

Package: epidatatools (via r-universe)

October 11, 2026

Type Package

Title Data tools used at the Economic Policy Institute

Version 1.0.5

Description Tools used by the Economic Policy Institute.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports blsR, fredr, dplyr (>= 1.0.0), haven, httr, ipumsr, jsonlite,
purrr, lubridate, MetricsWeighted, rlang, rvest, stats,
stringr, tibble, tidyr, tidysselect, utils, lifecycle

Suggests knitr, pkgdown, rmarkdown, testthat (>= 3.0.0), tidyverse

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL <https://economic.github.io/epidatatools/>,
<https://github.com/Economic/epidatatools>

BugReports <https://github.com/Economic/epidatatools/issues>

Depends R (>= 4.1)

Config/Needs/website economic/epitemplate

Config/pak/sysreqs make libicu-dev libxml2-dev libssl-dev libx11-dev zlib1g-dev

Repository <https://economic.r-universe.dev>

Date/Publication 2026-04-15 16:13:37 UTC

RemoteUrl <https://github.com/economic/epidatatools>

RemoteRef HEAD

RemoteSha 27930c4ab382f62e447ba74def048fd2859bde92

Contents

averaged_median	2
averaged_quantile	3
binipolate	4
crosstab	4
dl_ipums_micro	5
find_bls	7
find_fred	8
get_bea_nipa	9
get_bea_regional	10
get_bls	11
get_fred	12
interpolated_median	13
interpolated_quantile	14
merge_status	14
state_geocodes	15
summarize_groups	16
Index	17

averaged_median	<i>Calculate the averaged (smoothed) median</i>
-----------------	---

Description

Calculate the averaged (smoothed) median

Usage

```
averaged_median(  
  x,  
  w = NULL,  
  na.rm = TRUE,  
  quantiles_n = 9L,  
  quantiles_w = c(1:4, 5, 4:1)  
)
```

Arguments

x	numeric vector or an R object
w	numeric vector of sample weights the same length as x giving the weights to use for elements of x
na.rm	logical; if true, any NA or NaN's are removed from x before computation
quantiles_n	integer number of quantiles used for averaging; must be odd
quantiles_w	weights used for average quantiles; length must equal quantiles_n

Value

a scalar

Examples

```
averaged_median(x = mtcars$mpg)
```

averaged_quantile	<i>Calculate the averaged (smoothed) quantile</i>
-------------------	---

Description

Calculate the averaged (smoothed) quantile

Usage

```
averaged_quantile(
  x,
  w = NULL,
  probs = 0.5,
  na.rm = TRUE,
  quantiles_n = 9L,
  quantiles_w = c(1:4, 5, 4:1)
)
```

Arguments

x	numeric vector or an R object
w	numeric vector of sample weights the same length as x giving the weights to use for elements of x
probs	numeric; percentile with value [0,1]
na.rm	logical; if true, any NA or NaN's are removed from x before computation
quantiles_n	integer number of quantiles used for averaging; must be odd
quantiles_w	weights used for average quantiles; length must equal quantiles_n

Value

a numeric vector with length probs

Examples

```
averaged_quantile(x = mtcars$mpg, probs = c(0.25, 0.5, 0.75))
```

binipolate	<i>Summarize a data frame as binned interpolated percentiles</i>
------------	--

Description

[Superseded]

This function was superseded and EPI insteads uses [averaged_quantile\(\)](#) to interpolate percentiles.

Usage

```
binipolate(data, x, probs = 0.5, bin_size, .by = NULL, w = NULL)
```

Arguments

data	data frame
x	column to compute
probs	numeric vector of percentiles with values $[0, 1]$
bin_size	size of binning
.by	optional, a tidy-selection of columns for single-operation grouping
w	numeric vector of weights the same length as x giving the weights to use for elements of x

Value

a tibble or data frame

Examples

```
binipolate(mtcars, mpg, bin_size = 0.25)
binipolate(mtcars, mpg, probs = c(0.25, 0.5, 0.75), bin_size = 0.25)
binipolate(mtcars, mpg, probs = c(0.25, 0.5, 0.75), bin_size = 0.25, .by = cyl, w = wt)
```

crosstab	<i>Cross-tabulate one or two variables</i>
----------	--

Description

Cross-tabulate one or two variables

Usage

```
crosstab(data, ..., w = NULL, percent = NULL)
```

Arguments

data	a data frame
...	one or two variables, for a one- or two-way cross-tabulation
w	weight
percent	for a two-way cross-tabulation, replace counts with row or column percentages • NULL, the default, shows counts rather than percentages • "row" will replace counts with row percentages, summing to 100% across columns within each row • "column" will replace counts with column percentages, adding to 100% across rows within each column

Value

a tibble

Examples

```
crosstab(mtcars, cyl)
crosstab(mtcars, cyl, gear)
crosstab(mtcars, cyl, gear, w = mpg, percent = "column")
```

dl_ipums_micro

Download a selection of IPUMS microdata extracts

Description

Convenience functions for downloading samples and variables from **IPUMS** microdata using their **API** and package **ipumsr**.

