# Package ‘rsunlight’

August 29, 2016

**Title** Interface to 'Sunlight' Foundation 'APIs'

**Description** Interface to three 'Sunlight' Foundation 'APIs' (http://sunlightfoundation.com/api/) for government data, including the Congress 'API' 'v3', the Capitol Words 'API', and the Open States 'API'. 'Sunlight' Foundation is a 'nonpartison' 'nonprofit' that collects and provides data on government activities, and those that influence government. Functions are provided to interact with each of the three 'APIs'.

**Version** 0.4.2

**License** MIT + file LICENSE

**URL** https://github.com/ropengov/rsunlight

**BugReports** https://github.com/ropengov/rsunlight/issues

**VignetteBuilder** knitr

**LazyData** true

**Imports** methods, utils, httr, plyr, jsonlite, stringr

**Suggests** knitr, testthat

**Enhances** ggplot2

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Author** Scott Chamberlain [aut, cre],
Thomas J. Leeper [ctb]

**Maintainer** Scott Chamberlain <myrmecocystus@gmail.com>

**Repository** CRAN

**Date/Publication** 2015-12-31 22:19:55

## R topics documented:

## rsunlight-package *Sunlight Foundation data from R*

### Description

You need API keys for Sunlight Foundation APIs. Please get your own API keys if you plant to use these functions for Sunlight Labs (http://services.sunlightlabs.com/). We set up the functions so that you can use either env vars, or R options. For env vars, put an entry in your `.Renviron` file with the name `SUNLIGHT_LABS_KEY`, so the full thing would be `SUNLIGHT_LABS_KEY=<key>`. For R options, put the key in your `.Rprofile` file like `options(SunlightLabsKey = "key")`. Both are called on R startup, and then you don't have to enter your API key for each run of a function.

### Details

Currently we have functions to interface with the following Sunlight Foundation APIs, where functions for each API are prefixed with a two letter code indicating the service.

- Congress API ('cg')
- Open States API ('os')
- Capitol Words API ('cw')

Note that Puerto Rico is not included in Sunlight Foundation data.

### Author(s)

Scott Chamberlain <`myrmecocystus@gmail.com`>

Thomas J. Leeper <`thosjleeper@gmail.com`>

## cg_amendments *Search for data on amendments.*

### Description

Search for data on amendments.

### Usage

```
cg_amendments(amendment_id = NULL, amendment_type = NULL, number = NULL,
  congress = NULL, chamber = NULL, house_number = NULL,
  introduced_on = NULL, last_action_at = NULL, amends_bill_id = NULL,
  amends_treaty_id = NULL, amends_amendment_id = NULL,
  sponsor_type = NULL, sponsor_id = NULL, query = NULL, fields = NULL,
  page = 1, per_page = 20, order = NULL, key = NULL, as = "table",
  ...)
```

### Arguments

`amendment_id` The unique ID for this amendment. Formed from the amendment_type, number, and congress.

`amendment_type` The type for this amendment. For the amendment 'H.Amdt. 10', the amendment_type represents the 'H.Amdt.' part. Amendment types can be either hamdt or samdt.

`number` The number for this amendment. For the amendment 'H.Amdt. 10', the number is 10.

`congress` The Congress in which this amendment was introduced. For example, amendments introduced in the '113th Congress' have a congress of 113.

`chamber` The chamber in which the amendment was introduced.

`house_number` If the amendment was introduced in the House, this is a relative amendment number, scoped to the bill or treaty the House it relates to. How this number gets assigned is complicated and involves multiple institutions within the House and the Library of Congress. You can read the gory details if you want, but this number will usually do the job of connecting to data from the House Clerk's Electronic Voting System.

`introduced_on` The date this amendment was introduced.

`last_action_at` The date or time of the most recent official action on the amendment. Often, there are no official actions, in which case this field will be set to the value of introduced_on.

`amends_bill_id` If this amendment relates to a bill, this field is the ID of the related bill.

`amends_treaty_id` If this amendment relates to a treaty, this field is the ID of the related treaty. Treaty IDs are of the form treatyX-Y, where X is the treaty's number, and Y is the Congress the treaty is being considered in.

`amends_amendment_id`
If this amendment amends an amendment, this field is the ID of the amended amendment.

`sponsor_type` Whether the amendment is sponsored by a 'person' or a 'committee'.

`sponsor_id` If the sponsor_type is 'person', this will be that legislator's bioguide ID. If the sponsor_type is 'committee', this will be that committee's ID.

`query` (character) Return results the API thinks best match your query. Queries are interpreted as phrases.

`fields` You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don't supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it's recommended to always specify which fields you will be using.

`page` Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page.

`per_page` Number of records to return. Default: 20. Max: 50.

`order` Sort results by one or more fields with the order parameter. order is optional, but if no order is provided, the order of results is not guaranteed to be predictable. Append `__asc` or `__desc` to the field names to control sort direction. The default direction is `desc`, because it is expected most queries will sort by a date. Any field which can be used for filtering may be used for sorting. On full-text search endpoints (URLs ending in `/search`), you may sort by score to order by relevancy.

`key` your SunlightLabs API key; loads from environment variable from .Renviron or from an option from .Rprofile

`as` (character) One of table (default), list, or response (httr response object). When table is requested, the default, a table is not always returned, but is returned almost always. You can then make a table yourself if you like.

`...` Optional additional curl options (debugging tools mostly). See examples.

## Examples

```
## Not run:
cg_amendments()
cg_amendments(chamber='house', congress=113)
cg_amendments(sponsor_type='committee', sponsor_id='HSRU')
cg_amendments(amends_bill_id='hr624-113')

# most parameters are vectorized, pass in more than one value
cg_amendments(chamber = c('house', 'senate'), per_page=2)

## End(Not run)
```

## cg_bills *Gets details for bills.*

### Description

Data on bills in Congress goes back to 2009, and comes from a mix of sources:

- Scrapers at github.com/unitedstates for most data, including core status and history information.
- Bulk data at GPO's FDSys for version information, and full text.
- The House' MajorityLeader.gov and Senate Democrats' official site for notices of upcoming debate.

### Usage

```
cg_bills(query = NULL, bill_id = NULL, bill_type = NULL, number = NULL,
  congress = NULL, chamber = NULL, introduced_on = NULL,
  last_action_at = NULL, last_vote_at = NULL, last_version_on = NULL,
  highlight = NULL, history.active = NULL, history.active_at = NULL,
  history.house_passage_result = NULL,
  history.house_passage_result_at = NULL,
  history.senate_cloture_result = NULL,
  history.senate_cloture_result_at = NULL,
  history.senate_passage_result = NULL,
  history.senate_passage_result_at = NULL, history.vetoed = NULL,
  history.vetoed_at = NULL, history.house_override_result = NULL,
  history.house_override_result_at = NULL,
  history.senate_override_result = NULL,
  history.senate_override_result_at = NULL,
  history.awaiting_signature = NULL,
  history.awaiting_signature_since = NULL, history.enacted = NULL,
  history.enacted_at = NULL, sponsor.party = NULL,
  enacted_as.law_type = NULL, bill_type__in = NULL,
  history.house_passage_result__exists = NULL,
  history.senate_passage_result__exists = NULL, nicknames = NULL,
  keywords = NULL, sponsor_id = NULL, cosponsor_ids = NULL,
  cosponsors_count = NULL, withdrawn_cosponsors_count = NULL,
  withdrawn_cosponsor_ids = NULL, committee_ids = NULL,
  related_bill_ids = NULL, enacted_as.congress = NULL,
  enacted_as.number = NULL, fields = NULL, page = 1, per_page = 20,
  order = NULL, key = NULL, as = "table", ...)
```

### Arguments

`query` Allows wildcards, quoting for phrases, and nearby word operators (full reference). You can also retrieve highlighted excerpts, and all normal operators and filters.

