Package: msarima (via r-universe)

November 13, 2026

Title Multi-Seasonal ARIMA Models

Version 0.1.0

Description Fit, forecast and simulate autoregressive integrated moving average (ARIMA) models with more than one seasonal component, following Lisi and Grigoletto (2025) <doi:10.1002/for.3283>. Series observed at high frequency often display several seasonal cycles at once, such as the daily and the weekly pattern of hourly data, which a single seasonal ARIMA operator cannot represent. Estimation is performed by conditional maximum likelihood through numerical optimisation, with support for fixed coefficients and a constant term. Multi-seasonal Ljung-Box and Monti portmanteau tests, following Grigoletto and Lisi (2026) <doi:10.48550/arXiv.2609.32381>, seasonal correlograms, forecasts with prediction intervals, and simulation from a model specification or from a fitted model are also provided.

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0), forecast

Imports polynom, stats

Suggests ggplot2, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/danigiro/msarima, https://danigiro.github.io/msarima/

BugReports https://github.com/danigiro/msarima/issues

LazyData true

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

Date/Publication 2026-11-13 11:10:00 UTC

RemoteUrl https://github.com/danigiro/msarima

RemoteRef HEAD

RemoteSha 67dac27b64068abd1483c557c776f70fb40e35ef

Contents

msarima-package 2
forecast.msarima 3
MAD_m40 4
msarima 6
msarima-methods 9
msarima.sim 11
msportmanteau 13
sacf 14
simulate.msarima 16

Index **18**

`msarima-package` *msarima: Multi-Seasonal ARIMA Models*

Description

Fit, forecast and simulate autoregressive integrated moving average (ARIMA) models with more than one seasonal component, following Lisi and Grigoletto (2025) doi:10.1002/for.3283. Series observed at high frequency often display several seasonal cycles at once, such as the daily and the weekly pattern of hourly data, which a single seasonal ARIMA operator cannot represent. Estimation is performed by conditional maximum likelihood through numerical optimisation, with support for fixed coefficients and a constant term. Multi-seasonal Ljung-Box and Monti portmanteau tests, following Grigoletto and Lisi (2026) doi:10.48550/arXiv.2609.32381, seasonal correlograms, forecasts with prediction intervals, and simulation from a model specification or from a fitted model are also provided.

Author(s)

Maintainer: Daniele Girolimetto <daniele.girolimetto@unipd.it> (ORCID) [copyright holder]

Authors:

- Francesco Lisi <francesco.lisi@unipd.it> (ORCID)
- Matteo Grigoletto <matteo.grigoletto@unipd.it> (ORCID)

See Also

Useful links:

- https://github.com/danigiro/msarima
- https://danigiro.github.io/msarima/
- Report bugs at https://github.com/danigiro/msarima/issues

`forecast.msarima` *Forecast from a Multi-Seasonal ARIMA Model*

Description

Returns forecasts and prediction intervals for a fitted multi-seasonal ARIMA model.

Usage

```
## S3 method for class 'msarima'
forecast(object, h = 10, level = c(80, 95), fan = FALSE, ...)
```

Arguments

`object`	An object of class `"msarima"`.
`h`	Number of periods to forecast. Default is 10.
`level`	Confidence levels for the prediction intervals, in percent. Default is `c(80, 95)`.
`fan`	If `TRUE`, `level` is set to `seq(51, 99, by = 3)`, which is suitable for fan plots. Default is `FALSE`.
`...`	Additional arguments (currently not used).

Details

Point forecasts are computed by running the model recursion forward, setting the future innovations to zero. Differencing and the constant term, when present, are taken from the fitted model, so the forecasts are on the scale of the original series.

Prediction intervals assume Gaussian innovations. The forecast variance V_h is obtained from the MA(∞) representation of the model.

Value

An object of class `"forecast"` containing the following elements:

`method`	The name of the forecasting method (same as `object$method`).
`model`	The fitted `"msarima"` model object.
`level`	The confidence levels for prediction intervals (as percentages).
`mean`	Point forecasts as a time series (ts or msts object).

