# Package 'partialAR'

April 14, 2020

**Type** Package

**Title** Partial Autoregression

**Version** 1.0.12

**Date** 2020-04-12

**Description** A time series is said to be partially autoregressive if it can be represented as a sum of a random walk and an autoregressive sequence without unit roots. This package fits partially autoregressive time series, where the autoregressive component is AR(1). This may be of use in modeling certain financial time series.

**License** GPL-2 | GPL-3

**Imports** Rcpp (>= 0.11.2), zoo, parallel, ggplot2, MASS, tseries, data.table, KFAS, urca, plot3D, methods

**Suggests** egcm, TTR

**LinkingTo** Rcpp

**NeedsCompilation** yes

**Author** Matthew Clegg [aut, cre, cph]

**Maintainer** Matthew Clegg <matthewcleggphd@gmail.com>

**Repository** CRAN

**Date/Publication** 2020-04-14 09:10:02 UTC

## R topics documented:

---

`partialAR-package` *Partial autoregression*

---

## Description

Fits time series models which consist of a sum of a permanent and a transient component. The permanent component is modeled as a random walk, while the transient component is modeled as an autoregressive series of order one.

## Details

| | |
|---|---|
| Package: | partialAR |
| Type: | Package |
| Version: | 1.0 |
| Date: | 2015-01-12 |
| License: | GPL-2 \| GPL-3 |

This package fits time series models which consist of a sum of a permanent and a transient component. In other words, the model fitted is:

$$X_t = M_t + R_t$$

$$M_t = \rho M_{t-1} + \epsilon_{M,t}$$

$$R_t = R_{t-1} + \epsilon_{R,t}$$

$$-1 < \rho < 1$$

$$\epsilon_{M,t} \sim N(0, \sigma_M^2)$$

$$\epsilon_{R,t} \sim N(0, \sigma_R^2)$$

This model may be useful when modeling a time series that is thought to be primarily mean-reverting but which may also contain some random drift.

## Disclaimer

DISCLAIMER: The software in this package is for general information purposes only. It is hoped that it will be useful, but it is provided WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. It is not intended to form the basis of any investment decision. USE AT YOUR OWN RISK!

### Author(s)

Matthew Clegg

Maintainer: Matthew Clegg <matthewcleggphd@gmail.com>

### References

Summers, Lawrence H. Does the stock market rationally reflect fundamental values? *Journal of Finance*, 41(3), 591-601.

Poterba, James M. and Lawrence H. Summers. Mean reversion in stock market prices: Evidence and implications. *Journal of Financial Economics*, 22(1), 27-59.

Clegg, Matthew. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

### See Also

`arima` ARIMA modeling of time series

`egcm` Engle-Granger cointegration model

### Examples

```
set.seed(1)
x <- rpar(1000, 0.8, 1, 0.5)  # Generate a random PAR sequence
fit.par(x)                    # Estimate its parameters
plot(fit.par(x))              # Plot the estimate
test.par(x)                   # Test the goodness of fit


# An example involving European stock market data
data(EuStockMarkets)   # European Stock Markets 1991-1998

# Check for cointegration between German DAX and Swiss SMI
library(egcm)
egcm(log(EuStockMarkets[,c("DAX", "SMI")]))

# The series are not found to be cointegrated.
# Perhaps they are partially cointegrated?  Check the residuals
# of the cointegration fit for partial autoregression:
fit.par(egcm(EuStockMarkets[,c("DAX", "SMI")])$residuals)

# A plot of the model looks promising:
## Not run: plot(fit.par(egcm(EuStockMarkets[,c("DAX", "SMI")])$residuals))

# 74% of the variance is attributed to a mean-reverting
# AR(1) process.  However, it is important to check whether this is
# a better explanation than a simple random walk:
test.par(egcm(EuStockMarkets[,c("DAX", "SMI")])$residuals)

# The p-value is found to be 0.36, so the random walk hypothesis
# cannot be rejected.
```

```
# Another example involving a potential pairs trade between
# Coca-Cola and Pepsi.

# Fetch the price series for Coca-Cola (KO) and Pepsi (PEP) in 2014
library(TTR)
KO <- getYahooData("KO", 20140101, 20141231)$Close
PEP <- getYahooData("PEP", 20140101, 20141231)$Close

# Check whether they were cointegrated
egcm(KO,PEP)

# It turns out that they are not cointegrated.  Perhaps a better
# fit can be obtained with the partially autoregressive model:
fit.par(egcm(KO,PEP)$residuals)

# The mean-reverting component of the above fit explains 90% of
# the variance of the daily returns.  Thus, it appears that the
# two series are close to being cointegrated.  A plot further
# confirms this:
plot(fit.par(egcm(KO,PEP)$residuals))

# Still, it is important to check whether or not the residual
# series is simply a random walk:
test.par(egcm(KO,PEP)$residuals)

# In this case, the p-value associated with the hypothesis that
# the series is partially autoregressive is 0.12.  Thus, the
# evidence of partial autoregression is marginal. The random walk
# may be a better explanation.
```

