Package 'swissparl'

March 4, 2026

Type Package

Title Interface to Swiss Parliament Web Services and the 'OpenParlData' API

Version 0.3.0

Description Provides machine-readable access to parliamentary data of the Swiss Federal Assembly via the 'OData' interface (<`https://ws.parlament.ch/odata.svc/`>) and the 'OpenParlData' REST API (<`https://api.openparldata.ch`>), which also offers harmonized data for selected cantonal and municipal parliaments.

URL `https://www.parlament.ch/de/%c3%bcber-das-parlament/fakten-und-zahlen/open-data-web-services`,

`https://api.openparldata.ch/documentation`

BugReports `https://github.com/zumbov2/swissparl/issues`

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports dplyr, jsonlite, magrittr, purrr, stringr, tibble, tidyr, crayon, httr, ggplot2, glue, utils

NeedsCompilation no

Author David Zumbach [aut, cre], Benjamin Gföhler [ctb]

Maintainer David Zumbach <`david.zumbach@gfzb.ch`>

Repository CRAN

Date/Publication 2026-03-04 07:30:02 UTC

Contents

`clean_text` *Clean texts retrieved from WebServices*

Description

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

Usage

```
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

```
## 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 *Retrieve data from WebServices*

Description

`get_data` retrieves data from the WebServices of the Swiss Parliament.

Usage

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

Arguments

`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 `attemps = 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 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

```
## 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"),
    Language = "DE"
    )

# Retrieve businesses on the subject of CO2
get_data(
    table = "Business",
    Title = "~CO2",
    Language = "DE"
    )

## End(Not run)
```

`get_data2` *Retrieve data from the OpenParlData.ch REST API*

Description

`get_data2` retrieves data from the OpenParlData.ch REST API for a given resource.

Usage

```
get_data2(table, max_rows, package_size = 1000, silent = FALSE, ...)
```

Arguments

`table`	name of the OpenParlData resource to download. For an overview of available endpoints use `get_tables2()`.
`max_rows`	maximum number of rows to return. If omitted, all available rows matching the query are downloaded.

`package_size`	number of rows to download per request (mapped to the API parameter `limit`). Default is 1000. If the result set exceeds `package_size`, multiple requests are made using the API's pagination links.
`silent`	if `TRUE`, no progress bar and messages are displayed.
`...`	additional query parameters passed to the OpenParlData endpoint as URL query parameters. Common parameters include: • `search`: search query string. • `search_mode`: one of `"partial"` (default), `"exact"`, `"natural"`, `"boolean"`. • `search_scope`: where to search • `search_language`: language-specific search (`de`, `fr`, `it`, `rm`, `en`). • `sort_by`: field(s) to sort by; prefix with `-` for descending. Resource-specific filters (e.g. `body_key`, `lang`, etc.) can also be supplied. Multiple values can be provided as an R vector and are encoded as a comma-separated query value (e.g. `body_key = c("AI", "AR")`).

Value

A tibble containing up to `max_rows` records. Column composition depends on the selected resource and query parameters.

Examples

```
## Not run:
# Retrieve first 10 persons
get_data2("persons", max_rows = 10)

# Retrieve a specific person by first and last name
get_data2(
  "persons",
  firstname = "Karin",
  lastname = "Keller-Sutter"
)

# Partial search (default mode) in affairs
get_data2("affairs", max_rows = 10, search = "Budget", search_mode = "partial")

# Boolean search with grouping (note: '&' is an operator in boolean mode)
get_data2(
  "affairs",
  max_rows = 10,
  search = "(Klima | Umwelt) & Schweiz",
  search_mode = "boolean"
)

# Combine search with scope/language and sorting
get_data2(
  "affairs",
  max_rows = 10,
  search = "Bundesrat Parlament",
  search_mode = "natural",
  search_language = "de",
  sort_by = "-begin_date"
)


## End(Not run)
```

---

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 its data structure.

Usage

```
get_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"
get_glimpse(table = "Person")

## End(Not run)
```

`get_glimpse2` *Retrieve the first rows of an OpenParlData resource*

Description

`get_glimpse2` retrieves the first rows of a resource provided by the OpenParlData.ch REST API and allows a first insight into its data structure.

