Package 'blsR'

October 13, 2022

Title Make Requests from the Bureau of Labor Statistics API

Version 0.4.0

Description Implements v2 of the B.L.S. API for requests of survey information
and time series data through 3-tiered API that allows users to interact with
the raw API directly, create queries through a functional interface, and
re-shape the data structures returned to fit common uses. The API definition
is located at: <https://www.bls.gov/developers/api_signature_v2.htm>.

License MIT + file LICENSE

URL https://github.com/groditi/blsR

BugReports https://github.com/groditi/blsR/issues

Encoding UTF-8

RoxygenNote 7.2.1

Imports dplyr, httr, purrr, rlang, readr, stringr

Suggests testthat (>= 3.0.0), zoo, jsonlite

Config/testthat/edition 3

Depends R (>= 3.4)

NeedsCompilation no

Author Guillermo Roditi Dominguez [aut, cre]
(<https://orcid.org/0000-0002-7127-8742>)

Maintainer Guillermo Roditi Dominguez <guillermo@newriverinvestments.com>

Repository CRAN

Date/Publication 2022-10-13 19:22:32 UTC

R topics documented:

---

blsR   *blsR: Retrieve Data From the U.S. Bureau Of Labor Statistics API*

Description

blsR provides functions for retrieving and processing data from the BLS API. The functions are divided into 5 categories: query generators, query requests, the spanning functions, result processors, and the user-friendly simplified interface.

API Key and Definition

The API key is an optional argument, but it is recommended you register for an API key and use it. Requests without a key are limited to 10 years of data per request, 25 series per query, and 25 queries per day. You can register at: https://data.bls.gov/registrationEngine/

This implementation was based on the signatures available at: https://www.bls.gov/developers/api_signature_v2.htm

The B.L.S. Frequently asked questions is available at: https://www.bls.gov/developers/api_faqs.htm

General Workflow

This package was designed with a three-step workflow in mind:

- Identify which data you would like to retrieve and create a query.

- Make an http request to execute a query (`bls_request()`)

- Modify the response data to fit the user workflow

You can customize this workflow by creating your own query objects which consist of a target URL and an optional payload as documented in the API Spec. You may also want to create a custom results processor to shape the data to suit individual needs and wrap those into a single call like `get_series_table()` does.

Query Generators

The query generators return a list suitable for passing to `bls_request()`. Most users should never need to access these functions directly but they are made available for advanced users and user-extensions.

- `query_series()` - Create a query for a single time series

- `query_n_series()` - Create a query to retrieve one or more time series and their catalog data

- `query_popular_series()` - Create a query to retrieve popular series

- `query_all_surveys()` - Create a query to retrieve all surveys

- `query_survey_info()` - Create a query to retrieve information about a survey

- `query_latest_observation()` - Create a Query to retrieve the latest observation for a time series

Query Requests

The query-requester functions will execute the query by making the API request and returning a minimally-processed response. These are likely to be the most suitable functions to use for users who want to access the raw results.

- `bls_request()` - Execute a query and return the unprocessed results

- `get_series()` - Create and execute query for a single time series

- `get_n_series()` - Create and execute a query to retrieve one or more time series and their catalog data

- `get_popular_series()` - Create and execute a query to retrieve popular series

- `get_all_surveys()` - Create and execute a query to retrieve all surveys

- `get_survey_info()` - Create and execute a query to retrieve information about a survey

- `get_latest_observation()` - Create and execute a query to retrieve the latest observation for a time series

Spanning functions

The spanning functions implement the behavior around breaking up a request that exceeds the API limits into multiple requests within the API limits and then reducing the results. Currently, spanning is only supported across time but there is plans to also support spanning across the number of series requested. These functions are low-level internal implementations and most users should never need to interact with them directly.

- `span_series_request()` - Breaks up a request into multiple queries, executes the queries, and returns the reduced results

- `span_request_queries()` - Breaks up a request into a list of queries

- `reduce_spanned_responses()` - Reduces a list of responses into one series list

Result Processors

The result-processor functions will transform the raw API response data structures into data structures more likely to be suitable for modern user workflows. The functions generally take as input the values returned by the query-requester functions and make transform the data to different formats or modify the output of another result-processor function.

