

# Package ‘MARSS’

February 14, 2012

**Type** Package

**Title** Multivariate Autoregressive State-Space Modeling

**Version** 2.8

**Date** 2011-1-30

**Depends** MASS, mvtnorm, nlme, KFAS

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

**Maintainer** Eli Holmes <eli.holmes@noaa.gov>

**Description** The MARSS package provides maximum-likelihood parameter estimation for constrained and unconstrained linear multivariate autoregressive state-space (MARSS) models fit to multivariate time-series data. Fitting is primarily via an Expectation-Maximization (EM) algorithm, although fitting via the BFGS algorithm (using the `optim` function) is also provided. MARSS models are a class of dynamic linear model (DLM) and vector autoregressive model (VAR) model. Functions are provided for parametric and innovations bootstrapping, Kalman filtering and smoothing, bootstrap model selection criteria (AICb), confidence intervals via the hessian approximation and via bootstrapping and calculation of auxiliary residuals for detecting outliers and shocks. The user guide shows examples of using MARSS for parameter estimation for a variety of applications, model selection, dynamic factor analysis, outlier and shock detection, and addition of covariates. Type `RShowDoc("UserGuide", package="MARSS")` at the R command line to open the MARSS user guide. Online workshops (lecture material) at <http://faculty.washington.edu/eeholmes/workshops.shtml>

**License** GPL-2

**LazyData** yes

**BuildVignettes** yes

Repository CRAN

Date/Publication 2012-01-30 20:18:12

## R topics documented:

MARSS-package . . . . .	3
allowed . . . . .	4
checkPopWrap . . . . .	5
CSEGriskfigure . . . . .	6
CSEGtmfigure . . . . .	7
graywhales . . . . .	8
harborSeal . . . . .	10
is.blockdiag . . . . .	11
loggerhead . . . . .	12
MARSS . . . . .	13
MARSSaic . . . . .	18
MARSSapplynames . . . . .	20
MARSSboot . . . . .	21
MARSScheckdims . . . . .	23
MARSShatyt . . . . .	24
MARSShessian . . . . .	26
MARSSinits . . . . .	27
MARSSinnovationsboot . . . . .	28
MARSSkem . . . . .	30
MARSSkemcheck . . . . .	33
MARSSkf . . . . .	34
MARSSLLprofile . . . . .	37
marssm . . . . .	38
marssm-class . . . . .	41
MARSSmcinit . . . . .	41
marssMLE . . . . .	43
marssMLE-class . . . . .	45
MARSSoptim . . . . .	45
MARSSoptions . . . . .	48
MARSSparamCIs . . . . .	49
MARSSresids . . . . .	50
MARSSsimulate . . . . .	51
MARSSvectorizeparam . . . . .	53
plankton . . . . .	54
popWrap . . . . .	55
popWrap-class . . . . .	57
stdInnov . . . . .	58

Index

59

**Description**

The MARSS package fits constrained and unconstrained multivariate autoregressive time-series models to multivariate time series data. To open the user guide from the command line, type `RShowDoc("UserGuide", package="MARSS")`. To open an overview page with package information, type `RShowDoc("index", package="MARSS")`.

The MARSS model is

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

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

$$\mathbf{x}(1) \sim \text{MVN}(\mathbf{x}_0, \mathbf{V}_0) \text{ or } \mathbf{x}(0) \sim \text{MVN}(\mathbf{x}_0, \mathbf{V}_0)$$

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  $m \times 1$
- $\mathbf{Q}$  is  $m \times m$
- $\mathbf{A}$  is  $n \times 1$
- $\mathbf{R}$  is  $n \times n$
- $\mathbf{x}_0$  is  $m \times 1$
- $\mathbf{V}_0$  is  $m \times m$

The package functions estimate the model parameters using an 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**

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 EM algorithm.

[MARSSkf](#) and [MARSSkfas](#) Kalman filter and smoother.

- MARSSoptim** Estimates MARSS parameters using a quasi-Newton algorithm via [optim](#).
- MARSSaic** Calculates AICc, AIC, 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 user guide: 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 `RShowDoc("UserGuide", package="MARSS")` at the R command line to open the MARSS user guide.

Type `RShowDoc("index", package="MARSS")` to see all the package documentation, tutorials, and case study scripts.

---

allowed

*MARSS function defaults and allowed methods*

---

### Description

Defaults and allowed fitting methods for the [MARSS](#) function are specified in the file `MARSSsettings.R`. These are hidden thus to see them preface with `MARSS:::`.

### Details

`allowed` is a list with the allowed model structure shortcuts for each fitting method (used in [checkPopWrap](#). `allowed.methods` is a vector with the allowed method arguments for the [MARSS](#) function. `kem.methods` and `optim.methods` are vectors of method arguments that fall in each of these two categories; used by [MARSS](#). `alldefaults` is a list that specifies the defaults for [MARSS](#) arguments if they are not passed in; used by [checkPopWrap](#). `model.elem` and `model.elem.w.V0` specify the parameters names used in lists such as `par` lists.

---

checkPopWrap	<i>Check Arguments to popWrap()</i>
--------------	-------------------------------------

---

**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**

wrapperObj	An object of class <a href="#">popWrap</a> .
wrapper.el	Wrapper elements; generally set by <a href="#">MARSS</a>
allowed	Allowed model structures. This changes depending on the fitting method the user has specified (in <a href="#">MARSS</a> or in the <a href="#">marssMLE</a> object). Lists of for allowed for different fitting methods is set in <code>MARSSsettings.R</code> . Use <code>MARSS:::allowed</code> to view since <code>allowed</code> is hidden.
silent	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](mailto: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=MARSS:::allowed$kem, model=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, miss.value=NA)
```

**Arguments**

<code>data</code>	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 <a href="#">MARSS-package</a> functions.
<code>te</code>	Length of forecast period (positive integer)
<code>absolutethresh</code>	Is extinction threshold an absolute number? (T/F)
<code>threshold</code>	Extinction threshold either as an absolute number, if <code>absolutethresh=TRUE</code> , or as a fraction of current population count, if <code>absolutethresh=FALSE</code> .
<code>datalogged</code>	Are the data already logged? (T/F)
<code>silent</code>	Suppress printed output? (T/F)
<code>return.model</code>	Return state-space model as <a href="#">marssMLE</a> object? (T/F)
<code>CI.method</code>	Confidence interval method: "hessian", "parametric", "innovations", or "none". See <a href="#">MARSSparamCIs</a> .
<code>CI.sim</code>	Number of simulations for bootstrap confidence intervals (positive integer).
<code>miss.value</code>	missing value (default is NA)

**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 guide accessed via `RShowDoc("UserGuide", package="MARSS")`
- (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, datalogged = TRUE)
```

---

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

N	Time steps between the first and last population data point (positive integer)
u	Per time-step decline (-0.1 means a 10% decline per time step; 1 means a doubling per time step.)
s2p	Process variance (Q). (a positive number)
make.legend	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 user guide (`RShowDoc("UserGuide", package="MARSS")`) 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

