Package ‘tsissm’

April 28, 2025

Type Package

Title Linear Innovations State Space Unobserved Components Model

Version 1.0.1

Maintainer Alexios Galanos <`alexios@4dscape.com`>

Description Unobserved components time series model using the linear innovations state space representation (single source of error) with choice of error distributions and option for dynamic variance. Methods for estimation using automatic differentiation, automatic model selection and ensembling, prediction, filtering, simulation and backtesting. Based on the model described in Hyndman et al (2012) <doi:10.1198/jasa.2011.tm09771>.

Depends R (>= 4.1.0), Rcpp (>= 0.12.9), tsmethods (>= 1.0.0)

LinkingTo Rcpp (>= 0.12.9), TMB, RcppEigen

Imports TMB (>= 1.7.20), methods, tsaux, tsdistributions, zoo, xts, copula, flextable, data.table, sandwich, nloptr, RTMB, viridisLite, future, future.apply, progressr

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

License GPL-2

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

LazyData true

URL `https://github.com/tsmodels/tsissm`, `https://www.nopredict.com/packages/tsissm`

Config/testthat/edition 3

NeedsCompilation yes

Author Alexios Galanos [aut, cre] (<`https://orcid.org/0009-0000-9308-0457`>)

Repository CRAN

Date/Publication 2025-04-28 18:30:02 UTC

Contents

AIC.tsissm.estimate 2
as_flextable.summary.tsissm.estimate 3
BIC.tsissm.estimate 4
bread.tsissm.estimate 5
coef.tsissm.estimate 5
estfun.tsissm.estimate 6
estimate.tsissm.spec 6
fitted.tsissm.estimate 8
hresiduals.tsissm.estimate 8
issm_modelspec 9
logLik.tsissm.estimate 11
maunaloa 12
nobs.tsissm.estimate 12
plot.tsissm.estimate 13
predict.tsissm.estimate 13
print.summary.tsissm.estimate 15
print.tsissm.diagnose 16
residuals.tsissm.estimate 16
sigma.tsissm.estimate 17
simulate.tsissm.estimate 17
spy 19
summary.tsissm.estimate 20
tsbacktest.tsissm.autospec 20
tsdecompose.tsissm.estimate 22
tsdiagnose.tsissm.estimate 23
tsensemble.tsissm.selection 24
tsequation.tsissm.estimate 25
tsfilter.tsissm.estimate 26
tsmetrics.tsissm.predict 26
tsmoments.tsissm.estimate 27
tsprofile.tsissm.estimate 28
tsspec.tsissm.estimate 29
us_retail_sales 30
vcov.tsissm.estimate 31

Index **32**

`AIC.tsissm.estimate` *Akaike's An Information Criterion*

Description

Extract the AIC from an estimated model.

Usage

```
## S3 method for class 'tsissm.estimate'
AIC(object, ..., k = 2)

## S3 method for class 'tsissm.selection'
AIC(object, ..., k = 2)
```

Arguments

`object`	an object of class “tsissm.estimate”.
`...`	not currently used.
`k`	the penalty per parameter to be used; the default k = 2 is the classical AIC.

Value

a numeric value.

`as_flextable.summary.tsissm.estimate`
Transform a summary object into flextable

Description

Transforms a “summary.tsissm.estimate” object into a flextable with options on symbolic representation and model equation.

Usage

```
## S3 method for class 'summary.tsissm.estimate'
as_flextable(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  include.symbols = TRUE,
  include.equation = TRUE,
  include.statistics = TRUE,
  table.caption = paste0("ISSM Model: ", x$model),
  ...
)
```

Arguments

`x`	an object of class “summary.tsissm.estimate”.
`digits`	integer, used for number formatting. Optionally, to avoid scientific notation, set ‘options(scipen=999)’.
`signif.stars`	logical. If TRUE, ‘significance stars’ are printed for each coefficient.

`include.symbols`	logical. If TRUE, replaces parameter names with their symbols (if they exist).
`include.equation`	logical. If TRUE, adds a section with the symbolic model equation.
`include.statistics`	logical. If TRUE, adds a section with summary statistics on the model.
`table.caption`	an optional string for the table caption.
`...`	additional arguments passed to flextable method.

Value

A flextable object.

`BIC.tsissm.estimate` *Bayesian Information Criterion*

Description

Extract the BIC from an estimated model.

Usage

```
## S3 method for class 'tsissm.estimate'
BIC(object, ...)

## S3 method for class 'tsissm.selection'
BIC(object, ...)
```

Arguments

`object`	an object of class “tsissm.estimate”.
`...`	not currently used.

Value

A numeric value.

`bread.tsissm.estimate` *Bread Method*

Description

Bread Method

Usage

```
## S3 method for class 'tsissm.estimate'
bread(x, ...)
```

Arguments

`x`	an object of class “tsissm.estimate”.
`...`	not currently used.

Value

The analytic hessian of the model.

Author(s)

Alexios Galanos

`coef.tsissm.estimate` *Extract Model Coefficients*

Description

Extract the estimated coefficients of a model.