- `data_as_table()` - Flatten the data list into a table

- `merge_tables()` - Merge multiple tables by period

- `tidy_periods()` - Transform periods to a more useful format

- `data_as_tidy_table()` - Flatten the data list and transform period data

- `merge_tidy_tables()` - Merge multiple tables with tidy period data

- `tidy_table_as_zoo()` - Turn a table produced by `data_as_tidy_table`, `merge_tidy_tables`, or `tidy_periods` as a `zoo` object, which can be further turned into an `xts` object

Simplified Interface

These functions simplify the query generation, execution, and response processing into a single function call, including extended request periods that have to be broken down into multiple API requests. For most common use cases these are likely to be the only functions needed.

- `get_series_table()` - Request one series and return a data table

- `get_series_tables()` - Request series and return list of data tables

- `get_n_series_table()` - Request series and return one table of values

---

bls_request   *Retrieve Data From the U.S. Bureau Of Labor Statistics API v2*

Description

`bls_request()` will execute queries against the BLS API. Queries are generated using one of the following query-generating functions: query_series(), query_n_series(), query_popular_series(), query_all_surveys(), query_survey_info(), query_latest_observation(). The result is the "Results" block as defined in the API v2 signatures at https://www.bls.gov/developers/api_signature_v2.htm

Usage

```
bls_request(
  query,
  api_key = NA,
  user_agent = "http://github.com/groditi/blsR",
  process_response = .process_response,
  ...
)
```

Arguments

query	list generated by one of the query generating functions
api_key	string, only necessary for retrieving multiple series in one request, requesting calculations, or custom time frames and catalog data
user_agent	string, optional
process_response	
	function, optional. processes the `httr` response object. The default function will return the JSON payload parsed into a list
...	further arguments will be passed to `process_response` when called

Value

a list of information returned by the API request

See Also

Other blsR-requests: get_all_surveys(), get_latest_observation(), get_n_series_table(), get_n_series(), get_popular_series(), get_series_tables(), get_series_table(), get_series(), get_survey_info(), reduce_spanned_responses(), span_series_request()

Examples

```
## Not run:
library(blsR)
uer_query <- query_series('LNS14000000') #monthly unemployment rate series
uer_results <- bls_request(uer_query) #API response

## End(Not run)
```

---

data_as_table   *Convert a list of data entries as returned by BLS API to a table*

Description

Convert a list of data entries as returned by BLS API to a table

Usage

```
data_as_table(data, parse_values = TRUE)
```

Arguments

data   a list of individual datum entries as returned by the API

parse_values   optional boolean. If set to `true` (default) it will attempt to parse the contents of
                value and cast numeric strings as numeric values. If set to `false` it will retain
                value as a column of strings.

Details

currently data_as_table is very similar to `dplyr::bind_rows()`

Value

tibble flattening data into rows for entries and columns for fields

See Also

Other blsR-utils: `data_as_tidy_table()`, `merge_tables()`, `merge_tidy_tables()`, `reduce_spanned_responses()`,
`span_request_queries()`, `span_series_request()`, `tidy_periods()`, `tidy_table_as_zoo()`

Examples

```
## Not run:
series <- get_series('LNS14000001')
table <- data_as_table(series$data)

## End(Not run)
```

---

data_as_tidy_table   *Convert a list of data entries as returned by BLS API to a table*

Description

Convert a list of data entries as returned by BLS API to a table

Usage

```
data_as_tidy_table(data, parse_values = TRUE)
```

Arguments

data   a list of individual datum entries as returned the API

parse_values   optional boolean. If set to `true` (default) it will attempt to parse the contents of
                `value` and cast numeric strings as numeric values. If set to `false` it will retain
                `value` as a column of strings.

Details

An extension of [data_as_table](#) that replaces the BLS period format by removing columns `period`
and `periodName` and adding `month` or `quarter` where appropriate.

