Package: aTSA (via r-universe)

August 20, 2024

Type Package

Title Alternative Time Series Analysis

Version 3.1.2.1

Date 2015-06-19

Author Debin Qiu

Maintainer Debin Qiu <debinqiu@uga.edu>

Description Contains some tools for testing, analyzing time series data and fitting popular time series models such as ARIMA, Moving Average and Holt Winters, etc. Most functions also provide nice and clear outputs like SAS does, such as identify, estimate and forecast, which are the same statements in PROC ARIMA in SAS.

License GPL-2 | GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2024-02-21 07:59:33 UTC

Contents

`accurate` *Accurate Computation*

Description

Computes the accurate criterion of smoothed (fitted) values.

Usage

```
accurate(x, x.hat, k, output = TRUE)
```

Arguments

`x`	a numeric vector of original values.
`x.hat`	a numeric vector of smoothed (fitted) values.
`k`	the number of parameters in obtaining the smoothed (fitted) values.
`output`	a logical value indicating to print the results in R console. The default is `TRUE`.

Details

See `http://www.dms.umontreal.ca/~duchesne/chap12.pdf` in page 616 - 617 for the details of calculations for each criterion.

Value

A vector containing the following components:

`SST`	the total sum of squares.
`SSE`	the sum of the squared residuals.
`MSE`	the mean squared error.
`RMSE`	the root mean square error.
`MAPE`	the mean absolute percent error.
`MPE`	the mean percent error.
`MAE`	the mean absolute error.
`ME`	the mean error.
`R.squared`	R^2 = 1 - SSE/SST.

R.adj.squared	the adjusted R^2.
RW.R.squared	the random walk R^2.
AIC	the Akaike's information criterion.
SBC	the Schwarz's Bayesian criterion.
APC	the Amemiya's prediction criterion

Note

If the model fits the series badly, the model error sum of squares SSE may be larger than SST and the R.squared or RW.R.squared statistics will be negative. The RW.R.squared uses the random walk model for the purpose of comparison.

Author(s)

Debin Qiu

Examples

```
X <- matrix(rnorm(200),100,2)
y <- 0.1*X[,1] + 2*X[,2] + rnorm(100)
y.hat <- fitted(lm(y ~ X))
accurate(y,y.hat,2)
```

---

adf.test *Augmented Dickey-Fuller Test*

Description

Performs the Augmented Dickey-Fuller test for the null hypothesis of a unit root of a univarate time series x (equivalently, x is a non-stationary time series).

Usage

```
adf.test(x, nlag = NULL, output = TRUE)
```

Arguments

x	a numeric vector or univariate time series.
nlag	the lag order with default to calculate the test statistic. See details for the default.
output	a logical value indicating to print the test results in R console. The default is TRUE.

Details

The Augmented Dickey-Fuller test incorporates three types of linear regression models. The first type (`type1`) is a linear model with no drift and linear trend with respect to time:

$$dx[t] = \rho * x[t-1] + \beta[1] * dx[t-1] + ... + \beta[nlag-1] * dx[t-nlag+1] + e[t],$$

where d is an operator of first order difference, i.e., $dx[t] = x[t] - x[t-1]$, and $e[t]$ is an error term.

The second type (`type2`) is a linear model with drift but no linear trend:

$$dx[t] = \mu + \rho * x[t-1] + \beta[1] * dx[t-1] + ... + \beta[nlag-1] * dx[t-nlag+1] + e[t].$$

The third type (`type3`) is a linear model with both drift and linear trend:

$$dx[t] = \mu + \beta * t + \rho * x[t-1] + \beta[1] * dx[t-1] + ... + \beta[nlag-1] * dx[t-nlag+1] + e[t].$$

We use the default `nlag = floor(4*(length(x)/100)^(2/9))` to calcuate the test statistic. The Augmented Dickey-Fuller test statistic is defined as

$$ADF = \rho.hat / S.E(\rho.hat),$$

where $\rho.hat$ is the coefficient estimation and $S.E(\rho.hat)$ is its corresponding estimation of standard error for each type of linear model. The p.value is calculated by interpolating the test statistics from the corresponding critical values tables (see Table 10.A.2 in Fuller (1996)) for each type of linear models with given sample size n = length(x). The Dickey-Fuller test is a special case of Augmented Dickey-Fuller test when `nlag` = 2.

Value

A list containing the following components:

`type1`	a matrix with three columns: `lag`, `ADF`, `p.value`, where `ADF` is the Augmented Dickey-Fuller test statistic.
`type2`	same as above for the second type of linear model.
`type3`	same as above for the third type of linear model.

Note

Missing values are removed.

Author(s)

Debin Qiu

References

Fuller, W. A. (1996). Introduction to Statistical Time Series, second ed., New York: John Wiley and Sons.

See Also

`pp.test`, `kpss.test`, `stationary.test`

Examples

```
# ADF test for AR(1) process
x <- arima.sim(list(order = c(1,0,0),ar = 0.2),n = 100)
adf.test(x)
# ADF test for co2 data
adf.test(co2)
```

`arch.test` *ARCH Engle's Test for Residual Heteroscedasticity*

Description

Performs Portmanteau Q and Lagrange Multiplier tests for the null hypothesis that the residuals of a ARIMA model are homoscedastic.

Usage

```
arch.test(object, output = TRUE)
```

Arguments

`object`	an object from arima model estimated by `arima` or `estimate` function.
`output`	a logical value indicating to print the results in R console, including the plots. The default is `TRUE`.

