Package ‘tfarima’

October 14, 2022

Type Package

Title Transfer Function and ARIMA Models

Version 0.3.2

Date 2022-05-20


Author Jose L. Gallego [aut, cre]

Maintainer Jose L. Gallego <jose.gallego@unican.es>

URL https://github.com/gallegoj/tfarima

License GPL-2

Imports Rcpp (>= 1.0.0), stats, numDeriv, zoo

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Depends R (>= 2.10)

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-05-20 10:50:02 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>tfarima-package</td>
<td>3</td>
</tr>
<tr>
<td>as.lagpol</td>
<td>4</td>
</tr>
<tr>
<td>as.um</td>
<td>5</td>
</tr>
<tr>
<td>autocorr</td>
<td>5</td>
</tr>
<tr>
<td>autocov.stsm</td>
<td>6</td>
</tr>
<tr>
<td>bsm</td>
<td>7</td>
</tr>
<tr>
<td>calendar.tfm</td>
<td>8</td>
</tr>
<tr>
<td>CalendarVar</td>
<td>10</td>
</tr>
<tr>
<td>ccf.tfm</td>
<td>11</td>
</tr>
<tr>
<td>coef.tfm</td>
<td>11</td>
</tr>
<tr>
<td>coef.um</td>
<td>12</td>
</tr>
<tr>
<td>diagchk.tfm</td>
<td>12</td>
</tr>
<tr>
<td>display</td>
<td>13</td>
</tr>
<tr>
<td>easter</td>
<td>14</td>
</tr>
<tr>
<td>fit.stsm</td>
<td>15</td>
</tr>
<tr>
<td>fit.tfm</td>
<td>16</td>
</tr>
<tr>
<td>fit2autocov</td>
<td>17</td>
</tr>
<tr>
<td>ide</td>
<td>18</td>
</tr>
<tr>
<td>intervention.tfm</td>
<td>19</td>
</tr>
<tr>
<td>InterventionVar</td>
<td>21</td>
</tr>
<tr>
<td>inv</td>
<td>21</td>
</tr>
<tr>
<td>lagpol</td>
<td>22</td>
</tr>
<tr>
<td>logLik.um</td>
<td>23</td>
</tr>
<tr>
<td>modify.tfm</td>
<td>24</td>
</tr>
<tr>
<td>nabla</td>
<td>25</td>
</tr>
<tr>
<td>noise</td>
<td>26</td>
</tr>
<tr>
<td>outlierDates</td>
<td>26</td>
</tr>
<tr>
<td>outliers.tfm</td>
<td>27</td>
</tr>
<tr>
<td>output.tf</td>
<td>29</td>
</tr>
<tr>
<td>pccf</td>
<td>29</td>
</tr>
<tr>
<td>phi</td>
<td>30</td>
</tr>
<tr>
<td>pi.weights</td>
<td>31</td>
</tr>
<tr>
<td>predict.tfm</td>
<td>32</td>
</tr>
<tr>
<td>predict.um</td>
<td>33</td>
</tr>
<tr>
<td>printLagpol</td>
<td>34</td>
</tr>
<tr>
<td>printLagpolList</td>
<td>34</td>
</tr>
<tr>
<td>psi.weights</td>
<td>35</td>
</tr>
<tr>
<td>residuals.tfm</td>
<td>35</td>
</tr>
<tr>
<td>residuals.um</td>
<td>36</td>
</tr>
<tr>
<td>rform</td>
<td>37</td>
</tr>
<tr>
<td>roots</td>
<td>37</td>
</tr>
<tr>
<td>roots.lagpol</td>
<td>38</td>
</tr>
<tr>
<td>rsales</td>
<td>39</td>
</tr>
<tr>
<td>S</td>
<td>39</td>
</tr>
<tr>
<td>sdummies</td>
<td>40</td>
</tr>
<tr>
<td>seasadj</td>
<td>40</td>
</tr>
</tbody>
</table>
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**

**Lag polynomial**

**Description**

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

**Usage**

as.lagpol(pol, p = 1)

**Arguments**

- **pol**: a numeric vector.
- **p**: integer power.

