Package ‘forecastLSW’

April 25, 2023

Type Package

Title Forecasting Routines for Locally Stationary Wavelet Processes

Version 1.0

Date 2023-04-24

Author Rebecca Killick [aut, cre],
Matt Nunes [aut],
Guy Nason [aut],
Marina Knight [aut],
Idris Eckley [ctb]

Maintainer Rebecca Killick <r.killick@lancs.ac.uk>

Description Implementation to perform forecasting of locally stationary wavelet processes by examining the local second order structure of the time series.

Depends R(>= 3.5.0), stats, locits, wavethresh, parallel, lpacf, methods

Imports forecast

License GPL-2

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2023-04-25 17:30:02 UTC

R topics documented:

`forecastLSW-package` *Forecasting for locally stationary (wavelet) time series based on the local partial autocorrelation function.*

Description

This package computes forecasts for a time series with prediction errors. The forecasting methodology is designed with an underlying locally stationary wavelet model in mind. However, it is possible that the forecasting methodology will work well for other time series, including those where an underlying model is not necessarily known. Note: the methodology can work with any length of time series. The package also contains functions to display the forecasts and their prediction intervals or a fan chart, a function to evaluate the performance of the new forecasting methods and compare it to Box-Jenkins ARMA-based forecasting and a routine to identify wavelets that enable the forecasting routines to perform well.

Details

Package:	lpacf
Type:	Package
Version:	1.0
Date:	2023-04-24
License:	GPL-2

The `forecastlpacf` function computes forecasts of a locally stationary (wavelet) time series using the localized partial autocorrelation to help with history identification. The results of such forecasting can be printed using `print.forecastlpacf` or plotted with `plot.forecastlpacf`.

Two other useful functions are `testforecast` which runs some testing on forecasting some end values of a series using earlier values and compares the new forecasting with standard Box-Jenkins ARMA forecasting (visualisation via `forecastpanel`) and `which.wavelet.best` which attempts to identify which wavelet is well-suited to forecasting a particular series.

Author(s)

Rebecca Killick, Marina Knight, Guy Nason, Matt Nunes

Maintainer: Rebecca Killick <r.killick@lancs.ac.uk>

References

Killick, R., Knight, M.I., Nason, G.P., Nunes M.A., Eckley I.A. (2023) Automatic Locally Stationary Time Series Forecasting with application to predicting U.K. Gross Value Added Time Series under sudden shocks caused by the COVID pandemic arXiv:2303.07772

See Also

forecastlpacf, testforecast, which.wavelet.best

Examples

```
#
# See examples in each of the functions' help pages linked above.
#
```

---

abmld2 *Gross Value Added (GVA, Average) at basis prices: CP SA time series / second differenced series*

Description

Essentially GVA is a component in the estimator for UK Gross Domestic Product (GDP) an important economic time series. The series can be downloaded from the UK Office of National Statistics website, see below for references.

Usage

```
data("abml")
data("abmld2")
```

Format

The GVA series that we obtain are the quarterly reports from Q1 1955 until Q4 2020. This is a series of 264 observations. The series has a strong mean trend which we have removed using twice differencing (diff(abml, diff=2)) to obtain the series abmld2. This vector is of length 262.

Source

www.statistics.gov.uk/statbase/TSDtables1.asp and www.statistics.gov.uk/cci/nugget.asp?id=254 analyze.abmld2 *Analyzes the abmld2 data, see below for more details.*

Description

Takes the abmld2 data and analyzes it.

Usage

```
analyze.abmld2(h=10,atTime=NULL,atLag=NULL)
```

Arguments

h	Numeric value for a 1:h-steps ahead forecast. In reality we treat the `data[1:(length(data)-h)]` as known and try to forecast h-steps ahead from `data[length(data)-h]`
atTime	Vector of the times (rows) of the `lpacf` to be plotted. Note that not all times can be plotted, the range of plausible values depends on the bandwidth selected for the data. At the time of writing binwidth for abmld2 is 147 and thus the plausible values are [74,147].
atLag	Vector of the lags (columns) of the `lpacf` to be plotted. The default maximum lag is `floor(10 * log10(n))` which is 23 for abmld2.

