

Package ‘tfarima’

July 22, 2025

Type Package

Title Transfer Function and ARIMA Models

Version 0.3.2

Date 2022-05-20

Description Building customized transfer function and ARIMA models with multiple operators and parameter restrictions. Functions for model identification, model estimation (exact or conditional maximum likelihood), model diagnostic checking, automatic outlier detection, calendar effects, forecasting and seasonal adjustment. See Bell and Hillmer (1983) <doi:10.1080/01621459.1983.10478005>, Box, Jenkins, Reinsel and Ljung <ISBN:978-1-118-67502-1>, Box, Pierce and Newbold (1987) <doi:10.1080/01621459.1987.10478430>, Box and Tiao (1975) <doi:10.1080/01621459.1975.10480264>, Chen

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

Contents

tfarima-package	3
as.lagpol	4
as.um	5
autocorr	5
autocov.stsm	6
bsm	7
calendar.tfm	8
CalendarVar	10
ccf.tfm	11
coef.tfm	11
coef.um	12
diagchk.tfm	12
display	13
easter	14
fit.stsm	15
fit.tfm	16
fit2autocov	17
ide	18
intervention.tfm	19
InterventionVar	21
inv	21
lagpol	22
logLik.um	23
modify.tfm	23
nabla	25
noise	25
outlierDates	26
outliers.tfm	27
output.tf	28
pccf	29
phi	30
pi.weights	31
predict.tfm	31
predict.um	32
printLagpol	34
printLagpolList	34
psi.weights	35
residuals.tfm	35
residuals.um	36
rform	37
roots	37
roots.lagpol	38
rsales	39
S	39
sdummies	40
seasadj	40

seriesC	41
seriesJ	42
setinputs.tfm	42
sform	43
signal	44
sim.tfm	44
sincos	45
spec	46
std	47
stsm	47
summary.tfm	48
summary.um	49
sum_um	50
tf	50
tfest	51
tfm	52
theta	54
tsdiag.tfm	54
tsdiag.um	55
tsvalue	55
ucomp.tfm	56
um	57
varsel	58
Wtelephone	59
Index	60

tfarima-package

Transfer Function and ARIMA Models.

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

- Bell, W.R. and Hillmer, S.C. (1983) Modeling Time Series with Calendar Variation, Journal of the American Statistical Association, Vol. 78, No. 383, pp. 526-534.
- Box, G.E., Jenkins, G.M., Reinsel, G.C. and Ljung, G.M. (2015) Time Series Analysis: Forecasting and Control. John Wiley & Sons, Hoboken.

Box, G.E.P., Pierce, D.A. and Newbold, D. A. (1987) Estimating Trend and Growth Rates in Seasonal Time Series, Journal of the American Statistical Association, Vol. 82, No. 397, pp. 276-282.

Box, G.E.P. and Tiao, G.C. (1975) "Intervention Analysis with Applications to Economic and Environmental Problems", Journal of the American Statistical Association, Vol. 70, No. 349, pp. 70-79.

Chen, C. and Liu, L. (1993) Joint Estimation of Model Parameters and Outlier Effects in Time Series, Journal of the American Statistical Association, Vol. 88, No. 421, pp. 284-297

Thompson, H. E. and Tiao, G. C. (1971) "Analysis of Telephone Data: A Case Study of Forecasting Seasonal Time Series," Bell Journal of Economics, The RAND Corporation, vol. 2(2), pages 515-541, Autumn.

as.lagpol

Lag polynomial

Description

as.lagpol converts a numeric vector $c(1, -a_1, \dots, -a_d)$ into a lag polynomial $(1 - a_1B - \dots - a_pB^p)$.

