Package 'forecastHybrid'

July 22, 2018

Title Convenient Functions for Ensemble Time Series Forecasts

Version 3.0.14

Date 2018-07-22

Description Convenient functions for ensemble forecasts in R combining approaches from the 'forecast' package. Forecasts generated from auto.arima(), ets(), thetam(), nnetar(), stlm(), and tbats() can be combined with equal weights, weights based on in-sample errors (introduced by Bates & Granger (1969) <doi:10.1057/jors.1969.103>), or cross-validated weights. Cross validation for time series data with user-supplied models and forecasting functions is also supported to evaluate model accuracy.

Depends R (>= 3.1.1), forecast (>= 8.1),

Imports doParallel (>= 1.0.10), foreach (>= 1.4.3), ggplot2 (>= 2.2.0), zoo (>= 1.7)

Suggests GMDH, knitr, rmarkdown, roxygen2, testthat

VignetteBuilder knitr

License GPL-3

URL https://gitlab.com/dashaub/forecastHybrid

BugReports https://github.com/ellisp/forecastHybrid/issues

LazyData true

RoxygenNote 6.0.1

ByteCompile true

NeedsCompilation no

Author David Shaub [aut, cre], Peter Ellis [aut]

Maintainer David Shaub <davidshaub@gmx.com>

Repository CRAN

Date/Publication 2018-07-22 20:40:03 UTC

R topics documented:

`accuracy.cvts` *Accuracy measures for cross-validated time series*

Description

Returns range of summary measures of the cross-validated forecast accuracy for `cvts` objects.

Usage

```
## S3 method for class 'cvts'
accuracy(f, ...)
```

Arguments

`f`	a `cvts` objected created by `cvts`.
`...`	other arguments (ignored).

Details

Currently the method only implements ME, RMSE, and MAE. The accuracy measures MPE, MAPE, and MASE are not calculated. The accuracy is calculated for each forecast horizon up to maxHorizon

Author(s)

David Shaub

---

accuracy.hybridModel *Accuracy measures for hybridModel objects*

Description

Accuracy measures for hybridModel objects.

Usage

```
## S3 method for class 'hybridModel'
accuracy(f, individual = FALSE, ...)
```

Arguments

f	the input hybridModel.
individual	if TRUE, return the accuracy of the component models instead of the accuracy for the whole ensemble model.
...	other arguments (ignored).

Details

Return the in-sample accuracy measures for the component models of the hybridModel

Value

The accuracy of the ensemble or individual component models.

Author(s)

David Shaub

See Also

accuracy

checkModelArgs *Helper function to test all the model arguments (e.g. a.args, e.args, etc)*

Description

Helper function to test all the model arguments (e.g. a.args, e.args, etc)

Usage

```
checkModelArgs(modelArguments, models)
```

Arguments

modelArguments	A list of containing the model arguments
models	A character vector containing all the model codes

checkParallelArguments *Helper function to check the that the parallel arguments are valid*

Description

Helper function to check the that the parallel arguments are valid

Usage

```
checkParallelArguments(parallel, num.cores)
```

Arguments

parallel	A logic to indicate if paralle processing should be used
num.cores	An integer for the number of threads to use

cvts *Cross validation for time series*

Description

Perform cross validation on a time series.

Usage

```
cvts(x, FUN = NULL, FCFUN = NULL, rolling = FALSE, windowSize = 84,
  maxHorizon = 5, horizonAverage = FALSE, xreg = NULL,
  saveModels = ifelse(length(x) > 500, FALSE, TRUE),
  saveForecasts = ifelse(length(x) > 500, FALSE, TRUE), verbose = TRUE,
  num.cores = 2L, extraPackages = NULL, ...)
```