`lower`	Lower limits for prediction intervals. Matrix with one column for each confidence level specified in `level`.
`upper`	Upper limits for prediction intervals. Matrix with one column for each confidence level specified in `level`.
`x`	The original time series.
`series`	The name of the series.
`fitted`	Fitted values from the model.
`residuals`	Residuals from the model.

See Also

`msarima`, `simulate.msarima`

Examples

```
# Simulate and fit model
set.seed(123)
y <- msarima.sim(nsim = 200,
                 order = c(1, 0, 0),
                 seasonal = list(c(1, 0, 0), c(0, 0, 1)),
                 seasonal.period = c(4, 11),
                 ar = c(0.5, 0.5), ma = 0.3, sigma2 = 1)

fit <- msarima(y, order = c(1, 0, 0),
               seasonal = list(c(1, 0, 0), c(0, 0, 1)),
               seasonal.period = c(4, 11))

# Forecast 20 steps ahead with default confidence levels (80%, 95%)
fc <- forecast(fit, h = 20)
print(fc)

# Forecast with custom prediction intervals
fc2 <- forecast(fit, h = 20, level = c(90, 95, 99))
```

MAD_m40 *Hourly Traffic Volume on the M-40 Ring Road, Madrid*

Description

Number of vehicles per hour recorded by a single traffic sensor on the M-40, the outer ring road of Madrid, over the eight weeks from 3 February to 29 March 2024.

The series shows the two seasonal patterns that `msarima` is meant for: a daily cycle of period 24, with the morning and evening commuting peaks, and a weekly cycle of period 168, with markedly lower volumes at the weekend.

Usage

```
MAD_m40
```

Format

An `msts` object of length 1344 (56 days, or eight complete weeks, of hourly data) with seasonal periods 24 and 168. The first observation is midnight of Saturday 3 February 2024 and the last is 11 p.m. of Friday 29 March 2024.

Details

The raw measurements are half-hourly counts from sensor 3753, recorded between 1 February and 31 May 2024 and summed in pairs to obtain hourly volumes; the eight weeks kept here are the window studied by Grigoletto and Lisi (2026), who chose this sensor for having the fewest missing observations in the network. A few of the underlying half-hourly counts had been interpolated by the provider, so 3 of the hourly values are not whole numbers. There are no missing values and no gaps in the hourly grid.

The script that builds the object from the raw file is available at https://github.com/danigiro/msarima/blob/main/data-raw/MAD_m40.R.

Source

Madrid City Council open data portal, https://datos.madrid.es.

References

Grigoletto, M. and Lisi, F. (2026). Portmanteau goodness-of-fit tests in the presence of simple, mixed, and multiple seasonal autocorrelation. *arXiv*. doi:10.48550/arXiv.2609.32381

Examples

```
# The daily and weekly cycles
plot(window(MAD_m40, end = 4), ylab = "vehicles per hour")

# A multi-seasonal ARIMA with one daily and one weekly difference

fit <- msarima(MAD_m40,
               order = c(1, 0, 2),
               seasonal = list(c(0, 1, 1), c(0, 1, 1)))
summary(fit)
plot(forecast(fit, h = 48))
```

`msarima` *Fit a Multi-Seasonal ARIMA Model*

Description

Fits an ARIMA model with one or more seasonal components to a univariate time series. This function extends the standard ARIMA framework to time series with several seasonal patterns, such as daily and weekly seasonality in hourly data, or weekly and yearly seasonality in daily data.