## `as.data.frame.par.fit` *Convert a fit of the PAR model to a single row data.frame*

### Description

Convert a fit of the PAR model to a single row data.frame

### Usage

```
## S3 method for class 'par.fit'
 as.data.frame(x, row.names, optional, ...)
```

### Arguments

| | |
|---|---|
| `x` | An object of class `par.fit`. See `fit.par` |
| `row.names` | Not used |
| `optional` | Not used |
| `...` | Not used |

### Value

Returns a single row data.frame, with the following columns:

| | |
|---|---|
| robust | TRUE if robust estimation was used. |
| nu | If robust is TRUE, then this is the degrees-of-freedom parameter used in the t-distribution for the robust estimation. |
| opt_method | The optimization method that was used for finding these parameters. |
| n | Length of the vector that was fit to the PAR model |
| rho | Estimate of the coefficient of mean reversion |
| sigma_M | Estimate of the standard deviation of the innovations of the transient (mean-reverting) component. |
| sigma_R | Estimate of the standard deviation of the innovations of the permanent (random walk) component. |
| M0 | Estimate of the initial value of the transient component. |
| R0 | Estimate of the initial value of the permanent component. |
| rho.se | Standard error of the estimate of rho. |
| sigma_M.se | Standard error of the estimate of sigma_M. |
| sigma_R.se | Standard error of the estimate of sigma_R. |
| M0.se | Standard error of the estimate of M0. |
| R0.se | Standard error of the estimate of R0. |
| lambda | Value of the penalty factor lambda that was used in computing the estimates. |
| pvmr | Proportion of variance attributable to mean reversion. |
| negloglik | Negative log-likelihood of the model given these parameters. |

### Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

### See Also

fit.par

### Examples

```
require(TTR)
L <- getYahooData("L", 20120101, 20131231)$Close
fit.par(L)
as.data.frame(fit.par(L))
```

`estimate.par` *Estimates the parameters of a partially autoregressive fit using lagged variances*

## Description

Estimates the parameters of a partially autoregressive fit using lagged variances

## Usage

```
estimate.par(X, useR = FALSE, rho.max = 1)
```

## Arguments

| | |
|---|---|
| `X` | A numeric vector or `zoo` vector representing the time series whose parameters are to be estimated |
| `useR` | If `TRUE`, the estimation is performed using `R` code. If `FALSE`, the estimation is performed using a faster `C++` implementation. Default: `FALSE`. |
| `rho.max` | An artificial upper bound to be imposed on the value of `rho`. |

## Details

The method of lagged variances provides an analytical formula for the parameter estimates in terms of the variances of the lags $X[t+1] - X[t]$, $X[t+2] - X[t]$ and $X[t+3] - X[t]$. Let

$$V[k] = var(X[t+k] - X[t]).$$

Then, the estimated parameter values are given by the following formulas:

$$rho = -(V[1] - 2V[2] + V[3])/(2V[1] - V[2])$$

$$sigma_M^2 = (1/2)((rho + 1)/(rho - 1))(V[2] - 2V[1])$$

$$sigma_R^2 = (1/2)(V[2] - 2sigma_M^2)$$

## Value

Returns a numeric vector containing three named components

| | |
|---|---|
| `rho` | The estimated value of `rho` |
| `sigma_M` | The estimated value of `sigma_M` |
| `sigma_R` | The estimated value of `sigma_R` |

## Author(s)

Matthew Clegg <`matthewcleggphd@gmail.com`>

## References

Clegg, Matthew. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

## See Also

fit.par

## Examples

```
set.seed(1)
x <- rpar(1000, 0.5, 1, 2)  # Generate a random PAR sequence
estimate.par(x)
fit.par(x)  # For comparison
```

---

`fit.par` *Fit a partially autoregressive model*

---

## Description

Fit a partially autoregressive model

## Usage

```
fit.par(Y,
  robust = FALSE,
  model = c("par", "ar1", "rw"),
  lambda = 0,
  opt_method = c("css", "kfas", "ss"),
  rho.max = 1,
  nu = par.nu.default())
```

## Arguments

`Y` A numeric vector or `zoo` vector representing the time series whose parameters are to be estimated

`robust` If `TRUE`, then the error terms in the fit are assumed to follow a Student's t-distribution with degrees of freedom parameter given by `nu`. Otherwise, the error terms are assumed to be normally distributed. Default: `FALSE`.