Details

The ARCH Engle's test is constructed based on the fact that if the residuals (defined as $e[t]$) are heteroscedastic, the squared residuals ($e^2[t]$) are autocorrelated. The first type of test is to examine whether the squares of residuals are a sequence of white noise, which is called Portmanteau Q test and similar to the Ljung-Box test on the squared residuals. The second type of test proposed by Engle (1982) is the Lagrange Multiplier test which is to fit a linear regression model for the squared residuals and examine whether the fitted model is significant. So the null hypothesis is that the squared residuals are a sequence of white noise, namely, the residuals are homoscedastic. The lag parameter to calculate the test statistics is taken from an integer sequence of $1 : min(24, n)$ with step 4 if $n > 25$, otherwise 2, where n is the number of nonmissing observations.

The plots of residuals, squared residuals, p.values of PQ and LM tests will be drawn if `output = TRUE`.

Value

A matrix with the following five columns:

`order`	the lag parameter to calculate the test statistics.
`PQ`	the Portmanteau Q test statistic.
`p.value`	the p.value for PQ test.
`LM`	the Lagrange Multiplier test statistic.
`p.value`	the p.value for LM test.

Note

Missing values are removed before analysis.

Author(s)

Debin Qiu

References

Engle, Robert F. (1982). Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50 (4): 987-1007.

McLeod, A. I. and W. K. Li. Diagnostic Checking ARMA Time Series Models Using Squared-Residual Autocorrelations. *Journal of Time Series Analysis*. Vol. 4, 1983, pp. 269-273.

Examples

```
x <- rnorm(100)
mod <- estimate(x,p = 1) # or mod <- arima(x,order = c(1,0,0))
arch.test(mod)
```

---

aTSA *Alternative Time Series Analysis*

Description

This is an alternative package to analyze the time series data, especially the univariate time series. Compared with other existing functions for time series analysis, most functions in this package provide nice outputs like SAS does for time series. Several functions are exactly the same names as 'arima' procedure in SAS, such as `identify`, `estimate`, and `forecast`, etc. They also have the similar outputs.

Details

Package:	aTSA
Type:	Package
Version:	3.1.2
Date:	2015-06-19
License:	GPL-2 | GPL-3

For a complete list of functions and dataset, use `library(help = aTSA)`.

Author(s)

Debin Qiu

Maintainer: Debin Qiu <debinqiu@uga.edu>

References

Engle, Robert F.; Granger, Clive W. J. (1987). Co-integration and error correction: Representation, estimation and testing. *Econometrica*, 55 (2): 251-276.

Engle, Robert F. (1982). Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50 (4): 987-1007.

Fuller, W. A. (1976). Introduction to Statistical Time Series. New York: John Wiley and Sons.

Hobijn B, Franses PH and Ooms M (2004). Generalization of the KPSS-test for stationarity. *Statistica Neerlandica*, vol. 58, p. 482-502.

Kwiatkowski, D.; Phillips, P. C. B.; Schmidt, P.; Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*, 54 (1-3): 159-178.

McLeod, A. I. and W. K. Li. Diagnostic Checking ARMA Time Series Models Using Squared-Residual Autocorrelations. *Journal of Time Series Analysis*. Vol. 4, 1983, pp. 269-27.

Phillips, P. C. B.; Perron, P. (1988). Testing for a Unit Root in Time Series Regression. *Biometrika*, 75 (2): 335-346.

`coint.test` *Cointegration Test*

Description

Performs Engle-Granger(or EG) tests for the null hypothesis that two or more time series, each of which is I(1), are not cointegrated.

Usage

```
coint.test(y, X, d = 0, nlag = NULL, output = TRUE)
```

Arguments

`y`	the response
`X`	the exogenous input variable of a numeric vector or a matrix.
`d`	difference operator for both `y` and `X`. The default is 0.
`nlag`	the lag order to calculate the test statistics. The default is `NULL`.
`output`	a logical value indicating to print the results in R console. The default is `TRUE`.

Details

To implement the original EG tests, one first has to fit the linear regression

$$y[t] = \mu + B * X[t] + e[t],$$

where B is the coefficient vector and $e[t]$ is an error term. With the fitted model, the residuals are obtained, i.e., $z[t] = y[t] - haty[t]$ and a Augmented Dickey-Fuller test is utilized to examine whether the sequence of residuals $z[t]$ is white noise. The null hypothesis of non-cointegration is equivalent to the null hypothesis that $z[t]$ is white noise. See `adf.test` for more details of Augmented Dickey-Fuller test, as well as the default `nlag`.

Value

A matrix for test results with three columns (`lag`, `EG`, `p.value`) and three rows (`type1`, `type2`, `type3`). Each row is the test results (including lag parameter, test statistic and p.value) for each type of linear regression models of residuals $z[t]$. See `adf.test` for more details of three types of linear models.

Author(s)

Debin Qiu

References

MacKinnon, J. G. (1991). Critical values for cointegration tests, Ch. 13 in Long-run Economic Relationships: Readings in Cointegration, eds. R. F. Engle and C. W. J. Granger, Oxford, Oxford University Press.

See Also

`adf.test`

Examples

```
X <- matrix(rnorm(200),100,2)
y <- 0.3*X[,1] + 1.2*X[,2] + rnorm(100)
# test for original y and X
coint.test(y,X)

# test for response = diff(y,differences = 1) and
# input = apply(X, diff, differences = 1)
coint.test(y,X,d = 1)
```

---

ecm *Error Correction Model*

Description

Fits an error correction model for univriate response.

Usage

```
ecm(y, X, output = TRUE)
```

Arguments

`y`	a response of a numeric vector or univariate time series.
`X`	an exogenous input of a numeric vector or a matrix for multivariate time series.
`output`	a logical value indicating to print the results in R console. The default is `TRUE`.

