 504
## title: Welfare State Expansion
## description_md: Favourable mentions of need to introduce, maintain or expand any public
## social service or social security scheme. This includes, for example,
## government funding of:
##
## -   Health care
##
## -   Child care
##
## -   Elder care and pensions
##
## -   Social housing
##
## *Note: This category excludes education.*
```

Finally, you can also use `mp_view_codebook(...)` to start an interactive website for browsing the category information.

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_200610"]])
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 `ManifestoDocument`s and `ManifestoCorpus`s and returns a vector of the CMP codings attached to the quasi-sentences of the document/corpus in a row:

```
doc <- my_corpus[["42110_200610"]]
head(codes(doc), n = 15)
```

```
##  [1] NA    NA    "501" "606" "501" "501" "501" "416" "416" "412" "503" "411"
## [13] "501" "416" NA
```

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

```
##  [1] "305" "305" "305" NA    NA    NA    "601" "416" "416" "107" "107" "107"
## [13] "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))
```

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

More detailed information on the CMP coding scheme can be found in Accessing the category scheme and category descriptions or in the online documentation of the Manifesto Project coding handbooks at https://manifesto-project.wzb.eu/information/documents/handbooks.

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_200809"]]))
```

```
## [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_200809"]]))
```

```
## [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: 32)>>
## Non-/sparse entries: 79/17
## Sparsity           : 18%
## Maximal term length: 8
## Weighting          : term frequency (tf)
## Sample             :
##           Docs
## Terms      42110_200211 42110_201309 42110_201909 42320_200211 42320_201710
##   familie             2            8           13            2           10
##   menschen           65          144          142           78          126
##   wahl                2            9            2            2            5
```

```
##           Docs
## Terms      42320_201909 42520_200610 42520_201309 42520_201710 42951_201309
##   familie            15           20           20           19           11
##   menschen          147           49           72          180          102
##   wahl                3            0            2            3           12
```

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

```
## $stadt
## numeric(0)
```

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 Interessen."
## [2] "In einer Demokratie sollen all jene wählen dürfen, die von den politischen
## Entscheidungen betroffen sind."
## [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] "Darüber hinaus kamen, über das gesamte Erwerbsleben gerechnet, nur rund ein
## Drittel der Arbeitnehmer in den Genuss einer Abfertigung."
## [2] "Die damalige sozialistisch dominierte Koalition hat Arbeitsplätze
## und Schutz für sozial Bedürftige versprochen, aber tatsächlich steigende
## Arbeitslosenzahlen hingenommen, soziale Notlagen zunehmend verschärft und mehr
## Menschen in Armut gedrängt."
## [3] "Verbesserung der Rechtsdurchsetzung in arbeits- und sozialgerichtlichen
## Verfahren."
## [4] "Ausbau des Service- und Dienstleistungscharakters der Arbeitsinspektorate."
## [5] "hat sich die Einkommensschere zwischen Männern und Frauen immer mehr
## geöffnet."
## [6] "Ein besonderes Problem bestand darin, dass die Asylverfahren sich oft
## mehrere Jahre erstreckten."
```

```
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: 2024-1

## Your query resulted in 125 requested document items containing corpus metadata. Of these
    items 81 could not be retrieved (reasons: 1 having no real documents coded (see progtype
    3 and 99), 24 having no machine-readable texts, 56 having no machine-readable annotations
    ).
```

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

```
## [1] "In the important area of defense alone, our defense white paper has made
## the greatest ever additional provision for the future defense needs of Australia
## of any government in more than a quarter of a century."
## [2] "Over the next ten years we will invest an additional $32 billion in the
## defense of Australia."
## [3] "And how proud I am to say to you that when we came into government in March
## of 1996 and we found not withstanding what Mr."
## [4] "Beazley had told us during the election campaign that our budget was $10."
## [5] "5 billion in deficit, that 'wed accumulated as a nation $96 billion of
## federal government debt, the one restriction I put on Peter Costello and John
## Fahey in getting the budget in shape was you will not cut any money out of
## defense."
## [6] "And not only 'didnt we cut any money out of defense we in fact increased
## defense expenditure, and just as well because in that five and a half year
## period 'weve had the demands of East Timor, of Bougainville, and now the
## commitment to the war against terrorism which is as much our war and our fight
## and our struggle as it is for the people of the United States."
```

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

