Package 'bridgr'

August 21, 2026

Type Package

Title Bridging Data Frequencies for Timely Economic Forecasts

Version 1.0.0

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

Description Implements bridge and MIDAS-style mixed-frequency models for nowcasting and forecasting macroeconomic variables by linking higher-frequency indicator variables to a lower-frequency target series. The package standardizes input data, infers regular frequencies, forecasts missing indicator observations, and aggregates indicators to the target frequency before fitting a regression with autoregressive target dynamics. Frequency alignment can be customized through user-supplied conversion rules. For more on bridge and MIDAS models, see Baffigi, A., Golinelli, R., & Parigi, G. (2004) <doi:10.1016/S0169-2070(03)00067-0>, Ghysels, Sinko, & Valkanov (2007) <doi:10.1080/07474930600972467>, Andreou, Ghysels, & Kourtellos (2010) <doi:10.1016/j.jeconom.2010.01.004>, Schumacher (2016) <doi:10.1016/j.ijforecast.2015.07.004>, and Burri (2026) <doi:10.1111/obes.70073>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

LazyData true

Imports dplyr, forecast, ggplot2, lifecycle, lubridate, rlang, scales, tsbox, withr

Suggests knitr, rmarkdown, srr, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 4.1.0)

URL https://github.com/marcburri/bridgr, https://marcburri.github.io/bridgr/

BugReports https://github.com/marcburri/bridgr/issues

VignetteBuilder knitr

NeedsCompilation no

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

Repository CRAN

Date/Publication 2026-08-21 13:10:24 UTC

Contents

`aggregation_parameters` *Estimated Parametric Aggregation Parameters*

Description

Extracts the estimated parameters of parametric aggregation schemes (`"expalmon"`, `"beta"`) from a fitted `mf_model()` object. These are the parameters whose implied weights are returned by `weights.mf_model()`.

Usage

```
aggregation_parameters(object, indicator = NULL, ...)

## S3 method for class 'mf_model'
aggregation_parameters(object, indicator = NULL, ...)
```

Arguments

`object`	A fitted `"mf_model"` object returned by `mf_model()`.
`indicator`	Optional indicator selector, either an indicator name or a position. When supplied, the parameters for that single indicator are returned directly; when `NULL` (default), a named list is returned.
`...`	Unused.

Value

When `indicator` is `NULL`, a named list with one element per indicator that used a parametric aggregator, each a numeric parameter vector. When `indicator` is supplied, that single element, or `NULL` when the indicator did not use a parametric aggregator.

See Also

`weights.mf_model()` for the implied aggregation weights.

Examples

```
gdp_growth <- tsbox::ts_pc(gdp)
gdp_growth <- tsbox::ts_na_omit(gdp_growth)
model <- mf_model(
  target = gdp_growth,
  indic = baro,
  indic_aggregators = "expalmon",
  h = 1
)

aggregation_parameters(model)
```

`as.forecast` *Coerce a Mixed-Frequency Forecast to a forecast Object*

Description

Converts a `"mf_model_forecast"` object into a genuine `"forecast"` object from the **forecast** package, so that functions such as `forecast::accuracy()` can be used on `bridgr` output.

Usage

```
as.forecast(object, ...)

## S3 method for class 'mf_model_forecast'
as.forecast(object, ...)
```

Arguments

object	A "mf_model_forecast" object returned by forecast.mf_model().
...	Unused.

Details

Conversion requires a target frequency that stats::ts() can represent exactly, that is, a whole number of target periods per calendar year. Annual, semi-annual, quarterly, bi-monthly and monthly targets qualify; daily, weekly and sub-daily targets do not, and are rejected with an informative error rather than silently approximated.

Value

An object of class "forecast", with mean, lower, upper, x, fitted and residuals stored as stats::ts() objects.

See Also

forecast.mf_model() for the native forecast object, which supports every target frequency.

