Package: reviser (via r-universe)

October 10, 2026

Type Package

Title Analyzing Revisions in Real-Time Time Series Vintages

Version 0.3.0

Description Analyzes revisions in real-time time series vintages. The package converts between wide revision triangles and tidy long vintages, extracts selected releases, computes revision series, visualizes vintage paths, and summarizes revision properties such as bias, dispersion, autocorrelation, and news-noise diagnostics. It also identifies efficient releases and estimates state-space models for revision nowcasting. Methods are based on Howrey (1978) <doi:10.2307/1924972>, Jacobs and Van Norden (2011) <doi:10.1016/j.jeconom.2010.04.010>, and Kishor and Koenig (2012) <doi:10.1198/jbes.2010.08169>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

RoxygenNote 7.3.3

Roxygen list (markdown = TRUE, roclets = c (` `namespace", ` `rd", ` `srr::srr_stats_roclet"))

Imports dplyr, stats, tidyr, pillar, ggplot2, car, sandwich, systemfit, calculus, rlang, scales, tibble, lubridate, KFAS, numDeriv, withr

Suggests tsbox, testthat (>= 3.0.0), knitr, rmarkdown, purrr

Config/testthat/edition 3

VignetteBuilder knitr

URL https://docs.ropensci.org/reviser/, https://github.com/ropensci/reviser

BugReports https://github.com/ropensci/reviser/issues

Config/pak/sysreqs cmake make libicu-dev

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

Date/Publication 2026-08-31 08:15:57 UTC

RemoteUrl https://github.com/ropensci/reviser

RemoteRef v0.3.1

RemoteSha 411cd720f23bd49efaa0ef7e6998309acd5477a2

Contents

coef.revision_model . . . 3
diagnose . . . 4
diagnose.revision_summary . . . 5
fitted.revision_model . . . 6
gdp . . . 7
get_days_to_release . . . 8
get_first_efficient_release . . . 9
get_first_release . . . 11
get_fixed_release . . . 12
get_latest_release . . . 13
get_nth_release . . . 14
get_releases_by_date . . . 15
get_revision_analysis . . . 16
get_revisions . . . 19
jvn_nowcast . . . 21
kk_nowcast . . . 24
logLik.revision_model . . . 27
nobs.revision_model . . . 28
plot.revision_model . . . 29
plot.tbl_vintage . . . 30
plot_vintages . . . 31
predict.revision_model . . . 33
print.lst_efficient . . . 34
print.revision_model . . . 35
print.revision_summary . . . 36
print.tbl_vintage . . . 37
residuals.revision_model . . . 38
revision_model . . . 39
states . . . 40
summary.lst_efficient . . . 41
summary.revision_model . . . 43
summary.revision_summary . . . 44
summary.tbl_vintage . . . 45
tbl_sum.tbl_vintage . . . 46
tbl_vintage . . . 47
theme_reviser . . . 48
validate_vintages . . . 49
vcov.revision_model . . . 51

vintages_long 52
vintages_wide 53

Index **55**

`coef.revision_model` *Extract Parameter Estimates from a Revision Model*

Description

Extract Parameter Estimates from a Revision Model

Usage

```
## S3 method for class 'revision_model'
coef(object, ...)
```

Arguments

`object`	A fitted model object inheriting from revision_model, such as a `kk_model` or a `jvn_model`.
`...`	Ignored.

Value

A named numeric vector of parameter estimates.

See Also

Other revision nowcasting: `fitted.revision_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.revision_model()`, `nobs.revision_model()`, `plot.revision_model()`, `predict.revision_model()`, `print.revision_model()`, `residuals.revision_model()`, `revision_model`, `states()`, `summary.revision_model()`, `vcov.revision_model()`

Examples

```
df <- get_nth_release(
  tsbox::ts_span(
    tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
    start = "1980-01-01"
  ),
  n = 0:1
)
df <- na.omit(dplyr::select(df, -c("id", "pub_date")))
fit <- kk_nowcast(df, e = 1, model = "KK", method = "OLS")
coef(fit)
```

diagnose *Diagnose Revision Quality*

Description

Generic function to provide diagnostic summaries for revision analysis objects.

Usage

```
diagnose(object, ...)
```

Arguments

object	An object for which diagnostics are desired.
...	Additional arguments passed to methods.

Value

Method-specific diagnostic output.

See Also

Other revision analysis: diagnose.revision_summary(), get_first_efficient_release(), get_revision_analysis(), print.lst_efficient(), print.revision_summary(), summary.lst_efficient(), summary.revision_summary()

Examples

```
# Example usage with revision analysis results
df <- dplyr::select(
  get_nth_release(
    na.omit(
      tsbox::ts_pc(
        dplyr::filter(reviser::gdp, id == "US")
      )
    ),
    n = 0:3
  ),
  -"pub_date"
)

final_release <- dplyr::select(
  get_nth_release(
    na.omit(
      tsbox::ts_pc(
        dplyr::filter(reviser::gdp, id == "US")
      )
    ),
    n = "latest"
```

```
  ),
  -"pub_date"
)

# Get revision analysis results
results <- get_revision_analysis(df, final_release, degree = 5)

# Diagnose revision quality
diagnose(results)
```

`diagnose.revision_summary` *Diagnose Method for Revision Summary*

Description

Provides a quick diagnostic summary of revision quality with color-coded pass/fail indicators for key metrics.

Usage

```
## S3 method for class 'revision_summary'
diagnose(object, alpha = 0.05, ...)
```

Arguments

`object`	An object of class `revision_summary`.
`alpha`	Significance level for hypothesis tests. Default is 0.05.
`...`	Additional arguments (not used).

Value

A tibble with diagnostic results.

See Also

Other revision analysis: `diagnose()`, `get_first_efficient_release()`, `get_revision_analysis()`, `print.lst_efficient()`, `print.revision_summary()`, `summary.lst_efficient()`, `summary.revision_summary()`

Examples

```
# Example usage with revision analysis results
df <- dplyr::select(
  get_nth_release(
    na.omit(
      tsbox::ts_pc(
        dplyr::filter(reviser::gdp, id == "US")
```

```
      )
    ),
    n = 0:3
  ),
  -"pub_date"
)

final_release <- dplyr::select(
  get_nth_release(
    na.omit(
      tsbox::ts_pc(
        dplyr::filter(reviser::gdp, id == "US")
      )
    ),
    n = "latest"
  ),
  -"pub_date"
)

# Get revision analysis results
results <- get_revision_analysis(df, final_release, degree = 5)

# Diagnose revision quality
diagnose(results)
```

`fitted.revision_model` *Fitted Latent Values from a Revision Model*

Description

Returns the smoothed estimate of the model's latent signal for the in-sample periods, i.e. the revision-adjusted series. The signal is the latent efficient value for a `kk_model` and the latent true value for a `jvn_model`.

Usage

```
## S3 method for class 'revision_model'
fitted(object, ...)
```

Arguments

`object`	A fitted model object inheriting from revision_model, such as a `kk_model` or a `jvn_model`.
`...`	Ignored.

Value

A tibble with columns `time`, `estimate`, `lower` and `upper`.

See Also

Other revision nowcasting: `coef.revision_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.revision_model()`, `nobs.revision_model()`, `plot.revision_model()`, `predict.revision_model()`, `print.revision_model()`, `residuals.revision_model()`, `revision_model`, `states()`, `summary.revision_model()`, `vcov.revision_model()`

Examples

```
df <- get_nth_release(
  tsbox::ts_span(
    tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
    start = "1980-01-01"
  ),
  n = 0:1
)
df <- na.omit(dplyr::select(df, -c("id", "pub_date")))
fit <- kk_nowcast(df, e = 1, model = "KK", method = "MLE")
head(fitted(fit))
```

---

gdp *Vintages Data*

Description

A collection of real-time datasets.

Usage

```
gdp
```

Format

A `tbl_pubdate` vintages object in the long layout: a tibble carrying the class attribute `c("tbl_pubdate", "tbl_vintage", "tbl_df", "tbl", "data.frame")`, with quarterly observations and 4 variables:

time Date of the observation

pub_date Publication date of the vintage

value Numeric, real GDP (seasonally adjusted)

id Country code

Because it is a vintages object rather than a plain tibble, the generics `print()`, `summary()` and `plot()` work on it directly. See `validate_vintages()` for the data contract and tbl_vintage for the class hierarchy.

Details

- GDP: Quarterly Vintages (Billions of real dollars, seasonally adjusted)
- Timeframe: Q1 1980 - Q4 2024
- Real-Time Vintages: Q4 2002 - Q4 2024

Sources

- All the data is from the realtime database of Indergand and Leist (2014). **Countries**:
- CHE:
  - Switzerland
  - Source: SECO
- US:
  - United States
  - Sources: FRED, OECD
- EA:
  - Euro Area
  - Sources: Eurostat, OECD
- JP:
  - Japan
  - Sources: Cabinet Office (Japan), OECD

References

Indergand, R., Leist, S. A Real-Time Data Set for Switzerland. Swiss J Economics Statistics 150, 331–352 (2014). doi:10.1007/BF03399410

See Also

`validate_vintages()`, tbl_vintage

Examples

```
# Load gdp dataset
data(gdp)
head(gdp)

# It is a vintages object, so the generics dispatch on it directly.
class(gdp)
summary(gdp)
```

`get_days_to_release` *Calculate the Number of Days Between Period End and First Release*

Description

Computes the number of days between the publication date (`pub_date`) of a release and the time period (`time`) end date for each record in the dataset.

Usage

```
get_days_to_release(df)
```

Arguments

df	A data frame containing data vintages. The data frame must include the columns pub_date (publication date of the release) and time (the corresponding time period for the data).

Details

The function calculates the difference between pub_date and time for each row in the dataset. The result is expressed as the number of days between the release publication date and the corresponding time period end. If the dataset is in wide format, it will first be transformed into long format using vintages_long.

Value

A data frame with an additional column days_to_release representing the number of days between the publication date (pub_date) and the time period (time) for each release.

