

Package ‘reviser’

August 22, 2026

Type Package

Title Analyzing Revisions in Real-Time Time Series Vintages

Version 0.2.0

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>](https://doi.org/10.2307/1924972), Jacobs and Van Norden (2011) [<doi:10.1016/j.jeconom.2010.04.010>](https://doi.org/10.1016/j.jeconom.2010.04.010), and Kishor and Koenig (2012) [<doi:10.1198/jbes.2010.08169>](https://doi.org/10.1198/jbes.2010.08169).

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

RoxygenNote 7.3.3

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>

NeedsCompilation no

Author Marc Burri [aut, cre, cph] (ORCID:
 [<https://orcid.org/0000-0001-8974-9090>](https://orcid.org/0000-0001-8974-9090)),
Philipp Wegmueller [aut, cph]

Maintainer Marc Burri <marc.burri91@gmail.com>

Repository CRAN

Date/Publication 2026-08-22 18:30:02 UTC

Contents

coef.jvn_model	3
coef.kk_model	4
diagnose	5
diagnose.revision_summary	6
fitted.jvn_model	7
fitted.kk_model	8
gdp	9
get_days_to_release	10
get_first_efficient_release	11
get_first_release	13
get_fixed_release	14
get_latest_release	15
get_nth_release	16
get_releases_by_date	17
get_revisions	18
get_revision_analysis	20
jvn_nowcast	23
kk_nowcast	26
logLik.jvn_model	29
logLik.kk_model	30
nobs.jvn_model	31
nobs.kk_model	32
plot.jvn_model	33
plot.kk_model	34
plot.tbl_pubdate	35
plot.tbl_release	36
plot_vintages	36
predict.jvn_model	39
predict.kk_model	40
print.jvn_model	41
print.kk_model	42
print.lst_efficient	43
print.revision_summary	44
print.tbl_pubdate	45
print.tbl_release	46
residuals.jvn_model	46
residuals.kk_model	47
states	48
summary.jvn_model	49
summary.kk_model	50
summary.lst_efficient	51

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)
coef(fit)

```

coef.kk_model

*Extract parameter estimates from a KK model***Description**

Extract parameter estimates from a KK model

Usage

```

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

```

Arguments

object	An object of class <code>kk_model</code> .
...	Ignored.

Value

A named numeric vector of parameter estimates.

See Also

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

<code>object</code>	An object for which diagnostics are desired.
<code>...</code>	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 Revision Quality

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.jvn_model

*Fitted true values from a JVN model***Description**

Returns the smoothed estimate of the latent true value for the in-sample periods, i.e. the model's revision-adjusted signal.

Usage

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

Arguments

object	An object of class jvn_model.
...	Ignored.

Value

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

See Also

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

fitted.kk_model

Fitted efficient estimates from a KK model

Description

Returns the smoothed estimate of the latent efficient value for the in-sample periods, i.e. the model's revision-adjusted signal.

Usage

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

Arguments

object	An object of class kk_model.
...	Ignored.

Value

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

See Also

Other revision nowcasting: `coef.jvn_model()`, `coef.kk_model()`, `fitted.jvn_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.jvn_model()`, `logLik.kk_model()`, `nobs.jvn_model()`, `nobs.kk_model()`, `plot.jvn_model()`, `plot.kk_model()`, `predict.jvn_model()`, `predict.kk_model()`, `print.jvn_model()`, `print.kk_model()`, `residuals.jvn_model()`, `residuals.kk_model()`, `states()`, `summary.jvn_model()`, `summary.kk_model()`, `vcov.jvn_model()`, `vcov.kk_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	<i>Vintages Data</i>
-----	----------------------

Description

A collection of real-time datasets.

Usage

gdp

Format

A tibble 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

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](https://doi.org/10.1007/BF03399410)

Examples

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

<code>get_days_to_release</code>	<i>Calculate the Number of Days Between Period End and First Release</i>
----------------------------------	--

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

<code>df</code>	A data frame containing data vintages. The data frame must include the columns <code>pub_date</code> (publication date of the release) and <code>time</code> (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

<code>df</code>	A data frame of class <code>tbl_release</code> containing the vintage data. It must include the columns: • <code>time</code>: The reference period (e.g., quarter or month). • <code>value</code>: The observed value for the given release. • <code>release</code>: The release number or identifier.
<code>final_release</code>	A data frame containing the final release data. This must include the columns: • <code>time</code>: The reference period. • <code>value</code>: The observed final value for the given period.
<code>significance</code>	A numeric value specifying the significance level for the hypothesis test (default is 0.05).
<code>test_all</code>	A logical value indicating whether to test all releases, even after finding the first efficient release (default is FALSE).
<code>robust</code>	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	<i>Extract the First Data Release (Vintage)</i>
-------------------	---

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 <code>pub_date</code> (publication date of the release) and <code>time</code> (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	<i>Extract Vintage Values from a Data Frame</i>
-------------------	---

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	<i>Extract the Latest Data Release (Vintage)</i>
--------------------	--

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 <code>pub_date</code> (publication date of the release) and <code>time</code> (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	<i>Extract the Nth Data Release (Vintage)</i>
-----------------	---

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_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)
```