Usage

```
dl_ipums_micro(extract)

dl_ipums_acs1(years = NULL, variables, description = NULL, ...)

dl_ipums_asec(years = NULL, variables, description = NULL, ...)

dl_ipums_cps(months = NULL, variables, description = NULL, ...)
```

Arguments

extract	an IPUMS microdata extract as defined by <code>ipumsr::define_extract_micro()</code>
years	a vector of years
variables	a vector of variable names, or a list of detailed variable specifications as created by <code>ipumsr::var_spec()</code>

description	description for the extract
...	arguments passed to <code>ipumsr::define_extract_micro()</code> other than collection, description, samples, variables
months	a vector of dates representing months of CPS samples.

Details

These functions are simply wrappers around `ipumsr` and require you to have an **IPUMS API key** saved in the `IPUMS_API_KEY` environment variable.

Value

a tibble of microdata from the IPUMS API

Functions

- `dl_ipums_micro()`: base function group
- `dl_ipums_acs1()`: Download IPUMS ACS 1-year files
- `dl_ipums_asec()`: Download IPUMS CPS ASEC
- `dl_ipums_cps()`: Download IPUMS Monthly CPS

Examples

```
# example ASEC download
dl_ipums_asec(2021:2023, c("YEAR", "OFFPOV", "ASECWT"))

# example monthly CPS download
begin_month = lubridate::ym("2022 September")
end_month = lubridate::ym("2024 August")
cps_months = seq(begin_month, end_month, by = "month")
dl_ipums_cps(cps_months, c("EARNWT", "HOURLWAGE2"))

# use dl_ipums_micro with a custom extract
extract = ipumsr::define_extract_micro(
  collection = "cps",
  description = "CPS ASEC extract",
  samples = c("cps2021_03s", "cps2022_03s", "cps2023_03s"),
  variables = c("YEAR", "OFFPOV", "ASECWT")
)
dl_ipums_micro(extract)
```

find_bls*Find BLS Series by Search String*

Description

Searches the [BLS Data Finder](#) for series matching a search string.

Usage

```
find_bls(  
  search_string,  
  max_results = 20,  
  metadata = FALSE,  
  survey = NULL,  
  seasonality = NULL  
)
```

Arguments

search_string	Character string to search for in series titles
max_results	Maximum number of results to return (default: 20)
metadata	Logical flag to retrieve catalog metadata (default: FALSE). When FALSE, only series_id and series_title are returned for faster performance.
survey	Character string specifying a BLS survey code to restrict the search (default: NULL). When NULL, searches across all surveys. See BLS Data Finder for available survey codes (e.g., "cw" for CPI Urban Wage Earners, "ln" for Labor Force Statistics, "ce" for Current Employment Statistics).
seasonality	Character string to filter by seasonal adjustment (default: NULL). When NULL, no filter is applied. Use "SA" to restrict to seasonally adjusted series, or "NSA" to restrict to not seasonally adjusted series.

Value

A tibble with columns series_id, series_title, and optionally metadata (a list column containing catalog metadata as one-row tibbles when metadata = TRUE)

Examples

```
find_bls("unemployment rate")  
  
find_bls("wage", survey = "cw")
```

find_fred

*Find FRED Series by Search String***Description**

Searches the **FRED database** for series matching a search string.

Usage

```
find_fred(
  search_string,
  max_results = 20,
  metadata = FALSE,
  seasonality = NULL,
  fred_api_key = Sys.getenv("FRED_API_KEY")
)
```

Arguments

search_string	Character string to search for in series titles
max_results	Maximum number of results to return (default: 20)
metadata	Logical flag to retrieve additional metadata (default: FALSE). When FALSE, only series_id and series_title are returned.
seasonality	Optional filter for seasonal adjustment: NULL (no filter, default), "NSA" (Not Seasonally Adjusted), or "SA" (Seasonally Adjusted)
fred_api_key	FRED API key (defaults to FRED_API_KEY environment variable)

Details

This function is a wrapper around `fredr::fredr_series_search_text()` and requires you to have an **FRED API key**.

Value

A tibble with columns series_id, series_title, and optionally metadata (a list column containing additional metadata when metadata = TRUE)

Examples

```
find_fred("unemployment rate")

find_fred("GDP", max_results = 10)

find_fred("inflation", metadata = TRUE)

find_fred("employment population ratio", seasonality = "SA")
```

get_bea_nipa

*Retrieve NIPA data from the BEA API***Description**

Retrieves National Income and Product Accounts data from the **Bureau of Economic Analysis API**. Requires a BEA API key saved in the BEA_API_KEY environment variable.

