R topics documented:

tfarima-package .......................................................... 3
as.lagpol ................................................................. 4
as.um ................................................................. 4
autocorr ................................................................. 5
autocov ................................................................. 6
calendar ................................................................. 6
CalendarVar ............................................................... 8
ccf.tf ................................................................. 8
diagchk.tf ............................................................... 9
display ................................................................. 10
easter ................................................................. 11
fit.tf ................................................................. 12
ide ................................................................. 13
InterventionVar ........................................................... 14
inv ................................................................. 15
lagpol ................................................................. 16
logLik.um ............................................................... 17
modify.tf ............................................................... 17
nabla ................................................................. 18
noise ................................................................. 19
outlierDates ............................................................. 20
outliers.tf .............................................................. 20
output.tf .............................................................. 22
pcdf ................................................................. 22
phi ................................................................. 23
pi.weights .............................................................. 24
predict.tf .............................................................. 25
predict.um .............................................................. 26
printLagpol .............................................................. 27
printLagpolList .......................................................... 27
psi.weights .............................................................. 28
residuals.tf ............................................................ 28
residuals.um ............................................................ 29
roots ................................................................. 30
roots.lagpol .............................................................. 30
rsales ................................................................. 31
seasadj ................................................................. 32
seriesC ................................................................. 33
seriesJ ................................................................. 33
setinputs ............................................................... 34
signal ................................................................. 34
sim.tf ................................................................. 35
spec ................................................................. 36
std ................................................................. 36
summary.tf ............................................................. 37
summary.um ............................................................. 38
Description

The tfarima package provides classes and methods to build customized transfer function and ARIMA models with multiple operators and parameter restrictions. The package also includes functions for model identification, model estimation (exact or conditional maximum likelihood), model diagnostic checking, automatic outlier detection, calendar effects, forecasting and seasonal adjustment.

Author(s)

Jose Luis Gallego <jose.gallego@unican.es>

References


as.lagpol  \textit{Lag polynomial}

\textbf{Description}

as.lagpol converts a numeric vector \(c(1, -a_1, \ldots, -a_d)\) into a lag polynomial \((1 - a_1 B - \ldots - a_p B^p)\).

\textbf{Usage}

\texttt{as.lagpol(pol, p = 1)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{pol} \hspace{1em} \text{a numeric vector.}
  \item \texttt{p} \hspace{1em} \text{integer power.}
\end{itemize}

\textbf{Value}

An object of class \texttt{lagpol}.

\textbf{Examples}

\begin{verbatim}
as.lagpol(c(1, -0.8))
as.lagpol(c(1, 0, 0, 0, -0.8))
\end{verbatim}

\textbf{as.um} \hspace{1em} \textit{Convert arima into um.}

\textbf{Description}

as.um converts an object of class \texttt{arima} into an object of class \texttt{um}.

\textbf{Usage}

\texttt{as.um(arima)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{arima} \hspace{1em} \text{an object of class \texttt{arima}.}
\end{itemize}

\textbf{Value}

An object of class \texttt{um}.
autocorr

Examples

z <- AirPassengers
a <- arima(log(z), order = c(0,1,1),
seasonal = list(order = c(0,1,1), frequency = 12))
uml <- as.um(a)

autocorr
Theoretical simple/partial autocorrelations of an ARMA model

Description

autocorr computes the simple/partial autocorrelations of an ARMA model.

Usage

autocorr(um, ...)

## S3 method for class 'um'
autocorr(um, lag.max = 10, par = FALSE, ...)

Arguments

um an object of class um.
...
additional arguments.
lag.max maximum lag for autocovariances.
par logical. If TRUE partial autocorrelations are computed.

Value

A numeric vector.

Note

The I polynomial is ignored.

Examples

ar1 <- um(ar = "1-0.8B")
autocorr(ar1, lag.max = 13)
autocorr(ar1, lag.max = 13, par = TRUE)
autocov  
*Theoretical autocovariances of an ARMA model*

**Description**

autocov computes the autocovariances of an ARMA model.

**Usage**

```r
autocov(um, ...)  
## S3 method for class 'um'
autocov(um, lag.max = 10, ...)
```

**Arguments**

- `um`: an object of class `um`.
- `...`: additional arguments.
- `lag.max`: maximum lag for autocovariances.

**Value**

A numeric vector.

**Note**

The I polynomial is ignored.

**Examples**

```r
ar1 <- um(ar = "1-0.8B")
autocov(ar1, lag.max = 13)
```

---

calendar  
*Calendar effects*

**Description**

calendar extends the ARIMA model `um` by estimating a transfer function model with seven deterministic variables to capture the calendar variation in a monthly time series. Two equivalent representations are available: (1) D1, D2, ..., D7, (2) L, D1-D7, ..., D6-D7 where D1, D2, ..., D7 are deterministic variables representing the number of Mondays, Tuesdays, ..., Sundays, L = D1 + D2 + ... + D7 is the of the month. Optionally, a deterministic variable to estimate the Easter effect can also be included.
Usage

```r
calendar(um, ...) 
```

```r
## S3 method for class 'um'

calendar(
  um,
  z = NULL,
  form = c("dif", "td"),
  easter = FALSE,
  n.ahead = 0,
  p.value = 1,
  envir = NULL,
  ...
)
```

Arguments

- **um**: an object of class `um`
- **...**: additional arguments.
- **z**: a time series.
- **form**: representation for calendar effects: `form = td` form (1) above, `form = dif` form (1).
- **easter**: logical. If TRUE an Easter effect is also estimated.
- **n.ahead**: a positive integer to extend the sample period of the deterministic variables with `n.ahead` observations, which could be necessary to forecast the output.
- **p.value**: estimates with a p-value greater than `p.value` are omitted.
- **envir**: environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.

