Package: tfarima (via r-universe)

August 25, 2024

Type Package

Title Transfer Function and ARIMA Models

Version 0.3.6

Date 2024-04-28

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 and Liu (1993) <doi:10.1080/01621459.1993.10594321>.

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, MASS, 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

Date/Publication 2020-11-14 13:00:02 UTC

Repository https://gallegoj.r-universe.dev

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

RemoteRef HEAD

RemoteSha 2fa8ee0b813a597e2b0cd4d882ed5685f497fd90

Contents

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.

add.um *Addition or substraction of univariate (ARIMA) models*

Description

`add.um` creates a univariate (ARIMA) model from the addition or substraction of two univariate (arima) models.

Usage

```
add.um(um1, um2, add = TRUE, tol = 1e-05)
```

Arguments

um1, um2	Two "um" S3 objects.
add	logical. If FALSE, the second model is substracted from the first one.
tol	tolerance to check if a value is null.

Value

A "um" S3 object.

Note

The + and - operators can also be used to add or substract ARIMA models.

Examples

```
um1 <- um(i = "(1 - B)", ma = "(1 - 0.8B)")
um2 <- um(i = "(1 - B12)", ma = "(1 - 0.8B^12)")
um3 <- add.um(um1, um2)
um4 <- um3 - um2
```

---

altform — *Alternative form for STS model*

Description

`altform` converts a STS model given in contemporaneous form into future form, and vice versa.

Usage

```
altform(mdl, ...)

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

Arguments

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

Value

An object of class `stsm`.

Examples

```
# Local level model
b <- 1
C <- as.matrix(1)
stsm1 <- stsm(Nile, b, C, S = diag(c(irr = 1, lvl = 0.5)) )
stsm1
stsm2 <- altform(stsm1)
```

`as.lagpol` *Lag polynomial*

Description

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

Usage

```
as.lagpol(pol, p = 1, coef.name = "a")
```

Arguments

`pol`	a numeric vector.
`p`	integer power.
`coef.name`	name prefix for coefficients.

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

```
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

```
## S3 method for class 'stsm'
autocov(mdl, lag.max = NULL, arma = TRUE, varphi = FALSE, tol = 1e-04, ...)

autocov(mdl, ...)

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

Arguments

mdl	an object of class um or stsm.
lag.max	maximum lag for autocovariances.
arma	logical. If TRUE, the autocovariances for the stationary ARMA model of the reduced form are computed. Otherwise, the autocovariances are only computed for the MA part.
varphi	logical. If TRUE, the varphi polynomial of the reduced form is also returned.
tol	tolerance to check if a root is close to one.
...	additional arguments.

Value

A numeric vector.

Note

The I polynomial is ignored.

Examples

```
# Local level model
stsm1 <- stsm(b = 1, C = 1, S = diag(c(irr = 0.8, lvl = 0.04))
autocov(stsm1)

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

`autocov2MA` *MA process compatible with a vector of autocovariances*

Description

`autocov2MA` computes the error variance and the MA polynomial from a vector of autocovriances.

Usage

```
autocov2MA(x, method = c("roots", "acov"), tol = 1e-05)
```

Arguments

`x`	a numeric vector with q autocovariances.
`method`	character. Two methods are provided to compute the MA parameters: roots, based on the roots of the autocovariance generating function; acov, based on the non-linear system of equations relating autocovariances and MA parameters.
`tol`	tolerance to check if an autocovariance is null.

Value

A vector of the form c(s2, 1, -theta1, ..., -thetaq).

Examples

```
ma1 <- um(ma = "1 - 0.8B", sig2 = 0.5)
autocov2MA(autocov(ma1, 1))
autocov2MA(autocov(ma1, 1), "acov")
```

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"),
  par = c(irr = 0.75, lvl = 1, slp = 0.05, seas = 0.075),
  fixed = c(lvl = TRUE),
  xreg = NULL,
  fit = TRUE,
  updmdl = 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))
par	real vector with the error variances of each unobserved component.
fixed	logical vector to fix parameters.
xreg	matrix of regressors.
fit	logical. If TRUE, model is fitted.
updmdl	function to update the parameters of the BSM.
...	additional 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 + ... + 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

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

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

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` *Cross-correlation check*

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` *Coefficients of a transfer function model*

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 *Coefficients of a univariate model*

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 *Diagnostic checking*

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

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 at which tick-marks are to be drawn.
std	logical. If TRUE standardized residuals are shown.
envir	environment in which the function arguments are evaluated. If NULL the calling environment of this function will be used.
...	additional arguments.
z	optional, an object of class ts.

