Package: MAR1 (via r-universe)

September 26, 2024

Type Package

Title Multivariate Autoregressive Modeling for Analysis of Community Time-Series Data

Version 2.3

Date 2023-06-02

Description The MAR1 package provides basic tools for preparing ecological community time-series data for MAR modeling, building MAR-1 models via model selection and bootstrapping, and visualizing and exporting model results. It is intended to make MAR analysis sensu Ives et al. (2003) Analysis of community stability and ecological interactions from time-series data) a more accessible tool for anyone studying community dynamics. The user need not necessarily be familiar with time-series modeling or command-based statistics programs such as R.

URL https://atsa-es.github.io/MAR1/

BugReports https://github.com/atsa-es/MAR1/issues

License GPL-2

Depends R (>= 4.0.0), leaps, bestglm, stats, tcltk

Suggests knitr, MARSS (>= 3.11.0), rmarkdown

NeedsCompilation no

VignetteBuilder knitr

RoxygenNote 7.2.3

Repository https://atsa-es.r-universe.dev

RemoteUrl https://github.com/atsa-es/MAR1

RemoteRef HEAD

RemoteSha 5e3a6530fc606d14679fd50beaa5d0b5526da945

Contents

MAR1-package *Multivariate Autoregressive Modeling for Analysis of Community Time-Series Data*

Description

Ives et al. (2003) describe the model framework implemented in this package:

$\mathbf{X}(t) = \mathbf{A} + \mathbf{BX}(t\text{-}1) + \mathbf{CU}(t\text{-}1) + \mathbf{E}(t)$

The first order multivariate autoregressive model may include both variates (factors expected to affect their own dynamics and the dynamics of other variates) and covariates (factors that may affect the dynamics of variates but are unlikely to be correspondingly influenced by them). The model estimates the value of each variate at time *t* as a linear function of the values of all variates and covariates at time *t*-1. The coefficients estimated in the MAR model represent the relative interaction strengths between each variate and covariate included in the analysis.

Details

The **MAR1** package includes the following basic functions:

`prepare.data` Formats a dataset into evenly-spaced time-steps and marks continuous blocks of time-steps in preparation for MAR analysis. The data can also be log-transformed or z-scored with this function.

`run.mar` Searches for and estimates a MAR model for a given dataset. The random best-fit model search and conditional least squares calculation components of this function are derived from Matlab scripts originally written by AR Ives and subsequently refined by SR Carpenter, KL Cottingham, and MD Scheuerell. Returns a list object of class MAR for which there are `print`, `summary`, and `plot` methods.

`export.MAR` Creates a new directory and exports all components of a MAR object into that directory as csv files.

Author(s)

Lindsay P. Scheef <<`scheef@utexas.edu`>>

Maintainer: Eli Holmes <<`eli.holmes@noaa.gov`>>

References

Ives AR, Dennis B, Cottingham KL, and Carpenter SR (2003) Estimating community stability and ecological interactions from time-series data. Ecological Monographs 73:301-330

`export.MAR` *Export all elements of a MAR object to a new directory*

Description

Creates a new directory and writes all elements of an object of class `MAR` to that directory as csv files.

Usage

```
export.MAR(model.out,export=TRUE)
```

Arguments

`model.out`	Object of class `MAR` to be saved as csv files
`export`	Optionally allows the name of the new directory to be set equal to a character string. Otherwise, the name of the new directory will be "MAR.results *Sys.time*"

Details

All elements of the given MAR class object are converted to csv format and written to a new directory created within the current working directory. The name of the new directory can be chosen by setting the `export` argument equal to a character string. If the name of the new directory matches one already present in the working directory, the name will be amended with `Sys.time()` to avoid overwriting the previous folder

Value

Writes to a directory.

Author(s)

LP Scheef

See Also

run.mar

hist.MARtop *Plot histogram of AIC values of top MAR models*

Description

Plots a histogram of the AIC values of the best-fit models contained in the `$top.bestfit` component of an object of class `MAR`

Usage

```
## S3 method for class 'MARtop'
hist(x, ...)
```

Arguments

`x`	The `$top.bestfit` component of an object of class `MAR`
`...`	Further arguments passed to or from other methods

Details

The AIC values of the top best-fit models are plotted as a frequency histogram. A blue asterisk indicates the AIC value of the selected best-fit model.