`bill_id` The unique ID for this bill. Formed from the bill_type, number, and congress.

`bill_type` The type for this bill. For the bill 'H.R. 4921', the bill_type represents the 'H.R.' part. Bill types can be: hr, hres, hjres, hconres, s, sres, sjres, sconres.

`number` The number for this bill. For the bill 'H.R. 4921', the number is 4921.

`congress` The Congress in which this bill was introduced. For example, bills introduced in the '111th Congress' have a congress of 111.

`chamber` The chamber in which the bill originated.

`introduced_on` The date this bill was introduced.

`last_action_at` The date or time of the most recent official action. In the rare case that there are no official actions, this field will be set to the value of introduced_on.

`last_vote_at` The date or time of the most recent vote on this bill.

`last_version_on`
The date the last version of this bill was published. This will be set to the introduced_on date until an official version of the bill's text is published.

`highlight` (logical) If TRUE, looks for words in query param close to each other.

`history.active` Whether this bill has had any action beyond the standard action all bills get (introduction, referral to committee, sponsors' introductory remarks). Only a small percentage of bills get this additional activity.

`history.active_at`
If this bill got any action beyond initial introduction, the date or time of the first such action. This field will stay constant even as further action occurs. For the time of the most recent action, look to the last_action_at field.

`history.house_passage_result`
The result of the last time the House voted on passage. Only present if this vote occurred. 'pass' or 'fail'.

`history.house_passage_result_at`
The date or time the House last voted on passage. Only present if this vote occurred.

`history.senate_cloture_result`
The result of the last time the Senate voted on cloture. Only present if this vote occurred. 'pass' or 'fail'.

`history.senate_cloture_result_at`
The date or time the Senate last voted on cloture. Only present if this vote occurred.

`history.senate_passage_result`
The result of the last time the Senate voted on passage. Only present if this vote occurred. 'pass' or 'fail'.

`history.senate_passage_result_at`
The date or time the Senate last voted on passage. Only present if this vote occurred.

`history.vetoed` Whether the bill has been vetoed by the President. Always present.

`history.vetoed_at`
The date or time the bill was vetoed by the President. Only present if this happened.

history.house_override_result
The result of the last time the House voted to override a veto. Only present if this vote occurred. 'pass' or 'fail'.

history.house_override_result_at
The date or time the House last voted to override a veto. Only present if this vote occurred.

history.senate_override_result
The result of the last time the Senate voted to override a veto. Only present if this vote occurred. 'pass' or 'fail'.

history.senate_override_result_at
The date or time the Senate last voted to override a veto. Only present if this vote occurred.

history.awaiting_signature
Whether the bill is currently awaiting the President's signature. Always present.

history.awaiting_signature_since
The date or time the bill began awaiting the President's signature. Only present if this happened.

history.enacted
Whether the bill has been enacted into law. Always present.

history.enacted_at
The date or time the bill was enacted into law. Only present if this happened.

sponsor.party XXX

enacted_as.law_type
Whether the law is a public or private law. Most laws are public laws; private laws affect individual citizens. 'public' or 'private'.

bill_type__in XXX

history.house_passage_result__exists
XXX

history.senate_passage_result__exists
XXX

nicknames An array of common nicknames for a bill that don't appear in official data. These nicknames are sourced from a public dataset at unitedstates/bill-nicknames, and will only appear for a tiny fraction of bills. In the future, we plan to auto-generate acronyms from bill titles and add them to this array.

keywords A list of official keywords and phrases assigned by the Library of Congress. These keywords can be used to group bills into tags or topics, but there are many of them (1,023 unique keywords since 2009, as of late 2012), and they are not grouped into a hierarchy. They can be assigned or revised at any time after introduction.

sponsor_id The bioguide ID of the bill's sponsoring legislator, if there is one. It is possible, but rare, to have bills with no sponsor.

cosponsor_ids An array of bioguide IDs for each cosponsor of the bill. Bills do not always have cosponsors.

cosponsors_count
The number of active cosponsors of the bill.

`withdrawn_cosponsors_count`
The number of withdrawn cosponsors of the bill.

`withdrawn_cosponsor_ids`
An array of bioguide IDs for each legislator who has withdrawn their cosponsorship of the bill.

`committee_ids` A list of IDs of committees related to this bill.

`related_bill_ids`
A list of IDs of bills that the Library of Congress has declared 'related'. Relations can be pretty loose, use this field with caution.

`enacted_as.congress`
The Congress in which this bill was enacted into law.

`enacted_as.number`
The number the law was assigned.

`fields` You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don't supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it's recommended to always specify which fields you will be using.

`page` Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page.

`per_page` Number of records to return. Default: 20. Max: 50.

`order` Sort results by one or more fields with the order parameter. order is optional, but if no order is provided, the order of results is not guaranteed to be predictable. Append `__asc` or `__desc` to the field names to control sort direction. The default direction is `desc`, because it is expected most queries will sort by a date. Any field which can be used for filtering may be used for sorting. On full-text search endpoints (URLs ending in `/search`), you may sort by score to order by relevancy.

`key` your SunlightLabs API key; loads from environment variable from .Renviron or from an option from .Rprofile

`as` (character) One of table (default), list, or response (httr response object). When table is requested, the default, a table is not always returned, but is returned almost always. You can then make a table yourself if you like.

`...` Optional additional curl options (debugging tools mostly). See examples.

## Details

History: The history field includes useful flags and dates/times in a bill's life. The above is a real-life example of H.R. 3590 - not all fields will be present for every bill. Time fields can hold either dates or times - Congress is inconsistent about providing specific timestamps.

## Value

Committee details including subcommittees and all members.

### Examples

```
## Not run:
# Bill lookup (i.e., filter)
cg_bills(congress=113, history.enacted=TRUE)
cg_bills(history.active=TRUE, order='last_action_at')
cg_bills(sponsor.party='R', history.vetoed=TRUE)
cg_bills(enacted_as.law_type='private', order='history.enacted_at')
cg_bills(bill_type__in='hjres|sjres', history.house_passage_result__exists=TRUE,
   history.senate_passage_result__exists=TRUE)

# Bill search
cg_bills(query='health care')
cg_bills(query='health care', history.enacted=TRUE)
cg_bills(query='freedom of information')
cg_bills(query='"freedom of information" accountab*')
cg_bills(query="'transparency accountability'~5", highlight=TRUE)

# Disable pagination
cg_bills(per_page='all')

# most parameters are vectorized, pass in more than one value
cg_bills(bill_id = c("hjres131-113", "hjres129-113", "s2921-113"))

## End(Not run)
```

---

## `cg_committees` *Gets details (subcommittees + membership) for a committee by id.*

---

### Description

Names, IDs, contact info, and memberships of committees and subcommittees in the House and Senate. All committee information is sourced from bulk data at github.com/unitedstates, which in turn comes from official House and Senate sources. Feel free to open a ticket with any bugs or suggestions. We only provide information on current committees and memberships. For historic data on committee names, IDs, and contact info, refer to the bulk data.

### Usage

```
cg_committees(member_ids = NULL, committee_id = NULL, chamber = NULL,
  subcommittee = NULL, parent_committee_id = NULL, query = NULL,
  fields = NULL, page = 1, per_page = 20, order = NULL, key = NULL,
  as = "table", ...)
```

### Arguments

| | |
|---|---|
| `member_ids` | An array of bioguide IDs of legislators that are assigned to this committee. |
| `committee_id` | Official ID of the committee, as it appears in various official sources (Senate, House, and Library of Congress). |

`chamber` The chamber this committee is part of. ’house’, ’senate’, or ’joint’.

`subcommittee` Whether or not the committee is a subcommittee.

`parent_committee_id` If the committee is a subcommittee, the ID of its parent committee.

`query` (character) Return results the API thinks best match your query. Queries are interpreted as phrases.