Usage

```
msarima(
  y,
  order = c(0, 0, 0),
  seasonal = list(),
  seasonal.period = NULL,
  fixed = NULL,
  include.constant = FALSE,
  optim_control = list(maxit = 200)
)
```

Arguments

`y`	A numeric vector or time series object (`ts` or `msts`) containing the data to be modelled. If `y` is a `ts` object and `seasonal.period` is not supplied, its frequency is used as the first seasonal period; if `y` is an `msts` object, its seasonal periods are used. Missing values are not allowed.
`order`	A numeric vector of length 3 specifying the non-seasonal part of the ARIMA model: the AR order (p), the degree of differencing (d), and the MA order (q). Default is `c(0, 0, 0)`.
`seasonal`	A list of numeric vectors, each of length 3, specifying the seasonal components. Each vector gives the seasonal AR order (P), the seasonal differencing order (D), and the seasonal MA order (Q) of one component. For example, `list(c(1, 0, 1), c(0, 1, 1))` specifies two seasonal components. Use `list()` for a model without seasonal components. Default is `list()`.
`seasonal.period`	A numeric vector of seasonal periods, one for each element of `seasonal`, for example `c(7, 30)` for weekly and monthly seasonality. If `NULL`, the periods are taken from `y` when it is a `ts` or `msts` object. Periods can be given in any order but must be distinct. Default is `NULL`.
`fixed`	A numeric vector with one entry per parameter, used to hold some of them at a given value: only the `NA` entries are estimated. The parameters are ordered as non-seasonal AR, seasonal AR (one block per seasonal component), non-seasonal MA, seasonal MA (one block per seasonal component), sigma, and the constant term when `include.constant = TRUE`. Default is `NULL`, meaning that all parameters are estimated.

include.constant
: Should the model include a constant term? The constant is the intercept of the model equation, not the mean of the series: for a stationary model the two are related by $\mu = c/\phi(1)$, where c is the intercept and $\phi(1)$ is the full AR polynomial (seasonal components included) evaluated at 1. For example, for an AR(1) model $\mu = c/(1-\phi_1)$. A constant cannot be combined with differencing and is dropped, with a warning, when `d` or any `D` is non-zero. Default is `FALSE`.

optim_control
: A list of control parameters passed to `optim`. Default is `list(maxit = 200)`.

Details

The model is estimated by conditional maximum likelihood: the residuals are computed recursively and the resulting Gaussian log-likelihood is maximised numerically with `optim`, using the BFGS algorithm.

When `d` or any `D` is non-zero, the series is differenced before estimation and the ARMA part is estimated on the differenced data. The first `d + sum(D * S)` observations are therefore not used, which is why the residuals and the fitted values start with as many `NA` values: these observations have no differenced value to predict, so no residual is defined for them under the conditional likelihood.

Standard errors are computed from the inverse of the observed information matrix (negative Hessian). If the Hessian is singular, a warning is issued and the standard errors are set to `NA`.

Value

An object of class `"msarima"`, which is a list containing:

order
: The non-seasonal ARIMA order (p, d, q).

seasonal
: List of seasonal orders (P, D, Q), one vector per seasonal component.

seasonal.period
: Vector of seasonal periods.

coef
: Named vector of estimated coefficients: the AR, SAR, MA and SMA parameters, followed by the intercept if `include.constant = TRUE`.

var_coef
: Variance-covariance matrix of the coefficient estimates, with the same dimension and order as `coef`. Rows and columns of fixed parameters are `NA`.

mask
: Logical vector with the same length and order as `coef`, `TRUE` for fixed parameters.

loglik
: Log-likelihood of the fitted model.

aic
: Akaike Information Criterion.

bic
: Bayesian Information Criterion.

aicc
: Corrected Akaike Information Criterion.

sigma2
: Estimated innovation variance.

x
: The original time series.

fitted
: One-step-ahead fitted values, on the scale of the original series.

residuals
: Model residuals. Together with `fitted`, the first `d + sum(D * S)` values are `NA`, since differencing leaves no observations to predict for them.

method	Character string describing the fitted model, e.g. "mSARIMA(1,0,0)(1,0,0)[4]".
series	Name of the series.
nobs	Number of observations in the original series.
n.used	Number of observations used in estimation (after differencing).
call	The matched call.
convergence	Convergence code from optim (0 indicates successful convergence).

Note

- The series must have at least 10 observations, both before and after differencing.
- Seasonal periods must be at least 2 and at most half the length of the series.
- If y is a ts or msts object, x, fitted and residuals keep the same time series structure. If y is a plain numeric vector, they are returned as ts with one seasonal period, or as msts with several.