Examples

```
gdp_growth <- tsbox::ts_pc(gdp)
gdp_growth <- tsbox::ts_na_omit(gdp_growth)
model <- mf_model(
  target = gdp_growth,
  indic = baro,
  indic_predict = "auto.arima",
  indic_aggregators = "mean",
  h = 1
)

converted <- as.forecast(forecast(model))
class(converted)
```

---

forecast.mf_model *Forecast a Mixed-Frequency Model*

Description

Forecast the target variable from a fitted mf_model() object.

Usage

```
## S3 method for class 'mf_model'
forecast(object, xreg = NULL, level = c(80, 95), ...)

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

Arguments

object	A "mf_model" object returned by mf_model().
xreg	Optional future regressors in a tsbox::ts_boxable() format. When omitted, the forecast regressor set stored inside object is used. When supplied, xreg must contain the same non-target regressors used when fitting the bridge equation.
level	Prediction interval levels used when the model was estimated with se = TRUE. When uncertainty is unavailable, forecast() still returns the se, lower, and upper components, filled with NA.
...	Unused.
x	A "mf_model_forecast" object returned by forecast.mf_model().

Details

In recursive bridge forecasts, uncertainty typically increases with horizon because later forecast steps depend on forecasted rather than observed target lags and, when needed, completed indicator paths. Under the package's residual-resampling and full-system bootstrap workflows, those simulated disturbances accumulate across steps, so standard errors and interval widths can widen as the forecast horizon extends. The uncertainty-and-scenarios vignette includes one worked example that trims forecast rows by an acceptable prediction-interval width.

Value

An object of class "mf_model_forecast" containing point forecasts, predictive uncertainty summaries, the observed target history, the target-period regressors used for forecasting, and optional full-system bootstrap metadata.

x, invisibly.

Interoperability with the forecast package

"mf_model_forecast" deliberately does *not* inherit from the forecast package's "forecast" class. Target frequencies supported by mf_model() include daily, weekly and sub-daily series, which stats::ts() cannot represent without silently approximating the calendar, so an object that claimed "forecast" inheritance could not honour it for every model this package fits. Instead, plot() and ggplot2::autoplot() methods are provided directly for "mf_model_forecast", and as.forecast() converts to a genuine "forecast" object whenever the target frequency is regular enough to allow it, for use with functions such as forecast::accuracy().

Examples

```
gdp_growth <- tsbox::ts_pc(gdp)
gdp_growth <- tsbox::ts_na_omit(gdp_growth)
model <- mf_model(
  target = gdp_growth,
  indic = baro,
  indic_predict = "auto.arima",
  indic_aggregators = "mean",
  h = 1
```

```
)

forecast(model)
```

---

gdp *Swiss Economic Indicators*

Description

A collection of datasets containing economic indicators for Switzerland.

Usage

```
gdp

baro

wea

fcurve
```

Details

- `baro`: The KOF barometer, a monthly business cycle indicator.
- `fcurve`: The F-curve, a daily business cycle indicator.
- `gdp`: Quarterly GDP data (real, seasonally adjusted).
- `wea`: The Weekly Economic Activity (WEA) indicator.

Datasets

- `baro`:
  - Source: KOF Swiss Economic Institute
  - Timeframe: January 2004 - December 2022
  - Frequency: Monthly
  - Format: A tibble with monthly observations and 2 variables:
    - `time`: Date, the month and year of the observation.
    - `values`: Numeric, the value of the KOF barometer.
- `fcurve`:
  - Source: Burri and Kaufmann GitHub
  - Timeframe: January 2004 - December 2022
  - Frequency: Daily
  - Format: A tibble with daily observations and 2 variables:
    - `time`: Date, the date of the observation.
    - `values`: Numeric, the value of the F-curve (inverted for compatibility).

- gdp:
  - Source: SECO
  - Timeframe: January 2004 - December 2022
  - Frequency: Quarterly
  - Format: A tibble with quarterly observations and 2 variables:
    - `time`: Date, the quarter and year of the observation.
    - `values`: Numeric, the real GDP (seasonally adjusted).
- wea:
  - Source: SECO WEA Indicator
  - Timeframe: January 2005 - December 2022
  - Frequency: Weekly
  - Format: A tibble with weekly observations and 2 variables:
    - `time`: Date, the week and year of the observation.
    - `values`: Numeric, the value of the WEA.

Examples

```
# Load and plot `baro`
data(baro)
library(tsbox)
suppressMessages(ts_plot(baro))
```

`indicators` *Indicator Names of a Mixed-Frequency Model*

Description

Returns the identifiers of the high-frequency indicators used by a fitted `mf_model()` object, in the order they enter the bridge equation.

Usage

```
indicators(object, ...)

