**Supplement: applying MAR(1) and MARSS to an ecological data set**

Here we step through an example of applying both MAR(1) and MARSS to an ecological data set, using R, a free open-source software environment for statistical computing and graphics (http://www.r-project.org/). The example R code is inset in the text, but also available as a block at the end of this supplement for those who would prefer to cut and paste the whole block into an R text editor before beginning the exercise.

**MAR1 R package**

The tools necessary to apply a basic MAR-1 model to time-series data have been built into the MAR1 package for R (http://www.r-project.org/). This package uses the tcltk package to create pop-up windows. On Mac OS X systems, installation of the Tcl/Tk 8.5.5 library for X11 may be necessary before the Tcl/Tk interface can successfully load. See http://cran.r-project.org/bin/macosx/tools/ for details.

From within the R console, the package can be installed with:

```
install.packages("MAR1", repos="http://cran.rstudio.com",
      dependencies=TRUE)
```

To load the MAR1 package in the current R session:

```
library(MAR1)
```

**Prepare the dataset**

The MAR1 package includes weekly marine plankton abundance data from the Western Channel Observatory (WCO) L4 station (www.pml.ac.uk/L4) as an example of a plankton time-series that is of appropriate temporal and taxonomic resolution for MAR application. See Eloire et al. (2010) and Southward et al. (2005) for L4 zooplankton and phytoplankton sample collection and processing methods. The L4 data subset included in the MAR1 package is a weekly time-series of sea surface temperature values (°C) and abundances for 15 zooplankton groups (individuals $m^{-3}$) and 3 phytoplankton groups (cells $ml^{-1}$). The plankton groups were created by summing the abundance values of taxonomically and ecologically similar taxa included in the raw L4 time-series for each time-step.

To take a look at the column heads and the first few lines of data, see a statistical summary of the dataset, and see additional details about the dataset in its associated help file:

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

?L4.AllDates
```

*Creating a time series with evenly spaced intervals*

We can see that the L4 dataset has a format common to time-series data: sampling dates are recorded in the first column and the measured values of each parameter are recorded in the following

columns. Columns containing information other than dates and the values of potential variables have been removed to simplify the dataset for MAR application. Because zooplankton and phytoplankton were not always sampled on the same date, we can see in the summary that the data include many missing (NA) values.

For MAR, the data need to be averaged into evenly-spaced time-steps. The MAR1 package includes a function for transforming datasets that can average data into monthly, yearly, weekly, or daily time-steps. To apply this function, the dataset must be arranged so that dates are in the first column and the numeric values of variables are in the following columns. Dates must include four-digit year values (yyyy) and some character separating year, month, and day. The order that year, month, and day are represented in the date values does not matter. The L4 dataset we are using conforms to these requirements, so the function can be directly applied. Monthly increments have been reported as an appropriate time-scale for the detection of plankton community interactions in other MAR studies, so we will average the L4 data into monthly time-steps.

To average the data into monthly increments and view the summary and dimensions (number of rows and columns) of the new time-series object:

```
L4.byMonth <- prepare.data(data=L4.AllDates, increment="month")

summary(L4.byMonth)
dim(L4.byMonth)
```

We see that the data have been averaged into 175 continuous time-steps, and all NA values have been eliminated in the process. Note that 175 is not the total number of time-steps in the dataset (which is 179 according to the dataset dimensions), but the number of time-steps which occur without an adjacent missing value in the series. This is the number of data points that will be used to build the model, as non-consecutive time-points will be thrown out during the analysis. A new column ‘contin’ has also been created, in which each run of equal values designates a continuous block of time-steps.

*Data transformations*

In plankton MAR studies, linear interpolation has sometimes been used to fill in small gaps in the time-series. However, when data consistently miss a certain time period (e.g., no winter sampling) investigators frequently just exclude those time periods from the time series (e.g. Ives et al 2003, Hampton et al 2006) and allow the MAR to skip estimations between non-consecutive data points. It is common practice to log-transform data in order to better approximate the non-linear relationships frequently present in ecological data (Ives et al 2003). If the data are log-transformed, zeros in the data need to be replaced with non-zero values. For some questions investigators also standardize the data to dimensionless units (Z-scores) so that model results can be directly compared among plankton groups (e.g., Hampton et al. 2006). The same function used to average the data into evenly spaced time-steps can perform these transformations as well. To see all arguments for the prepare.data function and possible values for each argument, and to see the help file for the function:

```
formals(prepare.data)

?prepare.data
```

If we decide to fill gaps in our time-series via linear interpolation, we can choose how large a gap we will allow to be filled with fill.gap. For example, if we set fill.gap= 2, gaps two months long or less will be interpolated, and larger gaps will be left in the time-series.

If we would like to log10-transform the data (by setting log=TRUE), zeros in the time-series must first be replaced. Zero values occurring in the time-series for a particular plankton group can be replaced with a random number between 0 and half the lowest non-zero value for that group (replace.0s =“rand.half”), or zeros can be eliminated by adding 1 to all values (replace.0s= “add.ones”).

There are two methods that can be used to Z-score the data as well. The first method (z.method=”standard”) standardizes each value by subtracting the overall mean abundance value for the group and dividing by the overall standard deviation for the group. The second method (z.method=”deseason”) is similar, except the means and standard deviations are month-specific so that mean seasonal trends are removed from the data-series. De-seasoning the abundance data could aid in the detection of interactions between plankton groups by dampening seasonal successions that relate to seasonally varying abiotic drivers.

We will apply the prepare.data function to the L4 data again, but in addition to averaging the data into monthly increments, we will transform the data so that gaps of one month are filled via linear interpolation, zeros are replaced with a random number between 0 and half the lowest value for the respective group, and the values are log10-transformed and then Z-scored using the “standard” method:

```
L4.mar1 <- prepare.data(data=L4.AllDates, increment="month",
      fill.gap=1, replace.0s="rand.half", log=T,
      z.method="standard")
```

R reports that two time gaps were filled, that the time-series now contains 179 continuous time-steps, and that the log and Z-score transformations were successful. The function that runs the MAR model requires that the data be arranged with the continuous time-block variable in the first column, dates in the second column, and the time-series for the variables in the following columns. Looking at the summary of the transformed dataset, we see that it is appropriately formatted for MAR application.

```
summary(L4.mar1)
```

To see how the Z-score method used affects the MAR results, we can create a second transformed dataset with the “deseason” Z-score method for comparison:

```
L4.mar2 <- prepare.data(data=L4.AllDates, increment="month",
      fill.gap=1, replace.0s="rand.half", log=T,
      z.method="deseason")

summary(L4.mar2)
```

**Build the MAR(1) model**

To see the arguments for the function that builds the MAR model and to see the help file for the function:

```
formals(run.mar)

?run.mar
```

The function run.mar allows the user to select which interactions will be included in the model. The variables argument designates whether variables will be included as variates (X in Equation 3) or covariates (U in Equation 3). The user can also set restrictions on interactions between the variables: interactions can be forced to be included in or excluded from the final model (the restrictions argument). By default, a random search for the best-fit model is performed (search="random") following Ives et al. (2003), but options for forward step and exhaustive searches are also available (see `?run.mar`). The function does a 500 iteration bootstrap of the best-fit model coefficients by default, and the number of iterations can be changed (boot=*n*), or bootstrapping can be skipped altogether (boot=FALSE). The top 10 models (i.e., lowest AIC models, including the best-fit model) are retained in the output by default, and this number also can be changed (ntop=*n*) or excluded (ntop=FALSE). All results of the MAR analysis are stored as a list object within R (see `?run.mar` for details on the structure and components of the list object). If export is set to TRUE or to a quoted name (e.g., export= "FolderName"), a folder will be created in the current working directory and all components of the result list will be saved as comma-delimited (.csv) files in that folder. A MAR result list object can also be exported later using the function `export.MAR()` (see ?export.MAR).

For the L4 data, one might be interested in using MAR to assess whether there are detectible interactions between invertebrate predators and copepods. It is thought that cnidarians and chaetognaths may have important predatory effects on calanoid copepods in the English Channel (Irigioen & Harris 2003, Bonnet et al. 2010, Eloire et al. 2010). To investigate this hypothesis with MAR, we will build a model that includes the cnidarian, chaetognath, large calanoid, and small calanoid zooplankton groups and the diatom and dinoflagellate phytoplankton groups as variates, and sea surface temperature as a covariate.

To initiate the analysis:

```
run1 <- run.mar(data=L4.mar1)
```

Because no values were provided for the variables argument, a window that allows the user to select variates and covariates with toggle buttons appears, along with instructions in the R Console. Click the buttons next to cnidarian, chaetognath, calanoid.lg, calanoid.sm, diatom, and dino once to select them as variates, and the button next to surface.temp twice to select it as a covariate. Clicking a button three times resets it to 'not included.' After clicking Done, another window appears to allow restrictions to be set on interactions with toggle buttons. To see what the MAR model will look like when we don't set any restrictions, click Done. A progress report should show in the console while the function is working, and all results are stored in the 'run1' object.

The analysis can also be run without using the windows to select the variables and set restrictions:

```
myvar <- c(0,0,1,0,1,0,0,0,0,1,1,0,0,0,1,1,0,0,0,0,2)
myres <- matrix(0.5, nrow=length(which(myvar==1)),
      ncol=length(which(myvar!=0)))
```

```
run1 <- run.mar(data=L4.mar1, variables=myvar,
      restrictions=myres)
```

To print the main results, see a summary of the results, and plot the model:

```
run1
summary(run1)
plot(run1)
```

In the plot of interaction strengths, bars extending to the right and left of the dotted lines represent positive and negative interactions, respectively. Interactions in the best-fit model that were excluded by bootstrapping are plotted as lighter, hatched bars. It is apparent that the analysis detected mostly positive interactions. All variates exhibit positive density dependence along the diagonal of the B-matrix, and large calanoid copepods and diatoms appear to have positive effects on the abundances of nearly all the other variates. Surface temperature, the only covariate included in the model, is shown to have negative effects on diatoms and large and small calanoids. Several of the interactions, including the positive effects of large calanoids on both phytoplankton groups and the negative effects of temperature on the variates, are ecologically unlikely. To assess whether the interaction patterns in this model might have been influenced by the detection of seasonal successions in the time-series, we can build another model with our "deseason" Z-scored dataset:

```
run2 <- run.mar(data=L4.mar2, variables=run1, restrictions=run1)
```

Since we are using a dataset with identical formatting to the one used in building the first model (i.e., same column order and column headers) and we would like to use the same variables and restrictions as those set in the first model, we can simply set the variables and restrictions arguments of the function equal to the 'run1' object.

To compare both models in one plot and create a legend for the plot:

```
plot(run1, run2, legend=T)
```

We can see that the second model for the de-seasoned data includes notably fewer interactions than the first model. None of the negative temperature effects were detected in the second model, and the only positive effect of large calanoids on any variate is their own density dependence. The second model also includes a plausible negative effect of large calanoids on diatoms that was not detected in the first model. An unlikely positive effect of cnidarians on chaetognaths was detected in both models. If we wanted to recreate the second, de-seasoned data model, but with the interaction of cnidarians on chaetognaths excluded, we can set that restriction by toggling the corresponding button (first column, second row) to 0 in the Specify Restrictions window before clicking Done:

```
run3 <- run.mar(data=L4.mar2, variables=run2)
```

To plot all three models:

```
plot(run1, run2, run3, legend=T)
```

In the plot, it is apparent that setting this cnidarian-chaetognath restriction (indicated by a red dot) had little effect on the results.

To compare the best-fit model to the other top models retained in the analysis, we can plot the $top.bestfit component of the result list object:

```
dev.new()
plot(run3$top.bestfit)
```

Here, the best-fit model is represented by the top-most set of bars, and nine other low-AIC models are plotted below it in order of increasing AIC. To view a histogram of the AIC values for the best models:

```
hist(run3$top.bestfit)
```

The AIC value for the selected best-fit model is denoted by the blue star along the bottom of the plot.

**Building a state-space MAR model (MARSS)**

A state-space MAR model is a MAR(1) model with an observation process:

$$\begin{aligned} \mathbf{X}_t &= \mathbf{A} + \mathbf{B}\mathbf{X}_{t-1} + \mathbf{C}\mathbf{U}_t + \mathbf{E}_t \\ \mathbf{Y}_t &= \mathbf{X}_t + \mathbf{V}_t \end{aligned} \qquad \text{S1}$$

The underlying community dynamics are described by the MAR(1) model (**X**) but the true values of **X** are "hidden", and we only observe $\mathbf{Y}_t$ which is $\mathbf{X}_t$ plus some error. State-space MAR (MARSS) might be particularly helpful with this data set, since we anticipate that the observation error is a real issue in this relatively open marine system.

The MARSS package (Holmes et al. 2012) is available in R for fitting state-space multivariate autoregressive models, and we can use it to fit a state-space version of the MAR(1) models in the examples above. The MAR1 package has a function `ss.mar1()` which will allow you to easily fit the model described in eqn. S1: our observations are $\mathbf{X}_t$ plus error and we will treat our covariates $\mathbf{U}_t$, in this case surface temperature, as known without observation error. Although in this example we use the simple observation process described by eqn. S2, the MARSS package will allow you to model much more general observation processes and can allow one to model, for example, situations with multiple measurement sites, measurement equipment or sampling designs that change over time, sampling sites with different levels of missing values, or covariates that are measured with error and in multiple locations. See the MARSS User Guide on the MARSS CRAN webpage (http://cran.r-project.org/web/packages/MARSS) for a discussion of analysis of species interactions using more complicated observation models.

The ss.mar1() function uses an aggregated dataset as input. This is a dataset already transformed by `prepare.data()`. It also needs to have the MAR model **B** and **C** matrices specified. One way to do this is to pass in a MAR fitted model object as output by `run.mar()` and `ss.mar1()` will use the bestfit B and C. (Note: If MARSS is not installed, you will receive an error message. Use `install.packages("MARSS", repos="http://cran.rstudio.com", dependencies=TRUE)` to install the package.)

```
ss.run3 <- ss.mar1(L4.mar2, MAR.obj=run3)
```

We can compare the **B** matrices from the state-space MAR and non-state-space MAR by looking at

```
ss.run3$B
run3$bestfit$B

ss.run3.plot <- list(restrictions.set=run3$restrictions.set,
      bestfit=list(B=ss.run3$B,C=ss.run3$C), bootstrap=NULL)
class(ss.run3.plot) <- "MAR"

dev.new()
plot(ss.run3.plot,run3,legend=T)
```

The first thing that is apparent is that the interaction strengths along the diagonal are larger, potentially indicating that the observation error is now being more appropriately accounted for. Observation error tends to make time-series look like white-noise and thus moves the diagonal terms towards 0. We also see that the off-diagonal terms are farther from 0 with the state-space MAR model. This is also suggestive of better partitioning of the variance. Observation error tends to make species look less correlated and thus their interactions strengths are closer to 0. Overall however, the direction of the interactions are largely concordant with the results from the MAR(1) model.

The L4.mar2 dataset had gaps of 1 filled by linear interpolation. A state-space MAR models can handle missing values fine because it incorporates an observation model. We can compare the **B** estimates if we do not do linear interpolation.

