MARSS Quick Start Guide

The default MARSS model (`form="marxss"`) is written as follows:

$$\begin{gathered}\mathbf{x}_t = \mathbf{B}_t\mathbf{x}_{t-1} + \mathbf{u}_t + \mathbf{C}_t\mathbf{c}_t + \mathbf{w}_t, \text{ where } \mathbf{w}_t \sim \text{MVN}(0, \mathbf{Q}_t) \\ \mathbf{y}_t = \mathbf{Z}_t\mathbf{x}_t + \mathbf{a}_t + \mathbf{D}_t\mathbf{d}_t + \mathbf{v}_t, \text{ where } \mathbf{v}_t \sim \text{MVN}(0, \mathbf{R}_t) \\ \mathbf{x}_1 \sim \text{MVN}(\boldsymbol{\pi}, \boldsymbol{\Lambda}) \text{ or } \mathbf{x}_0 \sim \text{MVN}(\boldsymbol{\pi}, \boldsymbol{\Lambda})\end{gathered} \tag{1}$$

c and **d** are inputs (aka, exogenous variables or covariates or indicator variables) and must have no missing values. They are not treated as 'data' in the likelihood but as inputs. The MARSS package is designed to handle linear constraints within the parameter matrices (the **B**, **u**, **C**, **Q**, **Z**, **a**, **D**, **R**, $\boldsymbol{\pi}$, and $\boldsymbol{\Lambda}$)). Linear constraint means you can write the elements of the matrix as a linear equation of all the other elements, although typically each matrix element is just a fixed or estimated value.

Model specification is based one-to-one on the MARSS equation above. Many of the parameter elements will be fixed or zero and others will be shared (equal) within a matrix. The following shows an example of a mean-reverting random walk model with three observation time series:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_t = \begin{bmatrix} b & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{t-1} + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_t, \quad \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_t \sim \text{MVN}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix} \right), \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_0 \sim \text{MVN}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}_t = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_t + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_t, \quad \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_t \sim MVN\left(\begin{bmatrix} a_1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} r_{11} & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{bmatrix} \right)$$

To fit this with MARSS, we translate this model as written mathematically into an equivalent matrix (or array if time-varying) in R. Matrices that combine fixed and estimated values are specified using a list matrix with numerical values for fixed values and character names for the estimated values. The following shows how to specify the model above.

```
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(1,0,1,1,1,0),3,2)
A1=matrix(list("a1",0,0),3,1)
R1=matrix(list("r11",0,0,0,"r",0,0,0,"r"),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,tinitx=0)
```

Try printing these out and you will see the one-to-one correspondence between the model in R and the math version of the model. BTW, your model can have simple linear constraints within the matrix elements. In that case, your matrix element might have $1 + 2a + 3b$ and you specify this using `"1+2*a+3*b"`, instead of simply a numerical value or name of an estimated value.

For `form="marxss"` (the default), matrix names in the model list must be B, U, C, c, Q, Z, A, D, d, R, x0, and V0, just like in equation (1). The `tinitx` element tells MARSS whether the initial state for x is at $t = 1$ (`tinitx=1`) or $t = 0$ (`tinitx=0`). The data must be entered as a $n \times T$ matrix; a dataframe is not a matrix nor is a vector nor is a time-series object. MARSS has a number of text shortcuts for common parameter forms, such as "diagonal and unequal"; see the User Guide for the possible shortcuts. You can leave off matrix names and the defaults will be used. Type ?MARSS.marxss to see the defaults for `form="marxss"`.

The call to MARSS is

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

The `R`, `Q` and `V0` variances can be set to zero to specify partially deterministic systems. This allows you to write MAR-p models in MARSS form for example. See the User Guide for examples. Type `?MARSS` at the command line for more info.

Important

- Specification of a properly constrained model with a unique solution is the responsibility of the user because MARSS has no way to tell if you have specified an insufficiently constrained model.

- The code in the MARSS package is not particularly fast and EM algorithms are famously slow. You can try method="BFGS" and see if that is faster. For some models, it will be much faster and for others, much slower.