Details

Takes the abmld2 data and analyzes it. Specifically the following is produced:

- time series plot of the abmld2 data
- the lpacf for the abmld2 data
- plots of the lpacf + CI for the specified times and lags
- the forecast for h to last data point(s) using the lpacf method
- the forecast for h to last data point(s) using the standard ARMA method
- plot of the original data, forecasts and confidence intervals for both methods, red=lpacf, blue=ARMA.

Value

List containing the lpacf, forecast + accuracy measures using the lpacf method and forecast +accuracy measures using the ARMA method.

Author(s)

Rebecca Killick

References

Killick, R., Knight, M.I., Nason, G.P., Nunes M.A., Eckley I.A. (2023) Automatic Locally Stationary Time Series Forecasting with application to predicting U.K. Gross Value Added Time Series under sudden shocks caused by the COVID pandemic arXiv:2303.07772

See Also

`lpacf.plot`, `forecastlpacf`

Examples

```
## Not run:
data(abmld2)
out=analyze.abmld2()

## End(Not run)
```

`analyze.windanomaly` *Analyzes the windanomaly data, see below for more details.*

Description

Takes the windanomaly data and analyzes it.

Usage

```
analyze.windanomaly(h=10,atTime=NULL,atLag=NULL)
```

Arguments

`h`	Numeric vector for a h-steps ahead forecast. In reality we treat the `data[1:(length(data)-h)]` as known and try to forecast h-steps ahead from `data[length(data)-h]`
`atTime`	Vector of the times (rows) of the `lpacf` to be plotted. Note that not all times can be plotted, the range of plausible values depends on the bandwidth selected for the data. At the time of writing binwidth for windanomaly is 1173 and thus the plausible values are [587,680].
`atLag`	Vector of the lags (columns) of the `lpacf` to be plotted. The default maximum lag is `floor(10 * log10(n))` which is 31 for windanomaly.

Details

Takes the windanomaly data and analyzes it. Specifically the following is produced:

- time series plot of the windanomaly data
- the lpacf for the windanomaly data
- plots of the lpacf + CI for the specified times and lags

- the forecast for h to last data point(s) using the lpacf method
- the forecast for h to last data point(s) using the standard ARMA method
- plot of the original data, forecasts and confidence intervals for both methods, red=lpacf, blue=ARMA.

Value

List containing the lpacf, forecast + accuracy measures using the lpacf method and forecast +accuracy measures using the ARMA method.

Author(s)

Rebecca Killick

References

Killick, R., Knight, M.I., Nason, G.P., Nunes M.A., Eckley I.A. (2023) Automatic Locally Stationary Time Series Forecasting with application to predicting U.K. Gross Value Added Time Series under sudden shocks caused by the COVID pandemic arXiv:2303.07772

See Also

`lpacf.plot`, `forecastlpacf`

Examples

```
## Not run:
data(windanomaly)
out=analyze.windanomaly()

## End(Not run)
```

`forecastlpacf` *Forecasts future values of the time series* `x` `h`*-steps ahead. (for the specified horizon* `h`*) using the lpacf to decide the dimension of the generalized Yule-Walker equations.*

Description

This function forecasts a `x` time series `h`-steps ahead. The time series is assumed to be locally stationary (actualy locally stationary wavelet) and uses a local prediction method. The function makes use of the localized partial autocorrelation function to decide the order of the local Yule-Walker equations used in the forecast.

Usage

```
forecastlpacf(x,h=1,regularize=TRUE,lag.max=max(10,2*h),forecast.type=NULL,...)
```

Arguments

`x`	Vector containing time series to generate forecasts for.
`h`	Integer. Maximum prediction horizon. Forecasts will be given for one to `h` time steps ahead. Currently, for `dforecastlpacf` `h` is hard-coded to be 1. If you want to forecast further ahead for differenced data then you will have to difference the time series manually and supply it to `forecastlpacf`.
`regularize`	Logical. If `regularize=TRUE` then the Yule-Walker matrix is regularized before prediction using the method from Xie et al. (2007). If `regularize=FALSE` then no regularization takes place.
`lag.max`	Maximum lag that the `lpacf` is calculated to. If this is set too low, i.e. the automated estimation of the dimension of the Yule-Walker matrix is equal to `max.lag`, then the function will print a warning message.
`forecast.type`	Options are `fixed`, `recursive` or `extend`, see details for further information.
`...`	Other parameters to be passed to the periodogram and lacv (local autocovariance) estimation, e.g. `filter.number` and `family` detailing the wavelet to be used.