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	<i>Convert arima into um.</i>
-------	-------------------------------

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

```
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	<i>Theoretical simple/partial autocorrelations of an ARMA model</i>
----------	---

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

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

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.
...	other arguments.

Value

An object of class stsm.

References

Durbin, J. and Koopman, S.J. (2012) Time Series Analysis by State Space Methods, 2nd ed., Oxford University Press, Oxford.

Examples

```
bsm1 <- bsm(AirPassengers, bc = TRUE)
```

calendar.tfm

Calendar effects

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, $L = D0 + D1 + \dots + D6$ 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

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

```
## S3 method for class 'um'
calendar(
  mdl,
  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,
    ...
)

```

Arguments

<code>mdl</code>	an object of class <code>um</code> or <code>tfm</code> .
<code>y</code>	a time series.
<code>form</code>	representation for calendar effects: (1) <code>form = dif</code> , <code>L</code> , <code>D1-D0</code> , ..., <code>D6-D0</code> ; (2) <code>form = td</code> , <code>LPY</code> , <code>D1-D0</code> , ..., <code>D6-D0</code> ; (3) <code>form = td7</code> , <code>D0</code> , <code>D2</code> , ..., <code>D6</code> ; (4) <code>form = td6</code> , <code>D1</code> , <code>D2</code> , ..., <code>D6</code> ; (5) <code>form = wd</code> , $(D1+...+D5) - 2(D6+D0)/5$.
<code>ref</code>	a integer indicating the the reference day. By default, <code>ref = 0</code> .
<code>lom</code> , <code>lpyear</code>	a logical value indicating whether or not to include the lom/lead year indicator.
<code>easter</code>	logical. If TRUE an Easter effect is also estimated.
<code>len</code>	the length of the Easter, integer.
<code>easter.mon</code>	logical. TRUE indicates that Easter Monday is a public holiday.
<code>n.ahead</code>	a positive integer to extend the sample period of the deterministic variables with <code>n.ahead</code> observations, which could be necessary to forecast the output.
<code>p.value</code>	estimates with a p-value greater than <code>p.value</code> are omitted.
<code>envir</code>	environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
<code>...</code>	other arguments.

Value

An object of class `"tfm"`.

References

W. R. Bell & S. C. Hillmer (1983) Modeling Time Series with Calendar Variation, Journal of the American Statistical Association, 78:383, 526-534, DOI: 10.1080/01621459.1983.10478005

Examples

```

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

```
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

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

Value

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

References

Bell, W.R. and Hillmer, S.C. (1983) "Modeling time series with calendar variation", *Journal of the American Statistical Society*, Vol. 78, pp. 526–534.

Examples

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

ccf.tfm	<i>Cross-correlation check</i>
---------	--------------------------------

Description

ccf displays ccf between prewhitened inputs and residuals.

Usage

```
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.

coef.tfm	<i>Coefficients of a transfer function model</i>
----------	--

Description

coef extracts the "coefficients" from a TF model.

Usage

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

Arguments

object	a tfm object.
...	other arguments.

Value

A numeric vector.

coef.um	<i>Coefficients of a univariate model</i>
---------	---

Description

coef extracts the "coefficients" from a um object.

Usage

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

Arguments

object	a um object.
...	other arguments.

Value

A numeric vector.

diagchk.tfm	<i>Diagnostic checking</i>
-------------	----------------------------

Description

diagchk displays tools for diagnostic checking.

Usage

```
## 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

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

Examples

```
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

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, ...)
```

Arguments

um	an object of class um or a list of these objects.
...	additional arguments.
lag.max	number of lags for ACF/PACF.
n.freq	number of frequencies for the spectrum.
log.spec	logical. If TRUE log spectrum is computed.
graphs	vector of graphs.
byrow	orientation of the graphs.
eq	logical. If TRUE the model equation is used as title.

Examples

```
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 <code>um</code> .
...	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)
```

fit.stsm

Estimation of a STS model

Description

`fit` fits the stsm to the time series `y`.

Usage

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

Arguments

mdl an object of class `stsm`.
 method argument of the `optim` function.
 show.iter logical value to show or hide the estimates at the different iterations.
 ... other arguments.

Value

An object of class "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

*Estimation of the ARIMA model***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

<code>mdl</code>	an object of class <code>um</code> or <code>tfm</code> .
<code>y</code>	a ts object.
<code>method</code>	Exact/conditional maximum likelihood.
<code>optim.method</code>	the method argument of the <code>optim</code> function.
<code>show.iter</code>	logical value to show or hide the estimates at the different iterations.
<code>fit.noise</code>	logical. If TRUE parameters of the noise model are fixed.
<code>envir</code>	environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
<code>...</code>	additional arguments.
<code>z</code>	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)