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
)

```

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.

<code>include_spillovers</code>	Logical; whether to include spillover effects.
<code>spillover_news</code>	Logical; whether spillovers apply to the news component.
<code>spillover_noise</code>	Logical; whether spillovers apply to the noise component.
<code>method</code>	Estimation method. Currently only "MLE" is supported.
<code>alpha</code>	Significance level used for confidence intervals. Must lie in $(0, 1)$.
<code>standardize</code>	Logical; whether to standardize the vintage matrix before estimation using <code>jvn_standardize()</code> . If TRUE, scaling metadata are returned in the <code>scale</code> element of the output.
<code>solver_options</code>	A named list of solver options. Valid names are <code>trace</code>, <code>method</code>, <code>maxiter</code>, <code>transform_se</code>, <code>startvals</code>, <code>se_method</code>, <code>n_starts</code>, <code>seed</code>, <code>return_states</code>, <code>qml_eps</code>, <code>qml_score_method</code>, <code>qml_scale</code>, <code>sigma_lower</code>, <code>sigma_upper</code>, <code>kfas_init</code>, and <code>ic_n</code>. Supported entries are: • <code>trace</code>: integer controlling console output. • <code>method</code>: optimization method; one of "L-BFGS-B", "BFGS", "Nelder-Mead", "nlminb", or "two-step". • <code>maxiter</code>: maximum number of optimizer iterations. • <code>transform_se</code>: logical; whether standard deviation parameters are optimized on the log scale. • <code>startvals</code>: 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 <code>jvn_param_table()</code>: AR coefficients <code>rho_*</code>, <code>sigma_e</code>, optional <code>sigma_nu_*</code>, optional <code>sigma_zeta_*</code>, and optional spillover parameters <code>T_nu_*</code> and <code>T_zeta_*</code>. • <code>se_method</code>: standard-error method; one of "hessian", "qml", or "none". • <code>n_starts</code>: number of random starting points for multi-start optimization. • <code>seed</code>: optional random seed used for multi-start perturbations. • <code>return_states</code>: logical; whether filtered and smoothed state estimates should be returned. • <code>qml_eps</code>: finite-difference step size used in QML covariance estimation. • <code>qml_score_method</code>: score approximation method; "forward" or "central". • <code>qml_scale</code>: scaling convention for the QML covariance; "sum", "mean", or "hc". • <code>sigma_lower</code>: lower bound for standard deviations on the natural scale. • <code>sigma_upper</code>: upper bound for standard deviations on the natural scale. • <code>kfas_init</code>: initialization used in the KFAS filtering/smoothing stage; "stationary" or "diffuse". This affects state extraction, not the custom likelihood engine. • <code>ic_n</code>: 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`.

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.jvn_model()`, `coef.kk_model()`, `fitted.jvn_model()`, `fitted.kk_model()`, `kk_nowcast()`, `logLik.jvn_model()`, `logLik.kk_model()`, `nobs.jvn_model()`, `nobs.kk_model()`, `plot.jvn_model()`, `plot.kk_model()`, `predict.jvn_model()`, `predict.kk_model()`, `print.jvn_model()`, `print.kk_model()`, `residuals.jvn_model()`, `residuals.kk_model()`, `states()`, `summary.jvn_model()`, `summary.kk_model()`, `vcov.jvn_model()`, `vcov.kk_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.jvn_model()`, `coef.kk_model()`, `fitted.jvn_model()`, `fitted.kk_model()`, `jvn_nowcast()`, `logLik.jvn_model()`, `logLik.kk_model()`, `nobs.jvn_model()`, `nobs.kk_model()`, `plot.jvn_model()`, `plot.kk_model()`, `predict.jvn_model()`, `predict.kk_model()`, `print.jvn_model()`, `print.kk_model()`, `residuals.jvn_model()`, `residuals.kk_model()`, `states()`, `summary.jvn_model()`, `summary.kk_model()`, `vcov.jvn_model()`, `vcov.kk_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")

result$params
```

logLik.jvn_model

Extract the log-likelihood of a JVN 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 'jvn_model'
logLik(object, ...)
```

Arguments

<code>object</code>	An object of class <code>jvn_model</code> .
<code>...</code>	Ignored.

Value

An object of class `logLik`.

See Also

Other revision nowcasting: `coef.jvn_model()`, `coef.kk_model()`, `fitted.jvn_model()`, `fitted.kk_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.kk_model()`, `nobs.jvn_model()`, `nobs.kk_model()`, `plot.jvn_model()`, `plot.kk_model()`, `predict.jvn_model()`, `predict.kk_model()`, `print.jvn_model()`, `print.kk_model()`, `residuals.jvn_model()`, `residuals.kk_model()`, `states()`, `summary.jvn_model()`, `summary.kk_model()`, `vcov.jvn_model()`, `vcov.kk_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)
logLik(fit)
AIC(fit)
BIC(fit)
```

logLik.kk_model	<i>Extract the log-likelihood of a KK model</i>
-----------------	---

