9

Dynamic factor analysis (DFA)

9.1 Overview

In this chapter, we use MARSS to do dynamic factor analysis (DFA), which allows us to look for a set of common underlying trends among a relatively large set of time series (Harvey, 1989, sec. 8.5). See also Zuur et al. (2003) which shows a number of examples of DFA applied to fisheries catch data and densities of zoobenthos. We will walk through some examples to show you the math behind DFA, and then in section 9.4, we will show a short-cut for doing a DFA with MARSS using `form="dfa"`.

DFA is conceptually different than what we have been doing in the previous applications. Here we are trying to explain temporal variation in a set of n observed time series using linear combinations of a set of m hidden random walks, where $m << n$. A DFA model is a type of MARSS model with the following structure:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{x}_{t-1} + \mathbf{w}_t \text{ where } \mathbf{w}_t \sim \text{MVN}(0, \mathbf{Q}) \\ \mathbf{y}_t &= \mathbf{Z}\mathbf{x}_t + \mathbf{a} + \mathbf{v}_t \text{ where } \mathbf{v}_t \sim \text{MVN}(0, \mathbf{R}) \\ \mathbf{x}_0 &\sim \text{MVN}(\boldsymbol{\pi}, \boldsymbol{\Lambda}) \end{aligned} \tag{9.1}$$

The general idea is that the observations ($\mathbf{y}$) are modeled as a linear combination of hidden trends ($\mathbf{x}$) and factor loadings ($\mathbf{Z}$) plus some offsets ($\mathbf{a}$). The DFA model in Equation 9.1 and the standard MARSS model in Equation 1.1 are equivalent–we have simply set the matrix $\mathbf{B}$ equal to an $m \times m$ identity matrix (i.e., a diagonal matrix with 1's on the diagonal and 0's elsewhere) and the vector $\mathbf{u} = \mathbf{0}$.

Type `RShowDoc("Chapter_DFA.R",package="MARSS")` at the R command line to open a file with all the code for the examples in this chapter.

9.1.1 Writing out a DFA model in MARSS form

Imagine a case where we had a data set with six observed time series ($n = 6$) and we want to fit a model with three hidden trends ($m = 3$). If we write out our DFA model in MARSS matrix form (ignoring the error structures and initial conditions for now), it would look like this:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_{t-1} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}_t$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix}_t = \begin{bmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \\ z_{31} & z_{32} & z_{33} \\ z_{41} & z_{42} & z_{43} \\ z_{51} & z_{52} & z_{53} \\ z_{61} & z_{62} & z_{63} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_t + \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix}_t . \tag{9.2}$$

The process errors of the hidden trends would be

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}_t \sim \text{MVN}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \right), \tag{9.3}$$

and the observation errors would be

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix}_t \sim \text{MVN}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} & r_{16} \\ r_{12} & r_{22} & r_{23} & r_{24} & r_{25} & r_{26} \\ r_{13} & r_{23} & r_{33} & r_{34} & r_{35} & r_{36} \\ r_{14} & r_{24} & r_{34} & r_{44} & r_{45} & r_{46} \\ r_{15} & r_{25} & r_{35} & r_{45} & r_{55} & r_{56} \\ r_{16} & r_{26} & r_{36} & r_{46} & r_{56} & r_{66} \end{bmatrix} \right). \tag{9.4}$$

9.1.2 Constraints to ensure identifiability

If $\mathbf{Z}$, $\mathbf{a}$, and $\mathbf{Q}$ in Equation 9.1 are not constrained, then the DFA model above is unidentifiable (Harvey, 1989, sec 4.4). Harvey (1989, sec. 8.5.1) suggests the following parameter constraints to make the model identifiable:

- in the first $m-1$ rows of $\mathbf{Z}$, the z-value in the j-th column and i-th row is set to zero if $j > i$;
- $\mathbf{a}$ is constrained so that the first m values are set to zero; and
- $\mathbf{Q}$ is set equal to the identity matrix ($\mathbf{I}_m$).