Arguments

x	the input time series.
FUN	the model function used. Custom functions are allowed. See details and examples.
FCFUN	a function that process point forecasts for the model function. This defaults to forecast. Custom functions are allowed. See details and examples. See details.
rolling	should a rolling procedure be used? If TRUE, nonoverlapping windows of size maxHorizon will be used for fitting each model. If FALSE, the size of the dataset used for training will grow by one each iteration.
windowSize	length of the window to build each model. When rolling == FALSE, the each model will be fit to a time series of this length, and when rolling == TRUE the first model will be fit to a series of this length and grow by one each iteration.
maxHorizon	maximum length of the forecast horizon to use for computing errors.
horizonAverage	should the final errors be an average over all forecast horizons up to maxHorizon instead of producing metrics for each individual horizon?
xreg	External regressors to be used to fit the model. Only used if FUN accepts xreg as an argument. FCFUN is also expected to accept it (see details)
saveModels	should the individual models be saved? Set this to FALSE on long time series to save memory.
saveForecasts	should the individual forecast from each model be saved? Set this to FALSE on long time series to save memory.
verbose	should the current progress be printed to the console?
num.cores	the number of cores to use for parallel fitting. If the underlying model that is being fit also utilizes parallelization, the number of cores it is using multiplied by 'num.cores' should not exceed the number of cores avaialble on your machine.
extraPackages	if a custom 'FUN' or 'FCFUN' is being used that requires packages to be loaded, these can be passed here
...	Other arguments to be passed to the model function FUN

Details

Cross validation of time series data is more complicated than regular k-folds or leave-one-out cross validation of datasets without serial correlation since observations x_t and x_{t+n} are not independent. The `cvts()` function overcomes this obstacle using two methods: 1) rolling cross validation where an initial training window is used along with a forecast horizon and the initial window used for training grows by one observation each round until the training window and the forecast horizon capture the entire series or 2) a non-rolling approach where a fixed training length is used that is shifted forward by the forecast horizon after each iteration.

For the rolling approach, training points are heavily recycled, both in terms of used for fitting and in generating forecast errors at each of the forecast horizons from `1:maxHorizon`. In constrast, the models fit with the non-rolling approach share less overlap, and the predicted forecast values are also only compared to the actual values once. The former approach is similar to leave-one-out cross validation while the latter resembles k-fold cross validation. As a result, rolling cross validation requires far more iterations and computationally takes longer to complete, but a disadvantage of the non-rolling approach is the greater variance and general instability of cross-validated errors.

The `FUN` and `FCFUN` arguments specify which function to use for generating a model and forecasting, respectively. While the functions from the "forecast" package can be used, user-defined functions can also be tested, but the object returned by `FCFUN` must accept the argument `h` and contain the point forecasts out to this horizon `h` in slot `$mean` of the returned object. An example is given with a custom model and forecast.

For small time series (default `length <= 500`), all of the individual fit models are included in the final `cvts` object that is returned. This can grow quite large since functions such as `auto.arima` will save fitted values, residual values, summary statistics, coefficient matrices, etc. Setting `saveModels = FALSE` can be safely done if there is no need to examine individual models fit at every stage of cross validation since the forecasts from each fold and the associated residuals are always saved.

External regressors are allowed via the `xreg` argument. It is assumed that both `FUN` and `FCFUN` accept the `xreg` parameter if `xreg` is not `NULL`. If `FUN` does not accept the `xreg` parameter a warning will be given. No warning is provided if `FCFUN` does not use the `xreg` parameter.

Author(s)

David Shaub

See Also

`accuracy.cvts`

Examples

```
series <- subset(AirPassengers, end = 50)
cvmod1 <- cvts(AirPassengers, FUN = stlm,
               windowSize = 25, maxHorizon = 12)
accuracy(cvmod1)

# We can also use custom model functions for modeling/forecasting
stlmClean <- function(x){stlm(tsclean(x))}
series <- subset(austres, end = 38)
cvmodCustom <- cvts(series, FUN = stlmClean, windowSize = 26, maxHorizon = 6)
```

```
accuracy(cvmodCustom)


## Not run:
cvmod2 <- cvts(USAccDeaths, FUN = ets,
               saveModels = FALSE, saveForecasts = FALSE,
               windowSize = 36, maxHorizon = 12)

cvmod3 <- cvts(AirPassengers, FUN = hybridModel,
               FCFUN = forecast, rolling = TRUE, windowSize = 48,
               maxHorizon = 12)

## End(Not run)

# Use the rwf() function from the "forecast" package.
# This function does not have a modeling function and
# instead calculates a forecast on the time series directly
series <- subset(AirPassengers, end = 26)
rwcv <- cvts(series, FCFUN = rwf, windowSize = 24, maxHorizon = 1)
```