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 'kk_model'
logLik(object, ...)
```

Arguments

object	An object of class <code>kk_model</code> .
...	Ignored.

Value

An object of class `logLik`.

See Also

Other revision nowcasting: `coef.jvn_model()`, `coef.kk_model()`, `fitted.jvn_model()`, `fitted.kk_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.jvn_model()`, `nobs.jvn_model()`, `nobs.kk_model()`, `plot.jvn_model()`, `plot.kk_model()`, `predict.jvn_model()`, `predict.kk_model()`, `print.jvn_model()`, `print.kk_model()`, `residuals.jvn_model()`, `residuals.kk_model()`, `states()`, `summary.jvn_model()`, `summary.kk_model()`, `vcov.jvn_model()`, `vcov.kk_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.jvn_model	<i>Number of observations used to fit a JVN model</i>
----------------	---

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 vintages modeled.

Usage

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

Arguments

<code>object</code>	An object of class <code>jvn_model</code> .
<code>...</code>	Ignored.

Value

A single integer.

See Also

Other revision nowcasting: [coef.jvn_model\(\)](#), [coef.kk_model\(\)](#), [fitted.jvn_model\(\)](#), [fitted.kk_model\(\)](#), [jvn_nowcast\(\)](#), [kk_nowcast\(\)](#), [logLik.jvn_model\(\)](#), [logLik.kk_model\(\)](#), [nobs.kk_model\(\)](#), [plot.jvn_model\(\)](#), [plot.kk_model\(\)](#), [predict.jvn_model\(\)](#), [predict.kk_model\(\)](#), [print.jvn_model\(\)](#), [print.kk_model\(\)](#), [residuals.jvn_model\(\)](#), [residuals.kk_model\(\)](#), [states\(\)](#), [summary.jvn_model\(\)](#), [summary.kk_model\(\)](#), [vcov.jvn_model\(\)](#), [vcov.kk_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)
nobs(fit)
```

nobs.kk_model	<i>Number of observations used to fit a KK model</i>
---------------	--

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 modeled.

Usage

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

Arguments

<code>object</code>	An object of class <code>kk_model</code> .
<code>...</code>	Ignored.

Value

A single integer.

See Also

Other revision nowcasting: [coef.jvn_model\(\)](#), [coef.kk_model\(\)](#), [fitted.jvn_model\(\)](#), [fitted.kk_model\(\)](#), [jvn_nowcast\(\)](#), [kk_nowcast\(\)](#), [logLik.jvn_model\(\)](#), [logLik.kk_model\(\)](#), [nobs.jvn_model\(\)](#), [plot.jvn_model\(\)](#), [plot.kk_model\(\)](#), [predict.jvn_model\(\)](#), [predict.kk_model\(\)](#), [print.jvn_model\(\)](#), [print.kk_model\(\)](#), [residuals.jvn_model\(\)](#), [residuals.kk_model\(\)](#), [states\(\)](#), [summary.jvn_model\(\)](#), [summary.kk_model\(\)](#), [vcov.jvn_model\(\)](#), [vcov.kk_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.jvn_model	<i>Plot JVN model results</i>
----------------	-------------------------------

Description

Plot filtered or smoothed estimates for a selected state from a fitted jvn_model.

Usage

```
## S3 method for class 'jvn_model'
plot(x, state = "true_lag_0", type = "filtered", ...)
```

Arguments

x	An object of class jvn_model.
state	Character scalar giving the state to visualize. Defaults to "true_lag_0".
type	Character scalar indicating whether "filtered" or "smoothed" estimates should be plotted.
...	Additional arguments passed to plot.revision_model().

Details

This method requires x\$states to be available. If the model was fitted with solver_options\$return_states = FALSE, plotting is not possible.

Value

A ggplot2 object.

See Also

Other revision nowcasting: [coef.jvn_model\(\)](#), [coef.kk_model\(\)](#), [fitted.jvn_model\(\)](#), [fitted.kk_model\(\)](#), [jvn_nowcast\(\)](#), [kk_nowcast\(\)](#), [logLik.jvn_model\(\)](#), [logLik.kk_model\(\)](#), [nobs.jvn_model\(\)](#), [nobs.kk_model\(\)](#), [plot.kk_model\(\)](#), [predict.jvn_model\(\)](#), [predict.kk_model\(\)](#), [print.jvn_model\(\)](#), [print.kk_model\(\)](#), [residuals.jvn_model\(\)](#), [residuals.kk_model\(\)](#), [states\(\)](#), [summary.jvn_model\(\)](#), [summary.kk_model\(\)](#), [vcov.jvn_model\(\)](#), [vcov.kk_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
)
plot(result)

```

plot.kk_model

Plot Kishor-Koenig Model Results

Description

Plot filtered or smoothed estimates for a selected state from a fitted `kk_model`.

Usage

```

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

```

Arguments

<code>x</code>	An object of class <code>kk_model</code> .
<code>state</code>	Character scalar giving the state to visualize. If <code>NULL</code> , the first available state is used.
<code>type</code>	Character scalar indicating whether "filtered" or "smoothed" estimates should be plotted.
<code>...</code>	Additional arguments passed to <code>plot.revision_model()</code> .

Details

This method requires `x$states` to be available. If the model was fitted with `solver_options$return_states = FALSE`, plotting is not possible.

Value

A `ggplot2` object visualizing the specified state estimates.

See Also

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

*Plot Method for Publication Date Vintages***Description**

Plot Method for Publication Date Vintages

Usage

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

Arguments

x An object of class `tbl_pubdate`.

... Additional arguments passed to `plot_vintages`.

Value

A `ggplot2` object.

See Also

Other revision graphs: [plot.tbl_release\(\)](#), [plot_vintages\(\)](#), [theme_reviser\(\)](#)

Examples

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

plot.tbl_release	<i>Plot Method for Release Vintages</i>
------------------	---

Description

Plot Method for Release Vintages

Usage

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

Arguments

- x An object of class tbl_release.
- ... Additional arguments passed to plot_vintages.

Value

A ggplot2 object.

See Also

Other revision graphs: [plot.tbl_pubdate\(\)](#), [plot_vintages\(\)](#), [theme_reviser\(\)](#)

Examples

