# Package ‘swissparl’

October 14, 2022

**Type** Package  
**Title** Interface to the Webservices of the Swiss Parliament  
**Version** 0.2.2  
**Description**  
Retrieves the most important data on parliamentary activities of the Swiss Federal Assembly via an open, machine-readable interface (see <https://ws.parlament.ch/odata.svc/>).

**URL** https://www.parlament.ch/en/services/open-data-webservices  
**BugReports** https://github.com/zumbov2/swissparl/issues  
**License** GPL (>= 2)  
**Encoding** UTF-8  
**LazyData** true  
**RoxygenNote** 7.1.2  
**Imports** dplyr, jsonlite, magrittr, purrr, stringr, tibble, tidyr, crayon, htr, ggplot2  
**NeedsCompilation** no  
**Author** David Zumbach [aut, cre], Benjamin Gföhler [ctb]  
**Maintainer** David Zumbach <david.zumbach@gfzb.ch>  
**Repository** CRAN  
**Date/Publication** 2021-11-02 08:10:02 UTC

### R topics documented:

- clean_text  
- get_data  
- get_glimpse  
- get_overview  
- get_tables  
- get_variables  
- ggswissparl  
- seating_plan  
- swissparl
### clean_text

**Description**

clean_text removes HTML code, brackets and their contents as well as line breaks from texts.

**Usage**

```r
clean_text(text, keep_round_brackets = T)
```

**Arguments**

- `text`: a character vector.
- `keep_round_brackets`: if TRUE, round brackets and their contents are not deleted.

**Value**

A character vector of same length as text.

**Examples**

```r
## Not run:
# Get clean version of transcripts
get_glimpse(table = "Transcript", rows = 1000, Language = "DE") %>%
mutate(Text2 = clean_text(Text))
## End(Not run)
```

### get_data

**Description**

get_data retrieves data from the WebServices of the Swiss Parliament.

**Usage**

```r
get_data(
    table,
    package_size = 1000,
    stop = T,
    attempts = 10,
    wtf = 1,
    silent = F,
    ...
)
```
### get_data

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| table         | name of the table to download. For an overview of available tables use `get_tables()`.
| package_size  | number of rows to download at once (maximum = 1000). If a query exceeds `package_size`, it is internally split into multiple subqueries of size `package_size`.
| stop          | if `TRUE`, the query process is interrupted if the query is invalid. It also indicates whether a non-existent table or variable was used in the query. If `FALSE`, nothing is returned.
| attempts      | maximum number of repetitions of a single subquery if it was not successful.
| wtf           | factor for extending the waiting time after unsuccessful queries. If `wtf = 1`, the waiting time corresponds to the number of unsuccessful attempts in seconds. For `attempts = 10` and `wtf = 1`, a query is repeated for a maximum of 45 seconds. The waiting time increases proportionally with `wtf`.
| silent        | if `TRUE`, no progress bar and messages are displayed.
| ...           | optional filter arguments with values. Since all entries are available in several languages, it is recommended to filter the calls by language, e.g. `get_data(table = "Person", Language = "DE")`. For a table-specific preview use `get_glimpse()` or `get_variables()`.

The following things are to consider:

- numbers for identification numbers, for example, must be entered as numeric vectors: e.g. `get_data(table = "Voting", PersonNumber = c(21, 4167), Language = "DE")`.
- dates must be entered as character vectors in yyyy-mm-dd format. `>` and `<` can be used to query periods: e.g. `get_data(table = "Bill", SubmissionDate = c(">2018-12-31", "<2019-02-01"), Language = "DE")`.
- the `~` can be used as a substring search for character variables: e.g. `get_data(table = "Bill", Title = "~CO2", Language = "DE")`.

#### Value

A tibble of different length and variable composition.

#### Examples

```r
## Not run:
# Retrieve data on the members of the Swiss Parliament
get_data(table = "Person", Language = "DE")

# Retrieve voting behavior of selected councillors
get_data(
  table = "Voting",
  PersonNumber = c(21, 4167),
  Language = "DE"
)

# Retrieve businesses submitted during a specified period
get_data(
  table = "Business",
  SubmissionDate = c(">2018-12-31", "<2019-02-01"),
```
get_glimpse

Retrieve the first rows of a table

Description

get_glimpse retrieves the first rows of a table of the Swiss Parliament WebServices and allows a first insight into the data structure.

