MARSS: Multivariate Autoregressive State-space Models for Analyzing Time-series Data

by Elizabeth E. Holmes, Eric J. Ward, Kellie Wills

Abstract MARSS is a package for fitting multivariate autoregressive state-space models to time-series data. The MARSS package implements state-space models in a maximum likelihood framework. The core functionality of MARSS is based on likelihood maximization using the Kalman filter/smoother, combined with an EM algorithm. To make comparisons with other packages available, parameter estimation is also permitted via direct search routines available in 'optim'. The MARSS package allows data to contain missing values and allows a wide variety of model structures and constraints to be specified (such as fixed or shared parameters). In addition to model-fitting, the package provides bootstrap routines for simulating data and generating confidence intervals, and multiple options for calculating model selection criteria (such as AIC).

The **MARSS** package (Holmes et al., 2012) is an R package for fitting linear multivariate autoregressive state-space (MARSS) models with Gaussian errors to time-series data. This class of model is extremely important in the study of linear stochastic dynamical systems, and these models are used in many different fields, including economics, engineering, genetics, physics and ecology. The model class has different names in different fields; some common names are dynamic linear models (DLMs) and vector autoregressive (VAR) state-space models. There are a number of existing R packages for fitting this class of models, including **sspir** (Dethlefsen et al., 2009) for univariate data and **dlm** (Petris, 2010), **dse** (Gilbert, 2009), **KFAS** (Helske, 2011) and **FKF** (Luethi et al., 2012) for multivariate data. Additional packages are available on other platforms, such as SsfPack (Durbin and Koopman, 2001), EViews (`www.eviews.com`) and Brodgar (`www.brodgar.com`). Except for Brodgar and **sspir**, these packages provide maximization of the likelihood surface (for maximum-likelihood parameter estimation) via quasi-Newton or Nelder-Mead type algorithms. The **MARSS** package was developed to provide an alternative maximization algorithm, based instead on an Expectation-Maximization (EM) algorithm and to provide a standardized model-specification framework for fitting different model structures.

The **MARSS** package was originally developed for researchers analyzing data in the natural and environmental sciences, because many of the problems often encountered in these fields are not commonly encountered in disciplines like engineering or finance. Two typical problems are high fractions of irregularly spaced missing observations and observation error variance that cannot be estimated or known a priori (Schnute, 1994). Packages developed for other fields did not always allow estimation of the parameters of interest to ecologists because these parameters are always fixed in the package authors' field or application. The **MARSS** package was developed to address these issues and its three main differences are summarized as follows.

First, maximum-likelihood optimization in most packages for fitting state-space models relies on quasi-Newton or Nelder-Mead direct search routines, such as provided in `optim` (for **dlm**) or `nlm` (for **dse**). Multidimensional state-space problems often have complex, non-linear likelihood surfaces. For certain types of multivariate state-space models, an alternative maximization algorithm exists; though generally slower for most models, the EM-algorithm (Metaxoglou and Smith, 2007; Shumway and Stoffer, 1982), is considerably more robust than direct search routines (this is particularly evident with large amounts of missing observations). To date, no R package for the analysis of multivariate state-space models has implemented the EM algorithm for maximum-likelihood parameter estimation (**sspir** implements it for univariate models). In addition, the **MARSS** package implements an EM algorithm for constrained parameter estimation (Holmes, 2010) to allow fixed and shared values within parameter matrices. To our knowledge, this constrained EM algorithm is not implemented in any package, although the Brodgar package implements a limited version for dynamic factor analysis.

Second, model specification in the **MARSS** package has a one-to-one relationship to a MARSS model as written in matrix form on paper. Any model that can be written in MARSS form can be fitted without extra code by the user. In contrast, other packages require users to write unique functions in matrix form (a non-trivial task for many non-expert R users). For example, while **dlm** includes linear and polynomial univariate models, multivariate regression is not readily accessible without these custom functions; in **MARSS**, all models written in matrix form are fitted using the same model specification.

The **MARSS** package also allows degenerate multivariate models to be fitted, which means that some or all observation or process variances can be set to 0. This allows users to include deterministic features in their models and to rewrite models with longer lags or moving averaged errors as a MARSS model.

Third, the **MARSS** package provides algorithms for computing model selection criteria that are specific for state-space models. Model selection criteria are used to quantify the data support for different model and parameter structures by balancing the ability of the model to fit the data against the flexibility of the model. The criteria computed in the **MARSS** package are based on Akaike's Information Criterion (AIC). Models with the lowest AIC are interpreted as receiving more data support. While AIC and its small sample corrected version AICc are easily calculated for fixed effects models (these are standard output for `lm` and `glm`, for instance), these criteria are biased for hierarchical or state-space autoregressive models. The **MARSS** package provides an unbiased AIC criterion via innovations bootstrapping (Cavanaugh and Shumway, 1997; Stoffer and Wall, 1991) and parametric bootstrapping (Holmes, 2010). The package also provides functions for approximate and bootstrap confidence intervals, and bias correction for estimated parameters.