See Also

Other revision utilities: get_first_release(), get_fixed_release(), get_latest_release(), get_nth_release(), get_releases_by_date(), get_revisions()

Examples

```
# Example data
df <- dplyr::filter(reviser::gdp, id == "US")

# Calculate days to release
df_with_days <- get_days_to_release(df)
```

---

get_first_efficient_release

Identify the First Efficient Release in Vintage Data

Description

Identifies the first release in a sequence of vintages that is "efficient" relative to the final release. A release is deemed efficient if it satisfies specific conditions of unbiasedness and efficiency, tested using a Mincer-Zarnowitz type linear regression and hypothesis testing.

Usage

```
get_first_efficient_release(
  df,
  final_release,
  significance = 0.05,
  test_all = FALSE,
```

```
  robust = TRUE
)
```

Arguments

df	A data frame of class tbl_release containing the vintage data. It must include the columns: • time: The reference period (e.g., quarter or month). • value: The observed value for the given release. • release: The release number or identifier.
final_release	A data frame containing the final release data. This must include the columns: • time: The reference period. • value: The observed final value for the given period.
significance	A numeric value specifying the significance level for the hypothesis test (default is 0.05).
test_all	A logical value indicating whether to test all releases, even after finding the first efficient release (default is FALSE).
robust	A logical value indicating whether to use robust HAC standard errors (default is TRUE).

Details

The function performs the following steps:

1. Validates inputs and ensures both df and final_release are in the correct format.
2. Iteratively tests each release for efficiency using a linear regression model of the form:

$$final = \beta_0 + \beta_1 \cdot release_i + \epsilon$$

   The null hypothesis for efficiency is:

   - $\beta_0 = 0$ (no bias)
   - $\beta_1 = 1$ (efficiency) Uses heteroskedasticity and autocorrelation consistent (HAC) standard errors for robust hypothesis testing.

3. Stops testing when the first efficient release is found (unless test_all = TRUE).

If no efficient release is found, a warning is issued.

Value

A list of class lst_efficient with the following elements:

- e: The index of the first efficient release. (0 indexed)
- data: A long-format data frame containing the vintage data with the final release appended.
- models: A list of linear regression models fitted for each release.
- tests: A list of hypothesis test results for each release.

References

Aruoba, S. Boragan, "Revisions Are Not Well Behaved", Journal of Money, Credit and Banking, 40(2-3), 319-340, 2008.

See Also

Other revision analysis: diagnose(), diagnose.revision_summary(), get_revision_analysis(), print.lst_efficient(), print.revision_summary(), summary.lst_efficient(), summary.revision_summary()

Examples

```
# Example data
df <- get_nth_release(
  tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
  n = 0:3
)

final_release <- get_nth_release(
  tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
  n = 10
)

# Identify the first efficient release
result <- get_first_efficient_release(
  df,
  final_release,
  significance = 0.05,
  robust = FALSE
)

result <- get_first_efficient_release(df, final_release, significance = 0.05)

# Access the index of the first efficient release
result$e
```

---

get_first_release *Extract the First Data Release (Vintage)*

Description

Filters the input dataset to return the earliest release (or vintage) for each time period.

Usage

```
get_first_release(df, diagonal = FALSE)
```

Arguments

df	A data frame containing data vintages. The data frame must include the columns pub_date (publication date of the release) and time (the corresponding time period for the data).
diagonal	Logical. If TRUE, the function only returns real first releases.

Details

For each time period, the function identifies the release with the earliest publication date (pub_date). A new column release is added and labels all rows in the resulting data frame as release_0. If diagonal is set to TRUE, the function only returns the real first releases. That is historic values for which no vintages exist are not returned.

Value

A filtered data frame containing only the first release(s). The resulting data frame is assigned the class tbl_release to indicate its structure.

See Also

Other revision utilities: get_days_to_release(), get_fixed_release(), get_latest_release(), get_nth_release(), get_releases_by_date(), get_revisions()

Examples

```
# Example data
df <- dplyr::filter(reviser::gdp, id == "US")

# Get the first release for each time period
first_release <- get_first_release(df)
```

---

get_fixed_release *Extract Vintage Values from a Data Frame*

Description

Some statistical agencies make a final revision of their data after a certain period of time in a give month in the year. This function extracts values from a given month or quarter a specified number of years after the initial release.

Usage

```
get_fixed_release(df, years, month = NULL, quarter = NULL)
```

Arguments

df	A data frame containing columns pub_date (publication date) and time (observation date).
years	A single whole number of years after pub_date for which the values should be extracted.
month	An optional parameter specifying the target month as a name ("July") or an integer (7). Cannot be used with quarter. At least one of month or quarter must be supplied.
quarter	An optional parameter specifying the target quarter (1-4). Cannot be used with month. At least one of month or quarter must be supplied.

Value

A filtered data frame containing values matching the specified criteria.

See Also

Other revision utilities: get_days_to_release(), get_first_release(), get_latest_release(), get_nth_release(), get_releases_by_date(), get_revisions()

Examples

```
df <- dplyr::filter(reviser::gdp, id == "US")
dta <- get_fixed_release(df, month = "July", years = 3)
dta <- get_fixed_release(df, month = 7, years = 3)
dta <- get_fixed_release(df, quarter = 3, years = 3)
```

---

get_latest_release *Extract the Latest Data Release (Vintage)*

Description

Filters the input dataset to return the most recent release (or vintage) for each time period.

Usage

```
get_latest_release(df)
```

Arguments

df	A data frame containing data vintages. The data frame must include the columns pub_date (publication date of the release) and time (the corresponding time period for the data).

Details

For each time period, the function identifies the release with the latest publication date (`pub_date`) and adds a column `release` that labels the release as `release_N`, where `N` is the release index (zero indexed).

Value

A filtered data frame containing only the most recent release(s). The resulting data frame is assigned the class `tbl_release` to indicate its structure.

See Also

Other revision utilities: `get_days_to_release()`, `get_first_release()`, `get_fixed_release()`, `get_nth_release()`, `get_releases_by_date()`, `get_revisions()`

Examples

```
# Example data
df <- dplyr::filter(reviser::gdp, id == "US")

# Get the latest release for each time period
latest_release <- get_latest_release(df)
```

`get_nth_release` *Extract the Nth Data Release (Vintage)*

Description

Filters the input dataset to return the Nth release (or vintage) of data for each time period. The function supports selecting the first, latest, or a specific numbered release.

Usage

```
get_nth_release(df, n = 0, diagonal = FALSE)
```

Arguments

`df`	A data frame containing data vintages. The data frame must include the columns: • `pub_date` (publication date of the release) • `time` (the corresponding time period for the data).
`n`	The release number to extract. Accepts: • Non-negative integer or vector (e.g., 0 for first release, 1 for second, etc.) • `"first"` to extract the first release. • `"latest"` to extract the most recent release. Default is 0 (the first release).
`diagonal`	Logical. If `TRUE`, the function only returns real first releases.

Details

The behavior depends on the value of n:

- **Non-negative integer**: The function retrieves the Nth release for each time period (e.g., 0 = first release, 1 = second release, etc.).
- **"first"**: Retrieves the first release for each time period (via get_first_release).
- **"latest"**: Retrieves the most recent release for each time period (via get_latest_release).

Value

A filtered data frame containing only the specified release(s). The resulting data frame is assigned the class tbl_release to indicate its structure. If diagonal is set to TRUE, the function only returns the real first releases. That is historic values for which no vintages exist are not returned.

See Also

Other revision utilities: get_days_to_release(), get_first_release(), get_fixed_release(), get_latest_release(), get_releases_by_date(), get_revisions()

Examples

```
# Example data
df <- dplyr::filter(reviser::gdp, id == "US")

# Get the first release (n = 0)
first_release <- get_nth_release(df, n = 0)

# Get the latest release
latest_release <- get_nth_release(df, n = "latest")

# Get the second release (n = 1)
second_release <- get_nth_release(df, n = 1)

# Get the first and second release (n = 0:1)
releases <- get_nth_release(df, n = 0:1)
```

---

get_releases_by_date *Get Data Releases for a Specific Date*

Description

Filters the input dataset to return the releases corresponding to a specific time period (date).

Usage

```
get_releases_by_date(df, date)
```

Arguments

df	A data frame containing data vintages. The data frame must include the columns pub_date (publication date of the release) and time (the corresponding time period for the data).
date	A Date object specifying the time period (date) for which releases should be retrieved.

Details

This function filters the input data based on the specified date in the time column. The input dataset must have the pub_date and time columns, with time being the period to match against the given date. If the dataset is in wide format, it will first be transformed into long format using the helper function vintages_long.

Value

A data frame containing the releases for the specified date. The returned data frame will include the same structure as the input, filtered to only include rows matching the date in the time column.

See Also

Other revision utilities: get_days_to_release(), get_first_release(), get_fixed_release(), get_latest_release(), get_nth_release(), get_revisions()

Examples

```
# Example data
df <- dplyr::filter(reviser::gdp, id == "US")

# Get releases for a specific date
date <- as.Date("2020-04-01")
releases_on_date <- get_releases_by_date(df, date)
```

---

get_revision_analysis *Revision Analysis Summary Statistics*

Description

Calculates a comprehensive set of summary statistics and hypothesis tests for revisions between initial and final data releases.

Usage

```
get_revision_analysis(df, final_release, degree = 1, grouping_var = NULL)
```

Arguments

df	A data frame containing the initial data releases. Must include columns: • time: The time variable. • value: The observed values in the initial release. • Optionally, release (release identifier) and id (grouping variable).
final_release	A data frame containing the final release data. Must include columns: • time: The time variable (matching the initial release data). • value: The observed values in the final release.
degree	An integer between 1 and 5 specifying the level of detail for the output: 1: Default, includes information about revision size. 2: includes correlation statistics of revision. 3: includes news and noise tests. 4: includes sign switches, seasonality analysis and Theil's U. 5: Full set of all statistics and tests.
grouping_var	A character string specifying the grouping variable in the data frame. Defaults to "pub_date" or "release" if available.

