Projects for Text Version Comparison and Analytics in R

Contents

1 Overview

Status

R code: 1614 *C++ code:* 112 *test code:* 1010

Version

0.1.12

Description

Provides data structures and methods for manual as wells as automated R based text comparison and text as well as change coding.

License

MIT + file LICENSE Peter Meissner retep.meissner@gmail.com [aut, cre] Ulrich Sieberer ulrich.sieberer@uni-bamberg.de [cph] University of Konstanz willkommen@uni-konstanz.de [cph]

Citation

Meißner P (2016). *diffrprojects: Using diffr for more than two texts*. R package version 0.1.12, <URL: https://github.com/petermeissner/diffrprojects>.

Sieberer U, Meißner P, Keh J and Müller W (2016). "Mapping and Explaining Parliamentary Rule Changes in Europe: A Research Program." *Legislative Studies Quarterly*, *41*(1), pp. 61-88. ISSN 1939-9162, doi: 10.1111/lsq.12106 (URL: http://doi.org/10.1111/lsq.12106), <URL: http://dx.doi.org/10.1111/lsq.12106>.

BibTex for citing

```
toBibtex(citation("diffrprojects"))
```

Installation

stable CRAN version

```
install.packages("diffrprojects")
library(rtext)
```

(stable) development version

```
standard_repos <- options("repos")$repos
install.packages( "diffrprojects", repos = c(standard_repos, "https://petermeissner.github.io/drat/"))
library(rtext)
```

2 Usage

2.1 Fast Introduction for the Impatient

For those in a hurry here is a very brief

```
# loading package
library(diffrprojects)


# the first chapter of Robinson Crusoe from three different sources
rcs <- rtext:::testfile(pattern="rc.*ch1.txt", full.names = TRUE)


# creating a new project
dp <- diffrproject$new()


# setting options
dp$options$verbose <- FALSE


# adding texts to the corpus
```

```
dp$text_add(text_file = rcs)
dp$text_data(1) %>% head(11)
```

```
##     i char         name
## 1   1    T rc_1_ch1.txt
## 2   2    h rc_1_ch1.txt
## 3   3    e rc_1_ch1.txt
## 4   4      rc_1_ch1.txt
## 5   5    P rc_1_ch1.txt
## 6   6    r rc_1_ch1.txt
## 7   7    o rc_1_ch1.txt
## 8   8    j rc_1_ch1.txt
## 9   9    e rc_1_ch1.txt
## 10 10    c rc_1_ch1.txt
## 11 11    t rc_1_ch1.txt
```

```
# linking the files (which file should be compared to which)
dp$text_link()
dp$link %>% as.data.frame()
```

```
##           from           to                       link
## 1 rc_1_ch1.txt rc_2_ch1.txt rc_1_ch1.txt~rc_2_ch1.txt
## 2 rc_2_ch1.txt rc_3_ch1.txt rc_2_ch1.txt~rc_3_ch1.txt
```

```
# calculating text alignments
dp$text_align(tokenizer=text_tokenize_words)
dp$alignment[[1]] %>% head(30)
```

```
##    alignment_i token_i_1 token_i_2 distance      type from_1 to_1 from_2  to_2
## 1            1         1      1932        0 no-change      1    3  10326 10328
## 2            2         2        NA        7  deletion      5   11     NA    NA
## 3            3         3        NA        9  deletion     13   21     NA    NA
## 4            4         4        NA        5  deletion     23   27     NA    NA
## 5            5         5      1932        0 no-change     30   32  10326 10328
## 6            6         6        NA        4  deletion     34   37     NA    NA
## 7            7         7        87        0 no-change     39   41    513   515
## 8            8         8        NA       10  deletion     43   52     NA    NA
## 9            9         9        57        0 no-change     54   55    355   356
## 10          10        10         3        0 no-change     57   64     15    22
## 11          11        11         4        0 no-change     66   71     24    29
## 12          12        12        85        0 no-change     74   75    497   498
## 13          13        13         1        0 no-change     77   82      1     6
## 14          14        14         2        0 no-change     84   88      8    12
## 15          15        15      1520        0 no-change     92   95   8187  8190
## 16          16        16        NA        5  deletion     97  101     NA    NA
## 17          17        17      3667        0 no-change    103  104  19215 19216
## 18          18        18       263        0 no-change    106  108   1495  1497
## 19          19        19        51        0 no-change    110  112    328   330
## 20          20        20        NA        3  deletion    114  116     NA    NA
## 21          21        21        57        0 no-change    118  119    355   356
## 22          22        22        NA        6  deletion    121  126     NA    NA
## 23          23        23        NA        8  deletion    128  135     NA    NA
## 24          24        24        50        0 no-change    137  138    325   326
## 25          25        25        51        0 no-change    140  142    328   330
## 26          26        26        NA        6  deletion    144  149     NA    NA
```

```
## 27          27        27        NA        6  deletion    151  156     NA    NA
## 28          28        28        87        0 no-change    158  160    513   515
## 29          29        29       513        0 no-change    162  165   2853  2856
## 30          30        30       306        0 no-change    167  171   1724  1728
```

