Package ‘tidyusmacro’

May 8, 2026

Title Downloading and Cleaning U.S. Macroeconomic Data

Version 0.1.0

Description Utilities to retrieve and tidy U.S. macroeconomic data series from public government data providers. Functions streamline access to series from the Federal Reserve Bank of St. Louis Federal Reserve Economic Data (FRED), the Bureau of Labor Statistics flat files, and the Bureau of Economic Analysis National Income and Product Accounts tables, then return consistent, tidy data frames ready for modeling and graphics. The package includes helpers for date alignment, log-linear projections, and common macro diagnostics, along with convenience plot builders for quick publication-quality charts.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

LazyData true

Imports dplyr, ggplot2, magrittr, purrr, readr, rlang, stringi, tidyr

NeedsCompilation no

Author Mike Konczal [aut, cre]

Maintainer Mike Konczal <konczal@gmail.com>

Repository CRAN

Date/Publication 2025-09-30 08:50:02 UTC

Contents

`cesDiffusionIndex` *cesDiffusionIndex Dataset*

Description

A tibble with 250 rows and 2 columns representing industry codes and corresponding industry titles.

Usage

```
cesDiffusionIndex
```

Format

A tibble with 250 rows and 2 variables:

ces_industry_code A character vector containing the industry codes (e.g., "10-11330000").

ces_industry_title A character vector containing the titles of the industries (e.g., "Logging").

Details

This dataset contains information on different industries, where each row corresponds to an industry defined by its unique code and a descriptive title. It is useful for analyses that require linking industry classifications to descriptive labels.

Source

U.S. Bureau of Labor Statistics (BLS)

Examples

```
# Load the dataset
data(cesDiffusionIndex)
```

`date_breaks_gg` *Date breaks anchored to last data month (for ggplot)*

Description

Create a breaks function for `scale_x_date()` that always includes the last actual data month and then selects every nth month counting backward.

Usage

```
date_breaks_gg(n = 6, last, decreasing = FALSE)
```

Arguments

`n`	Integer; keep every n-th month counting backward from `last`. Default 6.
`last`	Date; the last (max) date in your data. Required to ensure no break is placed after your actual data.
`decreasing`	Logical; if TRUE, return breaks in descending order. Default FALSE.

Value

A function usable in `scale_x_date(breaks = ...)`.

Examples

```
# Minimal reproducible example (avoid using the name `df`, which masks stats::df)
set.seed(1)
dat <- data.frame(
  date  = seq(as.Date("2023-01-01"), by = "month", length.out = 24),
  value = cumsum(rnorm(24))
)

library(ggplot2)

ggplot(dat, aes(date, value)) +
  geom_line() +
  scale_x_date(
    date_labels = "%b\n%Y",
    breaks = date_breaks_gg(n = 6, last = max(dat$date))
  ) +
  labs(x = NULL, y = NULL)
```

date_breaks_n *Create evenly spaced breaks*

Description

Generate a sequence of date breaks for ggplot scales, taking every nth unique date.

Usage

```
date_breaks_n(dates, n = 6, decreasing = TRUE)
```

Arguments

dates	A vector of dates.
n	Integer, keep every n-th date (default = 6).
decreasing	Logical, if TRUE (default) sorts dates in descending order.

Value

A vector of dates suitable for use as ggplot2 axis breaks.

Examples

```
library(ggplot2)
library(dplyr)

df <- tibble(
  date = seq.Date(as.Date("2020-01-01"), as.Date("2025-01-01"), by = "month"),
  value = rnorm(61)
)

ggplot(df, aes(date, value)) +
  geom_line() +
  scale_x_date(breaks = date_breaks_n(df$date, 6))
```

esp_navy *ESP Primary Color (Navy)*

Description

A standalone color value for quick use.

Usage

```
esp_navy
```

Format

An object of class `character` of length 1.

`esp_pal` *ESP Color Palette*

Description

Named vector of ESP-branded colors.

Usage

```
esp_pal
```

Format

An object of class `character` of length 3.

`esp_theme` *ESP Theme and Color Scales*

Description

Custom theme and color palette for Economic Security Project graphics.

Usage

```
theme_esp(base_family = "Public Sans")

scale_color_esp(...)

scale_fill_esp(...)

scale_colour_esp(...)
```

Arguments

`base_family`	Base font family for the theme. Defaults to "Public Sans".
`...`	Passed to the underlying ggplot2 scale functions.

Value

A ggplot2 theme or scale object.

getBLSFiles — *Download and Process BLS Time Series Files with Vectorized Date Assignment*

Description

This function downloads and processes data from the Bureau of Labor Statistics (BLS) for a given data source. It downloads several auxiliary files, merges them to enrich series metadata, downloads the main data file, and assigns dates based on the period code. For monthly data (codes "M01"–"M12") the date is set to the first day of the month; if the period is "M13", the date is set to December 31; and for quarterly data (codes "Q1"–"Q4") the date is assigned as the last day of the quarter's final month.

Usage

```
getBLSFiles(data_source, email)
```

Arguments

`data_source`	A character string specifying the data source. One of `"cpi"`, `"eci"`, `"jolts"`, `"cps"`, `"ces"`, `"averageprice"`, or `"food"`.
`email`	A character string containing your email address. This is used as the HTTP user agent when downloading files.

Value

A tibble containing the merged BLS data with an assigned date column.

