Package 'arima2'

May 7, 2026

Title Likelihood Based Inference for ARIMA Modeling

Version 3.4.3

Date 2025-12-10

Description Estimating and analyzing auto regressive integrated moving average (ARIMA) models. The primary function in this package is arima(), which fits an ARIMA model to univariate time series data using a random restart algorithm. This approach frequently leads to models that have model likelihood greater than or equal to that of the likelihood obtained by fitting the same model using the arima() function from the 'stats' package. This package enables proper optimization of model likelihoods, which is a necessary condition for performing likelihood ratio tests. This package relies heavily on the source code of the arima() function of the 'stats' package. For more information, please see Jesse Wheeler and Edward L. Ionides (2025) <doi:10.1371/journal.pone.0333993>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports ggplot2, methods

Depends R (>= 4.1.0)

LazyData true

BugReports https://github.com/jeswheel/arima2/issues/

NeedsCompilation yes

Author Jesse Wheeler [aut, cre, cph],
Noel McAllister [aut],
Dhajanae Sylvertooth [aut],
Edward Ionides [ctb],
Brian Ripley [ctb] (Author of arima source code in stats package.),
R Core Team [cph] (Author of arima source code in stats package.)

Maintainer Jesse Wheeler <jessewheeler@isu.edu>

Repository CRAN

Date/Publication 2025-12-10 19:00:02 UTC

Contents

`aicTable` *ARIMA AIC table*

Description

Construct table of AIC for all combinations 0<=p<=P and 0<=q<=Q

Usage

```
aicTable(data, P, Q, D = 0, ic = c("aic", "aicc"), ...)
```

Arguments

`data`	a time series object, or a dataset that can be used as input into the arima function.
`P`	a positive integer value representing the maximum number of AR coefficients that should be included in the table.
`Q`	a positive integer value representing the maximum number of MA coefficients that should be included in the table.
`D`	a positive integer value representing the degree of differencing
`ic`	Information criterion to be used in the table.
`...`	Additional arguments passed to `arima()`.

Details

This function creates an AIC table for ARMA models of varying sizes. Each row for the table corresponds to a different AR value, and each column of the table corresponds to a different MA value.

Value

A matrix containing the model AIC values.

Examples

```
set.seed(654321)
aicTable(presidents, 3, 2)
```

`arima` *ARIMA Modeling of Time Series*

Description

Fit an ARIMA model to a univariate time series. This function builds on the ARIMA model fitting approach used in `stats::arima()` by fitting model parameters via a random restart algorithm.

Usage

```
arima(
  x,
  order = c(0L, 0L, 0L),
  seasonal = list(order = c(0L, 0L, 0L), period = NA),
  xreg = NULL,
  include.mean = TRUE,
  transform.pars = TRUE,
  fixed = NULL,
  init = NULL,
  method = c("CSS-ML", "ML", "CSS"),
  init_method = c("DL", "UnifRoots"),
  n.cond,
  SSinit = c("Rossignol2011", "Gardner1980"),
  optim.method = "BFGS",
  optim.control = list(),
  kappa = 1e+06,
  diffuseControl = TRUE,
  max_iters = 100,
  max_repeats = 10,
  max_inv_root = 1,
  min_inv_root_dist = 0,
  eps_tol = 1e-04
)
```

Arguments

`x`	a univariate time series
`order`	a specification of the non-seasonal part of the ARIMA model: the three integer components (p, d, q) are the AR order, the degree of differencing, and the MA order.

`seasonal` a specification of the seasonal part of the ARIMA model, plus the period (which defaults to `frequency(x)`). This may be a `list` with components `order` and `period`, or just a numeric vector of length 3 which specifies the seasonal `order`. In the latter case the default period is used.

`xreg` Optionally, a vector or matrix of external regressors, which must have the same number of rows as `x`.

`include.mean` logical indicating if the ARMA model should include a mean/intercept term. The default is `TRUE` for undifferenced series, and it is ignored for ARIMA models with differencing.

`transform.pars` logical; if true, the AR parameters are transformed to ensure that they remain in the region of stationarity. Not used for `method = "CSS"`. For `method = "ML"`, it has been advantageous to set `transform.pars = FALSE` in some cases, see also `fixed`.