Details

The function calculates the wavelet periodogram followed by the lacv and `lpacf`. NOTE: Often when local (windowed) estimates are created one assigns the estimated value to the central point in the window. This is NOT the approach we take here when calculating the lacv and `lpacf`. Instead we operate a rear facing window where the estimate is assigned to the final point in the window.

The lpacf is used to decide the dimension of the local Yule-Walker equations used for forecasting. The periodogram is then smoothed using a running mean smoother, and then to get forecast lacv estimates. The Yule-Walker equations give the forecast mean for `h` steps ahead. The standard deviation of the forecasts is also returned.

When we are trying to forecast `h` steps ahead we use the lpacf to decide how many values (`p`) we should use for prediction. The original method of Fryzlewicz et al. (2003) decides on `p` and then does a `h` step ahead forecast only using the `p` last values. This is what `forecast.type='fixed'` does, regardless of the size of `p` in relation to `h`. Note that the left hand side of the Yule-Walker matrix is fixed and only the right hand size (the forecast lacv) is changing. Thus the size of `h` is not explicitly taken into account, there is just an inflated variance in the lacv estimate. One other option is to use the intermediate forecast values as if they were observed and perform a recursive forecast - this is what `forecast.type='recursive'` does. Here everything in the Yule-Walker equations is different for each forecast value.

A third option is to use `forecast.type='fixed'` when `p` is greater or equal to `h` but then when we are trying to forecast beyond this we extend the Yule-Walker equations to be the same dimension as the forecast horizon. Thus using `h` previous values instead of `p`. This is what `forecast.type='extend'` does.

The method closest to the stationary world is `forecast.type='recursive'`.

The `dforecastlpacf` internally differences the time series and then performs the local forecasting as in `forecastlpacf` but only for one-step ahead. The advantage is that subsequent plotting routines can nicely show the original time series, with the forecasts on the original (not differenced) scale with the forecast and appropriate confidence interval.

Value

An object of class `forecastlpacf` which is a list with the following components.

- `mean` — Returns time series forecasts from one to h-steps ahead. When h is greater than one multiple predictions are returned in this vector. In this case, item in position n corresponds to n steps ahead. For example, if h=2 then this vector will contain two elements. The first one corresponds to the prediction one-step-head and the second entry to the two-steps-ahead prediction.
- `std.err` — Returns the prediction error, which can be used for assessing the prediction intervals. Item n corresponds to the prediction n-steps ahead, as for the mean component.
- `lpacf` — Returns the estimated local partial autocovariance function
- `ci` — The confidence interval on lpacf which was used used for the automatic calculation of p
- `binwidth` — The automatic bandwidth used for the running mean smoother
- `p` — Returns the automatic choice of p - the dimension of the generalized Yule-Walker equations.
- `x` — The supplied original time series
- `d` — Differencing that was applied to the input series before forecasting. For `forecastlpacf` this is d=0. For differencing once see the function `dforecastlpacf` which returns d=1.

Author(s)

R. Killick

References

Killick, R., Knight, M.I., Nason, G.P., Nunes M.A., Eckley I.A. (2023) Automatic Locally Stationary Time Series Forecasting with application to predicting U.K. Gross Value Added Time Series under sudden shocks caused by the COVID pandemic arXiv:2303.07772

Fryzlewicz, P., Van Bellegem, S. and von Sachs, R. (2003) Forecasting non-stationary time series by wavelet process modelling. *Annals of the Institute of Statistical Mathematics*, **55**, 737-764.

Nason, G.P., von Sachs, R., Kroisandt, G. (2000) Wavelet processes and adaptive estimation of the evolutionary wavelet spectrum. *J. Roy. Statist. Soc. B*, **62**, 271-292.

Xi, Y., Yu, J., Ranneby, B. (2007) Forecasting Using Locally Stationary Wavelet Processes.

