

Package ‘MARSS’

June 22, 2010

Type Package

Title Multivariate Autoregressive State-Space Modeling

Version 1.0

Date 2010-6-14

Depends MASS, mvtnorm, nlme, time

Author Eli Holmes, Eric Ward, and Kellie Wills, NOAA, Seattle, USA

Maintainer eli.holmes@noaa.gov <eli.holmes@noaa.gov>

Description The MARSS package fits constrained and unconstrained linear multivariate autoregressive state-space (MARSS) models to multivariate time series data.

License GPL-2

LazyLoad yes

LazyData yes

Repository CRAN

Date/Publication 2010-06-22 08:29:05

R topics documented:

MARSS-package	2
checkPopWrap	4
CSEGriskfigure	5
CSEGtmfigure	6
find.degenerate	8
graywhales	9
harborSeal	10
is.blockdiag	11
loggerhead	12
MARSS	13

MARSSaic	18
MARSSapplynames	20
MARSSboot	21
MARSScheckdims	23
MARSShessian	24
MARSSinits	25
MARSSinnovationsboot	27
MARSSkem	28
MARSSkemcheck	32
MARSSkf	32
marssm	35
marssm-class	37
MARSSmcinit	37
marssMLE	39
marssMLE-class	41
MARSSoptim	41
MARSSparamCIs	44
MARSSsimulate	45
MARSSvectorizeparam	46
plankton	47
popWrap	48
popWrap-class	51
show.doc	51
stdInnov	52
Index	54

Description

The MARSS package fits constrained and unconstrained multivariate autoregressive time-series models to multivariate time series data. The MARSS model is

$$\mathbf{x}(t+1) = \mathbf{B} \mathbf{x}(t) + \mathbf{U} + \mathbf{w}(t), \text{ where } \mathbf{w}(t) \sim \text{MVN}(\mathbf{0}, \mathbf{Q})$$

$$\mathbf{y}(t) = \mathbf{Z} \mathbf{x}(t) + \mathbf{A} + \mathbf{v}(t), \text{ where } \mathbf{v}(t) \sim \text{MVN}(\mathbf{0}, \mathbf{R})$$

$$\mathbf{x}(1) \sim \text{MVN}(\mathbf{x}_0, \mathbf{V}_0)$$

The parameters, hidden state processes (\mathbf{x}), and observations (\mathbf{y}) are matrices:

- $\mathbf{x}(t)$ is $m \times 1$
- $\mathbf{y}(t)$ is $n \times 1$ ($m \leq n$)
- \mathbf{Z} is $n \times m$
- \mathbf{B} is $m \times m$
- \mathbf{U} is $m \times 1$

- Q is $m \times m$
- A is $n \times 1$
- R is $n \times n$
- x_0 is $m \times 1$
- V_0 is $m \times m$

The package functions estimate the parameters U , Q , A , R , and x_0 using a Kalman-EM algorithm (primarily but see [MARSSoptim](#)). Parameters may be constrained to have shared elements (elements which are constrained to have the same value) or fixed elements (with the other elements estimated). The states and smoothed state estimates are provided via a Kalman filter and smoother. Bootstrapping, confidence interval estimation, bias estimation, model selection and simulation functions are provided. The main user interface to the package is the top-level function [MARSS](#).

Details

Package:	MARSS
Type:	Package
Version:	1.0
Date:	2010-07-01
License:	GPL 2.0
LazyLoad:	yes

Important MARSS functions:

[MARSS](#) Top-level function for specifying and fitting MARSS models.

[MARSSsimulate](#) Produces simulated data from a MARSS model.

[MARSSkem](#) Estimates MARSS parameters using an Kalman-EM algorithm.

[MARSSkf](#) Kalman filter and smoother.

[MARSSoptim](#) Estimates MARSS parameters using a quasi-Newton algorithm via [optim](#).

[MARSSaic](#) Calculates AICc, AICc, and various bootstrap AICs.

[MARSSboot](#) Creates bootstrap MARSS parameter estimates.

[MARSSparamCIs](#) Computes confidence intervals for maximum-likelihood estimates of MARSS parameters.

Author(s)

Eli Holmes, Eric Ward and Kellie Wills, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov,

kellie(dot)wills(at)noaa(dot)gov

References

The MARSS manual: Holmes, E. E. and E. J. Ward (2010) Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112.

Type `show.doc(MARSS, manual)` at the R command line to open the MARSS Manual.

Type `show.doc(MARSS, index)` to see all the package documentation, tutorials, and case study scripts.

checkPopWrap *Check Arguments to popWrap()*

Description

Checks inputs for wrapper object (`popWrap`) creation to ensure that the wrapper object can be handled by `as.marssm`. This is a utility function in the `MARSS-package`.

Usage

```
checkPopWrap(wrapperObj, wrapper.el, allowed, silent=FALSE)
```

Arguments

<code>wrapperObj</code>	An object of class <code>popWrap</code> .
<code>wrapper.el</code>	Wrapper elements; generally set by <code>MARSS</code>
<code>allowed</code>	Allowed constraints. This changes depending on the fitting method the user has specified (in <code>MARSS</code> or in the <code>marssMLE</code> object). Lists of for allowed for different fitting methods is set in <code>MARSSsettings.R</code> .
<code>silent</code>	Suppresses errors and warnings printing.

Details

Called by `popWrap` to ensure that user inputs are valid and can be handled by `as.marssm`.

Value

TRUE if object passes all checks.

Author(s)

Kellie Wills, NOAA, Seattle, USA.

kellie(dot)wills(at)noaa(dot)gov

See Also

`popWrap` `as.marssm`

Examples

```
## Not run:
## Error:
dat = t(harborSeal)
dat = dat[2:nrow(dat),]
wrapperObj = popWrap(dat, allowed=allowed$kem, constraint=list(Z="wrong"))

## End(Not run)
```

CSEGriskfigure

*Plot Extinction Risk Metrics***Description**

Generates a six-panel plot of extinction risk metrics used in Population Viability Analysis (PVA). This is a function used by one of the vignettes in the [MARSS-package](#).

Usage

```
CSEGriskfigure(data, te = 100, absolutethresh = FALSE, threshold = 0.1,
  datalogged = FALSE, silent = FALSE, return.model = FALSE,
  CI.method = "hessian", CI.sim = 1000)
```

Arguments

data	A data matrix with 2 columns; time in first column and counts in second column. Note time is down rows, which is different than the base MARSS-package functions.
te	Length of forecast period (positive integer)
absolutethresh	Is extinction threshold an absolute number? (T/F)
threshold	Extinction threshold either as an absolute number, if absolutethresh=TRUE, or as a fraction of current population count, if absolutethresh=FALSE.
datalogged	Are the data already logged? (T/F)
silent	Suppress printed output? (T/F)
return.model	Return state-space model as marssMLE object? (T/F)
CI.method	Confidence interval method: "hessian", "parametric", "innovations", or "none". See MARSSparamCIs .
CI.sim	Number of simulations for bootstrap confidence intervals (positive integer).

Details

Panel 1: Time-series plot of the data. Panel 2: CDF of extinction risk. Panel 3: PDF of time to reach threshold. Panel 4: Probability of reaching different thresholds during forecast period. Panel 5: Sample projections. Panel 6: TMU plot (uncertainty as a function of the forecast).

Value

If `return.model=TRUE`, an object of class `marssMLE`.

Author(s)

Eli Holmes, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov

References

Holmes, E. E. and E. J. Ward. 2010. Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112; this is the user manual accessed via `show.doc(MARSS, manual)`

(theory behind the figure) Holmes, E. E., J. L. Sabo, S. V. Viscido, and W. F. Fagan. (2007) A statistical approach to quasi-extinction forecasting. *Ecology Letters* 10:1182-1198.

(CDF and PDF calculations) Dennis, B., P. L. Munholland, and J. M. Scott. (1991) Estimation of growth and extinction parameters for endangered species. *Ecological Monographs* 61:115-143.

(TMU figure) Ellner, S. P. and E. E. Holmes. (2008) Resolving the debate on when extinction risk is predictable. *Ecology Letters* 11:E1-E5.

See Also

[MARSSboot](#) [marssMLE](#) [CSEGTmufigure](#)

Examples

```
d = harborSeal[,1:2]
kem = CSEGriskfigure(d)
```

CSEGTmufigure *Plot Forecast Uncertainty*

Description

Plot the uncertainty in the probability of hitting a percent threshold (quasi-extinction) for a single random walk trajectory. This is the quasi-extinction probability used in Population Viability Analysis. The uncertainty is shown as a function of the forecast, where the forecast is defined in terms of the forecast length (number of time steps) and forecasted decline (percentage). This is a function used by one of the vignettes in the [MARSS-package](#).

Usage

```
CSEGTmufigure(N = 20, u = -0.1, s2p = 0.01, make.legend = TRUE)
```

Arguments

<code>N</code>	Time steps between the first and last population data point (positive integer)
<code>u</code>	Per time-step decline (-0.1 means a 10% decline per time step; 1 means a doubling per time step.)
<code>s2p</code>	Process variance (Q). (a positive number)
<code>make.legend</code>	Add a legend to the plot? (T/F)

Details

This figure shows the region of high uncertainty in dark grey. In this region, the minimum 95 percent confidence intervals on the probability of quasi-extinction span 80 percent of the 0 to 1 probability. Green hashing indicates where the 95 percent upper bound does not exceed 5% probability of quasi-extinction. The red hashing indicates, where the 95 percent lower bound is above 95% probability of quasi-extinction. The light grey lies between these two certain/uncertain extremes. The extinction calculation is based on Dennis et al. (1991). The minimum theoretical confidence interval is based on Fieberg and Ellner (2000). This figure was developed in Ellner and Holmes (2008).

Examples using this figure are shown in the manual (`show.doc(MARSS, manual)`) in the PVA case study.

Author(s)

Eli Holmes, NOAA, Seattle, USA, and Steve Ellner, Cornell Univ.

`eli(dot)holmes(at)noaa(dot)gov`

References

Dennis, B., P. L. Munholland, and J. M. Scott. (1991) Estimation of growth and extinction parameters for endangered species. *Ecological Monographs* 61:115-143.

Fieberg, J. and Ellner, S.P. (2000) When is it meaningful to estimate an extinction probability? *Ecology*, 81, 2040-2047.

Ellner, S. P. and E. E. Holmes. (2008) Resolving the debate on when extinction risk is predictable. *Ecology Letters* 11:E1-E5.

See Also

[CSEGriskfigure](#)

Examples

```
CSEGtmufigure(N = 20, u = -0.1, s2p = 0.01)
```

find.degenerate *Find degenerate variance parameters*

Description

A helper function to find degenerate variance parameters in Kalman-EM estimates in the package [MARSS-package](#).

Usage

```
find.degenerate(MLEobj)
```

Arguments

MLEobj An object of class `marssMLE` as output by [MARSSkem](#). Typically after a call to [MARSS](#): `MARSS(data, method="kem")`.

Details

This function plots the log of the absolute value of the variance element (on the diagonal) against the log iteration number. Such log-log plots are commonly used to assess convergence in iterative routines. In state-space models with both process and non-process (observation) variance, it is entirely possible that the highest likelihood occurs when one of the variance element is zero. In this case, that element is degenerate (=0). Since the likelihood computations will generate an error, this means you will not be able to compute the true likelihood. Technically, it is certainly possible to compute the likelihood, but the code in [MARSSkf](#) will throw an error if you try to set one of the variances to zero.

Value

A plot where each diagonal variance element appears in a panel. Converged elements will have a flat log-log plot. Degenerate variance elements will have a declining log-log plot. Typically, degenerate variance parameters will show a slope of ca -1.

Author(s)

Eli Holmes, NOAA, Seattle, USA.
eli(dot)holmes(at)noaa(dot)gov

See Also

[MARSSkem](#)

Examples

```
dat = t(harborSealnomiss)
#shorten the data set to 10 yrs in order to produce a degenerate solution
#in this case there is not enough data to estimate all the Q variances
dat = dat[2:nrow(dat),1:10]
#force 200 iterations to get a nice log(param):log(iteration) plot
MLEobj = MARSS(dat, control=list(minit=200))
find.degenerate(MLEobj)
```

graywhales

Population Data Sets

Description

Example data sets for use in MARSS PVA vignettes in the [MARSS-package](#) manual. All are UNLOGGED population counts. The data sets are matrices with year in the first column and counts in other columns. Since MARSS functions require time to be across columns, these data matrices must be transposed before passing into MARSS functions.

