Package 'inflationkit'

March 26, 2026

Title Inflation Decomposition, Core Measures, and Trend Estimation

Version 0.1.0

Description Tools for analysing inflation dynamics. Computes weighted contributions of price index components, core inflation measures (trimmed mean, weighted median, exclusion-based) following Bryan and Cecchetti (1994) <doi:10.1016/0304-3932(94)90030-2>, inflation persistence via sum-of-AR-coefficients, diffusion indices, Phillips curve estimation, breakeven inflation, and trend inflation using the Beveridge-Nelson decomposition and Hodrick-Prescott filter. All functions are pure computation and work with price data from any source.

Depends R (>= 4.1.0)

License MIT + file LICENSE

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

Imports cli (>= 3.6.0), grDevices, graphics, stats

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/charlescoverdale/inflationkit

BugReports https://github.com/charlescoverdale/inflationkit/issues

NeedsCompilation no

Author Charles Coverdale [aut, cre]

Maintainer Charles Coverdale <charlesfcoverdale@gmail.com>

Repository CRAN

Date/Publication 2026-03-26 10:40:14 UTC

Contents

`ik_breakeven` *Compute breakeven inflation rates*

Description

Calculates breakeven inflation as the spread between nominal and real (inflation-linked) bond yields. This provides a market-based measure of inflation expectations.

Usage

```
ik_breakeven(nominal_yield, real_yield, maturity = NULL)
```

Arguments

`nominal_yield`	Numeric vector. Nominal bond yields.
`real_yield`	Numeric vector. Real (inflation-linked) bond yields, same length as `nominal_yield`.
`maturity`	Numeric vector or `NULL`. Bond maturities in years. If provided, must be the same length as `nominal_yield`.

Details

Note: breakeven inflation is a simplified measure that does not account for inflation risk premium or liquidity premium. It should be interpreted as a rough proxy for inflation expectations, not a precise measure.

Value

An S3 object of class `"ik_breakeven"` with elements:

breakeven If `maturity` is provided, a data.frame with columns: maturity, nominal, real, breakeven. Otherwise, a numeric vector of breakeven rates.

maturity The maturity vector (or `NULL`).

Examples

```
# Breakeven term structure
be <- ik_breakeven(
  nominal_yield = c(4.2, 4.5, 4.8, 5.0),
  real_yield = c(1.8, 2.0, 2.3, 2.5),
  maturity = c(2, 5, 10, 30)
)
print(be)
plot(be)

# Simple breakeven (no maturity)
be2 <- ik_breakeven(nominal_yield = 4.5, real_yield = 2.0)
print(be2)
```

`ik_compare` *Compare multiple core inflation measures*

Description

Takes multiple `ik_core` objects and facilitates side-by-side comparison. Produces summary statistics and a multi-line chart of all measures.

Usage

```
ik_compare(..., labels = NULL)
```

Arguments

`...`	One or more objects of class `"ik_core"`, as returned by `ik_core()`.
`labels`	Character vector or `NULL`. Labels for each measure. If `NULL`, labels are derived from each object's method name.

Value

An S3 object of class `"ik_comparison"` with elements:

measures List of `ik_core` objects.

labels Character vector of labels.

Examples

```
data <- ik_sample_data("components")
core_tm <- ik_core(data, method = "trimmed_mean")
core_wm <- ik_core(data, method = "weighted_median")
core_ex <- ik_core(data, method = "exclusion", exclude = c("Food", "Transport"))

comp <- ik_compare(core_tm, core_wm, core_ex)
print(comp)
plot(comp)
```

`ik_core` *Compute core inflation measures*

Description

Estimates core (underlying) inflation using one of three standard methods: trimmed mean, weighted median, or exclusion-based. These measures aim to strip out transitory price movements and reveal the persistent trend.

Usage

```
ik_core(
  data,
  method = c("trimmed_mean", "weighted_median", "exclusion", "asymmetric_trim"),
  trim = 0.08,
  trim_lower = 0.24,
  trim_upper = 0.31,
  exclude = NULL,
  date_col = "date",
  item_col = "item",
  change_col = "price_change",
  weight_col = "weight"
)
```

Arguments

`data`	A data.frame containing component-level inflation data.
`method`	Character. One of `"trimmed_mean"`, `"weighted_median"`, `"exclusion"`, or `"asymmetric_trim"`.
`trim`	Numeric. Fraction to trim from each tail (for `"trimmed_mean"` only). Default `0.08` (8% symmetric trim, following the Cleveland Fed).
`trim_lower`	Numeric. Fraction to trim from the lower tail (for `"asymmetric_trim"` only). Default `0.24`, matching the Dallas Fed trimmed mean PCE.
`trim_upper`	Numeric. Fraction to trim from the upper tail (for `"asymmetric_trim"` only). Default `0.31`, matching the Dallas Fed trimmed mean PCE.
`exclude`	Character vector. Items to exclude (for `"exclusion"` only).
`date_col`	Character. Name of the date column. Default `"date"`.
`item_col`	Character. Name of the item/component column. Default `"item"`.
`change_col`	Character. Name of the price change column. Default `"price_change"`.
`weight_col`	Character. Name of the weight column. Default `"weight"`.