```

fit2autocov

Estimation of a STS model by the method of moments

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

<code>mdl</code>	an object of class <code>stsm</code> .
<code>...</code>	other arguments.
<code>g</code>	a vector of theoretical autocovariances ($\gamma[k]$, $k=0, \dots, K$).
<code>method</code>	optimization method.
<code>show.iter</code>	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

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

```

Description

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

Usage

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

intervention(mdl, ...)

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

Arguments

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

Value

an object of class `"tfm"` or a table.

InterventionVar	<i>Intervention variables</i>
-----------------	-------------------------------

Description

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

Usage

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

Arguments

Y	an object of class <code>ts</code> used to determine the sample period and frequency.
date	the date of the event, <code>c(year, month)</code> .
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

G. E. P. Box, G. C. Tiao, "Intervention Analysis with Applications to Economic and Environmental Problems", Journal of the American Statistical Association, Vol. 70, No. 349. (Mar., 1975), pp. 70-79.

Examples

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

inv	<i>Inverse of a lag polynomial</i>
-----	------------------------------------

Description

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

Usage

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

S3 method for class 'lagpol'

```
inv(lp, lag.max = 10, ...)
```

Arguments

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

Value

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

Examples

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

lagpol	<i>Lag polynomials</i>
--------	------------------------

Description

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

Usage

```
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

```
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

<code>mdl</code>	an object of class <code>um</code> or <code>tfm</code> .
<code>...</code>	additional arguments.
<code>ar</code>	list of stationary AR lag polynomials.
<code>i</code>	list of nonstationary AR (I) polynomials.
<code>ma</code>	list of MA polynomials.
<code>mu</code>	mean of the stationary time series.
<code>sig2</code>	variance of the error.
<code>bc</code>	logical. If TRUE logs are taken.
<code>fit</code>	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	<i>Unscramble I polynomial</i>
-------	--------------------------------

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 $c(1, a_1, \dots, a_d)$

Note

This function returns the member variable um\$nabla.

Examples

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

noise	<i>Noise of a transfer function model</i>
-------	---

Description

noise computes the noise of a linear transfer function model.

Usage

```
noise(tfm, ...)

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

Arguments

<code>tfm</code>	an object of the class <code>tfm</code> .
<code>...</code>	additional arguments.
<code>y</code>	output of the TF model if it is different to that of the <code>tfm</code> object.
<code>diff</code>	logical. If TRUE, the noise is differenced with the "i" operator of the univariate model of the noise.
<code>exp</code>	logical. If TRUE, the antilog transformation is applied.
<code>envir</code>	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 indeces and dates of outliers.

Usage

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

Arguments

<code>x</code>	an <code>ts</code> object.
<code>c</code>	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.

outliers.tfm*Outliers detection at known/unknown dates*

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,
  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 <code>um</code> or <code>tfm</code> .
y	an object of class <code>ts</code> , optional.
types	a vector with the initials of the outliers to be detected, <code>c("AO", "LS", "TC", "IO")</code> .
dates	a list of dates <code>c(year, season)</code> . If <code>dates = NULL</code> , 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 <code>dates = NULL</code> .
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 <code>n.ahead</code> observations, which could be necessary to forecast the output.
p.value	estimates with a p-value greater than <code>p.value</code> 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 <code>NULL</code> 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

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

Arguments

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

Value

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

<i>phi</i>	<i>Unscramble AR polynomial</i>
------------	---------------------------------

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 $c(1, a_1, \dots, a_d)$

Note

This function returns the member variable *um*\$*phi*.

