Package 'tsdecomp'

October 14, 2022

Version 0.2

Date 2017-01-03

Title Decomposition of Time Series Data

Description ARIMA-model-based decomposition of quarterly and monthly time series data.
The methodology is developed and described, among others, in Burman (1980) <DOI:10.2307/2982132> and Hillmer and Tiao (1982) <DOI:10.2307/2287770>.

Author Javier López-de-Lacalle <javlacalle@yahoo.es>

Maintainer Javier López-de-Lacalle <javlacalle@yahoo.es>

Depends R (>= 3.0.0)

NeedsCompilation no

Encoding UTF-8

License GPL-2

URL https://jalobe.com

Repository CRAN

Date/Publication 2017-01-04 10:56:16

R topics documented:

`tsdecomp-package` *ARIMA-Model-Based Decomposition of Time Series Data*

Description

ARIMA-model-based decomposition of a time series.

Details

The methods implemented in the package are developed and described (among others) in the references given below. The package is mainly intended for annual, quarterly and monthly time series. The bottom line of the procedure can be summarized as follows. An ARIMA model is fitted to the observed series. Then the pseudo-spectrum of the model is computed and decomposed into partial fractions. This gives ARIMA models for the unobserved components (e.g., trend, seasonal and irregular), which are then used to obtain the weights of double-sided linear filters upon which estimates of the components are obtained.

For practical purposes, the main function provided in the package is `ARIMAdec`. This function relies on other procedures that implement different stages of the procedure: `roots.allocation`, `pseudo.spectrum`, `partial.fraction`, `canonical.decomposition`, `filtering`.

An introduction to the methodology and the package in the form of a vignette is available here:

https://www.jalobe.com/doc/tsdecomp.pdf

Author(s)

Javier López-de-Lacalle <javlacalle@yahoo.es>

https://jalobe.com

References

Box, G. E. P., Hillmer, S. C. and Tiao, G. C. (1978) 'Analysis and Modeling of Seasonal Time Series' in *Seasonal Analysis of Economic Time Series*, Editor Zellner, A. pp. 309-334. U.S. Dept. of Commerce - Bureau of the Census. http://www.nber.org/chapters/c3904.pdf

Brockwell, P. J. and Davis, R. A. (1991) *Time Series: Theory and Methods*, Second Edition. Springer. doi: 10.1007/978-1-4419-0320-4

Burman, J. P. (1980) 'Seasonal Adjustment by Signal Extraction'. *Journal of the Royal Statistical Society. Series A (General)*, **143**(3), pp. 321-337. doi: 10.2307/2982132

Gómez, V. and Maravall, A. (2001) 'Programs **TRAMO** and **SEATS**. Instructions for the User (Beta Version: June 1997)'. Ministerio de Economía y Hacienda. Dirección General de Análisis y Programación Presupuestaria, Working paper SGAPE-97001. http://www.bde.es/f/webbde/SES/servicio/Programas_estadisticos_y_econometricos/Programas/ficheros/manualdos.pdf

Gómez, V. (2015) '**SSMMATLAB**: A Set of MATLAB Programs for the Statistical Analysis of State Space Models'. *Journal of Statistical Software*, **66**(1), pp. 1-37. doi: 10.18637/jss.v066.i09 http://www.sepg.pap.minhap.gob.es/sitios/sepg/en-GB/Presupuestos/Documentacion/Paginas/SSMMATLAB.aspx

Hillmer, S. C. and Tiao, G. C. (1982) 'An ARIMA-Model-Based Approach to Seasonal Adjustment'. *Journal of the American Statistical Association*, **77**(377), pp. 63-70. doi: 10.1080/01621459.1982.10477767

Maravall, A. and Pierce, D. A. (1987) 'A Prototypical Seasonal Adjustment Model'. *Journal of Time Series Analysis*, **8**(2), pp.177-193. doi: 10.1111/j.1467-9892.1987.tb00431.x

Planas, C. (1997) *Applied Time Series Analysis: Modelling, Forecasting, Unobserved Components Analysis and the Wiener-Kolmogorov Filter*. Eurostat: Series E, Methods. https://bookshop.europa.eu/en/applied-time-series-analysis-pbCA0897484/

Pollock, D. S. G. (1999) *A Handbook of Time-Series Analysis Signal Processing and Dynamics*. Academic Press. doi: 10.1016/B978-012560990-6/50002-6

`acgf2poly` *Change of Variable in the AutoCovariance Generating Function*

Description

Change of variable in the autocovariance generating function (ACGF). This transformation allows the pseudo-spectrum to be represented as a polynomial liable to be decomposed in partial fractions.

