Package ‘votesmart’

October 12, 2022

Type Package

Title Wrapper for the Project 'VoteSmart' API

Version 0.1.0

Maintainer Amanda Dobbyn <amanda@deck.tools>

Description An R interface to the Project 'VoteSmart' <https://justfacts.votesmart.org/> API.

License MIT + file LICENSE

URL https://github.com/decktools/votesmart/

BugReports https://github.com/decktools/votesmart/issues/

Depends R (>= 3.2)

Imports dplyr (>= 1.0.0), gestalt (>= 0.1.8), glue (>= 1.3.1), http (> = 1.4.1), jsonlite (>= 1.6.1), lubridate (>= 1.7.4), magrittr (>= 1.5), purrr (>= 0.3.3), snakecase (>= 0.11.0), stringr (>= 1.4.0), tidytr (>= 1.0.2)

Suggests conflicted (>= 1.0.4), covr (>= 3.4.0), knitr (>= 1.27), markdown (>= 2.1), spelling (>= 2.1), testthat (>= 2.1.0)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Deck Technologies [cph, fnd], Amanda Dobbyn [aut, cre], Max Wood [aut], Alyssa Frazee [aut]

Repository CRAN

Date/Publication 2021-02-22 11:30:03 UTC
**R topics documented:**

- `candidates_get_by_lastname` ................................. 2
- `candidates_get_by_levenshtein` ..................... 3
- `candidates_get_by_office_state` ...................... 4
- `election_get_election_by_year_state` .................. 5
- `endpoint_input_mapping` ................................. 6
- `endpoint_input_mapping_nested` ....................... 6
- `measure_get_measures` .................................. 7
- `measure_get_measures_by_year_state` ................... 7
- `office_get_levels` ........................................ 8
- `office_get_offices_by_level` ............................ 9
- `rating_get_candidate_ratings` ........................... 9
- `rating_get_categories` ................................... 10
- `rating_get_sig` ........................................... 11
- `rating_get_sig_list` ..................................... 12
- `votes_get_by_official` .................................. 12

Index 14

---

`candidates_get_by_lastname`

*Get candidate data by last name*

**Description**

Get candidate data by last name

**Usage**

```r
candidates_get_by_lastname(
  last_names,
  election_years = lubridate::year(lubridate::today()),
  stage_ids = "",
  all = TRUE,
  verbose = TRUE
)
```

**Arguments**

- `last_names` Vector of candidate last names
- `election_years` Vector of election years. Default is the current year.
- `stage_ids` The stage_id of the election ("P" for primary or "G" for general). See also `election_get_election_by_year_state`.
- `all` Boolean: should all possible combinations of the variables be searched for, or just the exact combination of them in the order they are supplied?
- `verbose` Should cases when no data is available be messaged?
candidates_get_by_levenshtein

Value

A dataframe of candidates and their attributes. If a given last_name + election_year + stage_id combination returns no data, that row will be filled with NAs.

Examples

```r
## Not run:
candidates_get_by_lastname(c("Ocasio-Cortez", "Omar"), 2018)
## End(Not run)
```

candidates_get_by_levenshtein

Get candidate data by Levenshtein distance from last name

Description

From the API docs, http://api.votesmart.org/docs/Candidates.html, "This method grabs a list of candidates according to a fuzzy lastname match."

Usage

```r
candidates_get_by_levenshtein(
  last_names,
  election_years = lubridate::year(lubridate::today()),
  stage_ids = "",
  all = TRUE,
  verbose = TRUE
)
```

Arguments

- `last_names`: Vector of candidate last names
- `election_years`: Vector of election years. Default is the current year.
- `stage_ids`: The stage_id of the election ("P" for primary or "G" for general). See also `election_get_election_by_year_state`.
- `all`: Boolean: should all possible combinations of the variables be searched for, or just the exact combination of them in the order they are supplied?
- `verbose`: Should cases when no data is available be messaged?

Details

The actual Levenshtein distance of the result from the last_name provided is not available from the API.
candidates_get_by_office_state

Value

A dataframe of candidates and their attributes. If a given last_name + election_year + stage_id combination returns no data, that row will be filled with NAs.

Examples

```r
## Not run:
candidates_get_by_levenshtein(c("Bookr", "Klobucar"), 2020)
## End(Not run)
```

candidates_get_by_office_state

Get candidates by the state in which they hold office

Description

Get candidates by the state in which they hold office

Usage

```r
candidates_get_by_office_state(
  state_ids = NA,
  office_ids,
  election_years = lubridate::year(lubridate::today()),
  all = TRUE,
  verbose = TRUE
)
```

Arguments

- **state_ids**: Optional: vector of state abbreviations. Default is NA, for national elections.
- **office_ids**: Required: vector of office ids that candidates hold. See `office_get_levels` and `office_get_offices_by_level` for office ids.
- **election_years**: Optional: vector of election years in which the candidate held office. Default is the current year.
- **all**: Boolean: should all possible combinations of the variables be searched for, or just the exact combination of them in the order they are supplied?
- **verbose**: Should cases when no data is available be messaged?