Examples

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

pi.weights	<i>Pi weights of an AR(I)MA model</i>
------------	---------------------------------------

Description

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

Usage

```
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

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

predict.tfm	<i>Forecasting with transfer function models</i>
-------------	--

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.

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 <code>um</code> .
z	an object of class <code>ts</code> .
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 <code>um</code> object.
envir	environment in which the function arguments are evaluated. If <code>NULL</code> 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	<i>Print numeric vector as a lagpol object</i>
-------------	--

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	<i>Print a list of lagpol objects</i>
-----------------	---------------------------------------

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	<i>Psi weights of an AR(I)MA model</i>
-------------	--

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	<i>Residuals of a transfer function model</i>
---------------	---

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	<i>Residuals of the ARIMA model</i>
--------------	-------------------------------------

Description

residuals computes the exact or conditional residuals.

Usage

```
## 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

```
z <- AirPassengers
air1 <- um(z, i = list(1, c(1, 12)), ma = list(1, c(1, 12)), bc = TRUE)
r <- residuals(air1)
summary(r)
```

rform	<i>Reduce form for STS model</i>
-------	----------------------------------

Description

rform finds the reduce form for a STS model.

Usage

```
rform mdl, ...

## S3 method for class 'stsm'
rform mdl, ...
```

Arguments

mdl an object of class stsm.
... other arguments.

Value

An object of class um.

Examples

```
b <- 1
C <- as.matrix(1)
stsm1 <- stsm(b = b, C = C, s2v = c(lv1 = 1469.619), s2u = c(irr = 15103.061))
rf1 <- rform(stsm1)
nabla(rf1)
theta(rf1)
```

roots	<i>Roots of the lag polynomials of an ARIMA model</i>
-------	---

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	<i>Roots of a lag polynomial</i>
--------------	----------------------------------

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

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

Chen, C. and Liu, L. (1993) Joint Estimation of Model Parameters and Outlier Effects in Time Series, *Journal of the American Statistical Association*, Vol. 88, No. 421, pp. 284-297

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	<i>Seasonal dummies</i>
----------	-------------------------

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 <code>ts</code> 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

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

seasadj	<i>Seasonal adjustment</i>
---------	----------------------------

Description

seasadj removes the seasonal component of time series.

Usage

```
seasadj(mdl, ...)

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


Arguments

<code>mdl</code>	an object of class <code>um</code> or <code>tfm</code> .
<code>...</code>	additional arguments.
<code>z</code>	an object of class <code>ts</code> .
<code>method</code>	forward/backward forecasts or a mixture of the two.
<code>envir</code>	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

Box, G.E., Jenkins, G.M., Reinsel, G.C. and Ljung, G.M. (2015) Time Series Analysis: Forecasting and Control. John Wiley & Sons, Hoboken.

seriesJ	<i>Gas furnace data</i>
---------	-------------------------

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

Box, G.E., Jenkins, G.M., Reinsel, G.C. and Ljung, G.M. (2015) Time Series Analysis: Forecasting and Control. John Wiley & Sons, Hoboken.

setinputs.tfm	<i>setinputs adds new inputs into a transfer function model.</i>
---------------	--

Description

setinputs adds new inputs into a transfer function model.

Usage

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

setinputs(mdl, ...)

## 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.

sform	<i>Structural form for an ARIMA model</i>
-------	---

Description

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

Usage

```
sform(mdl, ...)

## S3 method for class 'um'
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	<i>Signal component of a TF model</i>
--------	---------------------------------------

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.

sim.tfm	<i>Time series simulation form an ARIMA or TF model</i>
---------	---

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,
    n0 = 0,
    a = NULL,
    seed = NULL,
    envir = parent.frame(),
    ...
)

```

Arguments

<code>mdl</code>	an object of class <code>um</code> or <code>tfm</code> .
<code>n</code>	number of observations.
<code>y0</code>	initial conditions for the nonstationary series.
<code>seed</code>	an integer.
<code>...</code>	other arguments.
<code>z0</code>	initial conditions for the nonstationary series.
<code>n0</code>	remove the <code>n0</code> first observation, integer.
<code>a</code>	vector of innovations, optional.
<code>envir</code>	environment in which the function arguments are evaluated. If <code>NULL</code> the calling environment of this function will be used.

Value

An object of class `ts`.

<code>sincos</code>	<i>Trigonometric variables</i>
---------------------	--------------------------------

Description

`sincos` creates an full set of trigonometric variables.

Usage

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

Arguments

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

Value

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	<i>Standardize time series</i>
-----	--------------------------------

Description

std standardizes a time series.

Usage

std(x)

Arguments

x a ts object.

Value

The standardized time series.

stsm	<i>Structural Time Series models</i>
------	--------------------------------------

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 stsm.

References

Durbin, J. and Koopman, S.J. (2012) Time Series Analysis

Examples

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

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.um	<i>Summary of um model</i>
------------	----------------------------

Description

summary prints a summary of the estimation and diagnosis.

Usage