`fixed` optional numeric vector of the same length as the total number of coefficients to be estimated. It should be of the form

$$(\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q, \Phi_1, \ldots, \Phi_P, \Theta_1, \ldots, \Theta_Q, \mu),$$

where ϕ_i are the AR coefficients, θ_i are the MA coefficients, Φ_i are the seasonal AR coefficients, Θ_i are the seasonal MA coefficients and μ is the intercept term. Note that the μ entry is required if and only if `include.mean` is `TRUE`. In particular it should not be present if the model is an ARIMA model with differencing.

The entries of the `fixed` vector should consist of the values at which the user wishes to "fix" the corresponding coefficient, or `NA` if that coefficient should *not* be fixed, but estimated.

The argument `transform.pars` will be set to `FALSE` if any AR parameters are fixed. A warning will be given if `transform.pars` is set to (or left at its default) `TRUE`. It may be wise to set `transform.pars = FALSE` even when fixing MA parameters, especially at values that cause the model to be nearly non-invertible.

`init` optional numeric vector of initial parameter values. Missing values will be filled in, by zeroes except for regression coefficients. Values already specified in `fixed` will be ignored.

`method` fitting method: maximum likelihood or minimize conditional sum-of-squares. The default (unless there are missing values) is to use conditional-sum-of-squares to find starting values, then maximum likelihood. Can be abbreviated.

`init_method` Method used to randomly sample parameter initializations. `init_method = "DL"` will sample parameters using the Durbin-Levinson algorithm, described by Monahan (1984). If `init_method = "UnifRoots"`, then inverted roots of AR and MA polynomials will be sampled uniformly from the complex unit circle.

`n.cond` only used if fitting by conditional-sum-of-squares: the number of initial observations to ignore. It will be ignored if less than the maximum lag of an AR term.

`SSinit` a string specifying the algorithm to compute the state-space initialization of the likelihood; see `KalmanLike` for details. Can be abbreviated.

`optim.method` The value passed as the `method` argument to `optim`.

`optim.control` List of control parameters for `optim`.

`kappa` the prior variance (as a multiple of the innovations variance) for the past observations in a differenced model. Do not reduce this.

`diffuseControl` Boolean indicator of whether or initial observations will have likelihood values ignored if controlled by the diffuse prior, i.e., have a Kalman gain of at least 1e4.

`max_iters` Maximum number of random restarts for methods "CSS-ML" and "ML". If set to 1, the results of this algorithm is the same as `stats::arima()` if argument `diffuseControl` is also set as TRUE. `max_iters` is often not reached because the condition `max_repeats` is typically achieved first.

`max_repeats` Integer. If the last `max_repeats` random starts did not result in improved likelihoods, then stop the search. Each result of the optim function is only considered to improve the likelihood if it does so by more than `eps_tol`.

`max_inv_root` positive numeric value less than or equal to 1. This number represents the maximum size of the inverted MA or AR polynomial roots for a new parameter estimate to be considered an improvement to previous estimates. Concerns of numeric stability arise when the size of polynomial roots are near unity circle. The default value 1 means that the the parameter values corresponding with the best log-likelihood will be returned, even if they are near unity. Suitable values of this parameter are near the value 1.

`min_inv_root_dist` positive numeric value less than 1. This number represents the minimum distance between AR and MA polynomial roots for a new parameter estimate to be considered an improvement on previous estimates. This is intended to avoid the possibility of returning parameter estimates with nearly canceling roots. Appropriate choices are values near 0.

`eps_tol` Tolerance for accepting a new solution to be better than a previous solution in terms of log-likelihood. The default corresponds to a one ten-thousandth unit increase in log-likelihood.

Value

A list of class `c("Arima2", "Arima")`. This list contains all of the same elements as the output of stats::arima, along with some additional elements. All elements of the output list are:

`coef` A vector of AR, MA, and regression coefficients. These can be extracted by the stats::coef method.

`sigma2` The MLE of the variance of the innovations.

`var.coef` The estimated variance matrix of the coefficients `coef`, which can be extracted by the stats::vcov method.

`mask` A vector containing boolean values, indicating which parameters of the model were estimated.

`loglik` The maximized log-likelihood (of the differenced data).

`aic` The AIC value corresponding to the log-likelihood.

`arma` A compact form of the model specification, as a vector giving the number of AR, MA, seasonal AR and seasonal MA coefficients, plus the period and the number of non-seasonal and seasonal differences.