The package comes with an extensive user guide that introduces users to the package and walks the user through a number of case studies involving ecological data. The selection of case studies includes estimating trends with univariate and multivariate data, making inferences concerning spatial structure with multi-location data, estimating inter-species interaction strengths using multi-species data, using dynamic factor analysis to reduce the dimension of a multivariate dataset, and detecting structural breaks in data sets. Though these examples have an ecological focus, the analysis of multivariate time series models is cross-disciplinary work and researchers in other fields will likely benefit from these examples.

The MARSS model

The MARSS model includes a process model and an observation model. The process component of a MARSS model is a multivariate first-order autoregressive (MAR-1) process. The multivariate process model takes the form

$$\mathbf{x}_t = \mathbf{B}\mathbf{x}_{t-1} + \mathbf{u} + \mathbf{w}_t; \quad \mathbf{w}_t \sim \text{MVN}(0, \mathbf{Q}) \tag{1}$$

The $\mathbf{x}$ is an $m \times 1$ vector of state values, equally spaced in time, and $\mathbf{B}$, $\mathbf{u}$ and $\mathbf{Q}$ are the state process parameters. The $m \times m$ matrix $\mathbf{B}$ allows interaction between state processes; the diagonal elements of $\mathbf{B}$ can also be interpreted as coefficients of autoregression in the state vectors through time. The vector $\mathbf{u}$ describes the mean trend or mean level (depending on the $\mathbf{B}$ structure), and the correlation of the process deviations is determined by the structure of the matrix $\mathbf{Q}$. In version 2.x of the **MARSS** package, the $\mathbf{B}$ and $\mathbf{Q}$ parameters are time-invariant; however, in version 3.x, all parameters can be time-varying and also the $\mathbf{u}$ can be moved into the $\mathbf{x}$ term to specify models with an auto-regressive trend process (called a stochastic level model).

Written out for two state processes, the MARSS process model would be:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_t = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{t-1} + \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_t,$$

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_t \sim \text{MVN}\left(0, \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix}\right)$$

Some of the parameter elements will generally be fixed to ensure identifiability. Within the MAR-1 form, MAR-p or autoregressive models with lag-p and models with exogenous factors (covariates) can be included by properly defining the $\mathbf{B}$, $\mathbf{x}$ and $\mathbf{Q}$ elements. See for example Tsay (2010) where the formulation of various time-series models as MAR-1 models is covered.

The initial state vector is specified at $t = 0$ or $t = 1$ as

$$\mathbf{x}_0 \sim \text{MVN}(\boldsymbol{\pi}, \boldsymbol{\Lambda}) \quad \text{or} \quad \mathbf{x}_1 \sim \text{MVN}(\boldsymbol{\pi}, \boldsymbol{\Lambda}) \tag{2}$$

where $\boldsymbol{\pi}$ is the mean of the initial state distribution or a constant. $\boldsymbol{\Lambda}$ specifies the variance of the initial states or is specified as 0 if the initial state is treated as fixed (but possibly unknown).

The multivariate observation component in a MARSS model is expressed as

$$\mathbf{y}_t = \mathbf{Z}\mathbf{x}_t + \mathbf{a} + \mathbf{v}_t; \quad \mathbf{v}_t \sim \text{MVN}(0, \mathbf{R}) \tag{3}$$

where $\mathbf{y}_t$ is an $n \times 1$ vector of observations at time t, $\mathbf{Z}$ is an $n \times m$ matrix, $\mathbf{a}$ is an $n \times 1$ matrix, and the correlation structure of observation errors is specified with the matrix $\mathbf{R}$. Including $\mathbf{Z}$ and $\mathbf{a}$ is not required for every model, but these parameters are used when some state processes are observed multiple times (perhaps with different scalings) or when observations are linear combinations of the state processes (such as in dynamic factor analysis). Note that time steps in the model are equidistant, but there is no requirement that there be an observation at every time step; $\mathbf{y}$ may have missing values scattered thoughout it.

Written out for three observation processes and two state processes, the MARSS observation model would be

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}_t = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \\ z_{31} & z_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_t + \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_t$$

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_t \sim \text{MVN}\left(0, \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\right)$$

Again, some of the parameter elements will generally be fixed to ensure identifiability.

The MARSS process and observation models are flexible and many multivariate time-series models can be rewritten in MARSS form. See textbooks on time series analysis where reformulating AR-p, ARMA-p as a MARSS model is covered (such as Tsay, 2010; Harvey, 1989).

Package overview

The package is designed to fit MARSS models with fixed and shared elements within the parameter matrices. The following shows such a model using a mean-reverting random walk model with three observation time series as an example:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_t = \begin{bmatrix} b & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{t-1} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_t \tag{4a}$$

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_t \sim \text{MVN}\left(0, \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix}\right) \tag{4b}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_0 \sim \text{MVN}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \tag{4c}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_t + \begin{bmatrix} a_1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_t \tag{4d}$$

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_t \sim \text{MVN}\left(0, \begin{bmatrix} r_1 & 0 & 0 \\ 0 & r_2 & 0 \\ 0 & 0 & r_2 \end{bmatrix}\right) \tag{4e}$$

Notice that many parameter elements are fixed, while others are shared (have the same symbol).