`model` Specifies the model that is to be fit. Possible values are

- "par" The partially autoregressive model is fit.
- "ar1" An autoregressive model of order one is fit.
- "rw" A random walk is fit.

Default: `par`

`lambda` A penalty term $lambdasigma_R^2$ is added to the likelihood function. Default: $lambda = 0$.

`opt_method` Specifies the Kalman filter that will be used for optimization:

- "ss" Steady-state Kalman filter
- "css" Steady-state Kalman filter coded in C++
- "kfas" Kalman filter implementation of the KFAS package

Default: `css`

`rho.max` Specifies an upper limit on the value of `rho` that will be returned.

`nu` If `robust` is `TRUE`, this specifies the value of the degrees-of-freedom parameter used by the t-distribution. Default: 5

### Details

This routine determines the maximum likelihood fit of a time series to the partially autoregressive model, which is given by the specification:

$$X_t = M_t + R_t$$

$$M_t = \rho M_{t-1} + \epsilon_{M,t}$$

$$R_t = R_{t-1} + \epsilon_{R,t}$$

$$-1 < \rho < 1$$

$$\epsilon_{M,t} \sim N(0, \sigma_M^2)$$

$$\epsilon_{R,t} \sim N(0, \sigma_R^2)$$

The partially autoregressive model is a candidate for working with time series having both permanent and transient components.

If `robust` is `TRUE`, then a form of robust estimation is used. The error term is assumed to follow a Student's t-distribution with `nu` degrees of freedom.

The `model` parameter is used to alter the model that is fit. If `model` is `"par"`, then the partially autoregressive model is fit. If `model` is `"ar1"`, then an AR(1) model is fit. This is performed by fitting the partially autoregressive model with the restriction that $sigma_R = 0$. If `model` is `"rw"`, then a random walk model is fit. This is performed by fitting the partially autoregressive model with the restriction that $sigma_M = 0$.

The parameter `lambda` specifies the weighting of a penalty term that is added to the likelihood function. When $lambda > 0$, this drives the optimizer towards a solution that places a greater weight on the transient (mean-reverting) component, and when $lambda < 0$, this drives the optimizer towards a solution that places a greater weight on the permanent (random walk) component.

The fit is performed using maximum likelihood estimation for a Kalman filter representation of the model. When `opt_method` is `"ss"` or `"css"`, a steady-state Kalman filter is used. These two methods should give the same result, although `"css"` is to be preferred because the implementation is much faster. When `opt_method` is `"kfas"`, the KFAS Kalman Filter package `KFAS` is used. Because the Kalman gain matrix takes some time to converge to its steady state value, the `"kfas"` implementation will yield values that are close to but not the same as those of `"ss"` and `"css"`.

This routine prints the model that is found. The following is an example of the output obtained in one particular run:

```
Fitted model:
  X[t] = M[t] + R[t]
  M[t] = 0.9427 M[t-1] + eps_M,t,  eps_M,t ~ N(0, 0.8843^2)
        (0.0302)                                (0.0685)
  R[t] = R[t-1] + eps_R,t,         eps_R,t ~ N(0, 0.2907^2)
                                                (0.1710)
  M_0 = 0.0000, R_0 = -5.2574
           (NA)       (0.9625)
Proportion of variance attributable to mean reversion (pvmr) = 0.9050
Negative log likelihood = 339.51
```

In this ouptut, the coefficient of mean reversion rho is found to be 0.9427 with a standard error of 0.0302. This corresponds to a half-life of mean reversion of log(0.5)/log(0.9427) = 11.7 days. The parameter sigma_M is found to be 0.8843 with a standard error 0.0685. The parameter sigma_R is found to be 0.2907 with a standard error of 0.1710. The parameters M[0] and R[0] are 0.0 and -5.2574, respectively.

An important measure of the quality of fit of the partially autoregressive model is the proportion of variance attributable to mean reversion. This is a number between zero and one. When it is zero, the best fit is a pure random walk, and when it is one, the best fit is a pure mean-reverting series. In this case, it is found to be 0.9050, indicating that the mean-reverting component dominates.

The negative log likelihood of this particular fit is 339.51.

A plot method is available for plotting the fit, and the test.par method is available for testing the null hypotheses that an adequate fit can be obtained with a pure random walk or pure autoregressive series.