Usage

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

Arguments

`object`	an object of class “tsissm.estimate”.
`...`	not currently used.

Value

a numeric named vector.

---

estfun.tsissm.estimate *Score Method*

Description

Score Method

Usage

```
## S3 method for class 'tsissm.estimate'
estfun(x, ...)
```

Arguments

`x`	an object of class "tsissm.estimate".
`...`	not currently used.

Details

The function returns the scores of the negative of the log likelihood at the optimal solution.

Value

The score matrix

Author(s)

Alexios Galanos

---

estimate.tsissm.spec *Model Estimation*

Description

Estimates a model given a specification object using maximum likelihood.

Usage

```
## S3 method for class 'tsissm.spec'
estimate(
  object,
  control = issm_control(algorithm = "SLSQP", trace = 0),
  scores = TRUE,
  debug_mode = FALSE,
  ...
)

## S3 method for class 'tsissm.autospec'
estimate(object, control = NULL, trace = FALSE, ...)
```

Arguments

`object`	an object of class “tsissm.spec” or “tsissm.autospec”.
`control`	solver control parameters passed to the nloptr function.
`scores`	whether to calculate the analytic scores (Jacobian) of the likelihood. This is not available for the “tsissm.autospec” object.
`debug_mode`	for development testing, will include the TMB object.
`...`	not used.
`trace`	whether to show a progress bar for the automatic selection object and also output verbose messages.

Details

The maximum likelihood estimation for this model is described in the vignette.

Value

An object of class “tsissm.estimate” or “tsissm.selection”. In the case of automatic model selection an object of class “tsissm.estimate” will be returned based on AIC (minimum) if “top_n” is 1, else an object of class “tsissm.selection” with a list of length “top_n”. This object can then be used for filtering, prediction and simulation and then ensembled (based on user specified weights).

Note

When calculating the scores, a future promise is created so it is fastest if a future plan is pre-created with at least 2 workers so that the function can run in the background without having to wait for the estimation object to be returned.
If the control argument is NULL, then we hybrid strategy is adopted whereby the SLSQP algorithm is initially used and if it fails (status less than 0) then the Augmented Lagrange with MMA local solver is used which is slower but may be more reliable.
For the automatic selection estimation, this will benefit from the use of multiple processes which can be set up with a `plan`. For progress tracing, use `handlers`. The function will check for number of parallel workers initialized (using ‘nbrOfWorkers’), and if it finds only 1 then will revert to non-parallel execution of the code.

`fitted.tsissm.estimate`
Model Fitted Values

Description

Extract the fitted values from an estimated model.

Usage

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

Arguments

`object`	an object of class “tsissm.estimate”.
`...`	not currently used.

Value

an xts object of the fitted values.

`hresiduals.tsissm.estimate`
Multi-Step Ahead In-Sample Residuals

Description

Extract the multi-step ahead in-sample residual values from an estimated model.

Usage

```
## S3 method for class 'tsissm.estimate'
hresiduals(
  object,
  h = 12,
  transformed = TRUE,
  index_start = 0,
  simplify = TRUE,
  ...
)

hresiduals(object, ...)
```

Arguments

`object`	an object of class “tsissm.estimate”.
`h`	the forecast horizon
`transformed`	residuals based values in transformed space (Box Cox).
`index_start`	the time point from which to initiate the in-sample rolling forecasts. This is zero based to enable the first forecast to be t=1.
`simplify`	whether to return a matrix type data.table of error against date and horizon, else the long for data.table with the forecasts, actuals and errors.
`...`	not currently used.

Details

For each time point t (t>=index_start), in the data sample, an h-steps ahead forecast (predicting the observation at time t + h) is made, using the full sample estimated parameters and observed data up to t. These h-step-ahead fitted residuals, in either transformed or untransformed space, can sometimes be used for diagnosing the multi-step ahead in-sample performance of the model. This is not a substitute for a proper rolling out of sample forecast, but a quick method which may still be useful.

Value

A data.table in either long or wide format.

`issm_modelspec` *Model Specification*

Description

Specifies an ISSM model prior to estimation with option for automatic model selection.

Usage

```
issm_modelspec(
  y,
  auto = FALSE,
  slope = TRUE,
  slope_damped = FALSE,
  seasonal = FALSE,
  seasonal_frequency = 1,
  seasonal_harmonics = NULL,
  ar = 0,
  ma = 0,
  xreg = NULL,
  variance = "constant",
  garch_order = c(1, 1),
  lambda = NULL,
  lower = 0,
  upper = 1,
  distribution = "norm",
  sampling = NULL,
  init_garch = "unconditional",
  sample_n = 10,
  top_n = 1,
  ...
)
```

