Package 'koma'

July 29, 2026

Title Bayesian Simultaneous Equation Models for Forecasting

Version 0.3.1

Description Estimate and forecast Bayesian simultaneous equation models for macroeconomic time series. Provides tools to specify systems of behavioral equations and accounting identities, transform and manage time series, simulate from the posterior using a Metropolis-within-Gibbs sampler, and generate unconditional and conditional forecasts with user-defined priors and restrictions. Methods are described in Rathke A. and Sarferaz S. (forthcoming) ``Bayesian Estimation of Simultaneous Equations Model".

URL https://timothymerlin.github.io/koma/,
https://github.com/TimothyMerlin/koma

License GPL (>= 3)

Suggests devtools (>= 2.4.5), ggplot2 (>= 3.5.0), usethis (>= 3.1.0), pkgdown (>= 2.0.7), knitr (>= 1.43), rmarkdown (>= 2.24), future (>= 1.33.0), parallelly (>= 1.36.0), withr (>= 3.0.0), texreg (>= 1.39.4), testthat (>= 3.2.3), plotly (>= 4.11.0), webshot2 (>= 0.1.2)

Encoding UTF-8

RoxygenNote 7.3.3

Config/testthat/edition 3

Imports cli (>= 3.6.3), doFuture (>= 1.0.0), foreach (>= 1.5.2), glue (>= 1.8.0), Matrix (>= 1.5.4.1), methods, progressr (>= 0.15.0), purrr (>= 1.0.4), rlang (>= 1.1.6), stats (>= 4.0.2), tempdisagg (>= 1.1), utils (>= 4.4.1)

Depends R (>= 4.1.0)

LazyData true

VignetteBuilder knitr

BugReports https://github.com/TimothyMerlin/koma/issues

NeedsCompilation no

Author Samad Sarferaz [aut, cph] (ORCID:
<https://orcid.org/0000-0001-9283-3293>),

Laurent Florin [aut],
Merlin Scherer [aut, cre, cph] (ORCID:
<https://orcid.org/0009-0005-7479-168X>),
Andrew D. Martin [ctb, cph] (Adapted rwish/riwish code from MCMCpack),
Kevin M. Quinn [ctb, cph] (Adapted rwish/riwish code from MCMCpack),
Jong Hee Park [ctb, cph] (Adapted rwish/riwish code from MCMCpack),
Bill Venables [ctb, cph] (Adapted multivariate_norm code from
MASS::mvrnorm),
Brian D. Ripley [ctb, cph] (Adapted multivariate_norm code from
MASS::mvrnorm)

Maintainer Merlin Scherer <scherer@kof.ethz.ch>

Repository CRAN

Date/Publication 2026-07-29 17:00:02 UTC

Contents

`acf_plot` *Autocorrelation Function Plots for koma_estimate Objects*

Description

Visualize autocorrelation functions (ACF) for coefficient draws from a `koma_estimate` object. By default, beta, gamma, and sigma draws are shown when available.

Usage

```
acf_plot(x, ...)
```

Arguments

`x`	A `koma_estimate` object.
`...`	Additional arguments controlling the plot. See Details.

Details

Additional arguments supported in `...`:

variables Optional character vector of endogenous variables to plot.

params Optional character vector of parameter groups to plot (e.g., "beta", "gamma", "sigma"). Defaults to all available.

thin Optional integer thinning interval for the stored draws. Default is 1 (no thinning).

max_draws Optional integer cap on the number of draws used per ACF. When set, the most recent draws are kept.

max_lag Optional integer maximum lag for ACF computation. When NULL, defaults to `min(30, n_draws - 1)` per series.

conf_level Optional numeric confidence level in `(0, 1)` used for ACF significance bands. Default is `0.95`.

scales Facet scale option passed to `ggplot2::facet_wrap`. Default is "fixed".

facet_ncol Optional integer number of columns for facets.

interactive Logical. If TRUE and plotly is available, return an interactive plot via `plotly::ggplotly`. Default is FALSE.

ACF values are computed with `stats::acf(..., plot = FALSE)` for each retained coefficient draw series.

Note: `sigma` plots use `omega_tilde_jw` and show only variances (no covariances) from each covariance draw.

The red dashed horizontal lines show approximate significance bounds $\pm z_{1-\alpha/2}/\sqrt{n}$ for zero autocorrelation, where $\alpha = 1 - \text{conf\_level}$ and n is the number of retained draws for the corresponding coefficient series.

Value

A ggplot object, or a plotly object when `interactive = TRUE` and plotly is available.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  data("simulated_sem")
  set.seed(11)

  fit <- estimate(
    ts_data = simulated_sem$ts_data,
    sys_eq = simulated_sem$sys_eq,
    dates = simulated_sem$dates,
    options = list(gibbs = list(ndraws = 10))
  )
  acf_plot(fit, params = "beta", max_lag = 12)
}
```

---

as_list *Convert an mets koma_ts object to a list*

Description

Convert an mets koma_ts object to a list

Usage

```
as_list(x, ...)

## S3 method for class 'mts'
as_list(x, ...)

## S3 method for class 'list'
as_list(x, ...)
```

Arguments

x	An object to be converted.
...	Additional arguments.

Value

A list with ets koma_ts objects.

Examples

```
x <- as_mets(list(
  y = as_ets(ts(1:8, start = c(2020, 1), frequency = 4)),
  z = as_ets(ts(11:18, start = c(2020, 1), frequency = 4))
))
as_list(x)
```

---

as_mets *Convert a list object with ets koma_ts objects to a koma_ts multivariate time series (mets)*

Description

Convert a list object with ets koma_ts objects to a koma_ts multivariate time series (mets)

Usage

```
as_mets(x, ...)

## S3 method for class 'list'
as_mets(x, ...)
```

Arguments

x	An object to be converted.
...	Additional arguments.

Value

A koma_ts multivariate time series object.

Examples

```
x <- list(
  y = as_ets(ts(1:8, start = c(2020, 1), frequency = 4)),
  z = as_ets(ts(11:18, start = c(2020, 1), frequency = 4))
)
as_mets(x)
```

---

concat *Concatenate two time series*

Description

Concatenate two time series

Usage

```
concat(x, y, ...)

## S3 method for class 'ts'
concat(x, y, ...)

## S3 method for class 'list'
concat(x, y, ...)

## S3 method for class 'mts'
concat(x, y, ...)
```

Arguments

x	A koma_ts object.
y	A koma_ts object to be concatenated to x.
...	arguments passed to methods (unused for the default method).

Value

A koma_ts object containing x followed by y.

Examples

```
x <- as_ets(ts(c(100, 101, 102), start = c(2020, 1), frequency = 4))
y <- as_ets(ts(c(103, 104), start = c(2020, 4), frequency = 4))
concat(x, y)
```

`estimate` *Estimate the Simultaneous Equations Model (SEM)*

Description

Estimate a system of simultaneous equations model (SEM) using a Bayesian approach. This function incorporates Gibbs sampling and allows for both density and point forecasts.

Usage

```
estimate(
  ts_data,
  sys_eq,
  dates,
  ...,
  options = list(gibbs = list(), fill = list(method = "mean")),
  estimates = NULL
)