Details

This function performs a variety of statistical analyses to understand the nature of revisions between the initial and final data releases. The function:

- Checks the input data for consistency and transforms it as necessary.
- Merges the initial and final release datasets by their time variable and optional grouping variables (id or release).
- Computes summary statistics such as the mean, standard deviation, and range of the revisions.
- Performs hypothesis tests for bias, efficiency, and correlation using robust methods (e.g., Newey-West standard errors).
- Includes tests for seasonality, noise, and news effects.

Key tests include:

- **Bias Tests**: Tests for the presence of mean bias and regression bias.
- **Autocorrelation and Seasonality**: Tests for serial correlation and seasonal patterns in revisions.
- **Theil's U Statistics**: Measures predictive accuracy of the initial releases relative to the final values.
- **Noise vs. News**: Differentiates between unpredictable errors (noise) and systematic adjustments (news).

The function supports grouped calculations based on the presence of id or release columns in the input.

The following statistics and tests are calculated (See the vignette vignette("revision-analysis") for more details):

- **N**: The number of observations in the group.
- **Frequency**: The inferred data frequency (e.g., 12 for monthly or 4 for quarterly data).

- **Bias (mean)**: The mean revision, testing whether revisions are systematically biased.
- **Bias (p-value)**: p-value from a t-test evaluating the significance of the mean revision.
- **Bias (robust p-value)**: Newey-West HAC robust p-value for the mean revision test.
- **Minimum**: The minimum revision in the group.
- **Maximum**: The maximum revision in the group.
- **10Q**: The 10th percentile revision.
- **Median**: The median revision.
- **90Q**: The 90th percentile revision.
- **MAR**: The mean absolute revision.
- **Std. Dev.**: The standard deviation of revisions, indicating their variability.
- **Noise/Signal**: The ratio of the standard deviation of revisions to the standard deviation of final values.
- **Correlation**: The Pearson correlation between revisions and initial values, testing the relationship.
- **Correlation (p-value)**: p-value for the significance of the correlation.
- **Autocorrelation (1st)**: The first-order autocorrelation of revisions, measuring persistence.
- **Autocorrelation (1st p-value)**: p-value for the first-order autocorrelation test.
- **Autocorrelation up to 1yr (Ljung-Box p-value)**: p-value for the Ljung-Box test for higher-order autocorrelation.
- **Theil's U1**: A normalized measure of forecast accuracy, comparing the root mean squared error (RMSE) of revisions to the RMSE of final and initial values.
- **Theil's U2**: Compares forecast changes to actual changes.
- **Seasonality (Friedman p-value)**: p-value from the Friedman test for seasonality in revisions.
- **News joint test (p-value)**: p-value for the joint news test.
- **News test Intercept**: The estimated intercept from the news test regression.
- **News test Intercept (std.err)**: The standard error of the intercept in the news test regression.
- **News test Intercept (p-value)**: p-value for the intercept in the news test regression.
- **News test Coefficient**: The estimated coefficient for the `value` in the news test regression.
- **News test Coefficient (std.err)**: The standard error of the coefficient in the news test regression.
- **News test Coefficient (p-value)**: p-value for the coefficient in the news test regression.
- **Noise joint test (p-value)**: p-value for the joint noise test.
- **Noise test Intercept**: The estimated intercept from the noise test regression.
- **Noise test Intercept (std.err)**: The standard error of the intercept in the noise test regression.
- **Noise test Intercept (p-value)**: p-value for the intercept in the noise test regression.
- **Noise test Coefficient**: The estimated coefficient for the `final_value` in the noise test regression.
- **Noise test Coefficient (std.err)**: The standard error of the coefficient in the noise test regression.
- **Noise test Coefficient (p-value)**: p-value for the coefficient in the noise test regression.
- **Fraction of correct sign**: The fraction of correct sign changes in revisions.
- **Fraction of correct growth rate change**: The fraction of correct sign changes of growth rates in revisions.

Value

A data frame with one row per grouping (if applicable) and columns for summary statistics and test results. The resulting data frame is of class `revision_summary`.

See Also

Other revision analysis: `diagnose()`, `diagnose.revision_summary()`, `get_first_efficient_release()`, `print.lst_efficient()`, `print.revision_summary()`, `summary.lst_efficient()`, `summary.revision_summary()`

Examples

```
# Example usage:
df_small <- dplyr::filter(
  reviser::gdp,
  id == "US",
  time >= as.Date("2018-01-01")
)

df <- dplyr::select(
  get_nth_release(df_small, n = 0:2),
  -"pub_date"
)

final_release <- dplyr::select(
  get_nth_release(df_small, n = "latest"),
  -"pub_date"
)

results <- get_revision_analysis(
  df,
  final_release
)
```

---

get_revisions *Calculate Revisions in Vintage Data*

Description

Computes revisions in vintage data based on specified reference points: a fixed reference date, the nth release, or a specified interval. This function allows users to analyze differences between data vintages across time.

Usage

```
get_revisions(df, interval = NULL, nth_release = NULL, ref_date = NULL)
```

Arguments

df	A data frame containing vintage data. The data frame must include at least the following columns: • pub_date: The publication date of each vintage. • time: The reference period (e.g., quarter or month). • value: The observed value for the given vintage and reference period.
interval	A positive integer specifying the lag (in periods) between vintages to compute revisions. Defaults to 1 if no other parameter is specified.
nth_release	A positive integer or "latest", specifying the release to use as a reference for revisions. If "latest", the most recent vintage is used.
ref_date	A date specifying the fixed reference publication date to compare all vintages against.

Details

The function supports three mutually exclusive methods for calculating revisions:

- **Reference date** (ref_date): Computes revisions relative to a fixed publication date.
- **Interval** (interval): Computes revisions relative to vintages published interval periods earlier.
- **Nth release** (nth_release): Computes revisions relative to the nth vintage release for each reference period.

If no method is explicitly specified, interval = 1 is used by default.

Input validation ensures that only one of ref_date, nth_release, or interval is specified.

Value

A data frame (tibble) of class tbl_revision, with the following columns:

- pub_date: The publication date of the vintage.
- time: The reference period (e.g., quarter or month).
- value: The calculated revision, i.e., the difference between the observed value and the reference value.

See Also

Other revision utilities: get_days_to_release(), get_first_release(), get_fixed_release(), get_latest_release(), get_nth_release(), get_releases_by_date()

Examples

```
# Example data
df <- dplyr::filter(reviser::gdp, id == "US")

# Calculate revisions using an interval of 1
revisions_interval <- get_revisions(df, interval = 1)
```

```
# Calculate revisions using a fixed reference date
revisions_date <- get_revisions(df, ref_date = as.Date("2023-02-01"))

# Calculate revisions relative to the nth release (2nd release)
revisions_nth <- get_revisions(df, nth_release = 1)
```

`jvn_nowcast` *Jacobs-Van Norden Model for Data Revisions*

Description

Estimate the Jacobs and Van Norden (2011) state-space model for real-time data revisions, allowing for news and noise components and optional spillovers.

Usage

```
jvn_nowcast(
  df,
  e,
  ar_order = 1,
  h = 0,
  include_news = TRUE,
  include_noise = TRUE,
  include_spillovers = FALSE,
  spillover_news = TRUE,
  spillover_noise = TRUE,
  method = "MLE",
  alpha = 0.05,
  standardize = FALSE,
  solver_options = list()
)
```

Arguments

`df`	A matrix, data frame, or single-ID vintages object. Wide data should store one vintage per column. Long-format vintages data are also accepted and are converted internally. If `df` is a matrix, or a data frame without a `time` column, a synthetic `time` index is created.
`e`	A single integer giving the number of vintages used in estimation. The function uses the first `e` vintage columns after `time`, so `e` must be greater than `0` and no larger than the number of available vintage columns.
`ar_order`	A single integer giving the autoregressive order of the latent true-value process. Must be greater than `0`.
`h`	A single integer giving the forecast horizon. Must be greater than or equal to `0`.
`include_news`	Logical; whether to include a news component.

include_noise Logical; whether to include a noise component.

include_spillovers
Logical; whether to include spillover effects.

spillover_news Logical; whether spillovers apply to the news component.

spillover_noise
Logical; whether spillovers apply to the noise component.

method Estimation method. Currently only "MLE" is supported.

alpha Significance level used for confidence intervals. Must lie in (0, 1).

standardize Logical; whether to standardize the vintage matrix before estimation using jvn_standardize(). If TRUE, scaling metadata are returned in the scale element of the output.

solver_options A named list of solver options. Valid names are trace, method, maxiter, transform_se, startvals, se_method, n_starts, seed, return_states, qml_eps, qml_score_method, qml_scale, sigma_lower, sigma_upper, kfas_init, and ic_n. Supported entries are:

- trace: integer controlling console output.
- method: optimization method; one of "L-BFGS-B", "BFGS", "Nelder-Mead", "nlminb", or "two-step".
- maxiter: maximum number of optimizer iterations.
- transform_se: logical; whether standard deviation parameters are optimized on the log scale.
- startvals: optional numeric vector of starting values. The vector must have length equal to the number of estimated parameters and must follow the internal parameter order used by jvn_param_table(): AR coefficients rho_*, sigma_e, optional sigma_nu_*, optional sigma_zeta_*, and optional spillover parameters T_nu_* and T_zeta_*.
- se_method: standard-error method; one of "hessian", "qml", or "none".
- n_starts: number of random starting points for multi-start optimization.
- seed: optional random seed used for multi-start perturbations.
- return_states: logical; whether filtered and smoothed state estimates should be returned.
- qml_eps: finite-difference step size used in QML covariance estimation.
- qml_score_method: score approximation method; "forward" or "central".
- qml_scale: scaling convention for the QML covariance; "sum", "mean", or "hc".
- sigma_lower: lower bound for standard deviations on the natural scale.
- sigma_upper: upper bound for standard deviations on the natural scale.
- kfas_init: initialization used in the KFAS filtering/smoothing stage; "stationary" or "diffuse". This affects state extraction, not the custom likelihood engine.
- ic_n: sample-size convention used for BIC; "T" for the paper convention or "Tp" for T * n_vint.