Examples

```
# Download CPI data using your email address
bls_data <- getBLSFiles("cpi", "user@example.com")
```

getFRED — *Download and Merge FRED Series*

Description

A flexible wrapper that downloads one or more data series from the St. Louis Fed (FRED) API, optionally computes one-period percentage changes, and merges them into a tidy tibble keyed by `date`.

Usage

```
getFRED(..., keep_all = TRUE, rename_variables = NULL, lagged = NULL)
```

Arguments

...	One or more FRED series IDs. Each element may be either **Unnamed character string** The raw FRED ticker; column keeps the lowercase ticker name, e.g.\ "UNRATE". **Named character string** The value is the FRED ticker and the name becomes the column label, e.g.\ payroll = "PAYEMS". You may also pass a single character vector (named or unnamed) for compatibility with older code.
keep_all	Logical. TRUE (default) performs a full join that keeps all dates across series; FALSE performs an inner join.
rename_variables	Optional character vector of new column names (one per series), retained for backward compatibility. Supply *either* this argument *or* names in ..., not both.
lagged	Logical scalar or logical vector. If TRUE (or the corresponding element is TRUE), the series is replaced by its one-period percentage change $(x_t/x_{t-1}) - 1$. Recycled to match the number of series if length 1.

Details

You may supply the series in two ways:

- **Natural "..." style**: getFRED(unrate = "UNRATE", payroll = "PAYEMS"). Named arguments give friendly column names; unnamed arguments keep the (lower-case) ticker as the column name.
- **Legacy style**: pass a single (optionally named) character vector—e.g.\ c(unrate = "UNRATE", payroll = "PAYEMS")—and/or use the rename_variables= argument. This remains supported for backward compatibility.

If you provide names in ... *and* a non-NULL rename_variables vector, the function stops and prompts you to choose a single naming method.

Value

A tibble with a date column and one column per requested series.

Examples

```
# New interface
getFRED(unrate = "UNRATE", payroll = "PAYEMS")

# Multiple unnamed series (columns become 'unrate' and 'payems')
getFRED("UNRATE", "PAYEMS")
```

getNIPAFiles *Download and Process BEA NIPA Files with Fast Row Expansion*

Description

This function downloads and processes National Income and Product Accounts (NIPA) data files from the BEA website. It reads the necessary register files, formats the date column, and then uses the fast stringi functions together with tidyr's unnest() to split the combined `TableId:LineNo` field into separate rows and columns. Finally, it merges the datasets.

Usage

```
getNIPAFiles(
  location = "https://apps.bea.gov/national/Release/TXT/",
  type = "Q"
)
```

Arguments

`location`	The URL or path where the BEA files are located. Default: "https://apps.bea.gov/national/Release/TXT/".
`type`	A character string indicating the type of data to load. For example, "Q" for quarterly or "M" for monthly data. Default is "Q".

Value

A data frame containing the merged and formatted NIPA data.

Examples

```
nipadata <- getNIPAFiles(type = "Q")
```

getPCEInflation *Load and Process Personal Consumption Expenditures (PCE) Data*

Description

This function loads flat files containing various economic data at the specified frequency and processes them to compute the Personal Consumption Expenditures (PCE) series.

Usage

```
getPCEInflation(frequency = "M", NIPA_data = NULL)
```

Arguments

frequency	Character string indicating the frequency of the data. Defaults to "M" (monthly).
NIPA_data	Optional data frame. If provided, it will be used as the raw NIPA dataset instead of loading fresh data with getNIPAFiles().'

Details

It performs the following steps:

1. Loads the full dataset using load_flat_files.
2. Extracts total GDP data (from table "U20405" and series code "DPCERC").
3. Computes the PCE weight for each observation as the nominal consumption share (i.e., consumption value divided by total GDP).
4. Extracts a quantity measure from table "U20403".
5. Loads the PCE data from table "U20404", joins the computed weights and quantity data, and calculates several period-over-period growth measures.

Value

A tbl_df (data frame) containing the PCE data with calculated variables.

Examples

```
# Load monthly PCE data
pce_data <- getPCEInflation("M")
```

---

getUnrateFRED *Get Full Unemployment Rate from FRED*

Description

Downloads the civilian unemployment level and labor force level from FRED, and calculates the unemployment rate as unemploy_level/lf_level.

Usage

```
getUnrateFRED()
```

Value

A tibble with columns:

date	Observation date
unemploy_level	Civilian unemployment level (in thousands)
lf_level	Civilian labor force level (in thousands)
full_unrate	Unemployment rate (decimal)

Examples

```
getUnrateFRED()
```

`logLinearProjection` *Log-Linear Projection (data-masked, dplyr-native)*

Description

Fits a log-linear trend `log(value) ~ t` on a calibration window and projects it for rows on/after `start_date`. Designed for use inside dplyr verbs (no need to pass `.`).

Usage

```
logLinearProjection(
  date,
  value,
  start_date,
  end_date,
  group = NULL,
  data = NULL
)
```

Arguments

`date`	Bare column name for the date variable (coercible to Date).
`value`	Bare column name for the positive numeric series to project.
`start_date`	Date or string coercible to Date; start of calibration.
`end_date`	Date or string coercible to Date; end of calibration.
`group`	Optional bare column name to group by before projecting.
`data`	Optional data frame. If omitted, uses the current data mask (e.g., inside `mutate()`) via `dplyr::cur_data_all()`.

Value

A numeric vector `projection` aligned to the input rows; `NA` before `start_date`. Respects grouping if `group` is supplied.

Index