Examples

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

---

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,
  lags.at = NULL,
  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.
lags.at	the lags of the ACF/PACF at which tick-marks are to be drawn.
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 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)
```

equation *Equation of a univariate ARIMA model*

Description

equation prints the equation of an object of class um.

Usage

```
equation(um, ...)

## S3 method for class 'um'
equation(um, unscramble = TRUE, digits = 4, ...)
```

Arguments

um	an object of class um or a list of these objects.
...	additional arguments.
unscramble	logical. If TRUE, AR, I and MA polynomials are unscrambled.

Examples

```
equation(um(ar = "(1 - 0.8B)"))
```

---

factors *Lag polynomial factorization*

Description

factors extracts the simplifying factors of a polynomial in the lag operator by replacing, if needed, its approximate unit or real roots to exact unit or real roots.

Usage

```
factors(lp, ...)

## S3 method for class 'lagpol'
factors(lp, tol = 1e-05, expand = FALSE, ...)
```

Arguments

lp	an object of class lagpol.
...	additional arguments.
tol	tolerance for real and unit roots.
expand	logical if the expanded polynomial should be returned.

Value

factors returns a list with the simplifying factors of the lag polynomial or the expanded polynomial.

Examples

```
factors( as.lagpol(c(1, rep(0, 11), -1)) )
```

fit.stsm *Estimation of a STS model*

Description

`fit` is used to estimate a stsm model.

Usage

```
## S3 method for class 'stsm'
fit(mdl, updmdl, par, fixed = NULL, show.iter = FALSE, tol = 1e-04, ...)
```

Arguments

`mdl`	an object of class `stsm`.
`par`	argument passed to the updmod function.
`fixed`	vector of logical values indicating which parameters are fixed (TRUE) or estimated (FALSE).
`show.iter`	logical value to show or hide the estimates at the different iterations.
`tol`	tolerance to check if a root is close to one.
`...`	other arguments.
`updmod`	function to update the parameters of the model.

Value

An object of class "stsm" with estimated parameters.

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

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

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

`ide` *Identification plots*

Description

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

Usage

```
ide(
  Y,
  transf = list(),
  order.polreg = 0,
  lag.max = NULL,
  lags.at = NULL,
  freq.at = NULL,
  wn.bands = TRUE,
  graphs = c("plot", "acf", "pacf"),
  set.layout = TRUE,
  byrow = TRUE,
  main = "",
  plot.abline.args = NULL,
  plot.points.args = NULL,
  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 (cummulative 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.
plot.abline.args	Add straight lines to time series plot.
plot.points.args	Add points to time series plot.
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)))
```

---

ikf — *Initialization Kalman filter*

Description

ikf computes the starting values x0 and P0 by generalized least squares using the first n observations.

Usage

```
ikf(mdl, ...)

## S3 method for class 'stsm'
ikf(mdl, y = NULL, n = 0, ...)
```

Arguments

`mdl`	an object of class `stsm`.
`...`	additional arguments.
`y`	optional time series if it differes from model series.
`n`	integer, number of observations used to estimate the inical conditions. By default, n is the length of y.

Value

An list with the initial state and its covariance matrix.

`init` *Initial conditions for Kalman filter*

Description

`init` provides starting values to initialise the Kalman filter

Usage

```
init(mdl, ...)

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

Arguments

`mdl`	an object of class `stsm`.
`...`	additional arguments.

Value

An list with the initial state and its covariance matrix.

intervention.tfm *Intervention analysis/Outlier treatment*

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

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` *Intervention variables*

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 `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

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 *Inverse of a lag polynomial*

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)))
```

kf *Kalman filter for STS models*

Description

kf computes the innovations and the conditional states with the Kalman filter algorithm.

Usage

```
kf(mdl, ...)