`extractForecasts` *Extract cross validated rolling forecasts*

Description

Obtain cross validated forecasts when rolling cross validation is used. The object is not inspected to see if it was fit using a rolling origin

Usage

```
extractForecasts(cv, horizon = 1)
```

Arguments

`cv`	An object of class cvts
`horizon`	The forecast horizon from each fold to extract

Details

Combine the cross validated forecasts fit with a rolling origin. This may be useful to visualize and investigate the cross validated performance of the model

Value

Forecasts computed via a rolling origin

Author(s)

Ganesh Krishnan

Examples

```
cv <- cvts(AirPassengers, FUN = stlm, FCFUN = forecast,
        rolling = TRUE, windowSize = 134, horizon = 2)

extractForecasts(cv)
```

`fitted.hybridModel` *Extract Model Fitted Values*

Description

Extract the model fitted values from the `hybridModel` object.

Usage

```
## S3 method for class 'hybridModel'
fitted(object, individual = FALSE, ...)
```

Arguments

`object`	the input hybridModel.
`individual`	if `TRUE`, return the fitted values of the component models instead of the fitted values for the whole ensemble model.
`...`	other arguments (ignored).

Value

The fitted values of the ensemble or individual component models.

See Also

`accuracy`

`forecast.hybridModel` *Hybrid forecast*

Description

Forecast method for hybrid models.

Usage

```
## S3 method for class 'hybridModel'
forecast(object, h = ifelse(object$frequency > 1, 2 *
  object$frequency, 10), xreg = NULL, level = c(80, 95), PI = TRUE,
  fan = FALSE, PI.combination = c("extreme", "mean"), ...)
```

Arguments

object	a hybrid time series model fit with hybridModel.
h	number of periods for forecasting. If xreg is used, h is ignored and the number of forecast periods is set to the number of rows of xreg.
xreg	future values of regression variables (for use if one of the ensemble methods used in creating the hybrid forecast was auto.arima, nnetar, or stlm and the model(s) used xreg in the fit)
level	confidence level for prediction intervals. This can be expressed as a decimal between 0.0 and 1.0 or numeric between 0 and 100.
PI	should prediction intervals be produced? If a nnetar model is in the ensemble, this can be quite slow, so disabling prediction intervals will speed up the forecast generation. If FALSE, the arguments level and fan are ignored.
fan	if TRUE, level is set to seq(51, 99, by = 3). This is suitable for fan plots.
PI.combination	Method for combining the prediction intervals from each of the component forecasts. Supplying "mean" will simply average each of the lower/upper intervals from each model without using the model weights used for the point forecasts. The default value "extreme" will take the most pessimistic intervals (i.e. the highest upper interval from all the component models and the lowest prediction interval from all of the component models').
...	other arguments passed to the individual forecast generic methods.

Details

if xreg was used in construcing the hybridModel, it must also be passed into forecast.hybridModel.

While prediction intervals are produced for the final ensemble forecast model, these should be viewed conservatively as insights to the forecast's uncertainty. Currently these are constructed using the most extreme interval from each component model for each horizon, so the composite prediction intervals do not have statistical guarantees of asymptotic efficiency. More sophisticated and rigorous techniques are planned, however, particularly when cross validation approaches are used.

Value

An object of class forecast.

Author(s)

David Shaub

See Also

hybridModel

Examples

```
## Not run:
mod <- hybridModel(AirPassengers)
fc <- forecast(mod)

# View the point forecasts
fc$mean

# View the upper prediction interval
fc$upper

# View the lower prediction interval
fc$lower

# Plot the forecast
plot(fc)

## End(Not run)
```

---

forecast.thetam *Forecast using a Theta model*

Description

Returns forecasts and other information for univariate Theta "models"

Usage

```
## S3 method for class 'thetam'
forecast(object, h = ifelse(object$m > 1, 2 * object$m, 10),
  level = c(80, 95), fan = FALSE, ...)
```

Arguments

object	An object of class "thetam. Usually the result of a call to link{thetam}.
h	Number of periods for forecasting
level	Confidence level for prediction intervals
fan	If TRUE, level is set to seq(51, 99, by = 3). This is suitable for fan plots.
...	Ignored

Value

An object of class forecast

Author(s)