Arguments

`y`	an xts vector.
`auto`	whether to use automatic model selection.
`slope`	(Logical) slope component. If “auto” is TRUE, then this can be a vector of size 2 with TRUE and FALSE.
`slope_damped`	(Logical) slope dampening component. If “auto” is TRUE, then this can be a vector of size 2 with TRUE and FALSE.
`seasonal`	(Logical) seasonal component(s).
`seasonal_frequency`	vector of numeric seasonal frequencies (can be fractional).
`seasonal_harmonics`	the number of harmonics per seasonal frequency for the trigonometric seasonality. If “auto” is TRUE, this can be a list with slots for each seasonal frequency listing the sequence of harmonics to test for each.
`ar`	AR order.
`ma`	MA order.
`xreg`	an xts matrix of external regressors.
`variance`	either “constant” or “dynamic”. In the latter case a GARCH model will be used. For the automatic selection case, both can be provided as a vector.
`garch_order`	the order of the GARCH model (no automatic selection of GARCH order allowed).
`lambda`	the Box Cox lambda. If not NULL (no transformation), then either a numeric value or NA denoting automatic estimation.
`lower`	lower bound for the transformation (defaults to 0).
`upper`	upper bound for the transformation (defaults to 1.5).
`distribution`	a choice of the Normal (“norm”), Student (“std”) or Johnson’s SU (“jsu”) distributions. There is no choice for selecting multiple choices for automatic selection.
`sampling`	(optional) sampling frequency of the dataset. If NULL, will try to identify from the timestamps of y. This is useful for plotting and extending the timestamps in the prediction horizon.

`init_garch`	GARCH variance initialization method with options to use the “unconditional” variance or a “sample” of length “sample_n”.
`sample_n`	the sample length to use if choosing to initialize the GARCH variance using the “sample” method.
`top_n`	how many models to return from the top when using automatic model selection. If this is equal to 1 then the best selected model based on lowest AIC will be returned else a list of the top_n estimated models (based on AIC).
`...`	not used.

Details

The specification object holds the information and data which is then passed to the maximum likelihood estimation routines. Depending on whether automatic selection is chosen, it will dispatch to the appropriate estimation routine.

The specification performs some sanity checks on the arguments provided and sets up the required state space matrices and parameters which are used in the estimation stage.

Value

An object of class “tsissm.spec” or “tsissm.autospec”.

References

De Livera, Alysha M and Hyndman, Rob J and Snyder, Ralph D, 2011, Forecasting time series with complex seasonal patterns using exponential smoothing, *Journal of the American Statistical Association*, **106(496)**, 1513–1527.

`logLik.tsissm.estimate` *Model Log-Likelihood*

Description

Extract the log-likelihood from an estimated model.

Usage

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

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

Arguments

`object`	an object of class “tsissm.estimate”.
`...`	not currently used.

Value

Returns an object of class logLik. This is a number with at least one attribute, "df" (degrees of freedom), giving the number of (estimated) parameters in the model.

`maunaloa` *Mauna Loa CO2 Daily Measurements*

Description

The dataset represents the daily average of the number of carbon dioxide molecules in a given number of molecules of air, after removal of water vapor (molefrac) from the Mauna Loa observatory, spanning the period 1974-05-19 to 2025-02-25.

Usage

```
maunaloa
```

Format

## ‘maunaloa‘ A data.frame containing 15571 observations

date The YYYY-MM-DD date

co2 The CO2 measurements

Source

https://gml.noaa.gov/ccgg/trends/data.html

`nobs.tsissm.estimate` *Extract the Number of Observations*

Description

Extract the number of observations from an estimated model. This is principally intended to be used in computing BIC and used in other tidy methods

Usage

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

Arguments

`object`	an object of class “tsissm.estimate”.
`...`	not currently used.

Value

A numeric value.

`plot.tsissm.estimate` *Object Plots*

Description

Plots for objects generated from the tsissm functions.

Usage

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

## S3 method for class 'tsissm.simulate'
plot(x, y = NULL, ...)

## S3 method for class 'tsissm.profile'
plot(x, y = NULL, type = c("coef", "mape", "mase", "crps"), ...)
```

Arguments

`x`	an object of class “tsissm.estimate”, “tsissm.simulate” or “tsissm.profile”.
`y`	not used.
`...`	additional arguments passed to the underlying plot function.
`type`	type of profile plot for objects of class “tsissm.profile”.

Value

different plots depending on the input class.

`predict.tsissm.estimate`
Model Prediction

Description

Prediction function for class “tsissm.estimate” or “tsissm.selection”.

Usage

```
## S3 method for class 'tsissm.estimate'
predict(
  object,
  h = 12,
  seed = NULL,
  newxreg = NULL,
  nsim = 1000,
  forc_dates = NULL,
  innov = NULL,
  innov_type = "q",
  init_states = NULL,
  ...
)

