Package 'forecast'

June 4, 2011

Title Forecasting functions for time series

Description Methods and tools for displaying and analysing univariate time series forecasts including exponential smoothing via state space models and automatic ARIMA modelling.

Version 2.19

Date 2011-06-04

Depends R (>= 2.0.0), graphics, stats, tseries, fracdiff

LazyData yes

LazyLoad yes

Author Rob J Hyndman <Rob.Hyndman@monash.edu>

Maintainer Rob J Hyndman <Rob.Hyndman@monash.edu>

License GPL (>= 2)

URL http://robjhyndman.com/software/forecast/

Repository CRAN

Date/Publication 2011-06-04 08:07:23

R topics documented:

`accuracy` *Accuracy measures for forecast model*

Description

Returns range of summary measures of the forecast accuracy. If `x` is provided, the function measures out-of-sample forecast accuracy based on x-f. If `x` is not provided, the function produces in-sample accuracy measures of the one-step forecasts based on f["x"]-fitted(f). All measures are defined and discussed in Hyndman and Koehler (2006).

Usage

```
accuracy(f, x, test=1:length(x),lambda=NULL)
```

Arguments

f	An object of class "forecast", or a numerical vector containing forecasts.
x	An optional numerical vector containing actual values of the same length as object.
test	Indicator of which elements of x and f to test.
lambda	Box-Cox parameter. If present, function backtransforms data and forecasts using InvBoxCox.

Value

Vector giving forecast accuracy measures.

Author(s)

Rob J Hyndman

References

Hyndman, R.J. and Koehler, A.B. (2006) "Another look at measures of forecast accuracy". *International Journal of Forecasting*, **22**(4).

Examples

```
fit1 <- rwf(EuStockMarkets[1:200,1],h=100)
fit2 <- meanf(EuStockMarkets[1:200,1],h=100)
accuracy(fit1)
accuracy(fit2)
accuracy(fit1,EuStockMarkets[201:300,1])
accuracy(fit2,EuStockMarkets[201:300,1])
plot(fit1)
lines(EuStockMarkets[1:300,1])
```

Acf *(Partial) Autocorrelation Function Estimation*

Description

Largely wrappers for the acf function in the stats package. The main difference is that Acf does not plot a spike at lag 0 (which is redundant). Pacf is included for consistency.

Usage

```
Acf(x, lag.max=NULL, type = c("correlation", "partial"), plot=TRUE, main=NULL, ...)
Pacf(x, main=NULL, ...)
```

Arguments

x	a univariate time series
lag.max	maximum lag at which to calculate the acf. Default is 10*log10(N/m) where N is the number of observations and m the number of series. Will be automatically limited to one less than the number of observations in the series.
type	character string giving the type of acf to be computed. Allowed values are "correlation" (the default) or "partial".
plot	logical. If TRUE (the default) the acf is plotted.
main	Title for plot
...	Additional arguments passed to acf.

Details

See the acf function in the stats package.

Value

See the acf function in the stats package.

Author(s)

Rob J Hyndman

See Also

acf

Examples

```
Acf(wineind)
Pacf(wineind)
```

---

arfima *Fit a fractionally differenced ARFIMA model*

Description

An ARFIMA(p,d,q) model is selected and estimated automatically using the Hyndman-Khandakar (2008) algorithm to select p and q and the Haslett and Raftery (1989) algorithm to estimate the parameters including d.

Usage

```
arfima(x, drange = c(0, 0.5), estim = c("mle","ls"), ...)
```

Arguments

x	a univariate time series (numeric vector).
drange	Allowable values of d to be considered. Default of c(0,0.5) ensures a stationary model is returned.
estim	If estim=="ls", then the ARMA parameters are calculated using the Haslett-Raftery algorithm. If estim=="mle", then the ARMA parameters are calculated using full MLE via the arima function.
...	Other arguments passed to auto.arima when selecting p and q.

Details

This function combines fracdiff and auto.arima to automatically select and estimate an ARFIMA model. The fractional differencing parameter is chosen first assuming an ARFIMA(2,d,0) model. Then the data are fractionally differenced using the estimated d and an ARMA model is selected for the resulting time series using auto.arima. Finally, the full ARFIMA(p,d,q) model is re-estimated using fracdiff. If estim=="mle", the ARMA coefficients are refined using arima.

Value

A list object of S3 class "fracdiff", which is described in the fracdiff documentation. A few additional objects are added to the list including x (the original time series), and the residuals and fitted values.

Author(s)

Rob J Hyndman and Farah Yasmeen

References

J. Haslett and A. E. Raftery (1989) Space-time Modelling with Long-memory Dependence: Assessing Ireland's Wind Power Resource (with discussion); *Applied Statistics* **38**, 1-50.

Hyndman, R.J. and Khandakar, Y. (2008) "Automatic time series forecasting: The forecast package for R", *Journal of Statistical Software*, **26**(3).

See Also

fracdiff, auto.arima, forecast.fracdiff.

Examples

```
x <- fracdiff.sim( 100, ma = -.4, d = .3)$series
fit <- arfima(x)
tsdisplay(residuals(fit))
```

`Arima` *Fit ARIMA model to univariate time series*

Description

Largely a wrapper for the `arima` function in the stats package. The main difference is that this function allows a drift term. It is also possible to take an ARIMA model from a previous call to `Arima` and re-apply it to the data `x`.

Usage

```
Arima(x, order = c(0, 0, 0), seasonal = list(order = c(0, 0, 0), period = NA),
      xreg = NULL, include.mean = TRUE, include.drift = FALSE,
      transform.pars = TRUE, fixed = NULL, init = NULL,
      method = c("CSS-ML", "ML", "CSS"), n.cond,
      optim.control = list(), kappa = 1e6, model=NULL)
```

Arguments

`x`	a univariate time series
`order`	A specification of the non-seasonal part of the ARIMA model: the three components (p, d, q) are the AR order, the degree of differencing, and the MA order.
`seasonal`	A specification of the seasonal part of the ARIMA model, plus the period (which defaults to frequency(x)). This should be a list with components order and period, but a specification of just a numeric vector of length 3 will be turned into a suitable list with the specification as the order.
`xreg`	Optionally, a vector or matrix of external regressors, which must have the same number of rows as x.
`include.mean`	Should the ARIMA model include a mean term? The default is TRUE for undifferenced series, FALSE for differenced ones (where a mean would not affect the fit nor predictions).
`include.drift`	Should the ARIMA model include a linear drift term? (i.e., a linear regression with ARIMA errors is fitted.) The default is FALSE.
`transform.pars`	Logical. If true, the AR parameters are transformed to ensure that they remain in the region of stationarity. Not used for method = "CSS".
`fixed`	optional numeric vector of the same length as the total number of parameters. If supplied, only NA entries in fixed will be varied. transform.pars = TRUE will be overridden (with a warning) if any AR parameters are fixed. It may be wise to set transform.pars = FALSE when fixing MA parameters, especially near non-invertibility.
`init`	optional numeric vector of initial parameter values. Missing values will be filled in, by zeroes except for regression coefficients. Values already specified in fixed will be ignored.

method	Fitting method: maximum likelihood or minimize conditional sum-of-squares. The default (unless there are missing values) is to use conditional-sum-of-squares to find starting values, then maximum likelihood.
n.cond	Only used if fitting by conditional-sum-of-squares: the number of initial observations to ignore. It will be ignored if less than the maximum lag of an AR term.
optim.control	List of control parameters for optim.
kappa	the prior variance (as a multiple of the innovations variance) for the past observations in a differenced model. Do not reduce this.
model	Output from a previous call to Arima. If model is passed, this same model is fitted to x without re-estimating any parameters.