Zuur et al. (2003), however, found that with Harvey's second constraint, the EM algorithm is not particularly robust, and it takes a long time to converge. Zuur et al. found that the EM estimates are much better behaved if you instead constrain each of the time series in $\mathbf{x}$ to have a mean of zero across $t = 1$ to T. To do so, they replaced the estimates of the hidden states, $\mathbf{x}_t^T$, coming out of the Kalman smoother with $\mathbf{x}_t^T - \bar{\mathbf{x}}$ for $t = 1$ to T, where *NOTE*: $\bar{\mathbf{x}}$ is the mean of $\mathbf{x}_t$ across t. With this approach, you estimate all of the $\mathbf{a}$ elements, which represent the average level of $\mathbf{y}_t$ relative to $\mathbf{Z}(\mathbf{x}_t - \bar{\mathbf{x}})$. We found that demeaning the $\mathbf{x}_t^T$ in this way can cause the EM algorithm to have errors (decline in log-likelihood). Instead, we demean our data, and fix all elements of $\mathbf{a}$ to zero.

Using these constraints, the DFA model in Equation 12.3 becomes

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_{t-1} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}_t$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix}_t = \begin{bmatrix} z_{11} & 0 & 0 \\ z_{21} & z_{22} & 0 \\ z_{31} & z_{32} & z_{33} \\ z_{41} & z_{42} & z_{43} \\ z_{51} & z_{52} & z_{53} \\ z_{61} & z_{62} & z_{63} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_t + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix}_t . \tag{9.5}$$

The process errors of the hidden trends in Equation 9.3 would then become

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}_t \sim \text{MVN} \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right), \tag{9.6}$$

but the observation errors in Equation 9.4 would stay the same, such that

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix}_t \sim \text{MVN} \left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} & r_{16} \\ r_{12} & r_{22} & r_{23} & r_{24} & r_{25} & r_{26} \\ r_{13} & r_{23} & r_{33} & r_{34} & r_{35} & r_{36} \\ r_{14} & r_{24} & r_{34} & r_{44} & r_{45} & r_{46} \\ r_{15} & r_{25} & r_{35} & r_{45} & r_{55} & r_{56} \\ r_{16} & r_{26} & r_{36} & r_{46} & r_{56} & r_{66} \end{bmatrix} \right). \tag{9.7}$$

To complete our model, we still need the final form for the initial conditions of the state. Following Zuur et al. (2003), we set the initial state vector ($\mathbf{x}_0$) to have zero mean and a diagonal variance-covariance matrix with large variances, such that

$$\mathbf{x}_0 \sim \text{MVN} \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix} \right). \tag{9.8}$$

9.2 The data

We will analyze some of the Lake Washington plankton data included in the MARSS package. This dataset includes 33 years of monthly counts for 13 plankton species along with data on water temperature, total phosphorous (TP), and pH. First, we load the data and then extract a subset of columns corresponding to the phytoplankton species only. For the purpose of speeding up model fitting times and to limit our analysis to years with no missing covariate data, we will only examine 10 years of data (1980-1989).

```
# load the data (there are 3 datasets contained here)
data(lakeWAplankton)
# we want lakeWAplanktonTrans, which has been transformed
# so the 0s are replaced with NAs and the data z-scored
dat = lakeWAplanktonTrans
# use only the 10 years from 1980-1989
plankdat = dat[dat[,"Year"]>=1980 & dat[,"Year"]<1990,]
# create vector of phytoplankton group names
phytoplankton = c("Cryptomonas", "Diatoms", "Greens",
                  "Unicells", "Other.algae")
# get only the phytoplankton
dat.spp.1980 = plankdat[,phytoplankton]
```

Next, we transpose the data and calculate the number of time series and their length.

```
# transpose data so time goes across columns
dat.spp.1980 = t(dat.spp.1980)
# get number of time series
N.ts = dim(dat.spp.1980)[1]
# get length of time series
TT = dim(dat.spp.1980)[2]
```

It is normal in this type of analysis to standardize each time series by first subtracting its mean and then dividing by its standard deviation (i.e., create a z-score $\mathbf{y}_t^*$ with mean = 0 and SD = 1), such that

$$\mathbf{y}_t^* = \Sigma^{-1}(\mathbf{y}_t - \bar{\mathbf{y}}),$$

Σ is a diagonal matrix with the standard deviations of each time series along the diagonal, and $\bar{\mathbf{y}}$ is a vector of the means. In R, this can be done as follows

```
Sigma = sqrt(apply(dat.spp.1980, 1, var, na.rm=TRUE))
y.bar = apply(dat.spp.1980, 1, mean, na.rm=TRUE)
dat.z = (dat.spp.1980 - y.bar) * (1/Sigma)
rownames(dat.z) = rownames(dat.spp.1980)
```

Figure 9.1 shows time series of Lake Washington phytoplankton data following z-score transformation.