## Value

An S3 object of class fit.par is returned. The object contains the following values:

| | |
|---|---|
| data | The input vector Y |
| robust | The input parameter robust |
| nu | The input parameter nu |
| model | The input parameter model |
| lambda | The input parameter lambda |
| opt_method | The input parameter opt_method |
| rho.max | The input parameter rho.max |
| rho | The estimate of the parameter rho |
| sigma_M | The estimate of the parameter sigma_M |
| sigma_R | The estimate of the parameter sigma_R |
| M0 | The estimate of the parameter M[0] |
| R0 | The estimate of the parameter R[0] |
| par | The vector (rho,sigma_M,sigma_R,M0,R0) |
| stderr | The vector of standard errors |
| negloglik | The negative of the log likelihood score for these parameters |
| pvmr | The proportion of variance attributable to mean reversion (see pvmr.par) |

## Disclaimer

DISCLAIMER: The software in this package is for general information purposes only. It is hoped that it will be useful, but it is provided WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. It is not intended to form the basis of any investment decision. USE AT YOUR OWN RISK!

## Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

## References

Summers, Lawrence H. Does the stock market rationally reflect fundamental values? *Journal of Finance*, 41(3), 591-601.

Poterba, James M. and Lawrence H. Summers. Mean reversion in stock market prices: Evidence and implications. *Journal of Financial Economics*, 22(1), 27-59.

Clegg, Matthew. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

## See Also

arima ARIMA modeling of time series

egcm Engle-Granger cointegration model

## Examples

```
set.seed(1)
x <- rpar(1000, 0.8, 1, 0.5)  # Generate a random PAR sequence
fit.par(x)                  # Estimate its parameters
## Not run: plot(fit.par(x)   # Plot the estimate
test.par(x)                 # Test the goodness of fit


# An example involving European stock market data
data(EuStockMarkets)   # European Stock Markets 1991-1998

# Check for cointegration between German DAX and Swiss SMI
library(egcm)
egcm(log(EuStockMarkets[,c("DAX", "SMI")]))

# The series are not found to be cointegrated.
# Perhaps they are partially cointegrated?  Check the residuals
# of the cointegration fit for partial autoregression:
fit.par(egcm(EuStockMarkets[,c("DAX", "SMI")])$residuals)

# A plot of the model looks promising:
## Not run: plot(fit.par(egcm(EuStockMarkets[,c("DAX", "SMI")])$residuals))

# 74% of the variance is attributed to a mean-reverting
```

```
# AR(1) process.  However, it is important to check whether this is
# a better explanation than a simple random walk:
test.par(egcm(EuStockMarkets[,c("DAX", "SMI")])$residuals)

# The p-value is found to be 0.36, so the random walk hypothesis
# cannot be rejected.



# Another example involving a potential pairs trade between
# Coca-Cola and Pepsi.

# Fetch the price series for Coca-Cola (KO) and Pepsi (PEP) in 2014
library(TTR)
KO <- getYahooData("KO", 20140101, 20141231)$Close
PEP <- getYahooData("PEP", 20140101, 20141231)$Close

# Check whether they were cointegrated
library(egcm)
egcm(KO,PEP)

# It turns out that they are not cointegrated.  Perhaps a better
# fit can be obtained with the partially autoregressive model:
fit.par(egcm(KO,PEP)$residuals)

# The mean-reverting component of the above fit explains 90% of
# the variance of the daily returns.  Thus, it appears that the
# two series are close to being cointegrated.  A plot further
# confirms this:
plot(fit.par(egcm(KO,PEP)$residuals))

# Still, it is important to check whether or not the residual
# series is simply a random walk:
test.par(egcm(KO,PEP)$residuals)

# In this case, the p-value associated with the hypothesis that
# the series is partially autoregressive is 0.12.  Thus, the
# evidence of partial autoregression is marginal. The random walk
# may be a better explanation.
```

---

`kalman.gain.par` *Kalman gain matrix of the partially autoregressive model*

---

### Description

Kalman gain matrix of the partially autoregressive model

### Usage

```
kalman.gain.par(rho, sigma_M, sigma_R)
```

### Arguments

| | |
|---|---|
| rho | The coefficient of mean reversion |
| sigma_M | The standard deviation of the innovations of the mean-reverting component |
| sigma_R | The standard deviation of the innovations of the random walk component |

### Details

The state space representation of the partially autoregressive model is given as

```
[ M[t] ]     [ rho   0 ] [ M[t-1] ]     [ epsilon_M[t] ]
[      ]  =  [         ] [        ]  +  [              ]
[ R[t] ]     [ 0     1 ] [ R[t-1] ]     [ epsilon_R[t] ]
```

where the innovations epsilon_M[t] and epsilon_R[t] have the covariance matrix

```
[ epsilon_M[t] ]      [ sigma_M^2        0 ]
[              ]  ~   [                    ]
[ epsilon_R[t] ]      [ 0        sigma_R^2 ]
```

The steady state Kalman gain matrix is given by the matrix

```
[ K_M ]
[     ]
[ K_R ]
```

where

$K_M = 2sigma_M^2/(sigma_R*(sqrt((rho+1)^2 sigma_R^2+4sigma_M^2)+(rho+1)sigma_R)+2sigma_M^2)$

and $K_R = 1 - K_M$.