Model specification

Model specification has a one-to-one relationship to the model written on paper in matrix form (Equation 4). The user passes in a list which includes an element for each parameter in the model: `B`, `U`, `Q`, `Z`, `A`, `R`, `x0`, `V0`. The list element specifies the form of the corresponding parameter in the model.

The most general way to specify a parameter form is to use a list matrix. The list matrix allows one to combine fixed and estimated elements in one's parameter specification. For example, to specify the parameters in Equation 4, one would write the following list matrices:

```
> B1 = matrix(list("b", 0, 0, "b"), 2, 2)
> U1 = matrix(0, 2, 1)
> Q1 = matrix(c("q11", "q12", "q12", "q22"), 2, 2)
> Z1 = matrix(c(1, 0, 1, 0, 1, 0), 3, 2)
> A1 = matrix(list("a1", 0, 0), 3, 1)
> R1 = matrix(list("r1", 0, 0, 0, "r2", 0,
+     0, 0, "r2"), 3, 3)
> pi1 = matrix(0, 2, 1)
> V1 = diag(1, 2)
> model.list = list(B = B1, U = U1, Q = Q1, Z = Z1,
+     A = A1, R = R1, x0 = pi1, V0 = V1)
```

When printed at the R command line, each parameter matrix looks exactly like the parameters as written out in Equation 4. Numeric values are fixed and character values are names of elements to be estimated. Elements with the same character name are constrained to be equal (no sharing across parameter matrices, only within).

List matrices allow the most flexible model structures, but **MARSS** also has text shortcuts for many common model structures. The structure of the variance-covariance matrices in the model are specified via the `Q`, `R`, and `V0` components, respectively. Common structures are errors independent with one variance on the diagonal (specified with `"diagonal and equal"`) or all variances different (`"diagonal and unequal"`); errors correlated with the same correlation parameter (`"equalvarcov"`) or correlated with each process having unique correlation parameters (`"unconstrained"`), or all zero (`"zero"`). Common structures for the matrix **B** are an identity matrix (`"identity"`); a diagonal matrix with one value on the diagonal (`"diagonal and equal"`) or all different values on the diagonal (`"diagonal and unequal"`), or all values different (`"unconstrained"`). Common structures for the **u** and **a** parameters are all equal (`"equal"`); all unequal (`"unequal"` or `"unconstrained"`), or all zero (`"zero"`).

The **Z** matrix is idiosyncratic and there are fewer shortcuts for its specification. One common form for **Z** is a design matrix, composed of 0s and 1s, in which each row sum equals 1. This is used when each y in $\mathbf{y}$ is associated with one x in $\mathbf{x}$ (one x may be associated with multiple y's). In this case, **Z** can be specified using a factor of length n. The factor specifies to which x each of the n y's correspond. For example, the **Z** in Equation 4 could be specified `factor(c(1,2,1))` or `factor(c("a","b","a"))`.

Model fitting with the MARSS function

The main package function is `MARSS`. This function fits a MARSS model (Equations 1 and 3) to a matrix of data and returns the maximum-likelihood estimates for the $\mathbf{B}$, $\mathbf{u}$, $\mathbf{Q}$, $\mathbf{Z}$, $\mathbf{a}$, $\mathbf{R}$, $\boldsymbol{\pi}$, and $\boldsymbol{\Lambda}$ parameters or more specifically, the free elements in those parameters since many elements will be fixed for a given model form. The basic `MARSS` call takes the form:

```
> MARSS(data, model = model.list)
```

where `model.list` has the form shown in the R code above. The data must be passed in as an $n \times T$ matrix, that is time goes across columns. A vector is not a matrix, nor is a data frame. A matrix of 3 inputs ($n = 3$) measured for 6 time steps might look like

$$\mathbf{y} = \begin{bmatrix} 1 & 2 & \text{NA} & \text{NA} & 3.2 & 8 \\ 2 & 5 & 3 & \text{NA} & 5.1 & 5 \\ 1 & \text{NA} & 2 & 2.2 & \text{NA} & 7 \end{bmatrix}$$

where NA denotes a missing value. Note that the time steps are equidistant, but there may not be data at each time step, thus the observations are not constrained to be equidistant.

The only required argument to `MARSS` is a matrix of data. The `model` argument is an optional argument, although typically included, which specifies the form of the MARSS model to be fitted (if left off, a default form is used). Other `MARSS` arguments include initial parameter values (`inits`), a non-default value for missing values (`miss.value`), whether or not the model should be fitted (`fit`), whether or not output should be verbose (`silent`), and the method for maximum-likelihood estimation (`method`). The default method is the EM algorithm ('`method = "kem"`'), but the quasi-Newton method BFGS is also allowed ('`method = "BFGS"`'). Changes to the default fitting algorithm are specified with `control`; for example, `control$minit` is used to change the minimum number of iterations used in the maximization algorithm. Use `?MARSS` to see all the `control` options.