Usage

```
acgf2poly(x)
poly2acgf(x, type=c("roots2poly", "acov2ma"), tol = 1e-16, maxiter = 100,
  init.tol=1e-05, init.maxiter=100)
## S3 method for class 'tsdecMAroots'
print(x, units = c("radians", "degrees", "pi"), digits = 4, echo = TRUE, ...)
```

Arguments

`x`	numeric vector of autocovariances; for `poly2acgf`, an object of class `tsdecMAroots` returned by `type="roots2poly"`
`type`	character string selecting the method to undo the transformation.
`tol`	convergence tolerance to be used by `acov2ma`.
`maxiter`	maximum number of iterations allowed in `acov2ma`.
`init.tol`	convergence tolerance to be used by `acov2ma.init`.
`init.maxiter`	maximum number of iterations allowed in `acov2ma.init`.
`units`	character, the units in which the argument of the roots are printed. `units="pi"` prints the argument in radians as multiples of π.
`digits`	numeric, the number of significant digits to be used by `print`.
`echo`	logical, if `TRUE` the output is printed, otherwise a `invisible` copy of the matrix summarizing the results obtained by `poly2acgf` is returned.
`...`	further arguments to be passed to `print`.

Details

The ACGF is defined as a power series where the coefficients are the autocovariances γ_τ:

$$\gamma(z) = \gamma_0 + \gamma_1(z + z^{-1}) + \gamma_2(z^2 + z^{-2}) + \gamma_3(z^3 + z^{-3}) + \cdots$$

where z is a complex variable.

Replacing z by $e^{-i\omega}$ with $\omega \in [0, 2\pi]$ yields the spectral density multiplied by 2π. This gives a power series in the variable $2\cos(\omega j)$ (note that for $z = e^{-i\omega}$, which has unit modulus, the inverse $1/z$ is the complex-conjugate of z):

$$z^j + z^{-j} = \cos(\omega j) + i\sin(\omega j) + \cos(\omega j) - i\sin(\omega j) = 2\cos(\omega j)\,.$$

`acgf2poly` transforms the following expression in the variable $2\cos(\omega j)$:

$$A(2\cos(j\omega)) = a_0 + a_1 2\cos(\omega) + a_2 2\cos(2\omega) + \cdots + a_n 2\cos(n\omega)$$

into a polynomial in the variable $x = 2\cos(\omega)$:

$$B(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_n x^n\,.$$

`poly2acgf` recovers the vector of autocovariances by undoing the above transformation and computes the coefficients and the variance of the innovations of the moving average model related to those autocovariances. Two methods can be employed. 1) `type="acov2ma"`: this method recovers the autocovariances by undoing the change of variable; then, the the autocovariances are converted to the coefficients of a moving average by means of `acov2ma`. In the presence of non-invertible roots, this method may experience difficulties to converge.

2) `type="roots2poly"`: this method does not explicitly undo the change of variable `acgf2poly` (i.e., the vector of autocovariances is not recovered). Instead, the roots of the moving average polynomial $\theta(L)$ are obtained from the polynomial $\theta(L)\theta(L^{-1})$, where the coefficients are in terms of the polynomial $B(x)$ defined above; then, the coefficients of the moving average model are computed by means of `roots2poly`.

Value

`acgf2poly` returns the transformed vector of coefficients.

`poly2acgf` returns an object of class `tsdecMAroots` containing the coefficients and the variance of the innovations in the moving average model related to the autocovariances underlying the transformed coefficients. `print.tsdecMAroots` prints a summary of the results computed by `poly2acgf`.

Note

Method `type="roots2poly"` in `poly2acgf` is based on algorithm `pu2ma` in the software SSMMATLAB by Gómez, V. URL http://www.sepg.pap.minhap.gob.es/sitios/sepg/en-GB/Presupuestos/Documentacion/Paginas/SSMMATLAB.aspx.

See Also

acov2ma, roots2poly.