Value

An object of class "tfm".

References


Examples

```r
Y <- tfarima::rsales
um1 <- um(Y, i = list(1, c(1, 12)), ma = list(1, c(1, 12)), bc = TRUE)
tfm1 <- calendar(um1)
```
### CalendarVar

**Calendar variables**

#### Description

CalendarVar creates a set of deterministic variables to capture calendar effects.

#### Usage

```r
CalendarVar(x, form = c("dif", "lom", "td"), easter = FALSE, n.ahead = 0)
```

#### Arguments

- **x**: an object of class `ts` used to determine the sample period and frequency.
- **form**: a character indicating the set of calendar variables.
- **easter**: logical. If TRUE an additional deterministic variable is generated to capture Easter effects.
- **n.ahead**: number of additional observations to extend the sample period.

#### Value

A matrix of explanatory variables.

#### References


#### Examples

```r
Y <- rsales
X <- CalendarVar(Y, easter = TRUE)
```

### ccf.tfm

**Cross-correlation check**

#### Description

ccf displays ccf between prewhitened inputs and residuals.

#### Usage

```r
ccf.tfm(tfm, lag.max = NULL, method = c("exact", "cond"), envir = NULL, ...)
```
Arguments

- **tfm**: a `tfm` object.
- **lag.max**: number of lags.
- **method**: Exact/conditional residuals.
- **envir**: environment in which the function arguments are evaluated. If `NULL` the calling environment of this function will be used.
- **...**: additional arguments.

Description

diagchk displays tools for diagnostic checking.

Usage

```r
# S3 method for class 'tfm'
diagchk(
  mdl,
  y = NULL,
  method = c("exact", "cond"),
  lag.max = NULL,
  lags.at = NULL,
  freq.at = NULL,
  std = TRUE,
  envir = NULL,
  ...
)
```

```r
diagchk(mdl, ...)
```

```r
# S3 method for class 'um'
diagchk(
  mdl,
  z = NULL,
  method = c("exact", "cond"),
  lag.max = NULL,
  lags.at = NULL,
  freq.at = NULL,
  std = TRUE,
  envir = NULL,
  ...
)
```
Arguments

md1: an object of class um.
y: an object of class ts.
method: exact or conditional residuals.
lag.max: number of lags for ACF/PACF.
lags.at: the lags of the ACF/PACF at which tick-marks are to be drawn.
freq.at: the frequencies of the (cum) periodogram at which tick-marks are to be drawn.
std: logical. If TRUE standardized residuals are shown.
envir: environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
...: additional arguments.
z: optional, an object of class ts.

Examples

z <- AirPassengers
airl <- um(z, i = list(1, c(1,12)), ma = list(1, c(1,12)), bc = TRUE)
diagchk(airl)

Description

display shows graphs characterizing one or a list of ARMA models.

Usage

display(um, ...)

## S3 method for class 'um'
display(
  um,
  lag.max = 25,
  n.freq = 501,
  log.spec = FALSE,
  graphs = c("acf", "pacf", "spec"),
  byrow = FALSE,
  eq = TRUE,
  ...
)

## Default S3 method:
display(um, ...)

Graphs for ARMA models
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>um</code></td>
<td>an object of class <code>um</code> or a list of these objects.</td>
</tr>
<tr>
<td><code>...</code></td>
<td>additional arguments.</td>
</tr>
<tr>
<td><code>lag.max</code></td>
<td>number of lags for ACF/PACF.</td>
</tr>
<tr>
<td><code>n.freq</code></td>
<td>number of frequencies for the spectrum.</td>
</tr>
<tr>
<td><code>log.spec</code></td>
<td>logical. If TRUE log spectrum is computed.</td>
</tr>
<tr>
<td><code>graphs</code></td>
<td>vector of graphs.</td>
</tr>
<tr>
<td><code>byrow</code></td>
<td>orientation of the graphs.</td>
</tr>
<tr>
<td><code>eq</code></td>
<td>logical. If TRUE the model equation is used as title.</td>
</tr>
</tbody>
</table>

Examples

```r
um1 <- um(ar = "(1 - 0.8B)(1 - 0.8B^12)")
um2 <- um(ma = "(1 - 0.8B)(1 - 0.8B^12)"
display(list(um1, um2))
```

---

Description

*easter* extends the ARIMA model `um` by including a regression variable to capture the Easter effect.

Usage

```r
easter(um, ...)
```

## S3 method for class 'um'
easter(um, z = NULL, n.ahead = 0, envir = NULL, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>um</code></td>
<td>an object of class <code>um</code>.</td>
</tr>
<tr>
<td><code>...</code></td>
<td>additional arguments.</td>
</tr>
<tr>
<td><code>z</code></td>
<td>a time series.</td>
</tr>
<tr>
<td><code>n.ahead</code></td>
<td>a positive integer to extend the sample period of the Easter regression variable with <code>n.ahead</code> observations, which could be necessary to forecast the output.</td>
</tr>
<tr>
<td><code>envir</code></td>
<td>environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.</td>
</tr>
</tbody>
</table>

Value

An object of class "tfm".
Examples

Y <- rsales
um1 <- um(Y, i = list(1, c(1, 12)), ma = list(1, c(1, 12)), bc = TRUE)
tfm1 <- easter(um1)

Description

fit fits the univariate model to the time series z.