## S3 method for class 'stsm'
kf(mdl, y = NULL, x1 = NULL, P1 = NULL, filtered = FALSE, ...)
```

Arguments

`mdl`	an object of class `stsm`.
`...`	additional arguments.
`y`	time series to be filtered when it differs from the model series.
`x1`	initial state vector.
`P1`	covariance matrix of x1.
`filtered`	logical. If TRUE, the filtered states x_tlt and their covariances matrices P_tlt are returned. Otherwise, x_tlt-1 and P_tlt-1 are

Value

An list with the innovations, the conditional states and their covariance matrices.

`ks` *Kalman smoother for STS models*

Description

`ks` computes smoothed states and their covariance matrices.

Usage

```
ks(mdl, ...)

## S3 method for class 'stsm'
ks(mdl, x1 = NULL, P1 = NULL, ...)
```

Arguments

`mdl`	an object of class `stsm`.
`...`	additional arguments.
`x1`	initial state vector.
`P1`	covariance matrix of x1.

Value

An list with the smoothed states and their covariance matrices.

lagpol *Lag polynomials*

Description

lagpol creates a lag polynomial of the form $(1 - coef_1 B^s - ... - coef_d B^s d)^p$. 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.stsm` *Log-likelihood of an STS model*

Description

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

Usage

```
## S3 method for class 'stsm'
logLik(mdl, y = NULL, method = c("exact", "cond"), ...)
```

Arguments

`mdl`	an object of class `stsm`.
`y`	an object of class `ts`.
`method`	exact or conditional.
`...`	additional parameters.
`tol`	tolerance to check if a root is close to one.

Value

The exact or conditional log-likelihood.

`logLik.um` *Log-likelihood of an ARIMA model*

Description

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

Usage

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

Arguments

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

Value

The exact or conditional log-likelihood.

modify.tfm *Modifying a TF or an ARIMA model*

Description

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

Usage

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

modify(mdl, ...)

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

Arguments

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

Value

An object of class um or um.

Examples

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

---

msx *Minimum signal extraction*

Description

msx extracts a signal from a time series.

Usage

```
msx(mdl, ...)

## S3 method for class 'um'
msx(
  mdl,
  ar = NULL,
  i = NULL,
  canonical = TRUE,
  tol = 1e-05,
  check = TRUE,
  method = c("roots", "acov"),
  single = TRUE,
  ret = c("default", "min"),
  envir = parent.frame(),
  ...
)
```

Arguments

mdl	an object of class um or tfm.
...	additional arguments.
ar, i	AR and/or I lag polynomials for the signal.
canonical	logical value to set or not the canonical requirement.
tol	tolerance to check if a value is null.
check	logical value to check if ar and i are simplifying factors of the object mdl.
method	the procedure to obtain the MA parameters of the model of the signal is based either on the roots or on the autocovariances, method = c("roots", "acov").
single	logical. TRUE for single signal and FALSE for multiple signals.
ret	type of return, c("default", "min")
envir	environment.

Value

An object of class `msx`.

`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)
```

`nabla.stsm` *Non stationary AR polynomial of the reduced form*

Description

`nabla` returns the non stationary AR polynomialof the reduced form of an object of the `stsm` class.

Usage

```
## S3 method for class 'stsm'
nabla(mdl, tol = 1e-04)
```

Arguments

`mdl`	an object of class `stsm`.
`tol`	tolerance to check if a root is close to one.

Value

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

Examples

```
stsm1 <- stsm(b = 1, C = 1, S = diag(c(0.8, 0.04)))
nabla(stsm1)
```

---

noise *Noise of a transfer function model*

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

`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 indeces and dates of outliers.