### Value

Returns a two-component vector (K_M,K_R) representing the Kalman gain matrix.

### Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

### References

Clegg, Matthew. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

### See Also

fit.par

### Examples

```
kalman.gain.par(0, 1, 0)  # -> c(1, 0)   (pure AR(1))
kalman.gain.par(0, 0, 1)  # -> c(0, 1)   (pure random walk)
kalman.gain.par(0.5, 1, 1)  # -> c(0.3333, 0.6667)
```

## likelihood_ratio.par *Computes log likelihood ratio for partial autoregressive model*

### Description

Computes the log likelihood ratio for the partially autoregressive model.

First, a fit is performed for the specified null model. Then, a fit is performed for the alternative model that the sequence is partially autoregressive. The likelihood scores are computed for both models, and the log likelihood ratio is returned.

### Usage

```
likelihood_ratio.par(X, robust = FALSE, null_model = c("rw", "ar1"),
  opt_method = c("css", "kfas", "ss"), nu = par.nu.default())
```

### Arguments

| | |
|---|---|
| `X` | The numeric vector or zoo vector to which the partially autoregressive model is being fit. |
| `robust` | If `TRUE`, then errors are assumed to follow a t-distribution with `nu` degrees of freedom. If `FALSE`, then errors are assumed to follow a normal distribution. Default: `FALSE` |
| `null_model` | Specifies the null hypothesis:<br>• "rw" Pure random walk (e.g., `sigma_M` = `0`)<br>• "ar1" Pure autoregressive (e.g., `sigma_R` = `0`)<br>Default: `"rw"` |
| `opt_method` | The method to be used for calculating the negative log likelihood.<br>• `"ss"` Steady-state Kalman filter with normally distributed errors<br>• `"css"` Steady-state Kalman filter with normally distributed errors, coded in C++<br>• `"kfas"` Traditional Kalman filter of the `KFAS` package<br>Default: `"css"` |
| `nu` | If `robust` is `TRUE`, this specifies the number of degrees of freedom of the t-distribution. Default: 5 |

### Value

A numeric value representing the log likelihood ratio

### Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

## References

Clegg, Matthew. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

## See Also

`fit.par`

---

`loglik.par` *Negative log likelihood of a partially autoregressive fit*

---

## Description

Negative log likelihood of a partially autoregressive fit

## Usage

```
loglik.par(Y, rho, sigma_M, sigma_R, M0 = 0, R0 = Y[1],
  calc_method = c("css", "kfas", "ss", "sst", "csst"),
  nu = par.nu.default())
```

## Arguments

| | |
|---|---|
| `Y` | A numeric vector representing the time series to which the partially autoregressive model is being fit. |
| `rho` | The coefficient of mean reversion |
| `sigma_M` | Standard deviation of the innovations of the mean-reverting process |
| `sigma_R` | Standard deviation of the innovations of the random walk process |
| `M0` | Initial value of the mean-reverting process |
| `R0` | Initial value of the random walk process |
| `calc_method` | The method to be used for calculating the negative log likelihood.<br>• `"ss"` Steady-state Kalman filter with normally distributed errors<br>• `"css"` Steady-state Kalman filter with normally distributed errors, coded in C++<br>• `"kfas"` Traditional Kalman filter of the `KFAS` package<br>• `"sst"` Steady-state Kalman filter with t-distributed errors<br>• `"csst"` Steady-state Kalman filter with t-distributed errors, coded in C++<br>Default: `"css"` |
| `nu` | If `calc_method` is `"sst"` or `"csst"`, this specifies the number of degrees of freedom of the t-distribution. |

## Value

Returns the negative log likelihood of fitting the partially autoregressive model with parameters `(rho,sigma_M,sigma_R,M0,R0)` to the data series `Y`.

### Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

### References

Clegg, Matthew. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

### See Also

fit.par

### Examples

```
loglik.par(0,0,0,1)  # -> same as -log(dnorm(0))
loglik.par(0,0,1,0)  # -> same as -log(dnorm(0))
loglik.par(0,0,1,1)  # -> same as -log(dnorm(0,0,sqrt(2)))
```

---

## pvmr.par — *Proportion of variance attributable to mean reversion*

---

### Description

Proportion of variance attributable to mean reversion of a partially autoregressive model

### Usage

```
pvmr.par(rho, sigma_M, sigma_R)
```

### Arguments

| | |
|---|---|
| rho | The coefficient of mean reversion |
| sigma_M | The standard deviation of the innovations of the mean-reverting component |
| sigma_R | The standard deviation of the innovations of the random walk component |

### Details

This routine determines the proportion of variance attributable to mean reversion for a partially autoregressive model. The partially autoregressive model is given by the specification:

$$X_t = M_t + R_t$$

$$M_t = \rho M_{t-1} + \epsilon_{M,t}$$

$$R_t = R_{t-1} + \epsilon_{R,t}$$

$$-1 < \rho < 1$$

The proportion of variance attributable to mean reversion is defined as

$$R^2[MR] = Var((1-B)M[t])/Var((1-B)X[t])$$

where M[t] is the mean-reverting component of the system at time t, X[t] is the state of the entire system at time t, and B is the backshift operator.