For backward compatibility, the legacy aliases score_method and score_eps are also accepted and mapped to qml_score_method and qml_eps.

Value

An object of class "jvn_model" with components:

states A tibble of filtered and smoothed state estimates. If solver_options$return_states = FALSE, this is NULL.

jvn_model_mat A list containing the state-space matrices Z, Tmat, R, H, and Q.

params A data frame of parameter estimates and standard errors.

fit The raw optimizer output returned by the selected numerical optimizer.

loglik The maximized log-likelihood.

aic Akaike information criterion.

bic Bayesian information criterion.

convergence Optimizer convergence code.

data The input data after preprocessing.

scale Scaling metadata returned by jvn_standardize() when standardize = TRUE; otherwise NULL.

se_method The standard-error method used.

cov Estimated covariance matrix of the parameter estimates, if available; otherwise NULL.

p0_regularized Logical; whether computing the Kalman filter's stationary initial-state covariance at the converged estimate required numerical regularization, which can happen for a fit close to a non-stationary boundary. NULL when solver_options$return_states = FALSE, since the initial-state covariance is then not computed. summary() reports this when TRUE.

References

Jacobs, Jan P. A. M. and Van Norden, Simon (2011). Modeling data revisions: Measurement error and dynamics of "true" values. *Journal of Econometrics*.

See Also

Other revision nowcasting: coef.revision_model(), fitted.revision_model(), kk_nowcast(), logLik.revision_model(), nobs.revision_model(), plot.revision_model(), predict.revision_model(), print.revision_model(), residuals.revision_model(), revision_model, states(), summary.revision_model(), vcov.revision_model()

Examples

```
gdp_growth <- dplyr::filter(
  tsbox::ts_pc(reviser::gdp),
  id == "EA",
  time >= min(pub_date),
  time <= as.Date("2020-01-01")
)
gdp_growth <- tidyr::drop_na(gdp_growth)
df <- get_nth_release(gdp_growth, n = 0:3)

result <- jvn_nowcast(
```

```
  df = df,
  e = 4,
  ar_order = 2,
  h = 0,
  include_news = TRUE,
  include_noise = TRUE
)
summary(result)
```

---

kk_nowcast *Generalized Kishor-Koenig Model for Nowcasting*

Description

Implements a generalized Kishor-Koenig (KK) model for nowcasting and forecasting with state-space models, allowing for multiple vintages of data, efficient estimation, and Kalman filtering and smoothing.

Usage

```
kk_nowcast(
  df,
  e,
  h = 0,
  model = "Kishor-Koenig",
  method = "MLE",
  alpha = 0.05,
  solver_options = list()
)
```

Arguments

df	A data frame or single-ID vintages object in either long or wide format. Long-format vintages data are converted internally. After preprocessing, the data must contain a `time` column and at least `e + 1` releases.
e	An integer indicating the number of data vintages to include in the model. Must be greater than 0.
h	An integer specifying the forecast horizon. Default is 0, which implies no forecasts. Must be greater than or equal to 0.
model	A string specifying the type of model to use. Options are: • "Kishor-Koenig" or "KK" (default): Full Kishor-Koenig model. • "Howrey": Howrey's simplified framework. • "Classical": Classical model without vintage effects.

`method` A string specifying the estimation method to use. Options are "MLE" (Maximum Likelihood, default), "SUR", and "OLS".

`alpha` Significance level for confidence intervals (default = 0.05).

`solver_options` A named list controlling the SUR and MLE routines. Valid names are `trace`, `maxiter`, `startvals`, `solvtol`, `gradtol`, `steptol`, `transform_se`, `method`, `se_method`, `n_starts`, `seed`, `return_states`, `qml_eps`, `qml_score_method`, `qml_scale`, `sigma_lower`, `sigma_upper`, and `ic_n`. Supported entries are:

- `trace`: integer controlling console output.
- `maxiter`: maximum number of optimizer iterations.
- `startvals`: optional numeric vector of starting values. Unnamed vectors are interpreted in the canonical internal order returned by `kk_matrices(e, model, type = "character")$params`. Named vectors are reordered to that same canonical order.
- `solvtol`: tolerance passed to `systemfit::nlsystemfit()` in the SUR estimator.
- `gradtol`: gradient tolerance passed to `systemfit::nlsystemfit()` in the SUR estimator.
- `steptol`: step-length tolerance passed to `systemfit::nlsystemfit()` in the SUR estimator.
- `transform_se`: logical; whether variance parameters are optimized on the log scale in MLE.
- `method`: optimization method for MLE; one of `"L-BFGS-B"`, `"BFGS"`, `"Nelder-Mead"`, `"nlminb"`, or `"two-step"`.
- `se_method`: standard-error method for MLE; one of `"hessian"`, `"qml"`, or `"none"`.
- `n_starts`: number of random starting points used by the MLE multi-start routine.
- `seed`: optional random seed used for the MLE multi-start perturbations.
- `return_states`: logical; whether filtered and smoothed states and the fitted KFAS model should be returned.
- `qml_eps`: finite-difference step size used in QML covariance estimation.
- `qml_score_method`: score approximation method; `"forward"` or `"central"`.
- `qml_scale`: scaling convention for the QML covariance; `"sum"`, `"mean"`, or `"hc"`.
- `sigma_lower`: lower bound for variance parameters on the natural scale when `transform_se = TRUE`.
- `sigma_upper`: upper bound for variance parameters on the natural scale when `transform_se = TRUE`.
- `ic_n`: sample-size convention used for BIC; `"T"` or `"Tp"`.

For backward compatibility, the legacy aliases `score_method` and `score_eps` are also accepted and mapped to `qml_score_method` and `qml_eps`.

Details

The function supports multiple models, including the full Kishor-Koenig framework, Howrey's model, and a classical approach. It handles data preprocessing, estimation of system equations using Seemingly Unrelated Regressions (SUR), and application of the Kalman filter. This is the first

openly available implementation of the Kishor-Koenig model (See the vignette `vignette("nowcasting_revisions")` for more details).

Value

A list with the following components:

states A tibble containing filtered and smoothed state estimates. If `solver_options$return_states = FALSE`, this is `NULL`.

kk_model_mat A list of KK model matrices, such as transition and observation matrices.

ss_model_mat A list of state-space model matrices derived from the KK model.

model The fitted `KFAS::SSModel` object. If `solver_options$return_states = FALSE`, this is `NULL`.

params Estimated model parameters with standard errors.

fit The raw fit object returned by the selected estimator.

loglik Log-likelihood value for MLE fits; otherwise `NULL`.

aic Akaike information criterion for MLE fits; otherwise `NULL`.

bic Bayesian information criterion for MLE fits; otherwise `NULL`.

convergence Convergence status.

e The number of the efficient release (0-indexed).

data The input data after preprocessing to wide format.

se_method The standard-error method used by MLE; otherwise `NULL`.

cov Estimated covariance matrix of the parameter estimates, if available; otherwise `NULL`.

References

Kishor, N. Kundan and Koenig, Evan F., "VAR Estimation and Forecasting When Data Are Subject to Revision", Journal of Business and Economic Statistics, 2012.

See Also

Other revision nowcasting: `coef.revision_model()`, `fitted.revision_model()`, `jvn_nowcast()`, `logLik.revision_model()`, `nobs.revision_model()`, `plot.revision_model()`, `predict.revision_model()`, `print.revision_model()`, `residuals.revision_model()`, `revision_model`, `states()`, `summary.revision_model()`, `vcov.revision_model()`

Examples

```
# Example usage:
df <- get_nth_release(
  tsbox::ts_span(
    tsbox::ts_pc(
      dplyr::filter(reviser::gdp, id == "US")
    ),
    start = "1980-01-01"
  ),
```

```
  n = 0:1
)
df <- dplyr::select(df, -c("id", "pub_date"))
df <- na.omit(df)

e <- 1 # Number of efficient release
h <- 2 # Forecast horizon
result <- kk_nowcast(df, e, h = h, model = "Kishor-Koenig")

# Fitted models are reached through the extractor generics inherited from
# the parent class, not by indexing into the object.
coef(result)
logLik(result)
```

`logLik.revision_model` *Extract the Log-Likelihood of a Revision Model*

Description

The returned object carries the degrees of freedom and effective number of observations used by the model, so `stats::AIC()` and `stats::BIC()` reproduce the values reported by `summary()`.

Usage

```
## S3 method for class 'revision_model'
logLik(object, ...)
```

Arguments

`object`	A fitted model object inheriting from revision_model, such as a `kk_model` or a `jvn_model`.
`...`	Ignored.

Value

An object of class `logLik`.

See Also

Other revision nowcasting: `coef.revision_model()`, `fitted.revision_model()`, `jvn_nowcast()`, `kk_nowcast()`, `nobs.revision_model()`, `plot.revision_model()`, `predict.revision_model()`, `print.revision_model()`, `residuals.revision_model()`, `revision_model`, `states()`, `summary.revision_model()`, `vcov.revision_model()`

Examples

```
df <- get_nth_release(
  tsbox::ts_span(
    tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
    start = "1980-01-01"
  ),
  n = 0:1
)
df <- na.omit(dplyr::select(df, -c("id", "pub_date")))
fit <- kk_nowcast(df, e = 1, model = "KK", method = "MLE")
logLik(fit)
AIC(fit)
BIC(fit)
```

---

nobs.revision_model *Number of Observations Used to Fit a Revision Model*

Description

Returns the effective number of observations behind the reported information criteria. Under the default `ic_n = "Tp"` this is the number of time periods times the number of releases (KK) or vintages (JVN) modeled.

Usage

```
## S3 method for class 'revision_model'
nobs(object, ...)
```

Arguments

`object`	A fitted model object inheriting from revision_model, such as a `kk_model` or a `jvn_model`.
`...`	Ignored.

Value

A single integer.