Examples

```
# the matrix 'm' performs the mapping from the original
# to the transformed coefficients
n <- 30
m <- diag(1, n, n)
n2 <- n - 2
j <- -1
tmp <- seq.int(2, n-1)
for (i in seq.int(3, n-2, 2))
{
  id <- cbind(seq_len(n2),seq.int(i,n))
  m[id] <- j * tmp
  n2 <- n2 - 2
  j <- -1 * j
  tmp <- cumsum(tmp[seq_len(n2)])
}
if (2*floor(n/2) == n) {  # if (n %% 2 == 0)
  m[cbind(seq_len(n2),seq.int(n-1,n))] <- j * tmp
} else
  m[1,n] <- j * tmp
m[1:10,1:10]

# equivalence of the original and transformed coefficients,
# example with an ARMA(2,1) model
#
# method 1: compute the spectral density upon the
# the theoretical autocovariances ('gamma') of the ARMA model
gamma <- ARMAacov(ar=c(0.8,-0.6), ma=0.4, lag.max=n-1)
w <- seq(0, pi, len=length(gamma))
spec1 <- rep(gamma[1], length(w))
for (i in seq_along(w))
{
  z <- 2*cos(w[i] * seq_len(length(gamma)-1))
  spec1[i] <- spec1[i] + sum(gamma[seq.int(2, n)] * z)
}
spec1 <- spec1/(2*pi)
#plot(w, spec1)

# method 2: compute the spectral density upon the
# transformed coefficients
newcoefs <- m
spec2 <- rep(newcoefs[1], length(w))
for (i in seq_along(w))
{
  x <- (2*cos(w[i]))^seq_len(n-1)
  spec2[i] <- spec2[i] + sum(newcoefs[seq.int(2, n)] * x)
}
spec2 <- spec2/(2*pi)
```

```
# both representations are equivalent
all.equal(spec1, spec2, check.names=FALSE)
#[1] TRUE

# the original coefficients (the autocovariances)
# can be recovered premultiplying by the inverse of the
# transformation matrix 'm'
all.equal(c(solve(m) %*% newcoefs), gamma, check.names=FALSE)
#[1] TRUE
```

---

acov2ma *Convert Autocovariances to Coefficients of a Moving Average*

Description

Convert autocovariances to coefficients of a moving average.

Usage

```
acov2ma.init(x, tol = 0.00001, maxiter = 100)
acov2ma(x, tol = 1e-16, maxiter = 100, init = NULL)
```

Arguments

x	a numeric vector containing the autocovariances.
tol	numeric, convergence tolerance.
maxiter	numeric, maximum number of iterations.
init	numeric, vector of initial coefficients.

Details

`acov2ma.init` is based on procedure (17.35) described in Pollock (1999). `acov2ma` is the Newton-Raphson procedure (17.39) described in the same reference.

Value

A list containing the vector of coefficients and the variance of the innovations in the moving average model; convergence code and number of iterations.

References

Pollock, D. S. G. (1999) *A Handbook of Time-Series Analysis Signal Processing and Dynamics*. Academic Press. Chapter 17. doi: 10.1016/B978-012560990-6/50002-6

Examples

```
set.seed(123)
x <- arima.sim(n=200, model=list(ma=c(0.7,-0.3)))
#sample autocovariances
a <- c(var(x), cov(x[-1], x[-200]), cov(x[-c(1,2)], x[-c(199,200)]))
#inferred coefficients and variance
acov2ma(a, init=acov2ma.init(a, maxit=10)$macoefs)
#compare with maximum-likelihood
arima(x, order=c(2,0,0), include.mean=FALSE)
```

ARIMAdec *ARIMA-Model-Based Decomposition of Time Series*

Description

This is the main function for the ARIMA-model-based decomposition of a time series.

Usage

```
ARIMAdec(x, mod, width = c(0.035, 0.035), min.modulus = 0.4,
  extend = 16, drift = FALSE, optim.tol = 1e-04, ...)
## S3 method for class 'ARIMAdec'
print(x, units = c("radians", "degrees", "pi"), digits = 4, ...)
## S3 method for class 'ARIMAdec'
plot(x, ...)
```

Arguments

x	for ARIMAdec, a univariate time series; for plot.ARIMAdec and print.ARIMAdec, an object of class ARIMAdec returned by ARIMAdec.
mod	an object of class Arima. See arima.
width	numeric of length two, width of the interval of frequencies allocated to the trend and the seasonal components (measured in radians). If a numeric of length one is passed as argument, the same width is used for both components. See roots.allocation.
min.modulus	numeric, minimum modulus of the roots assigned to the trend component. See roots.allocation.
extend	integer; if greater than zero, the series is extended by means of forecasts and backcasts based on the fitted model mod. See filtering.
drift	logical, if TRUE the intercept in the fitted model mod or an external regressor named "drift" is treated as a deterministic linear trend. See filtering.
optim.tol	numeric, the convergence tolerance to be used by optimize.
units	character, the units in which the argument of the roots are printed. units="pi" prints the argument in radians as multiples of π.
digits	numeric, the number of significant digits to be used by print.
...	further arguments to be passed to poly2acgf or to plot.tsdecFilter and print methods.