Value

A graphics device containing the histogram of top best-fit model AIC values

Author(s)

LP Scheef

See Also

plot.MARtop

Examples

```
# load a sample MAR model fit. See \code{\link{run1}}.
data(run1)
hist(run1$top.bestfit)
```

`L4.AllDates` *L4 plankton abundance time-series*

Description

This dataset is a plankton abundance time-series collected at the Western Channel Observatory L4 station. It contains zooplankton abundances in #/m^3, phytoplankton abundances in #/ml, and sea surface temperature in degrees C. Sampling dates for zooplankton, phytoplankton, and surface temperature values are not always aligned, resulting in NA values within the time-series.

Usage

```
data(L4.AllDates)
```

Format

A data frame with 1123 observations on the following 20 variables.

`date`	sampling dates; often read into R as a factor variable
`cnidarian`	e.g., hydromedusae and siphonophores
`amphipod`	e.g., gammarids
`chaetognath`	*Sagitta spp.*
`krill`	Euphausiids (all stages)
`pteropod`	*Clione* and *Limacina*
`tunicate`	larvaceans
`cladoceran`	*Evadne spp.* and *Podon spp.*
`calanoid.lg`	large calanoid copepods (>2 mm)
`calanoid.sm`	small calanoid copepods (<2 mm)
`cyclopoid`	cyclopoid copepods
`poecilostom`	poecilostomatoid copepods
`harpact`	harpacticoid copepods
`diatom`	phytoplankton: diatoms
`dino`	phytoplankton: dinoflagellates
`other.algae`	phytoplankton: e.g., coccolithophores, green
`cirripedia`	meroplankton: barnacle nauplii and cyprids
`mero.grazers`	meroplankton: other grazers, e.g., gastropod and bivalve larvae
`decapod`	meroplankton: e.g., crab and shrimp larvae
`surface.temp`	sea surface temperature (degrees C)

Source

http://www.westernchannelobservatory.org.uk/data.php

References

Eloire D, Somerfield PJ, Conway DVP, Halsband-Lenk C, Harris R, and Bonnet D (2010) Temporal variability and community composition of zooplankton at station L4 in the Western Channel: 20 years of sampling. Journal of Plankton Research 32:657-679 [doi:10.1093/plankt/fbq009]

Southward AJ, Langmead O, Hardman-Mountford NJ, Aiken J, Boalch GT, Dando PR, Genner MJ, Joint I, Kendall MA, Halliday NC, Harris RP, Leaper R, Mieszkowska N, Pingree RD, Richardson AJ, Sims DW, Smith T, Walne AW, and Hawkins SJ (2005) Long-term oceanographic and ecological research in the western English Channel. Advances in Marine Biology 47:1-105 [doi:10.1016/S0065-2881(04)47001-1]

Scheef, LP, DE Pendleton, SE Hampton, SL Katz, EE Holmes, ME Scheuerell, and DG Johns (2012) Assessing marine plankton community structure from long-term monitoring data with multivariate autoregressive (MAR) models: a comparison of fixed station vs. spatially distributed sampling data. Limnology & Oceanography: Methods 10:54-64.

See Also

`L4.mar`

Examples

```
data(L4.AllDates)
summary(L4.AllDates)
```

L4.mar *L4 plankton abundance time-series prepared for MAR modeling*

Description

Plankton abundance data collected at the Western Channel Observatory L4 station, formatted for use in `run.mar`. The dataset was prepared for MAR analysis with a call to
`prepare.data(data=L4.AllDates, increment="month", fill.gap=0, replace.0s="rand.half", log=T, z.method="deseason")`

Usage

```
data(L4.mar)
```

Format

A data frame with 179 observations on the following 21 variables.