Each call to the `MARSS` function returns an object of class `"marssMLE"`, an estimation object. The `marssMLE` object is a list with the following elements: the estimated parameter values (`par`), Kalman filter and smoother output (`kf`), convergence diagnostics (`convergence`, `numIter`, `errors`, `logLik`), basic model selection metrics (`AIC`, `AICc`), and a model object `model` which contains both the MARSS model specification and the data. Further list elements, such as bootstrap AIC or confidence intervals, are added with functions that take a `marssMLE` object as input.

Model selection and confidence intervals

One of the most commonly used model selection tools in the maximum-likelihood framework is Akaike's Information Criterion (AIC) or the small sample variant (AICc). Both versions are returned with the call to `MARSS`. A state-space specific model selection metric, AICb (Cavanaugh and Shumway, 1997; Stoffer and Wall, 1991), can be added to a `marssMLE` object using the function `MARSSaic`.

The function `MARSSparamCIs` is used to add confidence intervals and bias estimates to a `marssMLE` object. Confidence intervals may be computed by resampling the Kalman innovations if all data are present (innovations bootstrapping) or implementing a parametric bootstrap if some data are missing (parametric bootstraping). All bootstrapping in the **MARSS** package is done with the lower-level function `MARSSboot`. Because bootstrapping can be time-consuming, `MARSSparamCIs` also allows approximate confidence intervals to be calculated with a numerically estimated Hessian matrix. If bootstrapping is used to generate confidence intervals, `MARSSparamCIs` will also return a bootstrap estimate of parameter bias.

Estimated state trajectories

In addition to outputting the maximum-likelihood estimates of the model parameters, the `MARSS` call will also output the maximum-likelihood estimates of the state trajectories (the $\mathbf{x}$ in the MARSS model) conditioned on the entire dataset. These states represent output from the Kalman smoother using the maximum-likelihood parameter values. The estimated states are in the `states` element in the `marssMLE` object output from a `MARSS` call. The standard errors of the state estimates are in element `states.se` of the `marssMLE` object. For example, if a MARSS model with one state ($m = 1$) was fitted to data and the output placed in `fit`, the state estimates and ± 2 standard errors could be plotted with the following code:

```
> plot(fit$states, type = "l", lwd = 2)
> lines(fit$states - 2*fit$states.se)
> lines(fit$states + 2*fit$states.se)
```

Example applications

To demonstrate the **MARSS** package functionality, we show a series of short examples based on a few of the case studies in the MARSS User Guide. The user guide includes much more extensive analysis and includes many other case studies. All the case studies are based on analysis of ecological data, as this was the original motivation for the package.

Trend estimation

A commonly used model in ecology describing the dynamics of a population experiencing density-dependent growth is the Gompertz model (Reddingius and den Boer, 1989; Dennis and Taper, 1994). The population dynamics are stochastic and driven by the combined effects of environmental variation and random demographic variation.

The log-population sizes from this model can be described by an AR-1 process:

$$x_t = bx_{t-1} + u + w_t; \quad w_t \sim Normal(0, \sigma^2) \qquad (5)$$

In the absence of density dependence ($b = 1$), the exponential growth rate or rate of decline of the population is controlled by the parameter u; when the population experiences density-dependence ($|b| < 1$), the population will converge to an equilibrium $u/(1-b)$. The initial states are given by

$$x_0 \sim Normal(\pi, \lambda) \qquad (6)$$

The observation process is described by

$$y_t = x_t + v_t; \quad v_t \sim Normal(0, \eta^2) \qquad (7)$$

where y is a vector of observed log-population sizes. The bias parameter, a, has been set equal to 0.

This model with b set to 1 is a standard model used for species of concern (Dennis et al., 1991) when the population is well below carrying capacity and the goal is to estimate the underlying trend (u) for use in population viability analysis (Holmes, 2001; Holmes et al., 2007; Humbert et al., 2009). To illustrate a simple trend analysis, we use the `graywhales` dataset included in the **MARSS** package. This data set consists of 24 abundance estimates of eastern North Pacific gray whales (Gerber et al., 1999). This population was depleted by commercial whaling prior to 1900 and has steadily increased since the first abundance estimates were made in the 1950s (Gerber et al., 1999). The dataset consists of 24 annual observations over 39 years, 1952–1997; years without an estimate are denoted as `NA`. The time step between observations is one year and the goal is to estimate the annual rate of increase (or decrease).

After log-transforming the data, maximum-likelihood parameter estimates can be calculated using `MARSS`:

```
> data(graywhales)
> years = graywhales[,1]
> loggraywhales = log(graywhales[,2])
> kem = MARSS(loggraywhales)
```

The `MARSS` wrapper returns an object of class `"marssMLE"`, which contains the parameter estimates (`kem$par`), state estimates (`kem$states`) and their standard errors (`kem$states.se`). The state estimates can be plotted with ± 2 standard errors, along with the original data (Figure 1).