Details

See the arima function in the stats package.

Value

See the arima function in the stats package. The additional objects returned are

x	The time series data
xreg	The regressors used in fitting (when relevant).

Author(s)

Rob J Hyndman

See Also

arima

Examples

```
fit <- Arima(WWWusage,order=c(3,1,0))
plot(forecast(fit,h=20))

air.model <- Arima(window(AirPassengers,end=1956+11/12),order=c(0,1,1),seasonal=list(order=c
plot(forecast(air.model,h=48))
lines(AirPassengers)

air.model2 <- Arima(window(AirPassengers,start=1957),model=air.model)
outofsample <- fitted(air.model2)
# in-sample one-step forecasts
accuracy(air.model)
# out-of-sample one-step forecasts
accuracy(air.model2)
# out-of-sample multi-step forecasts
accuracy(forecast(air.model,h=48),outofsample)
```

`arima.errors` *ARIMA errors*

Description

Returns original time series after adjusting for regression variables. These are not the same as the residuals. If there are no regression variables in the ARIMA model, then the errors will be identical to the original series. If there are regression variables in the ARIMA model, then the errors will be equal to the original series minus the effect of the regression variables, but leaving in the serial correlation that is modelled with the AR and MA terms. If you want the "residuals", then use `residuals(z)`..

Usage

```
arima.errors(z)
```

Arguments

`z`	Fitted ARIMA model from `arima`

Value

A time series containing the "errors".

Author(s)

Rob J Hyndman

See Also

`arima`, `residuals`

Examples

```
ukdeaths.fit <- Arima(UKDriverDeaths,c(1,0,1),c(0,1,1),xreg=Seatbelts[,"law"])
ukdeaths.errors <- arima.errors(ukdeaths.fit)
par(mfrow=c(2,1))
plot(UKDriverDeaths)
plot(ukdeaths.errors)
```

`auto.arima` *Fit best ARIMA model to univariate time series*

Description

Returns best ARIMA model according to either AIC, AICc or BIC value. The function conducts a search over possible model within the order constraints provided.

Usage

```
auto.arima(x, d = NA, D = NA, max.p = 5, max.q = 5,
           max.P = 2, max.Q = 2, max.order = 5,
           start.p=2, start.q=2, start.P=1, start.Q=1,
           stationary = FALSE, ic = c("aic","aicc", "bic"),
           stepwise=TRUE, trace=FALSE,
           approximation=length(x)>100 | frequency(x)>12, xreg=NULL,
           test=c("kpss","adf","pp"), allowdrift=TRUE)
```

Arguments

`x`	a univariate time series
`d`	Order of first-differencing. If missing, will choose a value based on KPSS test.
`D`	Order of seasonal-differencing. If missing, will choose a value based on CH test.
`max.p`	Maximum value of p
`max.q`	Maximum value of q
`max.P`	Maximum value of P
`max.Q`	Maximum value of Q
`max.order`	Maximum value of p+q+P+Q if model selection is not stepwise.
`start.p`	Starting value of p in stepwise procedure.
`start.q`	Starting value of q in stepwise procedure.
`start.P`	Starting value of P in stepwise procedure.
`start.Q`	Starting value of Q in stepwise procedure.
`stationary`	If `TRUE`, restricts search to stationary models.
`ic`	Information criterion to be used in model selection.
`stepwise`	If `TRUE`, will do stepwise selection (faster). Otherwise, it searches over all models. Non-stepwise selection can be very slow, especially for seasonal models.
`trace`	If `TRUE`, the list of ARIMA models considered will be reported.
`approximation`	If `TRUE`, estimation is via conditional sums of squares and the information criteria used for model selection are approximated. The final model is still computed using maximum likelihood estimation. Approximation should be used for long time series or a high seasonal period to avoid excessive computation times.

xreg	Optionally, a vector or matrix of external regressors, which must have the same number of rows as x.
test	Type of unit root test to use. See ndiffs for details.
allowdrift	If TRUE, models with drift terms are considered.

Details

Non-stepwise selection can be slow, especially for seasonal data. Non-seasonal differences chosen using the KPSS test. Seasonal differences chosen using a variation on the Canova-Hansen test. Stepwise algorithm outlined in Hyndman and Khandakar (2008).

Value

Same as for arima

Author(s)

Rob J Hyndman

References

Hyndman, R.J. and Khandakar, Y. (2008) "Automatic time series forecasting: The forecast package for R", *Journal of Statistical Software*, **26**(3).

See Also

Arima

Examples

```
fit <- auto.arima(WWWusage)
plot(forecast(fit,h=20))
```

BoxCox *Box Cox Transformation*

Description

BoxCox() returns a transformation of the input variable using a Box-Cox transformation. InvBoxCox() reverses the transformation.

Usage

```
BoxCox(x,lambda)
InvBoxCox(x,lambda)
```

Arguments

x	a numeric vector or time series
lambda	transformation parameter

Details

The Box-Cox transformation is given by

$$f_\lambda(x) = \frac{x^\lambda - 1}{\lambda}$$

if $\lambda \neq 0$. For $\lambda = 0$,

$$f_0(x) = \log(x)$$

.

Value

a numeric vector of the same length as x.

Author(s)

Rob J Hyndman

References

Box, G. E. P. and Cox, D. R. (1964) An analysis of transformations. *JRSS B* **26** 211–246.

Examples

```
lynx.sqrt <- BoxCox(lynx,0.5)
lynx.fit <- ar(lynx.sqrt)
plot(forecast(lynx.fit,h=20),lambda=0.5)
```

---

croston *Forecasts for intermittent demand using Croston's method*

Description

Returns forecasts and other information for Croston's forecasts applied to x.

Usage

```
croston(x, h=10, alpha=0.1)
```

Arguments

x	a numeric vector or time series
h	Number of periods for forecasting.
alpha	Value of alpha. Default value is 0.1.

Details

Based on Croston's (1972) method for intermittent demand forecasting, also described in Shenstone and Hyndman (2005). Croston's method involves using simple exponential smoothing (SES) on the non-zero elements of the time series and a separate application of SES to the times between non-zero elements of the time series. The smoothing parameters of the two applications of SES are assumed to be equal and are denoted by `alpha`.

Note that prediction intervals are not computed as Croston's method has no underlying stochastic model.

Value

An object of class `"forecast"` is a list containing at least the following elements:

`model`	A list containing information about the fitted model. The first element gives the SES model used for non-zero demands. The second element gives the SES model used for times between non-zero demands. Both models are of class `forecast`.
`method`	The name of the forecasting method as a character string
`mean`	Point forecasts as a time series
`x`	The original time series (either `object` itself or the time series used to create the model stored as `object`).
`residuals`	Residuals from the fitted model. That is x minus fitted values.
`fitted`	Fitted values (one-step forecasts)

The function `summary` is used to obtain and print a summary of the results, while the function `plot` produces a plot of the forecasts.