## S3 method for class 'mf_model'
indicators(object, ...)
```

Arguments

`object`	A fitted `"mf_model"` object returned by `mf_model()`.
`...`	Unused.

Value

A character vector of indicator identifiers.

Examples

```
gdp_growth <- tsbox::ts_pc(gdp)
gdp_growth <- tsbox::ts_na_omit(gdp_growth)
model <- mf_model(
  target = gdp_growth,
  indic = baro,
  indic_predict = "auto.arima",
  indic_aggregators = "mean",
  h = 1
)

indicators(model)
```

`mf_model` *Estimate a Mixed-Frequency Model*

Description

Estimate a bridge model that links one lower-frequency target series to one or more higher-frequency indicator series. Indicators are aligned to the target frequency by forecasting any missing higher-frequency observations and aggregating them within each target period.

Usage

```
mf_model(
  target,
  indic,
  missing = "error",
  indic_predict = NULL,
  indic_aggregators = NULL,
  indic_lags = 0,
  target_lags = 0,
  h = 1,
  frequency_conversions = NULL,
  se = FALSE,
  bootstrap = NULL,
  full_system_bootstrap = FALSE,
  stationarity = "none",
  solver_options = NULL
)

bridge(...)
```

Arguments

`target`	A single target series in a `tsbox::ts_boxable()` format, such as a data frame or tibble with `time` and `value`/`values` columns, or a regular time-series object supported by tsbox.

`indic` One or more indicator series in a `tsbox::ts_boxable()` format, such as data frames / tibbles with `time` and `value`/`values` columns, or regular time-series objects supported by tsbox.

`missing` Character string controlling how explicit missing values in submitted `target` or `indic` series are handled. `missing = "error"` keeps the strict default and aborts on explicit missing values. `missing = "drop"` removes explicit missing values with a warning before downstream analysis, which is most useful when they correspond to supported ragged-edge gaps at the ends of series. `missing = "impute"` replaces explicit missing values by series-wise linear interpolation with endpoint carry before model fitting. Missing timestamps are always an error.

`indic_predict` A character vector of indicator forecasting methods. Length must be 1 or equal to the number of indicator series. Setting `indic_predict = "direct"` switches to direct MIDAS-style alignment: the indicators are not forecasted. Each target period that overlaps the observed sample receives the most recent complete high-frequency block at the same period-relative position that the newest observation occupies within its own period – a MIDAS-with-leads alignment that stays calendar-consistent when target periods hold varying numbers of high-frequency observations (e.g. 13-Saturday quarters on a 12-slot weekly ladder). Forecast periods beyond the newest observation receive complete blocks assigned backward from the end of the sample. Direct alignment must be used for all indicators at once. For `indic_predict = "mean"`, missing high-frequency observations are filled with the mean of the latest available `obs_per_target` high-frequency observations, and that same mean is extended across the forecast horizon.

`indic_aggregators` A character vector of aggregation methods or a list of numeric weights. Length must be 1 or equal to the number of indicator series. Numeric weights must sum to one and have the appropriate length for the inferred target-period block size. `"unrestricted"` keeps one separate coefficient per high-frequency observation within the target period. The parametric aggregators use two coefficients each: `"expalmon"` uses `(linear, quadratic)`, and `"beta"` uses `(left_shape, right_shape)` as the normalized beta shape parameters. When `indic_predict = "direct"`, the supplied aggregation is applied directly to the aligned complete high-frequency blocks within each target period.

`indic_lags` A non-negative integer giving the number of target-period lags to add for each aggregated indicator.

`target_lags` A non-negative integer giving the autoregressive order in the target equation.

`h` A positive integer forecast horizon measured in target periods.

`frequency_conversions` A named numeric vector used to customize the regular frequency ladder. Supported names are `spm`, `mph`, `hpd`, `dpw`, `wpm`, `mpq`, and `qpy`.