`contin`	a numeric vector where matching values indicate blocks of continuous sampling dates
`date`	dates
`cnidarian`	e.g., hydromedusae and siphonophores
`amphipod`	e.g., gammarids
`chaetognath`	*Sagitta spp.*

`krill`	Euphausiids (all stages)
`pteropod`	*Clione* and *Limacina*
`tunicate`	larvaceans
`cladoceran`	*Evadne spp.* and *Podon spp.*
`calanoid.lg`	large calanoid copepods (>2 mm)
`calanoid.sm`	small calanoid copepods (<2 mm)
`cyclopoid`	cyclopoid copepods
`poecilostom`	poecilostomatoid copepods
`harpact`	harpacticoid copepods
`diatom`	phytoplankton: diatoms
`dino`	phytoplankton: dinoflagellates
`other.algae`	phytoplankton: e.g., coccolithophores, green
`cirripedia`	meroplankton: barnacle nauplii and cyprids
`mero.grazers`	meroplankton: other grazers, e.g., gastropod and bivalve larvae
`decapod`	meroplankton: e.g., crab and shrimp larvae
`surface.temp`	sea surface temperature

Source

http://www.westernchannelobservatory.org.uk/data.php

See Also

`L4.AllDates`, `prepare.data`

Examples

```
data(L4.mar)
summary(L4.mar)
```

`plot.MAR` *Plot coefficients of a MAR model*

Description

Plots the coefficients contained in an object of class `MAR` resulting from a call to `run.mar`

Usage

```
## S3 method for class 'MAR'
plot(x, y=NULL, ..., legend=FALSE)
```

Arguments

`x`	Object of class `MAR` to be plotted
`y`	Optional additional object of class `MAR` to be compared to `x`.
`...`	Not used.
`legend`	Should a legend be generated? If set to TRUE, an additional graphics device containing the legend will be generated

Details

The **B**- and **C**-matrix coefficients of the best-fit model are plotted as a grid of bar graphs. If bootstrapping was performed, discarded best-fit coefficients are faded. If any restrictions were set on interactions, red and green points at the base of bars are indicative of exclusion and inclusion, respectively.

Value

A graphics device containing the coefficient plot, and, if `legend=TRUE`, an additional device containing the corresponding legend

Author(s)

LP Scheef

Examples

```
# load a sample MAR model fit. See \code{\link{run1}}.
data(run1)
plot(run1)
```

plot.MARtop *Plot coefficients of top MAR models*

Description

Plots the coefficients of the best-fit models contained in the `$top.bestfit` component of an object of class `MAR`.

Usage

```
## S3 method for class 'MARtop'
plot(x, ...)
```

Arguments

`x`	The `$top.bestfit` component of an object of class `MAR`
`...`	Further arguments passed to or from other methods

Details

The **B**- and **C**-matrix coefficients of the top best-fit models are plotted as a grid of bar graphs, with the selected best-fit model represented by the top-most set of bars.

Value

A graphics device containing the coefficient plot

Author(s)

LP Scheef

See Also

`run.mar`

Examples

```
# load a sample MAR model fit. See \code{\link{run1}}.
data(run1)
plot(run1$top.bestfit)
```

`prepare.data` *Prepare data frame for MAR analysis*

Description

Formats a data frame of variable time-series for use in `run.mar`

Usage

```
prepare.data(data,
  increment = c("month", "year", "week", "day"), fill.gap = 0,
  replace.0s = c(FALSE, "rand.half", "add.ones"), log = FALSE,
  z.method = c(FALSE, "standard", "deseason"), order = NULL
)
```

Arguments

`data` Data frame to be transformed: first column dates, following columns variable time-series

`increment` Time-step increment data are to be averaged into: "month", "year", "week", or "day"

`fill.gap` Maximum length of gap between time-steps to be filled by linear interpolation

`replace.0s` How zeros in the time-series should be dealt with:

"rand.half" : replace zeros with random values less than 1/2 the minimum non-zero value for the variable
"add.ones" : add 1 to all values in time-series
FALSE : leave zeros in the data

`log` Should the data be log-transformed?