```
df <- dplyr::filter(reviser::gdp, id == "US")
df <- get_nth_release(df, n = 0:5)
plot(df)
```

plot_vintages	<i>Plot Vintages Data</i>
---------------	---------------------------

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

<code>df</code>	A data frame containing the vintage data to be plotted. Must include at least two columns: one for time (<code>time</code>) and one for value (<code>value</code>).
<code>type</code>	A character string specifying the type of plot to create. Options are: • "line": Line plot (default). • "point": Scatter plot. • "bar": Bar plot. • "boxplot": Boxplot.
<code>dim_col</code>	A character string specifying the column name in <code>df</code> that represents publication dates or other grouping dimensions (e.g. "release"). Defaults to "pub_date".
<code>time_col</code>	A character string specifying the column name in <code>df</code> that represents the time variable. Defaults to "time".
<code>title</code>	A character string specifying the title of the plot. Defaults to an empty string.
<code>subtitle</code>	A character string specifying the subtitle of the plot. Defaults to an empty string.
<code>ylab</code>	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_pubdate\(\)](#), [plot.tbl_release\(\)](#), [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.jvn_model	<i>Forecasts from a JVN model</i>
-------------------	-----------------------------------

Description

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

Usage

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

Arguments

object	An object of class <code>jvn_model</code> .
...	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.jvn_model()`, `coef.kk_model()`, `fitted.jvn_model()`, `fitted.kk_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.jvn_model()`, `logLik.kk_model()`, `nobs.jvn_model()`, `nobs.kk_model()`, `plot.jvn_model()`, `plot.kk_model()`, `predict.kk_model()`, `print.jvn_model()`, `print.kk_model()`, `residuals.jvn_model()`, `residuals.kk_model()`, `states()`, `summary.jvn_model()`, `summary.kk_model()`, `vcov.jvn_model()`, `vcov.kk_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, h = 2, include_noise = FALSE
)
predict(fit)
```

predict.kk_model	<i>Forecasts from a KK model</i>
------------------	----------------------------------

Description

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

Usage

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

Arguments

<code>object</code>	An object of class <code>kk_model</code> .
<code>...</code>	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.jvn_model()`, `coef.kk_model()`, `fitted.jvn_model()`, `fitted.kk_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.jvn_model()`, `logLik.kk_model()`, `nobs.jvn_model()`, `nobs.kk_model()`, `plot.jvn_model()`, `plot.kk_model()`, `predict.jvn_model()`, `print.jvn_model()`, `print.kk_model()`, `residuals.jvn_model()`, `residuals.kk_model()`, `states()`, `summary.jvn_model()`, `summary.kk_model()`, `vcov.jvn_model()`, `vcov.kk_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.jvn_model	<i>Print method for JVN model objects</i>
-----------------	---

Description

Default print method for jvn_model objects. This method dispatches to `summary.jvn_model()` for a consistent console display.

Usage

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

Arguments

x	An object of class jvn_model.
...	Additional arguments passed to <code>summary.jvn_model()</code> .

Value

The input object, invisibly.

See Also

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

```
result
```

print.kk_model	<i>Print Method for KK Model</i>
----------------	----------------------------------

Description

Default print method for `kk_model` objects. Wraps the `summary` method for a consistent output.

Usage

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

Arguments

<code>x</code>	An object of class <code>kk_model</code> .
<code>...</code>	Additional arguments passed to <code>summary.kk_model</code> .

Value

The function returns the input `x` invisibly.

See Also

Other revision nowcasting: [coef.jvn_model\(\)](#), [coef.kk_model\(\)](#), [fitted.jvn_model\(\)](#), [fitted.kk_model\(\)](#), [jvn_nowcast\(\)](#), [kk_nowcast\(\)](#), [logLik.jvn_model\(\)](#), [logLik.kk_model\(\)](#), [nobs.jvn_model\(\)](#), [nobs.kk_model\(\)](#), [plot.jvn_model\(\)](#), [plot.kk_model\(\)](#), [predict.jvn_model\(\)](#), [predict.kk_model\(\)](#), [print.jvn_model\(\)](#), [residuals.jvn_model\(\)](#), [residuals.kk_model\(\)](#), [states\(\)](#), [summary.jvn_model\(\)](#), [summary.kk_model\(\)](#), [vcov.jvn_model\(\)](#), [vcov.kk_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
h <- 2
```