See Also

`lpacf`, `forecastpanel`, `plot.forecastlpacf`, `print.forecastlpacf`, `summary.forecastlpacf`

Examples

```
# first generate some non-stationary data we want to forecast
set.seed(1)
x=tvar2sim()
```

```
#predict 1-step ahead using Daubechies wavelets with 2 vanishing moments, although
#other choices for the wavelet family and filter are possible (including Haar)
pred<-forecastlpacf(x,h=1,filter.number=2,family="DaubExPhase",forecast.type='recursive')

#pred$mean gives the predicted value, while pred$std.err gives the prediction error
```

`forecastpanel` *Function to produce a plot of data forecasts.*

Description

This function produces a plot of the data forecast with confidence intervals (if supplied) and, if supplied, against the truth. Optionally, summaries of the forecast fit are returned.

Usage

```
forecastpanel(forecastobj,truth=NULL,add=FALSE,summary=TRUE,test="all",move=0,
conf.level=95,col="red",pch=c(17,19,95),...)
```

Arguments

`forecastobj`	Either an object of class `forecast`, `forecastlpacf` or a vector of forecasts.
`truth`	The true values of the signal that has been forecast.
`add`	If FALSE a new plot is created, otherwise points are added to the active graphics device.
`summary`	If TRUE a summary of the forecast fit is supplied, see `accuracy`.
`test`	Argument supplied to `accuracy` to determine which summary measures are returned.
`move`	If move does not equal 0 then this is the amount to move the points+confidence intervals for the forecasts to the left (if negative) and to the right (if positive) to offset the plotted location (0) to potentially make the graphic clearer.
`conf.level`	Confidence level used for the `forecastobj`. If `forecastobj` is lpacf it can be calculated for any confidence level. If `forecastobj` is of class forecast then the level needs to match the one given when the forecast was calculated. A number between 1 and 100.
`col`	Specifies the colour of forecasts on the plot, see `par` for details.
`pch`	Length 3 vector specifying the plotting character (pch) of the truth, forecast and CI in that order.
`...`	Additional arguments can be supplied which will be passed to `plot`, `points` and `segments`.

Details

Plots the forecast data, confidence intervals and true signal if supplied. If `summary=TRUE` then the output of `accuracy` is returned.

Value

If summary=TRUE then the output of accuracy is returned.

Author(s)

Rebecca Killick

See Also

forecastlpacf,accuracy

Examples

```
# first generate a time-varying process
x=tvar2sim()

# forecast the last 12 data points using the lpacf
ans<-forecastlpacf(x[1:500],h=12,forecast.type='recursive')

# then plot it and get summaries to see how we did
## Not run: plot(ans,truth=x[501:512],move=0.05)
```

---

fp.forecast *Do automatic Box-Jenkins ARIMA fit and forecast.*

Description

This function merely wraps some excellent functions from the forecast package up and returns the forecast values and their lower and upper prediction intervals.

Usage

```
fp.forecast(x, h = 1, conf.level = 95)
```

Arguments

x	The time series you wish to forecast.
h	The number of steps ahead (forecast horizon)
conf.level	The confidence level for the forecast prediction interval expressed as a value between 0 and 100.

Details

This function entirely relies on existing functions from the forecast package. It applies auto.arima to x to fit an ARIMA model to the series with an automatic choice of parameters. Then the forecast function is applied to the ARIMA object to obtain forecasts and prediction intervals.

Value

A matrix with h rows and three columns. The first column contains the forecasted values. The second and third columns contrain the lower and upper prediction intervals.

Author(s)

G.P. Nason

References

Killick, R., Knight, M.I., Nason, G.P., Nunes M.A., Eckley I.A. (2023) Automatic Locally Stationary Time Series Forecasting with application to predicting U.K. Gross Value Added Time Series under sudden shocks caused by the COVID pandemic arXiv:2303.07772

Hyndman, R.J. and Khandakar, Y. (2008) Automatic Time Series Forecasting: The forecast package for R. *Journal of Statistical Software*, **27**, Issue 3.

Examples

```
#
# Generate random test series
#
x.test <- tvar2sim()
#
# Produce stationary Box-Jenkins forecasts and prediction intervals for
# two-steps ahead
#
fp.forecast(x.test, h=2)
```

---

plot — *Plot the results of forecasting using* forecastlpacf

Description

The forecastlpacf performs forecasting on a locally stationary (wavelet) time series. This function provides several options to plot the results in a user-friendly fashion.