Peter Ellis

See Also

thetam

Examples

```
mod1 <- thetam(Nile)
fc1 <- forecast(mod1)
plot(fc1)
```

`getModel` *Return a forecast model function for a given model character*

Description

Convert the single-letter representation used in the "forecastHybrid" package to the corresponding model function from the "forecast" package

Usage

```
getModel(modelCharacter)
```

Arguments

`modelCharacter`	a single character representing one of the models from the `models` argument passed to hybridModel

See Also

hybridModel

Examples

```
forecastHybrid:::getModel("a")
forecastHybrid:::getModel("s")
forecastHybrid:::getModel("z")
```

getModelName *Translate character to model name*

Description

Convert the single-letter representation used in the "forecastHybrid" package to the corresponding function name from the "forecast" package

Usage

```
getModelName(modelCharacter)
```

Arguments

modelCharacter	a single character representing one of the models from the models argument passed to hybridModel

See Also

hybridModel

Examples

```
forecastHybrid:::getModelName("a")
forecastHybrid:::getModelName("s")
forecastHybrid:::getModelName("z")
```

hybridModel *Hybrid time series modelling*

Description

Create a hybrid time series model with two to five component models.

Usage

```
hybridModel(y, models = "aefnst", lambda = NULL, a.args = NULL,
  e.args = NULL, n.args = NULL, s.args = NULL, t.args = NULL,
  z.args = NULL, weights = c("equal", "insample.errors", "cv.errors"),
  errorMethod = c("RMSE", "MAE", "MASE"), cvHorizon = frequency(y),
  windowSize = 84, horizonAverage = FALSE, parallel = FALSE,
  num.cores = 2L, verbose = TRUE)
```

Arguments

y	A numeric vector or time series.
models	A character string of up to seven characters indicating which contributing models to use: a (auto.arima), e (ets), f (thetam), n (nnetar), s (stlm), t (tbats), and z (snaive).
lambda	Box-Cox transformation parameter. Ignored if NULL. Otherwise, data transformed before model is estimated.
a.args	an optional list of arguments to pass to auto.arima. See details.
e.args	an optional list of arguments to pass to ets. See details.
n.args	an optional list of arguments to pass to nnetar. See details.
s.args	an optional list of arguments to pass to stlm. See details.
t.args	an optional list of arguments to pass to tbats. See details.
z.args	an optional list of arguments to pass to snaive. See details.
weights	method for weighting the forecasts of the various contributing models. Defaults to equal, which has shown to be robust and better in many cases than giving more weight to models with better in-sample performance. Cross validated errors–implemented with link{cvts} should produce the best forecast, but the model estimation is also the slowest. Note that extra arguments passed in a.args, e.args, n.args, s.args, and t.args are not used during cross validation. See further explanation in cvts. Weights utilizing in-sample errors are also available but not recommended.
errorMethod	method of measuring accuracy to use if weights are not to be equal. Root mean square error (RMSE), mean absolute error (MAE) and mean absolute scaled error (MASE) are supported.
cvHorizon	If weights = "cv.errors", this controls which forecast to horizon to use for the error calculations.
windowSize	length of the window to build each model, only used when weights = "cv.errors".
horizonAverage	If weights = "cv.errors", setting this to TRUE will average all forecast horizons up to cvHorizon for calculating the errors instead of using the single horizon given in cvHorizon.
parallel	a boolean indicating if parallel processing should be used between models. Parallelization will still occur within individual models that suport it and can be controlled using a.args and t.args.
num.cores	If parallel=TRUE, how many cores to use.
verbose	Should the status of which model is being fit/cross validated be printed to the terminal?

Details

The hybridModel function fits multiple individual model specifications to allow easy creation of ensemble forecasts. While default settings for the individual component models work quite well in most cases, fine control can be exerted by passing detailed arguments to the component models in the a.args, e.args, n.args, s.args, and t.args lists. Note that if xreg is passed to the a.args,

n.args, or s.args component models it must be passed as a dataframe instead of the matrix object that the "forecast" package functions usually accept. This is due to a limitation in how the component models are called.

