Package 'repoRter.nih'

May 17, 2022

Type Package

Title R Interface to the 'NIH RePORTER Project' API

Version 0.1.3

Description Methods to easily build requests in the non-standard JSON schema required by the National Institute of Health (NIH)'s 'RePORTER Project API' <https://api.reporter.nih.gov/#/Search/post_v2_projects_search>. Also retrieve and process result sets as either a ragged or flattened 'tibble'.

License MIT + file LICENSE

URL https://github.com/bikeactuary/repoRter.nih

BugReports https://github.com/bikeactuary/repoRter.nih/issues

Depends R (>= 4.1.0)

Imports assertthat (>= 0.2.1), crayon (>= 1.4.1), dplyr (>= 1.0.7), httr (>= 1.4.2), janitor (>= 2.1.0), jsonlite (>= 1.7.2), lubridate (>= 1.7.10), magrittr (>= 2.0.1), purrr (>= 0.3.4), tibble (>= 3.1.3)

Suggests devtools, ggplot2, ggrepel, knitr, tinytex, rmarkdown, scales, tufte, spelling

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.2

NeedsCompilation no

Author Michael Barr, ACAS, MAAA, CPCU [cre, aut]

Maintainer ``Michael Barr, ACAS, MAAA, CPCU" <mike@bikeactuary.com>

Repository CRAN

Date/Publication 2022-05-17 14:00:02 UTC

R topics documented:

`covid_response_codes` `covid_response_code` *translation*

Description

A `tibble` containing name translations between `covid_response_code` and the funding source(s)

Usage

```
data("covid_response_codes")
```

Format

A `tibble` with 6 rows and 3 columns:

`covid_response_code` the name for a data element when specified in the payload criteria of a request; NA indicates that this is not available as payload criteria (can not search/filter on)

`funding_source` the name of the funding source, often some federal legislation

`fund_src` a short name for the funding source

References

NIH RePORTER API Documentation

`get_nih_data` *get_nih_data*

Description

Easily send a pre-made JSON request to NIH RePORTER Project API, retrieve and process the results

Usage

```
get_nih_data(
  query,
  max_pages = NULL,
  flatten_result = FALSE,
  return_meta = FALSE
)
```

Arguments

query	A valid JSON request formatted for the RePORTER Project API, as returned by the make_req method
max_pages	numeric(1); default: NULL; An integer specifying to only fetch (up to) the first max_pages number of pages from the result set. Useful for testing your query/obtaining schema information. Default behavior is to fetch all pages.
flatten_result	(default: FALSE) If TRUE, flatten nested dataframes and collapse nested vectors to a single character column with elements delimited by a semi-colon
return_meta	(default: FALSE) If TRUE, will return a list containing your result set as well as the meta data - this includes a count of total projects matching your query and can be useful for programming.

Details

A request to the RePORTER Project API requires retrieving paginated results, combining them, and often flattening the combined ragged data.frame to a familiar flat format which we can use in analyses. This method handles all of that for you.

Value

When return_meta = FALSE: a tibble containing your result set (up to API max of 10,000 records); else if include_meta = TRUE, a named list containing the result set and the metadata from the initial API response.

If an API error occurs, this method will print an informative message and return NA.

Examples

```
library(repoRter.nih)

## make the usual request
req <- make_req(criteria =
                    list(advanced_text_search =
                        list(operator = "Or",
                             search_field = "all",
                             search_text = "sarcoidosis lupus") ),
                message = FALSE)

## get the data ragged
res <- get_nih_data(req,
                    max_pages = 1)

## get the data flattened
res_flattened <- get_nih_data(req,
                              flatten_result = TRUE,
                              max_pages = 1)
```

make_req *make_req*

Description

Easily generate a json request with correct schema to be passed to NIH RePORTER Project API

Usage

```
make_req(
  criteria = list(fiscal_years = lubridate::year(Sys.Date())),
  include_fields = NULL,
  exclude_fields = NULL,
  offset = 0,
  limit = 500,
  sort_field = NULL,
  sort_order = NULL,
  message = TRUE
)
```