## S3 method for class 'list'
estimate(
  ts_data,
  sys_eq,
  dates,
  ...,
  options = list(gibbs = list(), fill = list(method = "mean")),
  estimates = NULL
)
```

Arguments

`ts_data`	Time-series data set for the estimation.
`sys_eq`	A `koma_seq` object (system_of_equations) containing details about the system of equations used in the model.
`dates`	Key-value list for date ranges in various model operations.
`...`	Additional parameters.
`options`	Optional settings for estimation. Use `list(gibbs = list(), fill = list(method = "mean"))`. Elements: • `gibbs`: Gibbs sampler settings (see Gibbs Sampler Specifications). • `fill$method`: "mean" or "median" used to fill ragged edges during estimation. See Gibbs Sampler Specifications.
`estimates`	Optional. A `koma_estimate` object (see `estimate`) containing the estimates of the previously estimated simultaneous equations model. Use this parameter when some equations of the system need to be re-estimated.

Details

After estimation, use `summary` for a full table of posterior summaries (with optional credible intervals and texreg output) and `print` for a concise console-friendly overview of the estimated system.

Value

An object of class `koma_estimate`.

An object of class `koma_estimate`is a list containing the following elements:

estimates The estimated parameters and other relevant information obtained from the model.

sys_eq A `koma_seq` object containing details about the system of equations used in the model.

ts_data The time-series data used for the estimation, with any `NA` values removed and lagged variables created.

y_matrix The Y matrix constructed from the balanced data, used in the estimation process.

x_matrix The X matrix constructed from the balanced data, used in the estimation process.

gibbs_specifications The specifications used for the Gibbs sampling.

dates The date ranges used during estimation.

Parallel

This function provides the option for parallel computing through the `future::plan()` function. For a detailed example on executing `estimate` in parallel, see the vignette: `vignette("parallel")`. For more details, see the future package documentation.

Gibbs Sampler Specifications

- `ndraws`: Integer specifying the number of Gibbs sampler draws. Default is 2000.
- `burnin_ratio`: Numeric specifying the ratio for the burn-in period. Default is 0.5.
- `nstore`: Integer specifying the frequency of stored draws. Default is 1.
- `tau`: Numeric tuning parameter for enforcing an acceptance rate. Default is 1.1.

See Also

- To create a `koma_seq` object see `system_of_equations`.
- For a comprehensive example of using `estimate`, see `vignette("koma")`.
- Related functions within the package that may be of interest: `forecast`.

Examples

```
data("simulated_sem")
set.seed(11)

fit <- estimate(
  ts_data = simulated_sem$ts_data,
  sys_eq = simulated_sem$sys_eq,
  dates = simulated_sem$dates,
```

```
  options = list(gibbs = list(ndraws = 10))
)
print(fit)
```

`extract.koma_estimate` *Extract a texreg summary from a koma_estimate*

Description

Builds one or more **texreg** objects from a `koma_estimate`, so results can be rendered with `texreg::screenreg()` or similar helpers.

Usage

```
extract.koma_estimate(
  model,
  variables = NULL,
  central_tendency = "mean",
  ci_low = 5,
  ci_up = 95,
  digits = 2,
  ...
)
```

Arguments

`model`	A `koma_estimate` object.
`variables`	Optional character vector of endogenous variables to include. Defaults to all variables in `model$estimates`.
`central_tendency`	Central tendency used when summarizing estimates (e.g., "mean", "median"). Defaults to "mean".
`ci_low`	Lower bound (percent) for credible intervals. Defaults to 5.
`ci_up`	Upper bound (percent) for credible intervals. Defaults to 95.
`digits`	Number of digits to round numeric values. Defaults to 2.
`...`	Unused. Included for `texreg::extract()` compatibility.

Value

A `texreg` object when one variable is requested, otherwise a named list of `texreg` objects.

See Also

`summary.koma_estimate` for summary output with optional **texreg** formatting.

Examples

```
if (requireNamespace("texreg", quietly = TRUE)) {
  data("simulated_sem")
  set.seed(11)

  fit <- estimate(
    ts_data = simulated_sem$ts_data,
    sys_eq = simulated_sem$sys_eq,
    dates = simulated_sem$dates,
    options = list(gibbs = list(ndraws = 10))
  )
  texreg::extract(fit, variables = "consumption")
}
```

`filter_by_attribute` *Filter a koma_ts object by attribute value*

Description

Filter a koma_ts object by attribute value

Usage

```
filter_by_attribute(x, attribute, value, var = NULL, ...)

## S3 method for class 'mts'
filter_by_attribute(x, attribute, value, var = NULL, ...)

## S3 method for class 'list'
filter_by_attribute(x, attribute, value, var = NULL, ...)
```

Arguments

`x`	A koma_ts object or a list of koma_ts objects.
`attribute`	The attribute name to filter by.
`value`	The attribute value to match.
`var`	An optional variable to filter by.
`...`	arguments passed to methods (unused for the default method).

Value

A koma_ts object or list with the matching series.

Examples

```
x <- as_ets(
  ts(1:8, start = c(2020, 1), frequency = 4),
  series_type = "level",
  method = "diff_log"
)
filter_by_attribute(
  list(x = x),
  attribute = "series_type",
  value = "level"
)
```

---

forecast *Forecast the Simultaneous Equations Model (SEM)*

Description

This function produces forecasts for the SEM.

Usage

```
forecast(
  estimates,
  dates,
  ...,
  restrictions = NULL,
  options = list(approximate = FALSE, probs = NULL, fill = list(method = "mean"),
    conditional_innov_method = "projection")
)

## S3 method for class 'koma_estimate'
forecast(
  estimates,
  dates,
  ...,
  restrictions = NULL,
  options = list(approximate = FALSE, probs = NULL, fill = list(method = "mean"),
    conditional_innov_method = "projection")
)
```

Arguments

estimates	A koma_estimate object (estimate) containing the estimates for the simultaneous equations model, as well as a list of time series and a koma_seq object (system_of_equations) that were used in the estimation.
dates	Key-value list for date ranges in various model operations.
...	Additional parameters.

`restrictions` List of model constraints. Default is empty.

`options` Optional settings for forecasting. Use `list(approximate = FALSE, probs = NULL, fill = list(method = "mean"),conditional_innov_method = "projection")`. Elements:

- `approximate`: Logical. If FALSE (default), compute point forecasts from predictive draws. If TRUE, compute point forecasts from the mean/median of coefficient draws (fast approximation).
- `probs`: Numeric vector of quantile probabilities. If NULL, no quantiles are returned. When `approximate = FALSE` and `probs` is NULL, defaults to `setdiff(get_quantiles(), 0.5)`.
- `fill$method`: "mean" or "median" used for conditional fill before forecasting.
- `conditional_innov_method`: Method for drawing conditional innovations. One of `"projection"` (default) or `"eigen"`.

Details

The `forecast` function for SEM uses the estimates from the `koma_estimate` object to produce point forecasts and, optionally, quantile forecasts. When `options$approximate` is FALSE (default), point forecasts are computed from the predictive draws (with quantiles controlled by `options$probs`). When TRUE, point forecasts are computed from the mean and median of the coefficient draws for faster, approximate results.

The returned `koma_forecast` object keeps forecasts as named lists of `koma_ts` (for `mean`, `median`, and `quantiles`) alongside the input data and matrices used to produce them.

Use the `print` method to print a the forecast results, use the `plot` method, to visualize the forecasts and prediction intervals.

Value

An object of class `koma_forecast`.

An object of class `koma_forecast` is a list containing the following elements:

mean Mean point forecasts as a list of time series of class `koma_ts`.

median Median point forecasts as a list of time series of class `koma_ts`.

quantiles A list of quantiles, where each element is named according to the quantile (e.g., "q_5", "q_50", "q_95"), and contains the forecasts for that quantile. This element is NULL if `quantiles = FALSE`.

ts_data Time-series data set used in forecasting.

y_matrix The Y matrix constructed from the balanced data up to the current quarter, used for forecasting.

x_matrix The X matrix used for forecasting.

Parallel

This function provides the option for parallel computing through the `future::plan()` function. For a detailed example on executing `estimate` in parallel, see the vignette: `vignette("parallel")`. For more details, see the future package documentation.

See Also

- For a comprehensive example of using `forecast`, see `vignette("koma")`.
- Related functions within the package that may be of interest: `estimate`.

Examples