```
result <- kk_nowcast(df, e, h = h, model = "Kishor-Koenig", method = "MLE")
result
```

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_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

<code>x</code>	An object of class <code>revision_summary</code> .
<code>interpretation</code>	Logical. If <code>TRUE</code> , provides interpretation of key statistics. Default is <code>TRUE</code> .
<code>digits</code>	Integer. Number of digits to display. Default is 3.
<code>...</code>	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_pubdate	<i>Print Method for Publication Date Vintages</i>
-------------------	---

Description

Print method for objects of class `tbl_pubdate`. This method delegates to the tibble print method, which will automatically call `tbl_sum.tbl_pubdate` to generate the custom header.

Usage

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

Arguments

<code>x</code>	An object of class <code>tbl_pubdate</code> .
<code>...</code>	Additional arguments passed to the next print method.

Value

The input `x` is returned invisibly.

See Also

Other helpers: [print.tbl_release\(\)](#), [summary.tbl_pubdate\(\)](#), [summary.tbl_release\(\)](#), [tbl_sum.tbl_pubdate\(\)](#), [tbl_sum.tbl_release\(\)](#), [validate_vintages\(\)](#), [vintages_long\(\)](#), [vintages_wide\(\)](#)

Examples

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

<code>print.tbl_release</code>	<i>Print Method for Release Vintages</i>
--------------------------------	--

Description

Print method for objects of class `tbl_release`.

Usage

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

Arguments

<code>x</code>	An object of class <code>tbl_release</code> .
<code>...</code>	Additional arguments passed to the next print method.

Value

The input `x` is returned invisibly.

See Also

Other helpers: `print.tbl_pubdate()`, `summary.tbl_pubdate()`, `summary.tbl_release()`, `tbl_sum.tbl_pubdate()`, `tbl_sum.tbl_release()`, `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)
```

<code>residuals.jvn_model</code>	<i>Residuals of a JVN model</i>
----------------------------------	---------------------------------

Description

Difference between the most mature release included in the estimation and the smoothed estimate of the latent true value. These are measurement residuals of that release, not one-step-ahead prediction errors.

Usage

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

Arguments

object	An object of class <code>jvn_model</code> .
...	Ignored.

Value

A tibble with columns `time` and `residual`.

See Also

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

<code>residuals.kk_model</code>	<i>Residuals of a KK model</i>
---------------------------------	--------------------------------

Description

Difference between the observed efficient release and the smoothed estimate of the latent efficient value. These are measurement residuals of the release used as the model's target, not one-step-ahead prediction errors.

Usage

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

Arguments

object	An object of class <code>kk_model</code> .
...	Ignored.

Value

A tibble with columns `time` and `residual`.

See Also

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

`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`.

Usage

```
states(object, ...)

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

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

Arguments

<code>object</code>	A fitted model object, such as <code>kk_model</code> or <code>jvn_model</code> .
<code>...</code>	Additional arguments passed to methods.
<code>filter</code>	Which state estimates to return: <code>"smoothed"</code> (default) uses the full sample, <code>"filtered"</code> uses information available up to each date, and <code>"all"</code> returns both.
<code>state</code>	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.jvn_model()`, `coef.kk_model()`, `fitted.jvn_model()`, `fitted.kk_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.jvn_model()`, `logLik.kk_model()`, `nobs.jvn_model()`, `nobs.kk_model()`, `plot.jvn_model()`, `plot.kk_model()`, `predict.jvn_model()`, `predict.kk_model()`, `print.jvn_model()`, `print.kk_model()`, `residuals.jvn_model()`, `residuals.kk_model()`, `summary.jvn_model()`, `summary.kk_model()`, `vcov.jvn_model()`, `vcov.kk_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.jvn_model

Summary method for JVN model objects

Description

Print a compact summary of a fitted `jvn_model`, including convergence status, information criteria, and parameter estimates.

Usage

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

Arguments

`object` An object of class `jvn_model`.

`...` Unused; included for method compatibility.

Value

The input object, invisibly.

See Also

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

summary.kk_model

*Summary Method for KK Model***Description**

Computes and displays a summary of the results from a Kishor-Koenig (KK) model fit, including convergence status, information criteria, and parameter estimates.

Usage

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

Arguments

object	An object of class <code>kk_model</code> .
...	Additional arguments passed to or from other methods.

Value

The function returns the input object invisibly.

See Also

Other revision nowcasting: `coef.jvn_model()`, `coef.kk_model()`, `fitted.jvn_model()`, `fitted.kk_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.jvn_model()`, `logLik.kk_model()`, `nobs.jvn_model()`, `nobs.kk_model()`, `plot.jvn_model()`, `plot.kk_model()`, `predict.jvn_model()`, `predict.kk_model()`, `print.jvn_model()`, `print.kk_model()`, `residuals.jvn_model()`, `residuals.kk_model()`, `states()`, `summary.jvn_model()`, `vcov.jvn_model()`, `vcov.kk_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
h <- 2
result <- kk_nowcast(df, e, h = h, model = "Kishor-Koenig", method = "MLE")
summary(result)
```

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

<code>object</code>	An output object from the <code>get_first_efficient_release</code> function. The object must be of class <code>lst_efficient</code> .
<code>...</code>	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_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_pubdate	<i>Summary Method for Publication Date Vintages</i>
---------------------	---

Description

Summary Method for Publication Date Vintages

Usage

```

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

```

Arguments

object	An object of class <code>tbl_pubdate</code> .
...	Additional arguments (not used).