## S3 method for class 'tsissm.selection'
predict(
  object,
  h = 12,
  newxreg = NULL,
  nsim = 1000,
  forc_dates = NULL,
  init_states = NULL,
  ...
)
```

Arguments

`object`	an object of class “tsissm.estimate” or “tsissm.selection”.
`h`	the forecast horizon.
`seed`	an object specifying if and how the random number generator should be initialized (‘seeded’). Either NULL or an integer that will be used in a call to set.seed before simulating the response vectors.
`newxreg`	a matrix of external regressors in the forecast horizon.
`nsim`	the number of simulations to use for generating the simulated predictive distribution.
`forc_dates`	an optional vector of forecast dates equal to h. If NULL will use the implied periodicity of the data to generate a regular sequence of dates after the last available date in the data.
`innov`	an optional vector of innovations (see innov_type). The length of this vector should be equal to nsim x horizon.
`innov_type`	if ‘innov’ is not NULL, then this denotes the type of values passed, with “q” denoting quantile probabilities (default and backwards compatible) and “z” for standardized errors.
`init_states`	an optional vector of states to initialize the forecast and override the initial state vector. If NULL, will use the last available state from the estimated model.
`...`	not currently used.

Details

Like all models in the tsmodels framework, prediction is done by simulating h-steps ahead in order to build a predictive distribution.

Value

An object of class “tsissm.predict” which also inherits “tsmodel.predict”, with slots for the simulated prediction distribution, the original series (as a zoo object), the original specification object and the mean forecast. The predictive distribution is back transformed if lambda was not set to NULL in the specification. If the input class is “tsissm.selection” then the returned class with be “tsissm.selection_predict” which hold the list of predicted objects. For this dispatch method, custom innovations are not allowed since correlated innovations are passed to each model predicted (using a Normal Copula) in order to enable ensembling of the simulated predictive distributions.

`print.summary.tsissm.estimate` *Model Estimation Summary Print method*

Description

Print method for class “summary.tsissm.estimate”

Usage

```
## S3 method for class 'summary.tsissm.estimate'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

`x`	an object of class “summary.tsissm.estimate”.
`digits`	integer, used for number formatting. Optionally, to avoid scientific notation, set ‘options(scipen=999)’.
`signif.stars`	logical. If TRUE, ‘significance stars’ are printed for each coefficient.
`...`	not currently used.

Value

Invisibly returns the original summary object.

`print.tsissm.diagnose` *Model Diagnostics Print method*

Description

Print method for class “tsissm.diagnose”

Usage

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

Arguments

`x`	an object of class “tsissm.duagnose” generated from calling `tsdiagnose`.
`...`	not currently used.

Value

Invisibly returns the original object and prints the output to console.

`residuals.tsissm.estimate` *Model Residuals*

Description

Extract the residual values from an estimated model.

Usage

```
## S3 method for class 'tsissm.estimate'
residuals(object, standardize = FALSE, transformed = TRUE, ...)
```

Arguments

`object`	an object of class “tsissm.estimate”.
`standardize`	whether to scale the residuals by the estimated standard deviation.
`transformed`	residuals based values in transformed space (Box Cox).
`...`	not currently used.

Details

For h>1, this is like performing an in-sample backtest starting at time 1 with fixed coefficients. The purpose of having the matrix of h-step ahead residuals is in order to calculate the 1:h covariance matrix as well as the cross 1:h covariance matrix when ensembling series at multiple horizons.

Value

An xts vector of the model residuals for h = 1, else a data.table with rows representing the first prediction date and columns the h-ahead forecast residuals.

Note

The function can use parallel functionality (for h>1) as long as the user has set up a `plan` using the future package.

`sigma.tsissm.estimate` *The Standard Deviation of the model*

Description

The Standard Deviation of the model

Usage

```
## S3 method for class 'tsissm.estimate'
sigma(object, ...)
```

Arguments

`object`	an object of class “tsissm.estimate”.
`...`	none.

Value

If the model was estimated using dynamic variance (GARCH model), then an xts matrix is returned with the estimated GARCH volatility else the estimated standard deviation of the residuals in the case of constant variance.

`simulate.tsissm.estimate` *Model Simulation*

Description

Simulation function for class “tsissm.estimate”.

Usage

```
## S3 method for class 'tsissm.estimate'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  h = 1,
  newxreg = NULL,
  sim_dates = NULL,
  bootstrap = FALSE,
  innov = NULL,
  innov_type = "q",
  pars = coef(object),
  init_states = tail(object$model$states, 1),
  init_res = NULL,
  init_sigma = NULL,
  ...
)

## S3 method for class 'tsissm.selection'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  h = 1,
  newxreg = NULL,
  sim_dates = NULL,
  bootstrap = FALSE,
  pars = coef(object),
  init_states = tail(object$model$states, 1),
  init_res = NULL,
  init_sigma = NULL,
  ...
)
```

Arguments

`object`	an object of class "tsissm.estimate".
`nsim`	the number of paths per complete set of time steps (h).
`seed`	a value specifying if and how the random number generator should be initialized ('seeded'). Either NULL or an integer that will be used in a call to set.seed before simulating the response vectors.
`h`	the number of time steps to simulate paths for. If this is NULL, it will use the same number of periods as in the original series.
`newxreg`	an optional matrix of regressors to use for the simulation if xreg was used in the estimation. If NULL and the estimated object had regressors, and h was also set to NULL, then the original regressors will be used.