Value

tibble flattening data into rows for entries and columns for fields

See Also

Other blsR-utils: [data_as_table](#)(), [merge_tables](#)(), [merge_tidy_tables](#)(), [reduce_spanned_responses](#)(),
[span_request_queries](#)(), [span_series_request](#)(), [tidy_periods](#)(), [tidy_table_as_zoo](#)()

Examples

```
## Not run:
series <- get_series('LNS14000001')
table <- data_as_tidy_table(series$data)

## End(Not run)
```

---

get_all_surveys   *Create and execute a query to retrieve all surveys*

Description

Create and execute a query to retrieve all surveys

Usage

```
get_all_surveys(...)
```

Arguments

...	additional arguments to pass to [bls_request()](#)

Value

a table with a survey_abbreviation and survey_name columns

See Also

[query_all_surveys](#)

Other blsR-requests: [bls_request()](#), [get_latest_observation()](#), [get_n_series_table()](#), [get_n_series()](#), [get_popular_series()](#), [get_series_tables()](#), [get_series_table()](#), [get_series()](#), [get_survey_info()](#), [reduce_spanned_responses()](#), [span_series_request()](#)

---

get_latest_observation

                         Create and execute a query to retrieve the latest observation for a
                         series

Description

Create and execute a query to retrieve the latest observation for a series

Usage

```
get_latest_observation(survey_id, ...)
```

Arguments

survey_id	BLS series ID
...	additional arguments to pass to [bls_request()](#)

Value

a datum in the form of a list

See Also

query_latest_observation

Other blsR-requests: bls_request(), get_all_surveys(), get_n_series_table(), get_n_series(), get_popular_series(), get_series_tables(), get_series_table(), get_series(), get_survey_info(), reduce_spanned_responses(), span_series_request()

---

get_n_series	*Create and execute a query to retrieve one or more time series and their catalog data*

Description

Create and execute a query to retrieve one or more time series and their catalog data

Usage

```
get_n_series(
  series_ids,
  api_key,
  start_year = NA,
  end_year = NA,
  year_limit = 20,
  span = TRUE,
  catalog = FALSE,
  calculations = FALSE,
  annualaverage = FALSE,
  aspects = FALSE,
  ...
)
```

Arguments

series_ids	a list or character vector of BLS time-series IDs. If the list items are named then the names will be used in the returned list
api_key	a required API key, available from https://data.bls.gov/registrationEngine/
start_year	numeric 4-digit year
end_year	numeric 4-digit year
year_limit	optional number of years to paginate request by. Defaults to 20, the API request cap when using API key. Requests made without an API key are capped to 10 years.

span	when set to TRUE, requests where the number of years between `start_year` and `end_year` exceed `year_limit` will be performed as multiple requests automatically
catalog	boolean. If set to TRUE, element item in the list returned may include a named item `catalog`, a named list containing descriptive information about the series. Not all series have a catalog entry available.
calculations	boolean. If set to TRUE, each element in the `data` list for each series returned may include an additional named element `calculations`, a named list containing two items, `net_changes` and `pct_changes`, each of them a named list which may include items 1, 3, 6, 12 which represent 1, 3, 6, and 12 month net changes and percent changes respectively. Not all data series will have enough data points to include these calculations.
annualaverage	boolean. If set to TRUE, each `data` list may include an additional element for a an annual average of the time series, which is usually presented as month 13 in monthly data. Not all data series support this feature.
aspects	boolean. If set to TRUE, each item in the `data` list for each series returned may include an additional named element `aspects`, which will be a named list. Not all data series support this feature.
...	additional arguments to pass to `bls_request()`

Value

a list of series results. Each element of the returned list is a named list guaranteed to have two items, `SeriesID` and `data` and optionally `catalog`. The unnamed list `data` will have 0 or more elements, each one a named list representing an observation in the time series. Each observation is guaranteed to include the elements `year`, `period`, `periodName`, `value`, and `footnotes`. Footnotes are a list of named lists. The rest are scalar values. If the the most recent observation is included, that observation will have an element named `latest` which will contain the text 'true'. If `calculations` or `aspects` were requested they will be present as named elements in each observation.