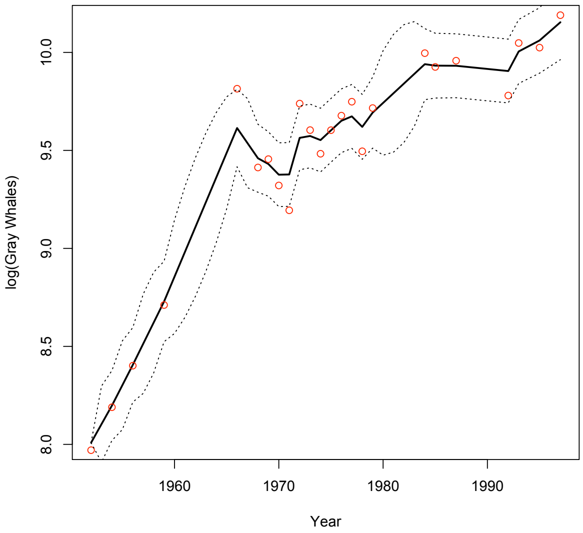

Figure 1: Maximum-likelihood estimates of eastern North Pacific gray whales (solid line) population size, with data (points) and 2 standard errors (dashed lines). Additional examples of trend estimation may be found in MARSS User Guide.

Identification of spatial population structure using model selection

In population surveys, censuses are often taken at multiple sites and those sites may or may not be connected through dispersal. Adjacent sites with high exchange rates of individuals will synchronize subpopulations (making them behave as a single subpopulation), while sites with low mixing rates will behave more independently as discrete subpopulations. We can use MARSS models and model selection to test hypotheses concerning the spatial structure of a population (Hinrichsen, 2009; Hinrichsen and Holmes, 2009; Ward et al., 2010).

For the multi-site application, $\mathbf{x}$ is a vector of abundance in each of m subpopulations and $\mathbf{y}$ is a vector of n observed time series associated with n sites. The number of underlying processes, m, is controlled by the user and not estimated. In the multi-site scenario, $\mathbf{Z}$, controls the assignment of survey sites to subpopulations. For instance, if the population is modeled as having four independent subpopulations and the first two were surveyed at one site each and the third was surveyed at two sites, then $\mathbf{Z}$ is a 4×3 matrix with (1, 0, 0, 0) in the first column, (0, 1, 0, 0) in the second and (0, 0, 1, 1) in the third. In the `model` argument for the `MARSS` function, we could specify this as '`Z = as.factor(c(1,2,3,3))`'. The $\mathbf{Q}$ matrix controls the temporal correlation in the process errors for each subpopulation; these errors could be correlated or uncorrelated. The $\mathbf{R}$ matrix controls the correlation structure of observation errors between the n survey sites.

The dataset `harborSeal` in the **MARSS** package contains survey data for harbor seals on the west coast of the United States. We can test the hypothesis that the four survey sites in Puget Sound (Washington state) are better described by one panmictic population measured at four sites versus four independent subpopulations.

```
> dat = t(log(harborSeal[,4:7]))
> harbor1 = MARSS(dat,
+     model = list(Z = factor(rep(1, 4))))
> harbor2 = MARSS(dat,
+     model = list(Z = factor(1:4)))
```

The default model settings of '`Q = "diagonal and unequal"`', '`R = "diagonal and equal"`', and '`B = "identity"`' are used for this example. The AICc value for the one subpopulation surveyed at four sites is considerably smaller than the four subpopulations model. This suggests that these four sites are within one subpopulation (which is what one would expect given their proximity and lack of geographic barriers).

```
> harbor1$AICc

[1] -280.0486
```

```
> harbor2$AICc

[1] -254.8166
```

We can add a bootstrapped AIC using the function `MARSSaic`.

```
> harbor1 = MARSSaic(harbor1,
+     output = c("AICbp"))
```

We can add a confidence intervals to the estimated parameters using the function `MARSSparamCIs`.

```
> harbor1 = MARSSparamCIs(harbor1)
```

The `MARSS` default is to compute approximate confidence intervals using a numerically estimated Hessian matrix. Two full model selection exercises can be found in the MARSS User Guide.

Analysis of animal movement data

Another application of state-space modeling is identification of true movement paths of tagged animals from noisy satellite data. Using the **MARSS** package to analyze movement data assumes that the observations are equally spaced in time, or that enough `NA`s are included to make the observations spaced accordingly. The **MARSS** package includes the satellite tracks from eight tagged sea turtles, which can be used to demonstrate estimation of location from noisy data.