Usage

```
## S3 method for class 'forecastlpacf'
plot(x, extra.y = NULL, f.col = 4, show.pi = "standard",
    pi.col = 2, xlab = "Time", ylab = "Time Series", zoom = FALSE, zoom.no = 30,
    sw = 0.2, conf.level = 95, pc.fan = (1:9) * 10, fan.seps = FALSE,
    fan.rgb.col=c(1,0,0), ...)
```

Arguments

`x`	The object returned by the `forecastlpacf` function.
`extra.y`	Sometimes other routines wish to add to the plot generated by this function. The y-axis extent of those extra values might be larger than the values that this plot alone would generate. So, you can use this argument to provide a set of y-values that you want to later plot and this plot takes those into account when setting the scale of the y-axis. So, if you have extra characters or lines to plot after this plot, and you want to ensure they'll get plotted and that the y-axis is going to be large enough, supply the y values as a vector (or just their maximum and minimum) and this function will use them to help set the y-axis scale.
`f.col`	The colour used to drae the forecasted values - both the points and line joining the forecasts.
`show.pi`	If set to "standard" then 100*conf.level percent prediction intervals are drawn for each forecasted point in the colour specified by `pi.col`. If set to "none" then no prediction intervals are drawn. If set to "fan" then a Bank-of-England-like fan-plot is produced with confidence levels set by the `pc.fan` argument.
`pi.col`	Colour of the prediction intervals or fan plot.
`xlab`	The x-axis label.
`ylab`	The y-axis label.
`zoom`	Sometimes for a long time series with a few forecasts the forecast values can be hard to see and particularly how they relate to the values of the series near to the end of the series. If `TRUE` then this argument causes the function to only plot the last `zoom.no` values of the time series and the associated forecasts. One can then focus on the end of the time series nearer to the forecast values and those values. If `FALSE` then the whole time series and the forecasts are plotted and `zoom.no` is ignored.
`zoom.no`	The number of time series values plotted if `zoom=TRUE`.
`sw`	The width of the prediction intervals if `show.pi="standard"`.
`conf.level`	A single confidence value associated with the prediction interval expressed as a numerical value from 0-100.
`pc.fan`	A vector of confidence values associated with the fan plot prediction intervals expressed as a percentage.
`fan.seps`	If `TRUE` then lines are drawn on the fan part of the fan plot to more clearly indicated the distinction between different prediction intervals. If `FALSE` then no extra lines are drawn.
`fan.rgb.col`	A vector of length three containing the red, green and blue intensities of the fan plot colour
`...`	Other arguments to plot.

Details

This function produces a plot of a time series and its forecasts generated by the `forecastlpacf` function.

Value

The function only returns information if show.pi="fan". In this case an array is returned that contained the coordinates of the fan part of the plot. The array is three-dimensional. Dimension 1 corresponds to the number of steps ahead that we computed for the forecast in the object x, dimension 2 corresponds to the number of fan prediction intervals specified by the number of confidence bands in pc.fan, dimension 3 always has two dimensions: 1 corresponding to the upper prediction interval and 2 correspond to the lower interval. For example, element[2, 3, 1] corresponds to the upper prediction interval, for the fan component associated with the third fan confidence level value in pc.fan for the h=2 step ahead forecast.

Author(s)

Guy Nason

References

Killick, R., Knight, M.I., Nason, G.P., Nunes M.A., Eckley I.A. (2023) Automatic Locally Stationary Time Series Forecasting with application to predicting U.K. Gross Value Added Time Series under sudden shocks caused by the COVID pandemic arXiv:2303.07772

See Also

forecastlpacf

Examples

```
#
# Simulate an example
#
x.test <- tvar2sim()
#
# Do a two-step ahead forecast
#
x.fl <- forecastlpacf(x.test, h=2, forecast.type="recursive")
#
# Now plot it.
#
# zoom=TRUE: so we only plot the last 30 time series observations, by default
#  change zoom.no if you want more or less.
# f.col=3: the forecasts and connecting lines are drawn in colour 3 (blue)
# show.pi="fan": do a fan chart for the forecasts
# fan.rgb.col=c(1,0,1): draw the fan in magenta (default is red)
# ylab="My Time Series": change the y label to something nice
#
plot(x.fl,zoom=TRUE, f.col=3, show.pi="fan", fan.rgb.col=c(1,0,1), ylab="My Time Series")
```

print — *Prints a* `forecastlpacf` *object*

Description

Prints a `forecastlpacf` object, basically telling you what's there.