Usage

```
get_glimpse2(table, rows = 20)
```

Arguments

`table`	name of the OpenParlData resource to glimpse into. For an overview of available endpoints use `get_tables2()`.
`rows`	number of records to download. Maximum is 1000.

Value

A tibble containing up to `rows` records. Column composition depends on the selected OpenParlData resource.

Examples

```
## Not run:
# Short excerpt of OpenParlData resource "persons"
get_glimpse2(table = "persons")

## End(Not run)
```

`get_overview` *Retrieve overview of all tables and variables*

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_overview2` *Retrieve overview of all OpenParlData endpoints and fields*

Description

`get_overview2` retrieves the names of the main resources provided by the OpenParlData.ch REST API and the fields they contain.

Usage

```
get_overview2(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:
# Overview for OpenParlData REST API
get_overview2()

## End(Not run)
```

get_related_data2 *Retrieve related data for an OpenParlData record*

Description

`get_related_data2` retrieves related records that are available for an OpenParlData record via its linked resources.

Usage

```
get_related_data2(res, table, silent = FALSE)
```

Arguments

`res`	an OpenParlData record (typically one row) as returned by `get_data2()`.
`table`	name of the related table to retrieve. Use `get_related_tables2()` to see which related tables are available.
`silent`	if `TRUE`, no progress bar and messages are displayed.

Details

The function downloads the related data for a specified related table and combines the results into a single tibble. If multiple entities are present, the function iterates over them and optionally displays a progress bar.

Value

A tibble containing the related records.

Examples

```
## Not run:
# Retrieve one person
res <- get_data2("persons", firstname = "Gerhard", lastname = "Andrey")

# List available related tables
get_related_tables2(res)

# Retrieve related data (replace "memberships" with an available table)
get_related_data2(res, table = "memberships")

## End(Not run)
```

get_related_tables2 *List related tables available*

Description

`get_related_tables2` returns the names of related tables that are available for an OpenParlData record.

Usage

```
get_related_tables2(res)
```

Arguments

`res`	an OpenParlData record (typically one row) as returned by `get_data2()`.

Value

A sorted character vector containing the names of available related tables for the provided record.

Examples

```
## Not run:
# Retrieve one person
res <- get_data2("persons", firstname = "Gerhard", lastname = "Andrey")

# List available related resources for that record
get_related_tables2(res)

## End(Not run)
```

get_tables *Retrieve available tables*

Description

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

Usage

```
get_tables()
```

Value

A sorted character vector containing the names of the available tables.

Examples

```
## Not run:
# Get all available tables
get_tables()

## End(Not run)
```

`get_tables2` *Retrieve available OpenParlData endpoints*

Description

`get_tables2` retrieves the names of the main REST API endpoints provided by the OpenParlData.ch API.

Usage

```
get_tables2()
```

Value

A sorted character vector containing the names of the available OpenParlData REST API endpoints.

Examples

```
## Not run:
# Get all available OpenParlData endpoints
get_tables2()

## 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 queried. 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 object. Not to be specified outside of `get_overview()`.

Value

A sorted character vector containing the names of the variables.

Examples

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

## End(Not run)
```

`get_variables2` *Retrieve available variables from an OpenParlData endpoint*

Description

`get_variables2` retrieves the field names of a resource provided by the OpenParlData.ch REST API.

Usage

```
get_variables2(table, pb.pos = NULL, pb = NULL)
```

Arguments

`table`	name of the OpenParlData resource to be queried. For an overview of available endpoints use `get_tables2()`.
`pb.pos`	value for the progress bar. Not to be specified outside of `get_overview()`.
`pb`	progress bar object. Not to be specified outside of `get_overview()`.

Value

A sorted character vector containing the names of the fields available in the selected OpenParlData endpoint.

Examples

```
## Not run:
# Get variables of OpenParlData resource "persons"
get_variables2(table = "persons")

## 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

`votes`	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 annontated.
`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

Index