`fields` You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don’t supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it’s recommended to always specify which fields you will be using.

`page` Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page.

`per_page` Number of records to return. Default: 20. Max: 50.

`order` Sort results by one or more fields with the order parameter. order is optional, but if no order is provided, the order of results is not guaranteed to be predictable. Append `__asc` or `__desc` to the field names to control sort direction. The default direction is `desc`, because it is expected most queries will sort by a date. Any field which can be used for filtering may be used for sorting. On full-text search endpoints (URLs ending in `/search`), you may sort by score to order by relevancy.

`key` your SunlightLabs API key; loads from environment variable from .Renviron or from an option from .Rprofile

`as` (character) One of table (default), list, or response (httr response object). When table is requested, the default, a table is not always returned, but is returned almost always. You can then make a table yourself if you like.

`...` Optional additional curl options (debugging tools mostly). See examples.

## Value

Committee details including subcommittees and all members.

## Examples

```
## Not run:
cg_committees(member_ids='L000551')
cg_committees(committee_id='SSAP')
cg_committees(committee_id='SSAP', fields='members')
cg_committees(chamber='joint', subcommittee=FALSE)
cg_committees(parent_committee_id='HSWM')

# Disable pagination
cg_committees(per_page='all')

# Output a list
cg_committees(member_ids='L000551', as='list')
# Output an httr response object, for debugging purposes
```

```
cg_committees(member_ids='L000551', as='response')

# most parameters are vectorized, pass in more than one value
cg_committees(committee_id = c('SSAP', 'SSGA01'))

## End(Not run)
```

## `cg_districts` *Get districts for a latitude/longitude or zips*

### Description

Get districts for a latitude/longitude or zips

### Usage

```
cg_districts(latitude = NULL, longitude = NULL, zip = NULL,
  query = NULL, per_page = 20, page = 1, order = NULL, key = NULL,
  as = "table", ...)
```

### Arguments

| | |
|---|---|
| `latitude` | (numeric) latitude of coordinate |
| `longitude` | (numeric) longitude of coordinate |
| `zip` | (integer) A 5 digit zip code |
| `query` | (character) Return results the API thinks best match your query. Queries are interpreted as phrases. |
| `per_page` | Number of records to return. Default: 20. Max: 50. |
| `page` | Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page. |
| `order` | Sort results by one or more fields with the order parameter. order is optional, but if no order is provided, the order of results is not guaranteed to be predictable. Append `__asc` or `__desc` to the field names to control sort direction. The default direction is `desc`, because it is expected most queries will sort by a date. Any field which can be used for filtering may be used for sorting. On full-text search endpoints (URLs ending in `/search`), you may sort by score to order by relevancy. |
| `key` | your SunlightLabs API key; loads from .Rprofile |
| `as` | (character) One of table (default), list, or response (httr response object) |
| `...` | Optional additional curl options (debugging tools mostly). See examples. |

### Details

A zip code may intersect multiple Congressional districts, so it is not as precise as using a latitude and longitude. In general, we recommend against using zip codes to look up members of Congress. For one, it's imprecise: a zip code can intersect multiple congressional districts. More importantly, zip codes are not shapes. They are lines (delivery routes), and treating them as shapes leads to inaccuracies.

### Value

List including data.frame and metadata about results, list, or httr response object.

### Examples

```
## Not run:
cg_districts(zip = 27511)
cg_districts(latitude = 35.778788, longitude = -78.787805)

# most parameters are vectorized, pass in more than one value
cg_districts(zip = c(27511, 97202))

## End(Not run)
```

## `cg_documents` *Search for data on documents*

### Description

Search for data on documents

### Usage

```
cg_documents(document_id = NULL, document_type = NULL, chamber = NULL,
  committee_id = NULL, committee_names = NULL, congress = NULL,
  house_event_id = NULL, hearing_type_code = NULL, hearing_title = NULL,
  published_at = NULL, bill_id = NULL, description = NULL,
  version_code = NULL, bioguide_id = NULL, occurs_at = NULL,
  urls = NULL, text = NULL, text_preview = NULL, witness = NULL,
  fields = NULL, page = 1, per_page = 20, order = NULL, key = NULL,
  as = "table", ...)
```

### Arguments

`document_id` A unique id for each document.

`document_type` Document types are taken from the House document type code. See Details.

`chamber` House or Senate

`committee_id` Acronym a committee is associated with the document.

`committee_names`
Full names of the committees associated with the document.

`congress` Session of Congress.

`house_event_id` Unique ID for each hearing, assigned by the House.

`hearing_type_code`
This describes if the meeting is a "markup", "meeting" or "hearing".

`hearing_title` Title of the hearing associated with the document.

`published_at` Date and time of publication.

`bill_id` Bill ID associated with the document.

`description` Description of the hearing.

`version_code` The short-code for what stage the version of the bill. See GPO for explanations of the version code.

`bioguide_id` Unique identifier for a member of Congress if they are associated with the document.

`occurs_at` Date and time of a hearing associated with the document.

`urls` The original link to the document. The permalink is a link to a copy of the document hosted by the Sunlight Foundation.

`text` Extracted text from the document.

`text_preview` A preview of the text.

`witness` Information about a witness associated with a document.

`fields` You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don't supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it's recommended to always specify which fields you will be using.

`page` Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page.

`per_page` Number of records to return. Default: 20. Max: 50.

`order` Sort results by one or more fields with the order parameter. order is optional, but if no order is provided, the order of results is not guaranteed to be predictable. Append `__asc` or `__desc` to the field names to control sort direction. The default direction is `desc`, because it is expected most queries will sort by a date. Any field which can be used for filtering may be used for sorting. On full-text search endpoints (URLs ending in `/search`), you may sort by score to order by relevancy.

`key` your SunlightLabs API key; loads from environment variable from .Renviron or from an option from .Rprofile

`as` (character) One of table (default), list, or response (httr response object). When table is requested, the default, a table is not always returned, but is returned almost always. You can then make a table yourself if you like.

`...` Optional additional curl options (debugging tools mostly). See examples.

### Details

- CV - Committee vote
- WS - Witness statement
- WT - Witness truth statement
- WB - Witness biography
- CR - Committee report
- BR - Bill
- FA - Floor amendment
- CA - Committee amendment
- HT - Transcript
- WD - Witness document

*other* is used for all other documents.

### Examples

```
## Not run:
cg_documents()
cg_documents(per_page=4)
cg_documents(per_page=4)
cg_documents(fields=c('document_id','type'))

# most parameters are vectorized, pass in more than one value
cg_documents(chamber = c("house", "senate"))

## End(Not run)
```

## cg_floor_updates *Search for data on upcoming bills.*

### Description

Search for data on upcoming bills.

### Usage

```
cg_floor_updates(chamber = NULL, timestamp = NULL, congress = NULL,
  legislative_day = NULL, year = NULL, bill_ids = NULL, roll_ids = NULL,
  legislator_ids = NULL, query = NULL, fields = NULL, page = 1,
  per_page = 20, order = NULL, key = NULL, as = "table", ...)
```

### Arguments

`chamber` The chamber this update took place in. 'house' or 'senate'.

`timestamp` The time this update took place. For Senate updates, this actually means the time our system first observed the update, and is susceptible to error; the Senate does not offer precise timestamps.

`congress` The number of the Congress this update took place during.

`legislative_day`
The 'legislative day' this update took place in. The 'legislative day' is a formal construct that is usually, but not always, the same as the calendar day. For example, if a day's session of Congress runs past midnight, the legislative_day will often stay the same as it was before midnight, until that session adjourns. On January 3rd, it is possible that the same legislative_day could span two Congresses. (This occurred in 2013.)

`year` The 'legislative year' of the update. This is not quite the same as the calendar year - the legislative year changes at noon EST on January 3rd. A vote taken on January 1, 2013 has a 'legislative year' of 2012.