Usage

```
data(graywhales)
data(grouse)
data(isleRoyal)
data(prairiechicken)
data(wilddogs)
```

Format

The data are supplied as a matrix with years in the first column and counts in the second (and third for isleRoyal) columns.

Source

- graywhales Gerber L. R., Master D. P. D. and Kareiva P. M. (1999) Gray whales and the value of monitoring data in implementing the U.S. Endangered Species Act. *Conservation Biology*, 13, 1215-1219.
- grouse Hays D. W., Tirhi M. J. and Stinson D. W. (1998) Washington state status report for the sharptailed grouse. Washington Department Fish and Wildlife, Olympia, WA. 57 pp.
- isleRoyal Peterson R. O., Thomas N. J., Thurber J. M., Vucetich J. A. and Waite T. A. (1998) Population limitation and the wolves of Isle Royale. In: *Biology and Conservation of Wild Canids* (eds. D. Macdonald and C. Sillero-Zubiri). Oxford University Press, Oxford, pp. 281-292.
- prairiechicken Peterson M. J. and Silvy N. J. (1996) Reproductive stages limiting productivity of the endangered Attwater's prairie chicken. *Conservation Biology*, 10, 1264-1276.
- wilddogs Ginsberg, J. R., Mace, G. M. and Albon, S. (1995). Local extinction in a small and declining population: Wild Dogs in the Serengeti. *Proc. R. Soc. Lond. B*, 262, 221-228.

Examples

```
str(graywhales)
str(grouse)
str(isleRoyal)
str(prairiechicken)
str(wilddogs)
```

harborSeal

Harbor Seal Population Count Data (Log counts)

Description

Data sets used in MARSS vignettes in the [MARSS-package](#). These are data sets based on LOGGED count data from Oregon, Washington and California sites where harbor seals were censused while hauled out on land. "harborSealnomiss" is an extrapolated data set where missing values in the original dataset have been extrapolated so that the data set can be used to demonstrate fitting population models with different underlying structures.

Usage

```
data(harborSeal)
data(harborSealnomiss)
data(harborSealWA)
```

Format

Matrix "harborSeal" contains columns "Years", "StraitJuanDeFuca", "SanJuanIslands", "EasternBays", "PugetSound", "HoodCanal", "CoastalEstuaries", "OlympicPeninsula", "CA.Mainland", "OR.NorthCoast", "CA.ChannelIslands", and "OR.SouthCoast". Matrix "harborSealnomiss" contains columns "Years", "StraitJuanDeFuca", "SanJuanIslands", "EasternBays", "PugetSound", "HoodCanal", "CoastalEstuaries", "OlympicPeninsula", "OR.NorthCoast", and "OR.SouthCoast". Matrix "harborSealWA" contains columns "Years", "SJF", "SJI", "EBays", "PSnd", and "HC", representing the same five sites as the first five columns of "harborSeal".

Details

Matrix "harborSealWA" contains the original 1978-1999 LOGGED count data for five inland WA sites. Matrix "harborSealnomiss" contains 1975-2003 data for the same sites as well as four coastal sites, where missing values have been replaced with extrapolated values. Matrix "harborSeal" contains the original 1975-2003 LOGGED data (with missing values) for the WA and OR sites as well as a CA Mainland and CA ChannelIslands time series.

Source

Jeffries et al. 2003. Trends and status of harbor seals in Washington State: 1978-1999. *Journal of Wildlife Management* 67(1):208-219.

Examples

```
str(harborSealWA)
str(harborSealnomiss)
str(harborSeal)
```

is.blockdiag *Matrix Utilities*

Description

Matrix utilities for MARSS functions in the [MARSS-package](#).

Usage

```
is.blockdiag(x)
is.blocequaltri(x, uniqueblocks=FALSE)
is.blockunconst(x, uniqueblocks=FALSE)
is.diagonal(x)
is.equaltri(x)
makediag(x, nrow=NA)
takediag(x)
is.design(x)
is.fixed(x)
is.identity(x)
vec(x)
unvec(x, dim=NULL)
is.wholenumber(x, tol = .Machine$double.eps^0.5)
as.design(fixed, free)
Imat(x)
```

Arguments

x	A matrix (or vector for 'makediag').
dim	Matrix dimensions.
fixed	A fixed matrix per the MARSS specification for fixed matrix syntax.
free	A free matrix per the MARSS specification for free matrix syntax.
nrow	Number of rows.
tol	Tolerance.
uniqueblocks	Must blocks be unique?

Details

'is...' tests for various matrix properties. `vec(x)` creates a column vector from a matrix per the standard `vec` math function. `unvec(c, dim)` takes the vector `c` and creates a matrix with the specified dimensions. `as.design(fixed, free)` returns the fixed vector and design matrix for a fixed/free pair. `Imat(nrow)` returns the identity matrix of dimension `nrow`.

Value

'`makediag(x)`': a matrix with diagonal x. '`takediag(x)`': the diagonal from matrix x.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

loggerhead

Loggerhead Turtle Tracking Data

Description

Data used in MARSS vignettes in the [MARSS-package](#). Tracking data from ARGOS tags on eight individual loggerhead turtles, 1997-2006.

Usage

```
data(loggerhead)
data(loggerheadNoisy)
```

Format

Data frames "loggerhead" and "loggerheadNoisy" contain the following columns:

turtle Turtle name.
day Day of the month (character).
month Month number (character).
year Year (character).
lon Longitude of observation.
lat Latitude of observation.

Details

Data frame "loggerhead" contains the original latitude and longitude data. Data frame "loggerhead-Noisy" has noise added to the lat and lon data to represent data corrupted by errors.

Source

Gray's Reef National Marine Sanctuary (Georgia) and WhaleNet: http://whale.wheelock.edu/whalenet-stuff/stop_cover_archive.html

Examples

```
str(loggerhead)
str(loggerheadNoisy)
```

Description

A top-level `MARSS-package` function to perform model specification and estimation for multivariate autoregressive state-space (MARSS) models:

$$\mathbf{x}(t+1) = \mathbf{B} \mathbf{x}(t) + \mathbf{U} + \mathbf{w}(t), \text{ where } \mathbf{w}(t) \sim \text{MVN}(\mathbf{0}, \mathbf{Q})$$

$$\mathbf{y}(t) = \mathbf{Z} \mathbf{x}(t) + \mathbf{A} + \mathbf{v}(t), \text{ where } \mathbf{v}(t) \sim \text{MVN}(\mathbf{0}, \mathbf{R})$$

$$\mathbf{x}(1) \sim \text{MVN}(\mathbf{x0}, \mathbf{V0})$$

MARSS provides an interface to the base `MARSS-package` functions so that users do not need to directly construct `marssm` and `marssMLE` objects.

Usage

```
MARSS(y,
      inits=NULL,
      constraint=NULL,
      fixed=NULL, free=NULL,
      miss.value=-99,
      method = "kem",
      fit=TRUE,
      silent = FALSE,
      control = NULL)
```

Arguments

The default settings for the optional arguments are set in `MARSSsettings.R` and are given below in the details section.

A $n \times T$ matrix of n time series over T time steps.

<code>y</code>	List with up to 7 matrices A, R, B, U, Q, x0, V0, specifying initial values for parameters.
<code>inits</code>	<ul style="list-style-type: none"> • B Initial value(s) for B matrix (length 1 or $m \times m$) • U Initial value(s) for U matrix (length 1 or $m \times 1$) • Q Initial value(s) for process error variance(s) (length 1 or $m \times m$) • A Initial value(s) for observation bias (length 1 or $n \times 1$) • R Initial value(s) for non-process (observation) error variance(s) (length 1 or $n \times n$) • x0 Initial value(s) for hidden state(s) at time=1 (length 1 or $m \times 1$) • V0 Initial variance(s) for hidden state(s) at time=1 (length 1 or $m \times m$)
<code>constraint</code>	Model specification using parameter constraint descriptions. See Details.
<code>fixed</code>	Optional model specification using matrices of fixed and free parameters. See Details.

<code>free</code>	Optional model specification using matrices of fixed and free parameters. See Details.
<code>miss.value</code>	How are missing values represented in the data?
<code>method</code>	Estimation method. MARSS 1.0 allows <code>method="kem"</code> and <code>"BFGS"</code> .
<code>fit</code>	TRUE/FALSE Whether to fit the model to the data. If FALSE, a <code>marssMLE</code> object with only the model is returned.
<code>silent</code>	TRUE/FALSE Suppresses printing of full error messages, warnings, progress bars and convergence information.
<code>control</code>	Estimation options for the maximization algorithm. The control options for <code>method="kem"</code> are <ul style="list-style-type: none"> • <code>minit</code> The minimum number of iterations to do in the maximization routine (if needed by method). If <code>method="kem"</code>, this is an easy way to up the iterations and see how your estimates are converging. (positive integer) • <code>maxit</code> Maximum number of iterations to be used in the maximization routine (if needed by method) (positive integer). • <code>abstol</code> Convergence tolerance for the maximization routine. (default is 0.01 which is a bit high.) • <code>iter.V0</code> The value of V0 to be used in place of 0 when <code>x0</code> is treated as fixed and <code>V0=0</code>. See manual for discussion of initial state variance. (default is 10 which works well) • <code>trace</code> A positive integer. If not 0, a record will be created during maximization iterations (what's recorded depends on the method of maximization). • <code>MCInit</code> If TRUE, do a Monte Carlo search of the initial condition space. (T/F) • <code>numInits</code> Number of random initial value draws if <code>MCInit=TRUE</code> (ignored otherwise). (positive integer) • <code>numInitSteps</code> Number of EM iterations for each random initial value draw if <code>MCInit=TRUE</code> (ignored otherwise). (positive integer) • <code>boundsInits</code> Bounds on the uniform distributions from which initial values will be drawn if <code>MCInit=TRUE</code> (ignored otherwise). • <code>silent</code> 1 or TRUE, Suppresses all printing including progress bars, error messages and convergence information. 0, Turns on all printing of progress bars, fitting information and error messages. 2, Prints a brief success/failure message.

Details

MARSS provides an interface to the base [MARSS-package](#) functions and allows specification and fitting of MARSS models. In [MARSS-package](#) 1.0, the available estimation methods are maximum-likelihood via a Kalman-EM algorithm (`method="kem"`) or via a quasi-Newton algorithm provided by function `optim` (`method="BFGS"`). The function `MARSS()` allows the user to specify models using text strings for common classes of parameter matrices via the argument `constraint`. It allows the user to specify fixed values for matrices by passing in numeric matrices in the `constraint` list. If the model classes available via the `constraint` strings are not sufficient, MARSS also allows specification using matrix pairs specified with argument `fixed`

and `free`. If `fixed/free` matrices are specified for some parameters, these will override any constraints for those parameters. See `marssm` or the manual (`show.doc(MARSS, manual)`) for documentation and instructions on specifying fixed and free matrices.

Valid constraints for `method="kem"` are below. See the manual (`show.doc(MARSS, manual)`) for details and type `allowed$kem` to see the allowed list specified in `MARSSsettings.R`.

- `Z` "identity" or a vector of factors specifying which of the `m` hidden state time series correspond to which of the `n` observation time series. May also be specified as a numeric `n x m` matrix to use a custom fixed `Z`.
- `B` "identity" or a vector of factors specifying shared diagonal elements. May also be specified as a numeric `m x m` matrix to use custom fixed `B`, but in this case all the eigenvalues of `B` must fall in the unit circle.
- `U` "unconstrained", "equal", "unequal" or "zero". May also be a vector of factors specifying shared `u` terms. May also be specified as a numeric `m x 1` matrix to use a custom fixed `U`. NAs can be put in this matrix to allow some elements to be fixed and others (the NAs) to be estimated.
- `Q` "unconstrained", "diagonal and unequal", "diagonal and equal", or "equalvarcov". May also be a vector of factors specifying shared diagonal elements. May also be specified as a numeric `m x m` matrix to use a custom fixed `Q`. If the matrix is diagonal (off-diagonals all zeros), then NAs may appear on the diagonal to allow some diagonal elements to be fixed while other elements (the NAs) are estimated.
- `A` "scaling" This treats `A` as an intercept or "zero" which sets `A` to a fixed value of all zeros. May also be specified as a numeric `n x 1` matrix to use a custom fixed `A`. NAs can be put in this matrix to allow some elements to be fixed and others (the NAs) to be estimated. Note all NAs in `A` would mean all `A` elements are estimated and would typically result in an under-constrained and unsolvable model; at least one `A` element per `X` state needs to be fixed.
- `R` "unconstrained", "diagonal and unequal", "diagonal and equal", or "equalvarcov". May also be a vector of factors specifying shared diagonal elements. May also be specified as a numeric `n x n` matrix to use a custom fixed `R`. If the matrix is diagonal (off-diagonals all zeros), then NAs may appear on the diagonal to allow some diagonal elements to be fixed while other elements (the NAs) are estimated.
- `x0` "unconstrained", "equal", "unequal" or "zero". May also be a vector of factors specifying shared initial state values. May also be specified as a numeric `m x 1` matrix to use a custom fixed `x0`. NAs can be put in this matrix to allow some elements to be fixed and others (the NAs) to be estimated.