Usage

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

Arguments

`x`	The `forecastlpacf` object
`...`	Other arguments (not used)

Details

Prints a `forecastlpacf` object, basically telling you what's there.

Value

None.

Author(s)

Guy Nason

References

Killick, R., Knight, M.I., Nason, G.P., Nunes M.A., Eckley I.A. (2023) Automatic Locally Stationary Time Series Forecasting with application to predicting U.K. Gross Value Added Time Series under sudden shocks caused by the COVID pandemic arXiv:2303.07772

See Also

`forecastlpacf`, `summary.forecastlpacf`

Examples

```
#
# Simulate an example
#
x.test <- tvar2sim()
#
# Do a two-step ahead forecast
#
x.fl <- forecastlpacf(x.test, h=2, forecast.type="recursive")
```

```
#
# Print out the object
#
print(x.fl)
#
# This is what gets output
#
#Class 'forecastlpacf' : Forecast from Locally Stationary Time Series:
#       ~~~~  : List with 8 components with names
#              mean std.err lpacf ci binwidth p x d
#
#
#summary(.):
#----------
#Number of steps ahead predicted:  2
#Predictions are (3dp):  1.52 -0.365
#Std err are (3dp):  0.952 0.955
#Smoothing binwidth was:  293
#Forecast was based on a p-backlag value selected as:  3
#There was no explicit differencing.
```

---

summary *Print out summary information about a* `forecastlpacf` *object*

Description

Print out summary information about a `forecastlpacf` object.

Usage

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

Arguments

`object`	The object you want to print out summary info for.
`...`	Other arguments

Details

Prints out the maximum number of steps ahead considered in the object, prints out the first few predictions (up to 6), and their standard errors. The smoothing binwidth associated with the localized partial autocorrelation object used to compute the predictions is printed. The order, p, of the localized partial autocorrelation is printed. A note of whether differencing was actioned is printed.

Value

None

Author(s)

Guy Nason

References

Killick, R., Knight, M.I., Nason, G.P., Nunes M.A., Eckley I.A. (2023) Automatic Locally Stationary Time Series Forecasting with application to predicting U.K. Gross Value Added Time Series under sudden shocks caused by the COVID pandemic arXiv:2303.07772

See Also

forecastlpacf, print.forecastlpacf

Examples

```
#
# Example for print.forecastlpacf contains a call to summary.forecastlpacf
```

`testforecast` *Compare locally stationary forecasting with Box-Jenkins-type forecasting, by predicting the final values of a time series.*

Description

A good way of evaluating a forecasting method is to apply the method to most of a series (apart from the last few values) to forecast those last few values. Then, the forecasts and the true values can be compared to see how good the forecast is. This function performs this for the locally stationary forecasting based on wavelet processes in forecastlpacf and a version of the Box-Jenkins forecasting, and also produces both plots and returns results of the testing.

Usage

```
testforecast(x, n.to.test, go.back=0,  plot.it = TRUE, regularize = TRUE,
    lag.max = max(10, 2 * n.to.test), truth.pch = 23, truth.col = 3, zoom = TRUE,
   zoom.no = 30, forecast.type = NULL, conf.level = 0.95, stycol = 6, silent = TRUE,
  lapplyfn=lapply, ...)
```

Arguments

`x`	The time series you want to use in testing.
`n.to.test`	Suppose the length of `x` is T. This function uses the first T-`n.to.test` observations to predict the last `n.to.test` observations.
`go.back`	If `go.back=0` then a single forecasting operation forecasting the last n.to.test observations from the previous data is conducted. If `go.back` is an integer greater than zero then the same forecasting as with `go.back` occurs but each time the end of the series is moved back one point. This shifting back occurs from one