Examples

```
# Example 1: ARIMA with two seasonal components
set.seed(123)
y <- msarima.sim(nsim = 200,
                 order = c(1, 0, 0),
                 seasonal = list(c(1, 0, 0), c(0, 0, 1)),
                 seasonal.period = c(4, 11),
                 ar = c(0.5, 0.5), ma = 0.3, sigma2 = 1)

fit <- msarima(y, order = c(1, 0, 0),
               seasonal = list(c(1, 0, 0), c(0, 0, 1)),
               seasonal.period = c(4, 11))
print(fit)
summary(fit)

# Example 2: msts input, seasonal periods detected automatically
y_msts <- msts(y, seasonal.periods = c(4, 11))
fit_msts <- msarima(y_msts, order = c(1, 0, 0),
                    seasonal = list(c(1, 0, 0), c(0, 0, 1)))

# Example 3: differencing
fit_diff <- msarima(cumsum(y), order = c(1, 1, 0),
                    seasonal = list(c(0, 0, 1)),
                    seasonal.period = 4)
head(residuals(fit_diff))  # the first residual is NA

# Example 4: fixing parameters
# Note: the length of 'fixed' must match AR + SAR + MA + SMA + sigma
fit_fixed <- msarima(y, order = c(1, 0, 0),
                     seasonal = list(c(1, 0, 0), c(0, 0, 1)),
                     seasonal.period = c(4, 11),
                     fixed = c(0.5, NA, NA, NA))
AIC(fit); AIC(fit_fixed)
```

`msarima-methods` *Methods for msarima Objects*

Description

Methods for extracting information from and displaying fitted multi-seasonal ARIMA models.

Usage

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

## S3 method for class 'msarima'
summary(object, ...)

## S3 method for class 'summary_msarima'
print(x, digits = 4, signif.stars = TRUE, ...)

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

## S3 method for class 'msarima'
fitted(object, ...)

## S3 method for class 'msarima'
residuals(object, ...)

## S3 method for class 'msarima'
logLik(object, ...)

## S3 method for class 'msarima'
AIC(object, ...)

## S3 method for class 'msarima'
BIC(object, ...)
```

Arguments

`x`, `object`	An object of class `"msarima"`.
`...`	Additional arguments (currently not used).
`digits`	Number of digits to display. Default is 4.
`signif.stars`	Logical; if `TRUE`, significance stars are printed. Default is `TRUE`.

Details

`print` Prints a concise summary of the fitted model including coefficients and information criteria.

`summary` Produces a detailed summary including coefficient estimates, standard errors, z-values, and p-values.

`coef` Extracts the estimated model coefficients.

`fitted` Extracts the fitted values.

`residuals` Extracts the model residuals.

`logLik` Extracts the log-likelihood value.

`AIC` Calculates the Akaike Information Criterion.

`BIC` Calculates the Bayesian Information Criterion.

Value

`print` Returns the input object invisibly.

`summary` Returns an object of class `"summary_msarima"` containing:

- `call`: The matched call.
- `series`: Name of the time series.
- `order`: Non-seasonal ARIMA order.
- `seasonal`: List of seasonal orders (P, D, Q), one vector per seasonal component.
- `seasonal.period`: Vector of seasonal periods.
- `coefficients`: Matrix with estimates, standard errors, z-values, and p-values.
- `sigma2`: Estimated innovation variance.
- `loglik`: Log-likelihood of the fitted model.
- `aic`: Akaike Information Criterion.
- `bic`: Bayesian Information Criterion.
- `aicc`: Corrected Akaike Information Criterion.
- `nobs`: Number of observations in the original series.
- `n.used`: Number of observations used in estimation.
- `method`: Character string describing the fitted model, e.g. `"mSARIMA(1,0,0)(1,0,0)[4]"`.
- `convergence`: Convergence code from optimization.

`coef` A named numeric vector of coefficients.

`fitted` A time series object of fitted values.

`residuals` A time series object of residuals.

`logLik` An object of class `"logLik"`.

`AIC` The AIC value.

`BIC` The BIC value.

See Also

`msarima`

Examples