`bill_ids` An array of IDs of bills that are referenced by or associated with this floor update.

`roll_ids` An array of IDs of roll call votes that are referenced by or associated with this floor update.

`legislator_ids` An array of bioguide IDs of legislators that are referenced by this floor update.

`query` (character) Return results the API thinks best match your query. Queries are interpreted as phrases.

`fields` You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don't supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it's recommended to always specify which fields you will be using.

`page` Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page.

`per_page` Number of records to return. Default: 20. Max: 50.

`order` Sort results by one or more fields with the order parameter. order is optional, but if no order is provided, the order of results is not guaranteed to be predictable. Append `__asc` or `__desc` to the field names to control sort direction. The default direction is `desc`, because it is expected most queries will sort by a date. Any field which can be used for filtering may be used for sorting. On full-text search endpoints (URLs ending in `/search`), you may sort by score to order by relevancy.

`key` your SunlightLabs API key; loads from environment variable from .Renviron or from an option from .Rprofile

`as` (character) One of table (default), list, or response (httr response object). When table is requested, the default, a table is not always returned, but is returned almost always. You can then make a table yourself if you like.

`...` Optional additional curl options (debugging tools mostly). See examples.

### Examples

```
## Not run:
cg_floor_updates()
cg_floor_updates(chamber='house', query='Agreed to by voice vote')

# most parameters are vectorized, pass in more than one value
cg_floor_updates(chamber = c("house", "senate"))

## End(Not run)
```

## `cg_hearings` *Search for data on hearings*

### Description

Search for data on hearings

### Usage

```
cg_hearings(committee_id = NULL, occurs_at = NULL, congress = NULL,
  chamber = NULL, dc = NULL, bill_ids = NULL, hearing_type = NULL,
  query = NULL, fields = NULL, page = 1, per_page = 20, order = NULL,
  key = NULL, as = "table", ...)
```

### Arguments

`committee_id` (numeric) The ID of the committee holding the hearing.

`occurs_at` (character) The time the hearing will occur. format: " e.g. "2014-05-01T13:00:00Z"

`congress` (numeric) The number of the Congress the committee hearing is taking place during.

`chamber` (character) The chamber ('house', 'senate', or 'joint') of the committee holding the hearing.

`dc` (logical) Whether the committee hearing is held in DC `TRUE` or in the field `FALSE`

`bill_ids` (numeric) The IDs of any bills mentioned by or associated with the hearing.

`hearing_type` (character) (House only) The type of hearing this is. Can be: 'Hearing', 'Markup', 'Business Meeting', 'Field Hearing'.

`query` (character) Return results the API thinks best match your query. Queries are interpreted as phrases.

`fields` You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don't supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it's recommended to always specify which fields you will be using.

`page` Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page.

`per_page` Number of records to return. Default: 20. Max: 50.

`order` Sort results by one or more fields with the order parameter. order is optional, but if no order is provided, the order of results is not guaranteed to be predictable. Append `__asc` or `__desc` to the field names to control sort direction. The default direction is `desc`, because it is expected most queries will sort by a date. Any field which can be used for filtering may be used for sorting. On full-text search endpoints (URLs ending in `/search`), you may sort by score to order by relevancy.

`key` your SunlightLabs API key; loads from environment variable from .Renviron or from an option from .Rprofile

`as` (character) One of table (default), list, or response (httr response object). When table is requested, the default, a table is not always returned, but is returned almost always. You can then make a table yourself if you like.

`...` Optional additional curl options (debugging tools mostly). See examples.

## Value

url (character) (House only) A permalink to that hearing's description on that committee's official website.

description (character) A description of the hearing.

room (character) If the hearing is in DC, the building and room number the hearing is in. If the hearing is in the field, the address of the hearing.

hearing_id (character) (House only) A permalink to that hearing's description on that committee's official website.

## References

https://sunlightlabs.github.io/congress/hearings.html

## Examples

```
## Not run:
cg_hearings(chamber='house', dc=TRUE)
cg_hearings(query='children')

# most parameters are vectorized, pass in more than one value
cg_hearings(chamber = c('house', 'senate'))

## End(Not run)
```

## cg_legislators *Search for legislators.*

### Description

Search for legislators.

### Usage

```
cg_legislators(title = NULL, first_name = NULL, middle_name = NULL,
  last_name = NULL, name_suffix = NULL, nickname = NULL, party = NULL,
  state = NULL, state_name = NULL, state_rank = NULL, district = NULL,
  in_office = NULL, chamber = NULL, gender = NULL, phone = NULL,
  fax = NULL, office = NULL, website = NULL, contact_form = NULL,
  email = NULL, congress_office = NULL, bioguide_id = NULL,
  ocd_id = NULL, thomas_id = NULL, lis_id = NULL, crp_id = NULL,
  icpsr_id = NULL, votesmart_id = NULL, fec_ids = NULL,
  govtrack_id = NULL, congresspedia_url = NULL, twitter_id = NULL,
  youtube_id = NULL, facebook_id = NULL, senate_class = NULL,
  term_start = NULL, term_end = NULL, birthday = NULL, latitude = NULL,
  longitude = NULL, zip = NULL, query = NULL, fields = NULL, page = 1,
  per_page = 20, order = NULL, key = NULL, as = "table", ...)
```

### Arguments

`title` Title held by this legislator, either Sen or Rep

`first_name` Legislator's first name

`middle_name` Legislator's middle name or initial

`last_name` Legislator's last name

`name_suffix` Legislator's suffix (Jr., III, etc.)

`nickname` Preferred nickname of legislator (if any)

`party` Legislator's political party (D, I, or R)

`state` two letter abbreviation of legislator's state

`state_name` The full state name of the state this member represents.

`state_rank` (Senate only) The seniority of that Senator for that state. "junior" or "senior".

`district` If legislator is a representative, their district. 0 is used for At-Large districts

`in_office` 1 if legislator is currently serving, 0 if legislator is no longer in office due to defeat/resignation/death/etc.

`chamber` Chamber the member is in. One of "senate" or "house".

`gender` M or F

`phone` Congressional office phone number

`fax` Congressional office fax number

`office` Office number for the member's DC office.

`website` URL of Congressional website

`contact_form` URL of web contact form

`email` Legislator's email address (if known)

`congress_office` Legislator's Washington DC Office Address

`bioguide_id` Legislator ID assigned by http://bioguide.congress.gov/biosearch/biosearch.asp Congressional Biographical Directory (also used by Washington Post/NY Times)

`ocd_id` Identifier for this member across all countries and levels of government, as defined by the Open Civic Data project.

`thomas_id` Identifier for this member as it appears on THOMAS.gov and Congress.gov.

`lis_id` Identifier for this member as it appears on some of Congress' data systems (namely Senate votes).

`crp_id` Identifier for this member as it appears on Center for Responsive Politics OpenSecrets (http://opensecrets.org)

`icpsr_id` Identifier for this member as it is maintained by the Inter-university Consortium for Political and Social Research.

`votesmart_id` Legislator ID assigned by http://votesmart.org Project Vote Smart

`fec_ids` http://fec.gov Federal Election Commission ID

`govtrack_id` ID assigned by http://govtrack.us Govtrack.us

`congresspedia_url` URL of Legislator's entry on http://congresspedia.org Congresspedia

`twitter_id` The Twitter username for a member's official legislative account. This field does not contain the handles of campaign accounts.

`youtube_id` The YouTube username or channel for a member's official legislative account. This field does not contain the handles of campaign accounts. A few legislators use YouTube "channels" instead of regular accounts. These channels will be of the form channel/[id].

`facebook_id` The Facebook username or ID for a member's official legislative Facebook presence. ID numbers and usernames can be used interchangeably in Facebook's URLs and APIs. The referenced account may be either a Facebook Page or a user account.