Details

An error correction model captures the short term relationship between the response y and the exogenous input variable X. The model is defined as

$$dy[t] = bold\beta[0] * dX[t] + \beta[1] * ECM[t-1] + e[t],$$

where d is an operator of the first order difference, i.e., $dy[t] = y[t] - y[t-1]$, and $bold\beta[0]$ is a coefficient vector with the number of elements being the number of columns of X (i.e., the number of exogenous input variables), and$ECM[t-1] = y[t-1] - haty[t-1]$ which is the main term in the sense that its coefficient $\beta[1]$ explains the short term dynamic relationship between y and X in this model, in which $haty[t]$ is estimated from the linear regression model $y[t] = bold\alpha * X[t] + u[t]$. Here, $e[t]$ and $u[t]$ are both error terms but from different linear models.

Value

An object with class "lm", which is the same results of lm for fitting linear regression.

Note

Missing values are removed before the analysis. In the results, dX or dX1, dX2, ... represents the first difference of each exogenous input variable X, and dy is the first difference of response y.

Author(s)

Debin Qiu

References

Engle, Robert F.; Granger, Clive W. J. (1987). Co-integration and error correction: Representation, estimation and testing. *Econometrica*, 55 (2): 251-276.

Examples

```
X <- matrix(rnorm(200),100,2)
y <- 0.1*X[,1] + 2*X[,2] + rnorm(100)
ecm(y,X)
```

---

estimate *Estimate an ARIMA Model*

Description

Estimates an ARIMA model for a univariate time series, including a sparse ARIMA model.

Usage

```
estimate(x, p = 0, d = 0, q = 0, PDQ = c(0, 0, 0), S = NA,
  method = c("CSS-ML", "ML", "CSS"), intercept = TRUE, output = TRUE, ...)
```

Arguments

x	a univariate time series.
p	the AR order, can be a positive integer or a vector with several positive integers. The default is 0.
d	the degree of differencing. The default is 0.
q	the MA order, can be a positive integer or a vector with several positive integers. The default is 0.
PDQ	a vector with three non-negative integers for specification of the seasonal part of the ARIMA model. The default is c(0,0,0).
S	the period of seasonal ARIMA model. The default is NA.
method	fitting method. The default is CSS-ML.
intercept	a logical value indicating to include the intercept in ARIMA model. The default is TRUE.
output	a logical value indicating to print the results in R console. The default is TRUE.
...	optional arguments to arima function.

Details

This function is similar to the ESTIMATE statement in ARIMA procedure of SAS, except that it does not fit a transfer function model for a univariate time series. The fitting method is inherited from arima in stats package. To be specific, the pure ARIMA(p,q) is defined as

$$X[t] = \mu + \phi[1] * X[t-1] + ... + \phi[p] * X[p] + e[t] - \theta[1] * e[t-1] - ... - \theta[q] * e[t-q].$$

The p and q can be a vector for fitting a sparse ARIMA model. For example, p = c(1,3),q = c(1,3) means the ARMA((1,3),(1,3)) model defined as

$$X[t] = \mu + \phi[1] * X[t-1] + \phi[3] * X[t-3] + e[t] - \theta[1] * e[t-1] - \theta[3] * e[t-3].$$

The PDQ controls the order of seasonal ARIMA model, i.e., ARIMA(p,d,q)x(P,D,Q)(S), where S is the seasonal period. Note that the difference operators d and D = PDQ[2] are different. The d is equivalent to diff(x,differences = d) and D is diff(x,lag = D,differences = S), where the default seasonal period is S = frequency(x).

The residual diagnostics plots will be drawn.

Value

A list with class "estimate" and the same results as arima. See arima for more details.

Note

Missing values are removed before the estimate. Sparse seasonal ARIMA(p,d,q)x(P,D,Q)(S) model is not allowed.

Author(s)

Debin Qiu

References

Brockwell, P. J. and Davis, R. A. (1996). Introduction to Time Series and Forecasting. Springer, New York. Sections 3.3 and 8.3.

See Also

arima, identify, forecast

Examples

```
estimate(lh, p = 1) # AR(1) process
estimate(lh, p = 1, q = 1) # ARMA(1,1) process
estimate(lh, p = c(1,3)) # sparse AR((1,3)) process

# seasonal ARIMA(0,1,1)x(0,1,1)(12) model
estimate(USAccDeaths, p = 1, d = 1, PDQ = c(0,1,1))
```

---

expsmooth *Simple Exponential Smoothing*

Description

Performs a simple exponential smoothing for univariate time series with no trend or seasonal pattern.

Usage

```
expsmooth(x, trend = 1, alpha = 0.2, beta = 0.10557, gamma = 0.07168,
  lead = 0, plot = TRUE)
```

Arguments

x	a numeric vector or univariate time series.
trend	the type of trend. See details.
alpha	the smoothing parameter for constant component. The default is 0.2.
beta	the smoothing parameter for linear component. The default is 0.10557.
gamma	the smoothing parameter for quadratic component. The default is 0.07168.
lead	the number of steps ahead for which prediction is required. The default is 0.
plot	a logical value indicating to print the plot of original data v.s smoothed data. The default is TRUE.

Details

Simple exponential smoothing is a weighted average between the most recent observation and the most recent forecasting, with weights α and $1 - \alpha$, respectively. To be precise, the smoothing equation of single exponential smoothing (constant model, `trend = 1`) is given by

$$level[t] = \alpha * x[t] + (1 - \alpha) * level[t - 1],$$

and the forecasting equation is

$$hatx[t + 1|t] = level[t],$$

for $t = 1, ..., n$. The initial value $level[0] = x[1]$. For example, $hatx[1|0] = level[0]$, $hatx[2|1] = level[1]$,..., etc.

Let $x1[t]$ be the smoothed values of single exponential smoothing. The double exponential smoothing (`trend = 2`, a linear model) is to apply a single exponential smoothing again to the smoothed sequence $x1[t]$, with a new smoothing parameter `beta`. Similarly, we denote the smoothed values of double exponential smoothing to be $x2[t]$. The triple exponential smoothing (`trend = 3`, a quadratic model) is to apply the single exponential smoothing to the smoothed sequence $x2[t]$ with a new smoothing parameter `gamma`. The default smoothing parameters (weights) `alpha`, `beta`, `gamma` are taken from the equation `1 - 0.8^{1/trend}` respectively, which is similar to the FORECAST procedure in SAS.