Value

A dataframe of candidates and their attributes. If a given state_id + office_id + election_year combination returns no data, that row will be filled with NAs.
election_get_election_by_year_state

Get election info by election year and state

Description
Get election info by election year and state

Usage

```r
election_get_election_by_year_state(
  years = lubridate::year(lubridate::today()),
  state_ids = "",
  all = TRUE,
  verbose = TRUE
)
```

Arguments

- **years**: A vector of election years.
- **state_ids**: A vector of state abbreviations.
- **all**: Boolean: should all possible combinations of the variables be searched for, or just the exact combination of them in the order they are supplied?
- **verbose**: Should cases when no data is available be messaged?

Value
A dataframe of candidates and their attributes. If a given year + state_id returns no data, that row will be filled with NAs.

Examples

```r
## Not run:
election_get_election_by_year_state(years = c(2016, 2017))

## End(Not run)
```
### Endpoint-Input Mapping

**Description**

Unnested tibble containing the mapping between each endpoint, the inputs it takes, and whether those inputs are required. One or more input rows per endpoint.

**Usage**

`endpoint_input_mapping`

**Format**

A tibble with 108 rows and 3 variables:

- `endpoint` name of the API endpoint
- `input` one or multiple inputs that can be used in the request to that endpoint
- `required` boolean: whether that input is required for that endpoint

**Source**

[http://api.votesmart.org/docs/](http://api.votesmart.org/docs/)

### Nested Endpoint-Input Mapping

**Description**

Nested tibble containing the mapping between each endpoint, the inputs it takes, and whether those inputs are required.

**Usage**

`endpoint_input_mapping_nested`

**Format**

A tibble with 70 rows and 2 variables:

- `endpoint` name of the API endpoint
- `inputs` a list column containing one or more inputs and a boolean indicating whether they are required for that endpoint. Can be unnested with `tidyr::unnest`
measure_get_measures  

*Get information on a ballot measure*

**Source**

http://api.votesmart.org/docs/

**Description**

Ballot measure ids can be found with the `measure_get_measures_by_year_state` function.

**Usage**

```r
measure_get_measures(measure_ids, verbose = TRUE)
```

**Arguments**

- `measure_ids`: Vector of ballot measure ids.
- `verbose`: Should cases when no data is available be messaged?

**Value**

A dataframe with the columns `measure_id`, `measure_code`, `title`, `election_date`, `election_type`, `outcome`, `yes_votes`, `no_votes`, `summary`, `summary_url`, `measure_text`, `text_url`, `pro_url`, `con_url`.

**Examples**

```r
## Not run:
measure_get_measures("1234")

## End(Not run)
```

measure_get_measures_by_year_state  

*Get a dataframe of ballot measures by year and state*

**Description**

More information about these ballot measures can be found using the `measure_get_measures` function.
Usage

```r
measure_get_measures_by_year_state(
  years = lubridate::year(lubridate::today()),
  state_ids = state.abb,
  all = TRUE,
  verbose = TRUE
)
```

Arguments

- `years`: A vector of election years.
- `state_ids`: A vector of state abbreviations.
- `all`: Boolean: should all possible combinations of the variables be searched for, or just the exact combination of them in the order they are supplied?
- `verbose`: Should cases when no data is available be messaged?

Value

A dataframe of ballot measures and their attributes. If a given year + state_id returns no data, that row will be filled with NAs.

Examples

```r
## Not run:
measure_get_measures_by_year_state(years = c(2016, 2018), state_ids = c("MO", "IL", "VT"))
## End(Not run)
```

office_get_levels

Get office levels

Description

These are currently: F for Federal, S for State, and L for Local.

Usage

```r
office_get_levels()
```

Value

A dataframe with the columns `office_level_id` and `name`.

Examples

```r
## Not run:
office_get_levels()
## End(Not run)
```
office_get_offices_by_level

Get offices by level

Description

Get offices by level

Usage

office_get_offices_by_level(office_level_ids)

Arguments

office_level_ids
Vector of office levels.

Value

A dataframe with columns office_id, name, title, office_level_id, office_type_id, office_branch_id, short_title.