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

---

graywhales

*Population Data Sets*

---

## Description

Example data sets for use in MARSS PVA vignettes in the [MARSS-package](#) user guide. 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)
data(kestrel)
data(okanaganRedds)
data(rockfish)
```

**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. UNLOGGED counts.
- 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. UNLOGGED counts.
- 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. UNLOGGED counts.
- 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. UNLOGGED counts.
- 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. UNLOGGED population counts.
- okanaganRedds A dataset of Chinook salmon redd (egg nest) surveys. This data comes from the Okanagan River in Washington state, a major tributary of the Columbia River (headwaters in British Columbia). UNLOGGED redd counts.
- rockfish LOGGED catch per unit effort data for Puget Sound total total rockfish (mix of species) from a series of different types of surveys.
- kestrel Three time series of American kestrel logged abundance from adjacent Canadian provinces along a longitudinal gradient (British Columbia, Alberta, Saskatchewan). Data have been collected annually, and corrected for changes in observer coverage and detectability. LOGGED abundance estimates.

**Examples**

```
str(graywhales)
str(grouse)
```

```
str(isleRoyal)
str(prairiechicken)
str(wilddogs)
str(kestrel)
str(okanaganRedds)
str(rockfish)
```

---

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, na.rm=FALSE)
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').
na.rm	How to treat NAs in the block diag test.
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](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. To open the user guide from the command line, type `RShowDoc("UserGuide", package="MARSS")`. To open an overview page with package information, type `RShowDoc("index", package="MARSS")`.

MARSS models take the form:

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

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

$$\mathbf{x}(1) \sim \text{MVN}(\mathbf{x0}, \mathbf{V0}) \text{ or } \mathbf{x}(0) \sim \text{MVN}(\mathbf{x0}, \mathbf{V0})$$

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,
      model=NULL,
      miss.value=NA,
      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.

`y`  
`inits` List with up to 7 matrices  $\mathbf{A}$ ,  $\mathbf{R}$ ,  $\mathbf{B}$ ,  $\mathbf{U}$ ,  $\mathbf{Q}$ ,  $\mathbf{x0}$ ,  $\mathbf{V0}$ , specifying initial values for parameters.

- $\mathbf{B}$  Initial value(s) for  $\mathbf{B}$  matrix (length 1 or  $m \times m$ )
- $\mathbf{U}$  Initial value(s) for  $\mathbf{U}$  matrix (length 1 or  $m \times 1$ )
- $\mathbf{Q}$  Initial value(s) for process error variance(s) (length 1 or  $m \times m$ )
- $\mathbf{A}$  Initial value(s) for observation bias (length 1 or  $n \times 1$ )
- $\mathbf{R}$  Initial value(s) for non-process (observation) error variance(s) (length 1 or  $n \times n$ )
- $\mathbf{x0}$  Initial value(s) for hidden state(s) at time=1 or 0 (length 1 or  $m \times 1$ )
- $\mathbf{V0}$  Initial variance(s) for hidden state(s) at time=1 or 0 (length 1 or  $m \times m$ )

`model` Model specification using parameter model descriptions. 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> (see <a href="#">MARSSkem</a> for more info) and <code>"BFGS"</code> (see <a href="#">MARSSoptim</a> for more info).
<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>	<p>Estimation options for the maximization algorithm. The typically used control options for <code>method="kem"</code> are below but see <a href="#">marssMLE</a> for the full list of control options. Note many of these are not allowed if <code>method="BFGS"</code>; see <a href="#">MARSSoptim</a> for the allowed control options for this method.</p> <ul style="list-style-type: none"> <li>• <code>min.iter</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)</li> <li>• <code>max.iter</code> Maximum number of iterations to be used in the maximization routine (if needed by method) (positive integer).</li> <li>• <code>kf.x0</code> Whether to set the prior at <code>t=0 ("x00")</code> or at <code>t=1 ("x10")</code>. The default is <code>"x00"</code>.</li> <li>• <code>min.iter.conv.test</code> Minimum iterations to run before testing convergence via the slope of the log parameter versus log iterations.</li> <li>• <code>conv.test.deltaT=9</code> Number of iterations to use for the testing convergence via the slope of the log parameter versus log iterations.</li> <li>• <code>conv.test.slope.tol</code> The slope of the log parameter versus log iteration to use as the cut-off for convergence. The default is 0.5 which is a bit high. For final analyses, this should be set lower.</li> <li>• <code>abstol</code> Optional convergence tolerance for the maximization routine. If set to something other than NULL, then the absolute tolerance will be used as the convergence test. Otherwise, the slope of the log of the parameters versus the log iteration is used as the test.</li> <li>• <code>allow.degen</code> Whether to try setting Q or R elements to zero if they appear to be going to zero.</li> <li>• <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). See <a href="#">MARSSkem</a> and <a href="#">MARSSoptim</a> for trace options and output details.</li> <li>• <code>MCInit</code> If TRUE, do a Monte Carlo search of the initial condition space. (T/F)</li> <li>• <code>numInits</code> Number of random initial value draws if <code>MCInit=TRUE</code> (ignored otherwise). (positive integer)</li> <li>• <code>numInitSteps</code> Number of EM iterations for each random initial value draw if <code>MCInit=TRUE</code> (ignored otherwise). (positive integer)</li> <li>• <code>boundsInits</code> Bounds on the uniform distributions from which initial values will be drawn if <code>MCInit=TRUE</code> (ignored otherwise). The Q (and analogously R) random matrices are created by a random draw from a wishart distribution where <code>df=bounds[1]</code> and <code>S=diag(bounds[2],m)</code>.</li> </ul>

- `silent` 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. The available estimation methods are maximum-likelihood via an 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 `model`. It allows the user to specify fixed values for matrices by passing in numeric matrices in the `model` list. Models with a combination of fixed values and shared estimated values can be specified using list matrices. See [marssm](#) or the user guide (reference and link below) for documentation and instructions on specifying list matrices for the model parameters.

Valid model structures for `method="kem"` are below. See the user guide (reference and link below) for details and type `allowed$kem` to see the allowed list specified in `MARSSsettings.R`.

- `Z` A text string, "identity", "unconstrained", "diagonal and unequal", "diagonal and equal", "equalvarcov", or "onestate", or a length `n` vector of factors specifying which of the `m` hidden state time series correspond to which of the `n` observation time series. May be specified as a `n x m` list matrix for general specification of both fixed and shared elements within the matrix. May also be specified as a numeric `n x m` matrix to use a custom fixed `Z`. "onestate" gives a `n x 1` matrix of 1s. "identity", "unconstrained", "diagonal and unequal", "diagonal and equal", and "equalvarcov" all specify `m x m` matrices.
- `B` "identity", "unconstrained", "diagonal and unequal", "diagonal and equal", "equalvarcov", "zero". Can be specified as a list matrix for general specification of both fixed and shared elements within the matrix. 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`, `A` and `x0` "unconstrained", "equal", "unequal" or "zero". May be specified as a `m x 1` (or `n x 1` for `A`) list matrix for general specification of both fixed and shared elements within the matrix. May also be specified as a numeric `m x 1` (or `n x 1`) matrix to use a custom fixed `U` (or `A`). NAs can be put in this matrix to allow some elements to be fixed and others (the NAs) to be estimated.
- `Q`, `R` and `V0` "identity", "unconstrained", "diagonal and unequal", "diagonal and equal", "equalvarcov", "zero". May be specified as a list matrix for general specification of both fixed and shared elements within the matrix. May also be specified as a numeric `m x m` matrix to use a custom fixed matrix. 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.
- `special A` option and comments The default setting for `A` is "scaling". This is used to treat `A` as an intercept where one `A` for each `X` (hidden state) is fixed at 0 and any other `Y`s associated with that `X` have an estimated `A` value. Care must be taken when specifying `A` so that the model is not under-constrained and unsolvable model; i.e. at least one `A` element per `X` state needs to be fixed.

Valid model structures for `method="BFGS"` are the same as for `method="kem"`. See [MARSSoptim](#) for the allowed options for this method.

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

- `inits = list(B=1, U=0, Q=0.05, A=0, R=0.05, x0=-99, V0=1)`
- `model = list(Z="identity", A="scaling", R="diagonal and equal", B="identity", U="unconstrained", Q="diagonal and unequal", x0="unconstrained")`
- `miss.value = NA`
- `control=list(minit=15, maxit=500, abstol=NULL, trace=0, safe=FALSE, allow.degen=TRUE, min.degen.iter=50, degen.lim=1.0e-04, MCinit=FALSE, numInits = 500, numInitSteps = 10, min.iter.conv.test=15, conv.test.deltaT=9, conv.test.slope.tol= 0.5, boundsInits=list(B=c(0,1), U=c(-1,1), Q = c(sqrt(0.1),0.1,0.1), Z=c(0,1), A=c(-1,1), R = c(sqrt(0.1),0.1,0.1) ) )`

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

The likelihood surface for MARSS models can be 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 <a href="#">marssm</a> ).
<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 <a href="#">MARSSkf</a> .
<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 <a href="#">MARSSkem</a> and <a href="#">MARSSoptim</a> .
<code>logLik</code>	Log-likelihood.
<code>AIC</code>	Akaike's Information Criterion.
<code>AICc</code>	Sample size corrected AIC.

**Author(s)**

Eli Holmes, Eric Ward and Kellie Wills, NOAA, Seattle, USA.  
 eli(dot)holmes(at)noaa(dot)gov, kellie(dot)wills(at)noaa(dot)gov

**References**

The user guide: 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 `RShowDoc("UserGuide", package="MARSS")` to open a copy.

Holmes, E. E. (2010). Derivation of the EM algorithm for constrained and unconstrained multivariate autoregressive state-space (MARSS) models. Link on the CRAN page under `EMDerivation.pdf`

**See Also**

[marssm](#) [marssMLE](#) [MARSSkem](#) [MARSSoptim](#) [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, model=list(Z=factor(c(1,1,1,1,1)),
  R="diagonal and equal"))
kemfit$model #This gives a description of the model
print(kemfit$model) # same as kemfit$model
summary(kemfit$model) #This shows the model structure

#fit the model using quasi-Newton algorithm
## Not run: #takes a long time
bfgsfit = MARSS(dat, model=list(Z=factor(c(1,1,1,1,1)),
  R="diagonal and unequal"), method="BFGS")

## End(Not run)

#add CIs to a marssMLE object
#default uses an estimated Hessian matrix
kem.with.hess.CIs = MARSSparamCIs(kemfit)
kem.with.hess.CIs #print with se's and CIs

#estimate CIs using a parametric bootstrap
tmp=kemfit
tmp$control$abstol=0.01
kem.with.boot.CIs = MARSSparamCIs(tmp,
  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
```

```

#fit a model with 5 correlated hidden states
# with one variance and one covariance using the delta logLik convergence test
# by setting abstol. This is fast, but does not actually test convergence (i.e. that
# log param value versus log iteration is flat
kemfit = MARSS(dat, model=list(Q="equalvarcov"), control=list(abstol=0.1))

#fit a model with 5 correlated hidden states
#abstol set rather huge and many of the Q's are not converged based on log-log test
kemfit = MARSS(dat, model=list(Q="unconstrained"), control=list(abstol=0.1))

#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, model=list(U=matrix(c("N","N","N","S","S"),5,1)))

#same model, but with fixed independent observation errors
#and the 4th and 5th x processes are forced to have a U=0
#Notice how a list matrix is used to combine fixed and estimated elements
#all parameters can be specified in this way using list matrices
kemfit = MARSS(dat, model=list(U=matrix(list("N","N","N",0,0),5,1),
  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, model=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)

## Not run:
#Compute bootstrap AIC for the model; this takes a long, long time
kemfit.with.AICb = MARSSaic(kemfit, output = "AICbp")
kemfit.with.AICb

## End(Not run)