```
# L4 data aggregated by month with gaps not filled

L4.mar3 <- prepare.data(data=L4.AllDates, increment="month",
      fill.gap=0, replace.0s="rand.half", log=TRUE,
      z.method="deseason")

ss.run3.no.interp <- ss.mar1(L4.mar3, MAR.obj=run3)

ss.run3.no.interp$B
```

These models take a long time to fit because estimating both the observation and process variance matrices is difficult. We can also fix the observation variance by passing in element model into `ss.mar1()`. The observation variance matrix is termed R, and we will set the observation variance of the zooplankton to be 0.1 and the dinoflagellates and diatoms to be 0.05. The model now fits much faster.

```
# Set observation variance matrix instead of estimating it

R <- diag(c(0.1,0.1,0.1,0.1,0.05,0.05))

ss.run3.fixed.R <- ss.mar1(L4.mar3, MAR.obj=run3,
      model=list(R=R))

ss.run3.fixed.R$B
```

**Code from this exercise**

Some readers may find it easiest to just cut and paste this whole block into an R text editor before working through the exercise.

```
# Load package

library(MAR1)

# Load data in MAR1 package

data(L4.AllDates)
head(L4.AllDates)
summary(L4.AllDates)

?L4.AllDates

# Begin transformations – average into monthly time steps

L4.byMonth <- prepare.data(data=L4.AllDates, increment="month")

summary(L4.byMonth)
dim(L4.byMonth)

# See all arguments for prepare.data function

formals(prepare.data)
?prepare.data

# Fill gaps with linear interpolation, perform log and z-score
      transform

L4.mar1 <- prepare.data(data=L4.AllDates, increment="month",
      fill.gap=1, replace.0s="rand.half", log=T,
      z.method="standard")

# Look at summary of transformed data set

summary(L4.mar1)

# Create another dataset using the de-seasoning z-score method

L4.mar2 <- prepare.data(data=L4.AllDates, increment="month",
      fill.gap=1, replace.0s="rand.half", log=T,
```

```
      z.method="deseason")

summary(L4.mar2)

# Look at the arguments available in run.mar

formals(run.mar)
?run.mar

# Initiate MAR(1) analysis – if variables is blank, pop-up
      windows will guide user through setting the interaction
      matrix

run1 <- run.mar(data=L4.mar1)

# Summarize and plot main results

run1
summary(run1)
plot(run1)

# Build the same model without using the pop-up windows

myvar <- c(0,0,1,0,1,0,0,0,0,1,1,0,0,0,1,1,0,0,0,0,2)
myres <- matrix(0.5, nrow=length(which(myvar==1)),
      ncol=length(which(myvar!=0)))

run1 <- run.mar(data=L4.mar1, variables=myvar,
      restrictions=myres)

# Build another model using the de-seasoned data set

run2<-run.mar(data=L4.mar2, variables=run1, restrictions=run1)

# Compare results of 1st and 2nd model in a plot

plot(run1, run2, legend=T)

# Run a 3rd model – this time you can toggle off one of the
      biologically implausible interactions (cnidarians on
      chaetognaths)

run3 <- run.mar(data=L4.mar2, variables=run2)

# Compare results of all three models in a plot

plot(run1, run2, run3, legend=T)

# For top model (lowest AIC) from 3rd model, compare to other top
      models in a plot

dev.new()
plot(run3$top.bestfit)

# Compare AIC values of top models using a histogram
```

```
hist(run3$top.bestfit)

# Compare state-space MAR

ss.run3 <- ss.mar1(L4.mar2, MAR.obj=run3)

ss.run3$B
run3$bestfit$B

# Use a state-space model with L4 data aggregated by month with
      gaps not filled

L4.mar3 <- prepare.data(data=L4.AllDates, increment="month",
      fill.gap=0, replace.0s="rand.half", log=TRUE,
      z.method="deseason")

ss.run3.no.interp <- ss.mar1(L4.mar3, MAR.obj=run3)

ss.run3.no.interp$B

# Use a state-space model with a fixed observation variance

R <- diag(c(.1,.1,.1,.1,.05,.05))

ss.run3.fixed.R <- ss.mar1(L4.mar3, MAR.obj=run3,
      model=list(R=R))

ss.run3.fixed.R$B
```