Package ‘tsmethods’

September 23, 2024

Type Package

Title Time Series Methods

Version 1.0.2

Date 2024-08-23

Maintainer Alexios Galanos <alexios@4dscape.com>

Description Generic methods for use in a time series probabilistic framework, allowing for a common calling convention across packages. Additional methods for time series prediction ensembles and probabilistic plotting of predictions is included. A more detailed description is available at <https://www.nopredict.com/packages/tsmethods> which shows the currently implemented methods in the 'tsmodels' framework.

License GPL-2

Imports methods, zoo, xts, data.table

Encoding UTF-8

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

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

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

Repository CRAN

Date/Publication 2024-09-22 22:00:01 UTC

Contents

`distribution_list` *Multiple Distributions*

Description

validates and returns an object of class 'tsmodel.distribution_list' which holds a validated list of tsmodel.distribution objects for use in multivariate models.

Usage

```
distribution_list(distributions = NULL, names = NULL)
```

Arguments

`distributions`	a list with 'tsmodel.distribution' objects.
`names`	an optional vector of names for each slot in the list.

Details

The function will validate the distributions passed as belonging to class 'tsmodel.distribution', check whether they are similar in terms of number of draws (rows), horizon (columns) and dates. If the list is not named, then unless the "names" argument is passed, then will name the slots as series1, series2, etc.

Value

an object of class 'tsmodel.distribution_list'

Examples

```
x1 <- matrix(rnorm(100), 10, 10)
colnames(x1) <- as.character(as.Date(1:10, origin = "1970-01-01"))
x2 <- matrix(rnorm(100), 10, 10)
colnames(x2) <- as.character(as.Date(1:10, origin = "1970-01-01"))
class(x1) <- class(x2) <- "tsmodel.distribution"
distributions <- list(s1 = x1, s2 = x2)
L <- distribution_list(distributions)
str(L)
```

`ensemble_modelspec` *Ensemble specification*

Description

Creates and validates an ensemble specification.

Usage

```
ensemble_modelspec(...)
```

Arguments

`...`	objects of either all of class "tsmodel.predict" or "tsmodel.distribution" representing the probabilistic forecasts for the same horizon of optionally different models on the same series and with same number of draws. It is expected that the predictive distributions are based on joint simulated draws passed to the `innov` argument in the `predict` function of the supporting models. Instead of ... it is also possible to pass a list of the objects.

Value

An object of class "ensemble.spec".

`estimate` *Model Estimation*

Description

Generic method for estimating (fitting) a model.

Usage

```
estimate(object, ...)
```

Arguments

`object`	an object of the model specification.
`...`	additional parameters passed to the method.

Value

An object holding the results of the estimated model.

`estimate_ad` *Estimation method using AD*

Description

Generic method for estimating a model using automatic differentiation. This may be deprecated in the future in favor of just using the estimate method.

Usage

```
estimate_ad(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

The estimated autodiff model.

halflife *Half Life*

Description

Generic method for calculating the half-life of a model.

Usage

```
halflife(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Details

The half life is defined as the period it takes a series to reach half its long-term average values.

Value

The half life of the model.

pit *Probability Integral Transform (PIT)*

Description

Generic method for calculating the conditional probability integral transform given the data and estimated density

Usage

```
pit(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Details

The PIT is essentially the probabilities returned from the cumulative distribution function (*p) given the data and estimated value of the mean, conditional standard deviation and any other distributional parameters.

Value

The conditional probabilities.

---

plot.tsmodel.distribution

Plots of predictive distributions

Description

Plots for objects generated from probabilistic models returning a forecast distribution.

Usage

```
## S3 method for class 'tsmodel.distribution'
plot(
  x,
  y = NULL,
  median_color = "black",
  median_type = 1,
  median_width = 3,
  interval_quantiles = c(0.025, 0.975),
  gradient_color = "steelblue",
  interval_color = "cyan",
  interval_type = 2,
  interval_width = 2,
  ylim = NULL,
  ylab = "",
  n_x = NCOL(x),
  x_axes = TRUE,
  add = FALSE,
  zero_out = FALSE,
  date_class = "Date",
  ...
)