```

**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 <a href="#">marssMLE</a> . 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"> <li>• nboot Number of bootstraps (positive integer)</li> <li>• return.logL.star Return the log-likelihoods for each bootstrap? (T/F)</li> <li>• silent Suppress printing of the progress bar during AIC bootstraps? (T/F)</li> </ul> |

**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 guide accessed via `RShowDoc("UserGuide", package="MARSS")`
- 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[2:3,]
kem = MARSS(dat, model=list(Z=factor(c(1,1)),
  R="diagonal and equal"))
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` and is not directly accessible. Use `MARSS:::MARSSapplynames()` to access.

## 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>

A.names	Names for MLEobj\$par\$A
R.names	Names for MLEobj\$par\$R
Q.names	Names for MLEobj\$par\$Q
B.names	Names for MLEobj\$par\$B
Z.names	Row and col names for MLEobj\$par\$Z
rows	Add row names?
cols	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](#)

---

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 <code>marssMLE</code> . Must have a <code>\$par</code> 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 <code>MLEobj\$control</code> are used by default. If supplied here, must contain all of the following: <code>max.iter</code> 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>allow.degen</code> Whether to try setting Q or R elements to zero if they appear to be going to zero.
silent	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 ( $n \times t \times nboot$ ) 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 guide accessed via RShowDoc("UserGuide", package="MARSS")
- 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**

```
#nboot is set low in these examples in order to run quickly
#normally nboot would be >1000 at least
dat = t(harborSealnomiss)
dat = dat[2:4,]
#normally one would not use the delta logLik test for convergence (abstol)
#but used here with big abstol and minit to speed up the example
kem = MARSS(dat, model=list(U="equal",Q="diagonal and equal"),
  control=list(minit=40,abstol=0.1))
hess.list = MARSSboot(kem, param.gen="hessian", nboot=5)
# (no missing values)
boot.list = MARSSboot(kem, output="all", sim="innovations", nboot=5)

# Bootstrap CIs for data with missing values
dat = t(harborSealWA)
dat = dat[2:4,]
kem = MARSS(dat, model=list(Q="diagonal and equal"),
  control=list(minit=40,abstol=0.1))
boot.list = MARSSboot(kem, output="all", sim="parametric", nboot=5)
```

---

 MARSScheckdims

---

 MARSS Utilities
 

---

**Description**

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

**Usage**

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

**Arguments**

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

---

MARSShatyt

*Compute Expected Value of Y, YY, and YX*

---

**Description**

Computes the expected value of random variables involving Y for the EM algorithm. This is a base function in the [MARSS-package](#).