The generic accessor functions `fitted.values` and `residuals` extract useful features of the value returned by `croston` and associated functions.

Author(s)

Rob J Hyndman

References

Croston, J. (1972) "Forecasting and stock control for intermittent demands", *Operational Research Quarterly*, **23**(3), 289-303.

Shenstone, L., and Hyndman, R.J. (2005) "Stochastic models underlying Croston's method for intermittent demand forecasting". *Journal of Forecasting*, **24**, 389-402.

See Also

`ses`.

Examples

```
x <- rpois(20,lambda=.3)
fcast <- croston(x)
plot(fcast)
```

CV *Cross-validation statistic*

Description

Computes cross-validation statistic, AIC, corrected AIC, BIC and adjusted R^2 values for a linear model.

Usage

```
CV(obj)
```

Arguments

obj	output from lm or tslm

Value

Numerical vector containing CV, AIC, AICc, BIC and AdjR2 values.

Author(s)

Rob J Hyndman

See Also

AIC

Examples

```
y <- ts(rnorm(120,0,3) + 20*sin(2*pi*(1:120)/12), frequency=12)
fit1 <- tslm(y ~ trend + season)
fit2 <- tslm(y ~ season)
CV(fit1)
CV(fit2)
```

`decompose` *Classical Seasonal Decomposition by Moving Averages*

Description

Decompose a time series into seasonal, trend and irregular components using moving averages. Deals with additive or multiplicative seasonal component.

Usage

```
decompose(x, type = c("additive", "multiplicative"), filter = NULL)
```

Arguments

`x`	A time series.
`type`	The type of seasonal component. Can be abbreviated.
`filter`	A vector of filter coefficients in reverse time order (as for AR or MA coefficients), used for filtering out the seasonal component. If `NULL`, a moving average with symmetric window is performed.

Details

The additive model used is:

$$Y_t = T_t + S_t + e_t$$

The multiplicative model used is:

$$Y_t = T_t \, S_t \, e_t$$

The function first determines the trend component using a moving average (if `filter` is `NULL`, a symmetric window with equal weights is used), and removes it from the time series. Then, the seasonal figure is computed by averaging, for each time unit, over all periods. The seasonal figure is then centered. Finally, the error component is determined by removing trend and seasonal figure (recycled as needed) from the original time series.

Value

An object of class `"decomposed.ts"` with following components:

`seasonal`	The seasonal component (i.e., the repeated seasonal figure)
`figure`	The estimated seasonal figure only
`trend`	The trend component
`random`	The remainder part
`type`	The value of `type`

Note

This function is identical to the `decompose` function in the `stats` package except that the seasonal component is not incorrectly truncated.

Author(s)

David Meyer <David.Meyer@wu-wien.ac.at>. Revised by Rob J Hyndman <Rob.Hyndman@monash.edu>.

References

M. Kendall and A. Stuart (1983) The Advanced Theory of Statistics, Vol.3, *Griffin*, 410–414.

See Also

decompose

Examples

```
m <- decompose(co2)
plot(m)
```

---

dm.test *Diebold-Mariano test for predictive accuracy*

Description

The Diebold-Mariano test compares the forecast accuracy of two forecast methods. The null hypothesis is that they have the same forecast accuracy.

Usage

```
dm.test(e1, e2, alternative = c("two.sided", "less", "greater"), h = 1, power = 2)
```

Arguments

e1	Forecast errors from method 1.
e2	Forecast errors from method 2.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
h	The forecast horizon used in calculating e1 and e2.
power	The power used in the loss function. Usually 1 or 2.

Value

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

statistic	the value of the DM-statistic.
parameter	the forecast horizon and loss function power used in the test.
alternative	a character string describing the alternative hypothesis.
p.value	the p-value for the test.
method	a character string with the value "Diebold-Mariano Test".
data.name	a character vector giving the names of the two error series.

Author(s)

George Athanasopoulos and Rob Hyndman

References

Diebold, F.X. and Mariano, R.S. (1995) Comparing predictive accuracy. *Journal of Business and Economic Statistics*, **13**, 253-263.

Examples

```
# Test on in-sample one-step forecasts
f1 <- ets(WWWusage)
f2 <- auto.arima(WWWusage)
accuracy(f1)
accuracy(f2)
dm.test(residuals(f1),residuals(f2),h=1)

# Test on out-of-sample one-step forecasts
f1 <- ets(WWWusage[1:80])
f2 <- auto.arima(WWWusage[1:80])
f1.out <- ets(WWWusage[81:100],model=f1)
f2.out <- Arima(WWWusage[81:100],model=f2)
accuracy(f1.out)
accuracy(f2.out)
dm.test(residuals(f1.out),residuals(f2.out),h=1)
```

---

ets *Exponential smoothing state space model*

Description

Returns ets model applied to `y`.

Usage

```
ets(y, model="ZZZ", damped=NULL, alpha=NULL, beta=NULL, gamma=NULL, phi=NULL,
    additive.only=FALSE, lower=c(rep(0.0001,3), 0.8), upper=c(rep(0.9999,3),0.98),
    opt.crit=c("lik","amse","mse","sigma"), nmse=3,
    bounds=c("both","usual","admissible"),  ic=c("aic","aicc","bic"),
    restrict=TRUE)
```

Arguments

`y`	a numeric vector or time series
`model`	Usually a three-character string identifying method using the framework terminology of Hyndman et al. (2002) and Hyndman et al. (2008). The first letter denotes the error type ("A", "M" or "Z"); the second letter denotes the trend type ("N","A","M" or "Z"); and the third letter denotes the season type ("N","A","M"

	or "Z"). In all cases, "N"=none, "A"=additive, "M"=multiplicative and "Z"=automatically selected. So, for example, "ANN" is simple exponential smoothing with additive errors, "MAM" is multiplicative Holt-Winters' method with multiplicative errors, and so on. It is also possible for the model to be equal to the output from a previous call to `ets`. In this case, the same model is fitted to `y` without re-estimating any parameters.
`damped`	If TRUE, use a damped trend (either additive or multiplicative). If NULL, both damped and non-damped trends will be tried and the best model (according to the information criterion `ic`) returned.
`alpha`	Value of alpha. If NULL, it is estimated.
`beta`	Value of beta. If NULL, it is estimated.
`gamma`	Value of gamma. If NULL, it is estimated.
`phi`	Value of phi. If NULL, it is estimated.
`additive.only`	If TRUE, will only consider additive models. Default is FALSE.
`lower`	Lower bounds for the parameters (alpha, beta, gamma, phi)
`upper`	Upper bounds for the parameters (alpha, beta, gamma, phi)
`opt.crit`	Optimization criterion. One of "mse" (Mean Square Error), "amse" (Average MSE over first `nmse` forecast horizons), "sigma" (Standard deviation of residuals), or "lik" (Log-likelihood, the default).
`nmse`	Number of steps for average multistep MSE (1<=`nmse`<=10).
`bounds`	Type of parameter space to impose: `"usual"` indicates all parameters must lie between specified lower and upper bounds; `"admissible"` indicates parameters must lie in the admissible space; `"both"` (default) takes the intersection of these regions.
`ic`	Information criterion to be used in model selection.
`restrict`	If TRUE, the models with infinite variance will not be allowed.