See Also

[query_n_series](#)

Other blsR-requests: [bls_request()](#), [get_all_surveys()](#), [get_latest_observation()](#), [get_n_series_table()](#), [get_popular_series()](#), [get_series_tables()](#), [get_series_table()](#), [get_series()](#), [get_survey_info()](#), [reduce_spanned_responses()](#), [span_series_request()](#)

Examples

```
## Not run:
series_ids <- list(uer.men ='LNS14000001', uer.women = 'LNS14000002')
uer_series <- get_n_series(series_ids, 'your-api-key-here' )


## End(Not run)
```

---

get_n_series_table *Retrieve multiple time series in one API request and return a single tibble*

Description

Retrieve multiple time series in one API request and return a single tibble

Usage

```
get_n_series_table(
  series_ids,
  api_key,
  start_year = NA,
  end_year = NA,
  tidy = FALSE,
  parse_values = TRUE,
  ...
)
```

Arguments

series_ids	a named list of BLS time-series IDs. If the list items are named then the names will be used in the returned list
api_key	a mandatory API key, available from <https://data.bls.gov/registrationEngine/>
start_year	optional numeric 4-digit year
end_year	optional numeric 4-digit year
tidy	optional boolean. Return will use tidy_periods() if true
parse_values	optional boolean. If set to `true` (default) it will attempt to parse the values of requested data series and cast numeric strings as numeric values. If set to `false` it will retain them as strings.
...	additional arguments to pass to get_series_tables

Value

a tibble of multiple merged time series

See Also

Other blsR-requests: bls_request(), get_all_surveys(), get_latest_observation(), get_n_series(), get_popular_series(), get_series_tables(), get_series_table(), get_series(), get_survey_info(), reduce_spanned_responses(), span_series_request()

Examples

```
## Not run:
get_n_series_table(
  list(uer.men ='LNS14000001', uer.women = 'LNS14000002'),
  'your-api-key-here',
  start_year = 2005, end_year=2006
)

## End(Not run)
```

---

get_popular_series   *Create and execute a query to retrieve popular series*

Description

Create and execute a query to retrieve popular series

Usage

```
get_popular_series(survey_id = NA, ...)
```

Arguments

survey_id	optional survey abbreviation
...	additional arguments to pass to `bls_request()`

Value

a character vector of series IDs

See Also

`query_popular_series`

Other blsR-requests: `bls_request()`, `get_all_surveys()`, `get_latest_observation()`, `get_n_series_table()`, `get_n_series()`, `get_series_tables()`, `get_series_table()`, `get_series()`, `get_survey_info()`, `reduce_spanned_responses()`, `span_series_request()`

---

get_series *Create and execute query for a single time series*

Description

Create and execute query for a single time series

Usage

```
get_series(
  series_id,
  start_year = NA,
  end_year = NA,
  year_limit = 10,
  span = TRUE,
  ...
)
```

Arguments

series_id	BLS series ID
start_year	numeric 4-digit year
end_year	numeric 4-digit year
year_limit	optional number of years to paginate request by. Defaults to 10, the API request cap when using no API key. Requests made with an API key, which can be provided in . . ., are capped to 20 years.
span	when set to TRUE, requests where the number of years between start_year and end_year exceed year_limit will be performed as multiple requests automatically
...	additional arguments to pass to bls_request()

Value

a single series result, in list form. The resulting list will have the following items:

- seriesID: a character vector of length 1 containing the series_id
- data: a list of lists containing the payload data. Each item of the list represents an observation. Each observation is a list with the following named items year, period, periodName, value, footnotes. Footnotes are a list. Additionally, the most recent observation will have an item named latest which will be marked as 'true'.

See Also

query_series

Other blsR-requests: bls_request(), get_all_surveys(), get_latest_observation(), get_n_series_table(), get_n_series(), get_popular_series(), get_series_tables(), get_series_table(), get_survey_info(), reduce_spanned_responses(), span_series_request()

Examples

```
## Not run:
series <- get_series('LNS14000001')

## End(Not run)
```

---

get_series_table   *Retrieve a single time series from BLS API and return a tibble*

Description

Retrieve a single time series from BLS API and return a tibble

Usage

```
get_series_table(
  series_id,
  api_key,
  start_year = NA,
  end_year = NA,
  year_limit = 20,
  parse_values = TRUE,
  ...
)
```

Arguments

series_id	a BLS time-series ID
api_key	a mandatory API key, available from <https://data.bls.gov/registrationEngine/>
start_year	optional numeric 4-digit year
end_year	optional numeric 4-digit year
year_limit	optional number of years to paginate request by. Defaults to 20, the API request cap when using API key. Requests made without an API key are capped to 10 years.
parse_values	optional boolean. If set to `true` (default) it will attempt to parse the contents of `value` and cast numeric strings as numeric values. If set to `false` it will keep return a `value` column of strings.
...	additional arguments to pass to get_series

Value

a tibble of observations or `NA` if the request had zero results.