```
# Fit a model
set.seed(123)
y <- msarima.sim(nsim = 200,
                 order = c(1, 0, 0),
                 seasonal = list(c(1, 0, 0), c(0, 0, 1)),
                 seasonal.period = c(4, 11),
                 ar = c(0.5, 0.5), ma = 0.3, sigma2 = 1)

fit <- msarima(y, order = c(1, 0, 0),
               seasonal = list(c(1, 0, 0), c(0, 0, 1)),
               seasonal.period = c(4, 11))

# Use methods
print(fit)
summary(fit)
coef(fit)
head(fitted(fit))
head(residuals(fit))
logLik(fit)
AIC(fit)
BIC(fit)
```

`msarima.sim` *Simulate from a Multi-Seasonal ARIMA Model (Direct Interface)*

Description

Simulates a time series from a multi-seasonal ARIMA model with specified parameters. This function provides a direct interface for simulation without requiring a fitted model object.

Usage

```
msarima.sim(
  nsim,
  order = c(0, 0, 0),
  seasonal = list(),
  seasonal.period = NULL,
  ar = NULL,
  ma = NULL,
  sigma2 = 1,
  burn = NULL,
  intercept = 0
)
```

Arguments

`nsim`	Length of the simulated series (required).
`order`	A numeric vector of length 3 specifying the non-seasonal part of the ARIMA model: the AR order (p), the degree of differencing (d), and the MA order (q). Default is `c(0, 0, 0)`.
`seasonal`	A list of numeric vectors, each of length 3, specifying the seasonal components. Each vector gives the seasonal AR order (P), the seasonal differencing order (D), and the seasonal MA order (Q) of one component. Use `list()` for a model without seasonal components. Default is `list()`.
`seasonal.period`	A numeric vector of seasonal periods, one for each element of `seasonal`. Periods can be given in any order but must be distinct. Default is `NULL`.
`ar`	Numeric vector of AR coefficients, non-seasonal first and then one block per seasonal component. Its length must match the total AR order implied by `order` and `seasonal`. Default is `NULL`.
`ma`	Numeric vector of MA coefficients, ordered as `ar`. Its length must match the total MA order implied by `order` and `seasonal`. Default is `NULL`.
`sigma2`	Innovation variance. Default is 1.
`burn`	Number of burn-in observations to discard. If `NULL`, it is set to `max(100, 2 * kk)`, where `kk` is the largest lag in the model. Default is `NULL`.
`intercept`	Constant term of the model equation, that is μ in $y_t = \mu + \sum \phi_i y_{t-i} + \sum \theta_j e_{t-j} + e_t$. This is the intercept, not the mean of the process: for a stationary model the mean is `intercept / prod(1 - ar)`. Default is 0.

Value

A time series object of length `nsim`: `ts` with one seasonal period, `msts` with several.

See Also

`msarima`, `simulate.msarima`

Examples

```
# Simulate data
set.seed(123)
y <- msarima.sim(nsim = 200,
                 order = c(1, 0, 0),
                 seasonal = list(c(1, 0, 0), c(0, 0, 1)),
                 seasonal.period = c(4, 11),
                 ar = c(0.5, 0.5), ma = 0.3, sigma2 = 1)
```

`msportmanteau` *Multi-Seasonal Portmanteau Tests*

Description

Multi-seasonal generalisation of the Ljung-Box and Monti portmanteau tests: residual autocorrelation is checked at low-order and seasonal lags at once, for an arbitrary number of seasonal periods.

Usage