Valid constraints for `method="BFGS"` are the same as for `method="kem"` except that the `Q` and `R` matrices must be diagonal if estimated. Thus "unconstrained" and "equalvarcov" are not allowed for `Q` or `R`. Type `allowed$BFGS` to see the allowed list specified in `MARSSsettings`.

The default estimation method, `method="kem"`, is the "Kalman-EM" algorithm described in the manual. The default settings for the optional arguments are set via `alldefaults$kem` in `MARSSsettings`. For this method, they are:

- `inits = list(B=1, U=0, Q=0.05, A=0, R=0.05, x0=-99, V0=10)`
- `constraint = list(Z="identity", A="scaling", R="diagonal and equal", B="identity", U="unconstrained", Q="diagonal and unequal", x0="unconstrained")`
- `miss.value = -99`

- `control=list(minit=1, maxit=5000, abstol=0.01, iter.V0=10, trace=0, safe=FALSE, MCInit=FALSE, numInits = 500, numInitSteps = 10, boundsInits=list(B=c(0,1), U=c(-1,1), logQ = c(log(1.0e-05),log(1.0)), Z=c(0,1), A=c(-1,1), logR = c(log(1.0e-05),log(1.0))))`

For `method="BFGS"`, type `alldefaults$BFGS` to see the defaults.

The likelihood surface for MARSS models can be highly multimodal. It is recommended that for final analyses the ML estimates are checked by using the Monte Carlo initial conditions search using `MCInit=TRUE` in the `control` list. This requires more computation time, but reduces the chance of the algorithm terminating at a local maximum and not reaching the true MLEs.

Value

An object of class `marssMLE` with the following components:

<code>model</code>	MARSS model specification (an object of class <code>marssm</code>).
<code>start</code>	List with 8 matrices A, R, B, U, Q, Z, x0, V0, specifying initial values for parameters.
<code>control</code>	A list of estimation options, as specified by arguments <code>control</code> .
<code>method</code>	Estimation method.

If `fit=TRUE`, the following are also added to the `marssMLE` object. If `fit=FALSE`, an `marssMLE` object ready for fitting via the specified `method` is returned.

<code>par</code>	A list with 8 matrices of estimated (+ fixed) parameter values Z, A, R, B, U, Q, x0, V0.
<code>kf</code>	A list containing Kalman filter/smoothing output from <code>MARSSkf</code> .
<code>numIter</code>	Number of iterations required for convergence.
<code>convergence</code>	Convergence status. 0 means converged successfully. Anything else is a warning or error. 2 means the MLEobj has an error; the MLEobj is returned so you can debug it. The other numbers are errors during fitting. The error code depends on the fitting method. See <code>MARSSkem</code> and <code>MARSSoptim</code> .
<code>logLik</code>	Log-likelihood.
<code>AIC</code>	Akaike's Information Criterion.
<code>AICc</code>	Sample size corrected AIC.

Author(s)

Eli Holmes and Kellie Wills, NOAA, Seattle, USA.

`eli(dot)holmes(at)noaa(dot)gov, kellie(dot)wills(at)noaa(dot)gov`

References

The user manual: Holmes, E. E. and E. J. Ward (2010) Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112 type `show.doc(MARSS, manual)` to see.

See Also

[marssm](#) [marssMLE](#) [MARSSkem](#) [MARSSopt](#) [im](#) [MARSS-package](#)

Examples

```
#harborSealWA is a n=5 matrix of logged population counts
dat = t(harborSealWA)
dat = dat[2:nrow(dat),] #remove the year row
#fit a model with 1 hidden state and 5 observation time series
kemfit = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1)),
  R="diagonal and unequal"))
kemfit$model #This gives a description of the model
print(kemfit$model) # same as kemfit$model
summary(kemfit$model) #This shows the model structure
show(kemfit$model) #shows the raw object

#The fit above throws a warning about convergence
#Determine which variance has not yet converged
find.degenerate(kemfit)

## Not run:
#Increase the number of iterations to ensure convergence
kemfit100 = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1)),
  R="diagonal and unequal"), control=list(minit=100))
find.degenerate(kemfit100) #now check convergence of variances

#fit the model using quasi-Newton algorithm
bfgsfit = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1)),
  R="diagonal and unequal"), method="BFGS")

#add CIs to a marssMLE object
#default uses an estimated Hessian matrix
kem.with.hess.CIs = MARSSparamCIs(kemfit100)
kem.with.hess.CIs #print with se's and CIs
#estimate CIs using a parametric bootstrap
kem.with.boot.CIs = MARSSparamCIs(kemfit100,
  method="parametric", nboot=10)
#nboot should be more like 1000, but set low for example sake
kem.with.boot.CIs #print with se's, CIs, and bias est

#fit a model with 5 hidden states (default)
kemfit = MARSS(dat, silent=TRUE) #suppress printing
kemfit #print information on the marssMLE object
show(kemfit) #look at the raw marssMLE object

#fit a model with 5 correlated hidden states
kemfit = MARSS(dat, constraint=list(Q="unconstrained"))

#fit a model with 5 correlated hidden states
# with one variance and one covariance
kemfit = MARSS(dat, constraint=list(Q="equalvarcov"))
```

```

#fit a model with 5 independent hidden states
#where each observation time series is independent
#the hidden trajectories 1-3 & 4-5 share their U parameter
kemfit = MARSS(dat, constraint=list(U=factor(c("N", "N", "N", "S", "S"))))

#same model, but with fixed independent observation errors
kemfit = MARSS(dat, constraint=list(U=factor(c("N", "N", "N", "S", "S")),
  R=diag(0.01,5)),control=list(minit=100))

#fit a model with 2 hidden states (north and south)
#where observation time series 1-3 are north and 4-5 are south
#Make the hidden state process independent with same process var
#Make the observation errors different but independent
#Make the growth parameters (U) the same
kemfit = MARSS(dat, constraint=list(Z=factor(c("N", "N", "N", "S", "S")),
  Q="diagonal and equal",R="diagonal and unequal",U="equal"),
  control=list(minit=100))

#print the model followed by the marssMLE object
kemfit$model
kemfit

#simulate some new data from our fitted model
sim.data=MARSSsimulate(kemfit$par, nsim=10, tSteps=100)

#Compute bootstrap AIC for the model; this takes a long time
kemfit.with.AICb = MARSSaic(kemfit, output = "AICbp")
kemfit.with.AICb

## End(Not run)

```

MARSSaic

AIC for MARSS models

Description

Calculates AIC, AICc, a parametric bootstrap AIC (AICbp) and a non-parametric bootstrap AIC (AICbb). This is a base function in the [MARSS-package](#).

Usage

```

MARSSaic(MLEobj, output = c("AIC", "AICc"),
  Options = list(nboot = 1000, return.logL.star = FALSE,
  silent = FALSE))

```

Arguments

MLEobj An object of class `marssMLE`. This object must have a `$par` element containing MLE parameter estimates from e.g. `MARSSkem()`.

output	A vector containing one or more of the following: "AIC", "AICc", "AICbp", "AICbb", "AICi", "boot.params". See Details.
Options	A list containing: <ul style="list-style-type: none"> • <code>nboot</code> Number of bootstraps (positive integer) • <code>return.logL.star</code> Return the log-likelihoods for each bootstrap? (T/F) • <code>silent</code> Suppress printing of the progress bar during AIC bootstraps? (T/F)

Details

When sample size is small, Akaike's Information Criterion (AIC) under-penalizes more complex models. The most commonly used small sample size corrector is AICc, which uses a penalty term of $Kn/(n-K-1)$, where K is the number of estimated parameters. However, for time series models, AICc still under-penalizes complex models; this is especially true for MARSS models.

Two small-sample estimators specific for MARSS models have been developed. Cavanaugh and Shumway (1997) developed a variant of bootstrapped AIC using Stoffer and Wall's (1991) bootstrap algorithm ("AICbb"). Holmes and Ward (2010) developed a variant on AICb ("AICbp") using a parametric bootstrap. The parametric bootstrap permits AICb calculation when there are missing values in the data, which Cavanaugh and Shumway's algorithm does not allow. More recently, Bengtsson and Cavanaugh (2006) developed another small-sample AIC estimator, AICi, based on fitting candidate models to multivariate white noise.

When output includes both "AICbp" and "boot.params", the bootstrapped parameters from "AICbp" will be added to `MLEobj`.

Value

Returns the `marssMLE` object that was passed in with additional AIC components added on top as specified in the 'output' argument.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

`eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov`

References

Holmes, E. E. and E. J. Ward. 2010. Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112; this is the user manual accessed via `show.doc(MARSS, manual)`

Bengtsson, T., and J. E. Cavanaugh. 2006. An improved Akaike information criterion for state-space model selection. *Computational Statistics & Data Analysis* 50:2635-2654.

Cavanaugh, J. E., and R. H. Shumway. 1997. A bootstrap variant of AIC for state-space model selection. *Statistica Sinica* 7:473-496.

See Also

[MARSSboot](#)

Examples

```
dat = t(harborSealWA)
dat = dat[2:nrow(dat),]
kem = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1)),
  R="diagonal and unequal"), control=list(minit=100))
kemAIC = MARSSaic(kem, output=c("AIC", "AICc"))
```

MARSSapplynames *Names for marssMLE Object Components*

Description

Puts names on the matrix components of `marssMLE` and `marssm` objects. This is a utility function in the `MARSS-package`.

Usage

```
MARSSapplynames(obj, Y.names = NA, X.names = NA, x0.names = NA,
  V0.names = NA, U.names = NA, A.names = NA, R.names = NA,
  Q.names = NA, B.names = NA, Z.names = NA,
  rows = TRUE, cols = TRUE)
```

Arguments

<code>obj</code>	An object of class <code>marssMLE</code> or <code>marssm</code> .
<code>Y.names</code>	Vector of names for observed time series.
<code>X.names</code>	Vector of names for the hidden state trajectories.
<code>x0.names</code>	Vector of names for <code>MLEobj\$par\$x0</code> , <code>MLEobj\$start\$x0</code> , <code>MLEobj\$model\$fixed\$x0</code> and <code>MLEobj\$model\$free\$x0</code> .
<code>V0.names</code>	Names for <code>MLEobj\$par\$V0</code>
<code>U.names</code>	Names for <code>MLEobj\$par\$U</code>
<code>A.names</code>	Names for <code>MLEobj\$par\$A</code>
<code>R.names</code>	Names for <code>MLEobj\$par\$R</code>
<code>Q.names</code>	Names for <code>MLEobj\$par\$Q</code>
<code>B.names</code>	Names for <code>MLEobj\$par\$B</code>
<code>Z.names</code>	Row and col names for <code>MLEobj\$par\$Z</code>
<code>rows</code>	Add row names?
<code>cols</code>	Add column names to B, Q, R and V0 matrices?

Details

Default behavior will use names rownames of data (if available, and if not Y1, Y2, ...) for the Ys and all matrices that are $n \times \dots$ and use X1, X2, ... for the Xs and all matrices that are $m \times \dots$.

Value

The object passed in, with row and column names on matrices as specified.

Author(s)

Eli Holmes and Kellie Wills, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, kellie(dot)wills(at)noaa(dot)gov

See Also

[marssMLE](#) [marssm](#)

Examples

```
dat = t(harborSealWA)
dat = dat[2:nrow(dat),]
kem = MARSS(dat)
kemNamed = MARSSapplynames(kem)
```

MARSSboot

Bootstrap MARSS Parameter Estimates

Description

Creates bootstrap parameter estimates and simulated (or bootstrapped) data (if appropriate). This is a base function in the [MARSS-package](#).

Usage

```
MARSSboot(MLEobj, nboot = 1000,
  output = "parameters", sim = "parametric",
  param.gen = "KalmanEM", control = NULL, silent = FALSE)
```

Arguments

MLEobj	An object of class marssMLE . Must have a \$par element containing MLE parameter estimates.
nboot	Number of bootstraps to perform.
output	Output to be returned: "data", "parameters" or "all".
sim	Type of bootstrap: "parametric" or "innovations". See Details.
param.gen	Parameter generation method: "hessian" or "KalmanEM".
control	The options in MLEobj\$control are used by default. If supplied here, must contain all of the following: max.iter Maximum number of EM iterations.