Value

A list with class `"es"` containing the following components:

`estimate`	the smoothed values.
`pred`	the predicted values when `lead` > 0.
`accurate`	the accurate measurements.

Note

Missing values are removed before the analysis.

Author(s)

Debin Qiu

See Also

`Winters`, `Holt`, `MA`

Examples

```
x <- rnorm(100)
es <- expsmooth(x) # trend = 1: a constant model
plot(x,type = "l")
lines(es$estimate,col = 2)
expsmooth(x,trend = 2) # trend = 2: a linear model
expsmooth(x,trend = 3) # trend = 3: a quadratic model
```

forecast *Forecast From ARIMA Fits*

Description

Forecasts from models fitted by `arima` or `estimate` function.

Usage

```
forecast(object, lead = 1, id = NULL, alpha = 0.05, output = TRUE)
```

Arguments

`object`	the result of an `arima` or `estimate` fit.
`lead`	the number of steps ahead for which prediction is required. The default is `1`.
`id`	the id of the observation which is the time. The default is `NULL`.
`alpha`	the significant level for constructing the confidence interval of prediction. The default is `0.05`.
`output`	a logical value indicating to print the results in R console. The default is `TRUE`.

Details

This function is originally from `predict.Arima` in `stats` package, but has a nice output including 100*(1 - α)% confidence interval and a prediction plot. It is similar to FORECAST statement in PROC ARIMA of SAS.

Value

A matrix with `lead` rows and five columns. Each column represents the number of steps ahead (`Lead`), the predicted values (`Forecast`), the standard errors (`S.E`) and the 100*(1 - α)% lower bound (`Lower`) and upper bound (`Upper`) of confidence interval.

Author(s)

Debin Qiu

See Also

`predict.Arima`

Examples

```
x <- arima.sim(list(order = c(3,0,0),ar = c(0.2,0.4,-0.15)),n = 100)
fit <- estimate(x,p = 3) # same as fit <- arima(x,order = c(3,0,0))
forecast(fit,lead = 4)

# forecast with id
t <- as.Date("2014-03-25") + 1:100
forecast(fit,lead = 4, id = t)
```

`Holt` *Holt's Two-parameter Exponential Smoothing*

Description

Performs Holt's two-parameter exponential smoothing for linear trend or damped trend.

Usage

```
Holt(x, type = c("additive", "multiplicative"), alpha = 0.2,
  beta = 0.1057, lead = 0, damped = FALSE, phi = 0.98, plot = TRUE)
```

Arguments

`x`	a numeric vector or univariate time series.
`type`	the type of interaction between the level and the linear trend. See details.
`alpha`	the parameter for the level smoothing. The default is `0.2`.
`beta`	the parameter for the trend smoothing. The default is `0.1057`.
`lead`	the number of steps ahead for which prediction is required. The default is `0`.
`damped`	a logical value indicating a damped trend. See details. The default is `FALSE`.
`phi`	a smoothing parameter for damped trend. The default is `0.98`, only valid for `damped = TRUE`.
`plot`	a logical value indicating to print the plot of original data v.s smoothed data. The default is `TRUE`.

Details

Holt's two parameter is used to forecast a time series with trend, but wihtout seasonal pattern. For the additive model (`type = "additive"`), the h-step-ahead forecast is given by $hatx[t+h|t] = level[t] + h * b[t]$, where

$$level[t] = \alpha * x[t] + (1-\alpha) * (b[t-1] + level[t-1]),$$

$$b[t] = \beta * (level[t] - level[t-1]) + (1-\beta) * b[t-1],$$

in which $b[t]$ is the trend component. For the multiplicative (`type = "multiplicative"`) model, the h-step-ahead forecast is given by $hatx[t+h|t] = level[t] + h * b[t]$, where

$$level[t] = \alpha * x[t] + (1-\alpha) * (b[t-1] * level[t-1]),$$

$$b[t] = \beta * (level[t]/level[t-1]) + (1-\beta) * b[t-1].$$

Compared with the Holt's linear trend that displays a constant increasing or decreasing, the damped trend generated by exponential smoothing method shows a exponential growth or decline, which is a situation between simple exponential smoothing (with 0 increasing or decreasing rate) and Holt's two-parameter smoothing. If `damped = TRUE`, the additive model becomes

$$hatx[t+h|t] = level[t] + (\phi + \phi^2 + ... + \phi^h) * b[t],$$

$$level[t] = \alpha * x[t] + (1 - \alpha) * (\phi * b[t-1] + level[t-1]),$$
$$b[t] = \beta * (level[t] - level[t-1]) + (1 - \beta) * \phi * b[t-1].$$

The multiplicative model becomes

$$hatx[t+h|t] = level[t] * b[t]^(\phi + \phi^2 + ... + \phi^h),$$
$$level[t] = \alpha * x[t] + (1 - \alpha) * (b[t-1]^\phi * level[t-1]),$$
$$b[t] = \beta * (level[t]/level[t-1]) + (1 - \beta) * b[t-1]^\phi.$$

See Chapter 7.4 for more details in R. J. Hyndman and G. Athanasopoulos (2013).

Value

A list with class "`Holt`" containing the following components:

`estimate`	the estimate values.
`alpha`	the smoothing parameter used for level.
`beta`	the smoothing parameter used for trend.
`phi`	the smoothing parameter used for damped trend.
`pred`	the predicted values, only available for `lead` > 0.
`accurate`	the accurate measurements.

Note

Missing values are removed before analysis.