**Value**

An object of class lagpol.

**Examples**

as.lagpol(c(1, -0.8))
as.lagpol(c(1, 0, 0, 0, -0.8))
**as.um**

Convert arima into um.

### Description

as.um converts an object of class arima into an object of class um.

### Usage

as.um(arima)

### Arguments

- **arima**: an object of class arima.

### Value

An object of class um.

### Examples

```r
z <- AirPassengers
a <- arima(log(z), order = c(0,1,1),
seasonal = list(order = c(0,1,1), frequency = 12))
um1 <- 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.stsm | Theoretical autocovariances of an ARMA model

Description

`autocov` computes the autocovariances of an ARMA model.

Usage

```r
## S3 method for class 'stsm'
autocov(mdl, ...)
```

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

Arguments

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

Value

A numeric vector.

Note

The I polynomial is ignored.
Examples

# Local level model
b <- 1
C <- as.matrix(1)
stsm1 <- stsm(b = b, C = C, s2v = c(lvl = 1469.619), s2u = c(irr = 15103.061))
autocov(stsm1)

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

bsm

Basic Structural Time Series models

Description

bsm creates/estimates basic structural models for seasonal time series.

Usage

bsm(
  y,
  bc = FALSE,
  seas = c("hd", "ht", "hs"),
  s2v = c(lvl = 0.2, slp = 0.05, seas = 0.075),
  s2u = 0.1,
  xreg = NULL,
  fSv = NULL,
  ...
)

Arguments

y an object of class ts, with frequency 4 or 12.
bc logical. If TRUE logs are taken.
seas character, type of seasonality (Harvey-Durbin (hd), Harvey-Todd (ht), Harrison-
Steven (ht))

s2v variances of the error vector v_t.

s2u variance of the error u_t.

xreg matrix of regressors.

fSv function to create the covariance matrix of v_t.

Value

An object of class stsm.
References


Examples

bsm1 <- bsm(AirPassengers, bc = TRUE)

description

calendar extends the ARIMA model um by including a set of deterministic variables to capture the calendar variation in a monthly time series. Two equivalent representations are available: (i) D0, D1, ..., D6, (ii) L, D1-D0, ..., D6-D0 where D0, D2, ..., D6 are deterministic variables representing the number of Sundays, Mondays, ..., Saturdays, L = D0 + D1 + ... + D6 is the of the month. Alternatively, the Leap Year indicator (LPY) can be included instead of L. The seven trading days can also be compacted into two variables: week days and weekends. Optionally, a deterministic variable to estimate the Easter effect can also be included, see “easter”.

Usage

```r
## S3 method for class 'tfm'
calendar(
  md1,
  y = NULL,
  form = c("dif", "td", "td7", "td6", "wd"),
  ref = 0,
  lom = TRUE,
  lpyear = TRUE,
  easter = FALSE,
  len = 4,
  easter.mon = FALSE,
  n.ahead = 0,
  p.value = 1,
  envir = NULL,
  ...
)

calendar(md1, ...)

## S3 method for class 'um'
calendar(
  md1,
```
Arguments

mdl
an object of class \texttt{um} or \texttt{tfm}.

\texttt{y}
a time series.

\texttt{form}
representation for calendar effects: (1) \texttt{form = dif}, L, D1-D0, ..., D6-D0; (2) \texttt{form = td}, LPY, D1-D0, ..., D6-D0; (3) \texttt{form = td7}, D0, D2, ..., D6; (4) \texttt{form = td6}, D1, D2, ..., D6; (5) \texttt{form = wd}, (D1+...+D5) - 2(D6+D0)/5.

\texttt{ref}
a integer indicating the reference day. By default, \texttt{ref = 0}.

\texttt{lom, lpyear}
a logical value indicating whether or not to include the lom/lead year indicator.

\texttt{easter}
logical. If \texttt{TRUE} an Easter effect is also estimated.

\texttt{len}
the length of the Easter, integer.

\texttt{easter.mon}
logical. \texttt{TRUE} indicates that Easter Monday is a public holiday.

\texttt{n.ahead}
a positive integer to extend the sample period of the deterministic variables with \texttt{n.ahead} observations, which could be necessary to forecast the output.

\texttt{p.value}
estimates with a \texttt{p.value} greater than \texttt{p.value} are omitted.

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

\texttt{...}
other arguments.

Value

An object of class "\texttt{tfm}".

References


Examples

\texttt{Y <- tfarima::rsales}
\texttt{um1 <- um(Y, i = list(1, c(1, 12)), ma = list(1, c(1, 12)), bc = TRUE)}
\texttt{tfm1 <- calendar(um1)}
## Description

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

## Usage