`sim_dates`	an optional vector of simulation dates equal to h. If NULL will use the implied periodicity of the data to generate a regular sequence of dates after the first available date in the data.
`bootstrap`	whether to bootstrap the innovations from the estimated object by re-sampling from the empirical distribution.
`innov`	an optional vector of innovations (see innov_type). The length of this vector should be equal to nsim x horizon.
`innov_type`	if ‘innov’ is not NULL, then this denotes the type of values passed, with “q” denoting quantile probabilities (default and backwards compatible) and “z” for standardized errors.
`pars`	an optional named vector of model coefficients which override the estimated coefficients. No checking is currently performed on the adequacy of these coefficients.
`init_states`	An optional vector of states to initialize the forecast. If NULL, will use the first available states from the estimated model.
`init_res`	For a dynamic variance model, the initialization for the ARCH recursion of length equal to max(p,q).
`init_sigma`	For a dynamic variance model, the standard deviation initialization for the GARCH recursion of length equal to max(p,q).
`...`	not currently used.

Value

An object of class “tsissm.simulate” with slots for the simulated series and states.

`spy` *SPY ETF Adjusted Close*

Description

The adjusted daily closing price of the SPY ETF.

Usage

```
spy
```

Format

## ‘spy‘ A data.frame containing 7597 observations

date The YYYY-MM-DD date

spy The adjusted stock closing price.

Source

From Yahoo Finance.

summary.tsissm.estimate *Model Estimation Summary*

Description

Summary method for class “tsissm.estimate”

Usage

```
## S3 method for class 'tsissm.estimate'
summary(object, vcov_type = "H", digits = 4, ...)
```

Arguments

object	an object of class “tsissm.estimate”.
vcov_type	the type of standard errors based on the vcov estimate (see vcov).
digits	integer, used for number formatting. Optionally, to avoid scientific notation, set ‘options(scipen=999)’.
...	not currently used.

Value

A printout of the parameter summary, model type and some model metrics.

tsbacktest.tsissm.autospec *Walk Forward Model Backtest*

Description

Generates an expanding window walk forward backtest.

Usage

```
## S3 method for class 'tsissm.autospec'
tsbacktest(
  object,
  start = floor(length(object$y)/2),
  end = length(object$y),
  h = 1,
  estimate_every = 1,
  rolling = FALSE,
  weights_scheme = c("AIC", "BIC", "U"),
  weights = NULL,
  seed = NULL,
  trace = FALSE,
  ...
)

## S3 method for class 'tsissm.spec'
tsbacktest(
  object,
  start = floor(length(object$target$y_orig)/2),
  end = length(object$target$y_orig),
  h = 1,
  estimate_every = 1,
  rolling = FALSE,
  seed = NULL,
  trace = FALSE,
  ...
)
```

Arguments

`object`	an object of class "tsissm.spec" "tsissm.autospec".
`start`	numeric data index from which to start the backtest.
`end`	numeric data index on which to end the backtest. The backtest will end 1 period before that date in order to have at least 1 out of sample value to compare against.
`h`	forecast horizon. As the expanding window approaches the "end", the horizon will automatically shrink to the number of available out of sample periods.
`estimate_every`	number of periods at which the model is re-estimated and new predictions are generated (defaults to 1).
`rolling`	this indicates whether forecasts are made only on the estimation date (FALSE) or whether to filter the data 1 period at a time and forecast from the filtered data (TRUE).
`weights_scheme`	the weighting scheme to use when using ensembling (see note).
`weights`	a vector of fixed user supplied weights of length `top_n` when choosing "U" in the "weights_scheme".
`seed`	an value specifying if and how the random number generator should be initialized ('seeded'). Either NULL or an integer that will be used in a call to set.seed before simulating the response vectors.
`trace`	whether to show the progress bar. The user is expected to have set up appropriate handlers for this using the "progressr" package.
`...`	not used.

Value

A list with the following data.tables:

- prediction : the backtest table with forecasts and actuals
- metrics: a summary performance table showing metrics by forecast horizon (MAPE, MSLRE, BIAS and MIS if alpha was not NULL).

Note

The function can use parallel functionality as long as the user has set up a `plan` using the future package. Model ensembling is used when the input object is of class “tsissm.autospec” and top_n is greeter than 1. The following weighting schemes are available:

U: User supplied fixed weights.

AIC: Models are weighted based on their Akaike Information Criterion (AIC), favoring models with lower AIC values. The weights are computed as:

$$w_i = \frac{\exp(-0.5 \times \Delta_i)}{\sum_j \exp(-0.5 \times \Delta_j)}$$

where

$$\Delta_i = AIC_i - \min(AIC)$$

BIC: Similar to AIC-based weighting but uses the Bayesian Information Criterion (BIC) instead, which penalizes model complexity more strongly.