Usage

## S3 method for class 'tfm'
fit(
  mdl,
  y = NULL,
  method = c("exact", "cond"),
  optim.method = "BFGS",
  show.iter = FALSE,
  fit.noise = TRUE,
  envir = NULL,
  ...
)
fit(mdl, ...)

## S3 method for class 'um'
fit(
  mdl,
  z = NULL,
  method = c("exact", "cond"),
  optim.method = "BFGS",
  show.iter = FALSE,
  envir = NULL,
  ...
)

Arguments

mdl an object of class um or tfm.
y a ts object.
method Exact/conditional maximum likelihood.
optim.method the method argument of the optim function.
show.iter logical value to show or hide the estimates at the different iterations.
ide

fit.noise logical. If TRUE parameters of the noise model are fixed.
envir environment in which the function arguments are evaluated. If NULL the calling
environment of this function will be used.
... additional arguments.
z a time series.

Value

A tfm object.
An object of class "um" with the estimated parameters.

Note

The um function estimates the corresponding ARIMA model when a time series is provided. The
fit function is useful to fit a model to several time series, for example, in a Monte Carlo study.

Examples

z <- AirPassengers
airl <- um(i = list(1, c(1, 12)), ma = list(1, c(1, 12)), bc = TRUE)
airl <- fit(airl, z)

ide Identification plots

Description

ide displays graphs useful to identify a tentative ARIMA model for a time series.

Usage

ide(
  Y, 
  transf = list(),
  order.polreg = 0,
  lag.max = NULL,
  lags.at = NULL,
  freq.at = NULL,
  wn.bands = TRUE,
  graphs = c("plot", "acf", "pacf"),
  set.layout = TRUE,
  byrow = TRUE,
  main = "",
  envir = NULL,
  ... 
)
Arguments

\textbf{Y} \quad \text{Univariate or multivariate time series.}
\textbf{transf} \quad \text{Data transformations, list(bc = F, d = 0, D = 0, S = F), where bc is the Box-Cox logarithmic transformation, d and D are the number of nonseasonal and seasonal differences, and S is the annual sum operator.}
\textbf{order.polreg} \quad \text{an integer indicating the order of a polynomial trend.}
\textbf{lag.max} \quad \text{number of autocorrelations.}
\textbf{lags.at} \quad \text{the lags of the ACF/PACF at which tick-marks are to be drawn.}
\textbf{freq.at} \quad \text{the frequencies of the (cum) periodogram at at which tick-marks are to be drawn.}
\textbf{wn.bands} \quad \text{logical. If TRUE confidence intervals for sample autocorrelations are computed assuming a white noise series.}
\textbf{graphs} \quad \text{graphs to be shown: plot, hist, acf, pacf, pgram, cpgram (cumulative periodogram), rm (range-median).}
\textbf{set.layout} \quad \text{logical. If TRUE the layout is set by the function, otherwise it is set by the user.}
\textbf{byrow} \quad \text{logical. If TRUE the layout is filled by rows, otherwise it is filled by columns.}
\textbf{main} \quad \text{title of the graph.}
\textbf{envir} \quad \text{environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.}
\textbf{...} \quad \text{additional arguments.}

Examples

Y <- AirPassengers
ide(Y, graphs = c("plot", "rm"))
ide(Y, transf = list(list(bc = TRUE, S = TRUE), list(bc = TRUE, d = 1, D = 1)))

\underline{InterventionVar} \quad \textit{Intervention variables}

Description

\texttt{InterventionVar} creates an intervention variable to capture the effect of an external event.

Usage

\texttt{InterventionVar(Y, date, type = c("P", "S", "R"), n.ahead = 0)}

Arguments

\textbf{Y} \quad \text{an object of class \texttt{ts} used to determine the sample period and frequency.}
\textbf{date} \quad \text{the date of the event, \texttt{c(year, month)}.}
\textbf{type} \quad \text{a character indicating the type of intervention variables: (P) pulse, (S) step, (R).}
\textbf{n.ahead} \quad \text{number of additional observations to extend the sample period.}
An intervention variable, a 'ts' object.


Y <- seriesJ$Y
P58 <- InterventionVar(Y, date = 58, type = "P")

inv
Inverse of a lag polynomial

inv inverts a lag polynomial until the indicated lag.

## S3 method for class 'lagpol'
inv(lp, lag.max = 10, ...)

lp an object of class lagpol.

... additional arguments.

lag.max largest order of the inverse lag polynomial.

inv returns a numeric vector with the coefficients of the inverse lag polynomial truncated at lag.max.

inv(as.lagpol(c(1, 1.2, -0.8)))
lagpol  

*Description*

lagpol creates a lag polynomial of the form \((1 - coef_1 B^s - \ldots - coef_d B^{s+d})^p\). This class of lag polynomials is defined by a vector of d coefficients \(c(coef_1, \ldots, coef_d)\), the powers \(s\) and \(p\), and a vector of \(k\) parameters \(c(param_1, \ldots, param_k)\). The vector \(c(coef_1, \ldots, coef_d)\) is actually a vector of math expressions to compute the value of each coefficient in terms of the parameters.

*Usage*

```r
lagpol(param = NULL, s = 1, p = 1, lags = NULL, coef = NULL)
```

*Arguments*

- `param` a vector/list of named parameters.
- `s` the seasonal period, integer.
- `p` the power of lag polynomial, integer.
- `lags` a vector of lags for sparse polynomials.
- `coef` a vector of math expressions.