Usage

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

Arguments

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

Value

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

`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 `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

```
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.

---

phi *Unscramble AR polynomial*

Description

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

Usage

```
phi(um)

## S3 method for class 'um'
phi(um)
```

Arguments

um	an object of class `um`.

Value

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

Note

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

Examples

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

`pi.weights` *Pi weights of an AR(I)MA model*

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 *Forecasting with transfer function models*

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 tree ways: (1) extending the time series with forecasts so that the length of the intput 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)
```

print.um *Print univariate models*

Description

Print univariate models

Usage

```
## S3 method for class 'um'
print(x, arima = FALSE, ...)

## S3 method for class 'msx'
print(x, ...)
```

Arguments

arima	logical. If TRUE, lag ARIMA polynomials are printed.

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

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

---

restr.lagpol *Restricted lag polynomials*

Description

rest.lagpol creates two special types of restricted lag polynomials that can be useful to model seasonal time series: 1 + theta^(1/s)B + theta^(2/s)B^2 + ... + + theta^((s-1)/s)B^s-1 and 1 - 2cos(2pif/s)sqrt(theta)B+ thetaB^2

Usage

```
restr.lagpol(param = c(Theta = 0.8), s = 12, f = NULL, p = 1)
```

Arguments

param	double, the value of the parameter with a name.
s	the seasonal period, integer.
f	frequency for second order lag polynomials.
p	the power of lag polynomial, integer.

Value

rest.lagpol returns an object of class "lagpol".

Examples

```
rest.lagpol(param = c(Theta = 0.8), s = 12)
rest.lagpol(param = c(Theta = 0.8), s = 12, f = 1)
```

rform *Reduce form for STS model*

Description

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

Usage

```
rform(mdl, ...)

## S3 method for class 'stsm'
rform(mdl, tol = 1e-04, ...)
```

Arguments

`mdl`	an object of class `stsm`.
`...`	other arguments.
`tol`	tolerance to check if a root is close to one.

Value

An object of class `um`.

Examples

```
b <- 1
C <- as.matrix(1)
stsm1 <- stsm(b = b, C = C, Sv = 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

```
roots(x, ...)

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

Arguments

x	an object of class um.
...	additional arguments.
opr	character that indicates which operators are selected.

Value

List of matrices with the roots of each single polynomial.

Examples

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

---

roots.lagpol *Roots of a lag polynomial*

Description

`roots.lagpol` computes the roots of a lag polynomial.

Usage

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

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

Arguments

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

Value

A vector or a table.

Examples

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

`roots2lagpol` *Lag polynomial from roots*

Description

`roots2lagpol` creates a lag polynomial from its roots.

Usage

```
roots2lagpol(x, ...)
```

Arguments

`x`	an object of class `complex`.
`...`	additional arguments.

Value

A lag polynomial.

Examples

```
roots2lagpol(polyroot(c(1,-1)))
```

`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 origianl 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

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

seasadj *Seasonal adjustment*

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

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

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

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 *Structural form for an ARIMA model*

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,
  z = NULL,
  msoe = TRUE,
  index = NULL,
  nnls = NULL,
  cform = TRUE,
  tol = 1.490116e-08,
  envir = NULL,
  ...
)
```

Arguments

`mdl`	an object of class `um`.
`...`	other arguments.
`z`	an optional time series.
`msoe`	logical, TRUE for multiple source of errors and FALSE for single source of error.
`index`	an optional vector of integers both to group common variances or to fix some variances to zero.
`cform`	logical. TRUE for contemporaneous form and FALSE for future form.
`tol`	tolerance to check if the elements of b and C are zero.
`envir`	environment, see "`um`".

Value

An object of class `stsm`

Examples

```
airl <- um(i = list(1, c(1, 12)), ma = "(1 - 0.8B)(1 - 0.8B12)")
sf <- sform(airl, index = c(1, 0, rep(2, 11)))
sf
```

signal — *Signal component of a TF model*

Description

`signal` extracts the signal of a TF model.

Usage

```
signal(mdl, ...)

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

Arguments

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

Value

A "ts" object.