```
## 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

```
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 serveral 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, $V(B) = w_0(1 - w_1B - \dots - w_qB^q)/(1 - d_1B - \dots - d_pB^p)B^dX_t$. 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

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

Arguments

<code>x</code>	input, a ts object or a numeric vector.
<code>delay</code>	integer.
<code>w0</code>	constant term of the polynomial $V(B)$, double.
<code>ar</code>	list of stationary AR polynomials.
<code>ma</code>	list of MA polynomials.
<code>um</code>	univariate model for stochastic input.
<code>n.back</code>	number of backcasts to extend the input.
<code>par.prefix</code>	prefix name for parameters.
<code>envir</code>	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

Box, G.E., Jenkins, G.M., Reinsel, G.C. and Ljung, G.M. (2015) Time Series Analysis: Forecasting and Control. John Wiley & Sons, Hoboken.

Wei, W.W.S. (2006) Time Series Analysis Univariate and Multivariate Methods. 2nd Edition, Addison Wesley, New York, 33-59.

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)")
```

tfest

Preestimates of a transfer function

Description

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

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

<code>output</code>	a ts object or a numeric vector.
<code>xreg</code>	a matrix of regressors.
<code>inputs</code>	a list of tf objects.
<code>noise</code>	a um object for the noise.
<code>fit</code>	logical. If TRUE, model is fitted.
<code>envir</code>	environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
<code>new.name</code>	logical. Argument used internally: if TRUE a new name is assigned to the output, otherwise it keeps its name saved in <code>noise\$z</code> .
<code>...</code>	additional arguments.

Value

An object of the class `tfm`.

References

Box, G.E., Jenkins, G.M., Reinsel, G.C. and Ljung, G.M. (2015) Time Series Analysis: Forecasting and Control. John Wiley & Sons, Hoboken.

See Also

[tf](#) and [um](#).

theta	<i>Unscramble MA polynomial</i>
-------	---------------------------------

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 $c(1, a_1, \dots, a_d)$

Note

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

Examples

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

tsdiag.tfm	<i>Diagnostic Plots for Time-Series Fits Description</i>
------------	--

Description

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

Usage

```
## 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

Diagnostic Plots for Time-Series Fits Description

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	<i>Unobserved components</i>
-----------	------------------------------

Description

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

Usage

```
## 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

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

um	<i>Univariate (ARIMA) model</i>
----	---------------------------------

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 = 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

Box, G.E.P., Jenkins, G.M., Reinsel, G.C. and Ljung, G.M. (2015) Time Series Analysis: Forecasting and Control. John Wiley & Sons, Hoboken.

Examples

```
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	<i>Variable selection</i>
--------	---------------------------

Description

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

Usage

```
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.

Wtelephone

Wisconsin Telephone Company

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/1LP8aMIQewMrxg0lrg9rN3eWHhZuUsY8K/view?usp=sharing>

References

Thompson, H. E. and Tiao, G. C. (1971) "Analysis of Telephone Data: A Case Study of Forecasting Seasonal Time Series," Bell Journal of Economics, The RAND Corporation, vol. 2(2), pages 515-541, Autumn.

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), 23
- modify.tfm, 23
- nabla, 25
- noise, 25
- outlierDates, 26
- outliers (outliers.tfm), 27
- outliers.tfm, 27
- output.tf, 28
- pccf, 29
- phi, 30
- pi.weights, 31
- predict.tfm, 31
- predict.um, 32
- 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

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](#)
tsdiag.tfm, [54](#)
tsdiag.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](#)