	shift to `go.back` shifts. The purpose of this is to repeat the exercise for using previous data to forecast `n.to.test` points at the end of the series, but to then repeat this for the series one step earlier, then two steps earlier, ..., back to `go.back` steps earlier. The results of each forecast are combined into an overall root-mean-squared error result for each forecast horizon (there will be `n.to.test` values) for both of the Box-Jenkins and the new forecast methodology provided by `forecastlpacf`. These additionally forecasts will be computed in parallel if the `parallel` package is loaded and `mclapply` is used as an argument to `lapplyfn`.
`plot.it`	If `TRUE` a plot is produced showing the original time series, the stationary and locally stationary forecasts, and their prediction intervals. If `FALSE` then no plot is produced.
`regularize`	Passed through to `forecastlpacf`
`lag.max`	Passed through to `forecastlpacf`
`truth.pch`	The type of plotting character used for the true values, see `pch` argument to `points` function in R.
`truth.col`	Colour of plot symbol used for true values.
`zoom`	Typically, we're interested in the later values of a time series when doing forecasting. If this argument is `TRUE` then only the last `zoom.no` observations are plotted, so one can focus on the end of the series.
`zoom.no`	If `zoom=TRUE` then this argument controls how much of the end of the series is plotted.
`forecast.type`	Passed through to `forecastlpacf`
`conf.level`	Controls the width of the prediction intervals for both stationary and nonstationary forecasting.
`stycol`	The colour of both the stationary forecasts and their confidence intervals.
`silent`	If `TRUE` then nothing gets printed, otherwise messages get printed.
`lapplyfn`	For single-processor use this argument should be `lapply` (the default). However, you can set the argument to `mclapply` if you have the `parallel` package loaded. Remember to set the number of processors you want to use with the `mc.cores` option, e.g. `options(mc.cores=4)` if you had four cores available.
`...`	Other arguments to the `forecastlpacf` call

Details

Suppose `n.to.test=1`. Then this function uses all the values of the time series `x` apart from the last to generate two forecasts of the last value. The two methods used to forecast are the locally stationary method `forecastlpacf` and a Box-Jenkins ARIMA alternative for stationary series coded in `fp.forecast`.

Then, if `plot.it=TRUE` a plot of the time series `x` is produced, overlaid with both types of forecast and their related prediction intervals (the locally stationary ones are hached thin rectangles, the stationary ones indicated by vertical < > symbols. The true value is also indicated by a character whose visual characteristics are controlled by the `truth.pch` and `truth.col` arguments, but by default are a green diamond.

If n.to.test is bigger than 1 then all of the data, apart from the last n.to.test values are used in constructing the forecasts (both stationary and locally stationary) for the last n.to.test values.

Values of the empirical root mean squared error of the two forecast methods are printed out (unless silent=TRUE). The predictions and their standard errors for the n.to.test values are printed out.

Value

If go.back=0 a matrix with n.to.test values with four columns is returned. The first column is the actual true value of the time series in the last n.to.test positions. The second and fourth columns are the forecast values from the locally stationary and stationary methods. The third column are the locally stationary prediction error values.

If go.back is a positive integer then a data frame with two columns. The first column corresponds to stationary forecasting using the standard Box-Jenkins type method encapsulated by fp.forecast. The second column corresponds to the locally stationary forecasting encapsulated by forecastlpacf. Each row of the frame corresponds to a different forecasting horizon, the horizon is indicated by the row name of the data frame.

Author(s)

G.P. Nason

References

Killick, R., Knight, M.I., Nason, G.P., Nunes M.A., Eckley I.A. (2023) Automatic Locally Stationary Time Series Forecasting with application to predicting U.K. Gross Value Added Time Series under sudden shocks caused by the COVID pandemic arXiv:2303.07772

See Also

forecastlpacf, fp.forecast, plot.forecastlpacf

Examples

```
#
# Generate simulated time series from TVAR(2) model.
#
x.test <- tvar2sim()
#
# Now run testforecast on this example time series.
# We've only supplied plot.it=FALSE because its in an R help page, normally
# you would set plot.it=TRUE, which is the default, because you want to see
# the plot.
#
tmp <- testforecast(x.test, n.to.test=3, forecast.type="recursive",
plot.it=FALSE)
```

which.wavelet.best *Find out what wavelet is good for forecasting your series.*

Description

A big question with many wavelet methods is which wavelet should one use for a particular task. This function tries some forecasting on your time series with all Daubechies compactly supported wavelets available to it and returns a list of the forecasting performance for each choice, and indicates which wavelet gave the best results. This wavelet can then be used in future forecasting.