`z.method` Standardize the data so all variables have equal means and standard deviations:

"standard" : subtracts the overall variable mean and divides by the overall variable standard deviation
"deseason" : subtracts the variable mean for the increment (across years) and divides by the variable standard deviation for the increment
FALSE : do not standardize the data

`order` The order in which year, month, and day are recorded in the date format in the first column of `data`. Defaults to `NULL`, for which an attempt is made by the function to distinguish the date format, but the argument should be provided as "ymd", "mdy", etc., if the function cannot distinguish the order on its own

Details

The `run.mar` function requires a data frame consisting of a continuous time-block indicator variable in the first column, dates/time-steps in the second column, and variable time-series with evenly spaced time-step increments in the remaining columns. This function automatically creates the continuous time-block variable column and aggregates the time-series into evenly spaced increments as indicated by the `increment` argument. Although it automatically tries to assess the format of the dates in `data`, the `order` argument may have to be provided if this fails (for example, in the case where all dates in the dataset fall before the 12th of each month).

Other transformations that are commonly applied to ecological data prior to MAR modeling (e.g., Hampton et al. 2006, Ives et al. 2003) can optionally be performed, but are skipped by default with their respective arguments set to FALSE.

Value

A data frame with a continuous time-block indicator variable in the first column, dates/time-steps in the second column, and variable time-series with evenly spaced time-step increments in the remaining columns

Author(s)

LP Scheef

References

Hampton SE, Scheuerell MD, and Schindler DE (2006) Coalescence in the Lake Washington story: Interaction strengths in a planktonic food web. Limnology and Oceanography 51:2042-2051

Ives AR, Dennis B, Cottingham KL, and Carpenter SR (2003) Estimating community stability and ecological interactions from time-series data. Ecological Monographs 73:301-330

Examples

```
## Not run:
data(L4.AllDates)
L4.mar <- prepare.data(
  data = L4.AllDates, increment = "month", fill.gap = 0,
  replace.0s = "rand.half", log = TRUE, z.method = "deseason"
)

## End(Not run)
```

`print.MAR` *Print output of a MAR model*

Description

Method for printing components of an object of class `MAR` that provide a concise overview of the model

Usage

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

Arguments

`x`	Object of class `MAR` to be printed
`...`	Further arguments passed to or from other methods

Details

Formats and prints the following components of a `MAR` object when the object's name is called: the time-series variables that were selected to be included in the model; any restrictions that were set on variable interactions; the search type used to select the best-fit model; and the coefficients, AIC values, and R^2 values of the best-fit and bootstrapped models. Other components of a `MAR` object can be viewed by specifying their name according to `str(object)`.

Value

Prints a selection of components of a MAR object.

Author(s)

LP Scheef

Examples

```
# load a sample MAR model fit. See \code{\link{run1}}.
data(run1)
print(run1)
```

print.MARsummary *Print the summary of a MAR model*

Description

Formats and prints the result of applying `summary` to a `MAR` object.

Usage

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

Arguments

`x` Object of class `MARsummary`

`...` Further arguments passed to or from other methods

Value

Prints a summary of a MAR object.

Author(s)

LP Scheef

See Also

`summary.MAR`

Examples

```
# load a sample MAR model fit. See \code{\link{run1}}.
data(run1)
summary(run1)
```

run.mar *Initiate MAR analysis*

Description

Allows the user to select variables and restrict interactions, finds best-fit MAR model, and applies a bootstrap to the best-fit model

Usage

```
run.mar(data, variables=NULL, restrictions=NULL, search=c("random","exhaustive",
 "fwdstep","exhaustive.true"), boot=500, ntop=10, export=FALSE)
```

Arguments

`data` Data frame with continuous time-block variable in first column, ordered by dates in second column, followed by columns of taxa abundance time-series

`variables` A vector as long as the number of columns in `data` indicating how each taxon column should be assigned for the analysis (first two values for the time-block and date columns should be 0):

- 0 : not included
- 1 : included as a variate
- 2 : included as a covariate

Alternatively, an object of class `MAR` resulting from a previous call to `run.mar` from which to extract variable assignments