Details

This function is a wrapper to the sequence of calls to `roots.allocation`, `pseudo.spectrum`, `canonical.decomposition` and `filtering`.

Value

An object of class `ARIMAdec` containing the following: 1) `ar`: the output from {`roots.allocation`}, 2) `spectrum`: the output from {`pseudo.spectrum`}, 3) `ma`: the output from {`canonical.decomposition`}, 4) `xextended`: the series extended with backcasts and forecasts (if `extend` > `0`), 5) `filters`: the filters returned by {`filtering`}, 6) `components`: the estimated components returned by {`filtering`}.

References

Burman, J. P. (1980) 'Seasonal Adjustment by Signal Extraction'. *Journal of the Royal Statistical Society. Series A (General)*, **143**(3), pp. 321-337. doi: 10.2307/2982132

Hillmer, S. C. and Tiao, G. C. (1982) 'An ARIMA-Model-Based Approach to Seasonal Adjustment'. *Journal of the American Statistical Association*, **77**(377), pp. 63-70. doi: 10.1080/01621459.1982.10477767

See Also

`canonical.decomposition`, `filtering`, `pseudo.spectrum`, `roots.allocation`.

Examples

```
# Airlines model and monthly data
y <- log(AirPassengers)
fit <- arima(y, order=c(0,1,1), seasonal=list(order=c(0,1,1)))
dec <- ARIMAdec(y, fit, extend=72)
dec
plot(dec)

# JohnsonJohnson quarterly data
y <- log(JohnsonJohnson)
fit <- arima(y, order=c(0,1,1), seasonal=list(order=c(0,1,1)))
dec <- ARIMAdec(y, fit, extend=16)
dec
plot(dec)

# Nile annual data
# this series is better modelled as a level shift at
# observation 29 and a mean (no ARMA structure),
# here the shift is ignored for illustration of the
# decomposition of the fitted ARIMA(0,1,1) model
y <- Nile
fit <- arima(y, order=c(0,1,1))
dec <- ARIMAdec(y, fit, extend=72)
dec
plot(dec, overlap.trend=TRUE, args.trend=list(col="blue"))
```

ARMAacov *Compute Theoretical Autocovariances of an ARMA Model*

Description

Compute the theoretical autocovariances of an ARMA model.

Usage

```
ARMAacov(ar = numeric(0), ma = numeric(0), lag.max = max(p, q + 1),
  sigma2 = 1)
```

Arguments

ar	numeric vector of AR coefficients.
ma	numeric vector of MA coefficients.
lag.max	integer, maximum lag to be computed. The default is max(p, q+1), where p and q are orders of the AR and MA terms, length(ar) and length(ma), respectively.
sigma2	numeric, the variance of the innovations.

Value

A vector of autocovariances named by lag order.

Note

Based on ARMAacf.

References

Brockwell, P. J. and Davis, R. A. (1991) *Time Series: Theory and Methods*, Second Edition. Springer. doi: 10.1007/978-1-4419-0320-4

Pollock, D. S. G. (1999) *A Handbook of Time-Series Analysis Signal Processing and Dynamics*. Academic Press. Chapter 17. doi: 10.1016/B978-012560990-6/50002-6

See Also

ARMAtoMA.

Examples

```
# Autocovariances of an ARMA(2,1)
# method 1: using ARMAacov()
a1 <- ARMAacov(ar=c(0.8,-0.6), ma=0.4, lag.max=10)

# method 2: upon the coefficients of the infinite MA representation
psi <- c(1, ARMAtoMA(ar=c(0.8,-0.6), ma=0.4, lag.max=50))
a2 <- c(sum(psi^2), rep(0, length(a1)-1))
for (i in seq_along(a2[-1]))
  a2[i+1] <- sum(psi[seq_len(length(psi)-i)] * psi[-seq_len(i)])

# for a high enough number of 'psi' coefficients
# both methods are equivalent
all.equal(a1, a2, check.names=FALSE)
#[1] TRUE
```

---

canonical.decomposition

Canonical Decomposition

Description

Given the partial fraction decomposition of the pseudo-spectrum, the canonical decomposition allocates the variance of each component so that the variance of the irregular is maximised. Then, the coefficients of the numerators in the pseudo-spectrum (relationship given in `pseudo.spectrum`) are converted into the MA coefficients of the model for each component by means of `acgf2poly`.