<code>tol</code>	Optional tolerance for log-likelihood change. If log-likelihood decreases less than this amount relative to the previous iteration, the EM algorithm exits.
<code>iter.V0</code>	The value of V0 to be used in place of 0 when x0 is treated as fixed and V0=0. See manual for a discussion of the setting of diffuse priors for iterative parts of the maximization algorithms.
<code>silent</code>	Suppresses printing of progress bar.

Details

Approximate confidence intervals (CIs) on the model parameters can be calculated by numerically estimating the Hessian matrix (the matrix of partial 2nd derivatives of the parameter estimates). The Hessian CIs are based on the asymptotic normality of ML estimates under a large-sample approximation. CIs that are not based on asymptotic theory can be calculated using parametric and non-parametric bootstrapping.

Stoffer and Wall (1991) present an algorithm for generating CIs via a non-parametric bootstrap for state-space models (`sim = "innovations"`). The basic idea is that the Kalman filter can be used to generate estimates of the residuals of the model fit. These residuals are then standardized and resampled and used to generate bootstrapped data using the MARSS model and its maximum-likelihood parameter estimates. One of the limitations of the Stoffer and Wall algorithm is that it cannot be used when there are missing data, unless all data at time t are missing. An alternative approach is a parametric bootstrap (`sim = "parametric"`), in which the ML parameter estimates are used to produce bootstrapped data directly from the state-space model.

Value

A list with the following components:

<code>boot.params</code>	Matrix (number of params x nboot) of parameter estimates from the bootstrap.
<code>boot.data</code>	Array (n x t x nboot) of simulated (or bootstrapped) data (if requested and appropriate).
<code>model</code>	The <code>marssm</code> object that was passed in via <code>MLEobj\$model</code> .
<code>nboot</code>	Number of bootstraps performed.
<code>output</code>	Type of output returned.
<code>sim</code>	Type of bootstrap.
<code>param.gen</code>	Parameter generation method: "hessian" or "KalmanEM".

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

`eli(dot)holmes(at)noaa(dot)gov`, `eric(dot)ward(at)noaa(dot)gov`

References

Holmes, E. E. and E. J. Ward. 2010. Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112; this is the user manual accessed via `show.doc(MARSS, manual)`

Stoffer, D. S., and K. D. Wall. 1991. Bootstrapping state-space models: Gaussian maximum likelihood estimation and the Kalman filter. *Journal of the American Statistical Association* 86:1024-1033.

Cavanaugh, J. E., and R. H. Shumway. 1997. A bootstrap variant of AIC for state-space model selection. *Statistica Sinica* 7:473-496.

See Also

[marssMLE](#) [marssm](#) [MARSSaic](#)

Examples

```
## Not run:
#nboot is set low in these example in order to run quickly
#normally nboot would be >1000 at least
dat = t(harborSealnomiss)
dat = dat[2:nrow(dat),]
kem = MARSS(dat, constraint=list(Q="diagonal and equal"),
  control=list(minit=100))
hess.list = MARSSboot(kem, param.gen="hessian", nboot=10)
# (no missing values)
boot.list = MARSSboot(kem, output="all", sim="innovations", nboot=10)

# Bootstrap CIs for data with missing values
dat = t(harborSealWA)
dat = dat[2:nrow(dat),]
kem = MARSS(dat, constraint=list(Q="diagonal and equal"),
  control=list(minit=100))
boot.list = MARSSboot(kem, output="all", sim="parametric", nboot=10)

## End(Not run)
```

MARSScheckdims

MARSS Utilities

Description

This is a utility function in the [MARSS-package](#) for checking MARSS matrix dimensions and parameter lists.

Usage

```
MARSScheckdims(e1, target, n, m)
MARSScheckpar(parList, n, m)
```

Arguments

<code>el</code>	Name of a MARSS list element ("A", "B", "Q", "R", "U", "V0", "x0", "Z").
<code>target</code>	List to be checked.
<code>n</code>	Number of time series.
<code>m</code>	Number of state processes.
<code>parList</code>	MARSS parameter list.

Value

TRUE if no problems; otherwise a message describing the problems.

Author(s)

Kellie Wills, NOAA, Seattle, USA.
kellie(dot)wills(at)noaa(dot)gov

MARSShessian	<i>MARSS Parameter Variance-Covariance Matrix from the Hessian Matrix</i>
--------------	---

Description

Calculates approximate parameter variance-covariance matrix and appends it to a `marssMLE` object. This is a utility function in the [MARSS-package](#).

Usage

```
MARSShessian(MLEobj)
```

Arguments

<code>MLEobj</code>	An object of class <code>marssMLE</code> . This object must have a <code>\$par</code> element containing MLE parameter estimates from e.g. MARSSkem .
---------------------	---

Details

Uses `fdHess` from package `nlme` to numerically estimate the Hessian matrix (the matrix of partial 2nd derivatives of the parameter estimates). Hessian CIs are based on the asymptotic normality of ML estimates under a large-sample approximation.

Value

`MARSShessian()` returns the `marssMLE` object passed in along with additional components `Hessian`, `gradient`, `parMean` and `parSigma` computed by the `MARSShessian` function.

Author(s)

Eli Holmes, NOAA, Seattle, USA.
 eli(dot)holmes(at)noaa(dot)gov

See Also

[MARSSparamCIs](#) [marssMLE](#)

Examples

```
## Not run:
dat = t(harborSeal)
dat = dat[2:nrow(dat),]
MLEobj = MARSS(dat)
MLEobj.w.hessian = MARSShessian(MLEobj)

## End(Not run)
```

 MARSSinits

Initial Values for MLE

Description

Sets up generic starting values for parameters for maximum-likelihood estimation algorithms that use an iterative maximization routine needing starting values. Examples of such algorithms are the Kalman-EM algorithm in [MARSSkem](#) and Newton methods in [MARSSoptim](#). This is a utility function in the [MARSS-package](#).

Usage

```
MARSSinits(modelObj, inits=list(B=1, U=0, Q=0.05,
  A=0, R=0.05, x0=-99, V0=10))
```

Arguments

modelObj	An object of class marssm . <code>MARSSinits</code> uses three elements of the model object. <ul style="list-style-type: none"> • <code>data</code> The data element is used to determine n, the dimension of the y in the MARSS model. • <code>fixed</code> The fixed matrices are used to determine which (if any) model parameters are fixed. • <code>miss.value</code> Used if a linear regression through the data is used to construct inits for x_0. In this case, the missing values are replaced by NA.
inits	A list of up to 8 values to construct starting matrices for each parameter in a MARSSmodel.

Details

Creates an `inits` parameter list for use by iterative maximization algorithms.

Defaults values for `inits` is supplied in `MARSSsettings.R`. The user can alter these and supply any of the following (m is the dim of X and n is the dim of Y in the MARSS model):

- `elem=A, U` A numeric vector or matrix which will be constructed into `inits$elem` by the command `array(inits$elem, dim=c(n or m, 1))`. If `elem` is fixed in the model, any `inits$elem` values will be overridden and replaced with the fixed value. Default is `array(0, dim=c(n or m, 1))`.
- `elem=Q, R, B` A numeric vector or matrix. If length equals the length `modelObj$fixed$elem` then `inits$elem` will be constructed by `array(inits$elem, dim=dim(modelObj$fixed$elem))`. If length is 1 or equals m or n then `inits$elem` will be constructed into a diagonal matrix by the command `diag(inits$elem, m or n)`. If `elem` is fixed in the model, any `inits$elem` values will be overridden and replaced with the fixed value. Default is `diag(0.05, m or n)` for `Q` and `R`. Default is `diag(1, m)` for `B`.
- `x0` If `inits$x0=-99`, then starting values for `x0` are estimated by a linear regression through the count data assuming `A=0`. This will be a poor start if `inits$A` is not 0. If `inits$x0` is a numeric vector or matrix, `inits$x0` will be constructed by the command `array(inits$x0, dim=c(m, 1))`. If `x0` is fixed in the model, any `inits$x0` values will be overridden and replaced with the fixed value. Default is `inits$x0=-99`.
- `Z` If `Z` is fixed in the model, `inits$Z` set to the fixed value. If `Z` is not fixed, then the user must supply `inits$Z`. There is no default.
- `elem=V0` `MARSSkem` and `MARSSoptim` will override `inits$V0` and use a diffuse prior (set with `marssMLE$control$iter.V0` if any `x0` are estimated and will use `modelObj$fixed$V0` if `x0` is fixed. The first case corresponds to `x0` fixed but unknown and treated as an estimated parameter with `V0=0`. The diffuse prior for `V0` is used only during the iterations of maximization to all `x0` estimation and then `V0` is reset to 0 for the final likelihood calculation. The second case corresponds to using a prior for the initial state. Although `inits$V0` is ignored by the fitting algorithm in MARSS 1.0, it is set here for forward compatibility. If `inits$V0` is a vector or matrix with length equal to the length `modelObj$fixed$V0` then `inits$V0` will be constructed by the command `array(inits$V0, dim=c(m, m))`. If length is 1 or equal to m then `inits$V0` will be constructed into a diagonal matrix by the command `diag(inits$V0, m)`. If `x0` is fixed in the model, `inits$V0` values will be overridden and replaced with `modelObj$fixed$V0`. Default is `diag(10, m)`.

Value

A list with 8 matrices `A`, `R`, `B`, `U`, `Q`, `x0`, `V0`, `Z`, specifying initial values for parameters in a MARSS model.

Author(s)

Eli Holmes, NOAA, Seattle, USA.

`eli(dot)holmes(at)noaa(dot)gov`

See Also

[marssm](#) [MARSSkem](#) [MARSSoptim](#)

MARSSinnovationsboot

Bootstrapped Data using Stoffer and Wall's Algorithm

Description

Creates bootstrap data via sampling from the standardized innovations matrix. This is a base function in the [MARSS-package](#).

Usage

```
MARSSinnovationsboot(MLEobj, nboot = 1000, minIndx = 3)
```

Arguments

MLEobj	An object of class marssMLE . This object must have a <code>\$par</code> element containing MLE parameter estimates from e.g. MARSSkem or MARSS . This algorithm may not be used if there are missing datapoints in the data.
nboot	Number of bootstraps to perform.
minIndx	Number of innovations to skip. Stoffer & Wall suggest not sampling from innovations 1-3.

Details

Stoffer and Wall (1991) present an algorithm for generating CIs via a non-parametric bootstrap for state-space models. The basic idea is that the Kalman filter can be used to generate estimates of the residuals of the model fit. These residuals are then standardized and resampled and used to generate bootstrapped data using the MARSS model and its maximum-likelihood parameter estimates. One of the limitations of the Stoffer and Wall algorithm is that it cannot be used when there are missing data, unless all data at time t are missing.

Value

A list containing the following components:

<code>boot.states</code>	Array (dim is $m \times tSteps \times nboot$) of simulated state processes.
<code>boot.data</code>	Array (dim is $n \times tSteps \times nboot$) of simulated data.
<code>model</code>	MARSS model (<code>\$model</code> element of the marssMLE object).
<code>nboot</code>	Number of bootstraps performed.

m is the number state processes (x in the MARSS model) and n is the number of observation time series (y in the MARSS model).

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.
 eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

References

Stoffer, D. S., and K. D. Wall. 1991. Bootstrapping state-space models: Gaussian maximum likelihood estimation and the Kalman filter. *Journal of the American Statistical Association* 86:1024-1033.

See Also

[stdInnov](#) [MARSSparamCIs](#) [MARSSboot](#)

Examples

```
dat = t(harborSealnomiss)
dat = dat[2:nrow(dat),]
MLEobj = MARSS(dat, constraint=list(Q="diagonal and equal"),
  control=list(minit=100))
boot.obj = MARSSinnovationsboot(MLEobj)
```

MARSSkem

Maximum Likelihood Estimation for Multivariate Autoregressive State-Space Models

Description

`MARSSkem()` performs maximum-likelihood estimation, using an EM algorithm for constrained and unconstrained MARSS models. This is one of the base functions in the [MARSS-package](#).

Usage

```
MARSSkem(MLEobj)
```

Arguments

`MLEobj` An object of class `marssMLE`.

Details

Objects of class `marssMLE` may be built from scratch but are easier to construct using `MARSS` with `MARSS(..., fit=FALSE)`.

Options for `MARSSkem()` may be set using `MLEobj$control`, as follows:

`MLEobj$control$minit` Minimum number of EM iterations. You can use this to force the algorithm to do a certain number of iterations. This is helpful if your soln is not converging.