## S3 method for class 'tsmodel.predict'
plot(
  x,
  y = NULL,
  plot_original = TRUE,
  median_color = "black",
  median_type = 1,
  median_width = 3,
  interval_quantiles = c(0.025, 0.975),
  gradient_color = "steelblue",
  interval_color = "cyan",
  interval_type = 2,
  interval_width = 2,
  ylim = NULL,
  ylab = "",
  n_original = NULL,
  x_axes = TRUE,
  zero_out = FALSE,
  ...
)
```

Arguments

`x`	an object of class “tsmodel.distribution” or “tsmodel.predict”.
`y`	not used.
`median_color`	the color used for plotting the median value.
`median_type`	the line type for the median.
`median_width`	the width of the median line.
`interval_quantiles`	the quantiles to include in the plot.
`gradient_color`	the gradient color to use for the distribution.
`interval_color`	the color of the quantile lines.
`interval_type`	the line type for the quantiles.
`interval_width`	the width of the quantile lines.
`ylim`	user specified limits for the y-axis.
`ylab`	user specified label for y-axis.
`n_x`	the number of time periods from the end to plot for x.
`x_axes`	whether to print the x-axis (usually time/date).
`add`	whether to overlay another “tsmodel.distribution” on top of a current plot. This will only plot the median and quantiles and not the full distribution with gradient color.
`zero_out`	whether to zero any negative value in the prediction intervals.
`date_class`	when overlaying (add argument) one distribution (“tsmodel.distribution” on top of another, particularly if it is added to a plot based on “tsmodel.predict”, then in order for this to work correctly the two date classes have to be the same. The “tsmodel.predict” plot method infers the class from the original time series which is contained in the object. Since the “tsmodel.distribution” carries no additional information other than the column names of the date/time stamps, then it is upto the user to supply what this should be.
`...`	additional arguments to the plot.default function.
`plot_original`	whether to include the original dataset in the plot.
`n_original`	the number of time periods from the end to plot for the original series. Defaults to plotting the whole series.

Value

a plot of the predicted distribution.

Note

Any matrix representing a distribution of values at points in time, with time in the columns (date labels in columns) and the distribution in rows can be set to class "tsmodel.distribution" and then passed to the plot function which can generate a nice distribution plot. The "tsmodel.predict" is a list with the posterior predictive (or simulated) distribution (the "tsmodel.distribution") in addition to the original series (original.series) of class zoo or xts.

Examples

```
library(xts)
months <- c("01","02","03","04","05","06","07","08","09","10","11","12")
dates <- as.Date(paste0(sort(rep(1973:1978, 12)),"-",rep(months, 6), "-",rep("01",12*6)))
y <- xts(as.numeric(USAccDeaths), dates)
samples <- do.call(cbind, lapply(1:12,
function(i){sample(as.numeric(y[format(index(y),"%m") == months[i]]), 100, replace = TRUE)}))
predict_dates <- as.Date(paste0(rep(1979, 12),"-",months, "-",rep("01",12)))
expected_value <- colMeans(samples)
p <- list()
colnames(samples) <- as.character(predict_dates)
class(samples) <- "tsmodel.distribution"
p$original_series <- y
p$distribution <- samples
p$mean <- xts(expected_value, predict_dates)
class(p) <- "tsmodel.predict"
actuals_available <- c(7798,7406,8363,8460,9217,9316)
plot(p, main = "USAccDeaths Resample Based Forecast", n_original = 12*3,
gradient_color = "orange", interval_color = "deepskyblue", median_width = 1.5)
points(predict_dates[1:6], actuals_available, col = "green", cex = 1.5)
```

`tsaggregate` *Time Series Aggregation*

Description

Generic method for aggregating of both observed series as well as certain models such as homogeneous coefficients whose dynamics aggregate (such as in the Vector ETS model).

Usage

```
tsaggregate(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

The aggregated distribution.

`tsbacktest` *Time Series Model Backtesting*

Description

Generic method for backtesting of a time series model.

Usage

```
tsbacktest(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

An object holding the results of the backtest.

`tsbenchmark` *Time Series Model Benchmarking*

Description

Generic method for benchmarking models.

Usage

```
tsbenchmark(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

An object holding the model benchmark which can be printed and referenced.

tscalibrate *Prediction Calibration method*

Description

Generic method for calibrating a model, targeting specific objectives.

Usage

```
tscalibrate(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

An object with details of the calibrated target.

tscokurt *Co-Kurtosis*

Description

Generic method for the co-kurtosis of a model.

Usage

```
tscokurt(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Details

The method calculates the conditional co-kurtosis of a model. Applications of this can be found in the Independent Factor Conditional Density Models.

Value

The co-kurtosis tensor (folded or unfolded).

`tsconvert` *Conversion method from one object to another*

Description

Generic special purpose method for converting one object to another.

Usage

```
tsconvert(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Details

This method can be used to convert one model to another when those models are related, one set of parameters to a different parameterization, or any other use case which involves some meaningful conversion in the context of the model being implemented for.

Value

The converted object.

`tsconvert.tsmodel.distribution` *Convert a distribution object to a long form data.table*

Description

Converts an object of class “tsmodel.distribution” or “tsmodel.distribution_list” to a long form data.table object.

Usage

```
## S3 method for class 'tsmodel.distribution'
tsconvert(object, to = "data.table", name = NULL, ...)

## S3 method for class 'tsmodel.distribution_list'
tsconvert(object, to = "data.table", ...)
```

Arguments

`object`	a “tsmodel.distribution” or “tsmodel.distribution_list” object.
`to`	output format. Currently only “data.table” supported.
`name`	an optional string for the name of the series which will be added to the table (only for the “tsmodel.distribution”, as the list object is already validated with names).
`...`	not currently used.

Value

a data.table object

Examples

```
x1 <- matrix(rnorm(100), 10, 10)
colnames(x1) <- as.character(as.Date(1:10, origin = "1970-01-01"))
class(x1) <- "tsmodel.distribution"
head(tsconvert(x1, name = "SeriesA"))
```

`tsconvolve` *Convolution of Distributions*

Description

Generic method for convolution of conditional distribution.

Usage

```
tsconvolve(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Details

The method is meant to apply the Fast Fourier Transform to the characteristic function of a distribution. Applications of this can be found in the Independent Factor Conditional Density Models.