`restrictions` A matrix with *n* variate rows and *n* variate + *n* covariate columns of values indicating the potential of a direct effect of each column variable on each row variable:

- 0.5 : possible (may be included in model)
- 0 : unlikely/implausible (won't be included in model)
- 1 : probable (will be included in model)

Alternatively, on object of class `MAR` resulting from a previous call to `run.mar` from which to extract interaction restrictions

`search` A character string indicating the type of search that should be used to find the best-fit model; either "random" (default), "exhaustive", "fwdstep", or "exhaustive.true"; see "Details" section below for descriptions of search types

`boot` Either an integer indicating the number of iterations that should be performed in the model bootstrap or `FALSE` to skip bootstrapping

`ntop` If `search="random"`, `"exhaustive"`, or `"exhaustive.true"`, the number of top best-fit models from the random search to be returned for potential comparison to the selected best-fit model

`export` If set to TRUE, a call to `export.MAR` is executed at the end of the analysis. `export.MAR` creates a new directory and saves all components of the MAR model object in that directory as csv files

Details

Variables and Constraints:

If the `variables` or `restrictions` arguments are not provided, the function creates windows that allow the user to pick which column variables in `data` should be included in the MAR model as variates or covariates and to set restrictions on potential interactions between model variables. Unless the `variables` argument is provided in the function call, the `restrictions` argument must be `NULL`.

Search Types:

If `search="random"`, which is the default, a random search is performed to find the best-fit model (as determined by AIC) for the included variate time-series. For each variate, 100 random models are constructed according to the restrictions that were set, and the model with the lowest AIC of these models is retained. This process is repeated 100 times, resulting in 100 "best-of-100" models. If any variable occurs in less than 15 of the 100 "best-of-100" models, that variable is discarded (i.e., the probability of that variable occurring in the random search is set to 0) and the search is repeated until the number of variables in successive searches remains constant (resulting in at least 2 search iterations per variate unless all variables are retained in the first iteration). The model with the lowest AIC of the final 100 "best-of-100" models is retained.

If `search="exhaustive"`, a search through possible models for each variate with respect to `restrictions` is performed using a leap and bound algorithm (Furnival and Wilson,1974) to find the "best-fit" (lowest AIC) model of all potential variable combinations without explicitly examining all possible subsets.

If `search="fwdstep"`, the best-fit model for each variate is built up from the NULL intercept model by sequentially adding whichever variable most improves the model AIC from the pool of potential variables. The model from the series with the lowest AIC is retained.

If `search="exhaustive.true"`, a true exhaustive search through all potential variable combinations with respect to `restrictions` is performed.

Statistics:

The coefficients of the final "best-fit" MAR model for all variates are attained using least-squares estimation. The coefficients of the B- and C-matrices represent interaction strengths of the column variables on the row variables. If the data were z-scored prior to analysis (see `prepare.data`), the A intercept values will not be significantly different from 0. Estimates of the stationary distribution mean and covariance for each variate, of the process errors, and of community stability (resilience and reactivity) are calculated following Ives (2003). If `bootstrap` is not set to FALSE, these statistics are also calculated for the bootstrapped model.

Value

Returns a list of class `MAR` containing:

`variables.selected`	corresponds to `variables` argument
`restrictions.set`	corresponds to `restrictions` argument
`search.type`	corresponds to `search` argument
`search.time.s`	time (in seconds) the best-fit model search took

And for each of `$bestfit` and `$bootstrap`:

`A`	each row is the a-value for the variate
`B`	B-matrix interaction coefficients of columns on rows

`C` C-matrix interaction coefficients of columns on rows

`log.likelihood` log.likelihood value for model

`AIC` AIC value for model

`BIC` BIC value for model

`R2.values` R^2 and conditional R^2 values for each variate

`stationary.distribution`

`mean` means of variates' stationary distributions
`covariance` covariance matrix of stationary distribution