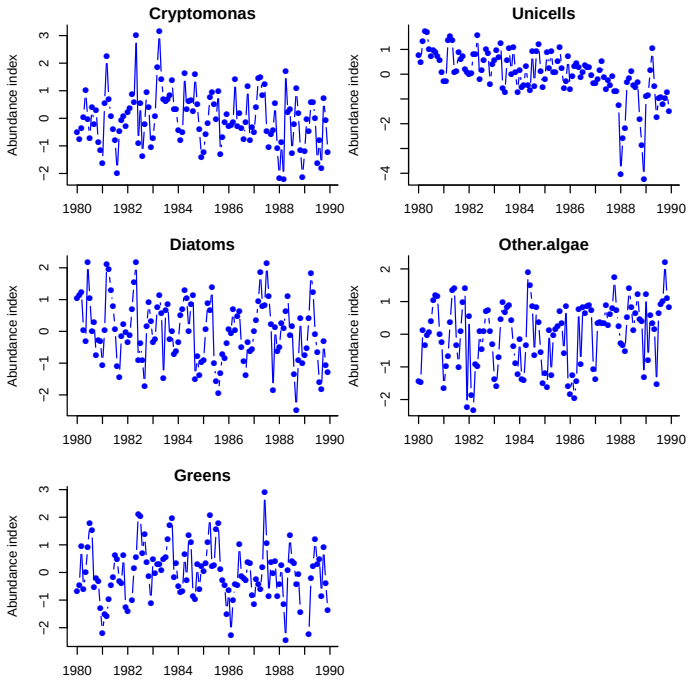

Fig. 9.1. Time series of Lake Washington phytoplankton data following z-score transformation.

9.3 Setting up the model in MARSS

As we have seen in other cases, setting up the model structure for MARSS requires that the parameter matrices have a one-to-one correspondence to the model as you would write it on paper (i.e., Equations 9.5 through 9.8). If a parameter matrix has a combination of fixed and estimated values, then you specify that using `matrix(list(), nrow, ncol)`. This is a matrix of class list and allows you to combine numeric and character values in a single matrix. MARSS recognizes the numeric values as fixed values and the character values as estimated values.

This is how we set up **Z** for MARSS, assuming a model with 5 observed time series and 3 hidden trends:

```
Z.vals = list(
"z11",  0  ,  0  ,
"z21","z22",  0  ,
"z31","z32","z33",
"z41","z42","z43",
```

```
"z51","z52","z53")
Z = matrix(Z.vals, nrow=N.ts, ncol=3, byrow=TRUE)
```

When specifying the list values, spacing and carriage returns were added to help show the correspondence with the **Z** matrix in Equation 9.3. If you print **Z** (at the R command line), you will see that it is a matrix with character values (the estimated elements) and numeric values (the fixed 0's).

```
print(Z)
```

```
     [,1]  [,2]  [,3]
[1,] "z11" 0     0
[2,] "z21" "z22" 0
[3,] "z31" "z32" "z33"
[4,] "z41" "z42" "z43"
[5,] "z51" "z52" "z53"
```

Notice that the 0's do not have quotes around them. If they did, it would mean the "0" is a character value and would be interpreted as the name of a parameter to be estimated rather than a fixed numeric value.

The **Q** and **B** matrices are both set equal to the identity matrix using diag().

```
Q = B = diag(1,3)
```

For our first analysis, we will assume that each time series of phytoplankton has a different observation variance, but that there is no covariance among time series. Thus, **R** should be a diagonal matrix that looks like:

$$\begin{bmatrix} r_{11} & 0 & 0 & 0 & 0 \\ 0 & r_{22} & 0 & 0 & 0 \\ 0 & 0 & r_{33} & 0 & 0 \\ 0 & 0 & 0 & r_{44} & 0 \\ 0 & 0 & 0 & 0 & r_{55} \end{bmatrix},$$

and each of the $r_{i,i}$ elements is a different parameter to be estimated. We can also specify this **R** structure using a list matrix as follows:

```
R.vals = list(
"r11",0,0,0,0,
0,"r22",0,0,0,
0,0,"r33",0,0,
0,0,0,"r44",0,
0,0,0,0,"r55")
R = matrix(R.vals, nrow=N.ts, ncol=N.ts, byrow=TRUE)
```

You can print **R** at the R command line to see what it looks like:

```
print(R)
```

```
     [,1]  [,2]  [,3]  [,4]  [,5]
[1,] "r11" 0     0     0     0
[2,] 0     "r22" 0     0     0
[3,] 0     0     "r33" 0     0
[4,] 0     0     0     "r44" 0
[5,] 0     0     0     0     "r55"
```

This form of variance-covariance matrix is commonly used, and therefore MARSS has a built-in shorthand for this structure. Alternatively, we could simply type:

```
R = "diagonal and unequal"
```

As mentioned in earlier chapters, there are other shorthand notations for many of the common parameter structures. Type ?MARSS at the R command line to see a list of the shorthand options for each parameter vector/matrix.