Value

The function returns a summary tibble invisibly.

See Also

Other helpers: [print.tbl_pubdate\(\)](#), [print.tbl_release\(\)](#), [summary.tbl_release\(\)](#), [tbl_sum.tbl_pubdate\(\)](#), [tbl_sum.tbl_release\(\)](#), [validate_vintages\(\)](#), [vintages_long\(\)](#), [vintages_wide\(\)](#)

Examples

```

df <- dplyr::filter(reviser::gdp, id == "US")
# Wide format
wide_data <- vintages_wide(df)
summary(wide_data$US)

# Long format
summary(get_revisions(df, interval = 1))

```

summary.tbl_release	<i>Summary Method for Release Vintages</i>
---------------------	--

Description

Summary Method for Release Vintages

Usage

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

Arguments

object	An object of class tbl_release.
...	Additional arguments (not used).

Value

The function returns a summary tibble invisibly.

See Also

Other helpers: [print.tbl_pubdate\(\)](#), [print.tbl_release\(\)](#), [summary.tbl_pubdate\(\)](#), [tbl_sum.tbl_pubdate\(\)](#), [tbl_sum.tbl_release\(\)](#), [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)
```

tbl_sum.tbl_pubdate	<i>Tibble Summary for Publication Date Vintages</i>
---------------------	---

Description

Provides a custom header for objects of class tbl_pubdate when printed. This method is called automatically by pillar when printing tibbles.

Usage

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

Arguments

x An object of class `tbl_pubdate`.
 ... 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.

See Also

Other helpers: [print.tbl_pubdate\(\)](#), [print.tbl_release\(\)](#), [summary.tbl_pubdate\(\)](#), [summary.tbl_release\(\)](#), [tbl_sum.tbl_release\(\)](#), [validate_vintages\(\)](#), [vintages_long\(\)](#), [vintages_wide\(\)](#)

Examples

```
df <- dplyr::filter(reviser::gdp, id == "US")
wide_data <- vintages_wide(df)
pillar::tbl_sum(wide_data$US)
```

tbl_sum.tbl_release	<i>Tibble Summary for Release Vintages</i>
---------------------	--

Description

Provides a custom header for objects of class `tbl_release` when printed.

Usage

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

Arguments

x An object of class `tbl_release`.
 ... Additional arguments (unused).

Value

A named character vector where names are labels and values are the corresponding information.

See Also

Other helpers: `print.tbl_pubdate()`, `print.tbl_release()`, `summary.tbl_pubdate()`, `summary.tbl_release()`, `tbl_sum.tbl_pubdate()`, `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)
```

 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

<code>base_size</code>	Numeric. The base font size for the theme. Default is 12.
<code>legend.position</code>	Character. Position of the legend. Default is "bottom".
<code>legend.direction</code>	Character. Direction of the legend. Default is "horizontal".
<code>...</code>	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_pubdate()`, `plot.tbl_release()`, `plot_vintages()`

Other revision graphs: `plot.tbl_pubdate()`, `plot.tbl_release()`, `plot_vintages()`

Other revision graphs: `plot.tbl_pubdate()`, `plot.tbl_release()`, `plot_vintages()`

Other revision graphs: `plot.tbl_pubdate()`, `plot.tbl_release()`, `plot_vintages()`

Examples

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

<code>validate_vintages</code>	<i>Vintages data classes and their validation</i>
--------------------------------	---

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.

Methods

Both classes support `print()`, `summary()` and `plot()`. The plot methods dispatch to `plot_vintages()`, which remains available for direct use when you want to pass its arguments explicitly. `print()` uses a pillar header that reports the layout, the number of periods and the number of vintages.

See Also

Other helpers: `print.tbl_pubdate()`, `print.tbl_release()`, `summary.tbl_pubdate()`, `summary.tbl_release()`, `tbl_sum.tbl_pubdate()`, `tbl_sum.tbl_release()`, `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.jvn_model

*Extract the parameter covariance matrix of a JVN model***Description**

Extract the parameter covariance matrix of a JVN model

Usage

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

Arguments

object	An object of class jvn_model.
...	Ignored.

Value

The estimated parameter covariance matrix.