*Value*

lagpol returns an object of class "lagpol" with the following components:

- **coef** Vector of coefficients \(c(coef_1, \ldots, coef_p)\) provided to create the lag polynomial.
- **pol** Base lag polynomial, \(c(1, -coef_1, \ldots, -coef_d)\).
- **Pol** Power lag polynomial when \(p > 1\).

*Examples*

```r
lagpol(param = c(phi = 0.8) )
lagpol(param = c(phi1 = 1.2, phi2 = -0.6), s = 4)
lagpol(param = c(delta = 1), p = 2)
```
logLik.um

Log-likelihood of an ARIMA model

Description

logLik computes the exact or conditional log-likelihood of object of the class um.

Usage

## S3 method for class 'um'
logLik(object, z = NULL, method = c("exact", "cond"), ...)

Arguments

object an object of class um.
z an object of class ts.
method exact or conditional.
... additional arguments.

Value

The exact or conditional log-likelihood.

modify.tfm

Modifying a TF or an ARIMA model

Description

modify modifies an object of class um or tfm by adding and/or removing lag polynomials.

Usage

## S3 method for class 'tfm'
modify(mdl, ...)

modify(mdl, ...)

## S3 method for class 'um'
modify(
  mdl,
  ar = NULL,
  i = NULL,
  ma = NULL,
  mu = NULL,
  sig2 = NULL,
  ...
bc = NULL,
fit = TRUE,
...)

Arguments

mdl an object of class um or tfm.
... additional arguments.
ar list of stationary AR lag polynomials.
i list of nonstationary AR (I) polynomials.
ma list of MA polynomials.
mu mean of the stationary time series.
sig2 variance of the error.
bc logical. If TRUE logs are taken.
fit logical. If TRUE, model is fitted.

Value

An object of class um or um.

Examples

um1 <- um(ar = "(1 - 0.8B)")
um2 <- modify(um1, ar = list(0, "(1 - 0.9B)"), ma = "(1 - 0.5B)")

nabla

Unscramble I polynomial

Description
	nabla multiplies the I polynomials of an object of the um class.

Usage

nabla(um)

## S3 method for class 'um'
nabla(um)

Arguments

um an object of class um.
noise

Value

A numeric vector \( c(a_1, \ldots, a_d) \)

Note

This function returns the member variable \( um.nabl.a \).

Examples

\[
\begin{align*}
&\text{um1 <- um}(i = "(1 - \text{B})(1 - \text{B}^12)") \\
&\text{nabl.a(um1)}
\end{align*}
\]

noise  Noise of a transfer function model

Description

\( \text{noise} \) computes the noise of a linear transfer function model.

Usage

\[
\text{noise}(\text{tfm}, \ldots)
\]

## S3 method for class 'tfm'
\[
\text{noise}(\text{tfm}, y = \text{NULL}, \text{diff} = \text{TRUE}, \text{exp} = \text{FALSE}, \text{envir} = \text{NULL}, \ldots)
\]

Arguments

- \( \text{tfm} \) an object of the class \( \text{tfm} \).
- \( \ldots \) additional arguments.
- \( y \) output of the TF model if it is different to that of the \( \text{tfm} \) object.
- \( \text{diff} \) logical. If \( \text{TRUE} \), the noise is differenced with the "i" operator of the univariate model of the noise.
- \( \text{exp} \) logical. If \( \text{TRUE} \), the antilog transformation is applied.
- \( \text{envir} \) environment in which the function arguments are evaluated. If \( \text{NULL} \) the calling environment of this function will be used.

Value

A "ts" object.
## Description

`outlierDates` shows the indeces and dates of outliers.

### Usage

```
outlierDates(x, c = 3)
```

### Arguments

- `x`: an `ts` object.
- `c`: critical value to determine whether or not an observation is an outlier.

### Value

A table with the indices, dates and z-scores of the outliers.

## Description

`outliers` performs a detection of four types of anomalies (AO, TC, LS and IO) in a time series described by an ARIMA model. If the dates of the outliers are unknown, an iterative detection process like that proposed by Chen and Liu (1993) is conducted.

### Usage

```
## S3 method for class 'tfm'
outliers(
  mdl, 
  y = NULL, 
  dates = NULL, 
  c = 3, 
  calendar = FALSE, 
  easter = FALSE, 
  resid = c("exact", "cond"), 
  n.ahead = NULL, 
  p.value = 1, 
  envir = NULL, 
  ...
)
```
outliers.mdl, ...)

## S3 method for class 'um'
outliers(
m1,
z = NULL,
dates = NULL,
c = 3,
calendar = FALSE,
easter = FALSE,
resid = c("exact", "cond"),
n.ahead = 0,
p.value = 1,
enir = NULL,
...
)

Arguments

mdl an object of class um or tfm.
y an object of class ts
dates a list of dates c(year, season). If dates = NULL, an iterative detection process is conducted.
c a positive constant to compare the z-ratio of the effect of an observation and decide whether or not it is an outlier. This argument is only used when dates = NULL.
calendar logical; if true, calendar effects are also estimated.
easter logical; if true, Easter effect is also estimated.
resid type of residuals (exact or conditional) used to identify outliers.
n.ahead a positive integer to extend the sample period of the intervention variables with n.ahead observations, which could be necessary to forecast the output.
p.value estimates with a p-value greater than p.value are omitted.
enir environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
... additional arguments.
z a time series.

Value

an object of class "tfm" or a table.

Examples

Y <- rsales
um1 <- um(Y, i = list(1, c(1, 12)), ma = list(1, c(1, 12)), bc = TRUE)
outliers(um1)
**output.tf**  
*Output of a transfer function*

**Description**
output filters the input using the transfer function.

**Usage**
`output.tf(tf)`

**Arguments**
- *tf*: an object of the S3 class "tf".

**Value**
A "ts" object

---

**pccf**  
*Prewhitened cross correlation function*

**Description**
*pccf* displays cross correlation function between input and output after prewhitening both through a univariate model.