`residuals` The fitted innovations.

`call` The matched call.

`series` The name of the series x.

`code` The convergence value returned by stats::optim.

`n.cond` The number of initial observations not used in the fitting.

`nobs` The number of observations used for the fitting.

`model` A list representing the Kalman Filter used in the fitting.

`x` The input time series.

`num_starts` Number of restarts before convergence criteria was satisfied.

`all_values` Numeric vector of length `num_starts` containing the loglikelihood of every parameter initialization.

References

Monahan, John F. (1984) A note on enforcing stationarity in autoregressive-moving average models. *Biometrika*, **71**(2), 403–404.

Examples

```
# example code
set.seed(12345)
arima(miHuron_level$Average, order = c(2, 0, 1), max_iters = 100)
```

`ARMApolyroots` *ARMA polyroots*

Description

This function calculates the roots of the AR or MA polynomials that correspond to an ARMA model.

Usage

```
ARMApolyroots(model, type = c("AR", "MA"))
```

Arguments

`model`	Either of fitted object of class `Arima` (i.e., the output of either `stats::arima()` or arima), a list with named elements at least one of the named elements `ar` or `ma`, or a vector with named elements, such as `c("ar1" = 0.3, "ar2" = -0.2, "ma1" = 0.14)` Seasonal coefficients are ignored.
`type`	character of value "AR" or "MA", indicating whether or not the AR or MA polynomial roots are desired.

Value

A numeric vector containing the roots of the MA or AR polynomials

Examples

```
set.seed(123456)
ARMApolyroots(sample_ARMA_coef((order = c(2, 1))), type = "AR")

mod <- arima(lh, order = c(3,0,0))
ARMApolyroots(mod, type = "AR")
```

---

miHuron_level *Annual January levels of lake Michigan-Huron*

Description

The dataset is a subset of the monthly average depth (ft) of lake Michigan-Huron. The data were retrieved online from the Great Lakes Environmental Research Laboratory. Various measurement gauges exist; this data was taken from the master gauge.

Usage

```
miHuron_level
```

Format

`miHuron_level`**:**

A data frame with 155 observations and two columns:

Date `Date` column that records when the observation was made.

Average `numeric` column representing the average depth in feet.

Source

NOAA. Annual January levels of lake Michigan-Huron. Great Lakes Environmental Research Laboratory, https://www.glerl.noaa.gov/data/dashboard/data/levels/mGauge/miHuronMog.csv. Accessed 08/24/2023.

The original url used to obtain the data no longer exists, but a partial subset of the same data can be found at https://www.greatlakescc.org/en/coordinating-committee-products-and-datasets/ under the subsection: Monthly Mean Water Levels.

plot.Arima2 *Plot* Arima2 *object*

Description

This function plots time series data loaded from an Arima2 object or plots inverse roots of the AR or MA polynomials in a fitted ARIMA model on the complex unit circle.

Usage

```
## S3 method for class 'Arima2'
plot(x, roots = TRUE, title = NULL, tick.lab = NULL, ...)
```

Arguments

x	An Arima2 object. This parameter is an object created using the function arima2().
roots	Would you instead prefer to plot the roots on a unit circle? Insert logical type here.
title	Title of plot
tick.lab	Time vector of numeric or character/string type.
...	Other parameters

Details

The output of this function is a ggplot object, so layers may be added to the output of this function using ggplot2's plus operator.

Value

Arima 2 plot, which is a ggplot2 object. Type of plot is indicated through roots parameter.

Examples

```
plot(arima(lh, order = c(1,0,1)))
plot(x = arima(lh, order = c(3,0,1)), roots = FALSE)
```

`profile.Arima2` *Profile for* `Arima2` *object*

Description

This function performs profile log-likelihood of an `Arima2` function.