2.2 Creating a Diffr Project

To create a diffrproject we use the diffrproject creator object - its simply an object with an function that knows how to create a project.

Creating a project looks like this:

```
library(diffrprojects)
dp <- diffrproject$new()
```

Et violà - we created a first, for now empty, project that we will use throughout the tutorial.

2.3 Some Help Please

To get a better idea about what this thing called *diffrproject* really is you can consult its help page which gives a broad overview over its capabilities:

```
?diffrproject
```

Another way is to call the ls() method. This will present us with a data frame listing all fields where data is stored and all the methods (aka object specific functions) of our diffrprojects instance. Those methods and fields located in *private* are not for the user to mess around with while non-private (*self* aka public) data fields can be read by the user and public methods can be triggered by the user to manipulate the data or retrieve data in a specific format.

```
dp$ls()
```

```
##                            name   where                     class
## 1                  execute_load private                  function
## 2                          hash private                  function
## 3                        hashed private                  function
## 5                  prepare_save private                  function
## 4                        hashes private                      list
## 9                alignment_data    self alignment_data_list, list
## 6                     alignment    self      alignment_list, list
## 21                         link    self      alignment_list, list
## 7                 alignment_add    self                  function
## 8                alignment_code    self                  function
## 10           alignment_data_full    self                  function
## 11            alignment_data_set    self                  function
## 12              alignment_delete    self                  function
## 13                         clone    self                  function
## 14                         debug    self                  function
## 15                    export_csv    self                  function
## 16                export_sqlite    self                  function
## 17                          get    self                  function
## 18                   import_csv    self                  function
## 19                import_sqlite    self                  function
## 20                   initialize    self                  function
## 22                         load    self                  function
```

```
## 23                              ls    self                 function
## 24                         message    self                 function
## 27                            save    self                 function
## 29                        text_add    self                 function
## 30                       text_align    self                 function
## 31                        text_code    self                 function
## 32       text_code_alignment_token    self                 function
## 33 text_code_alignment_token_regex    self                 function
## 34                  text_code_regex    self                 function
## 35                        text_data    self                 function
## 36                text_data_inherit    self                 function
## 37                      text_delete    self                 function
## 38                        text_link    self                 function
## 39                   text_meta_data    self                 function
## 40         tokenize_text_data_lines    self                 function
## 41         tokenize_text_data_regex    self                 function
## 42         tokenize_text_data_words    self                 function
## 43                         warning    self                 function
## 25                            meta    self                     list
## 26                         options    self                     list
## 28                            text    self                     list
```

The base R class() function furthermore reveals from which classes the diffrproject class inherits:

```
class(dp)
```

```
## [1] "diffrproject"      "dp_inherit"        "dp_align"          "dp_export"
## [5] "rtext_loadsave"    "dp_base"           "R6_rtext_extended" "R6"
```

2.4 Adding Texts to Projects

Our diffrproject (`dp`) has one method called `text_add()` that allows to add texts to the project. Basically the method can be used in three different flavors: adding character vectors, adding texts stored on disk, or by adding rtext objects (see rtext package: https://CRAN.R-project.org/package=rtext; rtext objects are the way individual texts are represented within diffrprojects). For each of these use cases there is one option: `text`, `text_file`, `rtext`; respectively.