**Usage**
`pccf(x, y, um.x = NULL, um.y = NULL, lag.max = NULL, plot = TRUE, envir = NULL, main = NULL, nu.weights = FALSE, ...)`
Arguments

x     input, a ‘ts’ object or a numeric vector.
y     output, a ‘ts’ object or a numeric vector.
um.x  univariate model for input.
num.y  univariate model for output.
lag.max number of lags, integer.
plot logical value to indicate if the ccf graph must be graphed or computed.
envir environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
main title of the graph.
nu.weights logical. If TRUE the coefficients of the IRF are computed instead of the cross-correlations.
... additional arguments.

Value

The estimated cross correlations are displayed in a graph or returned into a numeric vector.

---

phi

Unscramble AR polynomial

Description

phi multiplies the AR polynomials of an object of the um class.

Usage

phi(um)

## S3 method for class ‘um’
phi(um)

Arguments

um an object of class um.

Value

A numeric vector $c(1, a1, \ldots, ad)$

Note

This function returns the member variable um$phi.
Examples

```r
um1 <- um(ar = "(1 - 0.8B)(1 - 0.5B)")
phi(um1)
```

---

### Description

`pi.weights` computes the pi-weights of an AR(I)MA model.

### Usage

```r
pi.weights(um, ...)  
## S3 method for class 'um'
pi.weights(um, lag.max = 10, var.pi = FALSE, ...)
```

### Arguments

- `um`: an object of class `um`.
- `...`: additional arguments.
- `lag.max`: largest AR(Inf) coefficient required.
- `var.pi`: logical. If TRUE (FALSE), the I polynomials is considered (ignored).

### Value

A numeric vector.

### Examples

```r
um1 <- um(i = "(1 - B)(1 - B^12)", ma = "(1 - 0.8B)(1 - 0.8B^12)")
pi.weights(um1, var.pi = TRUE)
```
predict.tfm

Description

predict computes point and interval predictions for a time series based on a tfm object.

Usage

## S3 method for class 'tfm'
predict(
  object,
  newdata = NULL,
  y = NULL,
  ori = NULL,
  n.ahead = NULL,
  level = 0.95,
  i = NULL,
  envir = NULL,
  ...
)

Arguments

- **object**: an object of class um.
- **newdata**: new data for the predictors for the forecast period. This is a matrix if there is more than one predictor. The number of columns is equal to the number of predictors, the number of rows equal to n.ahead. If there is one predictor only the data may be provided alternatively as a vector.
- **y**: an object of class ts.
- **ori**: the origin of prediction. By default, it is the last observation.
- **n.ahead**: number of steps ahead.
- **level**: confidence level.
- **i**: transformation of the series y to be forecasted. It is a lagpol as those of a um object.
- **envir**: environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
- **...**: additional arguments.

Details

Forecasts for the inputs of a tfm object can be provided in three ways: (1) extending the time series with forecasts so that the length of the input is greater than the length of the output, (2) computed internally from the um object associated to the input and (3) with the newdata argument.
Forecasts from an ARIMA model

Description

predict computes point and interval predictions for a time series from models of class um.

Usage

```r
## S3 method for class 'um'
predict(
  object,
  z = NULL,
  ori = NULL,
  n.ahead = 1,
  level = 0.95,
  i = NULL,
  envir = NULL,
  ...
)
```

Arguments

- `object`: an object of class um.
- `z`: an object of class ts.
- `ori`: the origin of prediction. By default, it is the last observation.
- `n.ahead`: number of steps ahead.
- `level`: confidence level.
- `i`: transformation of the series z to be forecasted. It is a lagpol as those of a um object.
- `envir`: environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
- `...`: additional arguments.

Value

An object of class "tfm".

Examples

```r
Z <- AirPassengers
um1 <- um(Z, i = list(1, c(1, 12)), ma = list(1, c(1, 12)), bc = TRUE)
p <- predict(um1, n.ahead = 12)
p
plot(p, n.back = 60)
```
printLagpol

Print numeric vector as a lagpol object

Description

Print numeric vector as a lagpol object

Usage

printLagpol(pol, digits = 2)

Arguments

pol      numeric vectors with the coefficients of a normalized polynomial.
digits  number of decimals.

printLagpolList

Print a list of lagpol objects

Description

Print a list of lagpol objects

Usage

printLagpolList(llp, digits = 2)

Arguments

llp      a list of lagpol objects.
digits  number of decimals.
psi.weights  

**Psi weights of an AR(I)MA model**

**Description**

psi computes the psi-weights of an AR(I)MA model.

**Usage**

```r
psi.weights(um, ...)  
```

## S3 method for class 'um'  
```r
psi.weights(um, lag.max = 10, var.psi = FALSE, ...)  
```

**Arguments**

- **um**  
an object of class um.
- **...**  
additional arguments.
- **lag.max**  
Largest MA(Inf) coefficient required.
- **var.psi**  
logical. If TRUE the I polynomials is also inverted. If FALSE it is ignored.

**Value**

A numeric vector.

**Examples**

```r
um1 <- um(i = "(1 - B)(1 - B^12)", ma = "(1 - 0.8B)(1 - 0.8B^12)")  
psi.weights(um1)  
psi.weights(um1, var.psi = TRUE)  
```

residuals.tfm  

**Residuals of a transfer function model**

**Description**

residuals computes the exact or conditional residuals of a TF model.