```
msportmanteau(
  x,
  type = c("Ljung-Box", "Monti"),
  season = NULL,
  nlag = 5,
  lags_short = 0,
  npar = NULL
)
```

Arguments

`x`	A numeric vector, a `ts`/`msts` object, or a fitted msarima model (in which case the model residuals are used).
`type`	Either `"Ljung-Box"` (default), which builds the statistic from the autocorrelations, or `"Monti"`, which uses the partial autocorrelations. Partial matching is allowed.
`season`	Numeric vector of seasonal periods to test, of any length. If `NULL` (default) the periods are taken from the `msts`/`ts` attributes of `x`, or from the fitted model.
`nlag`	Numeric vector giving the number of seasonal multiples of each period to test, i.e. lags `season * (1:nlag)`. A single value is recycled across `season`; otherwise give one value per period, e.g. `c(5, 3)`. Default `5`.
`lags_short`	Number of low-order, non-seasonal lags `1:lags_short` added to the seasonal ones. Default `0` (none).
`npar`	Number of estimated model parameters. Default `0` for a raw series; for a fitted msarima model it defaults to the number of estimated (non-fixed) AR, seasonal AR, MA and seasonal MA coefficients.

Details

The lag set tested is `sort(unique(c(1:lags_short, outer(season, 1:nlag))))`, the same construction as in the original two-season versions of these tests, generalised to any number of periods.

As in sacf, leading and trailing `NA`s (such as those produced by differencing in msarima) are dropped (stats::na.contiguous) before computing the statistic, and the number of observations used is the length of the series that remains.

Value

An object of class "htest", with:

statistic	the test statistic ("X-squared").
parameter	the degrees of freedom ("df"), the number of lags tested minus npar.
p.value	the p-value, from the upper tail of a chi-squared distribution with parameter degrees of freedom.
method	a character string describing the test.
data.name	a character string naming the data.
lag	the lags actually tested.

References

Grigoletto, M. and Lisi, F. (2026). Portmanteau goodness-of-fit tests in the presence of simple, mixed, and multiple seasonal autocorrelation. *arXiv*. doi:10.48550/arXiv.2609.32381

Ljung, G. M. and Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika*, 65(2), 297-303.

Monti, A. C. (1994). A proposal for a residual autocorrelation test in linear models. *Biometrika*, 81(4), 776-780.

See Also

sacf, msarima, stats::Box.test

Examples

```
fit <- msarima(MAD_m40, order = c(1, 0, 2),
               seasonal = list(c(0, 1, 1), c(0, 1, 1)))

# periods 24 and 168 taken from the msts object
msportmanteau(MAD_m40, nlag = 3)
# residual test, periods and npar from the model
msportmanteau(fit, type = "Monti", nlag = 3)
```

---

sacf *Autocorrelation Plots for Multiple Seasonal Cycles*

Description

Sample autocorrelation (sacf) and partial autocorrelation (spacf) functions that highlight the lags belonging to one or more seasonal cycles, so that the seasonal structure of a multi-seasonal series is visible at a glance. ggsacf() and ggspacf() are the **ggplot2** equivalents.

Usage

```
sacf(
  x,
  season = NULL,
  lag.max = NULL,
  ci = 0.95,
  col = NULL,
  main = NULL,
  ylab = "ACF",
  plot = TRUE,
  ...
)

spacf(x, ..., ylab = "PACF")

ggsacf(
  x,
  season = NULL,
  lag.max = NULL,
  ci = 0.95,
  main = NULL,
  ylab = "ACF",
  ...
)