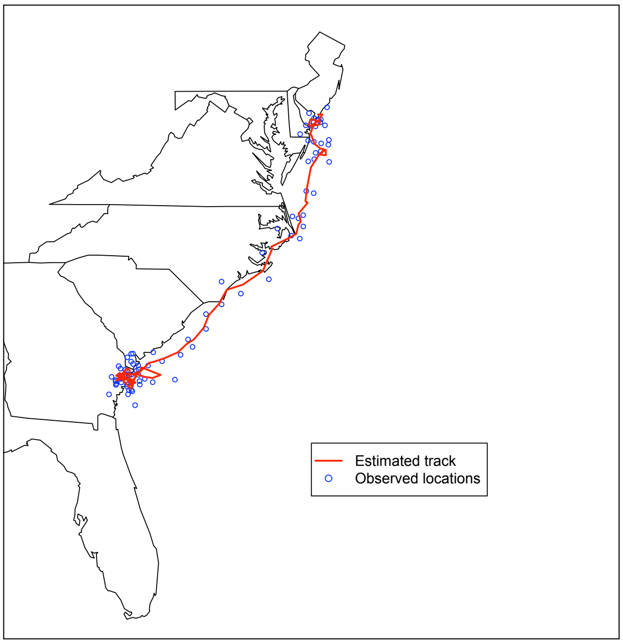

Figure 2: Estimated movement track for one turtle ("Big Mama") using a two-dimensional state-space model. The MARSS User Guide shows how to produce this figure from the satellite data provided with the package.

The data consist of daily longitude and latitude for each turtle. The true location is the underlying state **x**, which is two-dimensional (latitude and longitude) and we have one observation time series for each state process. To model movement as a random walk with drift in two dimensions, we set **B** equal to an identity matrix and constrain **u** to be unequal to allow for non-isotrophic movement. We constrain the process and observation variance-covariance matrices (**Q**, **R**) to be diagonal to specify that the errors in latitude and longitude are independent. Figure 2 shows the estimated locations for the turtle named "Big Mama".

Estimation of species interactions from multi-species data

Ecologists are often interested in inferring species interactions from multi-species datasets. For example, Ives et al. (2003) analyzed four time series of predator (zooplankton) and prey (phytoplankton) abundance estimates collected weekly from a freshwater lake over seven years. The goal of their analysis was to estimate the per-capita effects of each species on every other species (predation, competition), as well as the per-capita effect of each species on itself (density dependence). These types of interactions can be modeled with a multivariate Gompertz model:

$$\mathbf{x}_t = \mathbf{B}\mathbf{x}_{t-1} + \mathbf{u} + \mathbf{w}_t; \quad \mathbf{w}_t \sim \text{MVN}(0, \mathbf{Q}) \qquad (8)$$

The **B** matrix is the interaction matrix and is the main parameter to be estimated. Here we show an example of estimating **B** using one of the Ives et al. (2003) datasets. This is only for illustration; in a real analysis, it is critical to take into account the important environmental covariates, otherwise correlation driven by a covariate will affect the **B** estimates. We show how to incorporate covariates in the MARSS User Guide.

The first step is to log-transform and de-mean the data:

```
> plank.dat = t(log(ivesDataByWeek[,c("Large Phyto",
+   "Small Phyto","Daphnia","Non-daphnia")]))
> d.plank.dat = (plank.dat - apply(plank.dat, 1,
+     mean, na.rm = TRUE))
```

Second, we specify the MARSS model structure. We assume that the process variances of the two phytoplankton species are the same and that the process variances for the two zooplankton are the same. But we assume that the abundance of each species is an independent process. Because the data have been demeaned, the parameter **u** is set to zero. Following Ives et al. (2003), we set the observation error to 0.04 for phytoplankton and to 0.16 for zooplankton. The model is specified as follows

```
> Z = factor(rownames(d.plank.dat))
> U = "zero"
> A = "zero"
> B = "unconstrained"
> Q = matrix(list(0), 4, 4)
> diag(Q) = c("Phyto", "Phyto", "Zoo", "Zoo")
> R = diag(c(0.04, 0.04, 0.16, 0.16))
> plank.model = list(Z = Z, U = U, Q = Q,
+                    R = R, B = B, A = A, tinitx=1)
```

We do not specify x0 or V0 since we assume the default behavior which is that the initial states are treated as estimated parameters with 0 variance.

We can fit this model using MARSS:

```
> kem.plank = MARSS(d.plank.dat,
+     model = plank.model)

> B.est = matrix(kem.plank$par$B, 4, 4)
> rownames(B.est) = colnames(B.est) =
+     c("LP", "SP", "D", "ND")
> print(B.est, digits = 2)

       LP     SP      D     ND
LP  0.549 -0.23  0.096 0.085
SP -0.058  0.52  0.059 0.014
D   0.045  0.27  0.619 0.455
ND -0.097  0.44 -0.152 0.909
```

where "LP" and "SP" represent large and small phytoplankton, respectively, and "D" and "ND" represent the two zooplankton species, “Daphinia” and “Non-Daphnia”, respectively.