Examples

## Not run:
office_get_offices_by_level("F")

office_get_levels() %>%
pull(office_level_id) %>%
.[1] %>%
office_get_offices_by_level()

## End(Not run)

rating_get_candidate_ratings

Get SIG (Special Interest Group) ratings for candidates

Description

Get SIG (Special Interest Group) ratings for candidates
Usage

```r
rating_get_candidate_ratings(
  candidate_ids,
  sig_ids = "",
  all = TRUE,
  verbose = TRUE
)
```

Arguments

- **candidate_ids** A vector of candidate ids.
- **sig_ids** A vector of SIG ids. Default is "" for all SIGs.
- **all** Boolean: should all possible combinations of the variables be searched for, or just the exact combination of them in the order they are supplied?
- **verbose** Should cases when no data is available be messaged?

Value

A dataframe with the columns `rating_id`, `candidate_id`, `sig_id`, `rating`, `rating_name`, `timespan`, `categories`, `rating_text`.

Examples

```r
## Not run:
pelosi_id <- "26732"
rating_get_candidate_ratings(pelosi_id)
## End(Not run)
```

---

**rating_get_categories**  Get categories that contain ratings by state

Description

Get categories that contain ratings by state

Usage

```r
rating_get_categories(state_ids = NA)
```

Arguments

- **state_ids** A vector of state abbreviations. Defaults to NA for national.

Value

A dataframe with columns `category_id`, `name`, `state_id`. 
Examples

## Not run:
rating_get_categories("NM")

## End(Not run)

rating_get_sig  Get information on a SIG (Special Interest Group) by its ID

Description

Get information on a SIG (Special Interest Group) by its ID

Usage

rating_get_sig(sig_ids, verbose = TRUE)

Arguments

sig_ids  Vector of SIG ids.
verbose  Should cases when no data is available be messaged?

Value

A dataframe with the columns sig_id, name, description, state_id, address, city, state, zip, phone_1, phone_2, fax, email, url, contact_name.

Examples

## Not run:
rating_get_sig_list(2) %>%
  dplyr::pull(sig_id) %>%
  sample(3) %>%
  rating_get_sig()

## End(Not run)
**rating_get_sig_list**  
*Get SIG (Special Interest Group) list by category and state*

**Description**

Get SIG (Special Interest Group) list by category and state

**Usage**

```r
rating_get_sig_list(category_ids, state_ids = NA, all = TRUE, verbose = TRUE)
```

**Arguments**

- `category_ids` Vector of category ids.
- `state_ids` Vector of state abbreviations. Default NA for national.
- `all` Boolean: should all possible combinations of the variables be searched for, or just the exact combination of them in the order they are supplied?
- `verbose` Should cases when no data is available be messaged?

**Value**

A dataframe with the columns `sig_id`, `name`, `category_id`, `state_id`.

**Examples**

```r
## Not run:
rating_get_categories() %>%
  dplyr::pull(category_id) %>%
  sample(3) %>%
  rating_get_sig_list()

## End(Not run)
```

---

**votes_get_by_official**  
*Get votes by official*

**Description**

Get votes by official
votes_get_by_official

Usage

votes_get_by_official(
  candidate_ids,
  office_ids = "",
  category_ids = "",
  years = "",
  all = TRUE,
  verbose = TRUE
)

Arguments

candidate_ids  Vector of candidate_ids (required). See candidates_get_by_lastname, candidates_get_by_levenshtein, and candidates_get_by_office_state.
office_ids     Vector of office_ids. See office_get_offices_by_level.
category_ids   Vector of category_ids. See rating_get_categories.
years          Vector of years in which the vote was taken.
all             Boolean: should all possible combinations of the variables be searched for, or just the exact combination of them in the order they are supplied?
verbose        Should cases when no data is available be messaged?

Value

A dataframe of candidates’ votes on bills and their attributes. If a given input combination returns no data, that row will be filled with NAs.

Examples

## Not run:
aoc <- candidates_get_by_lastname(
  "ocasio-cortez",
  election_years = "2018"
)
votes_get_by_official(aoc$candidate_id)

## End(Not run)
Index

* datasets
  endpoint_input_mapping, 6
  endpoint_input_mapping_nested, 6

candidates_get_by_lastname, 2, 13
candidates_get_by_levenshtein, 3, 13
candidates_get_by_office_state, 4, 13

election_get_election_by_year_state, 2, 3, 5
endpoint_input_mapping, 6
endpoint_input_mapping_nested, 6

measure_get_measures, 7, 7
measure_get_measures_by_year_state, 7, 7

office_get_levels, 4, 8
office_get_offices_by_level, 4, 9, 13

rating_get_candidate_ratings, 9
rating_get_categories, 10, 13
rating_get_sig, 11
rating_get_sig_list, 12

votes_get_by_official, 12