```
data("simulated_sem")

dates <- list(
  current = c(2024, 4),
  estimation = simulated_sem$dates$estimation,
  forecast = list(start = c(2025, 1), end = c(2025, 4))
)

ts_data <- simulated_sem$ts_data
# Endogenous series must stop at the last observed quarter before forecasting.
ts_data[simulated_sem$sys_eq$endogenous_variables] <- lapply(
  simulated_sem$sys_eq$endogenous_variables,
  function(x) {
    stats::window(ts_data[[x]], end = dates$current)
  }
)

set.seed(11)
fit <- estimate(
  ts_data = ts_data,
  sys_eq = simulated_sem$sys_eq,
  dates = dates,
  options = list(gibbs = list(ndraws = 10))
)
fc <- forecast(fit, dates = dates)
print(fc)
```

`format.koma_seq` *Format System of Equations*

Description

This function formats an object of class `koma_seq` for better readability. It formats the equations to ensure proper spacing around operators and aligns the equations for a cleaner display.

Usage

```
## S3 method for class 'koma_seq'
format(x, ...)
```

Arguments

x	An object of class koma_seq.
...	Additional arguments passed to or from other methods.

Value

A character vector of the formatted equations.

---

generate_sample_data *Generate Data for the Simultaneous Equations Model*

Description

Compute Y and X matrix (data) Use reduced form of the model to compute Y $Y = XB\Gamma^{-1} + U\Gamma^{-1}$

Usage

```
generate_sample_data(
  sample_size,
  sample_start,
  burnin,
  gamma_matrix,
  beta_matrix,
  sigma_matrix,
  endogenous_variables,
  exogenous_variables,
  predetermined_variables
)
```

Arguments

sample_size	The number of observations.
sample_start	A vector of format c(YEAR, QUARTER) representing
burnin	The number of observations to discard to mitigate the dependency on starting values.
gamma_matrix	A matrix that defines the relationships between the variables.
beta_matrix	A matrix that defines the coefficients of the equation.
sigma_matrix	A matrix that defines the standard deviation of the error terms.
endogenous_variables	A vector of endogenous variables.
exogenous_variables	A vector of exogenous variables.
predetermined_variables	A vector of predetermined variables.

Details

It creates a sample of data where the relationship between the variables is determined by the defined parameters.

Value

A list containing three elements:

- y_matrix: The dependent variable matrix with newly generated data.
- x_matrix: The matrix of exogenous and predetermined variables with newly generated data.
- ts_data: A list containing time series in growth rates.

Examples

```
gamma_matrix <- matrix(1, nrow = 1)
beta_matrix <- matrix(c(0.2, 0.5, 0.3), nrow = 3)
sigma_matrix <- matrix(0.01, nrow = 1)

sample <- generate_sample_data(
  sample_size = 12,
  sample_start = c(2000, 1),
  burnin = 4,
  gamma_matrix = gamma_matrix,
  beta_matrix = beta_matrix,
  sigma_matrix = sigma_matrix,
  endogenous_variables = "y",
  exogenous_variables = "x",
  predetermined_variables = "y.L(1)"
)
names(sample)
```

---

hdi *Highest Density Intervals from Draws*

Description

Computes highest density intervals (HDIs) for a numeric sample. The HDI is defined as the shortest interval containing a target probability mass.

Usage

```
hdi(x, ...)

## Default S3 method:
hdi(x, probs = c(0.5, 0.99), ...)
```

Arguments

x	A numeric vector of draws.
...	Unused.
probs	Numeric vector of target mass levels. Values in $(0, 1]$ or $[0, 100]$ are accepted. Default is c(0.5, 0.99).

Value

A list with class "koma_hdi" containing:

intervals Named list of matrices with columns lower and upper, one matrix per probs level.

mode Sample median of the draws (used as a center reference).

cutoff Named numeric vector of NA values (not defined for HDI).

mass Named numeric vector of achieved mass for each level.

probs Numeric vector of target masses in (0, 1].

Examples

```
x <- rnorm(1000)
hdi(x, probs = c(0.5, 0.9))

data("simulated_sem")
set.seed(11)

fit <- estimate(
  ts_data = simulated_sem$ts_data,
  sys_eq = simulated_sem$sys_eq,
  dates = simulated_sem$dates,
  options = list(gibbs = list(ndraws = 10))
)
hdi_fit <- hdi(fit, probs = c(0.5, 0.9))
names(hdi_fit$intervals)
```

---

hdi.koma_estimate *Highest Density Intervals for koma_estimate Objects*

Description

Computes highest density intervals (HDIs) for coefficient draws from a koma_estimate object.

Usage

```
## S3 method for class 'koma_estimate'
hdi(x, variables = NULL, probs = c(0.5, 0.99), include_sigma = FALSE, ...)
```

Arguments

x	A koma_estimate object.
variables	Optional character vector of endogenous variables to include. Defaults to all variables in object$estimates.
probs	Numeric vector of target mass levels. Values in $(0, 1]$ or $[0, 100]$ are accepted. Default is c(0.5, 0.99).
include_sigma	Logical. If TRUE, compute HDIs for the error variance parameter (omega). Default is FALSE.
...	Unused.

Value

A list with class c("koma_estimate_hdi", "koma_hdi") containing HDI intervals per coefficient.

---

hdi.koma_forecast *Highest Density Intervals for koma_forecast Objects*

Description

Computes highest density intervals (HDIs) for predictive draws in a koma_forecast object.

Usage

```
## S3 method for class 'koma_forecast'
hdi(x, variables = NULL, probs = c(0.5, 0.99), ...)
```

Arguments

x	A koma_forecast object.
variables	Optional character vector of endogenous variables to include. Defaults to all variables in x$forecasts.
probs	Numeric vector of target mass levels. Values in $(0, 1]$ or $[0, 100]$ are accepted. Default is c(0.5, 0.99).
...	Unused.

Value

A list with class c("koma_forecast_hdi", "koma_hdi") containing HDI intervals per variable and horizon, along with per-horizon centers and achieved masses.

hdr *Highest Density Regions from a Kernel Density Estimate*

Description

Computes highest density regions (HDRs) for a numeric sample using a kernel density estimate. For multimodal densities, the HDR can consist of multiple disjoint intervals.

Usage

```
hdr(x, ...)

## Default S3 method:
hdr(
  x,
  probs = c(0.5, 0.99),
  n_grid = 4096,
  integration = c("monte_carlo", "grid"),
  mc_use_observed = FALSE,
  mc_draws = NULL,
  mc_quantile_type = 7,
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  ...
)
```

Arguments

x	A numeric vector of draws.
...	Additional arguments forwarded to density. Do not pass n, bw, adjust, or kernel here.
probs	Numeric vector of target mass levels. Values in $(0, 1]$ or $[0, 100]$ are accepted. Default is c(0.5, 0.99).
n_grid	Number of grid points for the KDE. Default is 4096.
integration	Integration approach for the HDR cutoff. Use "grid" for grid-based integration or "monte_carlo" for Monte Carlo integration. Default is "monte_carlo".
mc_use_observed	Logical. If TRUE, Monte Carlo integration uses the observed draws. This can be a reasonable approximation for large x. If FALSE, draws are sampled from the KDE mixture; only implemented for kernel = "gaussian".
mc_draws	Optional integer number of draws for Monte Carlo sampling when mc_use_observed = FALSE. Defaults to max(length(x), 2000).

`mc_quantile_type`
: Quantile type passed to `stats::quantile` for the Monte Carlo cutoff. Type 1 matches the order-statistic construction in Hyndman (1996), while type 7 (the default) interpolates between adjacent order statistics for a smoother, lower-variance cutoff that is asymptotically equivalent. Default is 7.

`bw`
: Bandwidth for `density`. Default is "nrd0".

`adjust`
: Bandwidth adjustment factor for `density`. Default is 1.

`kernel`
: Kernel for `density`. Default is "gaussian".

Details

The Monte Carlo integration option follows the approach described by Hyndman (1996) for computing HDRs from an estimated density.

Value

A list with class `"koma_hdr"` containing:

intervals Named list of matrices with columns `lower` and `upper`, one matrix per `probs` level.

mode Location of the KDE mode.

density The density object returned by `stats::density`.

cutoff Named numeric vector of density cutoffs for each level.

mass Named numeric vector of achieved mass for each level. For `integration = "grid"`, this is the area under the KDE above the cutoff (grid-based). For `integration = "monte_carlo"`, this is the Monte Carlo coverage, i.e. the fraction of sampled points with density above the cutoff.

probs Numeric vector of target masses in (0, 1].

References

Hyndman, R. J. (1996). Computing and graphing highest density regions. The American Statistician, 50(2), 120–126. doi:10.2307/2684423

Examples