**Usage**

## S3 method for class 'tfm'
```r
residuals(object, y = NULL, method = c("exact", "cond"), envir = NULL, ...)  
```
residuals.um

Arguments

- **object**: a tfm object.
- **y**: output of the TF model (if it is different to that of the "tfm" object).
- **method**: a character string specifying the method to compute the residuals, exact or conditional.
- **envir**: environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
- **...**: additional arguments.

Value

A "ts" object.

Description

residuals computes the exact or conditional residuals.

Usage

```r
## S3 method for class 'um'
residuals(object, z = NULL, method = c("exact", "cond"), envir = NULL, ...)
```

Arguments

- **object**: an object of class um.
- **z**: an object of class ts.
- **method**: exact/conditional residuals.
- **envir**: environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
- **...**: additional arguments.

Value

An object of class um.

Examples

```r
z <- AirPassengers
airl <- um(z, i = list(1, c(1, 12)), ma = list(1, c(1, 12)), bc = TRUE)
r <- residuals(airl)
summary(r)
```
roots 

Roots of the lag polynomials of an ARIMA model

Description

roots compute the roots of the AR, I, MA lag polynomials an ARIMA model.

Usage

roots(x, ...)

## S3 method for class 'um'
roots(x, opr = c("arma", "ar", "ma", "i", "arima"), ...)

Arguments

x an object of class um.

... additional arguments.

opr character that indicates which operators are selected.

Value

List of matrices with the roots of each single polynomial.

Examples

um1 <- um(ar = "(1 - 0.8B)(1 - 0.8B^12)"
roots(um1)

roots.lagpol 

Roots of a lag polynomial

Description

roots.lagpol computes the roots of a lag polynomial.

Usage

## S3 method for class 'lagpol'
roots(x, table = TRUE, ...)

## Default S3 method:
roots(x, ...)

Arguments

- **x**: an object of class `lagpol`.
- **table**: logical. If TRUE, it returns a five columns table showing the real and imaginary parts, the modulus, the frequency and the period of each root.
- **...**: additional arguments.

Value

A vector or a table.

Examples

```r
roots(c(1, 1.2, -0.8))
```

---

**rsales**

*Retail Sales of Variety Stores (U.S. Bureau of the Census)*

Description

156 monthly observations from January 1967 to December 1979.

Usage

`rsales`

Format

An object of class `ts` of length 156.

References

seasadj  

\textit{Seasonal adjustment}

\textbf{Description}

seasadj removes the seasonal component of time series.

\textbf{Usage}

\begin{verbatim}
seasadj(mdl, ...)  
## S3 method for class 'um'
seasadj(
  mdl,
  z = NULL,
  method = c("mixed", "forecast", "backcast"),
  envir = NULL,
  ...
)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{mdl}  
  an object of class \texttt{um} or \texttt{tfm}.
  \item \texttt{...}  
  additional arguments.
  \item \texttt{z}  
  an object of class \texttt{ts}.
  \item \texttt{method}  
  forward/backward forecasts or a mixture of the two.
  \item \texttt{envir}  
  environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
\end{itemize}

\textbf{Value}

seasadj returns a seasonal adjusted time series.

\textbf{Examples}

\begin{verbatim}
Y <- AirPassengers  
um1 <- um(Y, bc = TRUE, i = list(1, c(1,12)), ma = list(1, c(1,12)))  
Y <- seasadj(num1)  
ide(Y)
\end{verbatim}
## serieC

### Series C Chemical Process Temperature Readings: Every Minute.

**Description**

226 observations.

**Usage**

serieC

**Format**

An object of class numeric of length 226.

**References**


---

## serieJ

### Gas furnace data

**Description**

Sampling interval 9 seconds; observations for 296 pairs of data points.

**Usage**

serieJ

**Format**

A object of class data.frame with 296 rows and 2 columns:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.60-0.04 (input gas rate in cubic feet per minute.)</td>
<td>% CO2 in outlet gas.</td>
</tr>
</tbody>
</table>

**References**

setinputs

setinputs adds new inputs into a transfer function model.

Description

setinputs adds new inputs into a transfer function model.

Usage

setinputs(tfm, ...)

## S3 method for class 'tfm'
setinputs(tfm, xreg = NULL, inputs = NULL, ...)

Arguments

tfm a tfm object.
...
... further arguments to be passed to particular methods
xreg a matrix of inputs.
inputs a list of tf objects.

Value

A tfm object.

signal

Signal component of a TF model

Description

signal extracts the signal of a TF model.

Usage

signal(mdl, ...)

## S3 method for class 'tfm'
signal(mdl, y = NULL, diff = TRUE, envir = NULL, ...)
Arguments

mdl: an object of the class tfm.
...: additional arguments.
y: output of the TF model if it is different to that of the tfm object.
diff: logical. If TRUE, the noise is differenced with the "i" operator of the univariate model of the noise.
envir: environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.

Value

A "ts" object.

Description

sim generates a random time series from an object of class um or tfm.

Usage

## S3 method for class 'tfm'
sim(mdl, n = 100, y0 = NULL, seed = NULL, ...)
sim(mdl, ...)

## S3 method for class 'um'
sim(mdl, n = 100, z0 = NULL, seed = NULL, envir = NULL, ...)

Arguments

mdl: an object of class um or tfm.
n: number of observations.
y0: initial conditions for the nonstationary series.
seed: an integer.
...: additional arguments.
z0: initial conditions for the nonstationary series.
envir: environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.