```r
CalendarVar(
  x,
  form = c("dif", "td", "td7", "td6", "wd", "wd2", "null"),
  ref = 0,
  lom = TRUE,
  lpyear = TRUE,
  easter = FALSE,
  len = 4,
  easter.mon = FALSE,
  n.ahead = 0
)
```

## Arguments

- **x**: an object of class `ts` used to determine the sample period and frequency.
- **form**: a character indicated the set of calendar variables: td, td7, td6, wd.
- **ref**: a non-negative integer indicating the reference day.
- **lom**: logical. If TRUE length of the month effect is also estimated.
- **lpyear**: logical. If TRUE a leap year effect is also estimated.
- **easter**: logical. If TRUE an additional deterministic variable is generated to capture Easter effects.
- **len**: duration of the Easter, integer.
- **easter.mon**: logical. It is TRUE if Holy Monday is a public holiday.
- **n.ahead**: number of additional observations to extend the sample period.

## Value

An object of class `mts` or `ts`.

## References

Examples

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

tfm

Cross-correlation check

displays ccf between prewhitened inputs and residuals.

Usage

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

Arguments

tfm
lag.max
method
envir
... 

Coefficients of a transfer function model

directs the "coefficients" from a TF model.

Usage

## S3 method for class 'tfm'
coef(object, ...)

Arguments

object
... 

Value

A numeric vector.
### coef.um

**Coefficients of a univariate model**

**Description**

`coef` extracts the "coefficients" from a um object.

**Usage**

```r
## S3 method for class 'um'
coef(object, ...)
```

**Arguments**

- `object` a um object.
- `...` other arguments.

**Value**

A numeric vector.

### diagchk.tfm

**Diagnostic checking**

**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,
  ...
)

diagchk(mdl, ...)
```
## 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

- **mdl**: 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

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

### display

**Graphs for ARMA models**

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, ...)

Arguments

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

easter

Easter effect

Description

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

easter(um, ...)

## S3 method for class 'um'
easter(
    um,
    z = NULL,
    len = 4,
    easter.mon = FALSE,
    n.ahead = 0,
    envir = NULL,
    ...
)

Arguments

um an object of class um.
...
other arguments.
z a time series.
len a positive integer specifying the duration of the Easter.
easter.mon logical. If TRUE Easter Monday is also taken into account.
n.ahead a positive integer to extend the sample period of the Easter regression variable with n.ahead observations, which could be necessary to forecast the output.
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".

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 stsm to the time series y.

Usage

## S3 method for class 'stsm'
fit(mdl, method = "BFGS", show.iter = FALSE, ...)
Arguments

md1 an object of class \texttt{stsm}.
method argument of the \texttt{optim} function.
show.iter logical value to show or hide the estimates at the different iterations.
... other arguments.

Value

An object of class "\texttt{stsm}" with the estimated variances.

Examples

# Local level model
b <- 1
C <- as.matrix(1)
stsm1 <- stsm(Nile, b, C, s2v = c(lvl = 0.5), s2u = c(irr = 1), fit = FALSE)
stsm1 <- fit(stsm1, method = "L-BFGS-B")

fit.tfm \hspace{1cm} Estimation of the ARIMA model

Description

fit fits the univariate model to the time series \( z \).