```
x <- c(rnorm(500, -2, 0.5), rnorm(500, 2, 0.5))
hdr(x, probs = c(0.5, 0.9))

data("simulated_sem")
set.seed(11)

fit <- estimate(
  ts_data = simulated_sem$ts_data,
  sys_eq = simulated_sem$sys_eq,
  dates = simulated_sem$dates,
  options = list(gibbs = list(ndraws = 10))
)
hdr_fit <- hdr(fit, probs = c(0.5, 0.9))
names(hdr_fit$intervals)
```

`hdr.koma_estimate` *Highest Density Regions for koma_estimate Objects*

Description

Computes highest density regions (HDRs) for coefficient draws from a `koma_estimate` object.

Usage

```
## S3 method for class 'koma_estimate'
hdr(
  x,
  variables = NULL,
  probs = c(0.5, 0.99),
  n_grid = 4096,
  integration = c("monte_carlo", "grid"),
  mc_use_observed = FALSE,
  mc_draws = NULL,
  mc_quantile_type = 7,
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  include_sigma = FALSE,
  ...
)
```

Arguments

`x`	A `koma_estimate` object.
`variables`	Optional character vector of endogenous variables to include. Defaults to all variables in `object$estimates`.
`probs`	Numeric vector of target mass levels. Values in $(0, 1]$ or $[0, 100]$ are accepted. Default is `c(0.5, 0.99)`.
`n_grid`	Number of grid points for the KDE. Default is 4096.
`integration`	Integration approach for the HDR cutoff. Use `"grid"` for grid-based integration or `"monte_carlo"` for Monte Carlo integration. Default is `"monte_carlo"`.
`mc_use_observed`	Logical. If TRUE, Monte Carlo integration uses the observed draws. This can be a reasonable approximation for large `x`. If FALSE, draws are sampled from the KDE mixture; only implemented for `kernel = "gaussian"`.
`mc_draws`	Optional integer number of draws for Monte Carlo sampling when `mc_use_observed = FALSE`. Defaults to `max(length(x), 2000)`.
`mc_quantile_type`	Quantile type passed to `stats::quantile` for the Monte Carlo cutoff. Type 1 matches the order-statistic construction in Hyndman (1996), while type 7 (the

	default) interpolates between adjacent order statistics for a smoother, lower-variance cutoff that is asymptotically equivalent. Default is 7.
bw	Bandwidth for density. Default is "nrd0".
adjust	Bandwidth adjustment factor for density. Default is 1.
kernel	Kernel for density. Default is "gaussian".
include_sigma	Logical. If TRUE, compute HDRs for the error variance parameter (omega). Default is FALSE.
...	Additional arguments forwarded to density.

Value

A list with class c("koma_estimate_hdr", "koma_hdr") containing HDR intervals per coefficient. The object also includes an outliers list with draws outside the largest HDR level.

---

hdr.koma_forecast *Highest Density Regions for koma_forecast Objects*

Description

Computes highest density regions (HDRs) for predictive draws in a koma_forecast object.

Usage

```
## S3 method for class 'koma_forecast'
hdr(
  x,
  variables = NULL,
  probs = c(0.5, 0.99),
  n_grid = 4096,
  integration = c("monte_carlo", "grid"),
  mc_use_observed = FALSE,
  mc_draws = NULL,
  mc_quantile_type = 7,
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  ...
)
```

Arguments

x	A koma_forecast object.
variables	Optional character vector of endogenous variables to include. Defaults to all variables in x$forecasts.
probs	Numeric vector of target mass levels. Values in $(0, 1]$ or $[0, 100]$ are accepted. Default is c(0.5, 0.99).

n_grid	Number of grid points for the KDE. Default is 4096.
integration	Integration approach for the HDR cutoff. Use "grid" for grid-based integration or "monte_carlo" for Monte Carlo integration. Default is "monte_carlo".
mc_use_observed	Logical. If TRUE, Monte Carlo integration uses the observed draws. This can be a reasonable approximation for large x. If FALSE, draws are sampled from the KDE mixture; only implemented for kernel = "gaussian".
mc_draws	Optional integer number of draws for Monte Carlo sampling when mc_use_observed = FALSE. Defaults to max(length(x), 2000).
mc_quantile_type	Quantile type passed to stats::quantile for the Monte Carlo cutoff. Type 1 matches the order-statistic construction in Hyndman (1996), while type 7 (the default) interpolates between adjacent order statistics for a smoother, lower-variance cutoff that is asymptotically equivalent. Default is 7.
bw	Bandwidth for density. Default is "nrd0".
adjust	Bandwidth adjustment factor for density. Default is 1.
kernel	Kernel for density. Default is "gaussian".
...	Additional arguments forwarded to density. Do not pass n, bw, adjust, or kernel here.

Value

A list with class c("koma_forecast_hdr", "koma_hdr") containing HDR intervals per variable and horizon. The object also includes per-horizon modes, cutoff levels, and achieved masses.

---

init_koma_theme *Initiate Default Theme*

Description

Initiate default theme used to plot forecasts.

Usage

```
init_koma_theme(
  index = list(start = NULL, end = NULL),
  title = list(font = NULL, text = NULL),
  font = NULL,
  trace_name = NULL,
  xaxis = list(tickfont = NULL, range = list(start = NULL, end = NULL)),
  yaxis = list(y = list(title = list(text = "QoQ, in %"), tick_center = 0), y2 =
   list(title = list(text = "Level"), tick_center = 100), tickfont = NULL, number_ticks
    = 5),
  legend = list(font = NULL),
  color = list(marker = list(in_sample = "rgba(111,111,111,0.6)", forecast =
   "rgba(33,92,175,1)"), bar_textfont = list(in_sample = "black", forecast = "white"))
)
```

Arguments

`index` Optional list with two elements: `start` and `end`. These dates are used to anchor the time series data. Dates should be in the format: c(YEAR, QUARTER).

`title` A list to specify title options. Default is NULL.

- `text`: Sets the plot's title. If not provided, then the variable name will be displayed.
- `font`: Sets the plot title font.

`font` Sets the global font. Note that fonts used in traces and other layout components inherit from the global font.

`trace_name` Optional list to specify the naming of traces. If not provided, the names will be generated based on the start date of the forecasts. For instance, if the forecast starts at 2023-Q1, the default name for `in_sample_growth` would be "2023-Q1 Data (Growth Rate)", and similarly for other traces. The list can have four elements:

- `in_sample_growth`: Naming for in-sample growth data traces.
- `in_sample_level`: Naming for in-sample level data traces.
- `forecast_growth`: Naming for forecast growth data traces.
- `forecast_level`: Naming for forecast level data traces.

`xaxis` A list with custom labels for the x-axis.

- `range`: Sets the range of this axis. List with `start` and `end`.
- `tickfont`: Sets this axis' tick font, including tickfont for annual growth rates.

`yaxis` A list with custom labels for the y-axes `y` (left) and `y2` (right).

- `y`: A list with custom labels for the y-axis.
  - `title`: Label for the left y-axis, defaults to "QoQ, in %" (quarter on quarter).
  - `tick_center`: Value to center the ticks around, defaults to 0.
- `y2`: A list with custom labels for the y-axis.
  - `title`: Label for the left y-axis, defaults to "Level".
  - `tick_center`: Value to center the ticks around, defaults to 100.
- `tickfont`: Sets this axis' tick font.
- `number_ticks`: Integer for the desired number of ticks in the y-axis.

`legend` A list to specify legend options.

- `font`: Sets the legend font.

`color` A list containing color options for different elements of the plot. When working with RGBA colors, the alpha value (the 'A' in RGBA) determines the transparency of the color, ranging from 0 (fully transparent) to 1 (fully opaque).

- `marker`: A list specifying the colors for different traces.
  - `in_sample`: Color for in-sample data traces.
  - `forecast`: Color for forecast traces.
- `bar_textfont`: A list specifying the text color for different bars.
  - `in_sample`: Text color for in-sample data bars.
  - `forecast`: Text color for forecast data bars.

Value

A named list containing the theme settings.

Examples