Characteristics of the input series can cause problems for certain types of models and paramesters. For example, stlm models require that the input series be seasonal; furthemore, the data must include at least two seasons of data (i.e. length(y) >= 2 * frequency(y)) for the decomposition to succeed. If this is not the case, hybridModel() will remove the stlm model so an error does not occur. Similarly, nnetar models require that length(y) >= 2 * frequency(y), so these models will be removed if the condition is not satisfied The ets model does not handle a series well with a seasonal period longer than 24 and will ignore the seasonality. In this case, hybridModel() will also drop the ets model from the ensemble.

Value

An object of class hybridModel. The individual component models are stored inside of the object and can be accessed for all the regular manipulations available in the forecast package.

Author(s)

David Shaub

See Also

forecast.hybridModel, auto.arima, ets, thetam, nnetar, stlm, tbats

Examples

```
## Not run:

# Fit an auto.arima, ets, thetam, nnetar, stlm, and tbats model
# on the time series with equal weights
mod1 <- hybridModel(AirPassengers)
plot(forecast(mod1))

# Use an auto.arima, ets, and tbats model with weights
# set by the MASE in-sample errors
mod2 <- hybridModel(AirPassengers, models = "aet",
weights = "insample.errors", errorMethod = "MASE")

# Pass additional arguments to auto.arima() to control its fit
mod3 <- hybridModel(AirPassengers, models = "aens",
a.args = list(max.p = 7, max.q = 7, approximation = FALSE))

# View the component auto.arima() and stlm() models
mod3$auto.arima
mod3$stlm

## End(Not run)
```

is.hybridModel *Test if the object is a hybridModel object*

Description

Test if the object is a hybridModel object.

Usage

```
is.hybridModel(x)
```

Arguments

x	the input object.

Value

A boolean indicating if the object is a hybridModel is returned.

plot.hybridModel *Plot a hybridModel object*

Description

Plot a representation of the hybridModel.

Usage

```
## S3 method for class 'hybridModel'
plot(x, type = c("fit", "models"), ggplot = FALSE,
  ...)
```

Arguments

x	an object of class hybridModel to plot.
type	if type = "fit", plot the original series and the individual fitted models. If type = "models", use the regular plot methods from the component models, i.e. plot.Arima, plot.ets, plot.tbats. Note: no plot methods exist for nnetar and stlm objects, so these will not be plotted with type = "models".
ggplot	should the autoplot function be used (when available) for the plots?
...	other arguments passed to plot.

Details

For `type = "fit"`, the original series is plotted in black. Fitted values for the individual component models are plotted in other colors. For `type = "models"`, each individual component model is plotted. Since there is not plot method for `stlm` or `nnetar` objects, these component models are not plotted.

Value

None. Function produces a plot.

Author(s)

David Shaub

See Also

`hybridModel`

Examples

```
## Not run:
hm <- hybridModel(woolyrnq, models = "aenst")
plot(hm, type = "fit")
plot(hm, type = "models")

## End(Not run)
```

---

plot.thetam *Plot components from Theta model*

Description

Produces a plot of the level components from the ETS model underlying a Theta model

Usage

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

Arguments

`x`	Object of class "thetam".
`...`	Other plotting parameters passed through to `plot`

Details

The "state" component of the plot comes from the model `ets(..., model = "ANN")` that was fit as part of the theta method. The "seasonal" component is the multipliers from multiplicative classical decomposition seasonal adjustment that is performed before the `ets` model is fit. The "linear" component shows the direction and slope of drift that is used in the forecasting to come.

Value

None. Function produces a plot.

Author(s)

Peter Ellis

See Also

`thetam`

Examples

```
model <- thetam(wineind)
plot(model)
```

`prepareTimeseries` *Helper function to validate and clean the input timeseries*

Description

Helper function to validate and clean the input timeseries

Usage

```
prepareTimeseries(y)
```

Arguments

`y`	The input timeseries

print.hybridModel *Print information about the hybridModel object*

Description

Print information about the hybridModel object.

Usage

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

Arguments

x	the input hybridModel object.
...	other arguments (ignored).

Details

Print the names of the individual component models and their weights.

removeModels *Helper function to remove models that require more data*

Description

Helper function to remove models that require more data

Usage

```
removeModels(y, models)
```

Arguments

y	The input timeseries
models	The model codes to test

residuals.hybridModel *Extract Model Residuals*

Description

Extract the model residuals from the hybridModel object.