Usage

```
## S3 method for class 'Arima2'
profile(
  fitted,
  d = 0,
  npts = 100L,
  lower = -1,
  upper = 1,
  which = 1L,
  max_iters = 1,
  ...
)
```

Arguments

`fitted`	An `Arima2` object that has been fit to data.
`d`	Integer number of differences. Should match the differences used to obtain the `fitted` object.
`npts`	Integer number of points to evaluate the profile.
`lower`	Numeric lower bound for the profile search.
`upper`	Numeric upper bound for the profile search.
`which`	Integer indicating which parameter to perform the profile over. See Details section for more information.
`max_iters`	Maximum number of random restarts. See arima for more details.
`...`	additional arguments needed for the profile function

Details

The parameter `which` specifies parameter in the following vector will be profiled over:

$$\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q, \Phi_1, \ldots, \Phi_P, \Theta_1, \ldots, \Theta_Q, \mu$$

where p, q are non-negative integers representing the number of AR and MA coefficients of `fitted`, respectively, and ϕ_i are the AR coefficients, θ_i are the MA coefficients, Φ_i are the seasonal AR coefficients, Θ_i are the seasonal MA coefficients and μ is the model intercept.

Value

data.frame object containing the results of the profile likelihood.

Examples

```
# example code
set.seed(123)
mod <- arima(miHuron_level$Average, order = c(1, 0, 1), max_iters = 100)
prof <- profile(mod, which = 2L, lower = -0.5, upper = 0.5)
plot(prof$ma1, prof$loglik)
```

`sample_ARMA_coef` *Sample ARMA coef*

Description

This function randomly samples the ARMA coefficients of a specified ARMA model. The coefficients are sampled so that the resulting ARMA model is both causal and invertible.

Usage

```
sample_ARMA_coef(
  order = c(0L, 0L),
  seasonal = list(order = c(0L, 0L), period = NA),
  n = 1,
  Mod_bounds = c(0, 1),
  min_inv_root_dist = 0,
  method = c("UnifRoots", "DL")
)
```

Arguments

`order`	A specification of the non-seasonal part of the ARIMA model: this is different than the `order` input of `stats::arima()`, because the degree of differencing is not included. Thus order is a vector of length two of the form (p, q).
`seasonal`	A specification of the seasonal part of the ARIMA model. Can be either a vector of length 2, or a list with an `order` component; if a list, a period can be included, but it does not affect the function output.
`n`	An integer indicating how many sets of ARMA coefficients should be sampled.
`Mod_bounds`	Bounds on the magnitude of the roots.
`min_inv_root_dist`	This parameter is included so as to help avoid ARMA models that contain parameter redundancy, if desired. Specifically, this parameter ensures that the minimum distance between any of the inverted roots in the AR and MA polynomials is greater than `min_inv_root_dist`. Inverted roots that are near each other leads to canceling or nearly canceling roots, effectively reducing the size of the ARMA model.
`method`	Method used to randomly sample parameter initializations. `init_method = "DL"` will sample parameters using the Durbin-Levinson algorithm, described by Monahan (1984). If `init_method = "UnifRoots"`, then inverted roots of AR and MA polynomials will be sampled uniformly from the complex unit circle.

Details

For an ARMA model to be causal and invertible, the roots of the AR and MA polynomials must lie outside the the complex unit circle. The AR and MA polynomials are defined as:

$$\phi(z) = 1 - \phi_1 z - \phi_2 z^2 - \ldots - \phi_p z^p$$

$$\theta(z) = 1 + \theta_1 z + \theta_2 z^2 + \ldots + \theta_q z^q$$

where z is a complex number, $\phi_1, \ldots, \phi_p$ are the p AR coefficients of the ARMA model, and $\theta_1, \ldots, \theta_p$ are the q MA coefficients of the ARMA model.

This function implements two distinct sampling schemes. `init_method = "DL"` will sample parameters using the Durbin-Levinson algorithm, described by Monahan (1984). If `init_method = "UnifRoots"`, then inverted roots of AR and MA polynomials will be sampled uniformly from the complex unit circle. In the later method, to ensure that the resulting polynomial coefficients are real, we only sample half of the needed number of complex roots, and set the remaining half to be the complex conjugate of the sampled points. In the case where the number of coefficients is odd, the remaining root is sampled uniformly, satisfying the `Mod_bounds` parameter.

Value

a vector of randomly sampled ARMA coefficients.

References

Monahan, John F. (1984) A note on enforcing stationarity in autoregressive-moving average models. *Biometrika*, **71**(2), 403–404.

Examples

```
{
sample_ARMA_coef(
   order = c(2, 1),
   seasonal = list(order = c(1, 0), period = 2),
   n = 100
)
}
```

Index