```
## [1] "104" "104" "104" "104" "104" "104"
```

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_200610
##   party                      : 42110
##   date                       : 200610
##   language                   : german
##   source                     : MARPOR
##   has_eu_code                : FALSE
##   is_primary_doc             : TRUE
##   may_contradict_core_dataset: FALSE
##   md5sum_text                : b90378f0c6fca51b464bbe8cd2c96990
##   url_original               : /down/originals/42110_2006.pdf
##   md5sum_original            : 8fd5726c6363864c3ace6e2d497d647e
##   annotations                : TRUE
##   handbook                   : 3
##   is_copy_of                 : NA
##   title                      : Zeit für Grün. Das Grüne Programm
##   translation_en             : TRUE
##   id                         : 42110_200610
```

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_200610
##   party                      : 42110
##   date                       : 200610
##   language                   : german
##   source                     : MARPOR
##   has_eu_code                : FALSE
##   is_primary_doc             : TRUE
##   may_contradict_core_dataset: FALSE
##   md5sum_text                : b90378f0c6fca51b464bbe8cd2c96990
##   url_original               : /down/originals/42110_2006.pdf
##   md5sum_original            : 8fd5726c6363864c3ace6e2d497d647e
##   annotations                : TRUE
```

```
##   handbook                  : 3
##   is_copy_of                : NA
##   title                     : Zeit für Grün. Das Grüne Programm
##   translation_en            : TRUE
##   id                        : 42110_200610
##   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")
head(metas)
```

```
## # A tibble: 6 x 16
##   party   date language source has_eu_code is_primary_doc may_contradict_core_~1
##   <dbl>  <dbl> <chr>    <chr>  <lgl>       <lgl>          <lgl>
## 1 33220 197706 <NA>     <NA>   FALSE       NA             NA
## 2 33320 197706 spanish  CEMP   FALSE       TRUE           FALSE
## 3 33430 197706 spanish  CEMP   FALSE       TRUE           FALSE
## 4 33610 197706 <NA>     <NA>   FALSE       NA             NA
## 5 33901 197706 <NA>     <NA>   FALSE       NA             NA
## 6 33902 197706 <NA>     <NA>   FALSE       NA             NA
## # i abbreviated name: 1: may_contradict_core_dataset
## # i 9 more variables: manifesto_id <chr>, md5sum_text <chr>,
## #   url_original <chr>, md5sum_original <chr>, annotations <lgl>,
## #   handbook <chr>, is_copy_of <chr>, title <chr>, translation_en <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.
- … `has_eu_code` is TRUE for documents 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.
- … `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 whenever there are CMP codings available for the document.
- … `handbook` an integer that indicates the version of the coding instructions that was used for the coding (e.g. 4 or 5) (since 2016-6).
- … `title` the title of the manifesto (in original language) (since 2017-1).
- … `translation_en` is TRUE whenever an english translation is available for the document (since 2024-1).

The other metadata entries have primarily technical functions for communication between the `manifestoR` package and the online database (for more information have a look at the online documentation of the Manifesto Corpus).

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: 2024-1
## Connecting to Manifesto Project DB API... corpus version: 2024-1
```

```
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] "2024-1"
```

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")
```

Note that `mp_use_corpus_version` instantly updates already locally cached data to the desired new corpus version and that such updating is not yet implemented for translated manifestos.