See Also

Other revision nowcasting: `coef.revision_model()`, `fitted.revision_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.revision_model()`, `plot.revision_model()`, `predict.revision_model()`, `print.revision_model()`, `residuals.revision_model()`, `revision_model`, `states()`, `summary.revision_model()`, `vcov.revision_model()`

Examples

```
df <- get_nth_release(
  tsbox::ts_span(
    tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
    start = "1980-01-01"
  ),
  n = 0:1
)
df <- na.omit(dplyr::select(df, -c("id", "pub_date")))
fit <- kk_nowcast(df, e = 1, model = "KK", method = "MLE")
nobs(fit)
```

---

plot.revision_model *Plot Revision Model Results*

Description

Plot filtered or smoothed estimates for a selected state from a fitted revision model. Defined once for the parent class revision_model and inherited by `kk_model` and `jvn_model` objects alike; the state shown when `state` is not given is chosen by the concrete class.

Usage

```
## S3 method for class 'revision_model'
plot(x, state = NULL, type = "filtered", ...)
```

Arguments

`x`	A fitted model object inheriting from revision_model, such as a `kk_model` or a `jvn_model`.
`state`	String. The name of the state to visualize. If `NULL`, the family's default state is used: the latent true value for a `jvn_model`, and the first available state for a `kk_model`.
`type`	String. Type of estimate: "filtered" or "smoothed".
`...`	Additional arguments passed to theme_reviser.

Details

This method requires the state estimates to be available. A model fitted with `solver_options$return_states = FALSE` did not retain them, and plotting it fails with a message naming that option, in the same way `states()`, `fitted()`, `residuals()` and `predict()` do.

Value

A `ggplot2` object.

See Also

Other revision nowcasting: `coef.revision_model()`, `fitted.revision_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.revision_model()`, `nobs.revision_model()`, `predict.revision_model()`, `print.revision_model()`, `residuals.revision_model()`, `revision_model`, `states()`, `summary.revision_model()`, `vcov.revision_model()`

Examples

```
df <- get_nth_release(
  tsbox::ts_span(
    tsbox::ts_pc(
      dplyr::filter(reviser::gdp, id == "US")
    ),
    start = "1980-01-01"
  ),
  n = 0:1
)
df <- dplyr::select(df, -c("id", "pub_date"))
df <- na.omit(df)

e <- 1 # Number of efficient release
h <- 2 # Forecast horizon
result <- kk_nowcast(df, e, h = h, model = "Kishor-Koenig")

plot(result)
```

`plot.tbl_vintage` *Plot Method for Vintages Data*

Description

Plots a vintages object along the dimension implied by its class: publication date for a `tbl_pubdate`, release number for a `tbl_release`. Defined once for the parent class tbl_vintage and inherited by both. `plot_vintages()` remains the entry point when the dimension, title or plot type need to be set explicitly.

Usage

```
## S3 method for class 'tbl_vintage'
plot(x, ...)
```

Arguments

`x`	An object inheriting from tbl_vintage, such as a `tbl_pubdate` or a `tbl_release`.
`...`	Additional arguments passed to `plot_vintages()`.

Value

A ggplot2 object.

See Also

Other revision graphs: `plot_vintages()`, `theme_reviser()`

Examples

```
df <- dplyr::filter(reviser::gdp, id == "US")
plot(df)

release_data <- get_nth_release(df, n = 0:5)
plot(release_data)
```

---

plot_vintages *Plot Vintages Data*

Description

A flexible function to visualize vintage data using various plot types such as line plots, point plots, bar plots, or boxplots. The function ensures that input data is validated and appropriately transformed before plotting.

Usage

```
plot_vintages(
  df,
  type = "line",
  dim_col = "pub_date",
  time_col = "time",
  title = "",
  subtitle = "",
  ylab = ""
)
```

Arguments

`df`	A data frame containing the vintage data to be plotted. Must include at least two columns: one for time (`time`) and one for value (`value`).
`type`	A character string specifying the type of plot to create. Options are: • "line": Line plot (default). • "point": Scatter plot. • "bar": Bar plot. • "boxplot": Boxplot.

dim_col	A character string specifying the column name in df that represents publication dates or other grouping dimensions (e.g. "release"). Defaults to "pub_date".
time_col	A character string specifying the column name in df that represents the time variable. Defaults to "time".
title	A character string specifying the title of the plot. Defaults to an empty string.
subtitle	A character string specifying the subtitle of the plot. Defaults to an empty string.
ylab	A character string specifying the label for the y-axis. Defaults to an empty string.

Details

The plot_vintages function is designed to handle data frames in both wide and long formats. It ensures that the provided data frame includes the necessary columns for plotting. If the dim_col column contains more than 30 unique values, only the most recent 30 are plotted. Additionally, the function supports custom themes and color scales using scale_color_reviser, scale_fill_reviser, and theme_reviser.

The function raises an error if:

- The type argument is not one of "line", "point", "bar", or "boxplot".
- The specified dim_col is not a column in df.
- title, subtitle, or ylab are not character strings.

Value

A ggplot2 plot object representing the specified vintage data visualization.

See Also

theme_reviser(), scale_color_reviser(), scale_fill_reviser()

Other revision graphs: plot.tbl_vintage(), theme_reviser()

Examples

```
# Example data
df <- data.frame(
  time = rep(seq.Date(from = as.Date("2022-01-01"),
  by = "month", length.out = 12), 3),
  value = runif(36, 50, 100),
  pub_date = rep(c("2022-01-05", "2022-02-07", "2022-03-03"), each = 12)
)

# Line plot
plot_vintages(
  df,
  type = "line",
  dim_col = "pub_date",
  title = "Line plot",
  subtitle = "Randomly generated data"
  )
```

```
# Point plot
plot_vintages(
  df,
  type = "point",
  dim_col = "pub_date",
  title = "Scatter plot",
  subtitle = "Randomly generated data"
  )

# Bar plot
plot_vintages(
  df,
  type = "bar",
  dim_col = "pub_date",
  title = "Bar plot",
  subtitle = "Randomly generated data"
  )

# Boxplot
plot_vintages(
  df,
  type = "boxplot",
  dim_col = "pub_date",
  title = "Boxplot",
  subtitle = "Randomly generated data"
  )
```

`predict.revision_model` *Forecasts from a Revision Model*

Description

Returns the out-of-sample estimates of the model's latent signal produced by the forecast horizon `h` supplied to `kk_nowcast()` or `jvn_nowcast()`. The horizon is fixed at estimation time, so refit with a different `h` to change it.

Usage

```
## S3 method for class 'revision_model'
predict(object, ...)
```

Arguments

`object`	A fitted model object inheriting from revision_model, such as a `kk_model` or a `jvn_model`.
`...`	Ignored.

Value

A tibble with columns `time`, `estimate`, `lower` and `upper`. Has zero rows when the model was fitted with `h = 0`.

See Also

Other revision nowcasting: `coef.revision_model()`, `fitted.revision_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.revision_model()`, `nobs.revision_model()`, `plot.revision_model()`, `print.revision_model()`, `residuals.revision_model()`, `revision_model`, `states()`, `summary.revision_model()`, `vcov.revision_model()`

Examples

```
df <- get_nth_release(
  tsbox::ts_span(
    tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
    start = "1980-01-01"
  ),
  n = 0:1
)
df <- na.omit(dplyr::select(df, -c("id", "pub_date")))
fit <- kk_nowcast(df, e = 1, h = 2, model = "KK", method = "MLE")
predict(fit)
```

---

print.lst_efficient *Print Method for Efficient Release Results*

Description

Print Method for Efficient Release Results

Usage

```
## S3 method for class 'lst_efficient'
print(x, ...)
```

Arguments

`x`	An object of class `lst_efficient`.
`...`	Additional arguments (not used).

Value

The function returns the input `x` invisibly.

See Also

Other revision analysis: `diagnose()`, `diagnose.revision_summary()`, `get_first_efficient_release()`, `get_revision_analysis()`, `print.revision_summary()`, `summary.lst_efficient()`, `summary.revision_summary()`

Examples

```
df <- get_nth_release(
  tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
  n = 0:3
)

final_release <- get_nth_release(
  tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
  n = 10
)

result <- get_first_efficient_release(df, final_release, significance = 0.05)
print(result)
```

---

print.revision_model *Print Method for Revision Models*

Description

Default print method for objects inheriting from revision_model. Dispatches to summary.revision_model() for a consistent console display.

Usage

```
## S3 method for class 'revision_model'
print(x, ...)
```

Arguments

x	A fitted model object inheriting from revision_model, such as a kk_model or a jvn_model.
...	Additional arguments passed to summary.revision_model().

Value

The input x, invisibly.

See Also

Other revision nowcasting: coef.revision_model(), fitted.revision_model(), jvn_nowcast(), kk_nowcast(), logLik.revision_model(), nobs.revision_model(), plot.revision_model(), predict.revision_model(), residuals.revision_model(), revision_model, states(), summary.revision_model(), vcov.revision_model()

Examples

```
df <- get_nth_release(
  tsbox::ts_span(
    tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
    start = "1980-01-01"
  ),
  n = 0:1
)
df <- na.omit(dplyr::select(df, -c("id", "pub_date")))
fit <- kk_nowcast(df, e = 1, h = 2, model = "Kishor-Koenig", method = "MLE")
fit
```

---

print.revision_summary

Print Method for Revision Summary

Description

Print Method for Revision Summary

Usage

```
## S3 method for class 'revision_summary'
print(x, interpretation = TRUE, digits = 3, ...)
```

Arguments

x	An object of class revision_summary.
interpretation	Logical. If TRUE, provides interpretation of key statistics. Default is TRUE.
digits	Integer. Number of digits to display. Default is 3.
...	Additional arguments (not used).

Value

The function returns the input x invisibly.

See Also

Other revision analysis: diagnose(), diagnose.revision_summary(), get_first_efficient_release(), get_revision_analysis(), print.lst_efficient(), summary.lst_efficient(), summary.revision_summary()

Examples

```
df <- dplyr::select(
  get_nth_release(
    na.omit(
      tsbox::ts_pc(
        dplyr::filter(reviser::gdp, id == "US")
      )
    ),
    n = 0:3
  ),
  -"pub_date"
)

final_release <- dplyr::select(
  get_nth_release(
    na.omit(
      tsbox::ts_pc(
        dplyr::filter(reviser::gdp, id == "US")
      )
    ),
    n = "latest"
  ),
  -"pub_date"
)

results <- get_revision_analysis(df, final_release, degree = 1)
print(results)

# Print without interpretation
print(results, interpretation = FALSE)
```

`print.tbl_vintage` *Print Method for Vintages Data*

Description

Print method for objects inheriting from tbl_vintage. Delegates to the tibble print method, which calls `tbl_sum.tbl_vintage()` to generate the custom header.