The parameter vectors π (termed x0 in MARSS), **a** and **u** are each set to be a column vector of zeros. Either of the following can be used:

```
x0 = U = matrix(0, nrow=3, ncol=1)
A = matrix(0, nrow=6, ncol=1)
x0 = U = A = "zero"
```

The Λ matrix (termed V0 in MARSS) is a diagonal matrix with 5's along the diagonal:

```
V0 = diag(5,3)
```

Finally, we make a list of the model parameters to pass to the MARSS() function and set the control list:

```
dfa.model = list(Z=Z, A="zero", R=R, B=B, U=U, Q=Q, x0=x0, V0=V0)
cntl.list = list(maxit=50)
```

For the examples in this chapter, we have set the maximum iterations to 50 to speed up model fitting. Note, however, that the parameter estimates will not have converged to their maximum likelihood values, which would likely take 100s, if not 1000+, iterations.

9.3.1 Fitting the model

We can now pass the DFA model list to MARSS() to estimate the **Z** matrix and underlying hidden states (**x**). The output is not shown because it is voluminous, but the model fits are plotted in Figure 9.2. The warnings regarding non-convergence are due to setting maxit to 50.

```
kemz.3 = MARSS(dat.z, model=dfa.model, control=cntl.list)
```

```
Warning! Reached maxit before parameters converged. Maxit was 50.
 neither abstol nor log-log convergence tests were passed.

MARSS fit is
Estimation method: kem
Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
WARNING: maxit reached at  50  iter before convergence.
 Neither abstol nor log-log convergence test were passed.
 The likelihood and params are not at the ML values.
 Try setting control$maxit higher.
Log-likelihood: -782.202
AIC: 1598.404   AICc: 1599.463

                            Estimate
Z.z11                         0.4163
Z.z21                         0.5364
Z.z31                         0.2780
Z.z41                         0.5179
Z.z51                         0.1611
Z.z22                         0.6757
Z.z32                        -0.2381
Z.z42                        -0.2381
Z.z52                        -0.2230
Z.z33                         0.2305
Z.z43                        -0.1225
Z.z53                         0.3887
R.(Cryptomonas,Cryptomonas)   0.6705
R.(Diatoms,Diatoms)           0.0882
R.(Greens,Greens)             0.7201
R.(Unicells,Unicells)         0.1865
R.(Other.algae,Other.algae)   0.5441

Standard errors have not been calculated.
Use MARSSparamCIs to compute CIs and bias estimates.

Convergence warnings
10 warnings. First 10 shown.  Type cat(object$errors) to see the full list.
 Warning: the  Z.z51  parameter value has not converged.
 Warning: the  Z.z32  parameter value has not converged.
 Warning: the  Z.z52  parameter value has not converged.
 Warning: the  Z.z33  parameter value has not converged.
 Warning: the  Z.z43  parameter value has not converged.
 Warning: the  R.(Diatoms,Diatoms)  parameter value has not converged.
 Warning: the  R.(Greens,Greens)  parameter value has not converged.
 Warning: the  R.(Other.algae,Other.algae)  parameter value has not converged.
```

```
Warning: the  logLik  parameter value has not converged.
Type MARSSinfo("convergence") for more info on this warning.
```