Usage

```
get_bea_nipa(
  tables,
  years,
  frequency = c("year", "quarter", "month"),
  underlying = FALSE,
  metadata = FALSE,
  bea_api_key = Sys.getenv("BEA_API_KEY")
)
```

Arguments

tables	Character vector of NIPA table names (e.g., "T10101", "T20305"). Can be a named vector to add custom names.
years	Numeric vector of years or "ALL" for all available years
frequency	Character string: "year" (annual), "quarter" (quarterly), or "month" (monthly). Can specify multiple as comma-separated string.
underlying	Logical flag to use NIUnderlyingDetail dataset instead of NIPA (default: FALSE)
metadata	Logical flag to return additional metadata columns (default: FALSE)
bea_api_key	BEA API key (defaults to BEA_API_KEY environment variable)

Value

A tibble with columns: table_name, table_description, line_number, line_description, date_frequency, date, value, and date variables (year, quarter, month) appropriate to the frequency of data returned. Annual data includes only year; quarterly data includes year and quarter; monthly data includes year, quarter, and month. If metadata = TRUE, also includes unit_mult, metric_name, cl_unit, series_code, and note_text (list column). If tables is a named vector, includes a "name" column as the first column.

Examples

```
get_bea_nipa("T10101", years = 2020:2024, frequency = "quarter")

get_bea_nipa(
  c("gdp" = "T10101", "personal_income" = "T20305"),
  years = 2023:2024,
```

```

    frequency = "year"
  )

get_bea_nipa("T10101", years = 2023, frequency = "year", metadata = TRUE)

```

get_bea_regional

Retrieve Regional data from the BEA API

Description

Retrieves regional economic data from the **Bureau of Economic Analysis API**. Requires a BEA API key saved in the BEA_API_KEY environment variable.

Usage

```

get_bea_regional(
  geo_fips,
  table_name,
  line_code,
  year = NULL,
  metadata = FALSE,
  bea_api_key = Sys.getenv("BEA_API_KEY")
)

```

Arguments

geo_fips	Character vector of geographic FIPS codes, or special values: "COUNTY" for all counties, "STATE" for all states, "MSA" for all MSAs. A single value can also be a state abbreviation (e.g., "NY", "CA").
table_name	Single table name (e.g., "CAINC1" for personal income)
line_code	Single line code specifying the statistic to retrieve, or "ALL" for all line codes
year	Optional numeric vector of years or "ALL" for all available years. Defaults to "LAST5".
metadata	Logical flag to return additional metadata columns (default: FALSE). When TRUE, also includes cl_unit, mult_unit, and note_text (list column).
bea_api_key	BEA API key (defaults to BEA_API_KEY environment variable)

Value

A tibble with columns: geo_fips, geo_name, table_name, table_description, line_number, line_description, date_frequency, date, year, value. If metadata = TRUE, also includes cl_unit, mult_unit, and note_text (list column containing all notes from the API response).

Examples

```
get_bea_regional(geo_fips = "STATE", table_name = "SAINC1", line_code = 1)

get_bea_regional(geo_fips = c("36000", "06000"), table_name = "SAINC1", line_code = 1)
```

get_bls

*Retrieve data from the BLS API***Description**

This function is simply a wrapper around **blsR** and requires you to have an **BLS API key** saved in the BLS_API_KEY environment variable.

Usage

```
get_bls(
  series,
  start,
  end,
  metadata = FALSE,
  bls_api_key = Sys.getenv("BLS_API_KEY")
)
```

Arguments

series	BLS series code
start	Start year (numeric)
end	End year (numeric)
metadata	Flag for additional metadata
bls_api_key	BLS API key (defaults to BLS_API_KEY environment variable)

Value

a tibble

Examples

```
get_bls("LNU02300060", start = 2020, end = 2024)

bls_series_ids = c(
  emp_fb_2534 = "LNU02073399",
  pop_asianmen_2554 = "LNU02332330Q",
  cpi_semi = "CUUS0000SA0",
  "LNU02300060"
)
get_bls(bls_series_ids, start = 2024, end = 2024)
```

```
complete_results = get_bls(bls_series_ids, start = 2024, end = 2024, metadata = TRUE)
complete_results

complete_results |>
  tidyr::unnest(metadata)
```

get_fred

Retrieve data from the FRED API

Description

This function is simply a wrapper around `fredr` and requires you to have an **FRED API key**.