`se` Logical flag indicating whether coefficient standard errors and prediction intervals should be computed. When `TRUE`, `mf_model()` reports HAC standard errors for the linear target equation, or Delta-HAC standard errors when parametric aggregation weights are estimated jointly.

bootstrap
: A list of uncertainty controls. Currently only `list(N = 100, block_length = NULL)` is supported. `N` is the number of predictive simulation paths used when `se = TRUE`. If `full_system_bootstrap = TRUE`, the same `N` controls the number of full-system target-period block-bootstrap replications used for prediction intervals. `block_length` is only used by the full-system bootstrap. When `block_length` is `NULL`, `mf_model()` uses `ceiling(n^(1/3))` based on the final target-period sample size.

full_system_bootstrap
: Logical flag indicating whether prediction intervals and coefficient standard errors should be based on a full-system target-period block bootstrap instead of residual resampling and HAC / Delta-HAC uncertainty from the fitted target equation. This option is only used when `se = TRUE`. Because it refits the full bridge workflow on every draw, `full_system_bootstrap = TRUE` can be substantially slower than the default residual-resampling intervals.

stationarity
: Character string controlling optional heuristic lower-order stationarity diagnostics. `stationarity = "none"` skips diagnostics. `stationarity = "warn"` checks submitted target and indicator series for a KPSS-style differencing signal and large variance shifts, and warns when those heuristics suggest that upstream transformations such as differences, growth rates, log changes, or demeaning may be appropriate.

solver_options
: A list of optional controls for joint parametric-weight optimization. Supported entries are: `method` for the optimizer (`"L-BFGS-B"`, `"BFGS"`, `"Nelder-Mead"`, or `"nlminb"`), `maxiter` for the iteration budget per optimization run, `n_starts` for the number of multi-start attempts, `seed` for reproducible random restarts, `trace` for optimizer verbosity, `warn` to control whether non-convergence warnings are emitted, `reltol` for the relative convergence tolerance passed through to the selected optimizer backend, and `start_values` for user-supplied initial parameter values. Documented defaults are `method = "L-BFGS-B"`, `maxiter = 1000`, `n_starts = 5`, `trace = 0`, `warn = TRUE`, and `reltol = 1e-8`. `start_values` can be either a numeric vector or a named list. For a numeric vector, values are concatenated in indicator order across the parametric aggregators. Within each indicator, the parameter order is `(linear, quadratic)` for `"expalmon"` and `(left_shape, right_shape)` for `"beta"`. Named-list `start_values` must provide exactly the required number of values for each parametric indicator. Users can override `reltol` or suppress convergence warnings through `solver_options = list(reltol = ..., warn = FALSE)`. These controls are ignored unless at least one indicator uses a parametric aggregator.

...
: Arguments forwarded to `mf_model()`.

Details

Supported indicator forecasting methods are `"mean"`, `"last"`, `"auto.arima"`, `"ets"`, and `"direct"`. Supported aggregation methods are `"mean"`, `"last"`, `"sum"`, `"unrestricted"`, `"expalmon"`, `"beta"`, or a numeric weight vector supplied inside a `list()`. `"unrestricted"` expands each high-frequency observation within a target period into its own bridge regressor, which corresponds to a U-MIDAS style specification when the frequency gap is small. When one or more indicators use a parametric aggregator, the corresponding aggregation weights are estimated jointly against the final bridge-model objective rather than one indicator at a time.

Unrestricted mixed-frequency regressions can become parameter-heavy quickly. When `indic_aggregators = "unrestricted"`, `mf_model()` warns if the final estimation sample contains fewer than 10 observations per predictor in the bridge regression.

The package assumes a regular frequency ladder `second -> minute -> hour -> day -> week -> month -> quarter -> year`. The default number of lower-level observations per higher-level unit is:

- `spm = 60`
- `mph = 60`
- `hpd = 24`
- `dpw = 7`
- `wpm = 4`
- `mpq = 3`
- `qpy = 4`