Value

An S3 object of class `"ik_core"` with elements:

core data.frame with columns: date, core_inflation.

method Character. The method used.

trim Numeric. The trim fraction (if applicable).

exclude Character vector. Excluded items (if applicable).

headline data.frame with columns: date, headline.

References

Bryan, M. F. and Cecchetti, S. G. (1993). "The Consumer Price Index as a Measure of Inflation." NBER Working Paper No. 4505.

Bryan, M. F. and Cecchetti, S. G. (1994). "Measuring Core Inflation." In Monetary Policy, University of Chicago Press.

Examples

```
data <- ik_sample_data("components")

# Trimmed mean (default)
core_tm <- ik_core(data, method = "trimmed_mean")
print(core_tm)

# Weighted median
core_wm <- ik_core(data, method = "weighted_median")
print(core_wm)

# Exclusion-based
core_ex <- ik_core(data, method = "exclusion", exclude = c("Food", "Transport"))
print(core_ex)
```

---

ik_decompose *Decompose inflation into weighted component contributions*

Description

Computes the weighted contribution of each CPI component to headline inflation. The contribution of item i is `weight_i * price_change_i`, and headline inflation is the sum of all contributions for each period.

Usage

```
ik_decompose(
  data,
  date_col = "date",
  item_col = "item",
  change_col = "price_change",
  weight_col = "weight"
)
```

Arguments

data	A data.frame containing component-level inflation data.
date_col	Character. Name of the date column. Default "date".
item_col	Character. Name of the item/component column. Default "item".
change_col	Character. Name of the price change column. Default "price_change".
weight_col	Character. Name of the weight column. Default "weight".

Value

An S3 object of class "ik_decomposition" with elements:

contributions data.frame with columns: date, item, weight, price_change, contribution.

headline data.frame with columns: date, headline_inflation.

Examples

```
data <- ik_sample_data("components")
decomp <- ik_decompose(data)
print(decomp)
plot(decomp)
```

---

ik_diffusion *Compute an inflation diffusion index*

Description

Measures the breadth of price increases across CPI components. For each period, computes the (weighted or unweighted) fraction of items with price changes exceeding a threshold. A diffusion index above 0.5 indicates that more than half of items are experiencing above-threshold price increases.

Usage

```
ik_diffusion(
  data,
  threshold = c("zero", "mean", "target"),
  target = 0.02,
  weighted = TRUE,
  date_col = "date",
  item_col = "item",
  change_col = "price_change",
  weight_col = "weight"
)
```

Arguments

`data`	A data.frame containing component-level inflation data.
`threshold`	Character. The threshold rule: `"zero"` (price_change > 0), `"mean"` (above the cross-sectional mean), or `"target"` (above an annualised target rate).
`target`	Numeric. Annualised inflation target (for `"target"` threshold only). Default `0.02` (2%).
`weighted`	Logical. If `TRUE` (default), compute the weighted fraction. If `FALSE`, compute the simple (unweighted) fraction.
`date_col`	Character. Name of the date column. Default `"date"`.
`item_col`	Character. Name of the item/component column. Default `"item"`.
`change_col`	Character. Name of the price change column. Default `"price_change"`.
`weight_col`	Character. Name of the weight column. Default `"weight"`.

Value

An S3 object of class `"ik_diffusion"` with elements:

diffusion data.frame with columns: date, diffusion_index.

threshold Character. The threshold type used.

weighted Logical. Whether weights were used.

Examples

```
data <- ik_sample_data("components")

# Fraction of items with rising prices
d <- ik_diffusion(data, threshold = "zero")
print(d)
plot(d)
```

`ik_forecast_eval` *Evaluate inflation forecasts*

Description

Runs standard forecast evaluation tests. The bias test (Mincer-Zarnowitz) checks whether forecasts are unbiased. The efficiency test (Nordhaus) checks whether forecast errors are autocorrelated. The Diebold-Mariano test compares predictive accuracy of two competing forecasts.