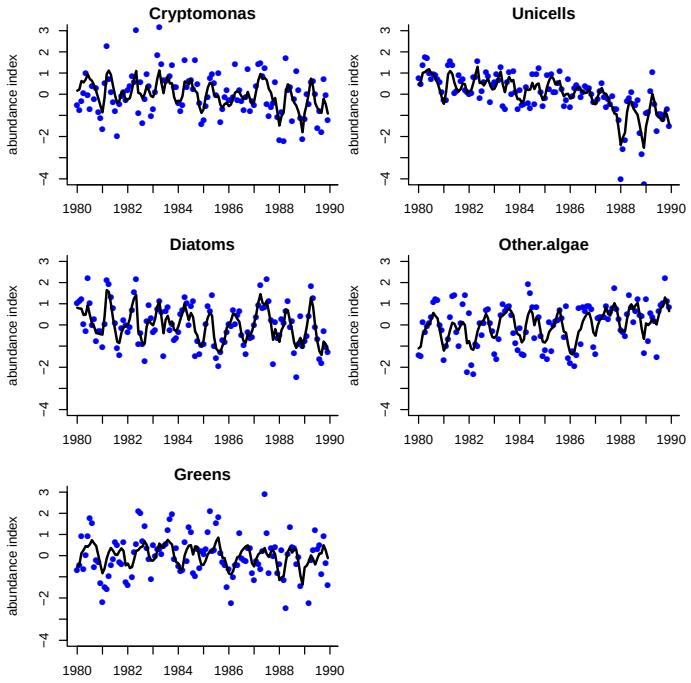

Fig. 9.2. Plots of Lake Washington phytoplankton data with model fits (dark lines) from a model with 3 trends and a diagonal and unequal variance-covariance matrix for the observation errors. This model was run to convergence so is different than that shown in the text which uses `maxit=50`.

9.4 Using model selection to determine the number of trends

Following Zuur et al. (2003), we use model selection criteria (specifically AICc) to determine the number of underlying trends that have the highest data support. Our first model had three underlying trends ($m = 3$). Let's compare this to a model with two underlying trends. The forms for parameter matrix **R** and vector **a** will stay the same but we need to change the other parameter vectors and matrices because m is different.

After showing you the matrix math behind a DFA model, we will now use the `form` argument for a MARSS call to specify that we want to fit a DFA model. This will set up the **Z** matrix and the other parameters for you. Specify how many trends you want by passing in `model=list(m=x)`. You can also pass in different forms for the **R** matrix in the usual way.

Here is how to fit two trends using `form="dfa"`:

```
model.list = list(m=2, R="diagonal and unequal")
kemz.2 = MARSS(dat.spp.1980, model=model.list,
    z.score=TRUE, form="dfa", control=cntl.list)
```

```
Warning! Reached maxit before parameters converged. Maxit was 50.
 neither abstol nor log-log convergence tests were passed.

MARSS fit is
Estimation method: kem
Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
WARNING: maxit reached at  50  iter before convergence.
 Neither abstol nor log-log convergence test were passed.
 The likelihood and params are not at the ML values.
 Try setting control$maxit higher.
Log-likelihood: -789.7433
AIC: 1607.487   AICc: 1608.209

                            Estimate
Z.11                          0.3128
Z.21                          0.1797
Z.31                          0.3061
Z.41                          0.5402
Z.51                          0.0791
Z.22                         -0.1174
Z.32                          0.4024
Z.42                         -0.0552
Z.52                          0.3895
R.(Cryptomonas,Cryptomonas)   0.7500
R.(Diatoms,Diatoms)           0.8565
R.(Greens,Greens)             0.5672
R.(Unicells,Unicells)         0.2292
R.(Other.algae,Other.algae)   0.6738

Standard errors have not been calculated.
Use MARSSparamCIs to compute CIs and bias estimates.

Convergence warnings
 Warning: the  Z.31  parameter value has not converged.
 Warning: the  Z.51  parameter value has not converged.
```

```
Warning: the  Z.32  parameter value has not converged.
Warning: the  Z.42  parameter value has not converged.
Warning: the  R.(Greens,Greens)  parameter value has not converged.
Warning: the  logLik  parameter value has not converged.
Type MARSSinfo("convergence") for more info on this warning.
```

and compare its AICc value to that from the 3-trend model.

```
print(cbind(model=c("3 trends", "2 trends"),
      AICc=round(c(kemz.3$AICc, kemz.2$AICc))),
      quote=FALSE)
```

```
     model    AICc
[1,] 3 trends 1589
[2,] 2 trends 1608
```

It looks like a model with 3 trends has much more support from the data because its AICc value is more than 10 units less than that for the 2-trend model.

9.4.1 Comparing many model structures

Now let's examine a larger suite of possible models. We will test from one to four underlying trends ($m = 1$ to 4) and four different structures for the **R** matrix:

1. same variances & no covariance (``diagonal and equal'');
2. different variances & no covariance (``diagonal and unequal'');
3. same variances & same covariance (``equalvarcov''); and
4. different variances & covariances (``unconstrained'').