Users can override any subset of these values with `frequency_conversions`. If a target period contains more high-frequency observations than implied by the current mapping, `mf_model()` keeps the most recent observations and emits a summarized warning. If a target period contains fewer observations than required, the call fails. Month-, quarter-, and year-based input dates are standardized to period starts when needed for frequency recognition.

Value

An object of class `"mf_model"` containing the standardized input series, inferred frequencies, aligned estimation and forecast datasets, the fitted target model, fitted indicator models, and metadata required by `forecast.mf_model()` and `summary.mf_model()`.

Model specification

`bridgr` does not use a formula interface. Mixed-frequency bridge models are specified through separate `target` and `indic` series plus the forecasting, aggregation, and lag controls because the package must standardize, align, extend, and aggregate the time-series inputs before the final target regression formula can be assembled.

Terminology

Throughout `bridgr`, a *target* is the lower-frequency response series to be forecasted. An *indicator* is any higher-frequency predictor series aligned to the target frequency before the final regression is fit. *Indicator forecasting* refers to how end-of-sample indicator values are completed when the target horizon extends beyond the latest observed indicator block. *Aggregation* refers to how high-frequency indicator values within a target period are combined into bridge regressors. *Direct* prediction skips indicator forecasting and aligns the latest complete high-frequency blocks backward from the forecast horizon, while *unrestricted* aggregation keeps one separate coefficient per within-period high-frequency observation.

Input standardization

Submitted series are converted to a common internal table with `id`, `time`, and numeric `values` columns before model fitting. This means that additional input attributes beyond the series identifier, timestamps, and numeric values are not preserved in the fitted object unless they are re-encoded in those standardized columns.

Input assumptions

`bridgr` assumes that submitted target and indicator series are ordered by time within each series, free of duplicate timestamps and explicit missing values, and regular enough for the package to infer a supported target and indicator frequency. It also assumes that indicator series are at least as high-frequency as the target. Violations of these assumptions are rejected during preprocessing and validation rather than being silently repaired.

Stationarity

`bridgr` assumes that users provide target and indicator series on a scale that is appropriate for bridge-style forecasting. In practice this often means working with growth rates, differences, or other transformed series prepared upstream. This expectation primarily concerns the lower-order moments that matter most for bridge-style forecasting, typically the mean and variance of the submitted series. By default the package does not automatically enforce stationarity, but `stationarity = "warn"` enables heuristic pre-fit diagnostics for strong linear trends and variance shifts and points users toward differences, growth rates, log changes, demeaning, or other variance-stabilizing transformations when those heuristics are triggered.

Deprecated `bridge()` wrapper

`bridge()` is retained for compatibility and forwards to `mf_model()` with a deprecation warning.

References

Baffigi, A., Golinelli, R., & Parigi, G. (2004). Bridge models to forecast the euro area GDP. *International Journal of Forecasting*, 20(3), 447-460. doi:10.1016/S01692070(03)000670

Ghysels, E., Sinko, A., & Valkanov, R. (2007). MIDAS regressions: Further results and new directions. *Econometric Reviews*, 26(1), 53-90. doi:10.1080/07474930600972467

Andreou, E., Ghysels, E., & Kourtellos, A. (2010). Regression models with mixed sampling frequencies. *Journal of Econometrics*, 158(2), 246-261. doi:10.1016/j.jeconom.2010.01.004

Schumacher, C. (2016). A comparison of MIDAS and bridge equations. *International Journal of Forecasting*, 32(2), 257-270. doi:10.1016/j.ijforecast.2015.07.004

Burri, M. (2026). Nowcasting Swiss GDP Growth From Public Lead Texts: Simple Methods Are Sufficient. *Oxford Bulletin of Economics and Statistics*, 1-25. doi:10.1111/obes.70073

Examples