**Usage**

```
MARSShatyt(y, parList, kfList, missing.matrix = NULL, miss.value= NULL)
```

**Arguments**

y	A matrix (not dataframe), sites (rows) x years (columns). See Details regarding handling of missing values.
parList	A list with 8 matrices Z, A, R, B, U, Q, x0, V0, specifying parameter values. An example is the par element in a <a href="#">marssMLE</a> object.
kfList	A list with the output from <a href="#">MARSSkf</a> .
missing.matrix	Optional matrix specifying which observations are missing. See Details.
miss.value	How are missing values represented in the data? Either miss.value or missing.matrix must be supplied. If both are supplied, then miss.value will be ignored with no warning(!).

## Details

For state space models, `MARSShatyt()` computes the expectations involving  $Y$ . If  $Y$  is completely observed, this entails simply replacing  $Y$  with the observed  $y$ . When  $Y$  is only partially observed, the expectation involves the conditional expectation of a multivariate normal.

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 NA or -99).

## 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>ytT</code>	Estimates $E[Y(t)   Y(1)=y(1)]$ ( $n \times T$ matrix).
<code>OtT</code>	Estimates $E[Y(t)t(Y(t)   Y(1)=y(1))]$ ( $n \times n \times T$ array).
<code>yxtT</code>	Estimates $E[Y(t)t(X(t)   Y(1)=y(1))]$ ( $n \times m \times T$ array).
<code>errors</code>	Any error messages due to ill-conditioned matrices.
<code>ok</code>	(T/F) Whether errors were generated.

## Author(s)

Eli Holmes, NOAA, Seattle, USA.

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

## References

Holmes, E. E. (2010) Derivation of the EM algorithm for constrained and unconstrained multivariate autoregressive state-space (MARSS) models. Technical report. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112

## See Also

[MARSS](#) [mar:ssm](#) [MARSSkem](#)

---

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

MLEobj            An object of class [marssMLE](#). This object must have a `$par` 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

```
dat = t(harborSeal)
dat = dat[2:4,]
MLEobj = MARSS(dat)
MLEobj.w.hessian = MARSShessian(MLEobj)
```

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 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,
                                Z=1, A=0, R=0.05, x0=-99, V0=5), method)
```

**Arguments**

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

**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` `V0` is never estimated, so this is never used.

### 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

<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. <a href="#">MARSSkem</a> or <a href="#">MARSS</a> . This algorithm may not be used if there are missing datapoints in the data.
<code>nboot</code>	Number of bootstraps to perform.
<code>minIndx</code>	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 <code>marssMLE</code> 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, model=list(U="equal",Q="diagonal and equal"))
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. The commonly used elements of control are follows (see [marssMLE](#)):

`min.iter` 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.

`max.iter` Maximum number of EM iterations.

`min.iter.conv.test` The minimum number of iterations before the log-log convergence test will be computed. If `max.iter` is set less than this, then convergence will not be computed (and the algorithm will just run for `max.iter` iterations).

`kf.x0` Whether to set the prior at  $t=0$  ("x00") or at  $t=1$  ("x10"). The default is "x00".

`conv.test.deltaT` The number of iterations to use in the log-log convergence test. This defaults to 9.

`abstol` Tolerance for log-likelihood change for the delta logLik convergence test. If log-likelihood changes less than this amount relative to the previous iteration, the EM algorithm exits. This is normally (default) set to NULL and the log-log convergence test is used instead.

`allow.degen` Whether to try setting Q or R elements to zero if they appear to be going to zero.

`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.

`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.

`MCInit` If TRUE, Monte Carlo initialization will be performed by [MARSSmccinit](#).

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

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

`boundsInits` Length 6 list. Each component is a length 2 vector of bounds on the uniform distributions (for B, U, and A) from which initial values will be drawn to be used with `MARSSmccinit()`. For R and Q, the 2 bounds specify the df and  $S(=\text{diag}(\text{bound}[2],m))$  for a wishart distribution. Ignored if `MCInit=FALSE`. See Examples.

`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 at each iteration.
<code>numIter</code>	Number of iterations needed for convergence.
<code>convergence</code>	Did estimation converge successfully? <b>convergence=0</b> Converged in less than <code>MLEobj\$control\$maxit</code> iterations and no evidence of degenerate solution using the log-log plot test. <b>convergence=1</b> Maximum number of iterations <code>MLEobj\$control\$maxit</code> was reached before <code>MLEobj\$control\$abstol</code> condition was satisfied. This value can only be output if <code>MLEobj\$control\$abstol</code> is passed in. If the default log-log convergence test is being used, <code>convergence=10</code> is returned when <code>maxit</code> is reached. <b>convergence=2</b> No convergence diagnostic were computed because there are problems with the MLE object. <b>convergence=3</b> No convergence diagnostic were computed because <code>MLEobj\$control\$maxit</code> was set to less than <code>control\$min.iter.conv.test</code> . This is not an error. <b>convergence=10</b> Some of the parameter estimates did not converge (based on the log-log plot test) before <code>MLEobj\$control\$maxit</code> was reached. This is not an error per se. Degenerate solutions are fine but the MARSS algorithm will not compute the proper likelihood for a degenerate solution. <b>convergence=11</b> Some of the parameter estimates did not converge (based on the log-log plot test) even though <code>MLEobj\$control\$abstol</code> was reached. This value can only be output if <code>MLEobj\$control\$abstol</code> is passed in. Try not setting <code>MLEobj\$control\$abstol</code> so that the log-log convergence test is used instead. If it takes too long, try making <code>MLEobj\$control\$maxit</code> smaller (like 500). <b>convergence=12</b> <code>MLEobj\$control\$abstol</code> was reached but the log-log plot test returned NAs. This is an odd error and you should set <code>control\$trace=TRUE</code> and look at the outputted <code>iter.record</code> to see what is wrong. This value can only be output if <code>MLEobj\$control\$abstol</code> is passed in.

**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. This is not an error per se. Most likely it means that your model is not very good for your data (too inflexible or too many parameters). Try setting `control$trace=1` to view a detailed error report.

**convergence=62** The algorithm was abandoned due to errors in the log-log convergence test. You should not get this error (it is included for debugging purposes to catch improper arguments passed into the log-log convergence test). If you get it, try passing in `control$abstol` to use delta logLik as the convergence test.

**convergence=72** Other convergence errors. This is included for debugging purposes to catch misc. errors. If you get it, try passing in `control$abstol` to use delta logLik as the convergence test.

<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/smoothing (`MARSSkf`) for hidden state estimation. The algorithm allows two options for the initial state conditions: fixed but unknown or a prior. In the first case, `x0` (whether at `t=0` or `t=1`) is treated as fixed but unknown (estimated); in this case, `fixed$V0=0` and `x0` is estimated. This is the default behavior. In the second case, the initial conditions are specified with a prior and `V0!=0`. In the later case, `x0` or `V0` may be estimated. MARSS will allow you to try to estimate both, but many researchers have noted that this is not robust so you should fix one or the other.

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).

## 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 Totronto, 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 guide: 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 RShowDoc("UserGuide", package="MARSS") to see.

Holmes, E. E. (2010). Derivation of the EM algorithm for constrained and unconstrained multivariate autoregressive state-space (MARSS) models. Link is on the CRAN page under EMDerivation.pdf.

## See Also

[MARSSmcmnit](#), [MARSSkf](#), [marssMLE](#), [MARSSoptim](#)

## Examples

```
dat = t(harborSeal)
dat = dat[2:nrow(dat),]
#you can use MARSS to construct a proper marssMLE object.
MLEobj = MARSS(dat, model=list(Q="diagonal and equal", U="equal"), fit=FALSE)
#Pass this MLEobj to MARSSkem to do the fit.
kemfit = MARSSkem(MLEobj)
```

---

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, method="kem", kf.x0=NULL)
```

## Arguments

modelObj	An object of class <a href="#">marssm</a> .
method	The method to be used. Currently only method="kem" is available.
kf.x0	Whether x0 is at t=0 (x00) or t=1 (x10)

**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/smoothen for MARSS models. This is a base function in the [MARSS-package](#). `MARSSkf()` is a native R implementation that allows `x0` to be at `t=0` or `t=1` (data starts at `t=1`). `MARSSkfas()` uses the Kalman filter-smoothen functions in the `KFAS` package. This requires that `x0` be at `t=1` (`kf.x0="x10"`), but the functions use more efficient and more stable algorithms. Exact diffuse priors are also allowed in the `KFAS` Kalman filter function.