Usage

glimpse(table, rows = 20, Language = "DE")

Arguments

table: name of the table to glimpse into. For an overview of available tables use `get_tables()`.

rows: number of records to download. Maximum is 1000.

Language: filter rows by language. Possible are DE, FR, IT, RM, and EN.

Value

A tibble of different length and variable composition.

Examples

## Not run:
# Short excerpt of table "Person"
glimpse(table = "Person")

## End(Not run)
get_overview

Description

get_overview retrieves the names of all available tables of the Swiss Parliament WebServices and the variables they contain.

Usage

get_overview(silent = F)

Arguments

silent if TRUE, no progress bar and messages are displayed.

Value

A tibble with the 2 columns table and variable.

Examples

## Not run:
get_overview()
## End(Not run)

get_tables

Description

get_tables retrieves the names of the available tables of the Swiss Parliament WebServices.

Usage

get_tables()

Value

A character vector that contains all the names of the available tables.

Examples

## Not run:
# Get all available tables
get_tables()
## End(Not run)
get_variables  

Retrieve available variables

Description
get_variables retrieves the variable names of a table of the Swiss Parliament WebServices.

Usage
get_variables(table, pb.pos = NULL, pb = NULL)

Arguments
- **table**: name of the table to be searched. For an overview of available tables use get_tables().
- **pb.pos**: value for the progress bar. Not to be specified outside of get_overview().
- **pb**: progress bar. Not to be specified outside of get_overview().

Value
A character vector that contains the names of the variables.

Examples
```r
## Not run:
# Get variables of table "Person"
get_variables(table = "Person")
## End(Not run)
```

ggswissparl  

Plot voting results

Description
ggswissparl plots voting results of the Swiss National Council according to the latest seating order.

Usage
ggswissparl(
  votes,
  seats = NULL,
  highlight,
  result = F,
  result_size = 6,
  point_shape = 16,
)
point_size = 4,
theme = "scoreboard"
)

Arguments
data of votes of the Swiss National Council as can be retrieved with get_data(table = "Voting"). The variables PersonNumber, Decision, and DecisionText must be available from the data.

seats data linking councillors (PersonNumber) to seats (SeatNumber). If is.null, the most current seating order is retrieved via get_data(table = "SeatOrganisationNr").

highlight named list with variable and values to specify highlighting of selected councillors.

result if TRUE, the result is annotated.

result_size font size of result.

point_shape shape of point as defined in ggplot2[geom_point].

point_size size of point.

theme name of predefined plot theme:
• "scoreboard" imitates the scoreboard in the council hall: neon-red (yes-votes), neon-green (no-votes) and white (abstentions) dots on black ground in white frames.
• "sym1" colored symbols on light background in black frames.
• "sym2" colored symbols on light background without frames.
• "poly1" color-filled polygons with black edges.
• "poly2" color-filled polygons with white edges.
• "poly3" color-filled polygons without edges.

Value
A ggplot object. If votes contains multiple ballots, ggplot2[facet_wrap] is used to create facets.

Examples
## Not run:
# Visualization of a vote of the 51st legislature
get_data("Voting", Language = "DE", IdVote = 23458) %>%
ggswissparl()

# Highlighting a parliamentary group
get_data("Voting", Language = "DE", IdVote = 23458) %>%
ggswissparl(highlight = list("ParlGroupNumber" = 2))

## End(Not run)
seating_plan  

**Seating plan of the National Council**

**Description**
A dataset containing the relative locations of the seats in the Swiss National Council to display schematic seating plans. A seat is defined by 4 corner points.

**Usage**
seating_plan

**Format**
A data frame with 800 rows and 5 variables:
- **SeatNumber**  seat identifier.
- order  corner identifier.
- x  position of a corner point on the x-axis.
- y  position of a corner point on the y-axis.
- center_x  position of the seat center on the x-axis.
- center_y  position of the seat center on the y-axis.

**Source**
https://www.parlament.ch/en/organe/national-council/groups-chamber-nc

---

**Description**
The Swiss Parliament Webservices R API

**Details**
See the README on GitHub
Index

* datasets
  seating_plan, 8

  clean_text, 2

get_data, 2
get_glimpse, 3, 4
get_overview, 5, 6
get_tables, 3, 4, 5, 6
get_variables, 3, 6
ggswissparl, 6

seating_plan, 8
swissparl, 8