Time-varying parameters and inputs

The default model form (`form="marxss"`) allows you to pass in an array of T matrices for a time-varying parameter (T is the number of time-steps in your data and is the 3rd dimension in the array):

$$\begin{aligned} \mathbf{x}_t &= \mathbf{B}_t\mathbf{x}_{t-1} + \mathbf{u}_t + \mathbf{C}_t\mathbf{c}_t + \mathbf{w}_t, \quad \mathbf{W}_t \sim \text{MVN}(0, \mathbf{Q}_t) \\ \mathbf{y}_t &= \mathbf{Z}_t\mathbf{x}_t + \mathbf{a}_t + \mathbf{D}_t\mathbf{d}_t + \mathbf{v}_t, \quad \mathbf{V}_t \sim \text{MVN}(0, \mathbf{R}_t) \\ \mathbf{x}_{t_0} &\sim \text{MVN}(\boldsymbol{\pi}, \boldsymbol{\Lambda}) \end{aligned} \tag{2}$$

Zeros are allowed on the diagonals of $\mathbf{Q}$, $\mathbf{R}$ and $\boldsymbol{\Lambda}$. NOTE(!!), the time indexing. Make sure you enter your arrays such that the right parameter (or input) at time t lines up with $\mathbf{x}_t$, e.g. it is common for state equations to have $\mathbf{B}_{t-1}$ lined up with $\mathbf{x}_t$ so you might need to enter the $\mathbf{B}$ array such that your $\mathbf{B}_{t-1}$ is entered at `Bt[,,t]` in the R code.

The length of the 3rd dimension must be the same as your data. For example, say in your mean-reverting random walk model (the example on the first page) you wanted $\mathbf{B}(2,2)$ to be one value before $t = 20$ and another value after but $\mathbf{B}(1,1)$ to be time constant. You can pass in the following:

```
TT=dim(data)[2]
B1=array(list(),dim=c(2,2,TT))
B1[,,1:20]=matrix(list("b",0,0,"b_1"),2,2)
B1[,,21:TT]=   matrix(list("b",0,0,"b_2"),2,2)
```

Notice the specification is one-to-one to your $\mathbf{B}_t$ matrices on paper.

Inputs are specified in exactly the same manner. $\mathbf{C}$ and $\mathbf{D}$ are the estimated parameters and $\mathbf{c}$ and $\mathbf{d}$ are the inputs. Let's say you have temperature data and you want to include a linear effect of temperature that is different for each $\mathbf{x}$ time series:

```
C1=matrix(c("temp1","temp2"),2,1)
model.list=list(B=B1,U=U1,C=C1,c=temp,Q=Q1,Z=Z1,A=A1,R=R1,x0=pi1,V0=V1,tinitx=0)
```

If you want a factor effect, then you'll need to recode your factor as a matrix with T columns and a row for each factor. Then you have 0 or 1 if that factor applies in time period t. $\mathbf{C}$ then has a column for each estimated factor effect.

Showing the model fits and getting the parameters

There `print`, `coef` and `residuals` functions for marssMLE objects. However, a good place to start is to look at `?print.MARSS`. This will show you how to get a lot of standard output from your fitted model objects (marssMLE objects). It will also show you alternative ways to get that output and where that output is stored in the marssMLE object. Type `?coef.MARSS` to see the different formats for displaying the estimated parameters.

The full time-varying model used in the MARSS EM algorithm

Expectation-Maximization algorithms for unconstrained MARSS models have been around for many years. What makes the EM algorithm in MARSS different is that it is a constrained algorithm. In mathematical form, the model that is being fit with the package is

$$\begin{aligned}
\mathbf{x}_t &= (\mathbf{x}_{t-1}^\top \otimes \mathbf{I}_m)\,\mathrm{vec}(\mathbf{B}_t) + (\mathbf{u}_t^\top \otimes \mathbf{I}_m)\,\mathrm{vec}(\mathbf{U}_t) + \mathbf{w}_t, \quad \mathbf{W}_t \sim \mathrm{MVN}(0, \mathbf{Q}_t) \\
\mathbf{y}_t &= (\mathbf{x}_t^\top \otimes \mathbf{I}_n)\,\mathrm{vec}(\mathbf{Z}_t) + (\mathbf{a}_t^\top \otimes \mathbf{I}_n)\,\mathrm{vec}(\mathbf{A}_t) + \mathbf{v}_t, \quad \mathbf{V}_t \sim \mathrm{MVN}(0, \mathbf{R}_t) \\
\mathbf{x}_{t_0} &= \boldsymbol{\pi} + \mathbf{F} \ll, \mathbf{L} \sim \mathrm{MVN}(0, \boldsymbol{\Lambda})
\end{aligned} \tag{3}$$