Usage

```
get_fred(
  series,
  start = NULL,
  end = NULL,
  metadata = FALSE,
  fred_api_key = Sys.getenv("FRED_API_KEY")
)
```

Arguments

<code>series</code>	FRED series code
<code>start</code>	Start year or date (numeric year or Date object)
<code>end</code>	End year or date (numeric year or Date object)
<code>metadata</code>	Flag for additional metadata
<code>fred_api_key</code>	FRED API key (defaults to FRED_API_KEY environment variable)

Value

A tibble

Examples

```
get_fred("UNRATE")

series = c(
  gdp = "GDP",
  urate = "UNRATE"
)
get_fred(series, start = as.Date("2024-07-01"), end = 2024)

complete_results = get_fred(series, start = 2020, end = 2025, metadata = TRUE)
```

```
complete_results  
  
complete_results |>  
  tidyr::unnest(metadata)
```

interpolated_median	<i>Calculate the binned interpolated median</i>
---------------------	---

Description

#' @description [**Superseded**]

Usage

```
interpolated_median(x, bin_size, w = NULL, na.rm = TRUE)
```

Arguments

x	numeric vector or an R object
bin_size	size used for binning
w	numeric vector of weights the same length as x giving the weights to use for elements of x
na.rm	logical; if true, any NA or NaN's are removed from x before computation

Details

This function is superceded by EPI's current preferred method for interpolating medians, [averaged_median\(\)](#).

Value

numeric vector

Examples

```
interpolated_median(x = mtcars$mpg, bin_size = 0.50)
```

`interpolated_quantile` *Calculate the binned interpolated quantile*

Description

[Superseded]

This function is superseded by EPI's current preferred method for interpolating quantiles, [averaged_quantile\(\)](#).

Usage

```
interpolated_quantile(x, bin_size, probs = 0.5, w = NULL, na.rm = TRUE)
```

Arguments

<code>x</code>	numeric vector or an R object
<code>bin_size</code>	size used for binning
<code>probs</code>	numeric; percentile with value $[0, 1]$
<code>w</code>	numeric vector of weights the same length as <code>x</code> giving the weights to use for elements of <code>x</code>
<code>na.rm</code>	logical; if true, any NA or NaN's are removed from <code>x</code> before computation

Value

a numeric vector

Examples

```
interpolated_quantile(x = mtcars$mpg, bin_size = 0.50, probs = c(0.25, 0.5, 0.75))
```

`merge_status` *Join data frames and create a merge indicator*

Description

Join data frames and create a merge indicator

Usage

```
merge_status(x, y, ...)
```

```
## S3 method for class 'data.frame'
```

```
merge_status(x, y, ...)
```

Arguments

`x, y` data frames
`...` passed to `dplyr::full_join()`

Value

a merged data frame from `full_join` with an extra column `_merge`

Examples

```
library(dplyr)
merge_status(band_members, band_instruments, by = "name")
```

state_geocodes	<i>State Geographic Codes</i>
----------------	-------------------------------

Description

A dataset containing geographic codes for US states including FIPS codes, Census codes, and region/division classifications.

Usage

```
state_geocodes
```

Format

A data frame with 51 rows and 8 variables:

state_name Full state name
state_abb Two-letter state abbreviation
state_fips State FIPS code
state_census State Census code
division Census division number (1-9)
division_name Census division name (e.g., "New England", "Pacific")
region Census region number (1-4)
region_name Census region name (Northeast, Midwest, South, West)

Source

https://github.com/Economic/state_geocodes

summarize_groups	<i>Summarize distinct groups</i>
------------------	----------------------------------

Description

Summarize distinct groups

Usage

```
summarize_groups(.data, .groups, ...)
```

Arguments

<code>.data</code>	a data frame
<code>.groups</code>	grouping variables as a tidy selection specification of columns, as used in <code>dplyr::select()</code>
<code>...</code>	name-value pairs passed to <code>dplyr::summarize()</code>

Value

a tibble

Examples

```
summarize_groups(mtcars, cyl|gear|carb, median(mpg), mean(hp))
```

Index

* datasets

state_geocodes, [15](#)

averaged_median, [2](#)

averaged_median(), [13](#)

averaged_quantile, [3](#)

averaged_quantile(), [4](#), [14](#)

binipolate, [4](#)

crosstab, [4](#)

dl_ipums_acs1 (dl_ipums_micro), [5](#)

dl_ipums_asec (dl_ipums_micro), [5](#)

dl_ipums_cps (dl_ipums_micro), [5](#)

dl_ipums_micro, [5](#)

find_bls, [7](#)

find_fred, [8](#)

get_bea_nipa, [9](#)

get_bea_regional, [10](#)

get_bls, [11](#)

get_fred, [12](#)

interpolated_median, [13](#)

interpolated_quantile, [14](#)

ipumsr::define_extract_micro(), [5](#), [6](#)

ipumsr::var_spec(), [5](#)

merge_status, [14](#)

state_geocodes, [15](#)

summarize_groups, [16](#)