Details

Based on the classification of methods as described in Hyndman et al (2008).

The methodology is fully automatic. The only required argument for ets is the time series. The model is chosen automatically if not specified. This methodology performed extremely well on the M3-competition data. (See Hyndman, et al, 2002, below.)

Value

An object of class `"ets"`.

The generic accessor functions `fitted.values` and `residuals` extract useful features of the value returned by `ets` and associated functions.

Author(s)

Rob J Hyndman

References

Hyndman, R.J., Koehler, A.B., Snyder, R.D., and Grose, S. (2002) "A state space framework for automatic forecasting using exponential smoothing methods", *International J. Forecasting*, **18**(3), 439–454.

Hyndman, R.J., Akram, Md., and Archibald, B. (2008) "The admissible parameter space for exponential smoothing models". *Annals of Statistical Mathematics*, **60**(2), 407–426.

Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) *Forecasting with exponential smoothing: the state space approach*, Springer-Verlag. http://robjhyndman.com/expsmooth.

See Also

HoltWinters, rwf, arima.

Examples

```
fit <- ets(USAccDeaths)
plot(forecast(fit))
```

---

fitted.Arima *One-step in-sample forecasts using ARIMA models*

Description

Returns one-step forecasts for the data used in fitting the ARIMA model.

Usage

```
fitted.Arima(object, ...)
```

Arguments

object	An object of class "Arima". Usually the result of a call to arima.
...	Other arguments.

Value

An time series of the one-step forecasts.

Author(s)

Rob J Hyndman

See Also

forecast.Arima.

Examples

```
fit <- Arima(WWWusage,c(3,1,0))
plot(WWWusage)
lines(fitted(fit),col=2)
```

---

forecast *Forecasting time series*

Description

`forecast` is a generic function for forecasting from time series or time series models. The function invokes particular *methods* which depend on the class of the first argument.

For example, the function `forecast.Arima` makes forecasts based on the results produced by `arima`.

The function `forecast.ts` makes forecasts using exponential smoothing state space models.

Usage

```
forecast(object,...)
## S3 method for class 'ts'
forecast(object, h, level=c(80,95), fan=FALSE, ...)
```

Arguments

`object`	a time series or time series model for which forecasts are required
`h`	Number of periods for forecasting
`level`	Confidence level for prediction intervals.
`fan`	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.
`...`	Additional arguments affecting the forecasts produced. `forecast.ts` passes these to `forecast.ets`

Details

The default behaviour is to use a model estimated using `ets`.

Value

An object of class `"forecast"`.

The function `summary` is used to obtain and print a summary of the results, while the function `plot` produces a plot of the forecasts and prediction intervals.

The generic accessor functions `fitted.values` and `residuals` extract various useful features of the value returned by `forecast$model`.

An object of class `"forecast"` is a list containing at least the following elements:

`model`	A list containing information about the fitted model

method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The original time series (either object itself or the time series used to create the model stored as object).
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values (one-step forecasts)

Author(s)

Rob J Hyndman

See Also

Other functions which return objects of class "forecast" are forecast.ets, forecast.Arima, forecast.HoltWinters, forecast.StructTS, meanf, rwf, splinef, thetaf, croston, ses, holt, hw.

---

forecast.Arima *Forecasting using ARIMA or ARFIMA models*

Description

Returns forecasts and other information for univariate ARIMA models.

Usage

```
## S3 method for class 'Arima'
forecast(object, h=ifelse(object$arma[5]>1,2*object$arma[5],10),
    level=c(80,95), fan=FALSE, xreg=NULL,...)
## S3 method for class 'ar'
forecast(object, h=10, level=c(80,95), fan=FALSE, ...)
## S3 method for class 'fracdiff'
forecast(object, h=10, level=c(80,95), fan=FALSE, ...)
```

Arguments

object	An object of class "Arima", "ar" or "fracdiff". Usually the result of a call to arima, auto.arima, ar, arfima or fracdiff.
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.
level	Confidence level for prediction intervals.
fan	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.
xreg	Future values of an regression variables (for class Arima objects only).
...	Other arguments.

Details

For Arima or ar objects, the function calls predict.Arima or predict.ar and constructs an object of class "forecast" from the results. For fracdiff objects, the calculations are all done within forecast.fracdiff using the equations given by Peiris and Perera (1988).

Value

An object of class "forecast".

The function summary is used to obtain and print a summary of the results, while the function plot produces a plot of the forecasts and prediction intervals.

The generic accessor functions fitted.values and residuals extract useful features of the value returned by forecast.Arima.

An object of class "forecast" is a list containing at least the following elements:

model	A list containing information about the fitted model
method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The original time series (either object itself or the time series used to create the model stored as object).
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values (one-step forecasts)

Author(s)

Rob J Hyndman

References

Peiris, M. & Perera, B. (1988), On prediction with fractionally differenced ARIMA models, *Journal of Time Series Analysis*, **9**(3), 215-220.

See Also

predict.Arima, predict.ar, auto.arima, Arima, arima, ar, arfima.

Examples

```
fit <- Arima(WWWusage,c(3,1,0))
plot(forecast(fit))

x <- fracdiff.sim( 100, ma = -.4, d = .3)$series
fit <- arfima(x)
plot(forecast(fit,h=30))
```

`forecast.ets` *Forecasting using ETS models*

Description

Returns forecasts and other information for univariate ETS models.

Usage

```
## S3 method for class 'ets'
forecast(object, h=ifelse(object$m>1, 2*object$m, 10), level=c(80,95),
fan=FALSE, simulate=FALSE, bootstrap=FALSE, npaths=5000,...)
```

Arguments

`object`	An object of class "`ets`". Usually the result of a call to `ets`.
`h`	Number of periods for forecasting
`level`	Confidence level for prediction intervals.
`fan`	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.
`simulate`	If TRUE, prediction intervals produced by simulation rather than using analytic formulae.
`bootstrap`	If TRUE, and if `simulate=TRUE`, then simulation uses resampled errors rather than normally distributed errors.
`npaths`	Number of sample paths used in computing simulated prediction intervals.
`...`	Other arguments.

Value

An object of class "`forecast`".

The function `summary` is used to obtain and print a summary of the results, while the function `plot` produces a plot of the forecasts and prediction intervals.

The generic accessor functions `fitted.values` and `residuals` extract useful features of the value returned by `forecast.ets`.

An object of class "`forecast`" is a list containing at least the following elements:

`model`	A list containing information about the fitted model
`method`	The name of the forecasting method as a character string
`mean`	Point forecasts as a time series
`lower`	Lower limits for prediction intervals
`upper`	Upper limits for prediction intervals
`level`	The confidence values associated with the prediction intervals
`x`	The original time series (either `object` itself or the time series used to create the model stored as `object`).
`residuals`	Residuals from the fitted model. That is x minus fitted values.
`fitted`	Fitted values (one-step forecasts)

Author(s)

Rob J Hyndman

See Also

ets, ses, holt, hw.

Examples

```
fit <- ets(USAccDeaths)
plot(forecast(fit,h=48))
```

---

forecast.HoltWinters

Forecasting using Holt-Winters objects

Description

Returns forecasts and other information for univariate Holt-Winters time series models.