`process.errors`

`residuals` E
`covariance` sigma
`corrmatrix` correlation matrix

`stability`

`resilience` `eigB` eigenvalues of the B matrix
`detB` determinant of the B matrix
`maxeigB` max eigenvalue of B matrix
`maxeigkrB` max eigenvalue of B matrix kronecker products

`reactivity` `sigma.over.Vinf` -tr(sigma)/tr(Vinf)
`maxeigBxB` max eigenvalue of B'B matrix ("worst-case" reactivity)

If bootstrapping is not performed, `$bootstrap` will be NULL. Otherwise, in addition to the statistics above, `$bootstrap` will also contain a `$limits` list with the upper and lower 95% confidence limits of the best-fit model elements.

If `search="random"`, `"exhaustive"`, or `"exhaustive.true"`, the result will also contain `$top.bestfit`, an array of the top best-fit models tested during the model search (the first of which is the best-fit model that was selected). The number of top models returned may be less than the value set for the `ntop` argument if `ntop` exceeds the potential number of model configurations that can be tested for the selected variables and search method. The dimension of the array in which each top best-fit model is stored is named by its respective AIC value.

Warning

The "random" model search may select different best-fit models when run multiple times on the same data, particularly for searches including a large number of variables.

The "exhaustive.true" search can become very time-consuming for models with more than 12 variables.

Author(s)

LP Scheef

References

Furnival GM, Wilson Jr RW (1974) Regressions by leaps and bounds. Technometrics 16:499-511

Ives AR, Dennis B, Cottingham KL, and Carpenter SR (2003) Estimating community stability and ecological interactions from time-series data. Ecological Monographs 73:301-330

See Also

prepare.data, plot.MAR, export.MAR

Packages used for exhaustive search methods:
leaps, bestglm

Examples

```
## Not run:
## These examples take 1-2 minutes to run

## construct a MAR model using 'run.mar' arguments to set variables and restrictions
data(L4.mar)

myvar <- c(0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 2, 2, 2) # 8 variates, 3 covariates
myres <- matrix(0.5,
  nrow = length(which(myvar == 1)),
  ncol = length(which(myvar != 0))
) # no restrictions (all 0.5)

run1 <- run.mar(L4.mar, variables = myvar, restrictions = myres, search = "exhaustive")

run1 # only some elements of the object are printed
str(run1) # to see all elements
summary(run1) # some summary statistics for the model
plot(run1)

# set a few restrictions on taxa interactions
myres[1, c(1, 6, 9)] <- c(1, 0, 0) # included, not included, not included

# re-run the analysis with same variates as 'run1' and new restrictions
run1b <- run.mar(L4.mar, run1, myres, "exhaustive")
plot(run1, run1b)

# 'run1' variables and restrictions with a different search method
run1c <- run.mar(L4.mar, run1, run1, "fwdstep")
plot(run1, run1c, legend = TRUE) # plot with legend

## construct a MAR model using windows to select variables and restrictions ##
run2 <- run.mar(L4.mar, search = "exhaustive")
run2
summary(run2)
plot(run2)

## End(Not run)
```

run1 *A sample*

Description

An example MAR1 model prepared with the code in the example.

Usage

```
data(run1)
```

Format

A sample MAR1 model fit to the L4.mar data. Object of class MAR.

Examples

```
data(run1)

## Not run:
# run1 created with
data(L4.mar)

myvar <- c(0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 2, 2, 2) # 8 variates, 3 covariates
myres <- matrix(0.5,
  nrow = length(which(myvar == 1)),
  ncol = length(which(myvar != 0))
) # no restrictions (all 0.5)

run1 <- run.mar(L4.mar, variables = myvar, restrictions = myres, search = "exhaustive")

## End(Not run)
```

ss.mar1 *Fit a state-space MAR model using the MARSS package.*

Description

Uses a fitted MAR object from the function `run.mar` and an aggregated data.frame from the function `transform.data` to fit a state-space MAR model. The bestfit model in the MAR object is used to determine where the 0s are in the B and C matrices. Users can specify the form of the observation error variance-covariance matrix (R) and process error variance-covariance matrix (Q).