See Also

Other blsR-requests: [bls_request](), [get_all_surveys](), [get_latest_observation](), [get_n_series_table](), [get_n_series](), [get_popular_series](), [get_series_tables](), [get_series](), [get_survey_info](), [reduce_spanned_responses](), [span_series_request]()

Examples

```
## Not run:
get_series_table('LNS14000001',2005,2006)

## End(Not run)
```

---

get_series_tables	*Retrieve multiple time series in one API request and return a list of tibbles*

Description

Retrieve multiple time series in one API request and return a list of tibbles

Usage

```
get_series_tables(
  series_ids,
  api_key,
  start_year = NA,
  end_year = NA,
  year_limit = 20,
  parse_values = TRUE,
  ...
)
```

Arguments

series_ids	a list or character vector of BLS time-series IDs. If the list items are named then the names will be used in the returned list
api_key	a mandatory API key, available from <https://data.bls.gov/registrationEngine/>
start_year	optional numeric 4-digit year
end_year	optional numeric 4-digit year
year_limit	optional number of years to paginate request by. Defaults to 20, the API request cap when using API key. Requests made without an API key are capped to 10 years.
parse_values	optional boolean. If set to true (default) it will attempt to parse the contents of value and cast numeric strings as numeric values. If set to false it will keep return a value column of strings.
...	additional arguments to pass to [get_n_series]()

Value

a list of tibbles. Series requests which return observations will be a tibble. Series with no observations will be NA

See Also

Other blsR-requests: `bls_request()`, `get_all_surveys()`, `get_latest_observation()`, `get_n_series_table()`, `get_n_series()`, `get_popular_series()`, `get_series_table()`, `get_series()`, `get_survey_info()`, `reduce_spanned_responses()`, `span_series_request()`

Examples

```
## Not run:
get_series_tables(
  list(uer.men ='LNS14000001', uer.women = 'LNS14000002')),
  'your-api-key-here'
)
get_series_tables(
  list(uer.men ='LNS14000001', uer.women = 'LNS14000002'),
  'your-api-key-here',
  2005,2006
)

## End(Not run)
```

---

get_survey_info   *Create and execute a query to retrieve information about a survey*

Description

Create and execute a query to retrieve information about a survey

Usage

```
get_survey_info(survey_id, ...)
```

Arguments

survey_id   survey abbreviation

...   additional arguments to pass to `bls_request()`

Value

a list of survey information

See Also

[query_survey_info](#)

Other blsR-requests: [bls_request()](#), [get_all_surveys()](#), [get_latest_observation()](#), [get_n_series_table()](#), [get_n_series()](#), [get_popular_series()](#), [get_series_tables()](#), [get_series_table()](#), [get_series()](#), [reduce_spanned_responses()](#), [span_series_request()](#)

| merge_tables | *Turn a list of one or more series into a single table of time series data* |

Description

merge_tables() turns a list of series as returned by [data_as_table()](#) into a single tibble

Usage

```
merge_tables(tables, join_by = c("year", "period"))
```

Arguments

tables	a named list of tables with matching periodicity. Mixing data with different (monthly, quarterly, annual) periodicity is unsupported. The list names will be used as column names in the output.
join_by	an optional character vector of columns to use to join tables. The result will be sorted in ascending order using these columns.

Value

tibble

See Also

Other blsR-utils: [data_as_table()](#), [data_as_tidy_table()](#), [merge_tidy_tables()](#), [reduce_spanned_responses()](#), [span_request_queries()](#), [span_series_request()](#), [tidy_periods()](#), [tidy_table_as_zoo()](#)

Examples

```
## Not run:
series_ids <- list(uer.men ='LNS14000001', uer.women = 'LNS14000002')
uer_series <- get_n_series(series_ids, 'your-api-key-here' )
uer_tables <- lapply(uer_series, function(x) data_to_table(x$data))
big_table <- merge_tables(uer_tables)

## End(Not run)
```

---

merge_tidy_tables *Turn a list of one or more series into a single table of time series data*

Description

merge_tidy_tables() turns a list of series as returned by [data_as_tidy_table()](#) into a single tibble

Usage

```
merge_tidy_tables(tidy_tables)
```

Arguments

tidy_tables a named list of tables with matching periodicity. Mixing data with different (monthly, quarterly, annual) periodicity is unsupported. The list names will be used as column names in the output.