```
theme <- init_koma_theme()
names(theme)
```

`is_system_of_equations` *Check if Object is a System of Equations*

Description

This function checks if the given object inherits from the class `koma_seq`, indicating that it represents a system of equations.

Usage

```
is_system_of_equations(x)
```

Arguments

`x`	An object to be checked.

Value

Logical. Returns `TRUE` if the object inherits from the class `koma_seq`, and `FALSE` otherwise.

Examples

```
sys <- system_of_equations("y ~ x", exogenous_variables = "x")
is_system_of_equations(sys)
```

`klein` *Klein macroeconomic time series (1970 Q1 onward)*

Description

A list of U.S. macro series from FRED (14 quarterly series).

Usage

```
klein
```

Format

A `list` of 14 `ts` objects:

gdp Real GDP (bil. 2017 USD, SAAR; FRED: GDPC1)

consumption Real PCE (bil. 2017 USD, SAAR; FRED: PCECC96)

investment Real private investment (bil. 2017 USD, SAAR; FRED: GPDIC1)

government Real government spending (bil. 2017 USD, SAAR; FRED: GCEC1)

net_exports Real net exports (bil. 2017 USD, SAAR; FRED: NETEXC)

n_gdp Nominal GDP (bil. USD, SAAR; FRED: GDP)

n_consumption Nominal PCE (bil. USD, SAAR; FRED: PCEC)

n_investment Nominal private investment (bil. USD, SAAR; FRED: GPDI)

n_government Nominal government spending (bil. USD, SAAR; FRED: GCE)

n_profits Nominal profits after tax (bil. USD, SAAR; FRED: CP)

n_wages Nominal private wages & salaries (bil. USD; FRED: A132RC1Q027SBEA)

n_government_wages Nominal government wages & salaries (bil. USD; FRED: B202RC1Q027SBEA)

n_taxes Nominal fed tax receipts (bil. USD; FRED: W007RC1Q027SBEA)

gdp_deflator GDP implicit price deflator (2017=100, SAAR; FRED: GDPDEF)

Source

Federal Reserve Economic Data (FRED), Federal Reserve Bank of St. Louis.

koma *koma: Large Scale Macroeconomic Model*

Description

Tools for estimating and forecasting large-scale macroeconomic models, including extended time series utilities used throughout the package.

Author(s)

Maintainer: Merlin Scherer <scherer@kof.ethz.ch> (ORCID) [copyright holder]

Authors:

- Samad Sarferaz <sarferaz@kof.ethz.ch> (ORCID) [copyright holder]
- Laurent Florin <florin@kof.ethz.ch>

Other contributors:

- Andrew D. Martin (Adapted rwish/riwish code from MCMCpack) [contributor, copyright holder]
- Kevin M. Quinn (Adapted rwish/riwish code from MCMCpack) [contributor, copyright holder]
- Jong Hee Park (Adapted rwish/riwish code from MCMCpack) [contributor, copyright holder]
- Bill Venables (Adapted multivariate_norm code from MASS::mvrnorm) [contributor, copyright holder]
- Brian D. Ripley (Adapted multivariate_norm code from MASS::mvrnorm) [contributor, copyright holder]

See Also

Useful links:

- https://timothymerlin.github.io/koma/
- https://github.com/TimothyMerlin/koma
- Report bugs at https://github.com/TimothyMerlin/koma/issues

koma_ts *Extended Time-Series Object*

Description

The function ets is used to create an extended time-series (ets) object. Any additional attribute passed is saved as such in the ets.

as_ets and is_ets coerce an object to a time- series and test whether an object is a time series of class koma_ts.

Usage

```
ets(
  data = NA,
  start = NULL,
  end = NULL,
  frequency = NULL,
  deltat = NULL,
  ts.eps = getOption("ts.eps"),
  ...
)

as_ets(x = stats::ts(), ...)

is_ets(x)
```

Arguments

data	a vector or matrix of the observed time-series values. A data frame will be coerced to a numeric matrix via data.matrix. (See also 'Details'.)
start	the time of the first observation. Either a single number or a vector of two numbers (the second of which is an integer), which specify a natural time unit and a (1-based) number of samples into the time unit. See the examples for the use of the second form.
end	the time of the last observation, specified in the same way as start.
frequency	the number of observations per unit of time.
deltat	the fraction of the sampling period between successive observations; e.g., 1/12 for monthly data. Only one of frequency or deltat should be provided.
ts.eps	time series comparison tolerance. Frequencies are considered equal if their absolute difference is less than ts.eps. It is also used to check consistency of end - start, frequency, and the length of the time-series.
...	Additional attributes.
x	An object to be coerced / checked.

Details

Mixed arithmetic between `koma_ts` and plain `ts` objects currently follows base R's group generic dispatch and may emit an "Incompatible methods" warning even when the resulting values are valid. Coercing both operands to `koma_ts` avoids that warning.

Value

A koma_ts object.

A koma_ts object.

TRUE if the object is of class `koma_ts`, otherwise FALSE.

See Also

`ts`

Examples

```
x <- ets(ts(1:8, start = c(2020, 1), frequency = 4),
  series_type = "level",
  method = "diff_log"
)
x

x <- ts(1:8, start = c(2020, 1), frequency = 4)
as_ets(x, series_type = "level", method = "diff_log")
x <- as_ets(ts(1:8, start = c(2020, 1), frequency = 4))
is_ets(x)
```

`level` *Compute the level for a time series*

Description

Required attributes of the input time series are:

- `series_type` - "rate" or "level"
- `method` - "percentage", "diff_log", "none", or an expression

Usage

```
level(x, ...)

## S3 method for class 'ts'
level(x, ...)

## S3 method for class 'mts'
level(x, ...)
```

```
## S3 method for class 'list'
level(x, ...)
```

Arguments

x	A time series object. Supported classes are ts and ets (a subclass of ts).
...	arguments passed to methods (unused for the default method).

Value

An ets object.

Examples

```
x <- ets(c(0.3, 0.1, 0.2, -0.1), series_type = "rate", method = "percentage")
level(x)
```

---

model_evaluation *Calculate Out-of-Sample RMSE for a Specified Horizon*

Description

Computes the Root Mean Square Error (RMSE) of a model for a given prediction horizon, incrementing the in-sample data by a quarter for each calculation until the specified horizon equals the end date in the forecast period.

Usage