Usage

```
## S3 method for class 'tbl_vintage'
print(x, ...)
```

Arguments

`x`	An object inheriting from tbl_vintage, such as a `tbl_pubdate` or a `tbl_release`.
`...`	Additional arguments passed to the next print method.

Value

The input x is returned invisibly.

See Also

Other helpers: summary.tbl_vintage(), tbl_sum.tbl_vintage(), tbl_vintage, validate_vintages(), vintages_long(), vintages_wide()

Examples

```
df <- dplyr::filter(reviser::gdp, id == "US")
release_data <- get_nth_release(df, n = 0:3)
print(release_data)
```

---

residuals.revision_model

Residuals of a Revision Model

Description

Difference between the observed target release and the smoothed estimate of the model's latent signal. These are measurement residuals of the release the model treats as its target – the efficient release for a kk_model, the most mature release included in the estimation for a jvn_model – not one-step-ahead prediction errors.

Usage

```
## S3 method for class 'revision_model'
residuals(object, ...)
```

Arguments

object	A fitted model object inheriting from revision_model, such as a kk_model or a jvn_model.
...	Ignored.

Value

A tibble with columns time and residual.

See Also

Other revision nowcasting: coef.revision_model(), fitted.revision_model(), jvn_nowcast(), kk_nowcast(), logLik.revision_model(), nobs.revision_model(), plot.revision_model(), predict.revision_model(), print.revision_model(), revision_model, states(), summary.revision_model(), vcov.revision_model()

Examples

```
df <- get_nth_release(
  tsbox::ts_span(
    tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
    start = "1980-01-01"
  ),
  n = 0:1
)
df <- na.omit(dplyr::select(df, -c("id", "pub_date")))
fit <- kk_nowcast(df, e = 1, model = "KK", method = "MLE")
head(residuals(fit))
```

---

revision_model *Fitted Revision Models*

Description

`reviser` represents every fitted revision-nowcasting model as an S3 object that inherits from the common parent class `revision_model`. The two concrete classes are `kk_nowcast()`, which returns a `kk_model`, and `jvn_nowcast()`, which returns a `jvn_model`; both carry `c("<family>_model", "revision_model", "list")` as their class attribute.

The parent class holds everything the two families share. The standard extractor generics `coef()`, `vcov()`, `logLik()`, `nobs()`, `fitted()`, `residuals()`, `predict()` and the `reviser` generic `states()`, together with `print()`, `summary()` and `plot()`, are defined once for `revision_model` and inherited by both families. Only the handful of behaviors that genuinely differ between the families are dispatched separately, through the internal generics `model_family()`, `spec_lines()`, `signal_state()`, `target_column()` and `default_plot_state()`.

A fitted object is a list with at least the components `params` (a data frame with columns `Parameter`, `Estimate` and `Std.Error`), `states` (a long tibble of state estimates, or `NULL` when the model was fitted with `return_states = FALSE`), `loglik`, `n_param`, `n_ic`, `cov` and `data`. A new model family becomes a full citizen of this system by returning an object with those components, prepending `"revision_model"` to its class attribute, and supplying methods for the five internal generics above.

Value

This topic documents a class rather than a function. `kk_nowcast()` returns a list of the components described above with class attribute `c("kk_model", "revision_model", "list")`, and `jvn_nowcast()` returns the same with `"jvn_model"` in place of `"kk_model"`.

See Also

`kk_nowcast()`, `jvn_nowcast()`, `states()`

Other revision nowcasting: `coef.revision_model()`, `fitted.revision_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.revision_model()`, `nobs.revision_model()`, `plot.revision_model()`, `predict.revision_model()`, `print.revision_model()`, `residuals.revision_model()`, `states()`, `summary.revision_model()`, `vcov.revision_model()`

Examples

```
df <- get_nth_release(
  tsbox::ts_span(
    tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
    start = "1980-01-01"
  ),
  n = 0:1
)
df <- na.omit(dplyr::select(df, -c("id", "pub_date")))
fit <- kk_nowcast(df, e = 1, model = "KK", method = "OLS")

# The fitted object carries the shared parent class.
class(fit)
inherits(fit, "revision_model")

# The extractor generics are inherited from that parent.
head(coef(fit))
head(states(fit))
```

`states` *Extract the Latent State Estimates of a Revision Model*

Description

Accessor for the state paths of a fitted revision-nowcasting model. Provides programmatic access to the estimated states instead of reaching into the object with `fit$states`. The method is defined once for the parent class revision_model and is inherited by `kk_model` and `jvn_model` objects alike.

Usage

```
states(object, ...)

## S3 method for class 'revision_model'
states(object, filter = c("smoothed", "filtered", "all"), state = NULL, ...)
```

Arguments

`object`	A fitted model object inheriting from revision_model, such as a `kk_model` or a `jvn_model`.
`...`	Additional arguments passed to methods.
`filter`	Which state estimates to return: `"smoothed"` (default) uses the full sample, `"filtered"` uses information available up to each date, and `"all"` returns both.
`state`	Optional character vector of state names to keep. Defaults to all states.

Value

A tibble with columns `time`, `state`, `estimate`, `lower`, `upper`, `filter` and `sample`.

See Also

Other revision nowcasting: coef.revision_model(), fitted.revision_model(), jvn_nowcast(), kk_nowcast(), logLik.revision_model(), nobs.revision_model(), plot.revision_model(), predict.revision_model(), print.revision_model(), residuals.revision_model(), revision_model, summary.revision_model(), vcov.revision_model()

Examples

```
gdp_growth <- dplyr::filter(
  tsbox::ts_pc(reviser::gdp),
  id == "EA",
  time >= min(pub_date),
  time <= as.Date("2020-01-01")
)
gdp_growth <- tidyr::drop_na(gdp_growth)
df <- get_nth_release(gdp_growth, n = 0:3)

fit <- jvn_nowcast(df = df, e = 4, ar_order = 2, include_noise = FALSE)
head(states(fit))
head(states(fit, filter = "filtered", state = "true_lag_0"))
```

---

summary.lst_efficient *Summary of Efficient Release Models*

Description

Provides a detailed summary of the regression model and hypothesis test for the first efficient release identified by the get_first_efficient_release() function.

Usage

```
## S3 method for class 'lst_efficient'
summary(object, ...)
```

Arguments

object	An output object from the get_first_efficient_release function. The object must be of class lst_efficient.
...	Additional arguments (not used).

Details

This function prints the following information:

- The index of the first efficient release.

- A summary of the regression model fitted for the efficient release, which includes coefficients, R-squared values, and other relevant statistics.
- The hypothesis test results for the efficient release, showing the test statistic and p-value for the null hypothesis of unbiasedness and efficiency.

The function assumes the object includes:

- `e`: The index of the first efficient release (0-based).
- `models`: A list of linear regression models for each release.
- `tests`: A list of hypothesis test results corresponding to each release.

Value

Returns a tibble with the following columns:

- `id`: The identifier of the time series (if present in input data).
- `e`: The index of the first efficient release.
- `alpha`: The intercept coefficient of the regression model.
- `beta`: The coefficient of the slope.
- `p-value`: The p-value for the joint hypothesis (alpha = 0 and beta = 1).
- `n_tested`: The number of releases tested.

See Also

Other revision analysis: `diagnose()`, `diagnose.revision_summary()`, `get_first_efficient_release()`, `get_revision_analysis()`, `print.lst_efficient()`, `print.revision_summary()`, `summary.revision_summary()`

Examples

```
# Example usage
df <- get_nth_release(
  tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
  n = 1:4
)

final_release <- get_nth_release(
  tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
  n = 10
)

# Identify the first efficient release
result <- get_first_efficient_release(df, final_release, significance = 0.05)
summary(result)
```

summary.revision_model *Summary Method for Revision Models*

Description

Computes and displays a summary of a fitted revision model, including the estimated specification, convergence status, information criteria, and parameter estimates. Defined once for the parent class revision_model; the family-specific header and specification block are supplied by the concrete class.

Usage

```
## S3 method for class 'revision_model'
summary(object, ...)
```

Arguments

object	A fitted model object inheriting from revision_model, such as a kk_model or a jvn_model.
...	Additional arguments passed to or from other methods.

Value

The input object, invisibly.

See Also

Other revision nowcasting: coef.revision_model(), fitted.revision_model(), jvn_nowcast(), kk_nowcast(), logLik.revision_model(), nobs.revision_model(), plot.revision_model(), predict.revision_model(), print.revision_model(), residuals.revision_model(), revision_model, states(), vcov.revision_model()

Examples

```
df <- get_nth_release(
  tsbox::ts_span(
    tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
    start = "1980-01-01"
  ),
  n = 0:1
)
df <- na.omit(dplyr::select(df, -c("id", "pub_date")))
fit <- kk_nowcast(df, e = 1, h = 2, model = "Kishor-Koenig", method = "MLE")
summary(fit)
```

summary.revision_summary *Summary Method for Revision Summary*

Description

Summary Method for Revision Summary

Usage

```
## S3 method for class 'revision_summary'
summary(object, interpretation = TRUE, ...)
```

Arguments

object	An object of class revision_summary.
interpretation	Logical. If TRUE, provides interpretation of key statistics. Default is TRUE.
...	Additional arguments passed to print.

Value

The function returns the input object invisibly.