The final ensemble prediction is computed as:

$$\hat{y}_{\text{ensemble}} = \sum_i w_i \hat{y}_i$$

where w_i are the model weights and $\hat{y}_i$ are the individual model predictions. AIC-based weighting is preferred when models have different complexities but are estimated on the same dataset, while BIC-based weighting may be better suited for large sample sizes due to its stronger penalty on complexity.

See Also

[tsensemble()]

`tsdecompose.tsissm.estimate`

Model Decomposition

Description

Decomposes the estimated model or prediction into its component parts (states).

Usage

```
## S3 method for class 'tsissm.estimate'
tsdecompose(object, simplify = FALSE, start = 1, ...)

## S3 method for class 'tsissm.predict'
tsdecompose(object, simplify = FALSE, start = 1, ...)

## S3 method for class 'tsissm.simulate'
tsdecompose(object, simplify = FALSE, start = 1, ...)
```

Arguments

`object`	an object of class “tsissm.estimate” or “tsissm.predict”
`simplify`	simplification of the returned states aggregates the level and slope (if present) into a Trend component, all Seasonal components, all ARMA components and the error terms into an Irregular component. This may be useful when bagging is carried out (assuming equal lambda in the box-cox transform). This also simplifies the ability to created custom overrides of the Trend and rebuilt the predictive distribution.
`start`	whether to return the predicted states from t=1 to h or the states from t=0 to (h-1). The latter is sometimes useful as the sum of the states equals the predicted value (since the predictions are based on the lagged state).
`...`	not currently used.

Details

The 1-step ahead prediction is given by the following equation:

$$y_t = x_{t-1}w + \varepsilon_t$$

Because the decomposition pre lags the states so that the seed state is aligned with the error term, then summing the state distribution of each component with the returned error distribution will ensure that the exact same predicted value matched to the correct date is returned.

Value

For the estimated object, returns an xts matrix of the state components (including Irregular term). For the predicted object, a list with the simulated state components of class “tsmodel.predict” which includes the predictive distribution and original (estimated) series components.

`tsdiagnose.tsissm.estimate` *Model Diagnostics*

Description

Creates a short summary of model based diagnostics.

Usage

```
## S3 method for class 'tsissm.estimate'
tsdiagnose(object, plot = FALSE, ...)
```

Arguments

`object`	an object of class “tsissm.estimate”.
`plot`	whether to generate diagnostic plots to accompany summary.
`...`	not currently used.

Value

A list of tests including the weighted Ljung-Box test for residual autocorrelation, system forecastability test, and outlier dates using the Rosner test (`rosnerTest`). Optionally generates a plot of relevant diagnostics.

`tsensemble.tsissm.selection` *Ensembling of Models and Predictions*

Description

Ensembles estimated, predicted and simulated objects arising from the automatic selection method.

Usage

```
## S3 method for class 'tsissm.selection'
tsensemble(object, weights = NULL, start = 1, ...)

## S3 method for class 'tsissm.selection_predict'
tsensemble(object, weights = NULL, start = 1, ...)

## S3 method for class 'tsissm.selection_simulate'
tsensemble(object, weights = NULL, start = 1, ...)
```

Arguments

`object`	an object of class “tsissm.selection”, “tsissm.selection_predict” or “tsissm.selection_simulate”.
`weights`	a vector of weights equal to the number of models to be ensembled.
`start`	the index for the state decomposition (when all lambda equal). If 1, will return the predicted states else will return the lagged predicted states (which can be summed to return the predictive distribution).
`...`	not used.

Value

For the estimated object, a list with the weighted fit and errors, whilst for the predicted and simulated objects a list with the ensemble predictions as a “tsmodel.predict” object. If the models were all estimated with the same Box Cox lambda, then the weighted state decomposition is also returned inside the “tsmodel.predict” object. Additionally, for the predicted object, the ensemble analytic mean is also returned.

Note

Only the size of the weights is checked (should be equal to number of selected models), but not checks are performed on the values of the weights or whether they sum to 1. This is left to the user. When lambda is 1, the sum of component will be off by 1 versus the weighted distribution since the Box Cox transform contains an offset (so it is to be expected). To replicate the value of the predicted distribution by summing the decomposed components, the argument "start" should be set to 0 to return the lagged predicted components.

`tsequation.tsissm.estimate` — *Model Equation (LaTeX)*

Description

Generates a list of model equations in LaTeX.

Usage

```
## S3 method for class 'tsissm.estimate'
tsequation(object, ...)
```

Arguments

`object`	an object of class "tsissm.estimate".
`...`	not currently used.

Details

This method is called in the summary when the format output option chosen is "flextable".

Value

A list of equations in LaTeX which can be used in documents. This is a list with 3 slots for the observation, state and variance equations,

tsfilter.tsissm.estimate
Online Model Filtering

Description

Online filter which updates the states and fitted values using new data.