Usage

\begin{verbatim}
## S3 method for class 'tfm'
fit(
  md1,
  y = NULL,
  method = c("exact", "cond"),
  optim.method = "BFGS",
  show.iter = FALSE,
  fit.noise = TRUE,
  envir = NULL,
  ...
)
\end{verbatim}

fit(md1, ...)

\begin{verbatim}
## S3 method for class 'um'
fit(
  md1,
  z = NULL,
  method = c("exact", "cond"),
\end{verbatim}
fit2autocov

```r
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.
- `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

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

**Description**

`fit2autocov` fits a STS model to a vector of theoretical autocovariances.
Usage

fit2autocov(mdl, ...)

## S3 method for class 'stsm'
fit2autocov(mdl, g, method = "BFGS", show.iter = FALSE, ...)

Arguments

mdl an object of class stsm.
...
other arguments.
g a vector of theoretical autocovariances (gamma[k], k= 0, ..., K).
method optimization method.
show.iter logical. If TRUE, estimates at each iteration are printed.

Value

An object of class stsm.

Examples

um1 <- um(Nile, i = 1, ma = 1)
g <- autocov(um1, lag.max = 1)
# Local level model
b <- 1
C <- as.matrix(1)
stsm1 <- stsm(Nile, b, C, s2v = c lvl = 0.5), s2u = c irr = 1), fit = FALSE)
stsm2 <- fit2autocov(stsm1, g)
stsm2

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

Y Univariate or multivariate time series.
transf 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.
order.polreg an integer indicating the order of a polynomial trend.
lag.max number of autocorrelations.
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 at which tick-marks are to be drawn.
wn.bands logical. If TRUE confidence intervals for sample autocorrelations are computed assuming a white noise series.
graphs graphs to be shown: plot, hist, acf, pacf, pgram, cpgram (cumulative periodogram), rm (range-median).
set.layout logical. If TRUE the layout is set by the function, otherwise it is set by the user.
byrow logical. If TRUE the layout is filled by rows, otherwise it is filled by columns.
main title of the graph.
envir environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
...
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)))

intervention.tfm Intervention analysis/Outlier treatment

Description

intervention estimates the effect of a intervention at a known time.
Usage

```r
## S3 method for class 'tfm'
intervention(
  mdl,
  y = NULL,
  type,
  time,
  n.ahead = 0,
  envir = parent.frame(),
  ...
)

intervention(mdl, ...)
```

```r
## S3 method for class 'um'
intervention(
  mdl,
  y = NULL,
  type,
  time,
  n.ahead = 0,
  envir = parent.frame(),
  ...
)
```

Arguments

- **mdl**: an object of class `um` or `tfm`.
- **y**: a "ts" object, optional.
- **type**: the type intervention (pulse, step, ramp) or the type of outlier (AO, LS, TC, IO).
- **time**: the date of the intervention, in format c(year, season).
- **n.ahead**: a positive integer to extend the sample period of the intervention variable with `n.ahead` observations, which could be necessary to forecast the output.
- **envir**: the environment in which to look for the time series z when it is passed as a character string.
- **...**: additional arguments.

Value

an object of class "tfm" or a table.
**InterventionVar**

**Description**

`InterventionVar` creates an intervention variable to capture the effect of an external event.

**Usage**

```r
InterventionVar(Y, date, type = c("P", "S", "R"), n.ahead = 0)
```

**Arguments**

- `Y`: an object of class `ts` used to determine the sample period and frequency.
- `date`: the date of the event, `c(year, month)`.
- `type`: a character indicating the type of intervention variables: (P) pulse, (S) step, (R).
- `n.ahead`: number of additional observations to extend the sample period.

**Value**

An intervention variable, a `ts` object.

**References**


**Examples**

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

---

**inv**

**Description**

`inv` inverts a lag polynomial until the indicated lag.
Usage

```r
inv(lp, ...)
```

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

Arguments

- **lp**: an object of class `lagpol`.
- **lag.max**: largest order of the inverse lag polynomial.
- **...**: additional arguments.

Value

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

Examples

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

---

**lagpol**  
Lag polynomials

Description

`lagpol` creates a lag polynomial of the form \((1 - \text{coef}_1 B^s - \ldots - \text{coef}_d B^{s_d})^p\). This class of lag polynomials is defined by a vector of \(d\) coefficients \(\text{c(coef}_1, ..., \text{coef}_d)\), the powers \(s\) and \(p\), and a vector of \(k\) parameters \(\text{c(param}_1, ..., \text{param}_k)\). The vector \(\text{c(coef}_1, ..., \text{coef}_d)\) 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, ..., coef_p) provided to create the lag polynomial.
- **pol**: Base lag polynomial, c(1, -coef_1, ..., -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)
```