See Also

Other revision analysis: diagnose(), diagnose.revision_summary(), get_first_efficient_release(), get_revision_analysis(), print.lst_efficient(), print.revision_summary(), summary.lst_efficient()

Examples

```
# Example usage with revision analysis results
df <- dplyr::select(
  get_nth_release(
    na.omit(
      tsbox::ts_pc(
        dplyr::filter(reviser::gdp, id == "US")
      )
    ),
    n = 0:3
  ),
  -"pub_date"
)

final_release <- dplyr::select(
  get_nth_release(
    na.omit(
      tsbox::ts_pc(
        dplyr::filter(reviser::gdp, id == "US")
```

```
      )
    ),
    n = "latest"
  ),
  -"pub_date"
)

# Get revision analysis results
results <- get_revision_analysis(df, final_release, degree = 5)

# Summarize revision quality
summary(results)
```

`summary.tbl_vintage` *Summary Method for Vintages Data*

Description

Reports the layout, time coverage, number of vintages and missing-value count of a vintages object. Defined once for the parent class tbl_vintage; the block describing the vintages themselves is supplied by the concrete class.

Usage

```
## S3 method for class 'tbl_vintage'
summary(object, ...)
```

Arguments

`object`	An object inheriting from tbl_vintage, such as a `tbl_pubdate` or a `tbl_release`.
`...`	Additional arguments (not used).

Value

The input `object`, invisibly.

See Also

Other helpers: `print.tbl_vintage()`, `tbl_sum.tbl_vintage()`, `tbl_vintage`, `validate_vintages()`, `vintages_long()`, `vintages_wide()`

Examples

```
df <- dplyr::filter(reviser::gdp, id == "US")

# Long format
release_data <- get_nth_release(df, n = 0:3)
summary(release_data)
```

```
# Wide format
wide_release <- vintages_wide(release_data, names_from = "release")
summary(wide_release$US)

# Publication-date vintages
summary(vintages_wide(df)$US)
```

`tbl_sum.tbl_vintage` *Tibble Summary for Vintages Data*

Description

Provides the custom header shown when a vintages object is printed. This method is called automatically by **pillar**. It is defined once for the parent class tbl_vintage and inherited by `tbl_pubdate` and `tbl_release` objects alike, so that both report the same fields.

Usage

```
## S3 method for class 'tbl_vintage'
tbl_sum(x, ...)
```

Arguments

`x`	An object inheriting from tbl_vintage, such as a `tbl_pubdate` or a `tbl_release`.
`...`	Additional arguments (unused).

Value

A named character vector where names are labels and values are the corresponding information. The vector is used by pillar to format the tibble header. An object whose columns no longer match either documented layout falls back to the plain tibble header, so that a broken object can still be inspected; `summary()` reports the problem.

See Also

Other helpers: `print.tbl_vintage()`, `summary.tbl_vintage()`, `tbl_vintage`, `validate_vintages()`, `vintages_long()`, `vintages_wide()`

Examples

```
df <- dplyr::filter(reviser::gdp, id == "US")
release_data <- get_nth_release(df, n = 0:3)
pillar::tbl_sum(release_data)
```

tbl_vintage *Vintages Data Objects*

Description

reviser represents real-time vintages as tibbles that carry an additional class recording which dimension indexes the vintages. Data indexed by publication date are of class tbl_pubdate, data indexed by release number are of class tbl_release, and both inherit from the common parent class tbl_vintage, ahead of the usual tibble classes. A fitted class attribute therefore reads c("tbl_pubdate", "tbl_vintage", "tbl_df", "tbl", "data.frame"). An object that carries both a pub_date and a release column is classed as both, with tbl_release taking precedence.

The parent class holds everything the two representations share. The print(), summary(), plot() and pillar::tbl_sum() methods are defined once for tbl_vintage and inherited by both, so that the two stay consistent with one another. Only the parts that genuinely depend on the indexing dimension are dispatched on the child classes, through the internal generics vintage_labels(), vintage_value_cols() and vintage_detail().

Both representations may be stored in either a long or a wide layout, and the methods below detect which and report accordingly. The layouts, and the columns each one requires, are specified under "Data contract" in validate_vintages(), which also checks an object against them. Use vintages_long() and vintages_wide() to convert between layouts.

Value

This topic documents a class rather than a function. The functions that build vintages data, such as get_nth_release(), vintages_long() and vintages_wide(), return tibbles whose class attribute is c("tbl_pubdate", "tbl_vintage", "tbl_df", "tbl", "data.frame") or the same with "tbl_release" in place of "tbl_pubdate".

See Also

validate_vintages(), vintages_long(), vintages_wide()

Other helpers: print.tbl_vintage(), summary.tbl_vintage(), tbl_sum.tbl_vintage(), validate_vintages(), vintages_long(), vintages_wide()

Examples

```
df <- dplyr::filter(reviser::gdp, id == "US")

# Release vintages carry the shared parent class.
releases <- get_nth_release(df, n = 0:3)
class(releases)
inherits(releases, "tbl_vintage")

# So do publication-date vintages, in either layout.
class(vintages_wide(df)$US)

# The print, summary and plot methods are inherited from the parent.
```

```
summary(releases)
```

`theme_reviser` *Custom Visualization Theme and Color Scales for reviser*

Description

These functions provide a custom visualization theme and color scales for use with ggplot2, inspired by the `tsbox` package.

Usage

```
theme_reviser(
  base_size = 12,
  legend.position = "bottom",
  legend.direction = "horizontal"
)

colors_reviser()

scale_color_reviser(...)

scale_fill_reviser(...)
```

Arguments

`base_size`	Numeric. The base font size for the theme. Default is 12.
`legend.position`	Character. Position of the legend. Default is "bottom".
`legend.direction`	Character. Direction of the legend. Default is "horizontal".
`...`	Additional arguments passed to the ggplot2 scale functions.

Details

- `theme_reviser`: Defines a minimal theme with custom adjustments for axis titles, plot titles, subtitles, captions, and legend positioning.
- `colors_reviser`: Provides a predefined set of colors, including a soft black, a palette suitable for colorblind readers, and additional colors for extended use.
- `scale_color_reviser`: A ggplot2 color scale that uses the custom `colors_reviser` palette.
- `scale_fill_reviser`: A ggplot2 fill scale that uses the custom `colors_reviser` palette.

Value

A customized ggplot2 theme, color palette, or scale.

See Also

Other revision graphs: plot.tbl_vintage(), plot_vintages()

Other revision graphs: plot.tbl_vintage(), plot_vintages()

Other revision graphs: plot.tbl_vintage(), plot_vintages()

Other revision graphs: plot.tbl_vintage(), plot_vintages()

Examples

```
library(ggplot2)
ggplot(mtcars, aes(x = wt, y = mpg, color = factor(cyl))) +
  geom_point(size = 3) +
  theme_reviser() +
  scale_color_reviser()
```

---

validate_vintages *Vintages Data Classes and Their Validation*

Description

reviser stores vintages in two S3 classes that sit on top of a tibble and record what the columns mean. Both are produced by the package's own constructors; you normally do not create them by hand.

tbl_pubdate Vintages identified by publication date, as returned by vintages_wide(), vintages_long() and get_revisions().

tbl_release Vintages identified by release number, as returned by get_first_release(), get_nth_release(), get_latest_release(), get_fixed_release() and get_releases_by_date().

validate_vintages() checks that an object conforms to the contract below. This is useful after manipulating a vintages object with external tools, which can leave the class attribute in place while breaking the assumptions the methods rely on.

Usage

```
validate_vintages(x)
```

Arguments

x	An object of class tbl_pubdate or tbl_release.

Value

x, invisibly, if it is valid. Otherwise an error describing the first problem found.

Data contract

Every object of either class has a `time` column of dates and may carry an optional `id` column identifying the series. Beyond that, each class has two permitted layouts:

long A key column (`pub_date` for `tbl_pubdate`, `release` for `tbl_release`) together with a `value` column.

wide One column per vintage. For `tbl_pubdate` the column names are publication dates in `%Y-%m-%d` form; for `tbl_release` they are release labels matching `release` or `final`.

Columns must be atomic and scalar-valued; list columns are not permitted. The two classes are not mutually exclusive: a long release table carries both a `release` and a `pub_date` column and holds both classes, with `tbl_release` taking precedence for method dispatch.

A valid object also inherits from the shared parent class tbl_vintage, which is where its methods live; `validate_vintages()` checks for that too. See tbl_vintage for the class hierarchy and the methods it provides.

Operations that drop the class

The vintages classes sit on top of a tibble, so the `dplyr` verbs (`filter()`, `mutate()`, `select()`, `arrange()`, `slice()`) and `[` preserve them. A few functions rebuild the object from scratch and return a plain tibble instead; `tidyr::drop_na()` is the one most likely to be met in a vintages workflow. The data are unaffected, but `plot()`, `summary()` and the vintages print header no longer dispatch. Pass the result back through `vintages_long()` or `vintages_wide()`, or apply the operation before the release-extraction step, to get the class back.

See Also

Other helpers: `print.tbl_vintage()`, `summary.tbl_vintage()`, `tbl_sum.tbl_vintage()`, `tbl_vintage`, `vintages_long()`, `vintages_wide()`

Examples

```
df <- dplyr::filter(reviser::gdp, id == "US")

releases <- get_nth_release(df, n = 0:3)
validate_vintages(releases)

# A malformed time column is rejected
broken <- releases
broken$time <- as.character(broken$time)
broken$time[1] <- "not a date"
try(validate_vintages(broken))

# So is a class attribute that contradicts the columns
mislabelled <- vintages_wide(df)$US
class(mislabelled) <- c("tbl_release", class(mislabelled))
try(validate_vintages(mislabelled))
```

`vcov.revision_model` *Extract the Parameter Covariance Matrix of a Revision Model*

Description

Extract the Parameter Covariance Matrix of a Revision Model

Usage

```
## S3 method for class 'revision_model'
vcov(object, ...)
```

Arguments

`object`	A fitted model object inheriting from revision_model, such as a `kk_model` or a `jvn_model`.
`...`	Ignored.

Value

The estimated parameter covariance matrix.