Usage

```
canonical.decomposition(num.trend, den.trend,
  num.trans, den.trans, num.seas, den.seas, quotient, optim.tol = 1e-04, ...)
## S3 method for class 'tsdecCanDec'
print(x, units = c("radians", "degrees", "pi"), digits = 4, ...)
```

Arguments

`num.trend`	numeric vector, the coefficients of the MA polynomial related to the trend component in the relationship given in `pseudo.spectrum`.
`den.trend`	numeric vector, the coefficients of the AR polynomial related to the trend component in the relationship given in `pseudo.spectrum`.
`num.trans`	numeric vector, the coefficients of the MA polynomial related to the transitory component in the relationship given in `pseudo.spectrum`.
`den.trans`	numeric vector, the coefficients of the AR polynomial related to the transitory component in the relationship given in `pseudo.spectrum`.
`num.seas`	numeric vector, the coefficients of the MA polynomial related to the seasonal component in the relationship given in `pseudo.spectrum`.

`den.seas`	numeric vector, the coefficients of the AR polynomial related to the seasonal component in the relationship given in `pseudo.spectrum`.
`quotient`	numeric vector, the quotient of the polynomial division of the polynomials in the LHS of the relationship given in `pseudo.spectrum`. (Different from zero only when the degree of the MA polynomial is equal or higher than the degree of the AR polynomial in the fitted model).
`optim.tol`	numeric, the convergence tolerance to be used by `optimize`.
`units`	character, the units in which the argument of the roots are printed. `units="pi"` prints the argument in radians as multiples of π.
`x`	an object of class `tsdecCanDec` returned by `canonical.decomposition`.
`digits`	numeric, the number of significant digits to be used by `print`.
`...`	further arguments to be passed to `poly2acgf` or `print`.

Value

An object of class `tsdecCanDec` containing the MA coefficients of the ARIMA models obtained for the unobserved components (e.g., trend, seasonal) and the variance of the corresponding disturbance terms.

References

Burman, J. P. (1980) 'Seasonal Adjustment by Signal Extraction'. *Journal of the Royal Statistical Society. Series A (General)*, **143**(3), pp. 321-337. doi: 10.2307/2982132.

Hillmer, S. C. and Tiao, G. C. (1982) 'An ARIMA-Model-Based Approach to Seasonal Adjustment'. *Journal of the American Statistical Association*, **77**(377), pp. 63-70. doi: 10.1080/01621459.1982.10477767.

See Also

`acgf2poly`, `pseudo.spectrum`, `optimize`.

`compare.acf` *Compare ACF of Theoretical, Estimator and Empirical Component*

Description

Compute the AutoCorrelation functions of the following elements: the theoretical ARMA model of each component, the estimator for each component, the filtered or estimated components.

Usage

```
compare.acf(x, mod, lag.max = 12, ...)
## S3 method for class 'tsdecAcf'
plot(x, component = c("trend", "transitory", "seasonal"), ci = 0.95,
 ci.type = c("ma", "white"), ci.class = c("estimator", "theoretical", "empirical"),
  plot = TRUE, ...)
```

Arguments

x	for compare.acf, an object of class ARIMAdec; for plot.tsdecAcf, an object of class tsdecAcf returned by compare.acf.
mod	the object of class Arima decomposed in x. See arima.
lag.max	maximum lag at which to calculate the autocorrelations.
component	a character, the label of the component for which the ACF is to be obtained.
ci	coverage probability for confidence interval. If this is zero or negative, confidence intervals are not computed
ci.type	a character, the type of confidence interval. See details.
ci.class	a character, the element that is taken as reference to computed the confidence intervals. Ignored if ci.class='white'.
plot	logical, if TRUE the ACF is plotted.
...	further arguments to be passed to acf and plot.

Details

The ACF is obtained upon the stationary transformation of the models for the components and the estimators; i.e., non-stationary roots (if any) are removed from the AR polynomials. The estimated components are also transformed according to the polynomials xarpolys.nonstationary that render the signals stationary.