Usage

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

Arguments

object	An object of class "HoltWinters". Usually the result of a call to HoltWinters.
h	Number of periods for forecasting
level	Confidence level for prediction intervals.
fan	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.
...	Other arguments.

Details

This function calls predict.HoltWinters and constructs an object of class "forecast" from the results.

It is included for completeness, but the ets is recommended for use instead of HoltWinters.

Value

An object of class "forecast".

The function summary is used to obtain and print a summary of the results, while the function plot produces a plot of the forecasts and prediction intervals.

The generic accessor functions fitted.values and residuals extract useful features of the value returned by forecast.HoltWinters.

An object of class "forecast" is a list containing at least the following elements:

model	A list containing information about the fitted model
method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The original time series (either object itself or the time series used to create the model stored as object).
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values (one-step forecasts)

Author(s)

Rob J Hyndman

See Also

predict.HoltWinters, HoltWinters.

Examples

```
fit <- HoltWinters(WWWusage,gamma=FALSE)
plot(forecast(fit))
```

---

forecast.lm *Forecast a linear model with possible time series components*

Description

forecast.lm is used to predict linear models, especially those involving trend and seasonality components.

Usage

```
## S3 method for class 'lm'
forecast(object, newdata, level=c(80,95), fan=FALSE, h=10, ...)
```

Arguments

object	Object of class "lm", usually the result of a call to lm or tslm.
newdata	An optional data frame in which to look for variables with which to predict. If omitted, it is assumed that the only variables are trend and season, and h forecasts are produced.
level	Confidence level for prediction intervals.
fan	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.
h	Number of periods for forecasting. Ignored if newdata present.
...	Other arguments passed to predict.lm().

Details

forecast.lm is largely a wrapper for predict.lm() except that it allows variables "trend" and "season" which are created on the fly from the time series characteristics of the data. Also, the output is reformatted into a forecast object.

Value

An object of class "forecast".

The function summary is used to obtain and print a summary of the results, while the function plot produces a plot of the forecasts and prediction intervals.

The generic accessor functions fitted.values and residuals extract useful features of the value returned by forecast.lm.

An object of class "forecast" is a list containing at least the following elements:

model	A list containing information about the fitted model
method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The historical data for the response variable.
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values

Author(s)

Rob J Hyndman

See Also

tslm, lm.

Examples

```
y <- ts(rnorm(120,0,3) + 1:120 + 20*sin(2*pi*(1:120)/12), frequency=12)
fit <- tslm(y ~ trend + season)
plot(forecast(fit, h=20))
```

`forecast.stl` *Forecasting using stl objects*

Description

Returns forecasts obtained by either ETS or ARIMA models applied to the seasonally adjusted data from an STL decomposition.

Usage

```
## S3 method for class 'stl'
forecast(object, method=c("ets","arima"),
    h = frequency(object$time.series)*2, level = c(80, 95), fan = FALSE, ...)
stlf(x, h=frequency(x)*2, s.window=7, method=c("ets","arima"), level = c(80, 95), f
```

Arguments

`object`	An object of class "`stl`". Usually the result of a call to `stl`.
`x`	A univariate numeric time series of class "`ts`"
`s.window`	Either the character string "periodic" (default) or the span (in lags) of the loess window for seasonal extraction.
`method`	Method to use for forecasting the seasonally adjusted series.
`h`	Number of periods for forecasting.
`level`	Confidence level for prediction intervals.
`fan`	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.
`...`	Other arguments passed to `ets()` or `auto.arima()`.

Details

`forecast.stl` seasonally adjusts the data from an STL decomposition, then uses either ETS or ARIMA models to forecast the result. The seasonal component from the last year of data is added back in to the forecasts. Note that the prediction intervals ignore the uncertainty associated with the seasonal component.

`stlf` takes a `ts` argument and applies a stl decomposition before calling `forecast.stl`.

Value

An object of class "forecast".

The function summary is used to obtain and print a summary of the results, while the function plot produces a plot of the forecasts and prediction intervals.

The generic accessor functions fitted.values and residuals extract useful features of the value returned by forecast.stl.

An object of class "forecast" is a list containing at least the following elements:

model	A list containing information about the fitted model
method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The original time series (either object itself or the time series used to create the model stored as object).
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values (one-step forecasts)

Author(s)

Rob J Hyndman

See Also

forecast.ets, forecast.Arima.

Examples

```
fit <- stl(USAccDeaths,s.window="periodic")
plot(forecast(fit))

plot(snaive(wineind))
```

---

forecast.StructTS *Forecasting using Structural Time Series models*

Description

Returns forecasts and other information for univariate structural time series models.

Usage

```
## S3 method for class 'StructTS'
forecast(object, h=ifelse(object$call$type=="BSM",2*object$xtsp[3],10),
    level=c(80,95), fan=FALSE, ...)
```

Arguments

object	An object of class "StructTS". Usually the result of a call to StructTS.
h	Number of periods for forecasting
level	Confidence level for prediction intervals.
fan	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.
...	Other arguments.

Details

This function calls predict.StructTS and constructs an object of class "forecast" from the results.

Value

An object of class "forecast".

The function summary is used to obtain and print a summary of the results, while the function plot produces a plot of the forecasts and prediction intervals.

The generic accessor functions fitted.values and residuals extract useful features of the value returned by forecast.StructTS.

An object of class "forecast" is a list containing at least the following elements:

model	A list containing information about the fitted model
method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The original time series (either object itself or the time series used to create the model stored as object).
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values (one-step forecasts)

Author(s)

Rob J Hyndman

See Also

StructTS.

Examples

```
fit <- StructTS(WWWusage,"level")
plot(forecast(fit))
```

gas *Australian monthly gas production*

Description

Australian monthly gas production: 1956–1995.

Usage

```
gas
```

Format

Time series data

Source

Australian Bureau of Statistics.

Examples

```
plot(gas)
seasonplot(gas)
tsdisplay(gas)
```

gold *Daily morning gold prices*

Description

Daily morning gold prices in US dollars. 1 January 1985 – 31 March 1989.

Usage

```
data(gold)
```

Format

Time series data

Source

Time Series Data Library. http://robjhyndman.com/TSDL/

Examples

```
tsdisplay(gold)
```

`logLik.ets` *Log-Likelihood of an ets object*

Description

Returns the log-likelihood of the ets model represented by `object` evaluated at the estimated parameters.

Usage

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

Arguments

`object`	an object of class `ets`, representing an exponential smoothing state space model.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the log-likelihood of the model represented by `object` evaluated at the estimated parameters.

Author(s)

Rob J Hyndman

References

Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008) *Forecasting with exponential smoothing: the state space approach*, Springer-Verlag. http://robjhyndman.com/expsmooth.

See Also

`ets`

Examples

```
fit <- ets(USAccDeaths)
logLik(fit)
```

ma *Moving-average smoothing*

Description

Computes a simple moving average smoother.

Usage

```
ma(x, order, centre=TRUE)
```

Arguments

x	Univariate time series
order	Order of moving average smoother
centre	If TRUE, then the moving average is centred.

Value

Numerical time series object containing the smoothed values.