Usage

```
ik_forecast_eval(
  actual,
  forecast,
  test = c("bias", "efficiency", "dm"),
  forecast2 = NULL,
  horizon = 1L,
  alternative = c("two.sided", "less", "greater")
)
```

Arguments

`actual`	Numeric vector. Realised inflation values.
`forecast`	Numeric vector. Forecast values, same length as `actual`.
`test`	Character. One of `"bias"`, `"efficiency"`, or `"dm"`.
`forecast2`	Numeric vector or `NULL`. Second forecast series (required for `"dm"` test), same length as `actual`.
`horizon`	Integer. Forecast horizon (used for Newey-West bandwidth in the DM test). Default `1`.
`alternative`	Character. Alternative hypothesis for the DM test: `"two.sided"`, `"less"`, or `"greater"`.

Value

An S3 object of class `"ik_forecast_eval"` with elements:

test Character. The test performed.

statistic Numeric. The test statistic.

p_value Numeric. The p-value.

coefficients Named numeric vector (for bias and efficiency tests).

conclusion Character. A plain-language summary of the result.

References

Diebold, F. X. and Mariano, R. S. (1995). "Comparing Predictive Accuracy." Journal of Business and Economic Statistics, 13(3), 253-263.

Examples

```
set.seed(42)
data <- ik_sample_data("headline")
actual <- data$inflation[5:80]
forecast1 <- data$inflation[4:79] + rnorm(76, 0, 0.2)
forecast2 <- data$inflation[4:79] + rnorm(76, 0, 0.4)

# Mincer-Zarnowitz bias test
bias <- ik_forecast_eval(actual, forecast1, test = "bias")
print(bias)

# Diebold-Mariano test
dm <- ik_forecast_eval(actual, forecast1, test = "dm", forecast2 = forecast2)
print(dm)
```

---

ik_persistence *Measure inflation persistence*

Description

Estimates the degree of persistence in an inflation series using one of three methods: sum of AR coefficients, half-life, or largest autoregressive root.

Usage

```
ik_persistence(
  x,
  method = c("sum_ar", "half_life", "largest_root"),
  ar_order = NULL,
  max_order = 12L,
  ic = c("bic", "aic")
)
```

Arguments

x	Numeric vector. An inflation time series.
method	Character. One of "sum_ar", "half_life", or "largest_root".
ar_order	Integer or NULL. The AR order to fit. If NULL, order is selected automatically using the information criterion specified by ic.
max_order	Integer. Maximum AR order to consider when selecting automatically. Default 12.
ic	Character. Information criterion for order selection: "bic" or "aic". Default "bic".

Value

An S3 object of class "ik_persistence" with elements:

value Numeric. The persistence measure.

method Character. The method used.

ar_order Integer. The AR order fitted.

ar_coefficients Numeric vector. The estimated AR coefficients.

interpretation Character. A plain-language interpretation ("High persistence", "Moderate persistence", or "Low persistence").

References

Andrews, D. W. K. and Chen, H.-Y. (1994). "Approximately Median-Unbiased Estimation of Autoregressive Models." Journal of Business and Economic Statistics, 12(2), 187-204.

Marques, C. R. (2004). "Inflation Persistence: Facts or Artefacts?" ECB Working Paper No. 371.

Examples

```
data <- ik_sample_data("headline")
p <- ik_persistence(data$inflation, method = "sum_ar")
print(p)

p_hl <- ik_persistence(data$inflation, method = "half_life")
print(p_hl)
```

---

ik_phillips *Estimate a Phillips curve*

Description

Fits a Phillips curve relating inflation to an economic slack measure (output gap or unemployment rate). Supports traditional, expectations- augmented, and hybrid specifications.

Usage

```
ik_phillips(
  inflation,
  slack,
  expectations = NULL,
  type = c("traditional", "expectations_augmented", "hybrid"),
  lags = 4L,
  robust_se = FALSE
)
```

Arguments

`inflation`	Numeric vector. Inflation rate series.
`slack`	Numeric vector. Slack measure (output gap or unemployment rate), same length as `inflation`.
`expectations`	Numeric vector or `NULL`. Inflation expectations series, required for `"expectations_augmented"` and `"hybrid"` types. Must be the same length as `inflation`.
`type`	Character. Phillips curve specification: `"traditional"`, `"expectations_augmented"`, or `"hybrid"`.
`lags`	Integer. Number of lagged inflation terms to include. Default `4`.
`robust_se`	Logical or character. If `FALSE`, use OLS standard errors. If `TRUE` or `"HC1"`, compute HC1 heteroskedasticity-robust standard errors. If `"HAC"`, compute Newey-West heteroskedasticity and autocorrelation consistent standard errors with automatic bandwidth selection (Newey and West, 1994). Default `FALSE`.

Value

An S3 object of class `"ik_phillips"` with elements:

coefficients Named numeric vector of estimated coefficients.

std_errors Named numeric vector of standard errors.

p_values Named numeric vector of p-values.

r_squared Numeric. R-squared of the regression.

type Character. The Phillips curve type.

slope_estimate Numeric. The estimated slope on the slack variable.

n_obs Integer. Number of observations used.

residuals Numeric vector. Regression residuals.