Value

The convolved density.

`tscor` *Correlation matrix.*

Description

Generic method for extracting the correlation matrix from a multivariate model or the autocorrelation matrix of a univariate model.

Usage

```
tscor(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

A correlation matrix or array of matrices (time varying) or other custom object related to this.

`tscoskew` *Co-Skewness*

Description

Generic method for the co-skewness of a model.

Usage

```
tscoskew(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Details

The method calculates the conditional co-skewness of a model. Applications of this can be found in the Independent Factor Conditional Density Models.

Value

The co-skewness tensor (folded or unfolded).

tscov *Covariance matrix.*

Description

Generic method for extracting the covariance matrix from a multivariate model or the autocovariance matrix of a univariate model.

Usage

```
tscov(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

A covariance matrix or array of matrices (time varying) or other custom object related to this.

tsdecompose *Time Series Model Decomposition*

Description

Generic method for decomposition of an estimated time series object.

Usage

```
tsdecompose(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

An object of the model decomposition.

tsdiagnose *Extract diagnostic model information*

Description

Generic method for extracting the diagnostics from an object.

Usage

```
tsdiagnose(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

The model diagnostics.

tsensemble.ensemble.spec *Ensembles predictions using their posterior predictive or simulated distribution*

Description

Generic method for ensembles of posterior predictive or simulated distributions spanning the same forecast horizon.

Usage

```
## S3 method for class 'ensemble.spec'
tsensemble(object, weights = NULL, ...)

tsensemble(object, weights = NULL, ...)
```

Arguments

`object`	currently only dispatches based on objects of class "ensemble.spec" which have been validated for ensembling.
`weights`	a numeric vector of length equal to the length of the ensembling predictions which represent the ensembling weights.
`...`	additional parameters passed to the method.

Details

Returns the weighted distribution, under the assumption that the predictions were generated using a joint error distribution whose values were passed to the `innov` argument of the predict function used for each model.

Value

An object of class “tsmodel.predict” or “tsmodel.distribution” depending on the input class in `ensemble_modelspec`.

The ensembled distribution.

`tsequation` *Model Equation Extractor*

Description

Generic method for extracting the equation of a model into either latex or some other format.

Usage

```
tsequation(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

The model equation(s) in a specified format.

`tsfilter` *Online Time Series Filter*

Description

Generic method for (online) time series filtering.

Usage

```
tsfilter(object, y, newxreg = NULL, ...)
```

Arguments

`object`	an object.
`y`	new outcome data to filter on.
`newxreg`	new optional regressors to filter on.
`...`	additional parameters passed to the method.

Value

The filtered value(s).

`tsgrowth.tsmodel.predict` *Growth Calculation*

Description

Generic method for calculating the growth distribution from an object.

Usage

```
## S3 method for class 'tsmodel.predict'
tsgrowth(object, d = 1, type = c("diff", "simple", "log"), ...)

tsgrowth(object, ...)
```

Arguments

`object`	an object.
`d`	the period back to look at for growth calculations.
`type`	the type of growth calculation. “diff” is simply the difference in values over n periods, “simple” if the rate of change and “log” the difference in logs.
`...`	additional parameters passed to the method.

Value

an object of class “tsmodel.predict” transformed to a growth distribution.

The growth distribution.

`tsmetrics` *Time Series Performance Metrics Method*

Description

Generic method for generating time series performance metrics.

Usage

```
tsmetrics(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

The performance metrics.

`tsmoments` *Analytic Moments*

Description

Generic method for generating the analytic (or approximated) moments of the model or forecast (usually mean and variance).

Usage

```
tsmoments(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

The moments of the class it was applied to (e.g. estimated or predicted object).

`tsprofile` *Profile a time series model*

Description

Generic method for profiling a time series model by simulating from the estimated data generating process and measuring the performance under different levels of system noise.

Usage

```
tsprofile(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

The profiled model with information such as root mean squared error per parameter.

`tsreport` *Report Method*

Description

Generic method for generating reports in different output formats.

Usage

```
tsreport(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

An object holding a ready to parse report.

tsspec *Extract specification object from estimation object*

Description

Generic method for extracting the specification object from an S3 object.

Usage

```
tsspec(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

A specification object extracted from an estimated or other model which retains this information.

unconditional *Unconditional Value*

Description

Generic method for extracting the unconditional value of a model.

Usage

```
unconditional(object, ...)
```

Arguments

`object`	an object.
`...`	additional parameters passed to the method.

Value

The unconditional value or values (e.g. mean and variance).

Index