Below are shown examples using each of these methods:

adding text files

```
test_file1 <- stringb:::test_file("rc_1_ch1.txt")
test_file2 <- stringb:::test_file("rc_2_ch1.txt")
dp$text_add(text_file = c(test_file1, test_file2) )
```

adding rtext objects

```
test_file <- stringb:::test_file("rc_1_ch1.txt")
rt <- rtext$new( text_file = test_file)
dp$text_add(rtext = rt)
```

adding character vectors

```
test_file1 <- stringb:::test_file("rc_1_ch1.txt")
test_file2 <- stringb:::test_file("rc_2_ch1.txt")
cv <- ""
cv[1] <- text_read(test_file1, NULL)
```

```
cv[2] <- text_read(test_file2, NULL)
dp$text_add(text = cv)
```

In the last case make sure to put each text in one separate line. Functions like readLines() or text_read() read in texts such that each line corresponds to one element in a character vector. With e.g. text_read()'s tokenize parameter to NULL the text will be read in as one long string.

2.5 Piping methods

Now is a good time to mention a feature of diffrprojects that comes in handy: All functions that do not explicitly extract data (those usually have some 'get' as part of their name) do return return the object itself so that one can pipe together a series of method calls.

Consider the following example where we initiate a new diffrprojects instance and add two texts in just one pipe:

```
dp <-
  diffrproject$
  new()$
  text_add(text_version_1, name = "version1")$
  text_add(text_version_2, name = "version2")

length(dp$text)
```

```
## [1] 2
```

2.6 Getting Infos About Texts

If we want to get some general overview about the texts gathered in our project we can use the text_meta_data() method to do so. The method has no parameters and return a data.frame with several variables informing us about its source, length, encoding used for storage, and its name.

```
dp$text_meta_data()
```

```
##   text_file character encoding sourcetype     name
## 1      <NA>       479    UTF-8       text version1
## 2      <NA>       539    UTF-8       text version2
```

2.7 Getting And Setting Infos About the Project

Similar to the text_meta_data() method we can access the projects meta data via data fields meta and options. But contrary to the text_meta_data() method that gathers data from all the texts within the project and does not allow for manipulation of the data, the data fields allow reading and writing.

First let us have a look and thereafter turn of the message notification service:

getting data fields

```
dp$options
```

```
## $verbose
## [1] TRUE
##
## $warning
## [1] TRUE
```

```
##
## $ask
## [1] TRUE
```

setting data fields

```
dp$options$verbose <- FALSE
```

(note, ask is deprecated and only remains for compatibility reasons but has no function anymore)

Now its time to have a look at the projects meta data. It tells us when the project was created, which path to use for SQLite exports, which path to use for saving data as in RData format and what is the projects id. The id is a hash of a time stamp as well as session information which should ensure uniqueness across space and time.

All these values can manipulated by the user to her liking.

```
dp$meta
```

```
## $ts_created
## [1] "2016-11-04 22:01:05 UTC"
##
## $db_path
## [1] "./diffrproject.db"
##
## $file_path
## [1] ""
##
## $project_id
## [1] "d6d933c5ecd7cb63b300d172d1a9dff4"
```

```
dp$meta$file_path = "./diffrproject.RData"
```

2.8 Deleting Texts

Of cause we can not only add texts but delete them from the project as well. For this purpose there is the text_delete() method.