Value

tibble

See Also

Other blsR-utils: [data_as_table()](#), [data_as_tidy_table()](#), [merge_tables()](#), [reduce_spanned_responses()](#), [span_request_queries()](#), [span_series_request()](#), [tidy_periods()](#), [tidy_table_as_zoo()](#)

---

query_all_surveys *Create a query to retrieve all surveys*

Description

Create a query to retrieve all surveys

Usage

```
query_all_surveys()
```

Value

list of query parameters

See Also

Other blsR-queries: [query_latest_observation()](#), [query_n_series()](#), [query_popular_series()](#), [query_series()](#), [query_survey_info()](#), [span_request_queries()](#)

---

query_latest_observation

Create a Query to retrieve the latest observation for a time series

---

Description

Create a Query to retrieve the latest observation for a time series

Usage

```
query_latest_observation(series_id)
```

Arguments

series_id   BLS series ID

Value

list of query parameters

See Also

Other blsR-queries: [query_all_surveys](), [query_n_series](), [query_popular_series](), [query_series](), [query_survey_info](), [span_request_queries]()

---

query_n_series   *Create a query to retrieve one or more time series and their catalog data*

Description

Create a query to retrieve one or more time series and their catalog data

Usage

```
query_n_series(
  series,
  start_year = NA,
  end_year = NA,
  catalog = FALSE,
  calculations = FALSE,
  annualaverage = FALSE,
  aspects = FALSE
)
```

Arguments

series	vector of BLS series IDs
start_year	numeric 4-digit year
end_year	numeric 4-digit year
catalog	boolean
calculations	boolean
annualaverage	boolean
aspects	boolean

Value

list of query parameters

See Also

Other blsR-queries: `query_all_surveys()`, `query_latest_observation()`, `query_popular_series()`, `query_series()`, `query_survey_info()`, `span_request_queries()`

Examples

```
a <- query_n_series(c('LNS14000001', 'LNS14000002'))
b <- query_n_series(c('LNS14000001', 'LNS14000002'), start_year = 2005, end_year=2010)
c <- query_n_series(c('LNS14000001', 'LNS14000002'), 2005, 2010)
d <- query_n_series(c('LNS14000001', 'LNS14000002'), catalog=TRUE)
```

---

query_popular_series   *Create a query to retrieve popular series*

Description

Create a query to retrieve popular series

Usage

```
query_popular_series(survey_id = NA)
```

Arguments

survey_id	string optional

Value

list of query parameters

See Also

Other blsR-queries: `query_all_surveys()`, `query_latest_observation()`, `query_n_series()`,
`query_series()`, `query_survey_info()`, `span_request_queries()`

Examples

```
popular_series_query <- query_popular_series()
popular_labor_force_series <- query_popular_series('LN')
```

---

query_series   *Create a query for a single time series*

Description

Create a query for a single time series

Usage

```
query_series(series_id, start_year = NA, end_year = NA)
```

Arguments

series_id	BLS series ID
start_year	numeric 4-digit year
end_year	numeric 4-digit year

Value

list of query parameters

See Also

Other blsR-queries: `query_all_surveys()`, `query_latest_observation()`, `query_n_series()`,
`query_popular_series()`, `query_survey_info()`, `span_request_queries()`

Examples

```
unemployment_rate_query <- query_series('LNS14000000')
unemployment_rate_query <- query_series('LNS14000000', 2005, 2010)
```

---

query_survey_info   *Create a query to retrieve information about a survey*

Description

Create a query to retrieve information about a survey

Usage

```
query_survey_info(survey_id)
```

Arguments

survey_id   BLS survey abbreviation (two letter code)

Value

list of query parameters

See Also

Other blsR-queries: `query_all_surveys()`, `query_latest_observation()`, `query_n_series()`,
`query_popular_series()`, `query_series()`, `span_request_queries()`