Usage

```
## S3 method for class 'tsissm.estimate'
tsfilter(object, y = NULL, newxreg = NULL, ...)

## S3 method for class 'tsissm.selection'
tsfilter(object, y = NULL, newxreg = NULL, ...)
```

Arguments

`object`	an object of class “tsissm.estimate” or “tsissm.selection”.
`y`	an xts vector of new information related to y. The function checks whether y contains indices (dates) which are not in the passed object and only filters new information.
`newxreg`	An xts matrix of new information related to external regressors (if those were used in the original model estimated).
`...`	not currently used.

Details

The new data is filtered (1 step ahead) using the last state of the object. Once this is complete, the object is updated with the new states and information so that the process can be continued on the same object as new information arrives.

Value

An object of class “tsissm.estimate” or “tsissm.selection”.

tsmetrics.tsissm.predict
Performance Metrics

Description

Performance metrics from an estimated or predicted tsissm model.

Usage

```
## S3 method for class 'tsissm.predict'
tsmetrics(object, actual, alpha = 0.1, ...)

## S3 method for class 'tsissm.estimate'
tsmetrics(object, ...)
```

Arguments

`object`	an object of class “tsissm.estimate” or “tsissm.predict”
`actual`	the actual data matched to the dates of the forecasts.
`alpha`	the coverage level for distributional forecast metrics.
`...`	not currently used.

Value

a data.frame of performance metrics including MAPE, MSLRE, BIAS and AIC for the estimate object and MAPE, MSLRE, BIAS, MASE, MIS and CRPS for predict object.

`tsmoments.tsissm.estimate` *Analytic Forecast Moments*

Description

Prediction function for class “tsissm.estimate”.

Usage

```
## S3 method for class 'tsissm.estimate'
tsmoments(
  object,
  h = 1,
  newxreg = NULL,
  init_states = NULL,
  transform = FALSE,
  ...
)
```

Arguments

`object`	an object of class “tsissm.estimate”.
`h`	the forecast horizon.
`newxreg`	a matrix of external regressors in the forecast horizon.

`init_states`	optional vector of states to initialize the forecast. If NULL, will use the last available state from the estimated model.
`transform`	whether to back transform the mean forecast. For the Box-Cox transformation this uses a Taylor Series expansion to adjust for the bias.
`...`	not currently used.

Details

For the constant variance model the conditional moments are given by:

$$E\left(y^{(\omega)}_{n+h|n}\right) = \mathbf{w}'\mathbf{F}^{h-1}\mathbf{x}_n$$

$$V\left(y^{(\omega)}_{n+h|n}\right) = \begin{cases} \sigma^2, & \text{if } h = 1, \\ \sigma^2\left[1 + \sum_{j=1}^{h-1} c_j^2\right], & \text{if } h \geq 2. \end{cases}$$

The backtransformed variance should be used with caution as it uses a higher-order expansion correction which is known to be inaccurate for h > 5. Instead, use the simulated distribution to extract this information.

Value

a list with a slot for the analytic mean and one for the process variance. In the case of dynamic variance, it returns an additional slot for the GARCH variance.

`tsprofile.tsissm.estimate`

Model Simulation Based Profiling

Description

Profiling of model dynamics using simulation/estimation/prediction.

Usage

```
## S3 method for class 'tsissm.estimate'
tsprofile(object, h = 1, nsim = 100, seed = NULL, trace = FALSE, ...)
```

Arguments

`object`	an object of class “tsissm.estimate”.
`h`	the forecast horizon on which to evaluate performance metrics.
`nsim`	the number of paths to generate.
`seed`	an object specifying if and how the random number generator should be initialized. See the simulate documentation for more details.
`trace`	whether to show the progress bar and additionally output verbose messages. The user is expected to have set up appropriate handlers for this using the `handlers` function from the “progressr” package.
`...`	not currently used.

Details

The function profiles an estimated model by simulating and then estimating multiple paths from the assumed DGP while leaving h values out for prediction evaluation. Each simulated path is equal to the size of the original dataset plus h additional values, and initialized with the initial state vector from the model. The resulting output contains the distribution of the MAPE, percent bias (BIAS) and mean squared log relative error (MSLRE) per horizon h. Since these matrices are of class “tsmodel.distribution” they can be readily plotted with the special purpose “plot” function for this class from the “tsmethods” package. Additionally, a data.table matrix is also returned with the distribution of the coefficients from each path estimation.

Value

An object of class “tsissm.profile”.

Note

The function can use parallel functionality as long as the user has set up a `plan` using the future package. The simulated states are checked for positivity and any paths which have negative values are excluded. If more than half of the paths have negative values then an error is raised and the function will stop.

`tsspec.tsissm.estimate` *Model Specification Extractor*

Description

Extracts a model specification (class “tsissm.spec”) from an object of class “tsissm.estimate”.

Usage

```
## S3 method for class 'tsissm.estimate'
tsspec(
  object,
  y = NULL,
  lambda = NULL,
  xreg = NULL,
  parameters = "initial",
  ...
)
```

Arguments

`object`	an object of class “tsissm.estimate”.
`y`	an optional new xts vector.
`lambda`	an optional lambda parameter for the Box Cox transformation (if previously used).