Description

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

Usage

```r
## 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.
nabla

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.

Value

A numeric vector c(1, a1, ..., ad)

Note

This function returns the member variable um$nabla.

Examples

um1 <- um(i = "(1 - B)(1 - B^12)"

nabla(um1)
noise  

*Noise of a transfer function model*

**Description**

`noise` computes the noise of a linear transfer function model.

**Usage**

```r
noise(tfm, ...)

## S3 method for class 'tfm'
noise(tfm, y = NULL, diff = TRUE, exp = FALSE, envir = NULL, ...)
```

**Arguments**

- `tfm`  
  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.
- `exp`  
  logical. If TRUE, the antilog transformation is applied.
- `envir`  
  environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.

**Value**

A "ts" object.

---

outlierDates  

*Outlier dates*

**Description**

`outlierDates` shows the indices and dates of outliers.

**Usage**

```r
dates <- 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

```r
## S3 method for class 'tfm'
outliers(
  mdl,
  y = NULL,
  types = c("AO", "LS", "TC", "IO"),
  dates = NULL,
  c = 3,
  calendar = FALSE,
  easter = FALSE,
  resid = c("exact", "cond"),
  n.ahead = NULL,
  p.value = 1,
  tc.fix = TRUE,
  envir = NULL,
  ...
)

outliers(mdl, ...)

## S3 method for class 'um'
outliers(
  mdl,
  y = NULL,
  types = c("AO", "LS", "TC", "IO"),
  dates = NULL,
  c = 3,
  calendar = FALSE,
  easter = FALSE,
  resid = c("exact", "cond"),
  n.ahead = 0,
  p.value = 1,
  tc.fix = TRUE,
  envir = NULL,
  ...)
```
Arguments

mdl: an object of class um or tfm.
y: an object of class ts, optional.
types: a vector with the initials of the outliers to be detected, c("AO", "LS", "TC", "IO").
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.
tc.fix: a logical value indicating if the AR coefficient in the transfer function of the TC is estimated or fix.
envir: environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
...
other arguments.

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**

```r
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.
- `um.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.

---

\[ \text{phi} \quad \text{Unscramble AR polynomial} \]

Description

\text{phi} multiplies the AR polynomials of an object of the \text{um} class.

Usage

\[
\text{phi}(\text{um})
\]

## S3 method for class 'um'
\[
\text{phi}(\text{um})
\]

Arguments

- `um`: an object of class \text{um}.

Value

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

Note

This function returns the member variable \text{um$phi}.}
pi.weights

Examples

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

## S3 method for class 'um'
pi.weights(um1, 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**  
*Forecasting with transfer function models*

**Description**

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

**Usage**

```r
## 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.
predict.um

Forecasts from an ARIMA model

Description

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

Usage

## 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

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

psi.weights(um, ...)

## S3 method for class 'um'
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

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'
residuals(object, y = NULL, method = c("exact", "cond"), envir = NULL, ...)
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.

---

residuals.um

*Residuals of the ARIMA model*

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)
```
**rform**

Reduce form for STS model

**Description**

`rform` finds the reduce form for a STS model.

**Usage**

```r
rform(mdl, ...)
```

**Arguments**

- `mdl`: an object of class `stsm`.
- `...`: other arguments.

**Value**

An object of class `um`.

**Examples**

```r
b <- 1
C <- as.matrix(1)
stsm1 <- stsm(b = b, C = C, s2v = c(lvl = 1469.619), s2u = c(irr = 15103.061))
rf1 <- rform(stsm1)
nabla(rf1)
theta(rf1)
```

---

**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**

```r
roots(x, ...)
```

**Arguments**

- `x`: an object of class `um`.
- `...`: other arguments.

**Examples**

```r
b <- 1
C <- as.matrix(1)
stsm1 <- stsm(b = b, C = C, s2v = c(lvl = 1469.619), s2u = c(irr = 15103.061))
rf1 <- rform(stsm1)

roots(rf1, 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

roots(c(1, 1.2, -0.8))
rsales

| 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**


---

S

**Annual sum**

**Description**

S generates the annual sum of a monthly or quarterly time series.