MLEobj\$control\$maxit Maximum number of EM iterations.

MLEobj\$control\$abstol Tolerance for log-likelihood change. If log-likelihood changes less than this amount relative to the previous iteration, the EM algorithm exits.

MLEobj\$control\$iter.V0 This is the value to which V0 will be set during the EM algorithm (!=0). See Details.

MLEobj\$control\$trace A positive integer. If not 0, a record will be created of each variable over all EM iterations and detailed warning messages (if appropriate) will be printed.

MLEobj\$control\$safe If TRUE, MARSSkem will rerun `MARSSkf` after each individual parameter update rather than only after all parameters are updated. The latter is slower and unnecessary for many models, but in some cases, the safer and slower algorithm is needed because the ML parameter matrices have high condition numbers.

MLEobj\$control\$MCInit If TRUE, Monte Carlo initialization will be performed by `MARSSmccinit`.

MLEobj\$control\$numInits Number of random initial value draws to be used with `MARSSmccinit`. Ignored if `control$MCInit=FALSE`.

MLEobj\$control\$numInitSteps Maximum number of EM iterations for each random initial value draw to be used with `MARSSmccinit`. Ignored if `control$MCInit=FALSE`.

MLEobj\$control\$boundsInits Length 6 list. Each component is a length 2 vector of bounds on the uniform distributions from which initial values will be drawn to be used with `MARSSmccinit()`. Ignored if `control$MCInit=FALSE`. See Examples.

MLEobj\$control\$silent Suppresses printing of progress bars, error messages, warnings and convergence information.

Value

The `marssMLE` object which was passed in, with additional components:

<code>method</code>	String "kem".
<code>kf</code>	Kalman filter output.
<code>iter.record</code>	If <code>MLEobj\$control\$trace = TRUE</code> , a list with <code>par</code> = a record of each estimated parameter over all EM iterations and <code>logLik</code> = a record of the log likelihood with <code>V0</code> set to the value of <code>iter.V0</code> . Note this is different than the log likelihood with <code>V0 = 0</code> (which is the final log likelihood value).
<code>numIter</code>	Number of iterations needed for convergence.
<code>convergence</code>	Did estimation converge successfully? convergence=0 Converged in less than <code>MLEobj\$control\$maxit</code> iterations and no evidence of degenerate solution. convergence=1 Maximum number of iterations <code>MLEobj\$control\$maxit</code> was reached before <code>MLEobj\$control\$abstol</code> condition was satisfied. convergence=10 Some of the variance elements appear to be degenerate. The <code>MLEobj\$control\$abstol</code> condition was satisfied before the maximum number of iterations <code>MLEobj\$control\$maxit</code> was reached, but some of the variance elements appear to still be changing. convergence=52 The algorithm was abandoned due to numerical errors. Usually this means one of the variances either went to zero or to all elements being equal. Try setting <code>control\$trace=1</code> to view a detailed error report.

<code>logLik</code>	Log-likelihood.
<code>states</code>	State estimates from the Kalman filter.
<code>states.se</code>	Confidence intervals based on state standard errors, see caption of Fig 6.3 (p. 337) Shumway & Stoffer.
<code>errors</code>	Any error messages.

Discussion

To ensure that the global maximum-likelihood values are found, it is recommended that initial parameter values be set using Monte Carlo initialization (`MLEobj$control$MCInit = TRUE`), particularly if the model is not a good fit to the data. This requires more computation time, but reduces the chance of the algorithm terminating at a local maximum and not reaching the true MLEs. For many models, this is unnecessary, but answers should be checked using an initial conditions search before reporting final values.

`MARSSkem()` calls a Kalman filter/smoothen (`MARSSkf`) for hidden state estimation. The algorithm allows two options for the initial state conditions: x at $t=1$. Either x_0 is treated as fixed but unknown (estimated); in this case, `fixed$V0=0` and x_0 is estimated. This is the default behavior. In the second case, the initial conditions are specified with a known prior, `fixed$x0` and `fixed$V0`, and x_0 is not estimated. In the first case, x_0 fixed but unknown and to be estimated, `V0=0` but the algorithm cannot be run with this since x_0 would never move from its initial value at `iteration=1`. Instead, during the EM iterations, a diffuse prior is used. This is done by setting `MLEobj$control$iter.V0` to a large value, say 10. A small value will cause the algorithm to converge very slowly and 0 will generate an error. Before reporting the final log likelihood, `MARSSkem()` runs the Kalman filter with the maximum-likelihood parameter estimates (using the diffuse prior) and `V0 = 0` to obtain the correct likelihood. This approach works well and is easy to implement. However when one fits a model with x_0 having shared values AND an unconstrained R matrix, an ill-conditioned matrix tends to appear in one of the steps of the Kalman filter algorithm (because `V0` must have elements with 100 percent correlation if you say that some x_0 's have the same value). `MARSSkem()` will report warnings and errors if this happens. Switching to a diagonal R or an unconstrained x_0 will fix the ill-conditioning problems. See the manual for discussion about how `V0` is treated in the algorithm.

If you get errors, it generally means that the solution involves an ill-conditioned matrix. For example, your Q or R matrix is going to a value in which all elements have the same value, for example zero. If for example, you tried to fit a model with fixed and high R matrix and the variance in that R matrix was much higher than what is actually in the data, then you might drive Q to zero. Also if you try to fit a structurally inadequate model, then it is not unusual that Q will be driven to zero. For example, if you fit a model with 1 hidden state trajectory to data that clearly have 2 quite different hidden state trajectories, you might have this problem. Comparing the likelihood of this model to a model with more structural flexibility should reveal that the structurally inflexible model is inadequate (much lower likelihood).

If you get a warning that the solution appears to be degenerate, it means that some of the elements in Q or R are going to zero and the log-likelihood is changing very slowly. You can either try decreasing `control$abstol` dramatically (e.g. 1E-6), use a Newton finisher `MARSSoptim(MLEobj)`, or fix the degenerate values to something very small (e.g. 1E-8) and re-estimate. Try `find.degenerate(MLEobj)` using the output from the `MARSSkem` call to find the degenerate elements.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

References

R. H. Shumway and D. S. Stoffer (2006). Chapter 6 in Time Series Analysis and its Applications. Springer-Verlag, New York.

Ghahramani, Z. and Hinton, G. E. (1996) Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, University of Toronto, Dept. of Computer Science.

Harvey, A. C. (1989) Chapter 5 in Forecasting, structural time series models and the Kalman filter. Cambridge University Press, Cambridge, UK.

The user manual: Holmes, E. E. and E. J. Ward (2010) Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112 type `show.doc(MARSS, manual)` to see.

The EM algorithm: Holmes, E. E. (2010). Derivation of the EM algorithm for constrained and unconstrained multivariate autoregressive state-space (MARSS) models. type `show.doc(MARSS, Holmes2010)` to see.

See Also

[MARSSmccinit](#), [MARSSkf](#), [marssMLE](#), [MARSSoptim](#), [find.degenerate](#)

Examples

```
dat = t(harborSeal)
dat = dat[2:nrow(dat),]
#you can use MARSS to construct a proper marssMLE object.
MLEobj = MARSS(dat, fit=FALSE)
kemfit = MARSSkem(MLEobj)
## Not run:
#see what a Newton method would find;
# wrapped in try because it tends to throw numerical errors
bfgsfit=kemfit; bfgsfit$method="BFGS"
bfgsfit = try(MARSSoptim(bfgsfit))
#look for degenerate estimates
#Use MARSS to do the fit
kemfit2 = MARSS(dat, control=list(minit=100))
#this will make a plot that will show if the vars converged.
find.degenerate(kemfit2)

## End(Not run)
```

`MARSSkemcheck`*Model Checking for MLE objects passed to MARSSkem*

Description

This is a helper function in the [MARSS-package](#) that checks that the model can be handled by the [MARSSkem](#) algorithm.

Usage

```
MARSSkemcheck(modelObj)
```

Arguments

`modelObj` An object of class `marssm`.

Value

A list with of the model elements A, B, Q, R, U, x0, Z, V0 specifying the structure of the model using text strings).

Author(s)

Eli Holmes, NOAA, Seattle, USA.

`eli(dot)holmes(at)noaa(dot)gov`

See Also

[marssm](#) [MARSSkem](#)

`MARSSkf`*Kalman Filtering and Smoothing*

Description

Implements the Kalman filter/smoother for MARSS models. This is a base function in the [MARSS-package](#).

Usage

```
MARSSkf(y, parList, missing.matrix = "not given",  
miss.value = "not given", init.state="x10", debugkf=FALSE)
```

Arguments

<code>y</code>	A matrix (not dataframe), sites (rows) x years (columns). See Details regarding handling of missing values.
<code>parList</code>	A list with 8 matrices Z, A, R, B, U, Q, x0, V0, specifying parameter values. An example is the <code>par</code> element in a <code>marssMLE</code> object.
<code>missing.matrix</code>	Optional matrix specifying which observations are missing. See Details.
<code>miss.value</code>	How are missing values represented in the data? Either <code>miss.value</code> or <code>missing.matrix</code> must be supplied. If both are supplied, then <code>miss.value</code> will be ignored with no warning(!).
<code>init.state</code>	Is the initial state, x0, treated as E(x) at time t=0 (<code>init.state="x00"</code>) or E(x) at t=1 (<code>init.state="x10"</code>)? Default is <code>init.state="x10"</code> (note! for Shumway and Stoffer's Kalman filter use <code>init.state="x00"</code>). See Details.
<code>debugkf</code>	Return detailed error messages?

Details

For state space models, the Kalman filter and smoother provide optimal (minimum mean square error) estimates of the hidden states. The Kalman filter is a forward recursive algorithm which computes estimates of the states $x(t)$ conditioned on the data up to time t . The Kalman smoother is a backward recursive algorithm which starts at time T and works backwards to $t = 1$ to provide estimates of the states conditioned on all data.

Missing values in the data may be handled in one of two ways: 1. Missing values may be replaced with zeroes prior to passing to `MARSSkf()`. Argument `missing.matrix` must then be a matrix of the same dimensions as the data, with 0 in the positions of observed values and 1 in the positions of missing values. 2. Data containing missing values may be passed in. Argument `miss.value` must then be the code used to represent missing values. The function requires that you specify either a missing matrix or a `miss.value`. If there are no missing values, just set `miss.value` to a value that is not in your data (like -99).

The expected value of the initial state, x_0 , is an estimated parameter (or treated as a prior). This $E(\text{initial state})$ can either be treated in two different ways. One can treat it as x_{00} , meaning $E(x \text{ at } t=0 \mid y \text{ at } t=0)$, and then compute x_{10} , meaning $E(x \text{ at } t=1 \mid y \text{ at } t=0)$, from x_{00} . Or one can simply treat the initial state as x_{10} , meaning $E(x \text{ at } t=1 \mid y \text{ at } t=0)$. The approaches are equivalent, but the likelihood is written slightly differently in each case (sum over 2 to T versus 1 to T) and you need your likelihood calculation to correspond to how the initial state is treated in your model (either x_{00} or x_{10}). The EM algorithm in the MARSS package (`MARSSkem`) follows Ghahramani's derivation and uses x_{10} , while Shumway and Stoffer uses x_{00} . The `init.state` argument specifies whether x_{00} (`init.state="x00"`) or x_{10} (`init.state="x10"`) is used. The default is `init.state="x10"`.

Value

A list with the following components (n is the number of state processes). Names ending in "T" are estimates from the Kalman smoother; J is also smoother output. Other components are output from the Kalman filter.

<code>xtT</code>	State estimates $E[x(t) y(1:T)]$ ($n \times T$ matrix).
<code>VtT</code>	State covariances $E[V(t) y(1:T)]$ ($n \times n \times T$ array).
<code>Vtt1T</code>	Conditional error covariances $E[V(t,t-1) y(1:T)]$ ($n \times n \times T$).
<code>x0T</code>	Initial state mean estimates ($n \times 1$).
<code>V0T</code>	Estimate of initial state covariance matrix ($n \times n$).
<code>J</code>	$n \times n \times T$
<code>xtt1</code>	Forecasts $E[x(t) Y(t-1)]$ ($n \times T$).
<code>Vtt</code>	State covariance estimates $E[V(t) y(1:t)]$ ($n \times n \times T$).
<code>Vtt1</code>	Conditional error covariances $E[V(t,t-1) y(1:t)]$ ($n \times n \times T$).
<code>Kt</code>	Kalman gain ($n \times n \times T$).
<code>Innov</code>	Innovations $y(t) - E[y(t) Y(t-1)]$ ($n \times T$) ($n \times n \times T$).
<code>Sigma</code>	Innovations variances.
<code>logLik</code>	Log-likelihood computed from <code>mssm.params</code> and innovations.
<code>errors</code>	Any error messages due to ill-conditioned matrices.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

`eli(dot)holmes(at)noaa(dot)gov`, `eric(dot)ward(at)noaa(dot)gov`

References

P. J. Brockwell and R. A. Davis (1991). Time Series: Theory and Methods.

A. C. Harvey (1989). Chapter 5, Forecasting, Structural Time Series Models and the Kalman Filter. Cambridge University Press.