Value

An object of class ts.
**spec**  
*Spectrum of an ARMA model*

**Description**

spec computes the spectrum of an ARMA model.

**Usage**

```r
spec(um, ...)  
## S3 method for class 'um'
spec(um, n.freq = 501, ...)
```

**Arguments**

- `um` an object of class `um`.
- `...` additional parameters.
- `n.freq` number of frequencies.

**Value**

A matrix with the frequencies and the power spectral densities.

**Note**

The I polynomial is ignored.

**Examples**

```r
um1 <- um(i = "(1 - B)(1 - B^12)", ma = "(1 - 0.8B)(1 - 0.8B^12)")  
s <- spec(um1, lag.max = 13)
```

---

**std**  
*Standardize time series*

**Description**

std standardizes a time series.

**Usage**

```r
std(x)
```
summary.tfm

Arguments

x

a ts object.

Value

The standardized time series.

summary.tfm

Summarizing Transfer Function models

Description

summary method for class "tfm".

Usage

## S3 method for class 'tfm'
summary(
  object,
  y = NULL,
  method = c("exact", "cond"),
  digits = max(3L,getOption("digits") - 3L),
  envir = NULL,
  ...
)

Arguments

object

a tfm object.

y

a "ts" object.

method

exact or conditional maximum likelihood.

digits

number of significant digits to use when printing.

envir

environment in which the function arguments are evaluated. If NULL the calling
environment of this function will be used.

... 

additional arguments.

Value

A tfm object.
Summary of um model

Description

summary prints a summary of the estimation and diagnosis.

Usage

```r
## S3 method for class 'um'
summary(
  object,
  z = NULL,
  method = c("exact", "cond"),
  digits = max(3L,getOption("digits") - 3L),
  envir = NULL,
  ...
)
```

Arguments

- **object**: an object of class `um`.
- **z**: an object of class `ts`.
- **method**: exact/conditional maximum likelihood.
- **digits**: number of significant digits to use when printing.
- **envir**: environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
- **...**: additional arguments.

Value

A list with the summary of the estimation and diagnosis.

Examples

```r
z <- AirPassengers
airl <- um(z, i = list(1, c(1,12)), ma = list(1, c(1,12)), bc = TRUE)
summary(airl)
```
sum_um

*Sum of univariate (ARIMA) models*

**Description**

`sum_um` creates a univariate (ARIMA) model from the sum of several univariate (arima) models.

**Usage**

```r
sum_um(...)  
```

**Arguments**

...  

List of "um" S3 objects.

**Value**

A "um" S3 object.

**Examples**

```r
um1 <- um(i = "(1 - B)", ma = "(1 - 0.8B)")
um2 <- um(i = "(1 - B12)", ma = "(1 - 0.8B^12)")
um3 <- sum_um(um1, um2)
```

tf

*Transfer function for input*

**Description**

`tf` creates a rational transfer function for an input, \( V(B) = w0(1 - w_1B - \ldots - w_qB^q)/(1-d_1B - \ldots - d_pB^p)B^dX_t \). Note that in this specification the constant term of the MA polynomial is factored out so that both polynomials in the numerator and denominator are normalized and can be specified with the `lagpol` function in the same way as the operators of univariate models.

**Usage**

```r
tf(  
  x = NULL,  
  delay = 0,  
  d0 = 0,  
  ar = NULL,  
  ma = NULL,  
  um = NULL,  
  n.back = NULL,  
  par.prefix = "",  
  envir = NULL  
)
```
`tfest` provides preestimates of the transfer function between an output and an input.

**Arguments**

- `x`: input, a ts object or a numeric vector.
- `delay`: integer.
- `w0`: constant term of the polynomial \(V(B)\), double.
- `ar`: list of stationary AR polynomials.
- `ma`: list of MA polynomials.
- `um`: univariate model for stochastic input.
- `n.back`: number of backcasts to extend the input.
- `par.prefix`: prefix name for parameters.
- `envir`: environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.

**Value**

An object of the class "tf".

**References**


**See Also**

`um`.

**Examples**

```r
x <- rep(0, 100)
x[50] <- 1
tfx <- tf(x, w0 = 0.8, ar = "(1 - 0.5B)(1 - 0.7B^12)")
```

---

**Description**

`tfest` provides preestimates of the transfer function between an output and an input.
Usage

\texttt{tfest(}
\begin{verbatim}
y, x, delay = 0, p = 1, q = 2, um.y = \text{NULL}, um.x = \text{NULL}, n.back = \text{NULL}, envir = \text{NULL}
\end{verbatim}
\texttt{)}

Arguments

\begin{description}
\item[y] output, a ts object or a numeric vector.
\item[x] input, a ts object or a numeric vector.
\item[delay] integer.
\item[p] order of the AR polynomial, double.
\item[q] order of the MA polynomial, double.
\item[um.y] univariate model for output, um object or \text{NULL}.
\item[um.x] univariate model for input, um object or \text{NULL}.
\item[n.back] number of backcasts.
\item[envir] environment in which the function arguments are evaluated.
\end{description}

Value

A "tf" S3 object

---

**tfm**

Transfer function models

Description

\texttt{tfm} creates a multiple input transfer function model.

Usage

\texttt{tfm(}
\begin{verbatim}
output = \text{NULL}, xreg = \text{NULL}, inputs = \text{NULL}, noise,
\end{verbatim}
\texttt{)}
Unscramble MA polynomial

Usage

theta(um)

## S3 method for class 'um'
theta(um)

Arguments

um an object of class um.
Value

A numeric vector $c(1,a_1,\ldots,a_d)$

Note

This function returns the member variable um$\theta$.