`senate_class` for senators I, II, or III depending on the Senator's election term

`term_start` The date a member's current term started.

`term_end` The date a member's current term will end.

`birthday` YYYY-MM-DD formatted birth date

`latitude` latitude of coordinate

`longitude` longitude of coordinate

`zip` zip code to search

`query` (character) Return results the API thinks best match your query. Queries are interpreted as phrases.

`fields` You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don't supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it's recommended to always specify which fields you will be using.

`page` Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page.

`per_page` Number of records to return. Default: 20. Max: 50.

`order` Sort results by one or more fields with the order parameter. order is optional, but if no order is provided, the order of results is not guaranteed to be predictable. Append `__asc` or `__desc` to the field names to control sort direction. The default direction is `desc`, because it is expected most queries will sort by a date. Any field which can be used for filtering may be used for sorting. On full-text search endpoints (URLs ending in `/search`), you may sort by score to order by relevancy.

`key` your SunlightLabs API key; loads from environment variable from .Renviron or from an option from .Rprofile

`as` (character) One of table (default), list, or response (httr response object). When table is requested, the default, a table is not always returned, but is returned almost always. You can then make a table yourself if you like.

`...` Optional additional curl options (debugging tools mostly). See examples.

## Details

Currently the Sunlight Labs API provides two methods for obtaining information about legislators: sll_cg_getlegislator and sll_cg_getlegislatorlist. Both of these methods operate in essentially the same way, the main difference being that sll_cg_getlegislator returns a single legislator (or an error if the query would have resulted in multiple legislators) and sll_cg_getlegislatorlist returns a list of legislators (a list of size one is allowed). In other words, sll_cg_getlegislator is purely a convenience method to avoid dealing with a list of values when only one value is needed.

All social media account values can be turned into URLs by preceding them with the domain name of the service in question:

- http://twitter.com/[username]
- http://youtube.com/[username or channel ID]
- http://facebook.com/[username or ID]

**Note about zip codes from Sunlight Foundation (a direct quote):** "A zip code may intersect multiple Congressional districts, so locating by zip may return multiple representatives, and possibly more than 2 senators if the zip code crosses state borders. In general, we recommend against using zip codes to look up members of Congress. For one, it's imprecise: a zip code can intersect multiple congressional districts. More importantly, zip codes are not shapes. They are lines (delivery routes), and treating them as shapes leads to inaccuracies."

## Value

List of output fields.

## Examples

```
## Not run:
cg_legislators(last_name = 'Pelosi')
cg_legislators(party = 'D')
cg_legislators(facebook_id = 'mitchmcconnell')
cg_legislators(latitude = 35.778788, longitude = -78.787805)
cg_legislators(zip = 77006)

# Output a list
cg_legislators(last_name = 'Pelosi', as='list')
# Output an httr response object, for debugging purposes
cg_legislators(last_name = 'Pelosi', as='response')

# Pagination
cg_legislators(party = 'D', per_page=2)

# Curl debugging
library('httr')
cg_legislators(party = 'D', config=verbose())
cg_legislators(party = 'D', config=timeout(0.1))

# most parameters are vectorized, pass in more than one value
cg_legislators(party = c('D', 'R'))
cg_legislators(last_name = c('Pelosi', 'Merkley'))

## End(Not run)
```

# cg_nominations *Search for data on nominations*

## Description

Search for data on nominations

## Usage

```
cg_nominations(nomination_id = NULL, congress = NULL, number = NULL,
  received_on = NULL, last_action_at = NULL, organization = NULL,
  committee_ids = NULL, nominees = NULL, nominees.position = NULL,
  nominees.state = NULL, query = NULL, fields = NULL, page = 1,
  per_page = 20, order = NULL, key = NULL, as = "table", ...)
```

## Arguments

| | |
|---|---|
| nomination_id | The unique identifier for this nomination, taken from the Library of Congress. Of the form ’PN[number]-[congress]’. |
| congress | The Congress in which this nomination was presented. |

`number` The number of this nomination, taken from the Library of Congress. Can occasionally contain hyphens, e.g. 'PN64-01'.

`received_on` The date this nomination was received in the Senate.

`last_action_at` The date this nomination last received action. If there are no official actions, then this field will fall back to the value of received_on.

`organization` The organization the nominee would be appointed to, if confirmed.

`committee_ids` An array of IDs of committees that the nomination has been referred to for consideration.

`nominees` An array of objects with fields (described below) about each nominee. Nominations for civil posts tend to have only one nominee. Nominations for military posts tend to have batches of multiple nominees. In either case, the nominees field will always be an array.

`nominees.position` The position the nominee is being nominated for.

`nominees.state` The which state in the United States this nominee hails from. This field is only available for some nominees, and never for batches of multiple nominees.

`query` (character) Return results the API thinks best match your query. Queries are interpreted as phrases.

`fields` You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don't supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it's recommended to always specify which fields you will be using.

`page` Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page.

`per_page` Number of records to return. Default: 20. Max: 50.

`order` Sort results by one or more fields with the order parameter. order is optional, but if no order is provided, the order of results is not guaranteed to be predictable. Append `__asc` or `__desc` to the field names to control sort direction. The default direction is `desc`, because it is expected most queries will sort by a date. Any field which can be used for filtering may be used for sorting. On full-text search endpoints (URLs ending in `/search`), you may sort by score to order by relevancy.

`key` your SunlightLabs API key; loads from environment variable from .Renviron or from an option from .Rprofile

`as` (character) One of table (default), list, or response (httr response object). When table is requested, the default, a table is not always returned, but is returned almost always. You can then make a table yourself if you like.

`...` Optional additional curl options (debugging tools mostly). See examples.

## Examples

```
## Not run:
cg_nominations(order='received_on')
cg_nominations(committee_ids='SSAS')
cg_nominations(organization='Privacy and Civil Liberties Oversight Board')
cg_nominations(query='Petraeus')

# most parameters are vectorized, pass in more than one value
cg_nominations(party = c('PN1873-111', 'PN604-112'))

## End(Not run)
```

## cg_upcoming_bills *Search for data on upcoming bills.*

### Description

Search for data on upcoming bills.

### Usage

```
cg_upcoming_bills(scheduled_at = NULL, legislative_day = NULL,
  range = NULL, congress = NULL, chamber = NULL, source_type = NULL,
  bill_id = NULL, query = NULL, fields = NULL, page = 1,
  per_page = 20, order = NULL, key = NULL, as = "table", ...)
```

### Arguments

`scheduled_at` The exact time at which our systems first spotted this bill on the schedule in this chamber and on this legislative day. Currently, we check the schedules every 15 minutes.

`legislative_day` The legislative day this bill is scheduled for. Combine with the range field to understand precision. May be null.

`range` How precise this information is. One of ’day’, ’week’, or NULL. See Details for more.

`congress` The number of the Congress this bill has been scheduled in.

`chamber` The chamber which has scheduled this bill.

`source_type` The source for this information. ’house_daily’ (Majority Leader daily schedule or ’senate_daily’ (Senate Democrats’ Floor feed.

`bill_id` The ID of the bill that is being scheduled.

`query` (character) Return results the API thinks best match your query. Queries are interpreted as phrases.

`fields` You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don’t supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it’s recommended to always specify which fields you will be using.

`page` Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page.

`per_page` Number of records to return. Default: 20. Max: 50.

`order` Sort results by one or more fields with the order parameter. order is optional, but if no order is provided, the order of results is not guaranteed to be predictable. Append `__asc` or `__desc` to the field names to control sort direction. The default direction is `desc`, because it is expected most queries will sort by a date. Any field which can be used for filtering may be used for sorting. On full-text search endpoints (URLs ending in `/search`), you may sort by score to order by relevancy.

`key` your SunlightLabs API key; loads from environment variable from .Renviron or from an option from .Rprofile

`as` (character) One of table (default), list, or response (httr response object). When table is requested, the default, a table is not always returned, but is returned almost always. You can then make a table yourself if you like.