**Usage**

S(x, extend = TRUE)

**Arguments**

x an ts object.
extend logical. If TRUE, the transformed series is extended with NA's to have the same length as the original series.

**Value**

The transformed time series, a ts object.
**sdummies**

*Seasonal dummies*

**Description**

sdummies creates an full set of seasonal dummies.

**Usage**

sdummies(Y, ref = 1, constant = FALSE, n.ahead = 0)

**Arguments**

- **Y**: an object of class ts used to determine the sample period and frequency.
- **ref**: the reference season, positive integer
- **constant**: logical indicator to include a column of ones.
- **n.ahead**: number of additional observations to extend the sample period.

**Value**

A matrix of trigonometric variables.

**Examples**

```r
Y <- AirPassengers
P58 <- sincos(Y)
```

---

**seasadj**

*Seasonal adjustment*

**Description**

seasadj removes the seasonal component of time series.

**Usage**

seasadj(mdl, ...)

```r
# S3 method for class 'um'
seasadj(
  mdl,
  z = NULL,
  method = c("mixed", "forecast", "backcast"),
  envir = NULL,
  ...
)
```
Arguments

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

Value

seasadj returns a seasonal adjusted time series.

Examples

Y <- AirPassengers
um1 <- um(Y, bc = TRUE, i = list(1, c(1,12)), ma = list(1, c(1,12)))
Y <- seasadj(um1)
ide(Y)

seriesC  Series C Chemical Process Temperature Readings: Every Minute.

Description

226 observations.

Usage

seriesC

Format

An object of class numeric of length 226.

References

seriesJ  

*Gas furnace data*

**Description**

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

**Usage**

`seriesJ`

**Format**

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

- **X**  0.60-0.04 (input gas rate in cubic feet per minute.)
- **Y**  % CO2 in outlet gas.

**References**


---

`setinputs.tfm`  

**Description**

`setinputs` adds new inputs into a transfer function model.

**Usage**

```r
## S3 method for class 'tfm'
setinputs(
  mdl,
  xreg = NULL,
  inputs = NULL,
  y = NULL,
  envir = parent.frame(),
  ...
)

setinputs(mdl, ...)
```

```r
## S3 method for class 'um'
setinputs(mdl, xreg = NULL, inputs = NULL, y = NULL, envir = NULL, ...)
```
Arguments

mdl a umm or tfm object.
xreg a matrix of inputs.
inputs a list of tf objects.
y an optional ts object.
envir an environment.
... other arguments.

Value

A tfm object.

Description

sform finds the structural form for an ARIMA model from its eventual forecast function.

Usage

sform(mdl, ...)

## S3 method for class 'umm'
sform(mdl, fSv = NULL, par = NULL, ...)

Arguments

mdl an object of class um.
... other arguments.
fSv optional function to create the covariance matrix.
par vector of parameters for function fSv.

Value

An object of class stsm

Examples

airl <- um(i = list(1, c(1, 12)), ma = "(1 - 0.86B)(1 - 0.8B12)"
sf <- sform(airl)
sf
signal

*Signal component of a TF model*

**Description**

signal extracts the signal of a TF model.

**Usage**

```r
signal(mdl, ...)
```

```r
## 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.

---

sim.tfm

*Time series simulation form an ARIMA or TF model*

**Description**

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

**Usage**

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