Examples

```r
um1 <- um(ma = "(1 - 0.8B)(1 - 0.5B)"
theta(um1)
```

---

**tsdiag.tfm**

*Diagnostic Plots for Time-Series Fits* Description

**Description**

`tsdiag.tfm` is a wrap of the `stats::tsdiag` function.

**Usage**

```r
## S3 method for class 'tfm'
 tsdiag(object, gof.lag = 10, ...)
```

**Arguments**

- `object` a fitted `um` object.
- `gof.lag` the maximum number of lags for a Portmanteau goodness-of-fit test
- `...` additional arguments.

**See Also**

`stats::tsdiag`. 
tsdiag.um

**Description**

tsdiaq.um is a wrap of the stats::tsdiag function.

**Usage**

```r
## S3 method for class 'um'
lsdiag(object, gof.lag = 10, ...)
```

**Arguments**

- **object**
  a fitted um object.
- **gof.lag**
  the maximum number of lags for a Portmanteau goodness-of-fit test
- **...**
  additional arguments.

**See Also**

stats::tsdiag.

ucomp.tfm

**Unobserved components**

**Description**

ucomp estimates the unobserved components of a time series (trend, seasonal, cycle, stationary and irregular) from the eventual forecast function.

**Usage**

```r
## S3 method for class 'tfm'
ucomp(
  mdl,
  y = NULL,
  method = c("mixed", "forecast", "backcast"),
  envir = envir,
  ...
)
```

```r
ucomp(mdl, ...)
```

```r
## S3 method for class 'um'
ucomp(
```
Arguments

mdl = NULL,  
y = NULL,  
method = c("mixed", "forecast", "backcast"),  
envir = NULL,  
...  

Value

A matrix with the unobserved components.

Examples

Z <- AirPassengers  
um1 <- um(Z, i = list(1, c(1, 12)), ma = list(1, c(1, 12)), bc = TRUE)  
uc <- ucomp(num1)

um  
Univariate (ARIMA) model

Description

um creates an S3 object representing a univariate ARIMA model, which can contain multiple AR, I  
and MA polynomials, as well as parameter restrictions.

Usage

um(  
z = NULL,  
ar = NULL,  
i = NULL,  
ma = NULL,  
mu = NULL,  
sig2 = 1,  
bc = FALSE,  
fit = TRUE,  
envir = NULL,  
...  
)

Arguments

z an object of class `ts`.

ar list of stationary AR lag polynomials.

i list of nonstationary AR (I) polynomials.

ma list of MA polynomials.

mu mean of the stationary time series.

sig2 variance of the error.

bc logical. If TRUE logs are taken.

fit logical. If TRUE, model is fitted.

envir environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.

... additional arguments.

Value

An object of class `um`.

References


Examples

```r
ar1 <- um(ar = "(1 - 0.8B)"
ar2 <- um(ar = "(1 - 1.4B + 0.8B^2)"
ma1 <- um(ma = "(1 - 0.8B)"
ma2 <- um(ma = "(1 - 1.4B + 0.8B^2)"
arma11 <- um(ar = "(1 - 1.4B + 0.8B^2", ma = "(1 - 0.8B)"
```

varsel

---

Variable selection

Description

`varsel` omits non-significant inputs from a transfer function model.

Usage

```r
varsel(tfm, ...)
```

```r
## S3 method for class 'tfm'
varsel(tfm, p.value = 0.1, ...)
```
Arguments

- `tfm` a `tfm` object.
- `...` additional arguments.
- `p.value` probability value to decide whether or not to omit an input.

Value

A `tfm` object or a "um" if no input is significant at that level.
Index

* datasets
  rsales, 31
  seriesC, 33
  seriesJ, 33

* package
  tfarima-package, 3

  as.lagpol, 4
  as.um, 4
  autocorr, 5
  autocov, 6

  calendar, 6
  CalendarVar, 8
  ccf.tfm, 8

  diagchk (diagchk.tfm), 9
  diagchk.tfm, 9
  display, 10

  easter, 11

  fit (fit.tfm), 12
  fit.tfm, 12

  ide, 13
  InterventionVar, 14
  inv, 15

  lagpol, 16
  logLlk.um, 17

  modify (modify.tfm), 17
  modify.tfm, 17

  nabla, 18
  noise, 19

  outlierDates, 20
  outliers (outliers.tfm), 20
  outliers.tfm, 20

  output.tf, 22

  pccf, 22
  phi, 23
  pi.weights, 24
  predict.tfm, 25
  predict.um, 26
  printLagpol, 27
  printLagpolList, 27
  psi.weights, 28

  residuals.tfm, 28
  residuals.um, 29
  roots, 30
  roots.default (roots.lagpol), 30
  roots.lagpol, 30
  rsales, 31

  seasadj, 32
  seriesC, 33
  seriesJ, 33
  setinputs, 34

  sim (sim.tfm), 35
  sim.tfm, 35
  spec, 36
  std, 36
  sum_um, 39
  summary.tfm, 37
  summary.um, 38

  tf, 39, 42
  tfarima (tfarima-package), 3
  tfarima-package, 3
  tfest, 40
  tfm, 7, 11, 12, 21, 26, 32, 41, 45
  theta, 42
  ts, 25, 26, 32, 45
  tsdiag.tfm, 43
  tsdiag.um, 44
INDEX

ucomp (ucomp.tfm), 44
ucomp.tfm, 44
um, 7, 11, 12, 21, 25, 26, 32, 40, 42, 45, 45

varsel, 46