Author(s)

Debin Qiu

References

R. J. Hyndman and G. Athanasopoulos, "Forecasting: principles and practice," 2013. [Online]. Available: http://otexts.org/fpp/.

See Also

`HoltWinters`, `expsmooth`, `Winters`

Examples

```
x <- (1:100)/100
y <- 2 + 1.2*x + rnorm(100)

ho0 <- Holt(y) # with additive interaction
ho1 <- Holt(y,damped = TRUE) # with damped trend

# multiplicative model for AirPassengers data,
# although seasonal pattern exists.
ho2 <- Holt(AirPassengers,type = "multiplicative")
```

`identify` *Identify a Time Series Model*

Description

Checks the white noise and stationarity of a univariate time series, and also identifies an appropriate ARIMA model using AICC criterion.

Usage

```
identify(x, p = NULL, q = NULL, nlag = 6, intercept = TRUE,
  stat.test = FALSE, method = c("adf", "pp", "kpss"), output = TRUE)
```

Arguments

`x`	a numeric vector or univariate time series.
`p`	the maximum lag order for AR process. The default is `NULL`.
`q`	the maximum lag order for MA process. The default is `NULL`.
`nlag`	the lag parameter to calculate the Ljung-Box test statistic. The default is `6`.
`intercept`	an intercept to be included in ARIMA model, only valid for `p` > 0 or `q` > 0. The default is `TRUE`.
`stat.test`	the stationary test for time series, see `stationary.test` for more details. The default is `FALSE`.
`method`	the method of stationary test, only valid for `stat.test = TRUE`.
`output`	a logical value indicating to print the results in R console. The default is `TRUE`.

Details

This function is similar to IDENTIFY statement in ARIMA procedure of SAS software, which is to check the white noise and stationarity for a univariate time series. The white noise check is accomplished by using `Box.test` in `stats` package, with the default method `type = "Ljung-Box"`. The stationary check uses the `stationary.test` implemented in this package.

The AICC criterion (Burnham and Anderson (2002)) is used to identify an optimal model which has the minimum AICC value. The AICC is defined as

$$AICC = AIC + 2k(k+1)/(n-k-1),$$

where $AIC = 2k - 2log(Loglik)$ which is called Akaike information criterion (Akaike (1974)). Here, k, n are the number of estimated parameters and observations, respectively. $Loglik$ is the maximized value of the likelihood function for the model.

Four plots are made: plot of original data, ACF plot, PACF plot and p.value of white noise check.

Value

A list with class "identify" containing the following components:

WNcheck	a matrix with three columns for results of white noise check for each lag.
aicc	the minimum AICC value, only available for p > 0 or q > 0.
min.p	the optimal order p for AR process, only available for p > 0 or q > 0.
min.q	the optimal order q for AR process, only available for p > 0 or q > 0.
stnt.test	a list of stationary test results with three components. See stationary.test for more details.

Note

Missing values are removed before the analysis.

Author(s)

Debin Qiu

References

Akaike, H. (1974), "A new look at the statistical model identification", *IEEE Transactions on Automatic Control*, 19 (6): 716-723.

Box, G. E. P. and Pierce, D. A. (1970), Distribution of residual correlations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association*, 65, 1509-1526.

Burnham, K. P.; Anderson, D. R. (2002), Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach (2nd ed.), Springer-Verlag

Ljung, G. M. and Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika* 65, 297-303.

Harvey, A. C. (1993) Time Series Models. 2nd Edition, Harvester Wheatsheaf, NY, pp. 44, 45.

Examples

```
x <- arima.sim(list(order = c(2,0,0),ar = c(0.2,0.4)),n = 100)
identify(x) # white noise check
identify(x,stat.test = TRUE) # white noise and stationarity check
identify(x,p = 3,q = 2) # white noise check and optimal model identification.
```

kpss.test *Kwiatkowski-Phillips-Schmidt-Shin Test*

Description

Performs Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test for the null hypothesis that x is a stationary univariate time series.

Usage

```
kpss.test(x, lag.short = TRUE, output = TRUE)
```

Arguments

x	a numeric vector or univariate time series.
lag.short	a logical value indicating whether the parameter of lag to calculate the test statistic is a short or long term. The default is a short term. See details.
output	a logical value indicating to print out the results in R console. The default is TRUE.

Details

The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test tends to decompose the time series into the sum of a deterministic trend, a random walk, and a stationary error:

$$x[t] = \alpha * t + u[t] + e[t],$$

where $u[t]$ satisfies $u[t] = u[t-1] + a[t]$, and $a[t]$ are i.i.d $(0, \sigma^2)$. The null hypothesis is that $\sigma^2 = 0$, which implies x is a stationary time series. In order to calculate the test statistic, we consider three types of linear regression models. The first type (type1) is the one with no drift and deterministic trend, defined as

$$x[t] = u[t] + e[t].$$

The second type (type2) is the one with drift but no trend:

$$x[t] = \mu + u[t] + e[t].$$

The third type (type3) is the one with both drift and trend:

$$x[t] = \mu + \alpha * t + u[t] + e[t].$$

The details of calculation of test statistic (kpss) can be seen in the references below. The default parameter of lag to calculate the test statistic is $max(1, floor(3 * sqrt(n)/13)$ for short term effect, otherwise, $max(1, floor(10 * sqrt(n)/13)$ for long term effect. The p.value is calculated by the interpolation of test statistic from tables of critical values (Table 5, Hobijn B., Franses PH. and Ooms M (2004)) for a given sample size n = length(x).

Value

A matrix for test results with three columns (`lag`, `kpss`, `p.value`) and three rows (`type1`, `type2`, `type3`). Each row is the test results (including lag parameter, test statistic and p.value) for each type of linear regression models.

Note

Missing values are removed.

Author(s)

Debin Qiu

References

Hobijn B, Franses PH and Ooms M (2004). Generalization of the KPSS-test for stationarity. *Statistica Neerlandica*, vol. 58, p. 482-502.