```
gdp_growth <- tsbox::ts_pc(gdp)
gdp_growth <- tsbox::ts_na_omit(gdp_growth)
gdp_growth <- dplyr::slice_tail(gdp_growth, n = 12)
baro_small <- dplyr::slice_tail(baro, n = 36)
```

```
mf_model(
  target = gdp_growth,
  indic = baro_small,
  indic_predict = "auto.arima",
  indic_aggregators = "mean",
  indic_lags = 1,
  target_lags = 1,
  h = 1,
  frequency_conversions = c(mpq = 3),
  se = TRUE,
  bootstrap = list(N = 2),
  solver_options = list(seed = 123, n_starts = 1)
)
```

`mf_model-accessors` *Accessor Methods for Mixed-Frequency Models*

Description

Access standard model summaries from a fitted `mf_model()` object.

Usage

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

## S3 method for class 'mf_model'
confint(object, parm = NULL, level = 0.95, ...)

## S3 method for class 'mf_model'
formula(x, ...)

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

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

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

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

## S3 method for class 'mf_model'
model.frame(formula, which = c("estimation", "forecast"), ...)
```

```
## S3 method for class 'mf_model'
variable.names(object, which = c("all", "xreg", "target_lags"), ...)

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

Arguments

object, x, formula	A fitted "mf_model" object returned by mf_model(). The formula spelling is required by the stats::model.frame() generic and carries the same meaning.
...	Unused.
parm, level	Passed to confint(). Confidence intervals are computed from the coefficient covariance matrix returned by stats::vcov(), which may be the HAC or Delta-HAC covariance when se = TRUE. Critical values use a t-distribution with residual degrees of freedom from the fitted target equation; this is conservative relative to asymptotic normal critical values but is common practice in applied econometrics.
which	For model.frame(), which modelling frame to return, "estimation" (default) or "forecast". For variable.names(), which group of regressor names to return, "all" (default), "xreg" or "target_lags".

Details

residuals.mf_model() returns target-equation residuals on the same standardized scale as the fitted target series, so they can be passed directly to downstream residual diagnostics.

model.frame.mf_model() returns the aligned modelling data. Use which = "estimation" for the in-sample bridge-equation data, and which = "forecast" for the future target-period regressor path the forecast is produced from. The latter is the frame to modify and pass back as xreg when constructing scenarios.

variable.names.mf_model() returns the names of the bridge-equation regressors. which = "xreg" returns the non-target-lag regressors, which are exactly the series a custom xreg must supply when forecasting a scenario, and pairs with model.frame(object, which = "forecast").

Value

The requested model summary, usually delegated from the stored target regression fit.

x, invisibly.

Examples

```
gdp_growth <- tsbox::ts_pc(gdp)
gdp_growth <- tsbox::ts_na_omit(gdp_growth)
model <- mf_model(
  target = gdp_growth,
  indic = baro,
  indic_predict = "auto.arima",
```

```
  indic_aggregators = "mean",
  h = 1
)

coef(model)
```

`plot.mf_model` *Plot a Mixed-Frequency Model*

Description

Visualize a fitted `mf_model()` object either as an in-sample fit or as a forecast with prediction intervals.

Usage

```
## S3 method for class 'mf_model'
plot(
  x,
  type = c("fit", "forecast"),
  level = 95,
  history_n = 50,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  ...
)
```

Arguments

`x`	A `"mf_model"` object returned by `mf_model()`.
`type`	Plot type. Use `"forecast"` to plot the observed target history together with the bridge forecast, or `"fit"` to compare the observed target to the in-sample fitted values.
`level`	Forecast interval level passed to `forecast.mf_model()` when `type = "forecast"`.
`history_n`	Number of historical target observations to display. Defaults to the most recent `50`. Set to `NULL` to show the full history.
`xlab, ylab, main`	Optional axis and title labels. When omitted, sensible defaults are chosen from `type`.
`...`	Additional arguments passed to `theme_bridgr()`.

Value

A ggplot2 object.

Examples