Usage

```
## S3 method for class 'hybridModel'
residuals(object, individual = FALSE, ...)
```

Arguments

object	The input hybridModel.
individual	If TRUE, return the residuals of the component models instead of the residuals for the whole ensemble model.
...	Other arguments (ignored).

Value

The residuals of the ensemble or individual component models.

See Also

accuracy

summary.hybridModel *Print a summary of the hybridModel object*

Description

Print a summary of the hybridModel object

Usage

```
## S3 method for class 'hybridModel'
summary(x)
```

Arguments

x	the input hybridModel object.

Details

Print the names of the individual component models and their weights.

thetam *Theta method 'model'*

Description

Create a model object as an interim step to a theta method forecast.

Usage

```
thetam(y)
```

Arguments

y	A numeric vector or time series.

Details

This fits an exponential smoothing state space model with `model = 'ANN'` to `y`, having first performed classic multiplicative seasonal adjustment. A drift value is also calculated by `lsfit(0:(length(y) - 1), y)$coef[`
In combination with `forecast.thetam()`, this provides identical results to `forecast::thetaf(...)`.
The purpose of splitting it into a 'model' and 'forecast' functions is to make the approach consistent with other modelling / forecasting approaches used in `hybridModel()`.

Value

An object of class `thetam`

Author(s)

Peter Ellis

See Also

`forecast.thetam`

Examples

```
mod1 <- thetam(Nile)
plot(mod1)
```

`tsCombine` *Combine multiple sequential time series*

Description

Combine multiple ts objects into a single ts object. It is assumed that the ts objects provided are sequential. In other words, it is assumed that a valid time series object can actually be constructed from the provided objects. The start time and frequency of the combined object will correspond to the start time and frequency of the first provided object

Usage

```
tsCombine(...)
```

Arguments

...	ts objects to combine

Details

Combine sequential time series objects into a single time series object. This might be useful, for example, when you want to combine the training and validation time series objects for plotting. The function assumes that the provided objects have no overlap. For example, a valid argument set would have two time series with periods from Jan-Dec 2015 and Jan-Dec 2016. An invalid set would be two time series t1 and t2 with periods from Jan-Dec 2015 and Aug 2015-Dec 2016 respectively. In that case, there is overlap between t1 and t2. The return value will depend on the order in which the arguments are provided. If the function call is tsCombine(t1, t2), the overlapping portion of t1 and t2 (Aug-Dec 2015 in this example), would have values from t1 as long as they are not NA. If the call is tsCombine(t2, t1), it will have values from t2 as long as they are not NA.

Value

A combined ts object generated from the individual ts objects

Author(s)

Ganesh Krishnan

Examples

```
tsCombine(window(AirPassengers, end = c(1951, 12)), window(AirPassengers, start = c(1952, 1)))
```

tsPartition *Generate training and test indices for time series cross validation*

Description

Training and test indices are generated for time series cross validation. Generated indices are based on the training windowSize, forecast horizons and whether a rolling or non-rolling cross validation procedure is desired.

Usage

```
tsPartition(x, rolling, windowSize, maxHorizon)
```

Arguments

x	A time series
rolling	Should indices be generated for a rolling or non-rolling procedure?
windowSize	Size of window for training
maxHorizon	Maximum forecast horizon

Value

List containing train and test indices for each fold

Author(s)

Ganesh Krishnan

Examples

```
tsPartition(AirPassengers, rolling = TRUE, windowSize = 10, maxHorizon = 2)
```

tsSubsetWithIndices *Subset time series with provided indices*

Description

Use provided indices to subset a time series. The provided indices must be contiguous

Usage

```
tsSubsetWithIndices(x, indices)
```

Arguments

x	A time series object
indices	A contiguous vector of indices to use for subsetting

Value

A time series object appropiately subsetted using provided indices

Author(s)

Ganesh Krishnan

Examples

```
tsSubsetWithIndices(AirPassengers, c(3:10))
```

---

unwrapParallelModels *Helper function used to unpack the fitted model objects from a list*

Description

Helper function used to unpack the fitted model objects from a list

Usage

```
unwrapParallelModels(fitModels, expandedModels)
```

Arguments

fitModels	A list containing the models to include in the ensemble
expandedModels	A character vector from the models argument of hybridModel

Details

See usage inside the hybridModel function.

See Also

hybridModel

Index