Kwiatkowski, D.; Phillips, P. C. B.; Schmidt, P.; Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*, 54 (1-3): 159-178.

See Also

`adf.test`, `pp.test`, `stationary.test`

Examples

```
# KPSS test for AR(1) process
x <- arima.sim(list(order = c(1,0,0),ar = 0.2),n = 100)
kpss.test(x)

# KPSS test for co2 data
kpss.test(co2)
```

MA *Moving Average Filter*

Description

Applies moving average filter to estimate the linear trend or nonseasonal pattern.

Usage

```
MA(x, nlag = NULL, plot = TRUE)
```

Arguments

`x`	a numeric vector or univariate time series.
`nlag`	the number of period to calculate the average. The default is `NULL`.
`plot`	a logical value indicating to print out the plot. The default is `TRUE`.

Details

The moving average filter uses the unweight mean of (2*nlag + 1) adjacent observations. That is,

$$hatX[t] = (X[t - nlag] + ... + X[t] + ... + X[t + nlag])/(2 * nlag + 1)$$

for $nlag < t < n - nlag$. For the values at the boundary $t \leq nlag$ or $n - nlag \leq t \leq n$, you can refer to Equation (7) in Qiu *et al.,* (2013) for details of calculations. The default method for choosing the optimal nlag uses the rule-of-thumb criterion proposed by Qiu, *et al*., (2013), in which they showed that the moving average is a special case of local linear estimator in the sense that the kernel function is the uniform one, and the moving average period nlag is a function of bandwidth. Thus, choosing the optimal nlag is equivalent to choosing the optimal bandwidth in local linear regression.

The plot of original values v.s fitted values will be displayed if plot = TRUE.

Value

A list with class "MA" containing the following components:

estimate	the smoothed values.
nlag	the period used to compute the average.
accurate	the accurate measurements.

Author(s)

Debin Qiu

References

D. Qiu, Q. Shao, and L. Yang (2013), Efficient inference for autoregressive coefficient in the presence of trend. *Journal of Multivariate Analysis* 114, 40-53.

P.J. Brockwell, R.A. Davis, Time Series: Theory and Methods, second ed., Springer, New York, 1991.

Examples

```
x <- arima.sim(list(order = c(1,0,0),ar = 0.4),n = 100)
y <- 5*(1:100)/100 + x
MA(y)

# moving average filter for co2 data
MA(co2)
```

pp.test *Phillips-Perron Test*

Description

Performs the Phillips-Perron test for the null hypothesis of a unit root of a univariate time series x (equivalently, x is a non-stationary time series).

Usage

```
pp.test(x, type = c("Z_rho", "Z_tau"), lag.short = TRUE, output = TRUE)
```

Arguments

x	a numeric vector or univariate time series.
type	the type of Phillips-Perron test. The default is Z_rho.
lag.short	a logical value indicating whether the parameter of lag to calculate the statistic is a short or long term. The default is a short term.
output	a logical value indicating to print the results in R console. The default is TRUE.

Details

Compared with the Augmented Dickey-Fuller test, Phillips-Perron test makes correction to the test statistics and is robust to the unspecified autocorrelation and heteroscedasticity in the errors. There are two types of test statistics, Z_ρ and Z_τ, which have the same asymptotic distributions as Augmented Dickey-Fuller test statistic, ADF. The calculations of each type of the Phillips-Perron test can be see in the reference below. If the lag.short = TRUE, we use the default number of Newey-West lags $floor(4 * (length(x)/100)^0.25)$, otherwise $floor(12 * (length(x)/100)^0.25)$ to calculate the test statistics. In order to calculate the test statistic, we consider three types of linear regression models. The first type (type1) is the one with no drift and linear trend with respect to time:

$$x[t] = \rho * x[t-1] + e[t],$$

where $e[t]$ is an error term. The second type (type2) is the one with drift but no linear trend:

$$x[t] = \mu + \rho * x[t-1] + e[t].$$

The third type (type3) is the one with both drift and linear trend:

$$x[t] = \mu + \alpha * t + \rho * x[t-1] + e[t].$$

The p.value is calculated by the interpolation of test statistics from the critical values tables (Table 10.A.1 for Z_rho and 10.A.2 for Z_tau in Fuller (1996)) with a given sample size n = length(x).

Value

A matrix for test results with three columns (lag,Z_rho or Z_tau, p.value) and three rows (type1, type2, type3). Each row is the test results (including lag parameter, test statistic and p.value) for each type of linear equation.

Note

Missing values are removed.

Author(s)

Debin Qiu

References

Phillips, P. C. B.; Perron, P. (1988). Testing for a Unit Root in Time Series Regression. *Biometrika*, 75 (2): 335-346.

Fuller, W. A. (1996). Introduction to statistical time series, second ed., Wiley, New York.

See Also

`adf.test`, `kpss.test`, `stationary.test`

Examples

```
# PP test for ar(1) process
x <- arima.sim(list(order = c(1,0,0),ar = 0.2),n = 100)
pp.test(x)

# PP test for co2 data
pp.test(co2)
```

`stationary.test` *Stationary Test for Univariate Time Series*

Description

Performs stationary test for a univariate time series.

Usage

```
stationary.test(x, method = c("adf", "pp", "kpss"), nlag = NULL,
  type = c("Z_rho", "Z_tau"), lag.short = TRUE, output = TRUE)
```

Arguments

`x`	a numeric vector or univariate time series.
`method`	a character indicating which test to use. The default is `"adf"` by Augmented Dickey-Fuller test.
`nlag`	the lag order to calculate the test statistic, only valid for `method = "adf"`. See `adf.test` for more details.
`type`	the test type, only valid for `method = "pp"`. See `pp.test` for more details.
`lag.short`	a logical value, only valid for `method = "pp"` or `"kpss"`. See `pp.test` and `kpss.test` for more details.
`output`	a logical value indicating to print the results in R console. The default is `TRUE`.