The following code builds our model matrices; you could also write out each matrix as we did in the first example, but this allows us to build and run all of the models together. (*NOTE*: the following piece of code will take a *very long* time to run!)

```
# set new control params
cntl.list = list(minit=200, maxit=5000, allow.degen=FALSE)
# set up forms of R matrices
levels.R = c("diagonal and equal",
             "diagonal and unequal",
             "equalvarcov",
             "unconstrained")
model.data = data.frame()
# fit lots of models & store results
# NOTE: this will take a long time to run!
for(R in levels.R) {
    for(m in 1:(N.ts-1)) {
```

```
    dfa.model = list(A="zero", R=R, m=m)
    kemz = MARSS(dat.z, model=dfa.model, control=cntl.list,
        form="dfa", z.score=TRUE)
    model.data = rbind(model.data,
                       data.frame(R=R,
                                  m=m,
                                  logLik=kemz$logLik,
                                  K=kemz$num.params,
                                  AICc=kemz$AICc,
                                  stringsAsFactors=FALSE))
    assign(paste("kemz", m, R, sep="."), kemz)
    } # end m loop
} # end R loop
```

Model selection results are shown in Table 9.1. The model with lowest AICc has 2 trends and an unconstrained **R** matrix. It also appears that, in general, models with an unconstrained **R** matrix fit the data much better than those models with less complex structures for the observation errors (i.e., models with unconstrained forms for **R** had nearly all of the AICc weight).

Table 9.1. Model selection results.

R	m	logLik	delta.AICc	Ak.wt	Ak.wt.cum
unconstrained	3	-762.5	0.0	0.39	0.39
unconstrained	2	-765.9	0.1	0.37	0.76
unconstrained	4	-761.5	2.3	0.12	0.89
unconstrained	1	-772.4	4.4	0.04	0.93
diagonal and unequal	4	-774.2	5.9	0.02	0.95
equalvarcov	2	-782.7	6.1	0.02	0.97
diagonal and unequal	3	-777.1	7.5	0.01	0.98
diagonal and equal	4	-779.3	7.7	0.01	0.99
diagonal and equal	3	-781.8	8.4	0.01	0.99
equalvarcov	4	-779.0	9.1	0.00	1.00
equalvarcov	3	-781.4	9.9	0.00	1.00
diagonal and unequal	2	-786.6	20.2	0.00	1.00
equalvarcov	1	-799.9	32.3	0.00	1.00
diagonal and equal	2	-798.4	35.4	0.00	1.00
diagonal and unequal	1	-798.4	35.4	0.00	1.00
diagonal and equal	1	-813.5	57.4	0.00	1.00

9.5 Using varimax rotation to determine the loadings and trends

As Harvey (1989, p. 450) discusses in section 8.5.1, there are multiple equivalent solutions to the dynamic factor loadings. We arbitrarily constrained **Z** in

such a way to choose only one of these solutions, but fortunately the different solutions are equivalent, and they can be related to each other by a rotation matrix $\mathbf{H}$. Let $\mathbf{H}$ be any $m \times m$ non-singular matrix. The following are then equivalent solutions:

$$\begin{aligned} \mathbf{y}_t &= \mathbf{Z}\mathbf{x}_t + \mathbf{a} + \mathbf{v}_t \\ \mathbf{x}_t &= \mathbf{x}_{t-1} + \mathbf{w}_t \end{aligned} \tag{9.9}$$

and

$$\begin{aligned} \mathbf{y}_t &= \mathbf{Z}\mathbf{H}^{-1}\mathbf{x}_t + \mathbf{a} + \mathbf{v}_t \\ \mathbf{H}\mathbf{x}_t &= \mathbf{H}\mathbf{x}_{t-1} + \mathbf{H}\mathbf{w}_t \end{aligned} \tag{9.10}$$

There are many ways of doing factor rotations, but a common approach is the varimax rotation which seeks a rotation matrix $\mathbf{H}$ that creates the largest difference between loadings. For example, let's say there are three trends in our model. In our estimated $\mathbf{Z}$ matrix, let's say row 3 is $(0.2, 0.2, 0.2)$. That would mean that data series 3 is equally described by trends 1, 2, and 3. If instead row 3 was $(0.8, 0.1, 0.1)$, this would make interpretation easier because we could say that data time series 3 was mostly described by trend 1. The varimax rotation finds the $\mathbf{H}$ matrix that makes the $\mathbf{Z}$ rows more like $(0.8, 0.1, 0.1)$ and less like $(0.2, 0.2, 0.2)$.