Each model parameter matrix, $\mathbf{B}_t$, $\mathbf{U}_t$, $\mathbf{Q}_t$, $\mathbf{Z}_t$, $\mathbf{A}_t$, and $\mathbf{R}_t$, is written as a time-varying linear model, $\mathbf{f}_t + \mathbf{D}_t\mathbf{m}$, where $\mathbf{f}$ and $\mathbf{D}$ are fully-known (not estimated and no missing values) and $\mathbf{m}$ is a column vector

of the estimates elements of the parameter matrix:

$$\begin{gathered}
\text{vec}(\mathbf{B}_t) = \mathbf{f}_{t,b} + \mathbf{D}_{t,b}\boldsymbol{\beta} \quad \text{vec}(\mathbf{U}_t) = \mathbf{f}_{t,u} + \mathbf{D}_{t,u}\boldsymbol{\upsilon} \quad \text{vec}(\mathbf{Q}_t) = \mathbf{f}_{t,q} + \mathbf{D}_{t,q}\mathbf{q} \\
\text{vec}(\mathbf{Z}_t) = \mathbf{f}_{t,z} + \mathbf{D}_{t,z}\boldsymbol{\zeta} \quad \text{vec}(\mathbf{A}_t) = \mathbf{f}_{t,a} + \mathbf{D}_{t,a}\boldsymbol{\alpha} \quad \text{vec}(\mathbf{R}_t) = \mathbf{f}_{t,r} + \mathbf{D}_{t,r}\mathbf{r} \\
\text{vec}(\boldsymbol{\Lambda}) = \mathbf{f}_{\lambda} + \mathbf{D}_{\lambda}\boldsymbol{\lambda} \quad \text{vec}(\boldsymbol{\pi}) = \mathbf{f}_{\pi} + \mathbf{D}_{\pi}\mathbf{p}
\end{gathered}$$

The internal MARSS model specification (form=marss) is a list with the $\mathbf{f}_t$ and $\mathbf{D}_t$ matrices for each parameter. The output from fitting are the vectors, $\boldsymbol{\beta}$, $\boldsymbol{\upsilon}$, etc. The trick is to rewrite the user's linear multivariate problem into the general form (equation 3). MARSS does this using functions that take more familiar arguments as input and then constructs the $\mathbf{f}_t$ and $\mathbf{D}_t$ matrices. Because the $\mathbf{f}_t$ and $\mathbf{D}_t$ can be whatever the user wants (assuming they are the right shape), this allows users to include covariates, trends (linear, sinusoidal, etc) or indicator variables in a variety of ways. It also means that terms like $1 + b + 2c$ can appear in the parameter matrices.

Although the above form looks unusual, it is equivalent to the commonly seen form but leads to a log-likelihood function where all terms have form $\mathbf{Mm}$, where $\mathbf{M}$ is a matrix and $\mathbf{m}$ is a column vector of only the different estimated values. This makes it easy to do the partial differentiation with respect to $\mathbf{m}$ necessary for the EM algorithm and as a result, easy to impose linear constraints and structure on the elements in a parameter matrix. See Holmes (2012) for a discussion of the general form of MARSS models being fit by the EM algorithm in the MARSS package.

Need more information?

The MARSS User Guide starts with some tutorials on MARSS models and walks through a bunch of simple examples so you get used to framing multivariate time-series models in matrix form. Then it has a series of vignettes: how to write AR(p) models in state-space form, dynamic linear models (regression models where the regression parameters are AR(p)), multivariate regression models with regression parameters that are time-varying and enter the non-AR part of your model or the AR part, detecting breakpoints using state-space models, and dynamic factor analysis. All of these can be written in MARSS form. It also has a series of case studies of analyzing multivariate biological data.