Author(s)

Rob J Hyndman

See Also

ksmooth, decompose

Examples

```
plot(wineind)
sm <- ma(wineind,order=12)
lines(sm,col="red")
```

meanf *Mean Forecast*

Description

Returns forecasts and prediction intervals for an iid model applied to x.

Usage

```
meanf(x, h=10, level=c(80,95), fan=FALSE)
```

Arguments

x	a numeric vector or time series
h	Number of periods for forecasting
level	Confidence levels for prediction intervals.
fan	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.

Details

The iid model is

$$Y_t = \mu + Z_t$$

where Z_t is a normal iid error. Forecasts are given by

$$Y_n(h) = \mu$$

where μ is estimated by the sample mean.

Value

An object of class "forecast".

The function summary is used to obtain and print a summary of the results, while the function plot produces a plot of the forecasts and prediction intervals.

The generic accessor functions fitted.values and residuals extract useful features of the value returned by meanf.

An object of class "forecast" is a list containing at least the following elements:

model	A list containing information about the fitted model
method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The original time series (either object itself or the time series used to create the model stored as object).
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values (one-step forecasts)

Author(s)

Rob J Hyndman

See Also

rwf

Examples

```
nile.fcast <- meanf(Nile, h=10)
plot(nile.fcast)
```

monthdays *Number of days in each season*

Description

Returns number of days in each month or quarter of the observed time period.

Usage

```
monthdays(x)
```

Arguments

x	time series

Details

Useful for month length adjustments

Value

Time series

Author(s)

Rob J Hyndman

Examples

```
par(mfrow=c(2,1))
plot(ldeaths,xlab="Year",ylab="pounds",
    main="Monthly deaths from lung disease (UK)")
ldeaths.adj <- ldeaths/monthdays(ldeaths)*365.25/12
plot(ldeaths.adj,xlab="Year",ylab="pounds",
    main="Adjusted monthly deaths from lung disease (UK)")
```

na.interp *Interpolate missing values in a time series*

Description

Uses linear interpolation to replace missing values.

Usage

```
na.interp(x)
```

Arguments

x	time series

Details

A more general and flexible approach is available using `na.approx` in the `zoo` package.

Value

Time series

Author(s)

Rob J Hyndman

Examples

```
data(gold)
plot(na.interp(gold))
```

---

naive *Naive forecasts*

Description

`naive()` returns forecasts and prediction intervals for an ARIMA(0,1,0) random walk model applied to x. `snaive()` returns forecasts and prediction intervals from an ARIMA(0,0,0)(0,1,0)m model where m is the seasonal period.

Usage

```
naive(x, h=10, level=c(80,95), fan=FALSE)
snaive(x, h=2*frequency(x), level=c(80,95), fan=FALSE)
```

Arguments

x	a numeric vector or time series
h	Number of periods for forecasting
level	Confidence levels for prediction intervals.
fan	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.

Details

These functions are simply convenient wrappers to `Arima` with the appropriate arguments to return naive and seasonal naive forecasts.

Value

An object of class "forecast".

The function summary is used to obtain and print a summary of the results, while the function plot produces a plot of the forecasts and prediction intervals.

The generic accessor functions fitted.values and residuals extract useful features of the value returned by naive or snaive.

An object of class "forecast" is a list containing at least the following elements:

model	A list containing information about the fitted model
method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The original time series (either object itself or the time series used to create the model stored as object).
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values (one-step forecasts)

Author(s)

Rob J Hyndman

See Also

Arima, rwf

Examples

```
plot(naive(gold,h=50),include=200)
plot(snaive(wineind))
```

---

ndiffs *Number of differences*

Description

Uses a unit root test to determine the number of differences required for time series x. If test="kpss", the KPSS test is used with the null hypothesis that x has a stationary root against a unit-root alternative. Then the test returns the least number of differences required to pass the test at the level alpha. If test="adf", the Augmented Dickey-Fuller test is used and if test="pp" the Phillips-Perron test is used. In both of these cases, the null hypothesis is that x has a unit root against a stationary root alternative. Then the test returns the least number of differences required to fail the test at the level alpha.

Usage

```
ndiffs(x, alpha=0.05, test=c("kpss","adf", "pp"))
```

Arguments

x	A univariate time series
alpha	Level of the test
test	Type of unit root test to use

Value

An integer.

Author(s)

Rob J Hyndman

Examples

```
ndiffs(WWWusage)
```

---

plot.ets *Plot components from ETS model*

Description

Produces a plot of the level, slope and seasonal components from an ETS model.

Usage

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

Arguments

x	Object of class "ets".
...	Other plotting parameters passed to par.

Value

None. Function produces a plot

Author(s)

Rob J Hyndman

See Also

ets

Examples

```
fit <- ets(USAccDeaths)
plot(fit)
plot(fit,plot.type="single",ylab="",col=1:3)
```

---

plot.forecast *Forecast plot*

Description

Plots a time series with forecasts and prediction intervals.

Usage

```
## S3 method for class 'forecast'
plot(x, include, plot.conf = TRUE, shaded = TRUE, shadebars=(length(x$mean)<5),
                 shadecols=NULL, lambda = NULL, col = 1, fcol = 4, pi.col=1, pi.lty
                 ylim = NULL, main = NULL, ylab = "", xlab = "", ...)

## S3 method for class 'splineforecast'
plot(x,fitcol=2,...)
```

Arguments

x	Forecast object produced by forecast.
include	number of values from time series to include in plot
plot.conf	Logical flag indicating whether to plot prediction intervals.
shaded	Logical flag indicating whether prediction intervals should be shaded (TRUE) or lines (FALSE)
shadebars	Logical flag indicating if prediction intervals should be plotted as shaded bars (if TRUE) or a shaded polygon (if FALSE). Ignored if shaded=FALSE. Bars are plotted by default if there are fewer than five forecast horizons.
shadecols	Colors for shaded prediction intervals
lambda	Box-Cox parameter. If present, function backtransforms data and forecasts using InvBoxCox.
col	the colour for the data line.
fcol	the colour for the forecast line.
pi.col	If shade=FALSE and plot.conf=TRUE, the prediction intervals are plotted in this colour.

pi.lty	If shade=FALSE and plot.conf=TRUE, the prediction intervals are plotted using this line type.
ylim	Limits on y-axis
main	Main title
ylab	Y-axis label
xlab	X-axis label
fitcol	Line colour for fitted values.
...	additional arguments to plot.

Value

None.

Author(s)

Rob J Hyndman

References

Makridakis, Wheelwright and Hyndman (1998) *Forecasting: methods and applications*, Wiley: New York. http://robjhyndman.com/forecasting.

See Also

plot.ts

Examples

```
deaths.fit <- hw(USAccDeaths,h=48)
plot(deaths.fit)
```

---

rwf *Random Walk Forecast*

Description

Returns forecasts and prediction intervals for a random walk with drift model applied to x.

Usage

```
rwf(x, h=10, drift=FALSE, level=c(80,95), fan=FALSE)
```

Arguments

x	a numeric vector or time series
h	Number of periods for forecasting
drift	Logical flag. If TRUE, fits a random walk with drift model.
level	Confidence levels for prediction intervals.
fan	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.

Details

The random walk with drift model is

$$Y_t = c + Y_{t-1} + Z_t$$

where Z_t is a normal iid error. Forecasts are given by

$$Y_n(h) = ch + Y_n$$

. If there is no drift, the drift parameter c=0. Forecast standard errors allow for uncertainty in estimating the drift parameter.