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] FALSE
##
## $versionid
## [1] "2024-1"
```

```
mp_update_cache()
```

```
## [1] "2024-1"
```

```
mp_check_for_corpus_update()
```

```
## $update_available
## [1] FALSE
##
## $versionid
## [1] "2024-1"
```

For **full reproducibility** of your final scripts you currently have to take into account that using the approach of directly specifying a logical expression for `mp_corpus*/mp_metadata` (e.g. `mp_corpus(party == 41320 & date >= 201709)`) makes use of the most recent version of the dataset (via `mp_maindataset()`) and thus to preserve the version of the dataset used for querying corpus documents and metadata you should query and filter the dataset directly and then provide the result to the `mp_corpus/mp_metadata` function:

```
mpds <- mp_maindataset(version = "MPDS2020a")
```

```
## Connecting to Manifesto Project DB API... corpus version: 2024-1
```

```
wanted <- mpds %>% filter(party == 41320 & date >= 201709)
wanted
```

```
## # A tibble: 1 x 174
##   country countryname oecdmember eumember edate        date party partyname
##     <dbl> <chr>            <dbl>    <dbl> <date>      <dbl> <dbl> <chr>
## 1      41 Germany             10       10 2017-09-24 201709 41320 Social Democr~
## # i 166 more variables: partyabbrev <chr>, parfam <dbl>, coderid <dbl>,
## #   manual <dbl>, coderyear <dbl>, testresult <dbl>, testeditsim <dbl>,
## #   pervote <dbl>, voteest <dbl>, presvote <dbl>, absseat <dbl>,
## #   totseats <dbl>, progtype <dbl>, datasetorigin <dbl>, corpusversion <chr>,
## #   total <dbl>, peruncod <dbl>, per101 <dbl>, per102 <dbl>, per103 <dbl>,
## #   per104 <dbl>, per105 <dbl>, per106 <dbl>, per107 <dbl>, per108 <dbl>,
## #   per109 <dbl>, per110 <dbl>, per201 <dbl>, per202 <dbl>, per203 <dbl>, ...
```

```
mp_corpus(wanted)
```

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

```
# ... can alternatively also be passed directly to the mp_corpus function
# mp_corpus(mpds %>% filter(party == 41320 & date >= 201709))
```

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_tibble(as.data.frame(doc))
head(doc_df)
```

```
## # A tibble: 6 x 5
##   text                                          cmp_code eu_code my_code   pos
##   <chr>                                         <chr>    <chr>   <chr>   <int>
## 1 1 Lebensqualität                              <NA>     <NA>    A           1
## 2 1.1 Grüne Energiewende                        <NA>     <NA>    B           2
## 3 Lebensqualität bedeutet in einer unversehrten ~ 501      <NA>    A           3
## 4 Die Verantwortung dafür liegt bei uns: Wir all~ 606      <NA>    B           4
## 5 Ein Umdenken in der Energiepolitik ist eine we~ 501      <NA>    A           5
## 6 Wir Grüne stehen für eine Energiewende hin zu ~ 501      <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"                "handbook"
## [19] "is_copy_of"                 "title"
## [21] "translation_en"             "id"
## [23] "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: 2024-1
```

```
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: 2024-1
## Connecting to Manifesto Project DB API... corpus version: 2024-1
```

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

```
##    party   date  logit_rile
## 1  93223 199611 -0.18965737
## 2  93322 199611 -0.25231044
## 3  93411 199611 -0.72593700
## 4  93711 199611  0.74721440
## 5  93712 199611  1.18958407
## 6  93951 199611  0.26700718
## 7  93221 200011  0.35216653
## 8  93223 200011  0.59703282
## 9  93430 200011 -1.20640886
## 10 93712 200011  0.71465339
## 11 93951 200011 -0.34998596
## 12 93001 200411 -1.19625076
## 13 93041 200411  0.26304477
## 14 93712 200411  0.10821358
## 15 93951 200411 -0.08659253
## 16 93002 200811 -0.65787904
## 17 93430 200811  0.64662716
## 18 93530 200811  0.25276581
## 19 93951 200811 -0.25629576
## 20 93031 201212  0.71766680
## 21 93061 201212 -0.48130318
## 22 93951 201212 -0.74497225
## 23 93981 201212 -0.78458139
## 24 93223 201612 -0.62659751
## 25 93420 201612  0.72593700
## 26 93430 201612  0.07145896
## 27 93440 201612 -1.21739582
## 28 93540 201612 -0.03587829
## 29 93951 201612 -0.53630471
```