See Also

Other revision nowcasting: [coef.jvn_model\(\)](#), [coef.kk_model\(\)](#), [fitted.jvn_model\(\)](#), [fitted.kk_model\(\)](#), [jvn_nowcast\(\)](#), [kk_nowcast\(\)](#), [logLik.jvn_model\(\)](#), [logLik.kk_model\(\)](#), [nobs.jvn_model\(\)](#), [nobs.kk_model\(\)](#), [plot.jvn_model\(\)](#), [plot.kk_model\(\)](#), [predict.jvn_model\(\)](#), [predict.kk_model\(\)](#), [print.jvn_model\(\)](#), [print.kk_model\(\)](#), [residuals.jvn_model\(\)](#), [residuals.kk_model\(\)](#), [states\(\)](#), [summary.jvn_model\(\)](#), [summary.kk_model\(\)](#), [vcov.kk_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)
vcov(fit)
```

vcov.kk_model	<i>Extract the parameter covariance matrix of a KK model</i>
---------------	--

Description

Extract the parameter covariance matrix of a KK model

Usage

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

Arguments

object	An object of class <code>kk_model</code> .
...	Ignored.

Value

The estimated parameter covariance matrix.

See Also

Other revision nowcasting: `coef.jvn_model()`, `coef.kk_model()`, `fitted.jvn_model()`, `fitted.kk_model()`, `jvn_nowcast()`, `kk_nowcast()`, `logLik.jvn_model()`, `logLik.kk_model()`, `nobs.jvn_model()`, `nobs.kk_model()`, `plot.jvn_model()`, `plot.kk_model()`, `predict.jvn_model()`, `predict.kk_model()`, `print.jvn_model()`, `print.kk_model()`, `residuals.jvn_model()`, `residuals.kk_model()`, `states()`, `summary.jvn_model()`, `summary.kk_model()`, `vcov.jvn_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	<i>Convert Vintages Data to Long Format</i>
---------------	---

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 data frame or tibble. If the input is a list of wide-format data frames, the output will be a single combined long-format data frame.

See Also

Other helpers: [print.tbl_pubdate\(\)](#), [print.tbl_release\(\)](#), [summary.tbl_pubdate\(\)](#), [summary.tbl_release\(\)](#), [tbl_sum.tbl_pubdate\(\)](#), [tbl_sum.tbl_release\(\)](#), [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

<code>df</code>	A data frame or tibble containing vintages data in long format.
<code>names_from</code>	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_pubdate\(\)](#), [print.tbl_release\(\)](#), [summary.tbl_pubdate\(\)](#), [summary.tbl_release\(\)](#), [tbl_sum.tbl_pubdate\(\)](#), [tbl_sum.tbl_release\(\)](#), [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, 9
- * **dataset**
 - gdp, 9
- * **helpers**
 - print.tbl_pubdate, 45
 - print.tbl_release, 46
 - summary.tbl_pubdate, 54
 - summary.tbl_release, 55
 - tbl_sum.tbl_pubdate, 55
 - tbl_sum.tbl_release, 56
 - validate_vintages, 58
 - vintages_long, 62
 - vintages_wide, 63
- * **revision analysis**
 - diagnose, 5
 - diagnose.revision_summary, 6
 - get_first_efficient_release, 11
 - get_revision_analysis, 20
 - print.lst_efficient, 43
 - print.revision_summary, 44
 - summary.lst_efficient, 51
 - summary.revision_summary, 53
- * **revision graphs**
 - plot.tbl_pubdate, 35
 - plot.tbl_release, 36
 - plot_vintages, 36
 - theme_reviser, 57
- * **revision nowcasting**
 - coef.jvn_model, 3
 - coef.kk_model, 4
 - fitted.jvn_model, 7
 - fitted.kk_model, 8
 - jvn_nowcast, 23
 - kk_nowcast, 26
 - logLik.jvn_model, 29
 - logLik.kk_model, 30
 - nobs.jvn_model, 31
 - nobs.kk_model, 32
 - plot.jvn_model, 33
 - plot.kk_model, 34
 - predict.jvn_model, 39
 - predict.kk_model, 40
 - print.jvn_model, 41
 - print.kk_model, 42
 - residuals.jvn_model, 46
 - residuals.kk_model, 47
 - states, 48
 - summary.jvn_model, 49
 - summary.kk_model, 50
 - vcov.jvn_model, 60
 - vcov.kk_model, 61
- * **revision utilities**
 - get_days_to_release, 10
 - get_first_release, 13
 - get_fixed_release, 14
 - get_latest_release, 15
 - get_nth_release, 16
 - get_releases_by_date, 17
 - get_revisions, 18
- coef.jvn_model, 3, 4, 8, 9, 25, 28–34, 39–42, 47–51, 60, 61
- coef.kk_model, 3, 4, 8, 9, 25, 28–34, 39–42, 47–51, 60, 61
- colors_reviser(theme_reviser), 57
- diagnose, 5, 7, 13, 22, 43, 44, 52, 53
- diagnose.revision_summary, 5, 6, 13, 22, 43, 44, 52, 53
- fitted.jvn_model, 3, 4, 7, 9, 25, 28–34, 39–42, 47–51, 60, 61
- fitted.kk_model, 3, 4, 8, 8, 25, 28–34, 39–42, 47–51, 60, 61
- gdp, 9
- get_days_to_release, 10, 14, 15, 17–19
- get_first_efficient_release, 5, 7, 11, 22, 43, 44, 52, 53

get_first_release, [11](#), [13](#), [15](#), [17–19](#)
 get_first_release(), [58](#)
 get_fixed_release, [11](#), [14](#), [14](#), [15](#), [17–19](#)
 get_fixed_release(), [58](#)
 get_latest_release, [11](#), [14](#), [15](#), [15](#), [17–19](#)
 get_latest_release(), [58](#)
 get_nth_release, [11](#), [14](#), [15](#), [16](#), [18](#), [19](#)