Usage

```
ss.mar1(aggregated.data, MAR.obj=NULL, model=list(), control=list(), silent=FALSE)
```

Arguments

`aggregated.data`
: Data frame with continuous time-block variable in first column, ordered by dates in second column, followed by columns of taxa abundance time-series. This type of data frame is output by the function `transform.data`.

`MAR.obj`
: A fitted MAR.obj as output by the function `run.mar`.

`model`
: An optional list with elements B, C, Q, or R that specify the form of those matrices. For Q and R, a numeric matrix can be used in which case Q or R will be fixed to those values. The text string “unconstrained” can be used for Q to specify that all elements are estimated (the default). The text string “diagonal and equal” can be used for Q or R to specify that the variance-covariance matrix is diagonal with one variance on the diagonal. The text string “diagonal and unequal” can be used for Q or R to specify that the variance-covariance matrix is diagonal but the variances on the diagonal are unconstrained. The text strings “zero” and “identity” can also be used for R or Q to specify those matrix forms. B and Q can be passed into the model list in order to use a B or C matrix other than the bestfit B and C in `MAR.obj`. In this case B and C must be a numeric matrix with 0s in the elements that will be fixed at 0. All non-zero values will be estimated. See ?MARSS (after installing the MARSS package) for a discussion of the model argument.

`control`
: A list of control elements for the MARSS package functions. The most useful may be `minit` to set a minimum number of iterations and `maxit` to set a maximum number of iterations.

`silent`
: If FALSE, the output from the MARSS fitting function is suppressed.

Details

The functions fits a simple observation model where each species in the B matrix is assumed to be observed with independent observation error. The covariates are assumed to be observed with no error. Missing variates and covariates are allowed.

The B and C matrices are constrained by default by the bestfit model in the `MLE.obj`. ss.mar1 will use the 0 locations in the bestfit model and constrain those B and C elements to be 0. Other B or C matrices can be passed in via the model argument and will override this behavior.

Value

A list with the elements

`ssfit`
: A marssMLE object output from `MARSS()`.

`A`
: The estimated A matrix

`B`
: The estimated B matrix

`C`
: The estimated C matrix

`process.errors`
: The process errors

`observation.errors`
: The observation errors

`AIC`	AIC
`AIC`	AICc
`log.likelihood`	log likelihood

Author(s)

Eli Holmes

References

The MARSS User Guide: Holmes, E. E., E. J. Ward, and M. D. Scheuerell (2012) Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112. Available at https://CRAN.R-project.org/package=MARSS.

Examples

```
## Not run:
## These examples take 1-2 minutes to run

## construct a MAR model using 'run.mar' arguments to set variables and restrictions ##

data(L4.mar)

myvar <- c(0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 2, 2, 2) # 8 variates, 3 covariates
myres <- matrix(0.5,
  nrow = length(which(myvar == 1)),
  ncol = length(which(myvar != 0))
) # no restrictions (all 0.5)

run1 <- run.mar(L4.mar, variables = myvar, restrictions = myres, search = "exhaustive")

# control can be passed in to limit the number of iterations run.
ss.fit <- ss.mar1(L4.mar, run1, control = list(maxit = 50))

# compare to best fit model
ss.fit$B
run1$bestfit$B

# Use a known observation error
R <- diag(0.2, 8)
ss.fit <- ss.mar1(L4.mar, run1, model = list(R = R), control = list(maxit = 50))

## End(Not run)
```

summary.MAR *Produce a summary of a MAR model*

Description

Method for producing a summary of an object of class MAR. Calculates the number of zero, non-zero, positive, and negative coefficients in the best-fit and bootstrapped models, gives the AIC, BIC, and summary of the R^2 values for the models, and lists the calculated stability statistics associated with the models.

Usage

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

Arguments

object	Object of class MAR
...	Further arguments passed to or from other methods

Value

Prints a summary of a MAR object.

Author(s)

LP Scheef

See Also

run.mar

Examples

```
# load a sample MAR model fit. See \code{\link{run1}}.
data(run1)
summary(run1)
```

Index