The varimax rotation is easy to compute because `R` has a built in function for this. To do so, we first get the model fits from the highest ranked model.

```
# get the "best" model
best.model = model.tbl[1,]
fitname = paste("kemz",best.model$m,best.model$R,sep=".")
best.fit = get(fitname)
```

Next, we retrieve the matrix used for varimax rotation.

```
# get the inverse of the rotation matrix
H.inv = varimax(coef(best.fit, type="matrix")$Z)$rotmat
```

Finally, we use $\mathbf{H}^{-1}$ to rotate the factor loadings and $\mathbf{H}$ to rotate the trends as in Equation 9.10.

```
# rotate factor loadings
Z.rot = coef(best.fit, type="matrix")$Z %*% H.inv
# rotate trends
trends.rot = solve(H.inv) %*% best.fit$states
```

Rotated factor loadings for the best model are shown in Figure 9.3. Oddly, some taxa appear to have no loadings on some trends (e.g., diatoms on trend 1). The reason is that, merely for display purposes, we chose to plot only those loadings that are greater than 0.05, and it turns out that after varimax rotation, several loadings are close to 0.

Recall that we set $\text{Var}(\mathbf{w}_t) = \mathbf{Q} = \mathbf{I}_m$ in order to make our DFA model identifiable. Does the variance in the process errors also change following varimax

rotation? Interestingly, no. Because $\mathbf{H}$ is a non-singular, orthogonal matrix, $\text{Var}(\mathbf{H}\mathbf{w}_t) = \mathbf{H}\text{Var}(\mathbf{w}_t)\mathbf{H}^\top = \mathbf{H}\mathbf{I}_m\mathbf{H}^\top = \mathbf{I}_m$.

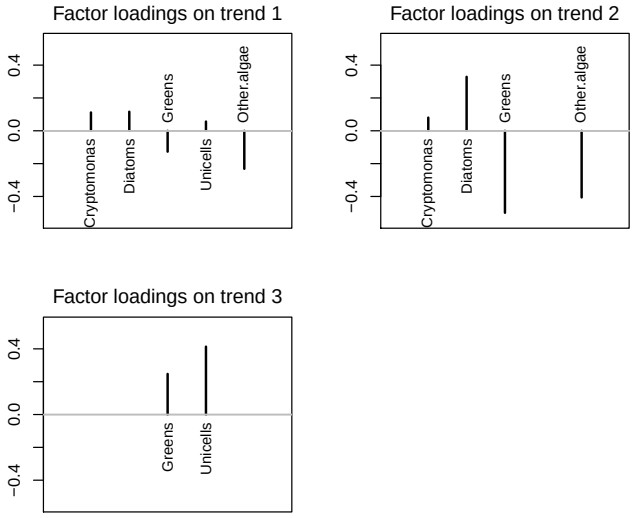

Fig. 9.3. Plot of the factor loadings (following varimax rotation) from the best model fit to the phytoplankton data.

9.6 Examining model fits

Now that we have found a "best" model and done the appropriate factor and trends rotations, we should examine some plots of model fits (see Figure 9.5). First, it looks like the model did an adequate job of capturing some of the high frequency variation (i.e., seasonality) in the time series. Second, some of the time series had much better overall fits than others (e.g., compare Diatoms versus Cryptomonas). Given the obvious seasonal patterns in the phytoplankton data, it might be worthwhile to first "detrend" the data and then repeat the model fitting exercise to see (1) how many trends would be favored, and (2) the shape of those trends.

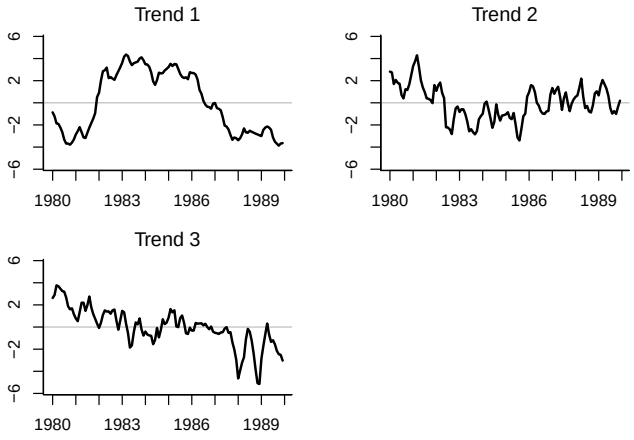

Fig. 9.4. Plot of the unobserved trends (following varimax rotation) from the best model fit to the phytoplankton data.