```
model_evaluation(
  sys_eq,
  variables,
  horizon,
  ts_data,
  dates,
  ...,
  evaluate_on_levels = TRUE,
  options = list(gibbs = list(), summary = "mean", approximate = FALSE),
  restrictions = NULL
)
```

Arguments

sys_eq	A koma_seq object (system_of_equations) containing details about the system of equations used in the model.
variables	A character vector of name(s) of the stochastic endogenous variable for which the forecast error(s) should be calculated. If NULL it is calculated for all variables.
horizon	The forecast horizon in quarters up to which the RMSE should be calculated.
ts_data	time series data set, must include data until end date of forecasting period.
dates	Key-value list for date ranges in various model operations.
...	Additional parameters.
evaluate_on_levels	Boolean, if TRUE RMSE is calculated on levels if FALSE on growth rates.
options	Optional settings for model evaluation. Use list(gibbs = list(), summary = "mean", approximate = FALSE). Elements: • gibbs: Gibbs sampler settings (see Gibbs Sampler Specifications). • summary: "mean" or "median" point forecast used for RMSE. • approximate: Logical; if TRUE, use the fast approximate point forecast (mean/median of coefficient draws).
restrictions	List of model constraints. Default is empty.

Details

The function initiates the RMSE calculation from dates$forecast$start and continues until dates$forecast$start + horizon equals dates$forecast$end. In each iteration, a quarter is added to both the in-sample data and to dates$forecast$start.

Value

DataFrame containing the RMSE of the selected Variables up to the desired horizon.

Examples

```
data("simulated_sem")

dates <- list(
  estimation = list(start = c(1977, 1), end = c(2018, 4)),
  forecast = list(start = c(2023, 2), end = c(2023, 3))
)

rmse <- model_evaluation(
  sys_eq = simulated_sem$sys_eq,
  variables = c("consumption", "investment"),
  horizon = 1,
  ts_data = simulated_sem$ts_data,
  dates = dates,
  evaluate_on_levels = TRUE,
  options = list(gibbs = list(ndraws = 10), summary = "mean")
```

```
)
head(rmse)
```

`model_identification` *Identify Model Parameters from Character Matrices*

Description

This function calculates the gamma and beta matrices based on the given character gamma and beta matrices, along with identity weights. It checks the identification of all stochastic equations and verifies the order and rank conditions for model identification.

Usage

```
model_identification(
  character_gamma_matrix,
  character_beta_matrix,
  identity_weights,
  call = rlang::caller_env()
)
```

Arguments

`character_gamma_matrix`	A character matrix representing the gamma structure of the model.
`character_beta_matrix`	A character matrix representing the beta structure of the model.
`identity_weights`	A list of identity weights to help construct constant vectors.
`call`	The environment from which the error is called. Defaults to `rlang::caller_env()`, and is used to provide context in case of an error.

Value

NULL. This function is used for side effects, stopping execution if model identification conditions are not met.

Examples

```
equations <- "consumption ~ gdp + consumption.L(1) + consumption.L(2),
investment ~ gdp + investment.L(1) + real_interest_rate,
current_account ~ current_account.L(1) + world_gdp,
manufacturing ~ manufacturing.L(1) + world_gdp,
service ~ service.L(1) + population + gdp,
gdp == 0.5*manufacturing + 0.5*service"

exogenous_variables <- c("real_interest_rate", "world_gdp", "population")
```

```
sys_eq <- system_of_equations(equations, exogenous_variables)

model_identification(
  sys_eq$character_gamma_matrix,
  sys_eq$character_beta_matrix,
  sys_eq$identities
)
```

`plot.koma_estimate_hdr` *Plot HDRs for koma_estimate Objects*

Description

Plot HDR intervals for coefficients from a `koma_estimate_hdr` object. Outliers beyond the outer HDR are shown if available in the object. The plot requires HDR levels matching `box_levels` to be present in the object.

Usage

```
## S3 method for class 'koma_estimate_hdr'
plot(x, y = NULL, ...)
```

Arguments

`x`	A `koma_estimate_hdr` object.
`y`	Ignored. Included for compatibility with the generic function.
`...`	Additional arguments controlling the plot. See Details.

Details

Additional arguments supported in `...`:

variables Optional character vector of endogenous variables to plot.

params Optional character vector of parameter groups to plot (e.g., "beta", "gamma", "sigma").

box_levels Optional numeric probabilities in (0, 1] (or $[0, 100]$) defining the inner and outer HDRs. By default, uses the available HDR levels closest to 50% and the maximum probability.

interactive Logical. If TRUE and plotly is available, return an interactive plot via `plotly::ggplotly`. Default is FALSE.

Value

A ggplot object, or a plotly object when `interactive = TRUE` and plotly is available.

`plot.koma_forecast` *Plot koma Forecasts*

Description

Plot koma forecasts

Usage

```
## S3 method for class 'koma_forecast'
plot(x, y = NULL, ...)
```

Arguments

`x`	A `koma_forecast` object (forecast).
`y`	Ignored. Included for compatibility with the generic function.
`...`	Additional parameters:

variables A vector of variable names to plot.

fig Optional. A Plotly figure object. Default is NULL.

theme Optional. A theme for the plot. Default is NULL.

fan Optional. Logical. If TRUE, add a fan chart from quantiles.

fan_quantiles Optional. Numeric probabilities in (0, 1] (or percentages in `[0, 100]`) to define fan-chart bands. Default uses the available quantiles.

central_tendency Optional. A string specifying the type of forecast to print. Can be "mean", "median", or a quantile name like "q_5", "q_50", "q_95". Default is "mean" if available, otherwise "median", or a specified quantile.

Value

A Plotly figure object displaying the data.

Examples

```
if (requireNamespace("plotly", quietly = TRUE)) {
  data("simulated_sem")

  dates <- list(
    current = c(2024, 4),
    estimation = simulated_sem$dates$estimation,
    forecast = list(start = c(2025, 1), end = c(2025, 4))
  )

  ts_data <- simulated_sem$ts_data
  ts_data[simulated_sem$sys_eq$endogenous_variables] <- lapply(
    simulated_sem$sys_eq$endogenous_variables,
    function(x) {
      stats::window(ts_data[[x]], end = dates$current)
```

```
    }
  )

  set.seed(11)
  fit <- estimate(
    ts_data = ts_data,
    sys_eq = simulated_sem$sys_eq,
    dates = dates,
    options = list(gibbs = list(ndraws = 10))
  )
  fc <- forecast(fit, dates = dates)
  plot(fc, variables = "consumption")
}
```

`print.koma_estimate` *Print method for koma_estimate objects*

Description

Provides a concise, console-friendly overview of the estimated system.

Usage

```
## S3 method for class 'koma_estimate'
print(
  x,
  ...,
  variables = NULL,
  central_tendency = "mean",
  ci_low = 5,
  ci_up = 95,
  digits = 2
)
```

Arguments

`x`	A `koma_estimate` object.
`...`	Additional arguments forwarded to formatting internals.
`variables`	Optional character vector of endogenous variables to print. Defaults to all variables.
`central_tendency`	Central tendency used when summarizing estimates (e.g., "mean", "median"). Defaults to "mean".
`ci_low`	Lower bound (percent) for credible intervals. Defaults to 5.
`ci_up`	Upper bound (percent) for credible intervals. Defaults to 95.
`digits`	Number of digits to print for numeric values. Defaults to 2.

Value

Invisibly returns `x` after printing.

See Also

`summary.koma_estimate` for detailed posterior summaries.

`print.koma_forecast` *Print Method for koma_forecast Objects*

Description

This function prints the forecasts contained in a `koma_forecast` object as a multivariate time series (mts). Users can specify which variables to print and select either the mean forecast, the median forecast, or specific quantiles of the forecast distribution.

Usage