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  93223 199611    2.1660650
## 2  93322 199611   -1.4571949
## 3  93411 199611    5.4545455
## 4  93711 199611   10.9243697
## 5  93712 199611   -1.6666667
```

```
## 6  93951 199611   -2.0100503
## 7  93221 200011   -1.2835473
## 8  93223 200011   -0.2988048
## 9  93430 200011   -2.0512821
## 10 93712 200011    1.2195122
## 11 93951 200011   -0.6134969
## 12 93001 200411    7.6923077
## 13 93041 200411   -1.6181230
## 14 93712 200411   10.9090909
## 15 93951 200411    1.3081395
## 16 93002 200811    1.9718310
## 17 93430 200811    1.5463918
## 18 93530 200811    0.4494382
## 19 93951 200811    0.0000000
## 20 93031 201212    1.5584416
## 21 93061 201212    3.2967033
## 22 93951 201212    2.3631841
## 23 93981 201212    2.9629630
## 24 93223 201612   14.6696529
## 25 93420 201612   23.3333333
## 26 93430 201612    7.3529412
## 27 93440 201612    1.3282732
## 28 93540 201612   -2.2613065
## 29 93951 201612    4.1666667
```

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_1`). For example, the ratio equivalent to the simple function above can be implemented as:

```
custom_scale <- function(data) {
    scale_ratio_1(data, pos = c("per402"), neg = c("per401"))
}
mp_scale(corpus, scalingfun = custom_scale)
```

```
##    party   date custom_scale
## 1  93223 199611   0.50000000
## 2  93322 199611  -0.28571429
## 3  93411 199611   1.00000000
## 4  93711 199611   0.86666667
## 5  93712 199611  -1.00000000
## 6  93951 199611  -0.72727273
## 7  93221 200011  -0.23404255
## 8  93223 200011  -0.05084746
## 9  93430 200011  -0.40000000
## 10 93712 200011   0.33333333
## 11 93951 200011  -0.42857143
## 12 93001 200411   1.00000000
## 13 93041 200411  -0.20000000
## 14 93712 200411   1.00000000
## 15 93951 200411   0.60000000
## 16 93002 200811   0.63636364
## 17 93430 200811   0.17647059
## 18 93530 200811   0.10000000
```

```
## 19 93951 200811         NaN
## 20 93031 201212  0.30000000
## 21 93061 201212  0.60000000
## 22 93951 201212  0.70370370
## 23 93981 201212  0.40000000
## 24 93223 201612  0.95620438
## 25 93420 201612  1.00000000
## 26 93430 201612  0.51724138
## 27 93440 201612  0.77777778
## 28 93540 201612 -0.52941176
## 29 93951 201612  1.00000000
```

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)
```

```
##             rile        sd
## 1   1.592900e+01  7.864324
## 2  -1.146300e+01  4.046633
## 3   1.027400e+01  4.700404
## 4   1.111100e+01  8.037227
## 5   7.176000e+00  4.092869
## 6   1.792300e+01  5.773106
## 7   5.405000e+00  4.298796
## 8   5.882000e+00  7.408528
## 9  -7.298000e+00  4.164510
## 10 -1.354000e+01  6.843101
## 11  6.012000e+00  5.719455
## 12  4.232200e+01  4.050045
## 13  1.431200e+01  2.774931
## 14  2.220446e-16  5.600339
## 15 -9.090000e+00 17.091461
## 16 -9.350000e-01  2.604984
## 17 -2.187000e+00  2.804622
## 18 -9.180000e-01  2.872520
## 19  5.596200e+01  5.344882
## 20 -1.304900e+01  2.400320
## 21  8.059000e+00  3.749017
## 22  9.919000e+00  5.857223
## 23 -4.166000e+00  4.550612
## 24  7.760000e-01  4.108355
## 25  5.710000e-01  5.034333
## 26 -2.187000e+00  2.844663
## 27 -9.180000e-01  2.726073
## 28  5.596200e+01  5.209681
```

```
## 29  2.718500e+01  7.248418
## 30  2.428600e+01  8.471866
## 31  9.919000e+00  6.067329
## 32 -4.166000e+00  4.665132
## 33  2.247200e+01  7.494012
## 34  5.710000e-01  5.079541
## 35 -2.187000e+00  2.776559
## 36 -9.180000e-01  2.775155
## 37 -4.166000e+00  4.523966
## 38  2.247200e+01  7.410709
```