ggspacf(x, ..., ylab = "PACF")
```

Arguments

x	A numeric vector, a ts/msts object, or a fitted msarima model (in which case the model residuals are used).
season	Numeric vector of seasonal periods to highlight, of any length. If NULL (default) the periods are taken from the msts/ts attributes of x, or from the fitted model.
lag.max	Maximum lag. Defaults to 3 * max(season), or to the stats::acf default when no seasonal period is available.
ci	Coverage of the confidence bands, as a probability. Default 0.95.
col	Colours used for the seasonal lags, one per element of season (recycled). Defaults to a colour-blind safe palette (Okabe and Ito, 2008), the same one used by ggsacf() and ggspacf().
main, ylab	Plot title and y-axis label.
plot	If TRUE (default) the correlogram is drawn.
...	Further arguments passed to stats::acf, and from spacf() and ggspacf() to sacf() and ggsacf().

Details

Lags that are multiples of more than one period (lag 168 with `season = c(24, 168)`, say) are attributed to the longest matching period.

The series is treated as a plain numeric vector, so lags are always reported in number of observations and not in units of time. Leading and trailing NAs, such as those produced by differencing in msarima, are dropped (stats::na.contiguous).

Value

`sacf()` and `spacf()` invisibly return the stats::acf object, with the seasonal periods in the `"season"` attribute and, for each lag, the index of the period it belongs to in `"season.group"`. `ggsacf()` and `ggspacf()` return a `ggplot` object.

References

Okabe, M. and Ito, K. (2008). Color Universal Design (CUD): How to make figures and presentations that are friendly to colorblind people. https://jfly.uni-koeln.de/color/

See Also

stats::acf, msarima

Examples

```
fit <- msarima(MAD_m40, order = c(1, 0, 2),
               seasonal = list(c(0, 1, 1), c(0, 1, 1)))

sacf(MAD_m40, lag.max = 200)   # periods 24 and 168 taken from the msts object
spacf(fit)                     # residual pacf, periods taken from the model
```

---

simulate.msarima *Simulate from a Multi-Seasonal ARIMA Model*

Description

Returns a simulated time series from a fitted ARIMA model with one or more seasonal components.

Usage

```
## S3 method for class 'msarima'
simulate(object = NULL, nsim = 100, seed = NULL, burn = NULL, ...)
```

Arguments

object	An object of class "msarima".
nsim	Length of the simulated series. Default is 100.
seed	Either NULL or an integer passed to set.seed before simulating, to make the result reproducible. Default is NULL.
burn	Number of burn-in observations to discard. If NULL, it is set by msarima.sim to max(100, 2 * kk), where kk is the largest lag in the model. Default is NULL.
...	Additional arguments (currently not used).

Details

The model specification and the estimated parameters, including the constant term when the model was fitted with include.constant = TRUE, are extracted from the fitted object and passed to msarima.sim.

Value

A time series object (ts or msts) of length nsim, with the seasonal periods of the fitted model.

See Also

msarima, msarima.sim

Examples

```
# Simulate data
set.seed(123)
y <- msarima.sim(nsim = 200,
                 order = c(1, 0, 0),
                 seasonal = list(c(1, 0, 0), c(0, 0, 1)),
                 seasonal.period = c(4, 11),
                 ar = c(0.5, 0.5), ma = 0.3, sigma2 = 1)

# Fit model
fit <- msarima(y, order = c(1, 0, 0),
               seasonal = list(c(1, 0, 0), c(0, 0, 1)),
               seasonal.period = c(4, 11))

# Simulate from fitted model
y_sim <- simulate(fit, nsim = 100)
```

Index

∗ **datasets**
MAD_m40, 4
∗ **package**
msarima-package, 2

AIC.msarima *(*msarima-methods*)*, 9

BIC.msarima *(*msarima-methods*)*, 9

coef.msarima *(*msarima-methods*)*, 9

fitted.msarima *(*msarima-methods*)*, 9
forecast.msarima, 3

ggsacf *(*sacf*)*, 14
ggspacf *(*sacf*)*, 14

logLik.msarima *(*msarima-methods*)*, 9

MAD_m40, 4
msarima, *4*, 6, *10*, *12–17*
msarima-methods, 9
msarima-package, 2
msarima.sim, 11, *17*
msportmanteau, 13

optim, *7*

print.msarima *(*msarima-methods*)*, 9
print.summary_msarima
*(*msarima-methods*)*, 9

residuals.msarima *(*msarima-methods*)*, 9

sacf, *13*, 14, *14*
set.seed, *17*
simulate.msarima, *4*, *12*, 16
spacf *(*sacf*)*, 14
stats::acf, *15, 16*
stats::Box.test, *14*
stats::na.contiguous, *13*, *16*
summary.msarima *(*msarima-methods*)*, 9