```r
sim(mdl, ...)
```

```r
## S3 method for class 'um'
sim(
```
sincos

Arguments

mdl, an object of class um or tfm.
n number of observations.
y0 initial conditions for the nonstationary series.
seed an integer.
... other arguments.
z0 initial conditions for the nonstationary series.
n0 remove the n0 first observation, integer.
a vector of innovations, optional.
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.

sincos Trigonometric variables

Description
sincos creates an full set of trigonometric variables.

Usage
sincos(Y, n.ahead = 0, constant = FALSE)

Arguments

Y an object of class ts used to determine the sample period and frequency.
n.ahead number of additional observations to extend the sample period.
constant logical indicator to include a column of ones.
spec

A matrix of trigonometric variables.

Examples

Y <- AirPassengers
P58 <- sincos(Y)

-----------------------------

spec                  Spectrum of an ARMA model
-----------------------------

Description

spec computes the spectrum of an ARMA model.

Usage

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

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**

`std(x)`

**Arguments**

- `x` a ts object.

**Value**

The standardized time series.

---

**stsm**

*Structural Time Series models*

**Description**

stsm creates an S3 object representing a time-invariant structural time series model.

**Usage**

`stsm(y, b, C, fSv, s2v, s2u = 1, xreg = NULL, bc = FALSE, fit = TRUE, ...)`

**Arguments**

- `y` an object of class ts.
- `b` vector of constants.
- `C` matrix of constants.
- `fSv` function to create the covariance matrix of v_t.
- `s2v` variances of the vector error v_t in the state equation.
- `s2u` variance of the error u_t in the observation equation.
- `xreg` matrix of regressors.
- `bc` logical. If TRUE logs are taken.
- `fit` logical. If TRUE, model is fitted.
- `...` other arguments.
Details

\[ y_t = b'x_t + u_t \] (observation equation), \[ x_t = Cx_{t-1} + v_t \] (state equation).

Value

An object of class \texttt{stsm}.

References


Examples

```r
# Local level model
b <- 1
C <- as.matrix(1)
stsm1 <- stsm(Nile, b, C, s2v = c(lvl = 0.5), s2u = c(irr = 1))
stsm1
```

---

\textbf{summary.tfm} \quad \textit{Summarizing Transfer Function models}

\textbf{Description}

Summary method for class "tfm".

\textbf{Usage}

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

\textbf{Arguments}

- \texttt{object} a \texttt{tfm} object.
- \texttt{y} a "ts" object.
- \texttt{method} exact or conditional maximum likelihood.
- \texttt{digits} number of significant digits to use when printing.
- \texttt{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.

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

sum_um(...)

Arguments

... List of "um" S3 objects.

Value

A "um" S3 object.

Examples

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, $V(B) = w_0(1 - w_1B - ... - w_qB^q)/(1-d_1B - ... - d_pB^p)B^dX_t$. Note that in this specification the constant term of the MA polynomial is factored out so that both polynomials in the numerator and denominator can be specified with the lagpol function in the same way as the operators of univariate models.

Usage

tf(  
x = NULL,  
delay = 0,  
w0 = 0,  
ar = NULL,  
ma = NULL,  
um = NULL,  
n.back = NULL,  
par.prefix = "",  
envir = NULL  )
tfest

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

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

tfest(
y, 
x, 
delay = 0,
p = 1,
q = 2, 
.um.y = NULL, 
.um.x = NULL, 
n.back = NULL, 
par.prefix = "", 
envir = NULL 
)

Arguments

y  output, a ts object or a numeric vector.
x  input, a ts object or a numeric vector.
delay integer.
p  order of the AR polynomial, integer
q  order of the MA polynomial, integer.
.um.y univariate model for output, um object or NULL.
.um.x univariate model for input, um object or NULL.
n.back number of backcasts.
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

A "tf" S3 object

---

tfm  Transfer function models

Description

tfm creates a multiple input transfer function model.
Usage

tfm(
  output = NULL,
  xreg = NULL,
  inputs = NULL,
  noise,
  fit = TRUE,
  envir = NULL,
  new.name = TRUE,
  ...
)

Arguments

  output a ts object or a numeric vector.
  xreg a matrix of regressors.
  inputs a list of tf objects.
  noise a um object for the noise.
  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.
  new.name logical. Argument used internally: if TRUE a new name is assigned to the
  output, otherwise it keeps its name saved in noise$z.
  ... additional arguments.

Value

  An object of the class tfm.

References


See Also

  tf and um.
theta

Unscramble MA polynomial

Description

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, a1, ..., ad)

Note

This function returns the member variable um$theta.

Examples

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

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

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.

diagnostics for time-series fits Description

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

Usage

## S3 method for class 'um'
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.

tsvalue   Value of a time series at a date

Description

tsvalue select a value from a time series by date.

Usage

tsvalue(x, date)
Arguments

- `x` an `ts` object.
- `date` the time of the specific observation, `c(year, month/quarter)`.

Value

The value of the observation, double.

---

**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 = NULL,
  ...
)

ucomp(mdl, ...)

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

Arguments

- `mdl` an object of class `um` or `tfm`.
- `y` an object of class `ts`.
- `method` forward/backward forecasts or a mixture of the two.
- `envir` environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
- `...` additional arguments.
- `z` an object of class `ts`. 
Value

A matrix with the unobserved components.

Examples

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

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

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

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`: the environment in which to look for the time series `z` when it is passed as a character string.
- `...`: 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)")
```

