Package: tidyusmacro (via r-universe)

May 29, 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

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

LazyData true

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

Config/pak/sysreqs libicu-dev libx11-dev

Repository https://mtkonczal.r-universe.dev

Date/Publication 2026-04-29 14:54:52 UTC

RemoteUrl https://github.com/mtkonczal/tidyusmacro

RemoteRef HEAD

RemoteSha 5f7cfdf915f793231d14219fed7243261791a088

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)
```

`dallasTrimPCEcomponents` *Dallas Fed Trimmed-Mean PCE component dictionary*

Description

A tibble mapping each of the 177 components used in the Federal Reserve Bank of Dallas's trimmed-mean PCE inflation rate to its corresponding series in BEA NIPA Table 2.4.4U (Fisher price index for personal consumption expenditures by type of product, monthly).

Usage

```
dallasTrimPCEcomponents
```

Format

A tibble with 177 rows and 5 variables:

dallas_idx Integer. Ordinal position in the Dallas Fed tech-notes list.

name Character. Component name as published by the Dallas Fed.

series_code Character. BEA NIPA series code in Table 2.4.4U.

line_no Integer. Line number in BEA Table 2.4.4U.

bea_label Character. Label BEA publishes alongside `series_code`.

Details

Components are organized as durables (1-41), nondurables (42-91), services (92-177), and one NPISH aggregate (178). The 2009 tech notes (Dolmas, "Trimmed Mean PCE Inflation," updated 2022-12-23) list 178 components; BEA combined two of them - Tenant-Occupied Stationary Homes and Tenant Landlord Durables - into a single line of Table 2.4.4U (`IA000629`) when the disaggregated series stopped in December 2001. The mapping reflects that combination, yielding 177 rows; `dallas_idx` runs 1..178 with 94 omitted (93 holds the merged item).

Source

Dolmas, J. (2009, updated 2022-12-23). "PCE Inflation: Technical Note." Federal Reserve Bank of Dallas. BEA NIPA Table 2.4.4U.

References

Atkinson, T., Dolmas, J., & Zarutskie, R. (2026). "Skewness warrants caution as Trimmed Mean PCE inflation eases." Federal Reserve Bank of Dallas, April 16, 2026.

Examples

```
data(dallasTrimPCEcomponents)
head(dallasTrimPCEcomponents)
```

`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 Bureau of Labor Statistics Data*

Description

Downloads and processes data from Bureau of Labor Statistics (BLS) flat files. Supports multiple data sources including CPI, ECI, JOLTS, CPS, CES, and others. The function retrieves the main data file along with associated metadata files, merges them, and returns a tidy tibble ready for analysis.

Usage

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

Arguments

`data_source`	Character string specifying the BLS data source. Available options: `"cpi"` Consumer Price Index - current data `"eci"` Employment Cost Index (quarterly) `"cex"` Consumer Expenditure Survey `"jolts"` Job Openings and Labor Turnover Survey `"cps"` Current Population Survey `"ces"` Current Employment Statistics - all series `"ces_allemp"` Current Employment Statistics - all employees, seasonally adjusted `"ces_total"` Current Employment Statistics - total nonfarm employment `"averageprice"` Average price data - current `"food"` Average price data - food items `"se"` State and metro area employment `"su"` State and local area unemployment
`email`	Character string with your email address. Required by BLS for identifying API users. Set as the HTTP User-Agent header.

Details

The function constructs URLs to BLS flat files at https://download.bls.gov/pub/time.series/, downloads the series metadata and auxiliary lookup tables, then downloads and merges the main data file. Date parsing handles both monthly (most sources) and quarterly (ECI) data frequencies.

Value

A tibble containing the merged data with columns for:

`series_id`	Unique identifier for each data series
`date`	Observation date

value	Numeric data value
...	Additional metadata columns vary by data source (e.g., item codes, industry codes, area codes)

Examples

```
# Download CPI data
cpi_data <- getBLSFiles("cpi", "your.email@example.com")

# Download JOLTS data
jolts_data <- getBLSFiles("jolts", "your.email@example.com")
```

---

getDallasTrimPCE *Build the Dallas Fed trimmed-mean PCE component panel*

Description

Returns a long tibble with the raw inputs to the Federal Reserve Bank of Dallas's trimmed-mean PCE inflation rate: monthly Fisher price index, nominal expenditure, real quantity, monthly price change, the Fisher (t-1, t) expenditure-share weight, and a flag indicating whether the component was trimmed in that month and on which tail. Users can replicate the trimmed-mean rate by collapsing this tibble to kept (non-trimmed) components each month and taking the weight-renormalized weighted mean of price changes.

Usage