**Usage**

```
MARSSkf(y, parList, missing.matrix = NULL, miss.value = NULL,
         init.state="x00", debugkf=FALSE)
MARSSkfas(y, parList, missing.matrix = NULL, miss.value= NULL, init.state="x10",
          debugkf=FALSE, diffuse=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 <a href="#">marssMLE</a> 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, <code>x0</code> , treated as $E(x)$ at time <code>t=0</code> ( <code>init.state="x00"</code> ) as in Shumway & Stoffer or $E(x)$ at <code>t=1</code> ( <code>init.state="x10"</code> ) as in Ghahramani et al.? Default is <code>init.state="x00"</code> .
<code>debugkf</code>	Return detailed error messages?
<code>diffuse</code>	Use an exact diffuse prior. See <code>KFAS</code> documentation. <code>V0</code> should be non-zero since the diffuse prior variance is $V0 \cdot \kappa$ , where $\kappa$ goes to infinity.

## 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 NA or -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 be treated in two different ways. One can treat it as `x00`, meaning  $E(x \text{ at } t=0 \mid y \text{ at } t=0)$ , and then compute `x10`, meaning  $E(x \text{ at } t=1 \mid y \text{ at } t=0)$ , from `x00`. Or one can simply treat the initial state as `x10`, meaning  $E(x \text{ at } t=1 \mid y \text{ at } t=0)$ . The approaches lead to the same parameter estimates, but the likelihood is written slightly differently in each case and you need your likelihood calculation to correspond to how the initial state is treated in your model (either `x00` or `x10`). The EM algorithm in the MARSS package ([MARSSkem](#)) follows Shumway and Stoffer's derivation and uses `x00`, while Ghahramani et al uses `x10`. The `init.state` argument specifies whether `x00` (`init.state="x00"`) or `x10` (`init.state="x10"`) is used. The default is `init.state="x00"`.

The Kalman filter/smoothing code has been altered to allow the user to specify partially deterministic models (some or all elements of the  $Q$  diagonal equal to 0) and partially perfect observation models (some or all elements of the  $R$  diagonal equal to 0). In addition, the code includes numerous checks to alert the user if matrices are becoming ill-conditioned and the algorithm unstable. In addition, the likelihood computation returns the exact likelihood when there are missing values (rather than the approximate likelihood typically used).

## Value

A list with the following components ( $m$  is the number of state processes). "V" elements are called "P" in Shumway and Stoffer.

<code>xtT</code>	State first moment $E[x(t) \mid y(1:T)]$ ( $m \times T$ matrix). Kalman smoother output.
<code>VtT</code>	State second moments $E[x(t)x(t)' \mid y(1:T)]$ ( $m \times m \times T$ array). Kalman smoother output. $P_{,t}^T$ in S&S.
<code>Vtt1T</code>	State lag-one second moments $E[x(t)x(t-1)' \mid y(1:T)]$ ( $m \times m \times T$ ). Kalman smoother output. $P_{,t,t-1}^T$ in S&S.
<code>x0T</code>	Initial state estimate $E[x(i) \mid y(1:T)]$ ( $m \times 1$ ). If <code>control\$kf.x0="x00"</code> , $i=0$ ; if <code>"x10"</code> , $i=1$ . Kalman smoother output.
<code>V0T</code>	Estimate of initial state covariance matrix $E[x(i)x(i)' \mid y(1:T)]$ ( $m \times m$ ). If <code>control\$kf.x0="x00"</code> , $i=0$ ; if <code>"x10"</code> , $i=1$ . Kalman smoother output. $P_{,0}^T$ in S&S.
<code>J</code>	( $m \times m \times T$ ) Kalman smoother output.
<code>J0</code>	$J$ at init time ( $t=0$ or $t=1$ ) ( $m \times m \times T$ ). Kalman smoother output.
<code>xtt</code>	$E[x(t) \mid y(1:t)]$ ( $m \times T$ ). Kalman filter output.

<code>x tt1</code>	$E[x(t)   y(1:t-1)]$ ( $m \times T$ ). Kalman filter output.
<code>V tt</code>	State second moment estimates, $E[x(t)x(t)'   y(1:t)]$ ( $n \times n \times T$ ). Kalman filter output. $P_t^t$ in S&S.
<code>V tt1</code>	State second moment estimates $E[x(t)x(t)'   y(1:t-1)]$ ( $m \times m \times T$ ). Kalman filter output. $P_t^{t-1}$ in S&S.
<code>K t</code>	Kalman gain ( $m \times m \times T$ ). Kalman filter output.
<code>Innov</code>	Innovations $y(t) - E[y(t)   Y(t-1)]$ ( $n \times T$ ). Kalman filter output.
<code>Sigma</code>	Innovations variances. Kalman filter output.
<code>logLik</code>	Log-likelihood computed from <code>mssm.params</code> and innovations.
<code>errors</code>	Any error messages.

**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**

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.

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

The user guide: 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 `RShowDoc("UserGuide", package="MARSS")` 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=NA)
```

---

MARSSLLprofile *Log-likelihood profiles for MARSS Parameters*

---

### Description

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

### Usage

```
MARSSLLprofile(MLEobj, param=NULL, x=NULL, LLlim=3, pstep=0.01,
               max.steps=20, plot=TRUE)
```

### Arguments

MLEobj	An object of class <a href="#">marssMLE</a> . Must have a \$par element containing the MLE parameter estimates.
param	A vector of parameter names. Must match those output from <a href="#">MARSSvectorizeparam</a> and output when a <a href="#">marssMLE</a> object is printed. If you leave this off (leave NULL), then profiles will be computed for all free variables.
x	An optional vector of parameters values at which to compute the log-likelihood. If x is not specified then LLlim must be. If you pass in x, then LLlim and pstep will be ignored. If you want a different x for each parameter, then pass in a list: x=list(U=c(...), Q=c(...)). Any left off parameters will use LLlim and pstep.
LLlim	If passed in, the LL profile will go from max(LL)-LLlim to max(LL) to max(LL)-LLlim. The range of parameter values will keep being stepped upward or downward to reach these limits.
pstep	The value of step=parameter(x(1))-parameter(x(2)) to use when stepping towards LLlim. If the parameters are from Q, R or V0, then step is on the log-scale: step=log(parameter(x(1)))-log(parameter(x(2))). If you want different step values for different parameters, pass this in as a list: pstep=list(U=0.01, Q=.1, R=.2). Any parameter names left off will use the default step of 0.01.
max.steps	Since it is not guaranteed that LLlim will be reached, max.steps is used to specify the maximum number of steps to do.
plot	Whether to produce a plot. If FALSE, then just the x values and LL values are returned.