It will be a value between zero and one, with zero indicating that none of the variance is attributable to the mean reverting component, and one indicating that all of the variance is attributable to the mean-reverting component.

In the case of the partially autoregressive model, the proportion of variance attributable to mean reversion is given by the following formula:

$$R^2[MR] = 2sigma_M^2/(2sigma_M^2 + (1+rho)sigma_R^2)$$

### Value

Returns the proportion of variance attributable to mean reversion for the parameter values (rho,sigma_M,sigma_R).

### Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

### References

Clegg, Matthew. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

### See Also

fit.par

### Examples

```
pvmr.par(0,0,1)  # -> 0
pvmr.par(0,1,0)  # -> 1
pvmr.par(0,1,1)  # -> 0.6667
pvmr.par(0.5,1,1) # -> 0.5714
pvmr.par(0.5,1,2) # -> 0.25
```

---

rpar *Random partially autoregressive sequence*

---

### Description

Random partially autoregressive sequence

## Usage

```
rpar(n, rho, sigma_M, sigma_R, M0 = 0, R0 = 0,
  include.state = FALSE, robust = FALSE, nu = par.nu.default())
```

## Arguments

| | |
|---|---|
| n | Length of sequence to generate |
| rho | The coefficient of mean reversion |
| sigma_M | The standard deviation of the innovations of the mean-reverting component |
| sigma_R | The standard deviation of the innovations of the random walk component |
| M0 | Initial state of mean-reverting component |
| R0 | Initial state of random walk component |
| include.state | If `TRUE`, a `data.frame` is returned containing the states of the mean-reverting and random walk components. Otherwise, a numeric vector is returned containing the state of the system. Default: `FALSE`. |
| robust | If TRUE, innovations are t-distributed. Otherwise, they are normally distributed. Default: `FALSE`. |
| nu | If `robust` is `TRUE`, then this is the degrees of freedom parameter to be used in the t-distributed innovations. |

## Details

Generates a random sequence according to the specification of the partially autoregressive model. The partially autoregressive model is given as

$$X_t = M_t + R_t$$

$$M_t = \rho M_{t-1} + \epsilon_{M,t}$$

$$R_t = R_{t-1} + \epsilon_{R,t}$$

$$-1 < \rho < 1$$

To generate the random sequence, the sequences `epsilon_M[t]` and `epsilon_R[t]` are first generated. These are then used to build up the sequences `M[t]`, `R[t]` and `X[t]`.

## Value

If `include.state` is `FALSE`, then returns the sequence `X[t]`. Otherwise, returns a `data.frame` with the following columns:

| | |
|---|---|
| X | State of the system |
| M | State of the mean-reverting component |
| R | State of the random walk component |
| eps_M | Innovations in the mean-reverting component |
| eps_R | Innovations in the random walk component |

### Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

### References

Clegg, Matthew. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

### See Also

fit.par

### Examples

```
set.seed(1)
x <- rpar(10000, 0.5, 2, 1)
library(tseries)
adf.test(x)      # Seems to contain a unit root, as expected
estimate.par(x)  # Estimate parameters using lagged variances
fit.par(x)       # Maximum likelihood estimate
```

---

## sample.likelihood_ratio.par

*Generates random samples of the likelihood ratio for the partially autoregressive model*

---

### Description

Generates random samples of the likelihood ratio for the partially autoregressive model

### Usage

```
sample.likelihood_ratio.par(n = 500, rho = 0.8, sigma_M = 1, sigma_R = 1,
  nrep = 1000, use.multicore = TRUE, robust = FALSE,
  nu = par.nu.default(), seed.start = 0)
```

### Arguments

| | |
|---|---|
| n | Length of the randomly generated sequence. Possibly a vector. |
| rho | The coefficient of mean reversion. Possibly a vector. |
| sigma_M | Standard deviation of the innovations of the mean-reverting process. Possibly a vector. |
| sigma_R | Standard deviation of the innovations of the random walk process. Possibly a vector. |
| nrep | Number of repetitions to perform |
| use.multicore | If TRUE, then the parallel package is used to speed up processing. |

| | |
|---|---|
| robust | If TRUE, then sequences containing t-distributed errors are generated, and robust fits are performed. Possibly a vector. |
| nu | If robust is TRUE, then this is the degrees-of-freedom parameter to be used. Possibly a vector. |
| seed.start | Starting seed to use for the random number generator. |

## Details

The purpose of this function is to facilitate studying the behavior of the `fit.par` function by generating random partially autoregressive sequences and determining the maximum likelihood fits to them. For each combination of parameter values given by `n`, `rho`, `sigma_M`, `sigma_R`, `robust` and `nu`, generates `nrep` random partially autoregressive sequences with these parameters. Then, uses `fit.par` to fit the sequence using the partially autoregressive model, the pure random walk model and the pure mean reversion model. Returns a `data.frame` containing the results of the fits.