R. H. Shumway and D. S. Stoffer (2006). Chapter 6, Time Series Analysis and its Applications. Springer-Verlag, New York.

The user manual: Holmes, E. E. and E. J. Ward (2010) Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112 type `show.doc` (MARSS, manual) to see.

See Also

[MARSS](#) [marssm](#) [MARSSkem](#)

Examples

```
dat = t(harborSeal)
dat = dat[2:nrow(dat),]
#you can use MARSS to construct a MLEobj
#MARSS calls MARSSinits to construct default initial values
MLEobj = MARSS(dat, fit=FALSE)
#Compute the kf output at the params used for the inits
kfList = MARSSkf(dat, MLEobj$start, miss.value=-99)
```

marssm

*Model Objects***Description**

These are model objects and utility functions for model objects in the package [MARSS-package](#). `marssm()` creates multivariate autoregressive state space model objects. `is.marssm()` ensures model consistency. `as.marssm()` attempts to coerce its argument to a `marssm` object.

Usage

```
marssm(data = NULL, fixed, free, miss.value = -99)
is.marssm(modelObj)
as.marssm(wrapperObj)
```

Arguments

<code>data</code>	An optional matrix (not dataframe), sites (rows) x years (columns), of observed population abundances. If the algorithm is to be applied to log-abundance, the log transformation should be done before the data is passed in.
<code>fixed</code>	A list with 8 matrices Z, A, R, B, U, Q, x0, V0, specifying which elements of each parameter are fixed. See Details.
<code>free</code>	A list with 8 matrices Z, A, R, B, U, Q, x0, V0, specifying which elements of each parameter are to be estimated. See Details.
<code>miss.value</code>	How are missing values represented in the data?
<code>modelObj</code>	An object of class <code>marssm</code> .
<code>wrapperObj</code>	An object of a wrapper class popWrap .

Details

A `marssm` object is an R representation of a MARSS model along with the data. Data for `marssm()` consists of multivariate time series data in which time is across columns and the `n` observed time series are in the `n` different rows.

The MARSS model is

$$\mathbf{x}(t+1) = \mathbf{B} \mathbf{x}(t) + \mathbf{U} + \mathbf{w}(t), \text{ where } \mathbf{w}(t) \sim \text{MVN}(\mathbf{0}, \mathbf{Q})$$

$$\mathbf{y}(t) = \mathbf{Z} \mathbf{x}(t) + \mathbf{A} + \mathbf{v}(t), \text{ where } \mathbf{v}(t) \sim \text{MVN}(\mathbf{0}, \mathbf{R})$$

$$\mathbf{x}(1) \sim \text{MVN}(\mathbf{x0}, \mathbf{V0})$$

In the `marssm` object, the MARSS model is specified by fixed/free pairs of matrices for each parameter: B, U, Q, Z, A, R, x0, V0. The dimensions for `fixed` and `free` matrices must be:

$$\mathbf{Z} \quad n \times m \quad (m \leq n)$$

$$\mathbf{B} \quad m \times m$$

$$\mathbf{U} \quad m \times 1$$

Q $m \times m$ **A** $n \times 1$ **R** $n \times n$ **x0** $m \times 1$ **V0** $m \times m$

The matrices in `fixed` and `free` work as pairs to specify the fixed and free elements for each parameter. In `fixed`, fixed elements must be numbers; values that are not fixed (i.e. are to be estimated) should be denoted NA. Elements in `free` will be interpreted as names for the free elements (even if they are numbers). Identical elements within a parameter matrix will be constrained to have the same value. Non-free (i.e. fixed) values should be denoted with NA, not 0, since the code will interpret 0 as "0" and assume that the user wants those parameters to be coded with the name "0" and to be estimated. See the manual (`show.doc(MARSS, manual)`) for many examples of MARSS models and their specification.

Value

An object of class "marssm".

<code>data</code>	Data supplied by user.
<code>fixed</code>	A list with 8 matrices Z, A, R, B, U, Q, x0, V0.
<code>free</code>	A list with 8 matrices Z, A, R, B, U, Q, x0, V0.
<code>M</code>	An array of dim $n \times n \times t$ (an $n \times n$ missing values matrix for each time point). Each matrix is diagonal with 0 at the i,i value if the i -th value of y is missing, and 1 otherwise.
<code>miss.value</code>	Specifies missing value representation. Default is -99

Author(s)

Kellie Wills, NOAA, Seattle, USA.

kellie(dot)wills(at)noaa(dot)gov

See Also

[popWrap](#)

Examples

```
n = m = 5
fixed = free = vector("list")

free$Q=array(seq(1,m*m),dim=c(m,m)); fixed$Q=array(NA,dim=c(m,m))
free$R=array(NA,dim=c(n,n)); diag(free$R)=1
fixed$R=array(0,dim=c(n,n)); diag(fixed$R)=NA
free$Z=array(NA,dim=c(m,m))
fixed$Z=array(0,dim=c(m,m)); diag(fixed$Z)=1
free$U=array(seq(1,m),dim=c(m,1)); fixed$U = array(NA,dim=c(m,1))
```

```

fixed$A = matrix(NA,n,1); free$A = matrix(1:n,n,1)
free$B=array(NA,dim=c(m,m));
fixed$B=array(0,dim=c(m,m)); diag(fixed$B)=1
free$x0=array(seq(1,m),dim=c(m,1)); fixed$x0 = array(NA,dim=c(m,1));
free$V0=array(NA,dim=c(m,m)); fixed$V0 = array(0,dim=c(m,m))

m1 <- marssm(fixed=fixed, free=free)
is.marssm(m1)

dat = t(harborSeal)
dat = dat[2:nrow(dat),]
aa = MARSS:::allowed$kem
wrapperObj = popWrap(dat, allowed=aa, method="kem")
modelObj = as.marssm(wrapperObj)

```

marssm-class	<i>Class "marssm"</i>
--------------	-----------------------

Description

marssm objects describe the multivariate autoregressive state space models used in the package [MARSS-package](#).

Methods

print signature(x = "marssm"): ...
summary signature(object = "marssm"): ...

Author(s)

Kellie Wills, NOAA, Seattle, USA. [kellie\(dot\)wills\(at\)noaa\(dot\)gov](mailto:kellie(dot)wills(at)noaa(dot)gov)

MARSSmcinit	<i>Monte Carlo Initialization</i>
-------------	-----------------------------------

Description

Performs a Monte Carlo search for optimal initial conditions iterative maximization algorithms ([MARSSkem](#) and [MARSSoptim](#)). This is a utility function in the [MARSS-package](#).

Usage

```
MARSSmcinit(MLEobj)
```

Arguments

`MLEobj` An object of class `marssMLE`.

Details

It is recommended that initial parameter values be set using `MARSSmcinit()`, particularly if the model is not a good fit to the data. This requires more computation time, but reduces the chance of the algorithm terminating at a local maximum and not reaching the true MLEs.

Options for `MARSSmcinit()` may be set using `MLEobj$control`, as follows:

`MLEobj$control$numInits` Number of random initial value draws.

`MLEobj$control$numInitSteps` Maximum number of EM iterations for each random initial value draw.

`MLEobj$control$boundsInits` Length 6 list. Each component is a length 2 vector of bounds on the uniform distributions from which initial values will be drawn. See Examples.

The default values for these are given in `MARSSsettings.R` and listed in [MARSS](#).

Value

A list with 8 matrices `Z`, `A`, `R`, `B`, `U`, `Q`, `x0`, `V0`, specifying initial values for parameters for iteration 1 of the EM algorithm.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

`eli(dot)holmes(at)noaa(dot)gov`, `eric(dot)ward(at)noaa(dot)gov`

References

The user manual: Holmes, E. E. and E. J. Ward (2010) Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112 type `show.doc(MARSS, manual)` to see.

See Also

[MARSSkem](#) [marssMLE](#) [MARSS](#)

Examples

```
## Not run:
dat = t(harborSeal)
dat = dat[2:nrow(dat),]
fit1=MARSS(dat, control=list(MCInit=TRUE))
fit1
#Show the inits that were used
fit1$start
#Try fewer initial start locations but more iterations
fit2=MARSS(dat, control=list(MCInit=TRUE, numInits=10, numInitSteps = 100))
```

```

fit2
#Show the inits that were used
fit2$start
#ignore the values for Z,B, and V0; those parameters are fixed

## End(Not run)

```

marssMLE

Maximum Likelihood MARSS Estimation Object

Description

An object in the [MARSS-package](#) that has all the elements needed for maximum-likelihood estimation of multivariate autoregressive state-space model: the data, model, estimation methods, and any control options needed for the method. If the model has been fit and parameters estimated, the object will also have the MLE parameters. Other functions add other elements to the marssMLE object, such as CIs, s.e.'s, AICb, and hessian.

Usage

```
is.marssMLE(MLEobj)
```

Arguments

MLEobj An object of class [marssMLE](#). See Details.

Details

The `is.marssMLE()` function checks components `model`, `start` and `control`, which must be present for estimation by functions e.g. [MARSSkem](#). Components returned from estimation must include at least `method`, `par`, `kf`, `numIter`, `convergence` and `logLik`. Additional components (e.g. AIC) may be returned, as described in function help files.

`model` MARSS model specification (an object of class [marssm](#)).

`start` List with 7 matrices A, R, B, U, Q, x0, V0, specifying initial values for parameters to be used (if needed) by the maximization algorithm.

B Initial value(s) for B matrix (m x m).

U Initial value(s) for U matrix (m x 1).

Q Initial value(s) for Q variance-covariance matrix (m x m).

A Initial value(s) for A matrix (n x 1).

R Initial value(s) for R variance-covariance matrix (n x n).

x0 Initial value(s) for initial state vector (m x 1).

V0 Initial variance(s) for initial state variance (m x m).

control A list specifying estimation options. The following options are needed by `MARSSkem`. Other control options can be set if needed for other estimation methods, e.g. the control options listed for `optim` for use with `MARSSoptim`. The default values for control options are set in `alldefaults[[method]]` which is specified in `MARSSsettings.R`.

minit The minimum number of iterations to do in the maximization routine (if needed by method).

maxit Maximum number of iterations to be used in the maximization routine (if needed by method).

abstol Optional convergence tolerance for the maximization routine.

iter.V0 The value of V0 to be used in place of 0 when x0 is treated as fixed and V0=0. See manual for discussion of initial state variance.

trace A positive integer. If not 0, a record will be created during maximization iterations (what's recorded depends on method of maximization).

MCInit If TRUE, do a Monte Carlo search of the initial condition space.

numInits Number of random initial value draws if MCInit=TRUE (ignored otherwise).

numInitSteps Number of EM iterations for each random initial value draw if MCInit=TRUE (ignored otherwise).

boundsInits Bounds on the uniform distributions from which initial values will be drawn if MCInit=TRUE (ignored otherwise).

silent Suppresses printing of progress bar, error messages and convergence information.

silent Suppresses printing of progress bar, error messages and convergence information.

method A string specifying the estimation method. MARSS 1.0 allows "kem" for Kalman-EM and "BFGS" for quasi-Newton.

par A list with 8 matrices of estimated parameter values Z, A, R, B, U, Q, x0, V0.

kf A list containing Kalman filter/smoothing output.

numIter Number of iterations which were required for convergence.

convergence Convergence status and errors. 0 means converged successfully. Anything else means an error or warning.

logLik Log-likelihood.

Value

TRUE if no problems; otherwise a message describing the problems.

Author(s)

Kellie Wills, NOAA, Seattle, USA. [kellie\(dot\)wills\(at\)noaa\(dot\)gov](mailto:kellie(dot)wills(at)noaa(dot)gov)

See Also

[marssm MARSSkem](#)

marssMLE-class *Class "marssMLE"*

Description

marssMLE objects specify maximum-likelihood estimation of multivariate autoregressive state-space models in the package [MARSS-package](#).

Methods

print signature(x = "marssMLE"): ...
summary signature(object = "marssMLE"): ...