Argument ci.type behaves similarly to the same argument in plot.acf. If ci.type = "white", the confidence bands are fixed to $t_{\alpha/2}/\sqrt{(n)}$, where n is the number of observations in the fitted model model. If ci.type = "ma", confidence bands are obtained upon Bartlett's approximations for the standard deviations of the autocorrelations.

Value

compare.acf returns the ACF of the components, respectively for their theoretical ARMA model, estimator and estimates.

plot.tsdecAcf displays a plot and returns a invisible copy of a matrix containing the confidence intervals.

See Also

ARIMAdec.

Examples

```
# Airlines model and monthly data
y <- log(AirPassengers)
fit <- arima(y, order=c(0,1,1), seasonal=list(order=c(0,1,1)))
dec <- ARIMAdec(y, fit, extend=72)
cacf <- compare.acf(x = dec, mod=fit, lag.max=24)
plot(cacf, component="seasonal")
# unexpected discrepancy between the ACF of the estimator and the
# ACF of the empirical signal
```

```
plot(cacf, component="trend")

# Nile time series
y <- Nile
fit <- arima(y, order=c(0,1,1))
dec <- ARIMAdec(y, fit, extend=16)
cacf <- compare.acf(x = dec, mod=fit, lag.max=24)
plot(cacf, component="trend")
```

`filtering` *Double-Sided Symmetric Linear Filter*

Description

Double-sided symmetric linear filter.

Usage

```
filtering(x, mod,
  trend = list(ar=1, ma=1, sigma2=NULL),
  transitory = list(ar=1, ma=1, sigma2=NULL),
  seasonal = list(ar=1, ma=1, sigma2=NULL),
  irregular.sigma2 = NULL,
  extend = 16, drift = FALSE)
dsfilter(x, w, mod, extend = 16)
```

Arguments

`x`	a univariate time series.
`mod`	an object of class `Arima`. See `arima`.
`trend`	a list containing the coefficients and variance of the ARIMA model related to the trend component.
`transitory`	a list containing the coefficients and variance of the ARIMA model related to the transitory component.
`seasonal`	a list containing the coefficients and variance of the ARIMA model related to the seasonal component.
`irregular.sigma2`	numeric, variance of the irregular component. If `NULL`, the estimate of the irregular component is not computed.
`extend`	integer; if greater than zero, the series is extended by means of forecasts and backcasts based on the fitted model `mod`.
`drift`	logical, if `TRUE` the `intercept` in the fitted model `mod` or an external regressor named `"drift"` is treated as a deterministic linear trend.
`w`	a vector of filter coefficients (one side).

Details

These functions perform the convolution of the time series and the double-sided symmetric filter. They perform:

```
stats::filter(c(rep(0, n-1), x, rep(0, n-1)),
+ filter=c(rev(w[-1]), w), method="convolution", sides=1)
```

where `n` is `length(x)`.

The design of the filter in the ARIMA-model-based decomposition procedure relies on the following result. The minimum mean squared error estimator of the component is given by the ACGF of the model:

$$\theta(L)x_t = \phi_n(L)\theta_s(L)a_t\,,$$

where $\theta(L)$ is the MA of the model fitted to the observed data, $\theta_s(L)$ is the MA of the component (signal) to be estimated and $\phi_n(L)$ is the product of the AR polynomials of the remaining components. The estimate of the signal, $\hat{s}_t$, is obtained by means of a double-sided symmetrical filter where the weights, w, are the theoretical autocovariances of the model above:

$$\hat{s}_t = \sum_{i=-\infty}^{\infty} w_i x_{t-i}\,.$$

Value

`filtering` returns a list of class `tsdecFilter` containing the series extended with forecasts (if `extend > 0`) (based on the ARMA model given as input), the weights of one side of the filter for each component and the corresponding estimate of the components.

`dsfilter` returns the filtered time series.

See Also

`ARIMAdec`, `filter`.

`partial.fraction` *Partial Fraction Decomposition*

Description

Partial fraction decomposition of the pseudo-spectrum of a fitted ARIMA model.

Usage

```
partial.fraction(numerator, den.trend, den.transitory, den.seasonal)
```

Arguments

numerator	numeric vector containing the coefficients of the numerator of the ratio of polynomials to be decomposed (numerator in the left-hand-side of the relationship given in pseudo.spectrum).
den.trend	numeric vector containing the coefficients of the denominator in the partial fraction related to the trend component.
den.transitory	numeric vector containing the coefficients of the denominator in the partial fraction related to the transitory component.
den.seasonal	numeric vector containing the coefficients of the denominator in the partial fraction related to the seasonal component.