Arguments

`criteria`	list(); the RePORTER Project API query criteria used to filter results (projects). **See Details for schema and other spec rules.**
`include_fields`	character(); optional; use to return only the specified fields from the result. **See Details for valid return field names**
`exclude_fields`	character(); optional; use to exclude specified fields from the result.
`offset`	integer(1); optional; default: 0; usually not explicitly passed by user. Used to set the start index of the results to be retrieved (indexed from 0). **See Details**.
`limit`	integer(1); optional; default: 500; restrict the number of project records returned per page/request inside the calling function. Defaulted to the maximum allowed value of 500. Reducing this may help with bandwidth/timeout issues.
`sort_field`	character(1); optional; use to sort the result by the specified field. May be useful in retrieving complete result sets above the API maximum of 10K (but below 2x the max = 20K)
`sort_order`	character(1): optional; one of "asc" or "desc"; `sort_field` must be specified.
`message`	logical(1); default: TRUE; print a message with the JSON to console/stdout. You may want to suppress this at times.

Details

The maximum number of records that can be returned from any result set is 10,000. Also, the maximum record index in the result set that can be returned is 9,999 - corresponding to the 10,000'th record in the set. These constraints from the NIH API defy any intuition that the `offset` argument would be useful to return records beyond this 10K limit. If you need to do this, you have two options:

- You can break your request into several smaller requests to be retrieved individually. For example, requesting records for one fiscal year (see: `fiscal_years`) at a time. This should be your first path
- If you have a result set between 10,001 and 20,000 records, you might try passing essentially the same request twice, but varying them by the sort order on some field (and taking care to avoid or remove overlapping results). See the `sort_field` and `sort_order` arguments.

`criteria` must be specified as a list and may include any of the following (all optional) top level elements:

- `use_relevance`: logical(1); if TRUE (default), it will sort the most closely matching records per the search criteria to the top (i.e. the NHI sorts descending according to a calculated match score)
- `fiscal_years`: numeric(); one or more fiscal years to retrieve projects that correspond to (or started in) one of the fiscal years entered
- `include_active_projects`: logical(1); if TRUE (default), adds in active projects without regard for `policy_years`
- `pi_names`: list(); API will return records with Project Investigators (PIs) wildcard-matching any of the strings requested.
  If provided, the list must contain three named character vector elements: `first_name`, `last_name`, `any_name`. Each vector must contain at least one element - use a length-1 vector with an empty string (= `""` or = `character(1)`) for any name field you do not wish to search on.
- `multi_pi_only`: `logical(1)`; default: FALSE; when multiple `pi_names` are matched, setting this value to TRUE changes the logic from returning project records associated with ANY matched PI name to those associated with ALL names.
- `po_names`: list(); Project Officers (POs), otherwise same comments as for `pi_names`
- `org_names`: character(); one or more strings to filter organization names. The **provided string is implicitly taken to include wildcards at head and tail** ends; "JOHN" and "HOP" will both match "JOHNS HOPKINS UNIVERSITY", etc.
- `org_names_exact_match`: character(); one or more strings to exactly match organization names
- `pi_profile_ids`: numeric(); one or more project investigator profile IDs; results will match projects associated with any of the IDs
- `org_cities`: character(); one or more cities in which associated organizations may be based.
- `org_states`: character(); one or more US States or Territories (note: requires the abbreviation codes: "NY", "PR", etc.) in which a project organization may be based.
- `project_nums`: character(); one or more project numbers (note: the alphanumeric variety of numbers); results will match any of the specified strings. You may include explicit wildcard operators ("*") in the strings, e.g. "5UG1HD078437-*"
- `project_num_split`: list(6); the `project_nums` can be broken down to meaningful components which can be searched individually using this argument. These component codes are defined here Your list must contain all of the following named elements:
  - `appl_type_code`: character();
  - `activity_code`: character();
  - `ic_code`: character();

  - serial_num: character();
  - support_year: character();
  - suffix_code: character();