```
gdp_growth <- tsbox::ts_pc(gdp)
gdp_growth <- tsbox::ts_na_omit(gdp_growth)
model <- mf_model(
  target = gdp_growth,
  indic = baro,
  indic_predict = "auto.arima",
  indic_aggregators = "mean",
  h = 1
)

plot(model)
```

---

plot.mf_model_forecast

Plot a Mixed-Frequency Forecast

Description

Visualize a `"mf_model_forecast"` object returned by `forecast.mf_model()`, showing the observed target history together with the bridge forecast and, when available, a prediction interval.

Usage

```
## S3 method for class 'mf_model_forecast'
plot(
  x,
  level = NULL,
  history_n = 50,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  ...
)

## S3 method for class 'mf_model_forecast'
autoplot(object, ...)
```

Arguments

`x`	A `"mf_model_forecast"` object returned by `forecast.mf_model()`.
`level`	Prediction interval level to display. Must be one of the levels the forecast was computed with. Defaults to the first available level.
`history_n`	Number of historical target observations to display. Defaults to the most recent `50`. Set to `NULL` to show the full history.
`xlab, ylab, main`	Optional axis and title labels.
`...`	Additional arguments passed to `theme_bridgr()`.
`object`	A `"mf_model_forecast"` object, for the `ggplot2::autoplot()` method.

Details

These methods are provided directly for `"mf_model_forecast"` rather than inherited from the **forecast** package, so that plotting works for every target frequency `mf_model()` supports, including daily and weekly targets that `stats::ts()` cannot represent. See `as.forecast()` to convert to a `"forecast"` object where the frequency allows it.

Value

A ggplot2 object.

Examples

```
gdp_growth <- tsbox::ts_pc(gdp)
gdp_growth <- tsbox::ts_na_omit(gdp_growth)
model <- mf_model(
  target = gdp_growth,
  indic = baro,
  indic_predict = "auto.arima",
  indic_aggregators = "mean",
  h = 1
)

plot(forecast(model))
```

---

print.summary.mf_model

Print a Mixed-Frequency Model Summary

Description

Print a Mixed-Frequency Model Summary

Usage

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

Arguments

`x`	A `"summary.mf_model"` object returned by `summary.mf_model()`.
`...`	Unused.

Value

`x`, invisibly.

Examples

```
gdp_growth <- tsbox::ts_pc(gdp)
gdp_growth <- tsbox::ts_na_omit(gdp_growth)
model <- mf_model(
  target = gdp_growth,
  indic = baro,
  indic_predict = "auto.arima",
  indic_aggregators = "mean",
  h = 1
)

print(summary(model))
```

`summary.mf_model` *Summarize a Mixed-Frequency Model*

Description

Computes the summary quantities for a fitted `mf_model()` object and returns them as a `"summary.mf_model"` object. Following the convention of `summary.lm()`, the summary is a data object in its own right: the report is rendered by `print.summary.mf_model()` rather than by `summary()` itself, so the individual quantities can be extracted programmatically.

Usage

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

Arguments

`object`	A `"mf_model"` object returned by `mf_model()`.
`...`	Unused.

Value

An object of class `"summary.mf_model"`, a list with components:

`target_name`, `target_frequency`, `h`, `nobs` Target series name, inferred target frequency unit, forecast horizon, and number of estimation rows.

`regressor_names` Character vector of bridge-equation regressors.

`coefficients` Numeric matrix with columns `"Estimate"`, `"Std. Error"`, `"t value"` and `"Pr(>|t|)"`. Standard errors come from `vcov.mf_model()`, so they are the HAC, Delta-HAC or bootstrap standard errors when the model was fitted with `se = TRUE`.

`coefficient_method` Method used for the coefficient standard errors, or `NULL` when the model was fitted without uncertainty.

`r.squared`, `adj.r.squared`, `sigma`, `df.residual` Fit measures for the target equation.

`indicators` Data frame of per-indicator frequency, completion method and aggregation scheme, one row per indicator.

`custom_weights` Named list of user-supplied numeric aggregation weights, empty when none were used.

`parametric_weights`, `parametric_parameters` Named lists of estimated parametric aggregation weights and their underlying parameters, empty when no parametric aggregator was used.

`uncertainty`, `bootstrap`, `optimization` Uncertainty settings, bootstrap diagnostics, and joint parametric-optimization diagnostics.

See Also

`coef.mf_model()`, `confint.mf_model()` and `vcov.mf_model()` for extracting individual quantities directly from the fitted model.

Examples