Value

An object of class "forecast".

The function summary is used to obtain and print a summary of the results, while the function plot produces a plot of the forecasts and prediction intervals.

The generic accessor functions fitted.values and residuals extract useful features of the value returned by rwf.

An object of class "forecast" is a list containing at least the following elements:

model	A list containing information about the fitted model
method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The original time series (either object itself or the time series used to create the model stored as object).
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values (one-step forecasts)

Author(s)

Rob J Hyndman

See Also

arima, meanf

Examples

```
gold.fcast <- rwf(gold[1:60],h=50)
plot(gold.fcast)
```

`seasadj` *Seasonal adjustment*

Description

Returns seasonally adjusted data constructed by removing the seasonal component.

Usage

```
seasadj(object)
```

Arguments

`object`	Object created by `decompose` or `stl`.

Value

Univariate time series.

Author(s)

Rob J Hyndman

See Also

stl, decompose

Examples

```
plot(AirPassengers)
lines(seasadj(decompose(AirPassengers,"multiplicative")),col=4)
```

`seasonaldummy` *Seasonal dummy variables*

Description

`seasonaldummy` and `seasonaldummyf` return matrices of dummy variables suitable for use in `arima`, `lm` or `tslm`. The last season is omitted and used as the control.

`fourier` and `fourierf` return matrices containing terms from a Fourier series, up to order `K`, suitable for use in `arima`, `lm` or `tslm`.

Usage

```
seasonaldummy(x)
seasonaldummyf(x,h)
fourier(x,K)
fourierf(x,K,h)
```

Arguments

`x`	Seasonal time series
`h`	Number of periods ahead to forecast
`K`	Maximum order of Fourier terms

Value

Numerical matrix with number of rows equal to the `length(x)` and number of columns equal to `frequency(x)-1` (for `seasonaldummy` and `seasonaldummyf` or `2*K` (for `fourier` or `fourierf`).

Author(s)

Rob J Hyndman

Examples

```
plot(ldeaths)

# Using seasonal dummy variables
month <- seasonaldummy(ldeaths)
deaths.lm  <- tslm(ldeaths ~ month)
tsdisplay(residuals(deaths.lm))
ldeaths.fcast <- forecast(deaths.lm, data.frame(month=I(seasonaldummyf(ldeaths,36))))
plot(ldeaths.fcast)

# A simpler approach to seasonal dummy variables
deaths.lm  <- tslm(ldeaths ~ season)
ldeaths.fcast <- forecast(deaths.lm, h=36)
plot(ldeaths.fcast)
```

```
# Using Fourier series
X <- fourier(ldeaths,3)
deaths.lm  <- tslm(ldeaths ~ X)
ldeaths.fcast <- forecast(deaths.lm, data.frame(X=I(fourierf(ldeaths,3,36))))
plot(ldeaths.fcast)
```

`seasonplot` *Seasonal plot*

Description

Plots a seasonal plot as described in Makridakis, Wheelwright and Hyndman (1998, chapter 2).

Usage

```
seasonplot(x, s, season.labels = NULL, year.labels = FALSE,
                 year.labels.left = FALSE, type = "o", main, ylab = "",
                 xlab = NULL, col = 1, ...)
```

Arguments

`x`	a numeric vector or time series.
`s`	seasonal frequency of x
`season.labels`	Labels for each season in the "year"
`year.labels`	Logical flag indicating whether labels for each year of data should be plotted on the right.
`year.labels.left`	Logical flag indicating whether labels for each year of data should be plotted on the left.
`type`	plot type (as for `plot`)
`main`	Main title.
`ylab`	Y-axis label
`xlab`	X-axis label
`col`	Colour
`...`	additional arguments to `plot`.

Value

None.

Author(s)

Rob J Hyndman

References

Makridakis, Wheelwright and Hyndman (1998) *Forecasting: methods and applications*, Wiley: New York. http://robjhyndman.com/forecasting/

See Also

monthplot

Examples

```
seasonplot(AirPassengers)
```

---

ses *Exponential smoothing forecasts*

Description

Returns forecasts and other information for exponential smoothing forecasts applied to x.

Usage

```
ses(x, h=10, level=c(80,95), fan=FALSE, ...)

holt(x, h=10, damped=FALSE, level=c(80,95), fan=FALSE, ...)

hw(x, h=2*frequency(x), seasonal="additive", damped=FALSE, level=c(80,95), fan=FALS
```

Arguments

x	a numeric vector or time series
h	Number of periods for forecasting.
damped	If TRUE, use a damped trend.
seasonal	Type of seasonality in hw model. "additive" or "multiplicative"
level	Confidence level for prediction intervals.
fan	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.
...	Other arguments passed to ets.

Details

ses, holt and hw are simply convenient wrapper functions for forecast(ets(...)).

Value

An object of class "forecast".

The function summary is used to obtain and print a summary of the results, while the function plot produces a plot of the forecasts and prediction intervals.

The generic accessor functions fitted.values and residuals extract useful features of the value returned by ets and associated functions.

An object of class "forecast" is a list containing at least the following elements:

model	A list containing information about the fitted model
method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The original time series (either object itself or the time series used to create the model stored as object).
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values (one-step forecasts)

Author(s)

Rob J Hyndman

References

Hyndman, R.J., Koehler, A.B., Snyder, R.D., Grose, S. (2002) "A state space framework for automatic forecasting using exponential smoothing methods", *International J. Forecasting*, **18**(3), 439–454.

Hyndman, R.J., Akram, Md., and Archibald, B. (2008) "The admissible parameter space for exponential smoothing models". *Annals of Statistical Mathematics*, **60**(2), 407–426.

See Also

ets, HoltWinters, rwf, arima.

Examples

```
fcast <- holt(airmiles)
plot(fcast)
deaths.fcast <- hw(USAccDeaths,h=48)
plot(deaths.fcast)
```

simulate.ets *Simulation from a time series model*

Description

Returns a time series based on the model object object.

Usage

```
## S3 method for class 'ets'
simulate(object, nsim=length(object$x), seed=NULL, future=TRUE, bootstrap=FALSE, ..
## S3 method for class 'ar'
simulate(object, nsim=object$n.used, seed=NULL, future=TRUE, bootstrap=FALSE, ...)
## S3 method for class 'Arima'
simulate(object, nsim=length(object$x), seed=NULL, xreg=NULL, future=TRUE, bootstra
## S3 method for class 'fracdiff'
simulate(object, nsim=object$n, seed=NULL, future=TRUE, bootstrap=FALSE, ...)
```

Arguments

object	An object of class "ets", "Arima" or "ar".
nsim	Number of periods for the simulated series
seed	Either NULL or an integer that will be used in a call to set.seed before simulating the time seriers. The default, NULL will not change the random generator state.
future	Produce sample paths that are future to and conditional on the data in object.
bootstrap	If TRUE, simulation uses resampled errors rather than normally distributed errors.
xreg	New values of xreg to be used for forecasting. Must have nsim rows.
...	Other arguments.

Value

An object of class "ts".

Author(s)

Rob J Hyndman

See Also

ets, Arima, auto.arima, ar, arfima.