  Provide a length-1 vector containing an empty string (="" or =character(1)) for any element you do not want to search on

- spending_categories: list(2); a list containing the following named elements:
  - values: numeric(): the NIH spending category code. These are congressionally defined and are available here
  - match_all: logical(1); TRUE to return projects found in all categories; FALSE to return projects matching any one of the categories.
- funding_mechanism: character(); one or more NIH funding mechanism codes used in the president's budget. Available here
- org_countries: character(); one or more country names; e.g. "United States"
- appl_ids: numeric(); one or more application IDs (note: appl. IDs are natural numbers, unlike project_nums)
- agencies: character(); one or more of the abbreviated NIH agency/institute/center names, available here
- is_agency_admin: logical(1); when specifying associated agencies, set this value to TRUE to further specify that these agencies are administering the grant/project.
- is_agency_funding: logical(1); when specifying associated agencies, set this value to TRUE to further specify that these agencies are funding the grant/project.
- activity_codes: character(); a 3-character code identifying the grant, contract, or intramural activity through which a project is supported. This is a more detailed description within each funding mechanism. Codes are available here
- award_types: character(); (aka Type of Application) one or more grant/application type codes numbered 1-9. See types here
- dept_types: character(); one or more of NIH standardized department type names (e.g. "PEDIATRICS"). Valid names are provided here
- cong_dists: character(); one or more US congressional districts (e.g. "NY-20") which the project can be associated with. See here
- foa: character(); one or more FOA (Funding Opportunity Announcements). Multiple projects may be tied to a single FOA. See here
- project_start_date: list(2); provide a range for the project start date. Must pass as list containing the following named elements:
  - from_date: character(1); string date in %Y-%m-%d format. See ?base::format for converting from date class.
  - to_date: character(1); string date in %Y-%m-%d format.
- project_end_date: list(2); provide a range for the project end date - similar to project_start_date.
  - from_date: character(1); string date in %Y-%m-%d format. See ?base::format for converting from date class.
  - to_date: character(1); string date in %Y-%m-%d format.

- `organization_type: character()`; one or more types of applicant organizations (e.g. "SCHOOLS OF MEDICINE"). There does not appear to be a documented list of valid values, but you can obtain one by pulling all records in a recent year and extracting unique values.
- `award_notice_date: list(2)`; the award notice date as a range, or you can provide just one of the min/max date, but if you do you must provide the other as an empty string.
  - `from_date`: character(1); string date in %Y-%m-%d format. See `?base::format` for converting from date class.
  - `to_date`: character(1); string date in %Y-%m-%d format.
- `award_amount_range: list(2)`; a numeric range - if you don't want to filter by this sub-criteria (but are filtering on some other award criteria), enter 0 for min and 1e9 for max
  - `min_amount: numeric(1)`; a real number between 0 and something very large
  - `max_amount: numeric(1)`; a real number between 0 and something very large
- `exclude_subprojects: logical(1)`; default: FALSE; related to multiproject research awards, TRUE will limit results to just the parent project.
- `sub_project_only: logical(1)`; default: FALSE; similar to `exclude_subprojects`, this field will limit results to just the subprojects, excluding the parent.
- `newly_added_projects_only: logical(1)`; default: FALSE; return only those projects "newly added" (this is left undefined in the official documentation) to the system.
- `covid_response: character()`; one or more special selector codes used to return projects awarded to study COVID-19 and related topics as funded and classified according to the below valid values/funding sources:
  - `All`: all COVID-19 projects
  - `Reg-CV`: those funded by regular NIH Appropriated funds
  - `CV`: those funded by the Coronavirus Preparedness and Response Supplemental Appropriations Act, 2020
  - `C3`: those funded by the CARES Act
  - `C4`: those funded by the Paycheck Protection Program and Health Care Enhancement Act
  - `C5`: those funded by the Coronavirus Response and Relief Supplemental Appropriations Act, 2021
  - `C6`: those funded by the American Rescue Plan Act, 2021
- `full_study_sections: list(6)`; (not documented in API notes) Review activities of the Center for Scientific Review (CSR) are organized into Integrated Review Groups (IRGs). Each IRG represents a cluster of study sections around a general scientific area. Applications generally are assigned first to an IRG, and then to a specific study section within that IRG for evaluation of scientific merit.
  This gets a bit complicated so we provide this resource for further reading. If providing this criteria, you must include each of the below named elements as character vectors:
  - `irg_code`: character(); Integrated Review Group
  - `sra_designator_code`: character(); Scientific Review Administrator
  - `sra_flex_code`: character();
  - `group_code`: character();
  - `name`: character();
  - `url`: character();