```
## S3 method for class 'koma_forecast'
print(x, ..., variables = NULL, central_tendency = NULL, digits = 4)
```

Arguments

`x`	A `koma_forecast` object.
`...`	Additional parameters.
`variables`	Optional. A character vector of variable names to print. Default is NULL, which prints all forecasted variables.
`central_tendency`	Optional. A string specifying the type of forecast to print. Can be "mean", "median", or a quantile name like "q_5", "q_50", "q_95". Default is "mean" if available, otherwise "median", or a specified quantile.
`digits`	Optional. Integer number of decimal digits to round the printed output. Default is 4.

Details

This function prints the forecasts contained in a `koma_forecast` object. Users can choose to print either the mean forecast, the median forecast, or specific quantiles of the forecast distribution.

If `variables` is specified, only the forecasts for those variables are printed. If `central_tendency` is not specified, the function defaults to printing the mean forecast if available, otherwise the median forecast, or a specified quantile.

Printing converts the selected forecast list to an `mts` for readability; the underlying `koma_forecast` object remains a list of `koma_ts`.

Value

Invisibly returns `x`, the original `koma_forecast` object (a list of `koma_ts` objects, one per forecasted variable). Called primarily for its side effect of printing the selected forecast as a multivariate time series (`mts`) to the console.

---

print.koma_hdi *Print method for koma_hdi objects*

Description

Delegates to `summary()` for console output.

Usage

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

Arguments

`x`	A `koma_hdi` object.
`...`	Additional arguments forwarded to `summary.koma_hdi()`.

Value

Invisibly returns `x`.

---

print.koma_hdr *Print method for koma_hdr objects*

Description

Delegates to `summary()` for console output.

Usage

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

Arguments

`x`	A `koma_hdr` object.
`...`	Additional arguments forwarded to `summary.koma_hdr()`.

Value

Invisibly returns `x`.

print.koma_seq *Print System of Equations*

Description

This function prints an object of class `koma_seq` to ensure the equations are displayed in a readable format.

Usage

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

Arguments

`x`	An object of class `koma_seq`.
`...`	Additional arguments passed to or from other methods.

Value

No return value, called for side effects. Prints a formatted representation of the system of equations to the console, with a section header followed by each equation on its own line.

rate *Compute the rate of change for a time series*

Description

Required attributes of the input time series are:

- `series_type` - "rate" or "level"
- `method` - "percentage", "diff_log", "none", or an expression

Usage

```
rate(x, ...)

## S3 method for class 'ts'
rate(x, ...)

## S3 method for class 'list'
rate(x, ...)

## S3 method for class 'mts'
rate(x, ...)
```

Arguments

x	A time series object. Supported classes are ts and ets (a subclass of ts).
...	arguments passed to methods (unused for the default method).

Value

An ets object.

Examples

```
x <- ets(1:10, series_type = "level", method = "diff_log")
rate(x)
```

---

rebase *Rebase Time Series Data Relative to a Base Period*

Description

Calculates a rebased time series using a given date range as the base period. The base period average is set to 100, and all other values are scaled accordingly.

Usage

```
rebase(x, start, end, ...)

## S3 method for class 'ts'
rebase(x, start, end, ...)

## S3 method for class 'list'
rebase(x, start, end, ...)

## S3 method for class 'mts'
rebase(x, start, end, ...)
```

Arguments

x	An ets object.
start	the time of the first observation. Either a single number or a vector of two numbers (the second of which is an integer), which specify a natural time unit and a (1-based) number of samples into the time unit. See the examples for the use of the second form.
end	the time of the last observation, specified in the same way as start.
...	arguments passed to methods (unused for the default method).

Value

An ets object with the level computed.

Examples

```
x <- as_ets(ts(c(90, 95, 100, 105), start = c(2020, 1), frequency = 4))
rebase(x, start = c(2020, 2), end = c(2020, 3))
```

---

running_mean *Running Means for koma_estimate Objects*

Description

Compute running means (cumulative averages) for coefficient draws from a `koma_estimate` object. By default, beta, gamma, and sigma draws are included when available.

Usage

```
running_mean(x, ...)
```

Arguments

`x`	A `koma_estimate` object.
`...`	Additional arguments controlling the output. See Details.

Details

Additional arguments supported in `...`:

variables Optional character vector of endogenous variables to include.

params Optional character vector of parameter groups to include (e.g., "beta", "gamma", "sigma"). Defaults to all available.

thin Optional integer thinning interval for the stored draws. Default is 1 (no thinning).

max_draws Optional integer cap on the number of draws returned. When set, the most recent draws are kept.

grace_draws Optional integer number of initial retained draws to flag as a grace window for convergence interpretation. If NULL, defaults to `max(50, ceiling(0.1 * n_retained))` per series.

Note: `sigma` values are based on `omega_tilde_jw` and use only variances (no covariances) from each covariance draw.

Value

A data.frame with columns `draw`, `value`, `variable`, `param`, `coef`, `draw_position`, `in_grace_window`, and `label`.

Examples

```
data("simulated_sem")
set.seed(11)

fit <- estimate(
  ts_data = simulated_sem$ts_data,
  sys_eq = simulated_sem$sys_eq,
  dates = simulated_sem$dates,
  options = list(gibbs = list(ndraws = 10))
)
rm_df <- running_mean(fit, params = "beta", max_draws = 100)
head(rm_df)
```

`running_mean_plot` *Running Mean Plots for koma_estimate Objects*

Description

Visualize running means (cumulative averages) from `running_mean()` with a light grey band over the grace-window region.

Usage

```
running_mean_plot(x, ...)
```

Arguments

`x`	A `koma_estimate` object.
`...`	Additional arguments controlling the plot. See Details in `running_mean()`.

Details

Additional plot arguments in `...`:

scales Facet scale option passed to `ggplot2::facet_wrap`. Default is "free_y".

facet_ncol Optional integer number of columns for facets.

interactive Logical. If TRUE and plotly is available, return an interactive plot via `plotly::ggplotly`. Default is FALSE.

Value

A ggplot object, or a plotly object when `interactive = TRUE` and plotly is available.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  data("simulated_sem")
  set.seed(11)

  fit <- estimate(
    ts_data = simulated_sem$ts_data,
    sys_eq = simulated_sem$sys_eq,
    dates = simulated_sem$dates,
    options = list(gibbs = list(ndraws = 10))
  )
  running_mean_plot(fit, params = "beta", max_draws = 100)
}
```

---

simulated_sem *Simulated SEM example data*

Description

A compact simulated example bundle for documentation and quick-start workflows with `estimate()`, `forecast()`, and diagnostic plots.

Usage

```
simulated_sem
```

Format

A `list` with three elements:

ts_data A named list of `ets` time-series objects.

sys_eq A `koma_seq` object defining the simulated system of equations.

dates A named list of date ranges for estimation examples.

Source

Simulated within the package from the SEM data generator in `data-raw/DATASET.R`.

`small_open_economy` *Quarterly time series of key macro variables for a small open economy (Switzerland)*

Description

Quarterly time series of key macro variables for a small open economy (Switzerland)

Usage

```
small_open_economy
```

Format

A `list` of 12 `ts` objects (quarterly):

consumption Private consumption \(C_t\).

investment Gross fixed capital formation \(I_t\).

exports Exports \(X_t\).

imports Imports \(M_t\).

prices Price level \(P_t\).

interest_rate Short-term domestic interest rate \(R_t\).

gdp Equilibrium output \(GDP_t\).

domestic_demand Domestic demand \(D_t\).

world_gdp World GDP (exogenous) \(WGDP_t\).

interest_rate_germany German short-term rate (exog) $R_{GE,t}$.

exchange_rate Nominal exchange rate (exog) \(ER_t\).

oil_price Oil price (exogenous) \(OP_t\).

Source

https://www.seco.admin.ch/, https://kof.ethz.ch/

summary.koma_estimate *Summary method for koma_estimate objects*

Description

This function provides a summary for koma_estimate objects. It can return either a texreg object or an ASCII table.

Usage

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

Arguments

object	A koma_estimate object.
...	Additional parameters:
	variables Optional. A character vector of variables to summarize. Default is NULL, which means all variables will be summarized.
	central_tendency Optional. A string specifying the measure of central tendency ("mean", "median"). Default is "mean".
	ci_low Optional. Lower bound of the confidence interval. Default is 5.
	ci_up Optional. Upper bound of the confidence interval. Default is 95.
	use_texreg Optional. If TRUE, prints a texreg summary when available. Defaults to TRUE when texreg is installed, otherwise FALSE.
	digits Optional. Number of digits to round numeric values. Default is 2.
	Additional arguments are forwarded to texreg output helpers (for example, arguments accepted by `texreg::screenreg()`) when `use_texreg = TRUE`.