## Value

A `data.frame` with the following columns

| | |
|---|---|
| n | The length of the sequence |
| rho | The value of rho that was used for generating the sequence |
| sigma_M | The value of sigma_M that was used for generating the sequence |
| sigma_R | The value of sigma_R that was used for generating the sequence |
| robust | 0 if normally distributed innovations, 1 if t-distributed innovations |
| nu | If t-distributed innovations, the value of the degrees of freedom parameter |
| seed | The value used for seeding the random number generator |
| rw_rho | The value of rho estimated using the pure random walk model (always 0) |
| rw_sigma_M | The value of sigma_M estimated using the pure random walk model (always 0) |
| rw_sigma_R | The value of sigma_R estimated using the pure random walk model |
| rw_negloglik | The negative log likelihood of the fit obtained with the pure random walk model |
| mr_rho | The value of rho estimated using the pure mean-reversion model |
| mr_sigma_M | The value of sigma_M estimated using the pure mean-reversion model |
| mr_sigma_R | The value of sigma_R estimated using the pure mean-reversion model (always 0) |
| mr_negloglik | The negative log likelihood of the fit obtained with the pure mean-reversion model |
| par_rho | The value of rho estimated using the PAR model |
| par_sigma_M | The value of sigma_M estimated using the PAR model |
| par_sigma_R | The value of sigma_R estimated using the PAR model |
| par_negloglik | The negative log likelihood of the fit obtained with the PAR model |
| rw_lrt | The log likelihood ratio of the random walk model vs. the PAR model |
| mr_lrt | The log likelihood ratio of the mean-reversion model vs. the PAR model |
| kpss_stat | Statistic computed by the KPSS test (see ur.kpss) |
| kpss_p | p-value associated with kpss_stat |
| pvmr | Proportion of variance attributable to mean reversion found for PAR fit |

## Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

## References

Clegg, Matthew. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

## See Also

`fit.par`

## Examples

```
sample.likelihood_ratio.par(500, c(0.5,0.75), 1, c(1,2),nrep=3)
```

---

`statehistory.par` *Estimates hidden states of a partially autoregressive model*

---

## Description

Estimates hidden states of a partially autoregressive model

## Usage

```
statehistory.par(A, data = A$data)
```

## Arguments

| | |
|---|---|
| `A` | A `par.fit` object returned from a previous call to `fit.par` |
| `data` | A sequence of observed states |

## Details

Based on the parameters of the model fitted by the previous call to `fit.par`, produces a `data.frame` containing the inferred hidden states of the process.

## Value

A `data.frame` with one row for each observation in `data`. The columns in the `data.frame` are as follows:

| | |
|---|---|
| `X` | Value of the observed state (`data`) at this time |
| `M` | Estimated value of the mean-reverting component at this time |
| `R` | Estimated value of the random walk component at this time |
| `eps_M` | Estimated innovation to the mean-reverting component |
| `eps_R` | Estimated innovation to the random walk component |

### Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

### References

Clegg, Matthew. Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*

### See Also

`fit.par`

### Examples

```
# A simple example to compare the fitted values of the mean-reverting
# component with the actual data
set.seed(1)
xactual <- rpar(1000, 0.9, 2, 1, include.state=TRUE)
xfit <- fit.par(xactual$X)
xstates <- statehistory.par(xfit)
summary(lm(xstates$M ~ xactual$M))

## Not run:
    require(ggplot)
    xdf <- rbind(data.frame(data="actual", x=1:nrow(xactual), value=xactual$M),
      data.frame(data="fitted", x=1:nrow(xstates), value=xstates$M))
    ggplot(xdf, aes(x=x, y=value, colour=data)) + geom_line()

## End(Not run)
```

---

`test.par` *Likelihood ratio test for partially autoregressive model*

---

### Description

Likelihood ratio test for partially autoregressive model

### Usage

```
test.par(Y, alpha = 0.05, null_hyp = c("rw", "ar1"),
  ar1test = c("lr", "kpss"), robust = FALSE)
```

### Arguments

| | |
|---|---|
| `Y` | A numeric vector or a `par.fit` object produced by a previous call to `fit.par` |
| `alpha` | The critical value to be used in determining whether or not to reject the null hypothesis. See `which.hypothesis.partest`. Default: 0.05. |
| `null_hyp` | The null hypothesis. This can be one or both of the following: |