`...` Optional additional curl options (debugging tools mostly). See examples.

## Details

More info for range parameter

- day: bill has been scheduled specifically for the legislative_day.
- week: bill has been scheduled for the 'Week of' the legislative_day.
- NULL: bill has been scheduled at an indefinite time in the future. (legislative_day is null.)

The 'legislative day' is a formal construct that is usually, but not always, the same as the calendar day. For example, if a day's session of Congress runs past midnight, the legislative_day will often stay the same as it was before midnight, until that session adjourns. On January 3rd, it is possible that the same legislative_day could span two Congresses. (This occurred in 2013.)

## Examples

```
## Not run:
cg_upcoming_bills()

# Pass in more than one value to a parameter
cg_upcoming_bills(chamber = c("house", "senate"))

## End(Not run)
```

## cg_votes *Search for data on votes.*

### Description

Search for data on votes.

### Usage

```
cg_votes(roll_id = NULL, chamber = NULL, number = NULL, year = NULL,
  congress = NULL, voted_at = NULL, vote_type = NULL, roll_type = NULL,
  required = NULL, result = NULL, bill_id = NULL, nomination_id = NULL,
  query = NULL, fields = NULL, page = 1, per_page = 20, order = NULL,
  key = NULL, as = "table", callopts = list(), ...)
```

### Arguments

| | |
|---|---|
| `roll_id` | A unique identifier for a roll call vote. Made from the first letter of the chamber, the vote number, and the legislative year. |
| `chamber` | The chamber the vote was taken in. 'house' or 'senate' |
| `number` | The number that vote was assigned. Numbers reset every legislative year. |
| `year` | The 'legislative year' of the vote. This is not quite the same as the calendar year - the legislative year changes at noon EST on January 3rd. A vote taken on January 1, 2013 has a 'legislative year' of 2012. |
| `congress` | The Congress this vote was taken in. |
| `voted_at` | The time the vote was taken. |
| `vote_type` | The type of vote being taken. This classification is imperfect and unofficial, and may change as we improve our detection. Valid types are 'passage', 'cloture', 'nomination', 'impeachment', 'treaty', 'recommit', 'quorum', 'leadership', and 'other'. |
| `roll_type` | The official description of the type of vote being taken. |
| `required` | The required ratio of Aye votes necessary to pass the legislation. A value of '1/2' actually means more than 1/2. Ties are not possible in the Senate (the Vice President casts a tie-breaker vote), and in the House, a tie vote means the vote does not pass. |
| `result` | The official result of the vote. This is not completely standardized (both 'Passed' and 'Bill Passed' may appear). In the case of a vote for Speaker of the House, the result field contains the name of the victor. |
| `bill_id` | If a vote is related to a bill, the bill's ID. |
| `nomination_id` | If a vote is related to a nomination, the nomination's ID. |
| `query` | (character) Return results the API thinks best match your query. Queries are interpreted as phrases. |

`fields` You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don't supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it's recommended to always specify which fields you will be using.

`page` Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page.

`per_page` Number of records to return. Default: 20. Max: 50.

`order` Sort results by one or more fields with the order parameter. order is optional, but if no order is provided, the order of results is not guaranteed to be predictable. Append `__asc` or `__desc` to the field names to control sort direction. The default direction is `desc`, because it is expected most queries will sort by a date. Any field which can be used for filtering may be used for sorting. On full-text search endpoints (URLs ending in `/search`), you may sort by score to order by relevancy.

`key` your SunlightLabs API key; loads from .Rprofile

`as` (character) One of table (default), list, or response (httr response object)

`callopts` Curl options to be passed on to httr::GET

`...` See Details. You can pass on additional parameters.

## Details

Two parameters can be passed on that vary with vote and/or party plus vote:

- breakdown.total.[vote] The number of members who cast [vote], where [vote] is a valid vote as defined above.
- breakdown.party.[party].[vote] The number of members of [party] who cast [vote], where [party] is one of 'D', 'R', or 'I', and [vote] is a valid vote as defined above.

## Examples

```
## Not run:
cg_votes(chamber='senate', order='voted_at')
cg_votes(query='guns')
cg_votes(voter_ids.A000055__exists=TRUE)

# Pass in more than one value for a parameter
cg_votes(chamber = c('house', 'senate'))

## End(Not run)
```

## `cw_dates` *Capitol words dates.json method. Search the congressional record for instances of a word or phrase over time.*

### Description

Capitol words dates.json method. Search the congressional record for instances of a word or phrase over time.

### Usage

```
cw_dates(phrase = NULL, title = NULL, date = NULL, start_date = NULL,
  end_date = NULL, chamber = NULL, state = NULL, party = NULL,
  bioguide_id = NULL, congress = NULL, session = NULL, cr_pages = NULL,
  volume = NULL, page_id = NULL, n = NULL, mincount = NULL,
  granularity = NULL, percentages = "true", entity_type = NULL,
  entity_value = NULL, as = "table", key = NULL, ...)
```

### Arguments

| | |
|---|---|
| `phrase` | Phrase to search. |
| `title` | Title of page to search. |
| `date` | Show results for only the given date. |
| `start_date` | Start date to search on. |
| `end_date` | End date to search on. |
| `chamber` | Chamber of congress, House or Senate. |
| `state` | State, capital two-letter abbreviation (e.g., AK,AZ,NM). |
| `party` | Political party (one of D,R,I). |
| `bioguide_id` | Bioguide ID for politician (e.g., B000243) |
| `congress` | Congressional session (e.g., 110,111,112) |
| `session` | Session within the current congress (e.g., 1,2) |
| `cr_pages` | No definition. |
| `volume` | No definition. |
| `page_id` | Page id. |
| `n` | (integer) The size of phrase, in words, to search for (up to 5). The length of the phrase to search against. |
| `mincount` | Only return results where mentions are at or above the supplied threshold |
| `granularity` | The length of time covered by each result. Valid values: 'year', 'month', 'day' (default) |
| `percentages` | Include the percentage of mentions versus total words in the result objects. Valid values: 'true', 'false' (default) (character) |

`entity_type` The entity type to get top phrases for. One of 'date', 'month', 'state', or 'legislator'.

`entity_value` The value of the entity given in `entity_type`. See Details.

`as` (character) One of table (default), list, or response (httr response object).

`key` Your SunlightLabs API key; loads from .Rprofile.

`...` Further curl options (debugging tools mostly)

## Details

Formats for `entity_value` parameter are as follows:

- date: 2011-11-09
- month: 201111
- state: NY
- legislator (bioguide id): L000551

## Value

Data frame of observations by date.

## Examples

```
## Not run:
cw_dates(phrase='I would have voted', start_date='2001-01-20',
   end_date='2009-01-20', granularity='year', party='D')

cw_dates(phrase='united states', entity_type='state',
   entity_value='VA', granularity='year', party='D')

cw_dates(phrase='voting', start_date='2009-01-01',
   end_date='2009-04-30', granularity='month', party='R')

# pass in more than one value for some parameters
cw_dates(phrase='voting', start_date='2009-01-01',
   end_date='2009-04-30', granularity='month', party=c('R', 'D'))
cw_dates(phrase=c('voting', 'hate'), start_date='2009-01-01',
   end_date='2009-04-30', granularity='month', party='D')

## End(Not run)
```

## `cw_phrases` *Capitol words phrases.json method. List the top phrases for a facet.*

### Description

Capitol words phrases.json method. List the top phrases for a facet.

### Usage

```
cw_phrases(entity_type, entity_value, n = NULL, page = NULL,
  per_page = NULL, sort = NULL, as = "table", key = NULL, ...)
```

### Arguments

`entity_type` The entity type to get top phrases for. One of 'date', 'month', 'state', or 'legislator'. Required.