Author(s)

Kellie Wills, NOAA, Seattle, USA. [kellie\(dot\)wills\(at\)noaa\(dot\)gov](mailto:kellie(dot)wills(at)noaa(dot)gov)

References

~put references to the literature/web site here ~

MARSSoptim *Parameter estimation for MARSS models using optim*

Description

Parameter estimation for MARSS models using R's [optim](#) function. This allows access to R's quasi-Newton algorithms available via the [optim](#) function. The `MARSSoptim` is called when [MARSS](#) is called with `method="BFGS"`. Only diagonal Q and R matrices are allowed, if they are estimated. This is a base function in the [MARSS-package](#). `neglogLik` is a helper function for `MARSSoptim` that returns the negative log-likelihood given a vector of the estimated parameters and a `marssMLE` object.

Usage

```
MARSSoptim(MLEobj)
neglogLik(x, MLEobj)
```

Arguments

`MLEobj` An object of class `marssMLE`.
`x` An vector of the estimated parameters as output by [MARSSvectorizeparam](#).

Details

Objects of class `marssMLE` may be built from scratch but are easier to construct using `MARSS` with `MARSS(..., fit=FALSE, method="BFGS")`.

Options for `optim` are passed in using `MLEobj$control`. See `optim` for a list of that function's control options. If `lower` and `upper` for `optim` need to be passed in, they should be passed in as part of `control` as `control$lower` and `control$upper`. Additional `control` arguments affect printing and initial conditions.

`MLEobj$control$iter.V0` This is the value to which `V0` will be set during the maximization algorithm (must not be equal to 0). See Details.

`MLEobj$control$MCInit` If TRUE, Monte Carlo initialization will be performed by `MARSSmcinit`.

`MLEobj$control$numInits` Number of random initial value draws to be used with `MARSSmcinit`. Ignored if `control$MCInit=FALSE`.

`MLEobj$control$numInitSteps` Maximum number of EM iterations for each random initial value draw to be used with `MARSSmcinit`. Ignored if `control$MCInit=FALSE`.

`MLEobj$control$boundsInits` Length 6 list. Each component is a length 2 vector of bounds on the uniform distributions from which initial values will be drawn to be used with `MARSSmcinit()`. Ignored if `control$MCInit=FALSE`. See Examples.

`MLEobj$control$silent` Suppresses printing of progress bars, error messages, warnings and convergence information.

Value

The `marssMLE` object which was passed in, with additional components:

<code>method</code>	String "BFGS".
<code>kf</code>	Kalman filter output.
<code>iter.record</code>	If <code>MLEobj\$control\$trace = TRUE</code> , then this is the <code>\$message</code> value from <code>optim</code> .
<code>numIter</code>	Number of iterations needed for convergence.
<code>convergence</code>	Did estimation converge successfully? convergence=0 Converged in less than <code>MLEobj\$control\$maxit</code> iterations and no evidence of degenerate solution. convergence=1 Maximum number of iterations <code>MLEobj\$control\$maxit</code> was reached before <code>MLEobj\$control\$abstol</code> condition was satisfied. convergence=10 Some of the variance elements appear to be degenerate. T convergence=52 The algorithm was abandoned due to errors from the "L-BFGS-B" method. convergence=53 The algorithm was abandoned due to numerical errors in the likelihood calculation from <code>MARSSkf</code> . This happens frequently with "BFGS" and can sometimes be helped with a better initial condition. Try using the Kalman-EM algorithm first (<code>method="kem"</code>), and then using the parameter estimates from that to as initial conditions for <code>method="BFGS"</code> .
<code>logLik</code>	Log-likelihood.

<code>states</code>	State estimates from the Kalman filter.
<code>states.se</code>	Confidence intervals based on state standard errors, see caption of Fig 6.3 (p. 337) Shumway & Stoffer.
<code>errors</code>	Any error messages.

Discussion

The function only returns parameter estimates. To compute CIs, use [MARSSparamCIs](#) but if you use parametric or non-parametric bootstrapping with this function, it will use the Kalman-EM algorithm to compute the bootstrap parameter estimates! The quasi-Newton estimates are too fragile for the bootstrap routine since one often needs to search to find a set of initial conditions that work (i.e. don't lead to numerical errors).

Estimates from `MARSSoptim` (which come from `optim`) should be checked against estimates from Kalman-EM algorithm. If the quasi-Newton algorithm works, it will tend to find parameters with higher likelihood faster than the Kalman-EM algorithm. However, the MARSS likelihood surface can be multimodal with sharp peaks at degenerate solutions where a Q or R diagonal element equals 0. The quasi-Newton algorithm tends to find and gets stuck on these peaks even when they are not the maximum. Neither an initial conditions search nor starting near the known maximum (or from the parameters estimates after the Kalman-EM algorithm) will necessarily solve this problem. Thus it is wise to check against Kalman-EM estimates to ensure that the BFGS estimates are close to the MLE estimates.

Note this is mainly a problem if the time series are short or very gappy. If the time series are long, then the likelihood surface should be nice with a single interior peak. In this case, the quasi-Newton algorithm works well but it can still be sensitive (and slow) if not started with a good initial condition. Thus starting it with the estimates from the Kalman-EM algorithm is often desirable.

The V0 matrix is set to a diffuse prior if `x0` is estimated (in which case V0 must be zero to treat `x0` as fixed but unknown). V0 is reset to zero when the final likelihood and state estimates are computed via [MARSSkf](#). See discussion in the manual and [MARSSkem](#). `MARSSoptim` only allows diagonal Q and R matrices, if they are estimated.

Author(s)

Eli Holmes, NOAA, Seattle, USA.
eli(dot)holmes(at)noaa(dot)gov

See Also

[MARSS](#) [MARSSkem](#) [marssMLE](#) [optim](#)

Examples

```
dat = t(harborSealWA)
dat = dat[2:nrow(dat),] #remove the year row
#fit a model with 1 hidden state where obs errors are iid
bfgsfit = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1))),
  method="BFGS")

#fit a model with Kalman-EM and then use that fit as the start for BFGS
```

```
kemfit = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1))))
bfgsfit = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1))),
  inits=kemfit$par, method="BFGS")
```

 MARSSparamCIs

Confidence Intervals for MARSS Parameters

Description

Computes confidence intervals for the maximum-likelihood estimates of MARSS model parameters. This is a base function in the [MARSS-package](#).

Usage

```
MARSSparamCIs(MLEobj, method = "hessian", alpha = 0.05, nboot=1000)
```

Arguments

MLEobj	An object of class <code>marssMLE</code> . Must have a <code>\$par</code> element containing the MLE parameter estimates.
method	Method for calculating the standard errors: "hessian", "parametric", and "innovations" implemented currently.
alpha	alpha level for the 1-alpha confidence intervals.
nboot	Number of bootstraps to use for "parametric" and "innovations" methods.

Details

Approximate confidence intervals (CIs) on the model parameters may be calculated from the Hessian matrix (the matrix of partial 2nd derivatives of the parameter estimates) or parametric or non-parametric (innovations) bootstrapping using `nboot` bootstraps. The Hessian CIs are based on the asymptotic normality of MLE parameters under a large-sample approximation. Bootstrap estimates of parameter bias are reported if method "parametric" or "innovations" is specified.

Value

MARSSparamCIs returns the `marssMLE` object passed in, with additional components `par.se`, `par.upCI`, `par.lowCI`, `par.CI.alpha`, `par.CI.method`, `par.CI.nboot` and `par.bias` (if method is "parametric" or "innovations").

Author(s)

Eli Holmes, NOAA, Seattle, USA.

`eli(dot)holmes(at)noaa(dot)gov`

References

Holmes, E. E. and E. J. Ward. 2010. Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112; this is the user manual accessed via `show.doc(MARSS, manual)`

See Also

[MARSSboot](#) [MARSSinnovationsboot](#) [MARSShessian](#)

Examples

```
dat = t(harborSealWA)
dat = dat[2:nrow(dat),]
kem = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1)),
  R="diagonal and unequal"))
kem.with.CIs.from.hessian = MARSSparamCIs(kem)
kem.with.CIs.from.hessian
```

MARSSsimulate

Simulate Data from a MARSS Model and Parameter Estimates

Description

Generates simulated data from a MARSS model with specified parameter estimates. This is a base function in the [MARSS-package](#).

Usage

```
MARSSsimulate(parList, tSteps = 100, nsim = 1, silent = TRUE,
  miss.loc = NULL, miss.value = NULL)
```

Arguments

<code>parList</code>	A list of parameter matrices structured like the <code>\$par</code> element of an object of class marssMLE .
<code>tSteps</code>	Number of time steps in each simulation.
<code>nsim</code>	Number of simulated data sets to generate.
<code>silent</code>	Suppresses progress bar.
<code>miss.loc</code>	Optional matrix specifying where to put missing values. See Details.
<code>miss.value</code>	Code representing missing values in <code>miss.matrix</code> . See Details.

Details

Argument `miss.loc` is an array of dimensions $n \times \text{tSteps} \times \text{nsim}$, specifying where to put missing values in the simulated data. Locations where missing data appear are specified using the `miss.value`; non-missing values can be specified by any other numeric value. If the locations of the missing values are the same for all simulations, `miss.loc` can be a matrix of $\text{dim}=\text{c}(n, \text{tSteps})$ (the original data for example). If `miss.loc` is passed in, `miss.value` must be specified. The default is that there are no missing values. If one's data has missing values in it and one want to replicate those locations in the simulated data, `miss.loc` can simply be set to the original data (see examples).

Value

<code>sim.states</code>	Array (dim $m \times \text{tSteps} \times \text{nsim}$) of state processes simulated from parameter estimates.
<code>sim.data</code>	Array (dim $n \times \text{tSteps} \times \text{nsim}$) of data simulated from parameter estimates.
<code>par</code>	The list of parameter matrices from which the data were simulated.
<code>miss.loc</code>	Matrix identifying where missing values are located.
<code>tSteps</code>	Number of time steps in each simulation.
<code>nsim</code>	Number of simulated data sets generated.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.
 eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

See Also

[marssm](#) [marssMLE](#) [MARSSboot](#)

Examples

```
#do a parametric bootstrap.
#Same length as original data and same location of missing data
d = harborSeal[,2:ncol(harborSeal)]
dat = t(d)
MLEobj = MARSS(dat)
sim.obj = MARSSsimulate(parList=MLEobj$par, tSteps=dim(d)[1], nsim=10)
```

MARSSvectorizeparam

Vector to Parameter Matrix Conversion

Description

Converts `MLEobj$par` to a vector of the estimated parameter elements and vice versa. This is a utility function in the [MARSS-package](#).

Usage

```
MARSSvectorizeparam(MLEobj, parvec = NA)
```

Arguments

MLEobj	An object of class <code>marssMLE</code> .
parvec	NA or a vector. See Value.

Details

Utility function to generate parameter vectors for optimization functions, and to set `MLEobj$par` using a vector of parameter values (only the estimated values).

Value

If `parvec=NA`, a vector of estimated parameters. Otherwise, a `marssMLE` object with `$par` set by `parvec`.

Author(s)

Eli Holmes and Kellie Wills, NOAA, Seattle, USA.
eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

See Also

[marssMLE](#)

Examples

```
dat = t(harborSealWA)
dat = dat[2:nrow(dat), ]
kem = MARSS(dat)
paramvec = MARSSvectorizeparam(kem)
```

plankton

Plankton Data Sets

Description

Example data sets for use in MARSS vignettes for the [MARSS-package](#). The "lakeWAp plankton" plankton counts have been standardized to a mean of zero and variance of 1 (Z-score transformation). The second column in "lakeWAp plankton" is simply a index for the count. Columns 3 and 4 are month and month^2 but they have also been Z-score transformed. Since MARSS functions require time to be across columns, these data matrices must be transposed before passing into MARSS functions.

Usage

```
data(ivesDataLP)
data(lakeWAp plankton)
```

Format

The data are provided as a matrix with time running down the rows.

Source

- ivesDataLP Ives, A. R. Dennis, B. Cottingham, K. L. Carpenter, S. R. (2003) Estimating community stability and ecological interactions from time-series data. *Ecological Monographs*, 73, 301-330.
- lakeWAp plankton Hampton, S. E. Scheuerell, M. D. Schindler, D. E. (2006) Coalescence in the Lake Washington story: Interaction strengths in a planktonic food web. *Limnology and Oceanography*, 51, 2042-2051.