Value

Returns a list of summary statistics for each variable (invisibly) when `use_texreg` is FALSE. When `use_texreg` is TRUE and texreg is installed, returns a texreg extract object that prints via `screenreg()` when printed.

summary.koma_estimate_hdi *Summary for koma_estimate_hdi Objects*

Description

Prints HDI intervals for coefficients in a table per equation.

Usage

```
## S3 method for class 'koma_estimate_hdi'
summary(object, ..., variables = NULL, params = NULL, digits = 3)
```

Arguments

object	A koma_estimate_hdi object.
...	Unused.
variables	Optional character vector to filter variables.
params	Optional character vector to filter parameters (e.g., "beta", "gamma", "sigma").
digits	Number of digits to round numeric values. Default is 3.

Value

Invisibly returns object.

---

summary.koma_estimate_hdr

Summary for koma_estimate_hdr Objects

Description

Prints HDR intervals for coefficients in a table per equation.

Usage

```
## S3 method for class 'koma_estimate_hdr'
summary(object, ..., variables = NULL, params = NULL, digits = 3)
```

Arguments

object	A koma_estimate_hdr object.
...	Unused.
variables	Optional character vector to filter variables.
params	Optional character vector to filter parameters (e.g., "beta", "gamma", "sigma").
digits	Number of digits to round numeric values. Default is 3.

Value

Invisibly returns object.

summary.koma_forecast *Summary for koma_forecast Objects*

Description

Prints a summary table for forecast horizons, including available central tendencies (mean/median) and quantiles.

Usage

```
## S3 method for class 'koma_forecast'
summary(object, ..., variables = NULL, horizon = NULL, digits = 3)
```

Arguments

object	A koma_forecast object.
...	Unused.
variables	Optional character vector to filter variables.
horizon	Optional numeric or character vector selecting forecast horizon.
digits	Number of digits to round numeric values. Default is 3.

Value

Invisibly returns object.

summary.koma_forecast_hdi *Summary for koma_forecast_hdi Objects*

Description

Prints HDI intervals for forecast horizons.

Usage

```
## S3 method for class 'koma_forecast_hdi'
summary(object, ..., variables = NULL, horizon = NULL, digits = 3)
```

Arguments

object	A koma_forecast_hdi object.
...	Unused.
variables	Optional character vector to filter variables.
horizon	Optional numeric or character vector selecting forecast horizon.
digits	Number of digits to round numeric values. Default is 3.

Value

Invisibly returns `object`.

`summary.koma_forecast_hdr` *Summary for koma_forecast_hdr Objects*

Description

Prints HDR intervals for forecast horizons.

Usage

```
## S3 method for class 'koma_forecast_hdr'
summary(object, ..., variables = NULL, horizon = NULL, digits = 3)
```

Arguments

`object`	A `koma_forecast_hdr` object.
`...`	Unused.
`variables`	Optional character vector to filter variables.
`horizon`	Optional numeric or character vector selecting forecast horizon.
`digits`	Number of digits to round numeric values. Default is 3.

Value

Invisibly returns `object`.

`summary.koma_hdi` *Summary for koma_hdi Objects*

Description

Prints HDI intervals for a single numeric sample.

Usage

```
## S3 method for class 'koma_hdi'
summary(object, ..., digits = 3)
```

Arguments

`object`	A `koma_hdi` object.
`...`	Unused.
`digits`	Number of digits to round numeric values. Default is 3.

Value

Invisibly returns `object`.

`summary.koma_hdr` *Summary for koma_hdr Objects*

Description

Prints HDR intervals for a single numeric sample.

Usage

```
## S3 method for class 'koma_hdr'
summary(object, ..., digits = 3)
```

Arguments

`object`	A `koma_hdr` object.
`...`	Unused.
`digits`	Number of digits to round numeric values. Default is 3.

Value

Invisibly returns `object`.

`system_of_equations` *System of Equations Class*

Description

Create and manipulate a system of equations.

Usage

```
system_of_equations(equations = vector(), exogenous_variables = vector(), ...)
```

Arguments

`equations`	A character string or vector containing the system of equations. If a single string, equations should be separated by commas.
`exogenous_variables`	A character vector of exogenous variables.
`...`	Additional arguments for future extensions.

Details

This function constructs an object of class `koma_seq` representing a system of equations, extracting and organizing key components like endogenous variables, gamma matrix, beta matrix, and more. Equations should be separated by commas if provided as a single string.

Value

An object of class `koma_seq` with the following components:

- `equations`: A character vector of the equations.
- `endogenous_variables`: A character vector of endogenous variables.
- `stochastic_equations`: A character vector of stochastic equations.
- `identities`: A character vector of identity equations.
- `character_gamma_matrix`: A gamma matrix in character form.
- `character_beta_matrix`: A beta matrix in character form.
- `predetermined_variables`: A character vector of lagged variables.
- `total_exogenous_variables`: A character vector of combined constant, predetermined, and exogenous variables.
- `priors`: A list of priors per equation.

Equations

- Stochastic equations use `~` (e.g. `y ~ x1 + x2`).
- Identity equations use `==` (e.g. `y == 0.5*x1 + 0.5*x2`).
- Lagged variables are denoted by `X.L(x)` for variable `X` and lag `L(x)` (e.g. `.L(1)`, `.L(2)`).
- Intercept are included by default, you can also explicitly specify the constant by adding `constant` to the equation (e.g. `y ~ constant + x1`). To exclude the intercept, add `+0` or `-1` to the equation (e.g. `y ~ 0 + x1`).

For more details on the equation syntax, see `vignette("equations")`.

See Also

`get_endogenous_variables()`, `get_identities()`, `construct_gamma_matrix()`, `extract_lagged_vars()`, `construct_beta_matrix()`, `get_max_lag()`

Examples

```
equations <-
  "consumption ~ gdp + consumption.L(1) + consumption.L(2),
investment ~ gdp + investment.L(1) + real_interest_rate,
current_account ~ current_account.L(1) + world_gdp,
manufacturing ~ manufacturing.L(1) + world_gdp,
service ~ service.L(1) + population + gdp,
gdp == 0.4*manufacturing + 0.6*service"

exogenous_variables <- c("real_interest_rate", "world_gdp", "population")
```

```
system <- system_of_equations(equations, exogenous_variables)
print(system)
```

`trace_plot` *Trace Plots for koma_estimate Objects*

Description

Visualize MCMC trace plots for coefficient draws from a `koma_estimate` object. By default, beta, gamma, and sigma draws are shown when available.

Usage

```
trace_plot(x, ...)
```

Arguments

`x`	A `koma_estimate` object.
`...`	Additional arguments controlling the plot. See Details.

Details

Additional arguments supported in `...`:

variables Optional character vector of endogenous variables to plot.

params Optional character vector of parameter groups to plot (e.g., "beta", "gamma", "sigma"). Defaults to all available.

thin Optional integer thinning interval for the stored draws. Default is 1 (no thinning).

max_draws Optional integer cap on the number of draws per trace. When set, the most recent draws are kept.

scales Facet scale option passed to `ggplot2::facet_wrap`. Default is "free_y".

facet_ncol Optional integer number of columns for facets.

interactive Logical. If TRUE and plotly is available, return an interactive plot via `plotly::ggplotly`. Default is FALSE.

Note: `sigma` plots use `omega_tilde_jw` and show only variances (no covariances) from each covariance draw.

Value

A ggplot object, or a plotly object when `interactive = TRUE` and plotly is available.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  data("simulated_sem")
  set.seed(11)

  fit <- estimate(
    ts_data = simulated_sem$ts_data,
    sys_eq = simulated_sem$sys_eq,
    dates = simulated_sem$dates,
    options = list(gibbs = list(ndraws = 10))
  )
  trace_plot(fit, params = "beta", max_draws = 100)
}
```

Index