### Details

Computes log-likelihood profiles for the free parameters specified in param for a fitted [marssMLE](#) object using the method MLEobj\$method. A red line is plotted at max.LL-1.92 (corresponding to the 1 degree of freedom chi-square value) If you want to change the control values used for computing the log-likelihoods (to say speed things up), then change the \$control element of MLEobj. If you have a lot of free parameters, then just call MARSSLLprofile with a few parameters at a time. Otherwise the function will try to plot all of them in one plot and you'll get a plot margin error.

**Value**

MARSSLLprofile returns a list of x-values and LL profile values for each parameter in param.

**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 guide accessed via RShowDoc("UserGuide", package="MARSS")

**See Also**

[MARSSparamCIs](#)

**Examples**

```
dat = t(harborSealnomiss)
dat = dat[2:3,]
kem = MARSS(dat, model=list(Z=factor(c(1,1)),
  R="diagonal and equal",Q=matrix(0.01)))
profLL = MARSSLLprofile(kem,c("R.1","U.1"),pstep=list(U=0.01,R=0.1))
```

---

marssm

*Model Objects*

---

**Description**

These are model objects and utility functions for model objects in the package [MARSS-package](#). Users would not normally work directly with these functions. `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 = NA)
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 <code>popWrap</code> .

## 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:

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

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

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

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

$$\mathbf{A} \quad n \times 1$$

$$\mathbf{R} \quad n \times n$$

$$\mathbf{x0} \quad m \times 1$$

$$\mathbf{V0} \quad m \times m$$

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 user guide (`RShowDoc("UserGuide", package="MARSS")`) for many examples of MARSS models and their specification.

**Value**

An object of class "marssm".

data	Data supplied by user.
fixed	A list with 8 matrices Z, A, R, B, U, Q, x0, V0.
free	A list with 8 matrices Z, A, R, B, U, Q, x0, V0.
M	An array of dim n x n x t (an n x 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.
miss.value	Specifies missing value representation. Default is NA

**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$Q=array(NA,dim=c(n,n)); diag(free$Q)=1:m
fixed$Q=array(0,dim=c(n,n)); diag(fixed$Q)=NA
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),]
#allowed is a hidden variable which specifies what model structures are allowed
wrapperObj = popWrap(dat, MARSS:::allowed$kem, 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

---

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 (for A, B, U, and Z). For R and Q, variance-covariance matrices are generated from a wishart distribution with  $df=bound[1]$  and  $S=diag(bound[2],m)$ . Note, random initial conditions are only used for parameters that are not fixed.

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 guide: 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 `RShowDoc("UserGuide", package="MARSS")` to see.

### See Also

[MARSSkem](#) [mar ssMLE](#) [MARSS](#)

### Examples

```
## Not run:
#Note doing a Monte-Carlo search takes a long, long time
dat = t(harborSeal)
dat = dat[2:nrow(dat),]
fit1=MARSS(dat, control=list(MCInit=TRUE))
fit1
#Show the inits that were used
fit1$start
#Try fewer initial start locations and different mean variance (0.1 instead of 1) for R and Q
fit2=MARSS(dat, control=list(MCInit=TRUE, numInits=10, numInitSteps = 10, boundsInits=list(Q=c(1,0.1),R=c(1,0.1)
fit2
#Show the inits that were used
fit2$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).

`kf.x0` Whether to set the prior at t=0 ("x00") or at t=1 ("x10"). The default is "x00".

`min.iter.conv.test` Minimum iterations to run before testing convergence via the slope of the log parameter versus log iterations.  
`conv.test.deltaT=9` Number of iterations to use for the testing convergence via the slope of the log parameter versus log iterations.  
`conv.test.slope.tol` The slope of the log parameter versus log iteration to use as the cut-off for convergence. The default is 0.5 which is a bit high. For final analyses, this should be set lower.  
`abstol` Optional convergence tolerance for the maximization routine. If set to something other than NULL, then the absolute tolerance will be used as the convergence test. Otherwise, the slope of the log of the parameters versus the log iteration is used as the test.  
`trace` A positive integer. If not 0, a record will be created during maximization iterations (what's recorded depends on method of maximization).  
`safe` Logical. If TRUE, then the Kalman filter is run after each update equation in the EM algorithm. This slows down the algorithm. The default is FALSE.  
`allow.degen` If TRUE, replace Q or R diagonal elements by 0 when they become very small.  
`min.degen.iter` Number of iterations before trying to set a diagonal element of Q or R to zero).  
`degen.lim` How small the Q or R diagonal element should be before attempting to replace it with zero.  
`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.  
`method` A string specifying the estimation method. MARSS allows "kem" for 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

**See Also**

[marssm MARSSkem](#)

---

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

---

**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

---

MARSSoptim	<i>Parameter estimation for MARSS models using optim</i>
------------	--

---

**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". 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. When possible, the Kalman filter and smoother functions (function [MARSSkfas](#)) from the KFAS R package are used but the user can force the Kalman filter and smoother functions based on Shumway and Stoffer (function [MARSSkf](#)) to be used by setting method="BFGSkf".

**Usage**

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

**Arguments**

MLEobj	An object of class <a href="#">marssMLE</a> .
x	An vector of the estimated parameters as output by <a href="#">MARSSvectorizeparam</a> .

## 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$control$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$control$kf.x0` The initial condition is at  $t=0$  if `kf.x0="x00"`. The initial condition is at  $t=1$  if `kf.x0="x10"`.

`MLEobj$control$diffuse` If `diffuse=TRUE`, a diffuse initial condition is used. `MLEobj$par$V0` is then the scaling function for the diffuse part of the prior. Thus the prior is  $V0 \cdot \kappa$  where  $\kappa \rightarrow \text{Inf}$ . Note that setting a diffuse prior does not change the correlation structure within the prior. If `diffuse=FALSE`, a non-diffuse prior is used and `MLEobj$par$V0` is the non-diffuse prior variance on the initial states. The the prior is  $V0$ .

`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? <b>convergence=0</b> Converged in less than <code>MLEobj\$control\$maxit</code> iterations and no evidence of degenerate solution. <b>convergence=1</b> Maximum number of iterations <code>MLEobj\$control\$maxit</code> was reached before <code>MLEobj\$control\$abstol</code> condition was satisfied. <b>convergence=10</b> Some of the variance elements appear to be degenerate. T <b>convergence=52</b> 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 `MARSSkf`. This happens frequently with "BFGS" and can sometimes be helped with a better initial condition. Try using the EM algorithm first (`method="kem"`), and then using the parameter estimates from that to as initial conditions for `method="BFGS"`.

<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 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 the EM algorithm. If the quasi-Newton algorithm works, it will tend to find parameters with higher likelihood faster than the EM algorithm. However, the MARSS likelihood surface can be multi-modal with sharp peaks at degenerate solutions where a Q or R diagonal element equals 0. The quasi-Newton algorithm sometimes 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 EM algorithm) will necessarily solve this problem. Thus it is wise to check against EM estimates to ensure that the BFGS estimates are close to the MLE estimates (and vis-a-versa, it's wise to rerun with `method="BFGS"` after using `method="kem"`).

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 EM algorithm is often desirable.

One should be aware that the prior set on the variance of the initial states at  $t=0$  or  $t=1$  can have catastrophic effects on one's estimates if the presumed prior covariance structure conflicts with the structure implied by the MARSS model. For example, if you use a diagonal variance-covariance matrix for the prior but the model implies a matrix with non-zero covariances, your MLE estimates can be strongly influenced by the prior variance-covariance matrix. Setting a diffuse prior does not help because the diffuse prior still has the correlation structure specified by  $V_0$ . One way to detect priors effects is to compare the BFGS estimates to the EM estimates. Persistent differences typically signify a problem with the correlation structure in the prior conflicting with the implied correlation structure in the MARSS model. If this is the case, using  $V_0=0$  and estimating the  $x_0$  elements (with `control$kf.x0="x10"`) can often help.

## 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, model=list(Z=factor(c(1,1,1,1,1))),
  method="BFGS")

#fit a model with EM and then use that fit as the start for BFGS
kemfit = MARSS(dat, model=list(Z=factor(c(1,1,1,1,1))))
bfgsfit = MARSS(dat, model=list(Z=factor(c(1,1,1,1,1))),
  inits=kemfit$par, method="BFGS")
```