- advanced_text_search: list(3); used to perform string search in the Project Title ("projecttitle"), Abstract ("abstract"), and/or Project Terms ("terms") fields. If providing this criteria, you must include each of the below named elements:
  - operator: character(1); one of "and", "or", "advanced". "and", "or" will be the logical operator between all provided search terms. "advanced" allows the user to pass a boolean search string directly.
  - search_field: character(); can be one or multiple of "abstract", "terms", "projecttitle" passed as a vector of length 1 to 3. To search all fields, the user can alternatively pass a length 1 character vector containing the string "all" or "".
  - search_text: character(1); pass one or multiple search terms separated by spaces, without any quotations. If searching in "advanced" mode, provide a boolean search string - you may use parentheses, AND, OR, NOT, and *escaped* double quotes (e.g. search_text = "(brain AND damage) OR (\"insane in the membrane\") AND cure")

Field Names: Full listing of available field names which can be specified in include_fields, exclude_fields, and sort_field is located here

Value

A standard json (jsonlite flavor) object containing the valid JSON request string which can be passed to get_nih_data or elsewhere

Examples

```
library(repoRter.nih)

## all projects funded in the current (fiscal) year
req <- make_req()

## projects funded in 2019 through 2021
req <- make_req(criteria = list(fiscal_years = 2019:2021))

## projects funded in 2021 where the principal investigator first name is
## "Michael" or begins with "Jo"
req <- make_req(criteria =
                    list(fiscal_years = 2021,
                         pi_names =
                             list(first_name = c("Michael", "Jo*"),
                                  last_name = c(""), # must specify
                                  any_name = character(1) # same here
                                  )
                         )
                )

## all covid-related projects except those funded by American Rescue Plan
## and specify the fields to return, sorting ascending on ApplId column
req <- make_req(criteria =
                    list(covid_response = c("Reg-CV", "CV", "C3", "C4", "C5")
                    ),
                include_fields =
                    c("ApplId", "SubprojectId", "FiscalYear", "Organization",
```

```
                      "AwardAmount", "CongDist", "CovidResponse",
                      "ProjectDetailUrl"),
                sort_field = "ApplId",
                sort_order = "asc")

## using advanced_text_search with boolean search string

string <- "(head AND trauma) OR \"brain damage\" AND NOT \"psychological\""
req <- make_req(criteria =
                    list(advanced_text_search =
                         list(operator = "advanced",
                              search_field = c("terms", "abstract"),
                              search_text = string
                              )
                         )
                )
```

`nih_fields` *NIH RePORTER Field Translation*

Description

A `tibble` containing name translations between payload criteria, column selection/sorting arguments, and the result set.

Usage

```
data("nih_fields")
```

Format

A tibble with 43 rows and 5 columns:

`payload_name` the name for a data element when specified in the payload criteria of a request; NA indicates that this is not available as payload criteria (can not search/filter on).

`response_name` the name of the field returned by RePORTER (and what you will see in all cases when `flatten_result = FALSE`.

`include_name` the name of the field when specified in `include_fields`, `exclude_fields`, and `sort_field` argument.

`return_class` the class of the corresponding column in a tibble returned by `get_nih_data()`. The tibble contains nested data frames and lists of variable length vectors.

Note: when `flatten_result = TRUE`, the original field name will prefix the names of the new flattened columns. See: `jsonlite:flatten`.

References

NIH RePORTER API Documentation

Index