`entity_value` The value of the entity given in `entity_type`. See Details. Required.

`n` The size of phrase, in words, to search for (up to 5).

`page` The page of results to show. 100 results are shown at a time. To get more than 100 results, use the page parameter.

`per_page` Number of records to return. Default: 20. Max: 50.

`sort` The value on which to sort the results. You have to specify ascending or descending (see details), but if you forget, we make it ascending by default (prevents 500 error :)). Valid values are 'tfidf' (default) and 'count'.

`as` (character) One of table (default), list, or response (httr response object).

`key` Your SunlightLabs API key; loads from .Rprofile.

`...` Further curl options (debugging tools mostly)

### Value

Data frame of observations by date.

### Examples

```
## Not run:
cw_phrases(entity_type='month', entity_value=201007)
cw_phrases(entity_type='state', entity_value='NV')
cw_phrases(entity_type='legislator', entity_value='L000551')

library('httr')
head(cw_phrases(entity_type='month', entity_value=201007, config=verbose()))

## End(Not run)
```

## cw_text — *Capitol words text.json method. Search the congressional record for instances of a word or phrase.*

### Description

Capitol words text.json method. Search the congressional record for instances of a word or phrase.

### Usage

```
cw_text(phrase = NULL, title = NULL, date = NULL, start_date = NULL,
  end_date = NULL, chamber = NULL, state = NULL, party = NULL,
  bioguide_id = NULL, congress = NULL, session = NULL, cr_pages = NULL,
  volume = NULL, page = NULL, sort = NULL, as = "table", key = NULL,
  ...)
```

### Arguments

| | |
|---|---|
| `phrase` | Phrase to search. |
| `title` | Title of page to search. |
| `date` | Show results for only the given date. |
| `start_date` | Start date to search on. |
| `end_date` | End date to search on. |
| `chamber` | Chamber of congress, House or Senate. |
| `state` | State, capital two-letter abbreviation (e.g., AK,AZ,NM). |
| `party` | Political party (one of D,R,I). |
| `bioguide_id` | Bioguide ID for politician (e.g., B000243) |
| `congress` | Congressional session (e.g., 110,111,112) |
| `session` | Session within the current congress (e.g., 1,2) |
| `cr_pages` | No definition. |
| `volume` | No definition. |
| `page` | The page of results to show, 50 results are shown at a time. |
| `sort` | The value on which to sort the results. You have to specify ascending or descending (see details), but if you forget, we make it ascending by default (prevents 500 error :)). See Details. |
| `as` | (character) One of table (default), list, or response (httr response object). |
| `key` | Your SunlightLabs API key; loads from .Rprofile. |
| `...` | Further curl options (debugging tools mostly) |

## Details

Options for the `sort` parameter are:

- speaker_state
- congress
- title
- number
- volume
- chamber
- session
- id
- speaker_party
- date
- bioguide_id
- pages

Coupled with a direction, `asc` or `desc`. An example to sort by true chronological order and chamber (id works for this purpose) would be `id desc`.

## Value

Phrases matched in a data.frame.

## Examples

```
## Not run:
cw_text(phrase='climate change', start_date='2012-09-16', end_date='2012-09-20')
cw_text(phrase='I would have voted', start_date='2011-09-05', end_date='2011-09-16', party='D')
cw_text(phrase='I would have voted', start_date='2011-09-05', end_date='2011-09-16',
   chamber='House')
cw_text(title='personal explanation', start_date='2011-09-05', end_date='2011-09-16')

library('plyr')
out <- cw_text(phrase='climate change', start_date='2010-01-01', end_date='2012-12-01')
out2 <- ldply(2:6, function(x) cw_text(phrase='climate change', start_date='2010-01-01',
   end_date='2012-12-01', page=x))
rbind(out, out2)

cw_text(phrase='climate change', start_date='2012-09-16', end_date='2012-09-20', sort='title')

# pass in many values for some parametrs
cw_text(phrase=c('climate change', 'politics'), start_date='2012-09-16', end_date='2012-09-20')

## End(Not run)
```

## cw_timeseries *Find the popularity of a phrase over a period of time.*

### Description

Find the popularity of a phrase over a period of time.

### Usage

```
cw_timeseries(phrase = NULL, date = NULL, start_date = NULL,
  end_date = NULL, chamber = NULL, state = NULL, party = NULL,
  bioguide_id = NULL, mincount = NULL, percentages = NULL,
  granularity = "day", entity_type = NULL, entity_value = NULL,
  as = "table", key = NULL, ...)
```

### Arguments

`phrase` Phrase to search.

`date` Show results for only the given date.

`start_date` Start date to search on.

`end_date` End date to search on.

`chamber` Chamber of congress, House or Senate.

`state` State, capital two-letter abbreviation (e.g., AK,AZ,NM).

`party` Political party (one of D,R,I).

`bioguide_id` Bioguide ID for politician (e.g., B000243)

`mincount` Only return results where mentions are at or above the supplied threshold.

`percentages` Include the percentage of mentions versus total words in the result objects. Valid values: ’true’, ’false’ (default) (character)

`granularity` The length of time covered by each result. Valid values: ’year’, ’month’, ’day’ (default)

`entity_type` The entity type to get top phrases for. Valid values: ’date’, ’month’, ’state’, ’legislator’

`entity_value` The value of the entity given in entity_type. Formats are as follows: date: 2011-11-09; month: 201111; state: NY; legislator (bioguide): L000551

`as` (character) One of table (default), list, or response (httr response object).

`key` Your SunlightLabs API key; loads from .Rprofile.

`...` Further curl options (debugging tools mostly)

### Value

A data.frame with number of times (and percentages of the text result, if selected).

## Examples

```
## Not run:
# Search for a phrase to get a time series of
cw_timeseries(phrase='climate change')

# Get percentages back, which are not returned by default
cw_timeseries(phrase='climate change', percentages='true')

# Get a list of how many times the phrase "united states" was said by
# legislators from Virginia on each day of the most recent Congress:
cw_timeseries(phrase='united states', entity_type='state',
entity_value='VA')

# Get a list of how many times the phrase "united States" appears in the
# Congressional Record on each day between Jan. 1, 2010, and June 1, 2010:
cw_timeseries(phrase='united states', start_date='2009-01-01',
end_date='2009-06-01')

# Get a list of how many times the phrase "united states" appears in the
# Congressional Record in each month between January and June, 2010:
cw_timeseries(phrase='united states', start_date='2009-01-01',
end_date='2009-04-30', granularity='month')

# Plot data
library('ggplot2')
dat <- cw_timeseries(phrase='climate change')
ggplot(dat, aes(day, count)) +
   geom_line() +
   theme_grey(base_size=20)

dat_d <- cw_timeseries(phrase = 'climate change', party = "D")
dat_d$party <- rep("D", nrow(dat_d))
dat_r <- cw_timeseries(phrase = 'climate change', party = "R")
dat_r$party <- rep("R", nrow(dat_r))
dat_both <- rbind(dat_d, dat_r)
ggplot(dat_both, aes(day, count, colour = party)) +
   geom_line() +
   theme_grey(base_size = 20) +
   scale_colour_manual(values=c("blue", "red"))

## End(Not run)
```

---

## `os_billlookup` *Lookup bills on OpenStates.*

---

## Description

Lookup bills on OpenStates.

## Usage

```
os_billlookup(state = NULL, session = NULL, bill_id = NULL,
  fields = NULL, per_page = NULL, page = NULL, as = "table",
  key = NULL, ...)
```

## Arguments

`state` state two-letter abbreviation (character), required

`session` session of congress (integer), e.g., 2009-2010 = 20092010, required

`bill_id` One or more identification numbers of bills (character), required

`fields` You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don't supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it's recommended to always specify which fields you will be using.

`per_page` Number of records to return. Default: 20. Max: 50.

`page` Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page.