Details

This function combines the existing functions `adf.test`, `pp.test` and `kpss.test` for testing the stationarity of a univariate time series `x`.

Value

The results are the same as one of the `adf.test`, `pp.test`, `kpss.test`, depending on which test are used.

Note

Missing values are removed.

Author(s)

Debin Qiu

Examples

```
x <- arima.sim(list(order = c(1,0,0),ar = 0.2),n = 100)
stationary.test(x)  # same as adf.test(x)
stationary.test(x, method = "pp") # same as pp.test(x)
stationary.test(x, method = "kpss") # same as kpss.test(x)
```

---

stepar *Stepwise Autoregressive Model*

Description

Fit a stepwise autoregressive model

Usage

```
stepar(y, xreg = NULL, trend = c("linear", "quadratic", "constant"),
  order = NULL, lead = 0, newx = NULL, output = TRUE, ...)
```

Arguments

`y`	a numeric vector of response
`xreg`	a numeric vector or matrix of exogenous input variables. The default is `NULL`.
`trend`	the type of trend with respective to time. The default is `linear`.
`order`	the order to fit the AR model for residuals. The default is `NULL`.
`lead`	the number of steps ahead for which prediction is required. The default is `0`.
`newx`	a matrix of new data of `xreg` for predictions. The default is `NULL`.
`output`	a logical value indicating to print the results in R console. The default is `NULL`.
`...`	additional arguments for `ar` function.

Details

The stewise autoregressive model uses a two-stage procedure to fit time series. The first stage is to fit a (constant,linear,quadratic) model with respective to time sequence: $t = (1:n)/n$, where $n = length(y)$. If xreg is supplied, the fitted model is updated by

$$y = \mu + \beta * xreg + e[t]$$

for trend = "constant", and

$$y = \mu + \beta * xreg + \alpha * t + e[t]$$

for trend = "linear", and

$$y = \mu + \beta * xreg + \alpha[1] * t + \alpha[2] * t^2 + e[t]$$

for trend = "quadratic". The second stage is to fit an autoregressive process to the residuals of the fitted model obtained in the first stage, which is accomplished by using ar function in stats package.

Value

A list with class "stepar" containing the following components:

coef	a estimated coefficient matrix including the t-test results.
sigma	the square root of the estimated variance of the random error.
R.squared	the R^2 for fitted model in the first stage.
pred	the predictions, only available for lead > 0.

Note

If lead > 0 and xreg is supplied, newx must also be supplied in order to make a prediction. The newx must be a matrix with the same number of columns as xreg and the number of rows being equal to lead. The predictions should be used with cautions.

Author(s)

Debin Qiu

Examples

```
x <- 5*(1:100)/100
x <- x + arima.sim(list(order = c(1,0,0),ar = 0.4),n = 100)
stepar(x)
stepar(x,order = 1)

# with xreg supplied
X <- matrix(rnorm(200),100,2)
y <- 0.1*X[,1] + 1.2*X[,2] + rnorm(100)
stepar(y,X)
# make a prediction with lead = 1; used with caution.
newdat1 <- matrix(rnorm(2),nrow = 1)
```

```
fit1 <- stepar(y,X,lead = 1,newx = newdat1,output = FALSE)
# make a prediction with lead = 2; used with caution.
newdat2 <- matrix(rnorm(4),nrow = 2)
fit2 <- stepar(y,X,lead = 2,newx = newdat2,output = FALSE)
```

`trend.test` *Trend Test*

Description

Performs an approximate Cox-Stuart or Difference-Sign trend test.

Usage

```
trend.test(x, method = c("cox.stuart", "diff.sign"), plot = FALSE)
```

Arguments

`x`	a numeric vector or univariate time series.
`method`	test method. The default is `method = "cox.stuart"`.
`plot`	a logical value indicating to display the plot of data. The default is `FALSE`.

Details

Cox-Stuart or Difference-Sign test is used to test whether the data have a increasing or decreasing trend. They are useful to detect the linear or nonlinear trend. The Cox-Stuart test is constructed as follows. For the given data $x[1], ..., x[t]$, one can divide them into two sequences with equal number of observations cutted in the midpoint and then take the paired difference, i.e., $D = x[i] - x[i + c], i = 1, ..., floor(n/2)$, where c is the index of midpoint. Let S be the number of positive or negative values in D. Under the null hypothesis that data have no trend, for large n = length(x), S is approximately distributed as $N(n/2, n/4)$, such that one can immediately obtain the p value. The exact Cox-Stuart trend test can be seen in `cs.test` of `snpar` package.

The Difference-Sign test is constructed as the similar way as Cox-Stuart test. We first let $D = x[i] - x[i-1]$ for $i = 2, ..., n$ and then count the number of positive or negative values in D, defined as S. Under the null hypothesis, S is approximately distributed as $N((n-1)/2, (n+1)/12)$. Thus, p-value can be calculated based on the null distribution.

Value

A list with class "`htest`" containing the following components:

`data.name`	a character string giving the names of the data.
`method`	the type of test applied.
`alternative`	a character string describing the alternative hypothesis.
`p.value`	the p-value for the test.
`statistic`	the value of the test statistic with a name describing it.

Note

Missing values are removed.

Author(s)

Debin Qiu

References

D.R. Cox and A. Stuart (1955). Some quick sign tests for trend in location and dispersion. *Biometrika*, Vol. 42, pp. 80-95.

