

# Package ‘blsR’

April 11, 2022

**Title** Make Requests from the Bureau of Labor Statistics API

**Version** 0.3.0

**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](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** 7.1.2

**Imports** dplyr, httr, purrr, rlang, readr

**Suggests** testthat (>= 3.0.0), zoo, jsonlite

**Config/testthat/edition** 3

**NeedsCompilation** no

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

**Maintainer** Guillermo Roditi Dominguez <[guillermo@newriverinvestments.com](mailto:guillermo@newriverinvestments.com)>

**Repository** CRAN

**Date/Publication** 2022-04-11 06:22:29 UTC

## R topics documented:

blsR . . . . .	2
bls_request . . . . .	4
data_as_table . . . . .	5
data_as_tidy_table . . . . .	6
get_all_surveys . . . . .	7
get_latest_observation . . . . .	7

get_n_series . . . . .	8
get_n_series_table . . . . .	9
get_popular_series . . . . .	10
get_series . . . . .	11
get_series_table . . . . .	12
get_series_tables . . . . .	13
get_survey_info . . . . .	14
merge_tables . . . . .	15
merge_tidy_tables . . . . .	16
query_all_surveys . . . . .	16
query_latest_observation . . . . .	17
query_n_series . . . . .	17
query_popular_series . . . . .	18
query_series . . . . .	19
query_survey_info . . . . .	20
tidy_periods . . . . .	20
tidy_table_as_zoo . . . . .	21
<b>Index</b>	<b>23</b>

---

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 4 categories: query generators, query requests, result processors, and the user-friendly simplified interface.

## API Key and Definition

The API key is an optional parameter, 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](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](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

### 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](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

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

**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	<i>Convert a list of data entries as returned by BLS API to a table</i>
---------------	---

---

**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\(\)](#), [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\(\)](#), [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	<i>Create and execute a query to retrieve all surveys</i>
-----------------	---

---

**Description**

Create and execute a query to retrieve all surveys

**Usage**

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

**Arguments**

... additional parameters 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\(\)](#)

---

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 parameters 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\(\)](#)

---

get_n_series	<i>Create and execute a query to retrieve one or more time series and their catalog data</i>
--------------	--

---

**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, ...)
```

**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 <a href="https://data.bls.gov/registrationEngine/">https://data.bls.gov/registrationEngine/</a>
...	additional parameters to pass to <a href="#">query_n_series()</a>

**Value**

a list of series results (a list of lists). For more information on the shape of the series results see [get\\_series\(\)](#)

**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\(\)](#)

**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	<i>Retrieve multiple time series in one API request and return a single tibble</i>
--------------------	--

---

### 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 <a href="https://data.bls.gov/registrationEngine/">https://data.bls.gov/registrationEngine/</a>
start_year	optional numeric 4-digit year
end_year	optional numeric 4-digit year
tidy	optional boolean. Return will use <a href="#">tidy_periods()</a> 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 parameters to pass to <a href="#">get_series_tables</a>

### 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\(\)](#)

## 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 parameters 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\(\)](#)

---

get_series	<i>Create and execute query for a single time series</i>
------------	--

---

### Description

Create and execute query for a single time series

### Usage

```
get_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
...	additional parameters to pass to <a href="#">bls_request()</a>

### 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\(\)](#)

### Examples

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

---

get_series_table	<i>Retrieve a single time series from BLS API and return a tibble</i>
------------------	---

---

### 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 <a href="https://data.bls.gov/registrationEngine/">https://data.bls.gov/registrationEngine/</a>
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 parameters to pass to <a href="#">bls_request</a>