Value

A list containing the system of equations which is solved and the numerators of the partial fractions related, respectively, to the trend, transitory and seasonal components.

See Also

pseudo.spectrum.

---

plot.tsdecFilter *Plot Method for* tsdecFilter *Objects*

Description

Plot the time series containing the components in a tsdecFilter object.

Usage

```
## S3 method for class 'tsdecFilter'
plot(x, select = colnames(X),
  overlap.trend = FALSE, args.trend = list(col = "black"),
  set.pars = list(mar = c(0, 3, 0, 3), oma = c(4, 0, 2, 0), mfrow = c(nplot, 1)),
  main = NULL, range.bars = TRUE, ...,
  col.range = "light gray",
  args.xlab = list(text = "time", side = 1, line = 2),
  args.ylab = list(side = 3, adj = 0, line = -1),
  xaxis.line = -0.5)
```

Arguments

x	an object of class tsdecFilter returned by filtering.
select	character vector with the labels of the series to be plot. Allowed values are c("observed", "trend", "transitory", "seasonal", "sadj", "irregular").
overlap.trend	logical, if TRUE the trend component is plot over the observed data; otherwise, the trend is plot separately.

args.trend	a list containing the arguments to be passed to lines. If overlap.trend=TRUE, these options are used to plot the trend; otherwise, it is ignored.
set.pars	settings for par(.) when setting up the plot.
main	plot main title.
range.bars	logical indicating if each plot should have a bar at its right side which are of equal heights in user coordinates. The same as in plot.stl.
...	further arguments passed to plot.
col.range	colour to be used for the range bars, if plotted. Note this appears after ... and so cannot be abbreviated.
args.xlab	arguments to be passed to mtext when setting the title for the x axis.
args.ylab	arguments to be passed to mtext when setting the title for the y axis.
xaxis.line	the number of lines into the margin at which the x axis line will be drawn.

Details

This function is based on plot.stl.

See Also

filtering.

---

polyeval *Polynomial Operations and Utilities*

Description

Polynomial operations and utilities.

Usage

```
polystring(x, varchar = "x", brackets = FALSE, ndec = 2, emptychar = "")
polyeval(p, x)
polyprod(x, y, tol = 1.490116e-08)
polydiv(x, y)
roots2poly(x)
```

Arguments

x	numeric vector containing the coefficients of the polynomial (in increasing order and without gaps). For polyeval, this is the point at which the polynomial is to be evaluated. For roots2poly, this is a numeric vector containing the roots of the polynomial.
y	numeric vector containing the coefficients of the polynomial (in increasing order and without gaps).

p	numeric vector containing the coefficients of the polynomial (in increasing order and without gaps).
varchar	character string, the label to be printed representing the variable of the polynomial, defaults to "x".
brackets	logical, if TRUE the polynomial is printed within parentheses.
ndec	integer, coefficients are rounded up to this number of decimals.
emptychar	the character string to be printed if the polynomial is empty.
tol	a numeric, tolerance below which coefficients are set to zero.

Details

polystring returns a string of a numeric vector in the format of a polynomial.

polyeval evaluates the polynomial defined in the vector of coefficients p at the point x.

polyprod performs polynomial multiplication.

polydiv performs polynomial division (returning the quotient and the remainder).

roots2poly computes the coefficients of a polynomial from its roots.

Note

polyprod is based on convolve; it is equivalent to convolve(x, rev(y), type="open").

roots2poly is based on poly.from.zeros() in package **polynom**.

See Also

polyroot.

Examples

```
# print a fitted ARMA model
set.seed(123)
y <- arima.sim(n=120, model=list(ar=c(0.8, -0.3), ma=0.6))
fit <- arima(y, order=c(2,0,1), include.mean=FALSE)
cat(paste0(
polystring(c(1, -fit$model$phi), brackets=TRUE, ndec=3), "y_t = ",
polystring(c(1, fit$model$theta), brackets=TRUE, ndec=3), "e_t\n"))

# convert roots to coefficients
p <- c(1, 0.8, -0.3)
cat(polystring(p))
r <- polyroot(p)
roots2poly(r)
```

pseudo.spectrum *Pseudo-Spectrum of an ARIMA Model*

Description

Compute the polynomials in the numerators of a partial fraction decomposition of the pseudo-spectrum in an ARIMA model. The polynomials are in terms of the variable $2\cos\omega$, with $\omega \in [0, 2\pi]$.