`as` (character) One of table (default), list, or response (httr response object)

`key` your SunlightLabs API key; loads from .Rprofile

`...` Curl options passed on to `GET`

## Value

List of output fields.

## Examples

```
## Not run:
os_billlookup(state='ca', session=20092010, bill_id='AB 667')
os_billlookup(state='ca', session=20092010, bill_id='AB 667', per_page=1)
os_billlookup(state='ca', session=20092010, bill_id='AB 667', per_page=1, fields='id')
os_billlookup(state='ca', session=20092010, bill_id='AB 667',
   per_page=3, fields=c('id','title'))
os_billlookup(state='ca', session=20092010, bill_id='SB 425')
os_billlookup(state='ca', session=20092010, bill_id=c('AB 667','SB 425'))

library('httr')
os_billlookup(state='ca', session=20092010, bill_id='AB 667', config=verbose(), per_page=1)

## End(Not run)
```

## `os_billsearch` *Search OpenStates bills.*

### Description

Search OpenStates bills.

### Usage

```
os_billsearch(terms = NULL, state = NULL, window = NULL,
  chamber = "upper", sponsor_id = NULL, updated_since = NULL,
  subject = NULL, type = NULL, search_window = NULL, sort = NULL,
  page = NULL, per_page = NULL, fields = NULL, as = "table",
  key = NULL, ...)
```

### Arguments

`terms` search terms bill search (character)

`state` state two-letter abbreviation (character)

`window` a string representing what time period to search across. Pass 'session' to search bills from the state's current or most recent legislative session, 'term' to search the current or most recent term, 'all' to search as far back as the Open State Project has data for, or supply 'session:SESSION_NAME' or 'term:TERM_NAME' (e.g. 'session:2009' or 'term:2009-2010') to search a specific session or term.

`chamber` one of 'upper' or 'lower' (character)

`sponsor_id` only return bills sponsored by the legislator with the given id (corresponds to leg_id)

`updated_since` only return bills that have been updated since a given date, YYYY-MM-DD format

`subject` filter by bills that are about a given subject. If multiple subject parameters are supplied then only bills that match all of them will be returned. See list of subjects

`type` (character) Only bills of a given type (e.g. 'bill', 'resolution', etc.)

`search_window` By default all bills are searched, but if a time window is desired the following options can be passed to search_window: all (Default, include all sessions.), term (Only bills from sessions within the current term.), session (Only bills from the current session.), session:2009 (Only bills from the session named 2009.), term:2009-2011 (Only bills from the sessions in the 2009-2011 session.)

`sort` (character) One of 'first' (default), 'last', 'signed', 'passed_lower', 'passed_upper', 'updated_at', or 'created_at'.

`page` Page to return. Default: 1. You can use this in combination with the per_page parameter to get more than the default or max number of results per page.

`per_page` Number of records to return. Default: 20. Max: 50.

fields You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don't supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it's recommended to always specify which fields you will be using.

as (character) One of table (default), list, or response (httr response object)

key your SunlightLabs API key; loads from .Rprofile

... Curl options passed on to GET

### Value

a data.frame of bills.

### Examples

```
## Not run:
os_billsearch(terms = 'agriculture', state = 'tx')
os_billsearch(terms = 'agriculture', state = 'tx', chamber = 'upper')
os_billsearch(terms = 'taxi', state = 'dc')
os_billsearch(terms = 'taxi', state = 'dc', per_page=3)
os_billsearch(terms = 'taxi', state = 'dc', per_page=3, sort='created_at')
os_billsearch(terms = 'taxi', state = 'dc', type='resolution')

# Search window
os_billsearch(terms = 'climate change', search_window='term')
os_billsearch(terms = 'climate change', search_window='term:2009-2011')
os_billsearch(terms = 'climate change', search_window='session')
os_billsearch(terms = 'climate change', search_window='session:2009')

os_billsearch(terms = 'agriculture', state = 'tx', per_page=2)
os_billsearch(terms = 'agriculture', state = 'tx', per_page=2, page=2)
os_billsearch(terms = 'agriculture', state = 'tx', fields=c('id','created_at'), per_page=10)

# Pass in more than one value for some parameters
os_billsearch(terms = 'agriculture', state = c('tx', 'or'))
os_billsearch(terms = 'agriculture', state = ”or”, chamber = c('upper', 'lower'))

## End(Not run)
```

## os_legislatorsearch *Search Legislators on OpenStates.*

### Description

Search Legislators on OpenStates.

## Usage

```
os_legislatorsearch(state = NULL, first_name = NULL, last_name = NULL,
  chamber = NULL, active = NULL, term = NULL, district = NULL,
  party = NULL, fields = NULL, as = ”table”, key = NULL, ...)
```

## Arguments

`state` state two-letter abbreviation (character)

`first_name` first name of legislator (character)

`last_name` last name of legislator (character)

`chamber` one of ’upper’ or ’lower’ (character)

`active` TRUE or FALSE (character)

`term` filter by legislators who served during a certain term (character)

`district` legislative district (character)

`party` democratic or republican (character)

`fields` You can request specific fields by supplying a vector of fields names. Many fields are not returned unless requested. If you don’t supply a fields parameter, you will get the most commonly used subset of fields only. To save on bandwidth, parsing time, and confusion, it’s recommended to always specify which fields you will be using.

`as` (character) One of table (default), list, or response (httr response object)

`key` your SunlightLabs API key; loads from .Rprofile

`...` Curl options passed on to `GET`

## Value

List of output fields.

## Examples

```
## Not run:
os_legislatorsearch(state = 'ca', party = 'democratic')
os_legislatorsearch(state = 'tx', party = 'democratic', active = TRUE)
os_legislatorsearch(state = 'nv', party = 'republican')
os_legislatorsearch(state = 'dc', chamber = 'upper')

# pass in more than one value for some parameters
os_legislatorsearch(state = c('dc', 'or'), chamber = 'upper')
os_legislatorsearch(first_name = c('jane', 'bob'), chamber = 'upper')

## End(Not run)
```

## os_statemetasearch *Search OpenStates metadata.*

### Description

Search OpenStates metadata.

### Usage

```
os_statemetasearch(state = NULL, key = NULL, ...)
```

### Arguments

| | |
|---|---|
| state | One or more two-letter state abbreviations (character) |
| key | your SunlightLabs API key; or loads from .Rprofile |
| ... | Curl options passed on to GET |

### Value

A list with metadata for each state.

### Examples

```
## Not run:
os_statemetasearch()
os_statemetasearch('ca')
os_statemetasearch('tx')
out <- os_statemetasearch(c('tx','nv'))
out$tx
out$nv

library('httr')
os_statemetasearch('tx', config=verbose())

## End(Not run)
```

## rsunlight-defunct *Defunct functions in rsunlight*

## Description

- `ie_politicians`: Removed. See `Influence Explorer` below
- `ie_organizations`: Removed. See `Influence Explorer` below
- `ie_industries`: Removed. See `Influence Explorer` below
- `ie_individuals`: Removed. See `Influence Explorer` below
- `ie_lobbying`: Removed. See `Influence Explorer` below
- `ie_grants`: Removed. See `Influence Explorer` below
- `ie_contracts`: Removed. See `Influence Explorer` below
- `ie_faca`: Removed. See `Influence Explorer` below
- `ie_epa`: Removed. See `Influence Explorer` below
- `ie_entities`: Removed. See `Influence Explorer` below
- `ie_earmarks`: Removed. See `Influence Explorer` below
- `ie_contr_bundled`: Removed. See `Influence Explorer` below
- `ie_contr`: Removed. See `Influence Explorer` below

## Influence Explorer

The Influence Explorer API is now defunct/gone.

---

# `sessions` *Congressional sessions, up to the 114th (2015-2016)*

---

## Description

Congressional sessions, up to the 114th (2015-2016)

# Index