Examples

```
query_survey_info('LN')
```

---

reduce_spanned_responses

                     Reduce the multiple spanned responses into a list of series

---

Description

Reduce the multiple spanned responses into a list of series

Usage

```
reduce_spanned_responses(responses)
```

Arguments

responses   a list of API responses as returned by `bls_request()`

Value

series list

See Also

Other blsR-requests: bls_request(), get_all_surveys(), get_latest_observation(), get_n_series_table(), get_n_series(), get_popular_series(), get_series_tables(), get_series_table(), get_series(), get_survey_info(), span_series_request()

Other blsR-utils: data_as_table(), data_as_tidy_table(), merge_tables(), merge_tidy_tables(), span_request_queries(), span_series_request(), tidy_periods(), tidy_table_as_zoo()

---

span_request_queries   *Generate multiple queries that don't exceed a year limit*

Description

Generate multiple queries that don't exceed a year limit

Usage

```
span_request_queries(start_year, end_year, year_limit, query_fn)
```

Arguments

start_year	numeric
end_year	numeric
year_limit	numeric
query_fn	a function or closure that takes two arguments, start_year and end_year, and returns a query (see purrr::partial())

Value

a list of query objects in reverse chronological order

See Also

Other blsR-queries: query_all_surveys(), query_latest_observation(), query_n_series(), query_popular_series(), query_series(), query_survey_info()

Other blsR-utils: data_as_table(), data_as_tidy_table(), merge_tables(), merge_tidy_tables(), reduce_spanned_responses(), span_series_request(), tidy_periods(), tidy_table_as_zoo()

---

span_series_request   *Break up a long request into multiple API calls*

Description

Break up a long request into multiple API calls

Usage

```
span_series_request(start_year, end_year, year_limit, query_fn, ...)
```

Arguments

start_year	numeric
end_year	numeric
year_limit	numeric
query_fn	a function that takes two arguments, start_year and end_year and returns a query (see purrr::partial())
...	additional arguments to pass to bls_request()

Value

a list of API responses (what comes back from bls_re)

See Also

Other blsR-requests: bls_request(), get_all_surveys(), get_latest_observation(), get_n_series_table(), get_n_series(), get_popular_series(), get_series_tables(), get_series_table(), get_series(), get_survey_info(), reduce_spanned_responses()

Other blsR-utils: data_as_table(), data_as_tidy_table(), merge_tables(), merge_tidy_tables(), reduce_spanned_responses(), span_request_queries(), tidy_periods(), tidy_table_as_zoo()

---

tidy_periods   *Clean the period information returned by BLS*

Description

Clean the period information returned by BLS

Usage

```
tidy_periods(table)
```

Arguments

table	a tibble of the `data` slot in a series

Details

`tidy_periods` will return a tibble where the period and periodName columns have been deleted and replaced. Monthly periodicity data will have a new column `month` and quarterly data will have a new column `quarter`. Rows will be sorted from oldest to newest.

Value

a sorted tibble containing the period and the value

See Also

Other blsR-utils: `data_as_table()`, `data_as_tidy_table()`, `merge_tables()`, `merge_tidy_tables()`, `reduce_spanned_responses()`, `span_request_queries()`, `span_series_request()`, `tidy_table_as_zoo()`

Examples

```
## Not run:
series <- get_series('LNS14000001')
table <- data_as_table(series$data)
tidy_table <- tidy_periods(table)

## End(Not run)
```

---

tidy_table_as_zoo	*Convert a single series or n series tables into a zoo object*

Description

Convert a single series or n series tables into a zoo object

Usage

```
tidy_table_as_zoo(table, index_function = .zoo_index_function)
```

Arguments

table	a table of results
index_function	optional closure. The closure argument is the `table` and it should return a vector of values compatible with a zoo index. The default function will return a vector of `zoo::yearmon` for monthly series and `zoo::yearqtr` for quarterly or annual series.

Details

A utility function to easily convert retrieved BLS series into zoo or xts objects.

Value

a zooobject

See Also

Other blsR-utils: data_as_table(), data_as_tidy_table(), merge_tables(), merge_tidy_tables(),
reduce_spanned_responses(), span_request_queries(), span_series_request(), tidy_periods()

Examples

```
## Not run:
series <- get_series('LNS14000001')
table <- data_as_tidy_table(series$data)
zoo_obj <- tidy_table_as_zoo(table)

## End(Not run)
```

Index

27