```
gdp_growth <- tsbox::ts_pc(gdp)
gdp_growth <- tsbox::ts_na_omit(gdp_growth)
model <- mf_model(
  target = gdp_growth,
  indic = baro,
  indic_predict = "auto.arima",
  indic_aggregators = "mean",
  h = 1
)

model_summary <- summary(model)
model_summary

# The coefficient matrix is available programmatically, as for `lm()`.
coef(model_summary)
```

---

theme_bridgr *bridgr Plot Theme and Color Scales*

Description

Plot styling helpers for `bridgr` graphics, based on the visual defaults used in the `reviser` package.

Usage

```
theme_bridgr(
  base_size = 12,
  legend_position = "bottom",
  legend_direction = "horizontal",
  ...
)
```

```
colors_bridgr()

scale_color_bridgr(...)

scale_fill_bridgr(...)
```

Arguments

base_size	Base text size for the plot theme.
legend_position	Legend position passed to ggplot2::theme().
legend_direction	Legend direction passed to ggplot2::theme().
...	Additional arguments. In theme_bridgr(), legend.position and legend.direction are accepted for backward compatibility. In the scale helpers, ... is forwarded to the ggplot2 scale constructors.

Value

A ggplot2 theme, color palette, or scale.

Examples

```
ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg, color = factor(cyl))) +
  ggplot2::geom_point() +
  theme_bridgr()

colors_bridgr()[1:3]
```

---

weights.mf_model *Aggregation Weights of a Mixed-Frequency Model*

Description

Extracts the within-period aggregation weights applied to each indicator. For parametric aggregators ("expalmon", "beta") these are the weights implied by the estimated parameters; for user-supplied numeric aggregators they are the supplied weights.

Usage

```
## S3 method for class 'mf_model'
weights(object, indicator = NULL, ...)
```

Arguments

`object`	A fitted `"mf_model"` object returned by `mf_model()`.
`indicator`	Optional indicator selector, either an indicator name or a position. When supplied, the weights for that single indicator are returned directly; when `NULL` (default), a named list covering every indicator is returned.
`...`	Unused.

Details

The deterministic rules imply the fixed weight vectors of $\tilde{x}_t = \sum_m w_m x_{t,m}$: `"mean"` gives $w_m = 1/M$, `"last"` gives $w_M = 1$ and zero elsewhere, and `"sum"` gives $w_m = 1$. These are returned alongside the estimated and user-supplied weights, so the accessor reports the weights actually applied whichever aggregator was chosen. `"unrestricted"` has no single weight vector, because it estimates one coefficient per within-period observation instead of aggregating; it returns `NULL`, as does an indicator aligned with `indic_predict = "direct"`.

Value

When `indicator` is `NULL`, a named list with one element per indicator, each either a numeric weight vector or `NULL` for indicators whose aggregator does not imply a fixed weight vector (`"unrestricted"`). When `indicator` is supplied, that single element.

See Also

`aggregation_parameters()` for the underlying parametric parameters, and `indicators()` for the available indicator names.

Examples

```
gdp_growth <- tsbox::ts_pc(gdp)
gdp_growth <- tsbox::ts_na_omit(gdp_growth)
model <- mf_model(
  target = gdp_growth,
  indic = baro,
  indic_aggregators = "expalmon",
  h = 1
)

weights(model)
weights(model, indicator = 1)
```

Index