Description

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

Usage

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

## S3 method for class 'tfm'
varsel(tfm, y = NULL, p.value = 0.1, envir = NULL, ...)

Arguments

- `tfm` a tfm object.
- `...` other arguments.
- `y` a "ts" object.
- `p.value` probability value to decide whether or not to omit an input.
- `envir` environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.

Value

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

Monthly data from January 1951 to October 1966.

Usage

Wtelephone

Format

A object of class data.frame with 215 rows and 2 columns:

X  Monthly outward station movements.
Y  Monthly inward station movements.

Source

https://drive.google.com/file/d/1LP8aMIQewMrxgOlrg9rN3eWHHhZuJsY8K/view?usp=sharing

References

Index

* datasets
  - rsales, 39
  - seriesC, 41
  - seriesJ, 42
  - Wtelephone, 59

* package
  - tfarima-package, 3

  as.lagpol, 4
  as.um, 5
  autocorr, 5
  autocov (autocov.stsm), 6
  autocov.stsm, 6

  bsm, 7

  calendar (calendar.tfm), 8
  calendar.tfm, 8
  CalendarVar, 10
  ccf.tfm, 11
  coef.tfm, 11
  coef.um, 12

  diagchk (diagchk.tfm), 12
  diagchk.tfm, 12
  display, 13

  easter, 8, 14

  fit (fit.tfm), 16
  fit.stsm, 15
  fit.tfm, 16
  fit2autocov, 17

  ide, 18
  intervention (intervention.tfm), 19
  intervention.tfm, 19
  InterventionVar, 21
  inv, 21

  lagpol, 22

  logLik.um, 23
  modify (modify.tfm), 24
  modify.tfm, 24
  nabla, 25
  noise, 26

  outlierDates, 26
  outliers (outliers.tfm), 27
  outliers.tfm, 27
  output.tf, 29

  pccf, 29
  phi, 30
  pi.weights, 31
  predict.tfm, 32
  predict.um, 33
  printLagpol, 34
  printLagpolList, 34
  psi.weights, 35

  residuals.tfm, 35
  residuals.um, 36
  rform, 37
  roots, 37
  roots.default (roots.lagpol), 38
  roots.lagpol, 38
  rsales, 39

  S, 39
  sdummies, 40
  seasadj, 40
  seriesC, 41
  seriesJ, 42
  setinputs (setinputs.tfm), 42
  setinputs.tfm, 42
  sform, 43
  signal, 44
  sim (sim.tfm), 44
  sim.tfm, 44
INDEX

sincos, 45
spec, 46
std, 47
stsm, 16, 47
sum_um, 50
summary.tfm, 48
summary.um, 49
tf, 50, 53
tfarima(tfarima-package), 3
tfarima-package, 3
tfest, 51
tfm, 9, 15, 17, 20, 28, 33, 41, 52, 56
theta, 54
ts, 32, 33, 41, 56
tsdia.tfm, 54
tsdia.um, 55
tsvalue, 55
ucomp (ucomp.tfm), 56
ucomp.tfm, 56
um, 9, 15, 17, 20, 28, 32, 33, 41, 51, 53, 56, 57
varsel, 58
Wtelephone, 59