- "rw" Includes the pure random walk as a null hypothesis
- "ar1" Includes a purely mean-reverting AR(1) series as a null hypothesis

Default: Both "rw" and "ar1"

ar1test — Specifies the type of test to be performed to reject the AR(1) null hypothesis. This can be one of the following:

- "lr" Likelihood ratio rest
- "kpss" Unit root test of Kwiatkowski, Phillips, Schmidt and Shin, as implemented in the package urca.

Default: "lr"

robust — TRUE if robust estimation should be used when fitting the models

## Details

The partially autoregressive model is fit to Y (or a previously fitted model is re-used if Y is an object of class par.fit), representing the alternative hypothesis. The null models specified by null_hyp are also fit. The likelihood ratio test is then used to determine whether or not the null model(s) should be rejected. Statistics are output containing the test results.

If "ar1" is included in null_hyp and ar1test = "kpss", then the unit root test of Kwiatkowski, Phillips, Schmidt and Shin is used in place of the likelihood ratio test to reject the null hypothesis that Y is a pure AR(1) sequence.

An example invocation of this function is as follows:

```
> test.par(x)

  Test of [Random Walk or AR(1)] vs Almost AR(1) [LR test for AR1]

data:  x

Hypothesis              Statistic    p-value
Random Walk                 -0.62      0.476
AR(1)                       -0.11      0.062
Combined                               0.380
```

In this invocation, x is tested against the null hypothesis that it is either a pure random walk or a pure AR(1) series. The test of the random walk null hypothesis produces a likelihood ratio score of -0.62, which has a corresponding p-value of 0.476. The test of the AR(1) null nypothesis produces a likelihood ratio score of -0.11, which has a corresponding p-value of 0.062. The p-value for the combined test representing the union of these two conditions is 0.38. Thus, the null hypothesis cannot be rejected.

## Value

An object of class "partest"

## Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

### References

Matthew Clegg (2015): Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*.

Denis Kwiatkowski, Peter C.B. Phillips, Peter Schmidt, and Yongcheol Shin (1992): Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics* 54, 159-178.

### See Also

`fit.par` `which.hypothesis.partest`

### Examples

```
set.seed(1)
x <- rpar(1000, 0.8, 1, 1)
test.par(x)
```

---

## which.hypothesis.partest — *Returns the preferred hypothesis when testing for partial autoregression*

---

### Description

Returns the preferred hypothesis when testing for partial autoregression

### Usage

```
which.hypothesis.partest(AT)
```

### Arguments

| | |
|---|---|
| `AT` | An object of class `"partest"` returned from a previous call to `test.par`. |

### Details

Based upon the critical value `alpha` used in the call to `test.par`, and based upon the statistics computed by `test.par`, selects a preferred explanatory hypothesis for the data and returns a string representing the chosen hypothesis.

### Value

One of the following strings:

| | |
|---|---|
| `"RW"` | The preferred hypothesis is a pure random walk |
| `"AR1"` | The preferred hypothesis is a pure AR(1) series |
| `"PAR"` | The preferred hypothesis is a partially autoregressive series |

| | |
|---|---|
| "RRW" | The preferred hypothesis is a random walk with t-distributed innovations |
| "RAR1" | The preferred hypothesis is a pure AR(1) series with t-distributed innovations |
| "RPAR" | The preferred hypothesis is a partially autoregressive model with t-distributed innovations |

## Author(s)

Matthew Clegg <matthewcleggphd@gmail.com>

## References

Matthew Clegg (2015): Modeling Time Series with Both Permanent and Transient Components using the Partially Autoregressive Model. *Available at SSRN: http://ssrn.com/abstract=2556957*.

## See Also

fit.par test.par

## Examples

```
set.seed(1)
which.hypothesis.partest(test.par(rpar(1000, 0, 1, 0)))  # -> "AR1"
which.hypothesis.partest(test.par(rpar(1000, 0, 0, 1)))  # -> "RW"
which.hypothesis.partest(test.par(rpar(1000, 0, 1, 1)))  # -> "PAR"

which.hypothesis.partest(test.par(rpar(1000, 0, 1, 0), robust=TRUE))   # -> "RAR1"
which.hypothesis.partest(test.par(rpar(1000, 0, 0, 1), robust=TRUE))   # -> "RRW"
which.hypothesis.partest(test.par(rpar(1000, 0.5, 1, 1), robust=TRUE)) # -> "RPAR"
```

# Index