Examples

```
fit <- ets(USAccDeaths)
plot(USAccDeaths,xlim=c(1973,1982))
lines(simulate(fit, 36),col="red")
```

`sindexf` *Forecast seasonal index*

Description

Returns vector containing the seasonal index for `h` future periods. If the seasonal index is non-periodic, it uses the last values of the index.

Usage

```
sindexf(object, h)
```

Arguments

`object`	Output from `decompose` or stl.
`h`	Number of periods ahead to forecast

Value

Time series

Author(s)

Rob J Hyndman

Examples

```
uk.stl <- stl(UKDriverDeaths,"periodic")
uk.sa <- seasadj(uk.stl)
uk.fcast <- holt(uk.sa,36)
seasf <- sindexf(uk.stl,36)
uk.fcast$mean <- uk.fcast$mean + seasf
uk.fcast$lower <- uk.fcast$lower + cbind(seasf,seasf)
uk.fcast$upper <- uk.fcast$upper + cbind(seasf,seasf)
uk.fcast$x <- UKDriverDeaths
plot(uk.fcast,main="Forecasts from Holt's method with seasonal adjustment")
```

splinef *Cubic Spline Forecast*

Description

Returns local linear forecasts and prediction intervals using cubic smoothing splines.

Usage

```
splinef(x, h=10, level=c(80,95), fan=FALSE)
```

Arguments

x	a numeric vector or time series
h	Number of periods for forecasting
level	Confidence level for prediction intervals.
fan	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.

Details

The cubic smoothing spline model is equivalent to an ARIMA(0,2,2) model but with a restricted parameter space. The advantage of the spline model over the full ARIMA model is that it provides a smooth historical trend as well as a linear forecast function. Hyndman, King, Pitrun, and Billah (2002) show that the forecast performance of the method is hardly affected by the restricted parameter space.

Value

An object of class "forecast".

The function summary is used to obtain and print a summary of the results, while the function plot produces a plot of the forecasts and prediction intervals.

The generic accessor functions fitted.values and residuals extract useful features of the value returned by meanf.

An object of class "forecast" is a list containing at least the following elements:

model	A list containing information about the fitted model
method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The original time series (either object itself or the time series used to create the model stored as object).
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values (one-step forecasts)

Author(s)

Rob J Hyndman

References

Hyndman, King, Pitrun and Billah (2005) Local linear forecasts using cubic smoothing splines. *Australian and New Zealand Journal of Statistics*, **47**(1), 87-99. http://robjhyndman.com/papers/splinefcast.htm.

See Also

smooth.spline, arima, holt.

Examples

```
fcast <- splinef(uspop,h=5)
plot(fcast)
summary(fcast)
```

---

thetaf *Theta method forecast*

Description

Returns forecasts and prediction intervals for a theta method forecast.

Usage

```
thetaf(x, h=10, level=c(80,95), fan=FALSE)
```

Arguments

x	a numeric vector or time series
h	Number of periods for forecasting
level	Confidence levels for prediction intervals.
fan	If TRUE, level is set to seq(50,99,by=1). This is suitable for fan plots.

Details

The theta method of Assimakopoulos and Nikolopoulos (2000) is equivalent to simple exponential smoothing with drift. This is demonstrated in Hyndman and Billah (2003). Prediction intervals are computed using the underlying state space model.

Value

An object of class "forecast".

The function summary is used to obtain and print a summary of the results, while the function plot produces a plot of the forecasts and prediction intervals.

The generic accessor functions fitted.values and residuals extract useful features of the value returned by rwf.

An object of class "forecast" is a list containing at least the following elements:

model	A list containing information about the fitted model
method	The name of the forecasting method as a character string
mean	Point forecasts as a time series
lower	Lower limits for prediction intervals
upper	Upper limits for prediction intervals
level	The confidence values associated with the prediction intervals
x	The original time series (either object itself or the time series used to create the model stored as object).
residuals	Residuals from the fitted model. That is x minus fitted values.
fitted	Fitted values (one-step forecasts)

Author(s)

Rob J Hyndman

References

Assimakopoulos, V. and Nikolopoulos, K. (2000). The theta model: a decomposition approach to forecasting. *International Journal of Forecasting* **16**, 521-530.

Hyndman, R.J., and Billah, B. (2003) Unmasking the Theta method. *International J. Forecasting*, **19**, 287-290.

See Also

arima, meanf, rwf, ses

Examples

```
nile.fcast <- thetaf(Nile)
plot(nile.fcast)
```

`tsdisplay` *Time series display*

Description

Plots a time series along with its acf and either its pacf, lagged scatterplot or spectrum.

Usage

```
tsdisplay(x, plot.type = "partial", points = TRUE, ci.type =
                "white", lag.max = round(10 * log10(length(x))),
                na.action = na.interp, main = NULL, ylab = "", xlab =
                "", pch = 1, cex = 0.5, ...)
```

Arguments

`x`	a numeric vector or time series.
`plot.type`	type of plot to include in lower right corner. Possible values are "partial", "scatter" or "spectrum".
`points`	logical flag indicating whether to show the individual points or not in the time plot.
`ci.type`	type of confidence limits for ACF. Possible values are as for `acf`.
`lag.max`	the maximum lag to plot for the acf and pacf.
`na.action`	how to handle missing values. Default is to use linear interpolation.
`main`	Main title.
`ylab`	Y-axis label
`xlab`	X-axis label
`pch`	Plotting character
`cex`	Character size
`...`	additional arguments to `acf`.

Value

None.

Author(s)

Rob J Hyndman

References

Makridakis, Wheelwright and Hyndman (1998) *Forecasting: methods and applications*, Wiley: New York. http://robjhyndman.com/forecasting/

See Also

plot.ts, acf

Examples

```
tsdisplay(diff(WWWusage))
```

---

tslm *Fit a linear model with time series components*

Description

tslm is used to fit linear models to time series including trend and seasonality components.

Usage

```
tslm(formula,data,...)
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.
...	Other arguments passed to lm().

Details

tslm is largely a wrapper for lm() except that it allows variables "trend" and "season" which are created on the fly from the time series characteristics of the data. The variable "trend" is a simple time trend and "season" is a factor indicating the season (e.g., the month or the quarter depending on the frequency of the data).

Value

Returns an object of class "lm".

Author(s)

Rob J Hyndman

See Also

forecast.lm, lm.

Examples

```
y <- ts(rnorm(120,0,3) + 1:120 + 20*sin(2*pi*(1:120)/12), frequency=12)
fit <- tslm(y ~ trend + season)
plot(forecast(fit, h=20))
```

---

wineind *Australian total wine sales*

Description

Australian total wine sales by wine makers in bottles <= 1 litre. Jan 1980 – Aug 1994.

Usage

```
wineind
```

Format

Time series data

Source

Time Series Data Library. http://robjhyndman.com/TSDL/

Examples

```
tsdisplay(wineind)
```

---

woolyrnq *Quarterly production of woollen yarn in Australia*

Description

Quarterly production of woollen yarn in Australia: tonnes. Mar 1965 – Sep 1994.

Usage

```
woolyrnq
```

Format

Time series data

Source

Time Series Data Library. http://robjhyndman.com/TSDL/

Examples

```
tsdisplay(woolyrnq)
```

Index