 get_nth_release(), [58](#)
 get_releases_by_date, [11](#), [14](#), [15](#), [17](#), [17](#), [19](#)
 get_releases_by_date(), [58](#)
 get_revision_analysis, [5](#), [7](#), [13](#), [20](#), [43](#), [44](#),
 [52](#), [53](#)
 get_revisions, [11](#), [14](#), [15](#), [17](#), [18](#), [18](#)
 get_revisions(), [58](#)

 jvn_nowcast, [3](#), [4](#), [8](#), [9](#), [23](#), [28–34](#), [39–42](#),
 [47–51](#), [60](#), [61](#)
 jvn_nowcast(), [39](#)

 kk_nowcast, [3](#), [4](#), [8](#), [9](#), [25](#), [26](#), [29–34](#), [39–42](#),
 [47–51](#), [60](#), [61](#)
 kk_nowcast(), [40](#)

 logLik.jvn_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28](#), [29](#),
 [30–34](#), [39–42](#), [47–51](#), [60](#), [61](#)
 logLik.kk_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28](#), [29](#), [30](#),
 [31–34](#), [39–42](#), [47–51](#), [60](#), [61](#)

 nobs.jvn_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–30](#), [31](#),
 [32–34](#), [39–42](#), [47–51](#), [60](#), [61](#)
 nobs.kk_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–31](#), [32](#), [33](#),
 [34](#), [39–42](#), [47–51](#), [60](#), [61](#)

 plot(), [59](#)
 plot.jvn_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–32](#), [33](#), [34](#),
 [39–42](#), [47–51](#), [60](#), [61](#)
 plot.kk_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–33](#), [34](#),
 [39–42](#), [47–51](#), [60](#), [61](#)
 plot.tbl_pubdate, [35](#), [36](#), [38](#), [58](#)
 plot.tbl_release, [35](#), [36](#), [38](#), [58](#)
 plot_vintages, [35](#), [36](#), [36](#), [58](#)
 plot_vintages(), [59](#)
 predict.jvn_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–34](#), [39](#),
 [40–42](#), [47–51](#), [60](#), [61](#)
 predict.kk_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–34](#), [39](#),
 [40](#), [41](#), [42](#), [47–51](#), [60](#), [61](#)
 print(), [59](#)
 print.jvn_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–34](#), [39](#),
 [40](#), [41](#), [42](#), [47–51](#), [60](#), [61](#)

 print.kk_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–34](#), [39–41](#),
 [42](#), [47–51](#), [60](#), [61](#)
 print.lst_efficient, [5](#), [7](#), [13](#), [22](#), [43](#), [44](#),
 [52](#), [53](#)
 print.revision_summary, [5](#), [7](#), [13](#), [22](#), [43](#),
 [44](#), [52](#), [53](#)
 print.tbl_pubdate, [45](#), [46](#), [54–57](#), [59](#), [62](#), [63](#)
 print.tbl_release, [45](#), [46](#), [54–57](#), [59](#), [62](#), [63](#)

 residuals.jvn_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–34](#),
 [39–42](#), [46](#), [48–51](#), [60](#), [61](#)
 residuals.kk_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–34](#),
 [39–42](#), [47](#), [47](#), [49–51](#), [60](#), [61](#)
 reviser-vintages-classes
 (validate_vintages), [58](#)

 scale_color_reviser(theme_reviser), [57](#)
 scale_color_reviser(), [38](#)
 scale_fill_reviser(theme_reviser), [57](#)
 scale_fill_reviser(), [38](#)
 states, [3](#), [4](#), [8](#), [9](#), [25](#), [28–34](#), [39–42](#), [47](#), [48](#),
 [48](#), [50](#), [51](#), [60](#), [61](#)
 stats::AIC(), [29](#), [30](#)
 stats::BIC(), [29](#), [30](#)
 summary(), [59](#)
 summary.jvn_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–34](#),
 [39–42](#), [47–49](#), [49](#), [51](#), [60](#), [61](#)
 summary.kk_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–34](#),
 [39–42](#), [47–50](#), [50](#), [60](#), [61](#)
 summary.lst_efficient, [5](#), [7](#), [13](#), [22](#), [43](#), [44](#),
 [51](#), [53](#)
 summary.revision_summary, [5](#), [7](#), [13](#), [22](#), [43](#),
 [44](#), [52](#), [53](#)
 summary.tbl_pubdate, [45](#), [46](#), [54](#), [55–57](#), [59](#),
 [62](#), [63](#)
 summary.tbl_release, [45](#), [46](#), [54](#), [55](#), [56](#), [57](#),
 [59](#), [62](#), [63](#)

 tbl_sum.tbl_pubdate, [45](#), [46](#), [54](#), [55](#), [55](#), [57](#),
 [59](#), [62](#), [63](#)
 tbl_sum.tbl_release, [45](#), [46](#), [54–56](#), [56](#), [59](#),
 [62](#), [63](#)
 theme_reviser, [35](#), [36](#), [38](#), [57](#)
 theme_reviser(), [38](#)

 validate_vintages, [45](#), [46](#), [54–57](#), [58](#), [62](#), [63](#)
 vcov.jvn_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–34](#), [39–42](#),
 [47–51](#), [60](#), [61](#)
 vcov.kk_model, [3](#), [4](#), [8](#), [9](#), [25](#), [28–34](#), [39–42](#),
 [47–51](#), [60](#), [61](#)

vintages_long, [45](#), [46](#), [54–57](#), [59](#), [62](#), [63](#)
vintages_long(), [58](#)
vintages_wide, [45](#), [46](#), [54–57](#), [59](#), [62](#), [63](#)
vintages_wide(), [58](#)