---

MARSSoptions

*Change MARSS Defaults Utility*

---

**Description**

Utility in the [MARSS-package](#) to change the defaults, including default model structure, for the MARSS function.

**Usage**

```
MARSSoptions(..., method="kem")
```

**Arguments**

... A name or list of names in alldefaults. To see what alldefaults looks like, type MARSSoptions(). This is the same as (MARSS:::alldefaults)[[method]].

method Estimation method. MARSS allows method="kem" and "BFGS".

**Author(s)**

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

**Examples**

```
## Not run:
#Change the defaults maxit value
MARSSoptions(control=list(maxit=5000))

#Change the defaults minit, maxit and abstol value;
#setting abstol means the program defaults to
#using the delta logLik convergence test
```

```

MARSSoptions(control=list(minit=100, maxit=5000, abstol=0.01))

#Change lots of different defaults
MARSSoptions(control=list(minit=100, maxit=5000, abstol=0.01),
  model=list(U="unequal",Q="unconstrained",R=diag(0.01,4)))

#Show the control defaults
MARSSoptions("control")

#Show all the defaults for method="BFGS";
#this doesn't change the default method
MARSSoptions(method="BFGS")

## End(Not run)

```

---

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 <a href="#">marssMLE</a> . Must have a \$par 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 guide accessed via `RShowDoc("UserGuide", package="MARSS")`

**See Also**

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

**Examples**

```
dat = t(harborSealWA)
dat = dat[2:4,]
kem = MARSS(dat, model=list(Z=factor(c(1,1,1)),
  R="diagonal and unequal"))
kem.with.CIs.from.hessian = MARSSparamCIs(kem)
kem.with.CIs.from.hessian
```

---

MARSSresids

*MARSS standardized residuals*

---

**Description**

Calculates the standardized residuals sensu Harvey, Koopman and Penzer (1998). This is a utility function in the [MARSS-package](#).

**Usage**

```
MARSSresids(MLEobj)
```

**Arguments**

MLEobj            An object of class `marssMLE`. This object must have `$par`, `$model` and `$kf` elements containing MLE parameter estimates from e.g. [MARSSkem](#).

**Details**

Uses the algorithm on page 112 of Harvey, Koopman and Penzer (1998) to compute the standardized model residuals.

**Value**

A list with the following components

<code>et</code>	The model residuals as a $(n+m) \times TT$ matrix with <code>v_t</code> on top and <code>w_t</code> below. This is <code>hat(eta_t)</code> on page 112 of Harvey, Koopman and Penzer (1998).
<code>var.et</code>	The variance of the model residuals as a $(n+m) \times (n+m) \times TT$ matrix. This is <code>var(hat(eta_t))</code> .
<code>std.et</code>	The standardized model residuals as a $(n+m) \times TT$ matrix. This is <code>et</code> divided by the square root of <code>var.et</code> — although the code is using the matrix equivalent of that equation.

**Author(s)**

Eli Holmes, NOAA, Seattle, USA.

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

**References**

Harvey, A., S. J. Koopman, and J. Penzer. 1998. Messy time series: a unified approach. *Advances in Econometrics* 13: 103-144 (see page 112).

Koopman, S. J., N. Shephard, and J. A. Doornik. 1999. Statistical algorithms for models in state space using SsfPack 2.2. *Econometrics Journal* 2: 113-166. (see pages 147-148).

**See Also**

[MARSSkem mar ssMLE](#)

**Examples**

```
dat = t(harborSeal)
dat = dat[2:4,]
MLEobj = MARSS(dat)
std.resids = MARSSresids(MLEobj)$std.et
```

---

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 <code>marssMLE</code> .
<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  $n \times tSteps \times nsim$ , 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  $dim=c(n, tSteps)$  (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 $m \times tSteps \times nsim$ ) of state processes simulated from parameter estimates.
<code>sim.data</code>	Array (dim $n \times tSteps \times nsim$ ) 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[,2:4]
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 <a href="#">marssMLE</a> .
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[2:3,]
kem = MARSS(dat)
paramvec = MARSSvectorizeparam(kem)
paramvec
```

---

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. The Ives data are unlogged. `ivesDataLP` and `ivesDataByWeek` are the same data with LP having the missing weeks in winter removed while in `ByWeek`, the missing values are left in.

## Usage

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

## Format

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

## Source

- `ivesDataLP` and `ivesDataByWeek` 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 planktonHampton`, 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(ivesDataByWeek)
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,
        model=NULL,
        fixed=NULL, free=NULL,
        miss.value=NULL,
        control=NULL,
        method=NULL,
        silent=FALSE)
```

**Arguments**

y	A matrix (not dataframe), observations (rows) x time steps (columns), of data.
allowed	Allowed model structures. This is determined by the method used for parameter estimation. allowed\$kem (specified in MARSSsettings is the set of allowed model structures for the EM algorithm. allowed\$BFGS are the allowable model structures for BFGS.
inits	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 EM and quasi-Newton methods. These are ignored if the specified parameter is not being estimated. If not passed in by the user, <a href="#">MARSS</a> creates generic inits using <a href="#">MARSSinits</a> . B Initial value(s) for B parameter (length 1 or m x m). If length 1, inits\$B constructed as diag(value, m). U Initial value(s) for U parameter (length 1 or m x 1). If length 1, inits\$U constructed as matrix(value, nrow=m, ncol=1). Q Initial value(s) for Q parameter (length 1 or m x m). If length 1, inits\$Q constructed as diag(value, m). 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, inits\$A constructed as matrix(value, nrow=n, ncol=1). R Initial value(s) for R parameter (length 1 or n x n). If length 1, inits\$R constructed as diag(value, n).