Note that the argument `fun` can be any scaling function and the returned `data.frame` contains 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))
```

```
##    custom_scale        var      q0.025     q0.975
## 1         0.885  2.1677031  -1.7699823  3.5399646
## 2        -0.395  0.1527898  -1.1857945  0.0000000
## 3         0.685  1.4656942  -1.3699452  3.4248630
## 4        -1.111  1.3353776  -3.3332333  0.0000000
## 5         0.000  2.5718016  -2.8706699  2.8706699
## 6         0.000  1.8648284  -2.8299906  2.8299906
## 7        -1.081  1.1126497  -3.2435351  1.0811784
## 8        -5.882 15.7314639 -14.7057353  0.0000000
## 9         0.000  0.5563293  -1.4598978  1.4598978
## 10        3.125  3.2515193   0.0000000  7.2918854
## 11       -6.011  4.2766803 -10.3822022 -2.1857268
## 12       -0.341  0.8818332  -2.3890546  1.3651741
## 13        1.506  1.0625935  -0.5649661  3.5781186
## 14       -1.191  2.1341987  -4.1665417  1.7856607
## 15        0.000  0.0000000   0.0000000  0.0000000
## 16       -3.371  0.8319191  -5.2434981 -1.6854101
## 17        1.532  0.8399342  -0.2188184  3.2877462
## 18        3.131  1.4120138   0.7366409  5.3406464
## 19        2.752  6.1720520  -1.8348257  8.2567156
## 20        1.518  0.8430098  -0.3034810  3.3382914
## 21       -4.030  1.7273612  -6.5934725 -1.4652161
## 22       -3.306  5.3752492  -8.2639669  0.8263967
## 23        1.389  5.4964568  -2.7778333  6.9445833
## 24       -0.775  0.6279406  -2.3255814  0.0000000
## 25        3.428  3.1702347   0.0000000  6.8568686
## 26        1.532  0.8094542  -0.2188184  3.2822757
## 27        3.131  1.4646290   0.5524807  5.5248066
## 28        2.752  5.9938395  -1.8348257  8.2567156
## 29        6.796 11.6736880   0.0000000 13.5922330
## 30       -5.714  7.6106658 -11.4290286 -1.4286286
## 31       -3.306  4.8656928  -8.2639669  0.8263967
```

```
## 32       1.389  5.9705436  -2.7778333  6.9445833
## 33      -2.247  2.5691049  -5.6181461  0.0000000
## 34       3.428  3.1186111   0.0000000  6.8568686
## 35       1.532  0.7606666  -0.2188184  3.2822757
## 36       3.131  1.4948546   0.9161971  5.5248066
## 37       1.389  6.0630260  -2.7778333  6.9445833
## 38      -2.247  2.4401258  -5.6181461  0.0000000
```

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: https://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. https://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. https://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)