See Also

Other revision nowcasting: `coef.revision_model()`, `fitted.revision_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.revision_model()`, `nobs.revision_model()`, `plot.revision_model()`, `predict.revision_model()`, `print.revision_model()`, `residuals.revision_model()`, `revision_model`, `states()`, `summary.revision_model()`

Examples

```
df <- get_nth_release(
  tsbox::ts_span(
    tsbox::ts_pc(dplyr::filter(reviser::gdp, id == "US")),
    start = "1980-01-01"
  ),
  n = 0:1
)
df <- na.omit(dplyr::select(df, -c("id", "pub_date")))
fit <- kk_nowcast(df, e = 1, model = "KK", method = "MLE")
vcov(fit)
```

vintages_long *Convert Vintages Data to Long Format*

Description

Converts a vintages dataset from wide format to long format, optionally adding `id` if the input is a list of data frames. The long format contains one row per combination of `time` and `names_to` (e.g., `pub_date` or `release`), with values stored in a single `value` column.

Usage

```
vintages_long(df, names_to = "pub_date", keep_na = FALSE)
```

Arguments

`df`	A data frame, tibble, or list of data frames containing vintages data in wide format.
`names_to`	The name of the column to create from the wide-format column names. Must be either `"pub_date"` (default) or `"release"`.
`keep_na`	Logical. If `TRUE`, retains rows with `NA` values in the `value` column. Default is `FALSE`.

Value

A long-format vintages object: a tibble carrying the `tbl_pubdate` or `tbl_release` class and their shared parent tbl_vintage. If the input is a list of wide-format data frames, the output is a single combined long-format object.

Long-format input is returned with the vintages class attached, which is also how to recover the class after an operation that dropped it (see the "Operations that drop the class" section of `validate_vintages()`). Input that is already a long-format vintages object warns, because the call is then a no-op.

See Also

Other helpers: `print.tbl_vintage()`, `summary.tbl_vintage()`, `tbl_sum.tbl_vintage()`, `tbl_vintage`, `validate_vintages()`, `vintages_wide()`

Examples

```
# Example wide-format data
long_data <- dplyr::filter(reviser::gdp, id == "US")

# Convert to wide format
wide_data <- vintages_wide(long_data)

# Example list of wide-format data frames
wide_list <- list(
  A = wide_data$US,
```

```
  B = wide_data$US
)

# Convert list to long format
long_data <- vintages_long(wide_list, names_to = "pub_date")
```

`vintages_wide` *Convert Vintages Data to Wide Format*

Description

Converts a vintages dataset from long format to wide format, optionally grouping by `id` if present. The wide format uses one column per unique value of the `names_from` parameter, with observation dates (`time`) as rows and values (`value`) as cell contents.

Usage

```
vintages_wide(df, names_from = "pub_date")
```

Arguments

`df`	A data frame or tibble containing vintages data in long format.
`names_from`	The name of the column whose unique values will be used as column names in the wide format. Defaults to `"pub_date"`. Other: `"release"`.

Value

If an `id` column is present, the function returns a named list of wide-format data frames, one for each unique `id`. Otherwise, it returns a single wide-format data frame.

See Also

Other helpers: `print.tbl_vintage()`, `summary.tbl_vintage()`, `tbl_sum.tbl_vintage()`, `tbl_vintage`, `validate_vintages()`, `vintages_long()`

Examples

```
# Example wide-format data
long_data <- dplyr::filter(reviser::gdp, id == "US")

# Convert to wide format
wide_data <- vintages_wide(long_data)

# Example list of wide-format data frames
wide_list <- list(
  A = wide_data$US,
  B = wide_data$US
)
```

```
# Convert list to long format
long_data1 <- vintages_long(wide_data, names_to = "pub_date")
long_data2 <- vintages_long(wide_list, names_to = "pub_date")
```

Index

∗ **datasets**
gdp, 7
∗ **dataset**
gdp, 7
∗ **helpers**
print.tbl_vintage, 37
summary.tbl_vintage, 45
tbl_sum.tbl_vintage, 46
tbl_vintage, 47
validate_vintages, 49
vintages_long, 52
vintages_wide, 53
∗ **revision analysis**
diagnose, 4
diagnose.revision_summary, 5
get_first_efficient_release, 9
get_revision_analysis, 16
print.lst_efficient, 34
print.revision_summary, 36
summary.lst_efficient, 41
summary.revision_summary, 44
∗ **revision graphs**
plot.tbl_vintage, 30
plot_vintages, 31
theme_reviser, 48
∗ **revision nowcasting**
coef.revision_model, 3
fitted.revision_model, 6
jvn_nowcast, 21
kk_nowcast, 24
logLik.revision_model, 27
nobs.revision_model, 28
plot.revision_model, 29
predict.revision_model, 33
print.revision_model, 35
residuals.revision_model, 38
revision_model, 39
states, 40
summary.revision_model, 43
vcov.revision_model, 51
∗ **revision utilities**
get_days_to_release, 8
get_first_release, 11
get_fixed_release, 12
get_latest_release, 13
get_nth_release, 14
get_releases_by_date, 15
get_revisions, 19

coef(), *39*
coef.revision_model, 3, *7*, *23*, *26–28*, *30*, *34*, *35*, *38*, *39*, *41*, *43*, *51*
colors_reviser *(*theme_reviser*)*, 48

diagnose, 4, *5*, *11*, *19*, *34*, *36*, *42*, *44*
diagnose.revision_summary, *4*, 5, *11*, *19*, *34*, *36*, *42*, *44*

fitted(), *29*, *39*
fitted.revision_model, *3*, 6, *23*, *26–28*, *30*, *34*, *35*, *38*, *39*, *41*, *43*, *51*

gdp, 7
get_days_to_release, 8, *12–16*, *20*
get_first_efficient_release, *4*, *5*, 9, *19*, *34*, *36*, *42*, *44*
get_first_release, *9*, 11, *13–16*, *20*
get_first_release(), *49*
get_fixed_release, *9*, 12, *12*, *14–16*, *20*
get_fixed_release(), *49*
get_latest_release, *9*, *12*, 13, *13*, *15*, *16*, *20*
get_latest_release(), *49*
get_nth_release, *9*, *12*, *13*, 14, *14*, *16*, *20*
get_nth_release(), *47*, *49*
get_releases_by_date, *9*, *12–14*, 15, *15*, *20*
get_releases_by_date(), *49*
get_revision_analysis, *4*, *5*, *11*, 16, *34*, *36*, *42*, *44*
get_revisions, *9*, *12–16*, 19

get_revisions(), *49*

jvn_nowcast, *3*, *7*, 21, *26–28*, *30*, *34, 35*, *38, 39*, *41*, *43*, *51*
jvn_nowcast(), *33*, *39*

kk_nowcast, *3*, *7*, *23*, 24, *27, 28*, *30*, *34, 35*, *38, 39*, *41*, *43*, *51*
kk_nowcast(), *33*, *39*

logLik(), *39*
logLik.revision_model, *3*, *7*, *23*, *26*, 27, *28*, *30*, *34, 35*, *38, 39*, *41*, *43*, *51*

nobs(), *39*
nobs.revision_model, *3*, *7*, *23*, *26, 27*, 28, *30*, *34, 35*, *38, 39*, *41*, *43*, *51*

pillar::tbl_sum(), *47*
plot(), *7*, *39*, *47*
plot.revision_model, *3*, *7*, *23*, *26–28*, 29, *34, 35*, *38, 39*, *41*, *43*, *51*
plot.tbl_vintage, 30, *32*, *49*
plot_vintages, 31, *31*, *49*
plot_vintages(), *30*
predict(), *29*, *39*
predict.revision_model, *3*, *7*, *23*, *26–28*, *30*, 33, *35*, *38, 39*, *41*, *43*, *51*
print(), *7*, *39*, *47*
print.lst_efficient, *4, 5*, *11*, *19*, 34, *36*, *42*, *44*
print.revision_model, *3*, *7*, *23*, *26–28*, *30*, *34*, 35, *38, 39*, *41*, *43*, *51*
print.revision_summary, *4, 5*, *11*, *19*, *34*, 36, *42*, *44*
print.tbl_vintage, 37, *45–47*, *50*, *52, 53*

residuals(), *29*, *39*
residuals.revision_model, *3*, *7*, *23*, *26–28*, *30*, *34, 35*, 38, *39*, *41*, *43*, *51*
reviser-vintages-classes *(*validate_vintages*)*, 49
revision_model, *3*, *6, 7*, *23*, *26–30*, *33–35*, *38*, 39, *40, 41*, *43*, *51*

scale_color_reviser *(*theme_reviser*)*, 48
scale_color_reviser(), *32*
scale_fill_reviser *(*theme_reviser*)*, 48
scale_fill_reviser(), *32*
states, *3*, *7*, *23*, *26–28*, *30*, *34, 35*, *38, 39*, 40, *43*, *51*
states(), *29*, *39*
stats::AIC(), *27*
stats::BIC(), *27*
summary(), *7*, *39*, *46, 47*
summary.lst_efficient, *4, 5*, *11*, *19*, *34*, *36*, 41, *44*
summary.revision_model, *3*, *7*, *23*, *26–28*, *30*, *34, 35*, *38, 39*, *41*, 43, *51*
summary.revision_model(), *35*
summary.revision_summary, *4, 5*, *11*, *19*, *34*, *36*, *42*, 44
summary.tbl_vintage, *38*, 45, *46, 47*, *50*, *52, 53*

tbl_sum.tbl_vintage, *38*, *45*, 46, *47*, *50*, *52, 53*
tbl_sum.tbl_vintage(), *37*
tbl_vintage, *7, 8*, *30*, *37, 38*, *45, 46*, 47, *50*, *52, 53*
theme_reviser, *31, 32*, 48
theme_reviser(), *32*

validate_vintages, *38*, *45–47*, 49, *52, 53*
validate_vintages(), *7, 8*, *47*, *52*
vcov(), *39*
vcov.revision_model, *3*, *7*, *23*, *26–28*, *30*, *34, 35*, *38, 39*, *41*, *43*, 51
vintages_long, *38*, *45–47*, *50*, 52, *53*
vintages_long(), *47*, *49, 50*
vintages_wide, *38*, *45–47*, *50*, *52*, 53
vintages_wide(), *47*, *49, 50*