Let's just add two texts and delete one by providing its index number and the second by providing its name to the text_delete() method.

```
dp$text_add(text = "nonesense", "n1")
dp$text_add(text = "nonesense", "n2")

dp$text_delete(3)
dp$text_delete("n2")

length(dp$text)
```

```
## [1] 2
```

```
names(dp$text)
```

```
## [1] "version1" "version2"
```

2.9 Coding Texts

```
dp$text
```

```
## $version1
## <rtext>
##   Inherits from: <rtext_tokenize>
##   Public:
##     char_add: function (what = NULL, after = NULL)
##     char_data_get: function (from = 1, to = Inf, x = NULL, full = FALSE)
##     char_data_set: function (x = NULL, i = NULL, val = NA, hl = 0)
##     char_data_set_regex: function (x = NULL, pattern = NULL, val = NA, hl = 0, ...)
##     char_delete: function (n = NULL, from = NULL, to = NULL)
##     char_get: function (length = Inf, from = NULL, to = NULL, raw = FALSE)
##     char_length: function ()
##     char_replace: function (from = NULL, to = NULL, by = NULL)
##     clone: function (deep = FALSE)
##     debug: function (pos = 1)
##     encoding: UTF-8
##     export_csv: function (folder_name = "")
##     export_sqlite: function (db_name = "")
##     get: function (name = NULL)
##     hash_get: function (name = "")
##     id: 6eb7737ee041de8c
##     import_csv: function (folder_name = "")
##     import_sqlite: function (db_name = "")
##     info: function ()
##     initialize: function (text = NULL, text_file = NULL, encoding = "UTF-8",
##     load: function (file = NULL)
##     ls: function (what = c("self", "private"), class = NULL)
##     message: function (x, ...)
##     options: list
##     save: function (file = NULL, id = NULL)
##     save_file: NA
##     sourcetype: text
##     text_file: NA
##     text_get: function (length = Inf, from = NULL, to = NULL, split = NULL)
##     text_get_lines: function (length = Inf, from = NULL, to = NULL)
##     text_show: function (length = 500, from = NULL, to = NULL, coll = FALSE,
##     tokenize_data_lines: function (split = "\n", ignore.case = FALSE, fixed = FALSE, perl = FALSE,
##     tokenize_data_regex: function (split = NULL, ignore.case = FALSE, fixed = FALSE, perl = FALSE,
##     tokenize_data_sequences: function (token, join = c("full", "left", "right", ""), aggregate_functi
##     tokenize_data_words: function (split = "\\W+", ignore.case = FALSE, fixed = FALSE,
##     warning: function (x, ...)
##   Private:
##     char: T h i s   p a r t   o f   t h e
##      d o c u m e n t   h a  ...
##     char_data: list
##     execute_load: function (tmp)
##     hash: function (name = NULL)
##     hashed: function (name = NULL)
##     hashes: list
##     prepare_save: function (id = NULL)
##     text: function ()
```

```
##
## $version2
## <rtext>
##   Inherits from: <rtext_tokenize>
##   Public:
##     char_add: function (what = NULL, after = NULL)
##     char_data_get: function (from = 1, to = Inf, x = NULL, full = FALSE)
##     char_data_set: function (x = NULL, i = NULL, val = NA, hl = 0)
##     char_data_set_regex: function (x = NULL, pattern = NULL, val = NA, hl = 0, ...)
##     char_delete: function (n = NULL, from = NULL, to = NULL)
##     char_get: function (length = Inf, from = NULL, to = NULL, raw = FALSE)
##     char_length: function ()
##     char_replace: function (from = NULL, to = NULL, by = NULL)
##     clone: function (deep = FALSE)
##     debug: function (pos = 1)
##     encoding: UTF-8
##     export_csv: function (folder_name = "")
##     export_sqlite: function (db_name = "")
##     get: function (name = NULL)
##     hash_get: function (name = "")
##     id: 81edb193c9f94610
##     import_csv: function (folder_name = "")
##     import_sqlite: function (db_name = "")
##     info: function ()
##     initialize: function (text = NULL, text_file = NULL, encoding = "UTF-8",
##     load: function (file = NULL)
##     ls: function (what = c("self", "private"), class = NULL)
##     message: function (x, ...)
##     options: list
##     save: function (file = NULL, id = NULL)
##     save_file: NA
##     sourcetype: text
##     text_file: NA
##     text_get: function (length = Inf, from = NULL, to = NULL, split = NULL)
##     text_get_lines: function (length = Inf, from = NULL, to = NULL)
##     text_show: function (length = 500, from = NULL, to = NULL, coll = FALSE,
##     tokenize_data_lines: function (split = "\n", ignore.case = FALSE, fixed = FALSE, perl = FALSE,
##     tokenize_data_regex: function (split = NULL, ignore.case = FALSE, fixed = FALSE, perl = FALSE,
##     tokenize_data_sequences: function (token, join = c("full", "left", "right", ""), aggregate_functi
##     tokenize_data_words: function (split = "\\W+", ignore.case = FALSE, fixed = FALSE,
##     warning: function (x, ...)
##   Private:
##     char: T h i s   i s   a n   i m p o r t a n t
##      n o t i c e !  ...
##     char_data: list
##     execute_load: function (tmp)
##     hash: function (name = NULL)
##     hashed: function (name = NULL)
##     hashes: list
##     prepare_save: function (id = NULL)
##     text: function ()
```