	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.
model		Model specification as a list. See <a href="#">MARSS</a> for details.
fixed		Optional model specification using matrices of fixed and free parameters. See user guide for details.
free		Optional model specification using matrices of fixed and free parameters. See user guide for details.
miss.value		How are missing values represented in the data?
method		The method used for estimation. This is needed for setting default values for control.
control		Control options for maximization algorithms.
	minit	Minimum number of EM iterations.
	maxit	Maximum number of EM iterations.
	abstol	Optional tolerance for log-likelihood change. If log-likelihood changes less than this amount relative to the previous iteration, the algorithm exits.
	allow.degen	Whether to try setting Q or R elements to zero if they appear to be going to zero.
	trace	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.
	safe	If TRUE, <a href="#">MARSSkem</a> will rerun <a href="#">MARSSkf</a> after each individual parameter update rather than only after all parameters are updated.
	MCInit	Use Monte Carlo initialization? See discussion in <a href="#">MARSSkem</a> and <a href="#">MARSSmcinit</a> .
	numInits	Number of random initial value draws.
	numInitSteps	Number of EM iterations for each random initial value draw.
	boundsInits	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 the df and $S=\text{diag}(\text{bound}[2],m)$ for a wishart distribution.
silent		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`, `model`, or `control` are not provided by the user, they will be set by the `alldefaults[[method]]` object specified in `MARSSsettings`. Argument `model` specifies the model structure using a list of matrices; see [MARSS](#) or the user guide for instructions on how to specify model structure.

**Value**

An object of class 'popWrap'.

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

---

stdInnov	<i>Standardized Innovations</i>
----------	---------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

SIGMA	$n \times n \times T$ array of Kalman filter innovations variances. This is output from <a href="#">MARSSkf</a> .
INNOV	$n \times T$ matrix of Kalman filter innovations. This is output from <a href="#">MARSSkf</a> .

**Details**

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

**Value**

$n \times T$  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, [41](#)
  - marssMLE-class, [45](#)
  - popWrap-class, [57](#)
- \*Topic **datasets**
  - graywhales, [8](#)
  - harborSeal, [10](#)
  - loggerhead, [12](#)
  - plankton, [54](#)
- \*Topic **hplot**
  - CSEGriskfigure, [6](#)
  - CSEGtmfigure, [7](#)
- \*Topic **package**
  - MARSS-package, [3](#)
- alldefaults (allowed), [4](#)
- allowed, [4](#)
- as.design (is.blockdiag), [11](#)
- as.marssm, [5](#), [57](#)
- as.marssm (marssm), [38](#)
- checkPopWrap, [4](#), [5](#), [56](#), [57](#)
- CSEGriskfigure, [6](#), [8](#)
- CSEGtmfigure, [7](#), [7](#)
- describe.marssm (allowed), [4](#)
- fdHess, [26](#)
- graywhales, [8](#)
- grouse (graywhales), [8](#)
- harborSeal, [10](#)
- harborSealnomiss (harborSeal), [10](#)
- harborSealWA (harborSeal), [10](#)
- Imat (is.blockdiag), [11](#)
- is.blockdiag, [11](#)
- is.blocequaltri (is.blockdiag), [11](#)
- is.blockunconst (is.blockdiag), [11](#)
- is.design (is.blockdiag), [11](#)
- is.diagonal (is.blockdiag), [11](#)
- is.equaltri (is.blockdiag), [11](#)
- is.fixed (is.blockdiag), [11](#)
- is.identity (is.blockdiag), [11](#)
- is.marssm, [56](#)
- is.marssm (marssm), [38](#)
- is.marssMLE (marssMLE), [43](#)
- is.wholenumber (is.blockdiag), [11](#)
- isleRoyal (graywhales), [8](#)
- ivesDataByWeek (plankton), [54](#)
- ivesDataLP (plankton), [54](#)
- kem.methods (allowed), [4](#)
- kestrel (graywhales), [8](#)
- lakeWaplankton (plankton), [54](#)
- loggerhead, [12](#)
- loggerheadNoisy (loggerhead), [12](#)
- makediag (is.blockdiag), [11](#)
- MARSS, [3–5](#), [13](#), [25](#), [28](#), [30](#), [36](#), [42](#), [45](#), [46](#), [48](#), [55–57](#)
- MARSS-package, [5–8](#), [10–13](#), [15](#), [17](#), [19–21](#), [23](#), [24](#), [26–28](#), [30](#), [33](#), [34](#), [37](#), [38](#), [41](#), [43](#), [45](#), [48–51](#), [53–55](#), [57](#), [58](#)
- MARSS-package, [3](#)
- MARSSaic, [4](#), [18](#), [23](#)
- MARSSapplynames, [20](#)
- MARSSboot, [4](#), [7](#), [20](#), [21](#), [29](#), [50](#), [52](#), [58](#)
- MARSScheckdims, [23](#)
- MARSScheckpar (MARSScheckdims), [23](#)
- MARSShatyt, [24](#)
- MARSShessian, [26](#), [50](#)
- MARSSinits, [27](#), [55](#)
- MARSSinnovationsboot, [28](#), [50](#), [58](#)
- MARSSkem, [3](#), [14](#), [16](#), [17](#), [25–28](#), [30](#), [33–36](#), [41–43](#), [45](#), [48](#), [50](#), [51](#), [56](#)
- MARSSkemcheck, [33](#)
- MARSSkf, [3](#), [16](#), [24](#), [30](#), [32](#), [33](#), [34](#), [45](#), [47](#), [56](#), [58](#)

MARSSkfas, 3, 45  
MARSSkfas (MARSSkf), 34  
MARSSLprofile, 37  
marssm, 13, 15–17, 20–23, 25, 27, 28, 33, 34,  
36, 38, 43, 45, 52, 57  
marssm-class, 41  
MARSSmcinit, 30, 31, 33, 41, 46, 56  
marssMLE, 5–7, 13, 14, 17, 19–24, 26, 28–31,  
33, 34, 37, 41, 42, 43, 43, 45, 46,  
48–53  
marssMLE-class, 45  
MARSSoptim, 3, 4, 14, 16, 17, 27, 28, 33, 41,  
43, 45  
MARSSoptions, 48  
MARSSparamCIs, 4, 6, 26, 29, 38, 47, 49  
MARSSresids, 50  
MARSSsettings (MARSS), 13  
MARSSsimulate, 3, 51  
MARSSvectorizeparam, 37, 45, 53  
model.elem (allowed), 4  
  
negloglik (MARSSoptim), 45  
nlme, 26  
  
okanaganRedds (graywhales), 8  
optim, 4, 43, 45–48  
optim.methods (allowed), 4  
  
plankton, 54  
popWrap, 5, 39, 40, 55  
popWrap-class, 57  
prairiechicken (graywhales), 8  
print,marssm-method (marssm-class), 41  
print,marssMLE-method (marssMLE-class),  
45  
  
rockfish (graywhales), 8  
  
stdInnov, 29, 58  
summary,marssm-method (marssm-class), 41  
summary,marssMLE-method  
(marssMLE-class), 45  
  
takediag (is.blockdiag), 11  
  
unvec (is.blockdiag), 11  
  
vec (is.blockdiag), 11  
  
wilddogs (graywhales), 8