### Value

a tibble of observations

### 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\(\)](#)

**Examples**

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

## End(Not run)
```

---

get_series_tables	<i>Retrieve multiple time series in one API request and return a list of tibbles</i>
-------------------	--

---

**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 <a href="https://data.bls.gov/registrationEngine/">https://data.bls.gov/registrationEngine/</a>
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 parameters to pass to <a href="#">query_n_series</a>

**Value**

a list of tables

**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\(\)](#)

**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 parameters to pass to <a href="#">bls_request()</a>

**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\(\)](#)

---

merge_tables	<i>Turn a list of one or more series into a single table of time series data</i>
--------------	--

---

## 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.

## Value

tibble

## See Also

Other blsR-utils: [data\\_as\\_table\(\)](#), [data\\_as\\_tidy\\_table\(\)](#), [merge\\_tidy\\_tables\(\)](#), [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	<i>Turn a list of one or more series into a single table of time series data</i>
-------------------	--

---

**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\(\)](#), [tidy\\_periods\(\)](#), [tidy\\_table\\_as\\_zoo\(\)](#)

---

query_all_surveys	<i>Create a query to retrieve all surveys</i>
-------------------	---

---

**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\(\)](#)



---

`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\(\)](#)

---

`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\(\)](#)

**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\(\)](#)

**Examples**

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

---

query_series	<i>Create a query for a single time series</i>
--------------	--

---

**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\(\)](#)

**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\(\)](#)

**Examples**

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

---

`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\(\)](#), [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	<i>Convert a single series or n series tables into a zoo object</i>
-------------------	---

---

**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 parameter is the table and it should return a vector of values compatible with a zoo index. The default function will return a vector of <code>zoo::yearmon</code> for monthly series and <code>zoo::yearqtr</code> 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\(\)](#), [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

- \* **blsR-queries**
  - query\_all\_surveys, 16
  - query\_latest\_observation, 17
  - query\_n\_series, 17
  - query\_popular\_series, 18
  - query\_series, 19
  - query\_survey\_info, 20
- \* **blsR-requests**
  - bls\_request, 4
  - get\_all\_surveys, 7
  - get\_latest\_observation, 7
  - get\_n\_series, 8
  - get\_n\_series\_table, 9
  - get\_popular\_series, 10
  - get\_series, 11
  - get\_series\_table, 12
  - get\_series\_tables, 13
  - get\_survey\_info, 14
- \* **blsR-utils**
  - data\_as\_table, 5
  - data\_as\_tidy\_table, 6
  - merge\_tables, 15
  - merge\_tidy\_tables, 16
  - tidy\_periods, 20
  - tidy\_table\_as\_zoo, 21
- bls\_request, 4, 7–12, 14
- bls\_request(), 2, 3, 7, 10, 11, 14
- blsR, 2
- data\_as\_table, 5, 6, 15, 16, 21, 22
- data\_as\_table(), 3, 15
- data\_as\_tidy\_table, 5, 6, 15, 16, 21, 22
- data\_as\_tidy\_table(), 3, 16
- dplyr::bind\_rows(), 5
- get\_all\_surveys, 5, 7, 8–12, 14
- get\_all\_surveys(), 3
- get\_latest\_observation, 5, 7, 7, 8–12, 14
- get\_latest\_observation(), 3
- get\_n\_series, 5, 7, 8, 8, 9–12, 14
- get\_n\_series(), 3
- get\_n\_series\_table, 5, 7, 8, 9, 10–12, 14
- get\_n\_series\_table(), 4
- get\_popular\_series, 5, 7–9, 10, 11, 12, 14
- get\_popular\_series(), 3
- get\_series, 5, 7–10, 11, 12, 14
- get\_series(), 3, 8
- get\_series\_table, 5, 7–11, 12, 14
- get\_series\_table(), 3, 4
- get\_series\_tables, 5, 7–12, 13, 14
- get\_series\_tables(), 4
- get\_survey\_info, 5, 7–12, 14, 14
- get\_survey\_info(), 3
- merge\_tables, 5, 6, 15, 16, 21, 22
- merge\_tables(), 3
- merge\_tidy\_tables, 5, 6, 15, 16, 21, 22
- merge\_tidy\_tables(), 4
- query\_all\_surveys, 7, 16, 17–20
- query\_all\_surveys(), 3, 4
- query\_latest\_observation, 8, 16, 17, 18–20
- query\_latest\_observation(), 3, 4
- query\_n\_series, 8, 13, 16, 17, 17, 19, 20
- query\_n\_series(), 3, 4, 8
- query\_popular\_series, 10, 16–18, 18, 19, 20
- query\_popular\_series(), 3, 4
- query\_series, 11, 16–19, 19, 20
- query\_series(), 3, 4
- query\_survey\_info, 14, 16–19, 20
- query\_survey\_info(), 3, 4
- tidy\_periods, 5, 6, 15, 16, 20, 22
- tidy\_periods(), 3, 9
- tidy\_table\_as\_zoo, 5, 6, 15, 16, 21, 21
- tidy\_table\_as\_zoo(), 4
- zoo::yearmon, 21

zoo::yearqtr, [21](#)