2.10 Getting Text Codings

2.11 Aggregating Text Codings

2.12 Measuring Change and Aligning Texts

2.13 Text Coding Inheritence

2.14 __

2.15 __

2.16 __

2.17 __

2.18 __

2.19 __

2.20 __

2.21 __

2.22 __

2.23 __

2.24 __

2.25 __

2.26 __

3 Technicalities

3.1 Naming Conventions and General Structure of Methods and Data

The methods and data fields of diffrprojects can be categorized into five realms - *cursive*: methods; (paratheses): private; rest: data:

- **text**: everything related to individual texts starts with text
  - text, *text_add*, *text_delete*, *text_align*, *text_code*, *text_code_alignment_token*, *text_code_alignment_token_regex*, *text_code_regex*,
  - *text_data*, *text_data_inherit*, *tokenize_text_data_lines*, *tokenize_text_data_regex*, *tokenize_text_data_words*
  - text_meta_data
- **alignment**: everything that concerns the relation between two texts
  - alignment, *alignment_add*, *alignment_code*, *alignment_delete*, *alignemtn_data_full*, *alignment_data_set*

  - text_link, *link*
- **misc**:
  - meta, options, *load*, *save*, *export_sqlite*, *import_sqlite*, (*execute_load*), (*prepare_save*)
- **inherited from R6_rtext_extended**:
  - options, *message*, *warning*, (*hash*), (hashed), (hashes)
- **inherited from R6**:
  - *clone*, *initialize*

3.2 Data formats

3.2.1 meta

Meta is a list with only a few items providing/storing general information for the whole project - i.e. time stamp the project was created, path to store data, path to export data, an project id.

3.2.2 text

Text is a list of rtext instances. Each rtext instances stores text's actual text as data gathered on the text.

The text_data method will return a data.frame containing all text data, while tokenize_text_data_xxx mehtods will aggregate text data to specific token levels: words, lines or user defined patterns.

3.2.3 link

Link is a list of links between texts. Link defines for which text combination alignments should be calcualted. Each list item hold a from and to field which stores the names of texts to be aligned. The method to create links is text_link, it also allows to delete specific links.

Link data can be transformed to one big data.frame via: as.data.frame function.

3.2.4 alignment

Alignment is a list of data.frames. Each alignment list item stores the which part (character span) of one text is connected to which part (character span) of another text.

The list of alignments can be transformed to one big data.frame via: as.data.frame function.

3.2.5 alignment_data

The list of alignment_data can be transformed to one big data.frame via: as.data.frame function.

[[[???!!!]]]