Elements on the diagonal of **B** indicate the strength of density dependence. Values near 1 indicate no density dependence; values near 0 indicate strong density dependence. Elements on the off-diagonal of **B** represent the effect of species j on the per-capita growth rate of species i. For example, the second time series (small phytoplankton) appears to have relatively strong positive effects on both zooplankton time series. This is somewhat expected, as more food availability might lead to higher predator densities. The predators appear to have relatively little effect on their prey; again, this is not surprising as phytoplankton densities are often strongly driven by environmental covariates (temperature and pH) rather than predator densities. Our estimates of **B** are different from those presented by Ives et al. (2003); in the MARSS User Guide we illustrate how to recover the estimates in Ives et al. (2003) by fixing elements to zero and including covariates.

Dynamic factor analysis

In the previous examples, each observed time series was representative of a single underlying state process, though multiple time series may be observations of the same state process. This is not a requirement for a MARSS model, however. Dynamic factor analysis (DFA) also uses MARSS models but treats each observed time series as a linear combination of multiple state processes. DFA has been applied in a number of fields (e.g. Harvey, 1989) and can be thought of as a principal components analysis for time-series data (Zuur et al., 2003a,b). In a DFA, the objective is to estimate the number of underlying state processes, m, and the **Z** matrix now represents how these state processes are linearly combined to explain the larger set of n observed time series. Model selection is used to select the size of m and the objective is to find the most parsimonious number of underlying trends (size of m) plus their weightings (**Z** matrix) that explain the larger dataset.

To illustrate, we show results from a DFA analysis for six time series of plankton from Lake Washington (Hampton et al., 2006). Using model selection with models using $m = 1$ to $m = 6$, we find that the best (lowest AIC) model has four underlying trends, reduced from the original six. When $m = 4$, the DFA observation process then has the following form:

$$\begin{bmatrix} y_{1,t} \\ y_{2,t} \\ y_{3,t} \\ y_{4,t} \\ y_{5,t} \\ y_{6,t} \end{bmatrix} = \begin{bmatrix} \gamma_{11} & 0 & 0 & 0 \\ \gamma_{21} & \gamma_{22} & 0 & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & \gamma_{44} \\ \gamma_{51} & \gamma_{52} & \gamma_{53} & \gamma_{54} \\ \gamma_{61} & \gamma_{62} & \gamma_{63} & \gamma_{64} \end{bmatrix} \mathbf{x}_t + \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} + \begin{bmatrix} v_{1,t} \\ v_{2,t} \\ v_{3,t} \\ v_{4,t} \\ v_{5,t} \\ v_{6,t} \end{bmatrix}$$

Figure 3 shows the fitted data with the four state trajectories superimposed.

The benefit of reducing the data with DFA is that we can reduce a larger dataset into a smaller set of underlying trends and we can use factor analysis to identify whether some time series fall into clusters represented by similar trend weightings. A more detailed description of dynamic factor analysis, including the use of factor rotation to intrepret the **Z** matrix, is presented in the MARSS User Guide.

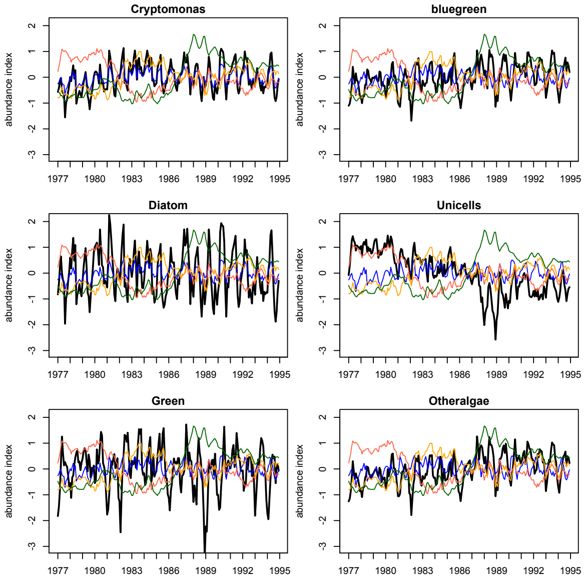

Figure 3: Results of a dynamic factor analysis applied to the Lake Washington plankton data. Colors represent estimated state trajectories and black lines represent the fitted values for each species.

Summary

We hope this package will provide scientists in a variety of disciplines some useful tools for the analysis of noisy univariate and multivariate time-series data, and tools to evaluate data support for different model structures for time-series observations. Future development will include Bayesian estimation and constructing non-linear time-series models.

Bibliography

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

B. Dennis and M. L. Taper. Density dependence in time series observations of natural populations: Estimation and testing. *Ecological Monographs*, 64 (2):205–224, 1994.

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

C. Dethlefsen, S. Lundbye-Christensen, and A. L. Christensen. *sspir: state space models in R*, 2009. URL `http://CRAN.R-project.org/package=sspir`. R package version 0.2.8.