`xreg`	an optional matrix of regressors.
`parameters`	whether to set the parameters to their initial values, “initial” or the optimally estimated “optimal”.
`...`	not currently used.

Value

An object of class “tsissm.spec”.

Note

This function is used by other functions in the package such as the backtest which requires rebuilding the specification for each re-estimation step with updated data but keeping all else equal.

`us_retail_sales` *Advance Retail Sales: Retail Trade (US)*

Description

The dataset represents the monthly, non-seasonally adjusted advance estimate of sales from the retail trade based on data from a subsample of firms from the larger Monthly Retail Trade Survey.

Usage

```
us_retail_sales
```

Format

## ‘us_retail_sales‘ A data.frame containing 398 observations

date The YYYY-MM-DD date

co2 The sales value in millions of dollars.

Source

U.S. Census Bureau, Advance Retail Sales: Retail Trade [RSXFSN]. Retrieved from FRED, Federal Reserve Bank of St. Louis: `https://fred.stlouisfed.org/series/RSXFSN` on March 27, 2025.

`vcov.tsissm.estimate` *The Covariance Matrix of the Estimated Parameters*

Description

The Covariance Matrix of the Estimated Parameters

Usage

```
## S3 method for class 'tsissm.estimate'
vcov(object, adjust = FALSE, type = c("H", "OP", "QMLE", "NW"), ...)
```

Arguments

`object`	an object of class “tsissm.estimate”.
`adjust`	logical. Should a finite sample adjustment be made? This amounts to multiplication with n/(n-k) where n is the number of observations and k the number of estimated parameters.
`type`	valid choices are “H” for using the analytic hessian for the bread, “OP” for the outer product of gradients, “QMLE” for the Quasi-ML sandwich estimator (Huber-White), and “NW” for the Newey-West adjusted sandwich estimator (a HAC estimator).
`...`	additional parameters passed to the Newey-West bandwidth function to determine the optimal lags.

Value

The variance-covariance matrix of the estimated parameters.

Index

∗ **datasets**
    maunaloa, 12
    spy, 19
    us_retail_sales, 30

AIC (AIC.tsissm.estimate), 2
AIC.tsissm.estimate, 2
as_flextable.summary.tsissm.estimate, 3

BIC (BIC.tsissm.estimate), 4
BIC.tsissm.estimate, 4
bread.tsissm.estimate, 5

coef (coef.tsissm.estimate), 5
coef.tsissm.estimate, 5

estfun.tsissm.estimate, 6
estimate (estimate.tsissm.spec), 6
estimate.tsissm.spec, 6

fitted (fitted.tsissm.estimate), 8
fitted.tsissm.estimate, 8

handlers, *7*, *28*
hresiduals (hresiduals.tsissm.estimate), 8
hresiduals.tsissm.estimate, 8

issm_modelspec, 9

logLik (logLik.tsissm.estimate), 11
logLik.tsissm.estimate, 11

maunaloa, 12

nobs (nobs.tsissm.estimate), 12
nobs.tsissm.estimate, 12

plan, *7*, *17*, *22*, *29*
plot (plot.tsissm.estimate), 13
plot.tsissm.estimate, 13
predict (predict.tsissm.estimate), 13
predict.tsissm.estimate, 13
print.summary.tsissm.estimate, 15
print.tsissm.diagnose, 16

residuals (residuals.tsissm.estimate), 16
residuals.tsissm.estimate, 16
rosnerTest, *24*

sigma (sigma.tsissm.estimate), 17
sigma.tsissm.estimate, 17
simulate (simulate.tsissm.estimate), 17
simulate.tsissm.estimate, 17
spy, 19
summary (summary.tsissm.estimate), 20
summary.tsissm.estimate, 20

tsbacktest (tsbacktest.tsissm.autospec), 20
tsbacktest.tsissm.autospec, 20
tsdecompose (tsdecompose.tsissm.estimate), 22
tsdecompose.tsissm.estimate, 22
tsdiagnose, *16*
tsdiagnose (tsdiagnose.tsissm.estimate), 23
tsdiagnose.tsissm.estimate, 23
tsensemble (tsensemble.tsissm.selection), 24
tsensemble.tsissm.selection, 24
tsequation (tsequation.tsissm.estimate), 25
tsequation.tsissm.estimate, 25
tsfilter (tsfilter.tsissm.estimate), 26

tsfilter.tsissm.estimate, 26
tsmetrics *(*tsmetrics.tsissm.predict*)*, 26
tsmetrics.tsissm.predict, 26
tsmoments *(*tsmoments.tsissm.estimate*)*, 27
tsmoments.tsissm.estimate, 27
tsprofile *(*tsprofile.tsissm.estimate*)*, 28
tsprofile.tsissm.estimate, 28
tsspec *(*tsspec.tsissm.estimate*)*, 29
tsspec.tsissm.estimate, 29

us_retail_sales, 30

vcov, *20*
vcov *(*vcov.tsissm.estimate*)*, 31
vcov.tsissm.estimate, 31