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

```
## 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

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.

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, nabla = FALSE, n.freq = 501, ...)
```

Arguments

um	an object of class um.
...	additional parameters.
nabla	logical. If TRUE, the pseudospectrum for a non stationary ARIMA model is calculated. By default, the spectrum is computed for the stationary ARMA model.
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, S, xreg = NULL, bc = FALSE, cform = TRUE)
```

Arguments

`y`	an object of class `ts`.
`b`	vector of constants.
`C`	matrix of constants.
`S`	covariance matrix of the error vector (u_t, v_t).
`xreg`	matrix of regressors.
`bc`	logical. If TRUE logs are taken.
`cform`	logical. If TRUE station equation is given in contemporaneous form, otherwise it is written in future form.

Details

y(t) = b'x(t) + u(t) (observation equation), x(t+j) = Cx(t+j-1) + v(t) (state equation), j = 0 for contemporaneous form or 1 for future form.

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.

Harvey, A.C. (1989) Forecasting, Structural Time Series Models and the Kalman Filter. Cambridge University Press, Cambridge.

Examples

```
# Local level model
b <- 1
C <- as.matrix(1)
stsm1 <- stsm(Nile, b, C, S = diag(c(irr = 1, lvl = 0.5)) )
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` *Summary of um model*

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 diagonosis.

Examples

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

tf — *Transfer function for input*

Description

`tf` creates a rational transfer function for an input, V(B) = w0(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 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

`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

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

`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

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 *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, ...)
```

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` *Unobserved components*

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("msx", "msx0", "mixed", "forecast", "backcast"),
  envir = parent.frame(),
  ...
)
```

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 *Univariate (ARIMA) model*

Description

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

Usage

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

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.
`warn`	logical. If TRUE, a warning is displayed for non-admissible models.
`...`	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)")
```

`unitcircle` *Unit circle*

Description

`unitcircle` plots the inverse roots of a lag polynomial together the unit circle.

Usage

```
unitcircle(lp, ...)

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

## S3 method for class 'lagpol'
unitcircle(lp, s = 12, ...)
```

Arguments

`lp`	an object of class `lagpol`.
`...`	additional arguments.
`s`	integer, seasonal period.

Value

`unitcircle` returns a NULL value.

Examples

```
unitcircle(as.lagpol(c(1, rep(0, 11), -1)))
```

`varsel` *Variable selection*

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.

`wkfilter` *Wiener-Kolmogorov filter*

Description

`wkfilter` extracts a signal for a time series.

Usage

```
wkfilter(um.z, ...)

## S3 method for class 'um'
wkfilter(um.z, um.uc, z = NULL, envir = parent.frame(), ...)
```

Arguments

`um.z`	an object of class `um`.
`...`	additional arguments.
`z`	optional, time series.
`envir`	environment.
`um.z, um.uc`	ARIMA models for the observed time series and the unobserved component.

Value

An object of class `wkfilter`.

Examples

```
um1 <- um(AirPassengers, bc = T, i = list(1, c(1,12)), ma = list(1, c(1,12)))
msx1 <- msx(um1, i = "(1-B)2")
trend <- wkfilter(um1, msx1$signal1)
seas <- wkfilter(um1, msx1$signal2)
```

wold.pol *Wold polynomial*

Description

Transforming a palindromic polymonial into a Wold polynomial/ Computing the Cramer-Wold factorization

Usage

```
wold.pol(a, type = c("wold", "palindromic", "cramer-wold"), tol = 1e-05)
```

Arguments

type	character indicating the type of polynomial: (1) Wold polynomial, (2) Palindromic polynomial and (3) Cramer-Wold factor.
tol	tolerance to check if an autocovariance is zero.
x	numeric vector, coefficients of a palindromic or a Wold polynomial.

Details

wold.pol can be used with three purposes:

(1) to transform a self-reciprocal or palindromic polynomial a_0 + a_1(B+F) + a_2(B^2+F^2) + ... + a_p(B^p+F^p) into a Wold polynomial b_0 + b_1(B+F) + b_2(B+F)^2 + ... + b_p(B+F)^p;

(2) to revert the previous transformation to obtain the palindromic polynominal from a Wold polynomial and

(3) to compute the Cramer-Wold factorization: b(B+F) = c(B)c(F).

Value

Numeric vector.

Examples

```
wold.pol(c(6, -4, 1))
```

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 Montly inward station movements.

Source

https://drive.google.com/file/d/1LP8aMIQewMrxgOlrg9rN3eWHhZuUsY8K/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