Examples

```
data <- ik_sample_data("headline")
pc <- ik_phillips(data$inflation, data$unemployment, type = "traditional")
print(pc)
plot(pc)
```

`ik_sample_data` *Generate sample inflation data*

Description

Creates synthetic data for testing and demonstrating inflationkit functions. Two types are available: component-level CPI data and headline macro data.

Usage

```
ik_sample_data(type = c("components", "headline"))
```

Arguments

type	Character. Either "components" for item-level CPI data with weights and price changes, or "headline" for quarterly macro data with inflation, output gap, and unemployment.

Value

A data.frame. For "components": columns date, item, weight, price_change (120 months, 10 items, 1200 rows). For "headline": columns date, inflation, output_gap, unemployment (80 quarterly observations).

Examples

```
comp <- ik_sample_data("components")
head(comp)

macro <- ik_sample_data("headline")
head(macro)
```

---

ik_sticky_flexible *Decompose inflation into sticky and flexible components*

Description

Splits CPI components into sticky-price and flexible-price categories based on a user-provided classification, then computes separate weighted inflation measures for each group. This follows the Atlanta Fed methodology.

Usage

```
ik_sticky_flexible(
  data,
  classification,
  date_col = "date",
  item_col = "item",
  change_col = "price_change",
  weight_col = "weight"
)
```

Arguments

data	A data.frame containing component-level inflation data.
classification	A named logical vector or a data.frame. If a named logical vector, names correspond to item names and TRUE indicates sticky. If a data.frame, it must have columns item (character) and sticky (logical).
date_col	Character. Name of the date column. Default "date".
item_col	Character. Name of the item/component column. Default "item".

change_col	Character. Name of the price change column. Default "price_change".
weight_col	Character. Name of the weight column. Default "weight".

Value

An S3 object of class "ik_sticky_flex" with elements:

result data.frame with columns: date, sticky, flexible, headline.

classification Named logical vector mapping items to sticky/flexible.

References

Bils, M. and Klenow, P. J. (2004). "Some Evidence on the Importance of Sticky Prices." Journal of Political Economy, 112(5), 947-985.

Examples

```
data <- ik_sample_data("components")
# Classify items
class_vec <- c(
  Food = FALSE, Housing = TRUE, Transport = FALSE,
  Clothing = FALSE, Health = TRUE, Education = TRUE,
  Communication = TRUE, Recreation = FALSE,
  Restaurants = TRUE, Other = FALSE
)
sf <- ik_sticky_flexible(data, classification = class_vec)
print(sf)
plot(sf)
```

---

ik_trend *Estimate trend inflation*

Description

Extracts the trend component from an inflation series using one of four methods: Hodrick-Prescott filter, Beveridge-Nelson decomposition, exponential smoothing, or centred moving average.

Usage

```
ik_trend(
  x,
  method = c("hp", "beveridge_nelson", "exponential_smooth", "moving_average"),
  frequency = c("quarterly", "monthly", "annual"),
  lambda = NULL,
  window = NULL
)
```

Arguments

x	Numeric vector. An inflation time series.
method	Character. One of "hp", "beveridge_nelson", "exponential_smooth", or "moving_average".
frequency	Character. Data frequency: "quarterly" (default), "monthly", or "annual". Used to set the HP filter lambda (when lambda = NULL) and moving average window (when window = NULL).
lambda	Numeric or NULL. Smoothing parameter for the HP filter. If NULL, defaults based on frequency: 6.25 for annual, 1600 for quarterly (Hodrick and Prescott, 1997), or 14400 for monthly (Backus and Kehoe, 1992).
window	Integer or NULL. Window size for the moving average method. Defaults to 4 for quarterly, 12 for monthly, or 3 for annual.

Value

An S3 object of class "ik_trend" with elements:

trend Numeric vector. The estimated trend component.

cycle Numeric vector. The cyclical component (original minus trend).

method Character. The method used.

lambda Numeric. HP filter lambda (if applicable).

window Integer. Moving average window (if applicable).

alpha Numeric. Exponential smoothing parameter (if applicable).

original Numeric vector. The original series.

References

Hodrick, R. J. and Prescott, E. C. (1997). "Postwar U.S. Business Cycles: An Empirical Investigation." Journal of Money, Credit and Banking, 29(1), 1-16.

Ravn, M. O. and Uhlig, H. (2002). "On Adjusting the Hodrick-Prescott Filter for the Frequency of Observations." Review of Economics and Statistics, 84(2), 371-376.

Examples

```
data <- ik_sample_data("headline")
tr <- ik_trend(data$inflation, method = "hp")
print(tr)
plot(tr)

tr_ma <- ik_trend(data$inflation, method = "moving_average", window = 4)
print(tr_ma)
```

Index