```
getDallasTrimPCE(
  frequency = "M",
  NIPA_data = NULL,
  alpha = 0.24,
  beta = 0.31,
  components = NULL
)
```

Arguments

frequency	Character. Frequency code passed to getNIPAFiles. Defaults to "M" (monthly). Currently the trimmed-mean panel is only meaningful at monthly frequency.
NIPA_data	Optional pre-loaded NIPA tibble from getNIPAFiles(). If NULL, the function downloads it.
alpha	Numeric in [0, 1]. Lower-tail trim share. Default 0.24, the Dallas Fed published value.
beta	Numeric in [0, 1]. Upper-tail trim share. Default 0.31.
components	Optional override for the component dictionary. Must be a tibble with columns dallas_idx, name, series_code, line_no. Defaults to the packaged dallasTrimPCEcomponents (177 components).

Details

Weights are Fisher-style: an unweighted average of the expenditure share evaluated at base prices `P[t-1]` with quantities `Q[t-1]` and `Q[t]`, renormalized to sum to 1 within each month. Real quantity is computed as `nominal / price` from BEA NIPA Tables 2.4.5U and 2.4.4U respectively (equivalent to Table 2.4.6U per the Dallas Fed's MATLAB note, but available across the full sample without chained-dollar gaps).

Trim assignment is the simple rank-based version: components are sorted within each month by `price_change`, cumulative weight is accumulated, and components whose running cumulative weight is below `alpha` are flagged `"lower"`, while components whose cumulative weight before adding their own contribution is at or above `1 - beta` are flagged `"upper"`. Boundary components that straddle either threshold are kept (treated as interior). The Dallas Fed's exact fractional-boundary handling enters the rate calculation itself, not this panel-builder; the resulting headline rate matches the Dallas Fed series to within a few basis points.

Months without full cross-sectional coverage (i.e., any component missing this month or last) have `weight`, `is_trimmed`, and `trim_side` set to `NA`.

Value

A `tbl_df` with one row per (date, component) and columns:

date Month observation date.

dallas_idx Component ordinal in the Dallas tech notes (1..178, 94 omitted).

name Dallas Fed component name.

series_code BEA NIPA series code (Table 2.4.4U).

line_no BEA NIPA Table 2.4.4U line number.

price Fisher price index (Table 2.4.4U).

nominal Current-dollar outlay (Table 2.4.5U).

quantity Real quantity (`nominal / price`).

price_change Period-over-period fractional change in `price`. `NA` for the first observation per component.

weight Fisher (t-1, t) expenditure-share weight, renormalized to sum to 1 within each full-coverage month. `NA` otherwise.

is_trimmed Logical. `TRUE` if the component is in either tail this month and so dropped from the trimmed mean. `NA` when the month lacks full coverage.

trim_side Character. `"lower"` or `"upper"` when trimmed; `NA` when kept (interior) or coverage incomplete.

References

Dolmas, J. (2005). "Trimmed Mean PCE Inflation." Federal Reserve Bank of Dallas Working Paper 0506.

Dolmas, J. (2009, updated 2022-12-23). "PCE Inflation: Technical Note." Federal Reserve Bank of Dallas.

Atkinson, T., Dolmas, J., & Zarutskie, R. (2026). "Skewness warrants caution as Trimmed Mean PCE inflation eases." Federal Reserve Bank of Dallas, April 16, 2026.

See Also

dallasTrimPCEcomponents, getNIPAFiles

Examples

```
# Default 24/31 Dallas Fed trim
panel <- getDallasTrimPCE()

# Replicate the monthly trimmed-mean rate (kept components, renormalized
# to sum to 1):
library(dplyr)
tm_rate <- panel |>
  dplyr::filter(!is_trimmed) |>
  dplyr::group_by(date) |>
  dplyr::summarize(
    rate = sum(price_change * weight) / sum(weight),
    .groups = "drop"
  )

# What got trimmed in the latest month, by tail and weight:
panel |>
  dplyr::filter(date == max(date), is_trimmed) |>
  dplyr::arrange(trim_side, dplyr::desc(weight)) |>
  dplyr::select(name, trim_side, weight, price_change)
```

---

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) Inflation Data*

Description

Downloads and processes BEA NIPA data to compute Personal Consumption Expenditures (PCE) price indices with weights and growth measures. This is the Federal Reserve's preferred inflation measure.

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

The function performs the following steps:

1. Loads NIPA data using `getNIPAFiles` (or uses pre-loaded data).
2. Extracts total PCE from table `"U20405"` (series code `"DPCERC"`).
3. Computes PCE component weights as the nominal consumption share (component value divided by total PCE).
4. Extracts quantity indices from table `"U20403"`.
5. Loads price indices from table `"U20404"`, joins weights and quantities, and calculates 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