Examples

```
str(ivesDataLP)
str(lakeWAp plankton)
```

popWrap

Wrapper Objects

Description

Wrapper objects are used by the function [MARSS](#) in the package [MARSS-package](#). `popWrap` creates a wrapper object containing specifications and options for estimation of a multivariate autoregressive state-space model.

Usage

```
popWrap(y, allowed,
        inits=NULL,
        constraint=NULL,
        fixed=NULL, free=NULL,
        miss.value=NULL,
        control=NULL,
        method=NULL,
        silent=FALSE)
```

Arguments

<code>y</code>	A matrix (not dataframe), observations (rows) x time steps (columns), of data.
<code>allowed</code>	Allowed constraints. This is determined by the method used for parameter estimation. <code>allowed\$kem</code> (specified in <code>MARSSsettings</code>) is the set of allowed constraints for the Kalman-EM algorithm under MARSS 1.0. <code>allowed\$BFGS</code> are the allowable constraints for BFGS.
<code>inits</code>	List with up to 8 matrices Z, A, R, B, U, Q, x0, V0, specifying initial values for parameters to use in iterative ML estimation algorithms, such as Kalman-EM and quasi-Newton methods. These are ignored if the specified parameter is not being estimated. If not passed in by the user, MARSS creates generic inits using MARSSinits . <ul style="list-style-type: none"> B Initial value(s) for B parameter (length 1 or m x m). If length 1, <code>inits\$B</code> constructed as <code>diag(value, m)</code>. U Initial value(s) for U parameter (length 1 or m x 1). If length 1, <code>inits\$U</code> constructed as <code>matrix(value, nrow=m, ncol=1)</code>. Q Initial value(s) for Q parameter (length 1 or m x m). If length 1, <code>inits\$Q</code> constructed as <code>diag(value, m)</code>. Z Initial value(s) for Z parameter (n x m). Ignored in MARSS 1.0; included for MARSS 2.0. A Initial value(s) for A parameter (length 1 or n x 1). If length 1, <code>inits\$A</code> constructed as <code>matrix(value, nrow=n, ncol=1)</code>. R Initial value(s) for R parameter (length 1 or n x n). If length 1, <code>inits\$R</code> constructed as <code>diag(value, n)</code>. x0 Initial value(s) for x0 parameter (length 1 or m x 1). If length 1, <code>inits\$x0</code> constructed as <code>matrix(value, nrow=m, ncol=1)</code>. V0 Initial variance(s) for hidden states (length 1 or m x m). Ignored in MARSS 1.0; included for forward compatibility.
<code>constraint</code>	Model specification using parameter constraint descriptions (recommended). See MARSS for details.
<code>fixed</code>	Optional model specification using matrices of fixed and free parameters. See manual for details.
<code>free</code>	Optional model specification using matrices of fixed and free parameters. See manual for details.
<code>miss.value</code>	How are missing values represented in the data?
<code>method</code>	The method used for estimation. This is needed for setting default values for <code>control</code> .
<code>control</code>	Control options for maximization algorithms. <ul style="list-style-type: none"> <code>minit</code> Minimum number of EM iterations. <code>maxit</code> Maximum number of EM iterations. <code>abstol</code> Optional tolerance for log-likelihood change. If log-likelihood changes less than this amount relative to the previous iteration, the algorithm exits. <code>iter.V0</code> This is the value to which V0 will be set during the algorithm iterations ($\neq 0$). See MARSSkem.

<code>trace</code>	Positive integer. If not 0, a record will be created of each variable over the maximization iterations. The information recorded depends on the maximization method.
<code>safe</code>	If TRUE, <code>MARSSkem</code> will rerun <code>MARSSkf</code> after each individual parameter update rather than only after all parameters are updated.
<code>MCInit</code>	Use Monte Carlo initialization? See discussion in <code>MARSSkem</code> and <code>MARSSmcinit</code> .
<code>numInits</code>	Number of random initial value draws.
<code>numInitSteps</code>	Number of EM iterations for each random initial value draw.
<code>boundsInits</code>	Bounds on the uniform distributions from which initial values will be drawn. Note that bounds for the covariance matrices Q and R, which require positive values, are specified in logs.
<code>silent</code>	Suppresses printing of progress bars, error messages, warnings and convergence information.

Details

Wrapper functions e.g. `MARSS` call `popWrap()` to create a 'popWrap' object, then `is.marssm` to coerce this object to class 'marssm' for the estimation function. The `popWrap()` function calls `checkPopWrap` to check user inputs.

If arguments `inits`, `constraint`, or `control` are not provided by the user, they will be set by the `alldefaults[[method]]` object specified in `MARSSsettings`. Argument `constraint` is a convenient way to specify model structure for certain common cases. If `fixed` is included, it provides matrices for some parameters, these will override any constraints for those parameters. See `marssm` or the manual for instructions on how to specify fixed and free matrices.

Value

An object of class 'popWrap'.

<code>data</code>	Data supplied by user.
<code>m</code>	Number of hidden state trajectories.
<code>constraint</code>	A list with up to 8 elements Z, A, R, B, U, Q, x0, V0 (unless some of these are specified in "fixed"). See <code>MARSS</code> for details on what values are allowed.
<code>fixed</code>	A list with (up to) 8 matrices Z, A, R, B, U, Q, x0, V0.
<code>free</code>	A list with (up to) 8 matrices Z, A, R, B, U, Q, x0, V0.
<code>inits</code>	A list specifying initial values for parameters to be used at iteration 1 in iterative maximum-likelihood algorithms.
<code>miss.value</code>	Specifies missing value representation.
<code>method</code>	The method used for estimation.
<code>control</code>	See Arguments.

Author(s)

Kellie Wills, NOAA, Seattle, USA.

kellie(dot)wills(at)noaa(dot)gov

See Also

[MARSS marssm checkPopWrap as.marssm](#)

Examples

```
dat = t(harborSeal)
dat = dat[2:nrow(dat),]
aa = MARSS:::allowed$kem
wrapperObj = popWrap(dat, allowed=aa, method="kem")
```

popWrap-class *Class "popWrap"*

Description

popWrap objects are wrapper object containing specifications and options for estimation of multivariate autoregressive state-space models in the package [MARSS-package](#).

Author(s)

Kellie Wills, NOAA, Seattle, USA. [kellie\(dot\)wills\(at\)noaa\(dot\)gov](mailto:kellie(dot)wills(at)noaa(dot)gov)

show.doc *Documentation Utility*

Description

Utility in the [MARSS-package](#) to open documentation for R packages. This is used to open files and show the directory of the doc directory of a package. This is similar to but a bit more flexible than the R utility [RShowDoc](#).

Usage

```
show.doc(pkg, filename, dir="doc")
```

Arguments

pkg	The name of an R package. Need not be in quotes.
filename	A file name or “manual”, “index” or “dir”. Need not be in quotes. Using “manual”, “index”, and “dir” has special behavior (no quotes required). “manual” opens (tries to open) the file manual.pdf. “index” tries to open the file index.html. “dir” lists the files in the directory (specified by the dir argument).
dir	The package subdirectory in which the file is located. Typically documentation is in the doc subdirectory and this is the default. However, other package directories can be specified. Use dir="." to see the directories available.

Author(s)

Eli Holmes eli(dot)holmes(at)noaa(dot)gov

See Also

See Also [RShowDoc](#)

Examples

```
## Not run:
#MARSS is used as the package in these examples, but you can replace
#it with any package name you have loaded into R. show.doc is not
#MARSS specific

#list the files in the "doc" subdirectory
show.doc(MARSS, dir)
#show the file manual.pdf in the "doc" subdirectory
show.doc(MARSS, manual)
#show the index.html file in the "doc" subdirectory
show.doc(MARSS, index)
#Open the file Case_study_2.R in the "doc" subdirectory
show.doc(MARSS, Case_study_2.R)
#show the subdirectories in the package directory
show.doc(MARSS, dir, dir=".")

## End(Not run)
```

stdInnov

Standardized Innovations

Description

Standardizes Kalman filter innovations. This is a helper function called by [MARSSinnovationsboot](#) in the [MARSS-package](#).

Usage

```
stdInnov(SIGMA, INNOV)
```

Arguments

SIGMA	n x n x T array of Kalman filter innovations variances. This is output from MARSSkf .
INNOV	n x T matrix of Kalman filter innovations. This is output from MARSSkf .

Details

n = number of observation (y) time series. T = number of time steps in the time series.

Value

n x T matrix of standardized innovations.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

References

Stoffer, D. S., and K. D. Wall. 1991. Bootstrapping state-space models: Gaussian maximum likelihood estimation and the Kalman filter. *Journal of the American Statistical Association* 86:1024-1033.

See Also

[MARSSboot](#) [MARSSkf](#) [MARSSinnovationsboot](#)

Examples

```
## Not run:  
std.innovations = stdInnov(kfList$Sigma, kfList$Innov)  
  
## End(Not run)
```

Index

*Topic **classes**

marssm-class, 36
marssMLE-class, 40
popWrap-class, 50

*Topic **datasets**

graywhales, 8
harborSeal, 9
loggerhead, 11
plankton, 46

*Topic **hplot**

CSEGriskfigure, 4
CSEGtmfigure, 6

*Topic **package**

MARSS-package, 2

as.design (*is.blockdiag*), 10
as.marssm, 3, 4, 50
as.marssm (*marssm*), 34

checkPopWrap, 3, 49, 50
CSEGriskfigure, 4, 7
CSEGtmfigure, 5, 6

fdHess, 23
find.degenerate, 7, 29, 30

graywhales, 8
grouse (*graywhales*), 8

harborSeal, 9
harborSealnomiss (*harborSeal*), 9
harborSealWA (*harborSeal*), 9

Imat (*is.blockdiag*), 10
is.blockdiag, 10
is.bloquequaltri (*is.blockdiag*), 10
is.blockunconst (*is.blockdiag*), 10
is.design (*is.blockdiag*), 10
is.diagonal (*is.blockdiag*), 10
is.equaltri (*is.blockdiag*), 10

is.fixed (*is.blockdiag*), 10
is.identity (*is.blockdiag*), 10
is.marssm, 49
is.marssm (*marssm*), 34
is.marssMLE (*marssMLE*), 38
is.wholenumber (*is.blockdiag*), 10
isleRoyal (*graywhales*), 8
ivesDataLP (*plankton*), 46

lakeWAplankton (*plankton*), 46
loggerhead, 11
loggerheadNoisy (*loggerhead*), 11

makediag (*is.blockdiag*), 10
MARSS, 2, 3, 7, 12, 26, 27, 33, 37, 40–42, 47–50
MARSS-package, 3, 4, 6–12, 14, 16, 17, 19, 20, 22–24, 26, 27, 31, 34, 36, 38, 40, 43–47, 50, 51

MARSS-package, 2
MARSSaic, 3, 17, 22
MARSSapplynames, 19
MARSSboot, 3, 5, 19, 20, 27, 44, 45, 52
MARSScheckdims, 22
MARSScheckpar (*MARSScheckdims*), 22
MARSShessian, 23, 44
MARSSinits, 24, 48
MARSSinnovationsboot, 26, 44, 51, 52
MARSSkem, 3, 7, 8, 15, 16, 23–26, 27, 31–33, 36–39, 42, 48, 49
MARSSkemcheck, 31
MARSSkf, 3, 7, 15, 28–30, 31, 41, 42, 49, 51, 52

marssm, 12, 14–16, 19–22, 24, 26, 31, 33, 34, 38, 39, 45, 49, 50
marssm-class, 36
MARSSmcinit, 28, 30, 36, 41, 49
marssMLE, 3, 5, 12, 16, 18–20, 22–24, 26–28, 30, 32, 37, 38, 38, 40–46
marssMLE-class, 40

MARSSoptim, 2, 3, 15, 16, 24–26, 29, 30, 36,
39, 40
MARSSparamCIs, 3, 5, 24, 27, 42, 43
MARSSsettings, 14
MARSSsettings (MARSS), 12
MARSSsimulate, 3, 44
MARSSvectorizeparam, 40, 45

neglogLik (MARSSoptim), 40
nlme, 23

optim, 3, 39–42

plankton, 46
popWrap, 3, 4, 34, 35, 47
popWrap-class, 50
prairiechicken (graywhales), 8
print, marssm-method
 (marssm-class), 36
print, marssMLE-method
 (marssMLE-class), 40

RShowDoc, 50, 51

show.doc, 50
stdInnov, 27, 51
summary, marssm-method
 (marssm-class), 36
summary, marssMLE-method
 (marssMLE-class), 40

takediag (is.blockdiag), 10

unvec (is.blockdiag), 10

vec (is.blockdiag), 10

wilddogs (graywhales), 8