Usage

```
pseudo.spectrum(mod, ar)
## S3 method for class 'tsdecPSP'
print(x, ...)
```

Arguments

mod	an object of class Arima, the fitted model.
ar	an object of class tsdecARroots returned by roots.allocation.
x	an object of class tsdecPSP returned by pseudo.spectrum.
...	further arguments to be passed to print.

Details

The coefficients of the ARIMA models for each component (e.g., trend, seasonal) are obtained from the following relationship.

$$\sigma^2 \frac{\theta(B)\theta(F)}{\phi(B)\phi(F)} = \sigma_a^2 \frac{\theta_T(B)\theta_T(F)}{\phi_T(B)\phi_T(F)} + \sigma_b^2 \frac{\theta_S(B)\theta_S(F)}{\phi_S(B)\phi_S(F)} + \sigma_e^2 \,,$$

where B is the backshift operator and $F = B^{-1}$ is the forward operator. Each term in the right-hand-side is related to the ARIMA models of each one of the unobserved components.

pseudo.spectrum computes the symmetric polynomials of the type $\varphi(B)\varphi(F)$ for the polynomials in the left-hand-side LHS (based on the fitted model) and for the polynomials in the denominators of the right-hand-side RHS (based on the allocation of roots of the fitted AR polynomial, roots.allocation). Then coefficients in the numerators of the RHS are obtained by means of partial.fraction .To do so the terms in the RHS are multiplied by the denominator in the LHS; then, the coefficients of the numerators in the RHS are obtained by equating the coefficients of the same order on both sides of the relationship (the orders of the unknown polynomials are set to one degree lower than those polynomials of the corresponding denominator).

Value

A list of class tsdecPSP containing: the quotient of the polynomial division (if the degree of the numerator in the LHS is equal or higher than the degree of the denominator); the coefficients of total polynomials (numerator and denominator in the LHS) and the denominators in the RHS.

References

Burman, J. P. (1980) 'Seasonal Adjustment by Signal Extraction'. *Journal of the Royal Statistical Society. Series A (General)*, **143**(3), pp. 321-337. doi: 10.2307/2982132

Hillmer, S. C. and Tiao, G. C. (1982) 'An ARIMA-Model-Based Approach to Seasonal Adjustment'. *Journal of the American Statistical Association*, **77**(377), pp. 63-70. doi: 10.1080/01621459.1982.10477767

See Also

arima, partial.fraction, roots.allocation.

`roots.allocation` *Allocation of Autoregressive Roots*

Description

Allocate the roots of the autoregressive polynomial from a fitted ARIMA model to trend, transitory and seasonal components.

Usage

```
roots.allocation(x, width = c(0.035, 0.035), min.modulus = 0.4)
## S3 method for class 'tsdecARroots'
plot(x, xlim, ylim, ...)
## S3 method for class 'tsdecARroots'
print(x, units = c("radians", "degrees", "pi"), digits = 4, ...)
```

Arguments

x	for `roots.allocation`, an object of class `Arima` (see arima); for `print.tsdecARroots` and `plot.tsdecARroots`, an object of class `tsdecARroots` returned by `type="roots.allocation"`.
width	numeric of length two, width of the interval of frequencies allocated to the trend and the seasonal components (measured in radians). If a numeric of length one is passed as argument, the same width is used for both components.
min.modulus	numeric, minimum modulus of the roots assigned to the trend component.
xlim	optional numerics, lower and upper limits of the x-axis.
ylim	optional numerics, lower and upper limits of the y-axis.
units	character, the units in which the argument of the roots are printed. `units="pi"` prints the argument in radians as multiples of π.
digits	numeric, the number of significant digits to be used by print.
...	further arguments to be passed to plot or print.

Details

The roots related to cycles with frequency within the range [0, `width[1]`] are allocated to the trend or transitory component. In particular,if the modulus is below `min.modulus`, then they are are allocated to the transitory component, otherwise to the trend.

The seasonal frequencies are defined as $\omega_j = 2\pi j/S$, for $j = 1, ..., S-1$, where S is the periodicity of the data (e.g., $S = 4$ in quarterly data and $S = 12$ in monthly data). Roots related to cycles of frequency within the range [ω_j−`seasonal.width`, ω_j+`seasonal.width`] are assigned to the seasonal component.

Value

`roots.allocation` returns a list of class `tsdecARroots`. `plot.tsdecARroots` displays the roots in the complex plane and `print.tsdecARroots` shows a summary.

Index