Usage

```
which.wavelet.best(x, n.to.test = 10, go.back=5,
forecast.type = "recursive", lapplyfn = lapply)
```

Arguments

x	Your times series (not necessarily of dyadic length!)
n.to.test	How many observations at the end to test as part of the assessment process. The default, 10, means that 10 observations at the end of the series will all be forecast. This number should be reasonably big to enable forecasts of more than a few data points, but not too large.
go.back	Controls the `go.back` argument to `testforecast`. Number of repeats of the procedure on successively one-unit of time earlier series.
forecast.type	The type of forecasting as detailed in `forecastlpacf`.
lapplyfn	Type of list processing function. By default it uses R's `lapply` function. However, if you use the `parallel` library you can replace this with `mclapply` which will make this function go faster using parallel computation. Don't forget to set the `options(mc.cores=4)` argument to what you wish (here it is set to 4 in this example, but you should set it to something that is appropriate for your machine environment).

Details

This function uses all choices of wavelet to forecast the last `n.to.test` observations of your time series. It works out the forecasting error in doing so for each choice of wavelet and returns a list telling you which wavelet did best.

Value

A data frame containing information on the root mean squared forecasting error performance of the locally stationary forecasting method for different wavelets. The data frame has four columns and a row for each wavelet tried. The first and second column give the filter number and family for each wavelet. The third column gives the root mean squared error for each combination of wavelet. The fourth column contains an indicator that shows which wavelet was best (there might be more than one).

Author(s)

Guy Nason

References

Killick, R., Knight, M.I., Nason, G.P., Nunes M.A., Eckley I.A. (2023) Automatic Locally Stationary Time Series Forecasting with application to predicting U.K. Gross Value Added Time Series under sudden shocks caused by the COVID pandemic arXiv:2303.07772

See Also

testforecast

Examples

```
#
# Generate simulated example
#
x <- tvar2sim()
#
# Work out which wavelet is best for forecasting this series
#
# Note: to speed up I also do:
# library("parallel")
# options(mc.cores=4) # You have a four core machine, eg
# tmp <- which.wavelet.best(x, lapplyfn=mclapply)
#
# Note2: The following command can take a few minutes to run, even on
# a fairly recent (2013) machine. You can speed it up by using
# parallel execution as noted above, or by reducing go.back or
# by reducing n.to.test, and also shortening the time series x to
# more recent values. However, you need to be careful if you shorten
# x too much then you are not basing the best wavelet decision on the
# right time series. Similarly, by reducing go.back you are not
# insuring your answer across runs across many internal forecasts.
#
## Not run: tmp <- which.wavelet.best(x)
#
# Print out what the result was:
#
## Not run: print(tmp)
#   filter.number      family       mse    min.mse
#1              1 DaubExPhase 0.2139173 <- Min MSE
#2              2 DaubExPhase 0.5040532
#3              3 DaubExPhase 0.4064091
#4              4 DaubExPhase 0.3077695
#5              5 DaubExPhase 0.3706422
#6              6 DaubExPhase 0.6617254
#7              7 DaubExPhase 0.5477581
#8              8 DaubExPhase 0.6881407
#9              9 DaubExPhase 0.5514298
#10            10 DaubExPhase 0.5551846
```

```
#11            4 DaubLeAsymm 0.3134285
#12            5 DaubLeAsymm 0.3910101
#13            6 DaubLeAsymm 0.7480980
#14            7 DaubLeAsymm 0.5700830
#15            8 DaubLeAsymm 0.5661297
#16            9 DaubLeAsymm 0.5689345
#17           10 DaubLeAsymm 0.5580267
```

`windanomaly` *Eq. Pacific meridional wind anomaly index, Jan 1900 - June 2005*

Description

This dataset gives the monthly ENSO meridional wind anomaly index for the region 12-2N, 160E-80W from January 1900 until June 2005.

Usage

```
data("windanomaly")
```

Format

A vector of wind anomalies with length 1266.

Source

http://jisao.washington.edu/data_sets/eqpacmeridwindts/

Index