P.J. Brockwell, R.A. Davis, Time Series: Theory and Methods, second ed., Springer, New York, 1991. (p. 37)

Examples

```
x <- rnorm(100)
trend.test(x,plot = TRUE) # no trend

x <- 5*(1:100)/100
x <- x + arima.sim(list(order = c(1,0,0),ar = 0.4),n = 100)
trend.test(x,plot = TRUE) # increasing trend
```

`ts.diag` *Diagnostics for ARIMA fits*

Description

Performs diagnostics for ARIMA model fitted by `arima` or `estimate` with output of diagnostic plots.

Usage

```
ts.diag(object, lag.seq = NULL)
```

Arguments

`object`	the result of an `arima` or `estimate` fit.
`lag.seq`	the sequence of lag to calculate the Ljung-Box test statistics. The default is `NULL`.

Details

This function is similar to `ts.diag` in `stats` package, but with one more diagnostic plot for the normality of residuals. Also, the default sequence of lags for a Ljung-Box test is set to be `seq(4,24,by = 4)` if sample size $n > 24$, otherwise `seq(1,n,4)`. This function has been automatically implemented in `estimate` function.

Diagnostics are plotted, including the ACF plot, PACF plot, p.value of white noise checking plot, and Q-Q plot for residuals.

Value

A matrix for the result of white noise checking by Ljung-Box test.

Author(s)

Debin Qiu

Examples

```
x <- arima.sim(list(order = c(3,0,0),ar = c(0.2,0.4,-0.15)),n = 100)
fit <- estimate(x,p = 3) # same as fit <- arima(x,order = c(3,0,0))
ts.diag(fit)
```

Winters *Winters Three-parameter Smoothing*

Description

Performs Winters three-parameter smoothing for a univariate time series with seasonal pattern.

Usage

```
Winters(x, period = NULL, trend = 2, lead = 0, plot = TRUE,
  seasonal = c("additive", "multiplicative"), damped = FALSE, alpha = 0.2,
  beta = 0.1057, gamma = 0.07168, phi = 0.98, a.start = NA,
  b.start = NA, c.start = NA)
```

Arguments

x	a univariate time series.
period	seasonal period. The default is NULL.
trend	the type of trend component, can be one of 1,2,3 which represents constant, linear and quadratic trend, respectively. The default is trend = 2.
lead	the number of steps ahead for which prediction is required. The default is 0.
plot	a logical value indicating to display the smoothed graph. The default is TRUE.
seasonal	character string to select an "additive" or "multiplicative" seasonal model. The default is "additive".
damped	a logical value indicating to include the damped trend, only valid for trend = 2. The default is FALSE.
alpha	the parameter of level smoothing. The default is 0.2.
beta	the parameter of trend smoothing. The default is 0.1057.
gamma	the parameter of season smoothing. The default is 0.07168.
phi	the parameter of damped trend smoothing, only valid for damped = TRUE. The default is 0.98.
a.start	the starting value for level smoothing. The default is NA.
b.start	the starting value for trend smoothing. The default is NA.
c.start	the starting value for season smoothing. The default is NA.

Details

The Winters filter is used to decompose the trend and seasonal components by updating equations. This is similar to the function HoltWinters in stats package but may be in different perspective. To be precise, it uses the updating equations similar to exponential smoothing to fit the parameters for the following models when seasonal = "additive". If the trend is constant (trend = 1):

$$x[t] = a[t] + s[t] + e[t].$$

If the trend is linear (trend = 2):

$$x[t] = (a[t] + b[t] * t) + s[t] + e[t].$$

If the trend is quadratic (trend = 3):

$$x[t] = (a[t] + b[t] * t + c[t] * t^2) + s[t] + e[t].$$

Here $a[t], b[t], c[t]$ are the trend parameters, $s[t]$ is the seasonal parameter for the season corresponding to time t. For the multiplicative season, the models are as follows. If the trend is constant (trend = 1):

$$x[t] = a[t] * s[t] + e[t].$$

If the trend is linear (trend = 2):

$$x[t] = (a[t] + b[t] * t) * s[t] + e[t].$$

If the trend is quadratic (trend = 3):

$$x[t] = (a[t] + b[t] * t + c[t] * t^2) * s[t] + e[t].$$

When seasonal = "multiplicative", the updating equations for each parameter can be seen in page 606-607 of PROC FORECAST document of SAS. Similarly, for the additive seasonal model, the 'division' (/) for $a[t]$ and $s[t]$ in page 606-607 is changed to 'minus' (-).

The default starting values for a, b, c are computed by a time-trend regression over the first cycle of time series. The default starting values for the seasonal factors are computed from seasonal averages. The default smoothing parameters (weights) alpha,beta, gamma are taken from the equation 1 - 0.8^{1/trend} respectively. You can also use the HoltWinters function to get the optimal smoothing parameters and plug them back in this function.

The prediction equation is $x[t + h] = (a[t] + b[t] * t) * s[t + h]$ for trend = 2 and seasonal = "additive". Similar equations can be derived for the other options. When the damped = TRUE, the prediction equation is $x[t + h] = (a[t] + (\phi + ... + \phi^(h)) * b[t] * t) * s[t + h]$. More details can be referred to R. J. Hyndman and G. Athanasopoulos (2013).

Value

A list with class "Winters" containing the following components:

season	the seasonal factors.
estimate	the smoothed values.
pred	the prediction, only available with lead > 0.
accurate	the accurate measurements.

Note

The sum of seasonal factors is equal to the seasonal period. This restriction is to ensure the identifiability of seasonality in the models above.

Author(s)

Debin Qiu

References

P. R. Winters (1960) Forecasting sales by exponentially weighted moving averages, *Management Science* 6, 324-342.

R. J. Hyndman and G. Athanasopoulos, "Forecasting: principles and practice," 2013. [Online]. Available: http://otexts.org/fpp/.

See Also

HoltWinters, Holt, expsmooth

Examples

```
Winters(co2)

Winters(AirPassengers, seasonal = "multiplicative")
```

Index