J. Durbin and S. J. Koopman. *Time series analysis by state space methods*. Oxford University Press, Oxford, 2001.

L. R. Gerber, D. P. D. Master, and P. M. Kareiva. Grey whales and the value of monitoring data in implementing the U.S. endangered species act. *Conservation Biology*, 13:1215–1219, 1999.

P. D. Gilbert. *Brief user's guide: dynamic systems estimation*, 2009. URL `http://cran.r-project.org/web/packages/dse/vignettes/Guide.pdf`.

S. E. Hampton, M. D. Scheuerell, and D. E. Schindler. Coalescence in the Lake Washington story: interaction strengths in a planktonic food web. *Limnology and Oceanography*, 51(5):2042–2051, 2006.

A. C. Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge University Press, Cambridge, UK, 1989.

J. Helske. *KFAS: Kalman filter and smoothers for exponential family state space models.*, 2011. URL `http://CRAN.R-project.org/package=KFAS`. R package version 0.6.1.

R. Hinrichsen. Population viability analysis for several populations using multivariate state-space models. *Ecological Modelling*, 220(9-10):1197–1202, 2009.

R. Hinrichsen and E. E. Holmes. Using multivariate state-space models to study spatial structure and dynamics. In R. S. Cantrell, C. Cosner, and S. Ruan, editors, *Spatial Ecology*. CRC/Chapman Hall, 2009.

E. Holmes, E. Ward, K. Wills, NOAA, Seattle, and USA. *MARSS: multivariate autoregressive state-space modeling*, 2012. URL `http://CRAN.R-project.org/package=MARSS`. R package version 2.9.

E. E. Holmes. Estimating risks in declining populations with poor data. *Proceedings of the National Academy of Sciences of the United States of America*, 98(9):5072–5077, 2001.

E. E. Holmes. Derivation of the EM algorithm for constrained and unconstrained MARSS models. Technical report, Northwest Fisheries Science Center, Mathematical Biology Program, 2010.

E. E. Holmes, J. L. Sabo, S. V. Viscido, and W. F. Fagan. A statistical approach to quasi-extinction forecasting. *Ecology Letters*, 10(12):1182–1198, 2007.

J.-Y. Humbert, L. S. Mills, J. S. Horne, and B. Dennis. A better way to estimate population trends. *Oikos*, 118(12):1940–1946, 2009.

A. R. Ives, B. Dennis, K. L. Cottingham, and S. R. Carpenter. Estimating community stability and ecological interactions from time-series data. *Ecological Monographs*, 73(2):301–330, 2003.

D. Luethi, P. Erb, and S. Otziger. *FKF: fast Kalman filter*, 2012. URL `http://CRAN.R-project.org/package=FKF`. R package version 0.1.2.

K. Metaxoglou and A. Smith. Maximum likelihood estimation of VARMA models using a state-space EM algorithm. *Journal of Time Series Analysis*, 28: 666–685, 2007.

G. Petris. An R package for dynamic linear models. *Journal of Statistical Software*, 36(12):1–16, 2010. URL `http://www.jstatsoft.org/v36/i12/`.

J. Reddingius and P. den Boer. On the stabilization of animal numbers. Problems of testing. 1. Power estimates and observation errors. *Oecologia*, 78:1–8, 1989.

J. T. Schnute. A general framework for developing sequential fisheries models. *Canadian Journal of Fisheries and Aquatic Sciences*, 51:1676–1688, 1994.

R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4): 253–264, 1982.

D. S. Stoffer and K. D. Wall. Bootstrapping state-space models: Gaussian maximum likelihood estimation and the Kalman filter. *Journal of the American Statistical Association*, 86(416):1024–1033, 1991.

R. S. Tsay. *Analysis of financial time series*. Wiley Series in Probability and Statistics. Wiley, 2010.

E. J. Ward, H. Chirakkal, M. González-Suárez, D. Aurioles-Gamboa, E. E. Holmes, and L. Gerber. Inferring spatial structure from time-series data: using multivariate state-space models to detect metapopulation structure of California sea lions in the Gulf of California, Mexico. *Journal of Applied Ecology*, 1(47):47–56, 2010.

A. F. Zuur, R. J. Fryer, I. T. Jolliffe, R. Dekker, and J. J. Beukema. Estimating common trends in multivariate time series using dynamic factor analysis. *Environmetrics*, 14(7):665–685, 2003a.

A. F. Zuur, I. D. Tuck, and N. Bailey. Dynamic factor analysis to estimate common trends in fisheries time series. *Canadian Journal of Fisheries and Aquatic Sciences*, 60(5):542–552, 2003b.

Elizabeth E. Holmes
Northwest Fisheries Science Center
2725 Montlake Blvd E, Seattle WA 98112
USA
eli.holmes@noaa.gov

Eric J. Ward
Northwest Fisheries Science Center
2725 Montlake Blvd E, Seattle WA 98112
USA
eric.ward@noaa.gov

Kellie Wills
University of Washington
2725 Montlake Blvd E, Seattle WA 98112
USA
willsk@u.washington.edu