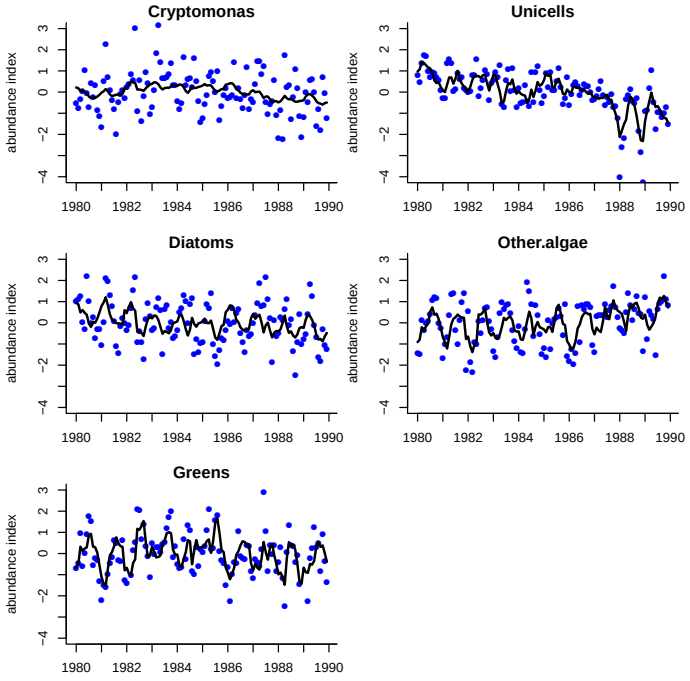

Fig. 9.5. Plot of the "best" model fits to the phytoplankton data.

9.7 Adding covariates

It is standard to add covariates to the analysis so that one removes known important drivers. The DFA with covariates is written:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{x}_{t-1} + \mathbf{w}_t \text{ where } \mathbf{w}_t \sim \text{MVN}(0, \mathbf{Q}) \\ \mathbf{y}_t &= \mathbf{Z}\mathbf{x}_t + \mathbf{a} + \mathbf{D}\mathbf{d}_t + \mathbf{v}_t \text{ where } \mathbf{v}_t \sim \text{MVN}(0, \mathbf{R}) \\ \mathbf{x}_0 &\sim \text{MVN}(\boldsymbol{\pi}, \boldsymbol{\Lambda}) \end{aligned} \tag{9.11}$$

where the $q \times 1$ vector $\mathbf{d}_t$ contains the covariate(s) at time t, and the $n \times q$ matrix $\mathbf{D}$ contains the effect(s) of the covariate(s) on the observations. Using `form="dfa"` and `covariates=`<*covariate name(s)*>, we can easily add covariates to our DFA, but this means that the covariates are input, not data, and there can be no missing values (see Chapter 6 for how to include covariates with missing values).

The Lake Washington dataset has two environmental covariates that we might expect to have effects on phytoplankton growth, and hence, abundance: temperature (Temp) and total phosphorous (TP). We need the covariate inputs to have the same number of time steps as the variate data, and thus we limit the covariate data to the years 1980-1994 also.

```
temp = t(plankdat[,"Temp",drop=FALSE])
TP = t(plankdat[,"TP",drop=FALSE])
```

We will now fit 3 different models that each add covariate effects (i.e., Temp, TP, Temp & TP) to our "best" model from Table 9.1 where $m = 2$ and $\mathbf{R}$ is "unconstrained".

```
model.list=list(m=2, R="unconstrained")
kemz.temp = MARSS(dat.spp.1980, model=model.list, z.score=TRUE,
        form="dfa", control=cntl.list, covariates=temp)
kemz.TP = MARSS(dat.spp.1980, model=model.list, z.score=TRUE,
        form="dfa", control=cntl.list, covariates=TP)
kemz.both = MARSS(dat.spp.1980, model=model.list, z.score=TRUE,
        form="dfa", control=cntl.list, covariates=rbind(temp,TP))
```

Next we can compare whether the addition of the covariates improves the model fit. (*NOTE*: The following results were obtained by letting the EM algorithm run for a *very long* time, so your results may differ.)

```
print(cbind(model=c("no covars", "Temp", "TP", "Temp & TP"),
      AICc=round(c(best.fit$AICc, kemz.temp$AICc, kemz.TP$AICc,
      kemz.both$AICc))), quote=FALSE)
```

```
     model     AICc
[1,] no covars 1582
[2,] Temp      1518
[3,] TP        1568
[4,] Temp & TP 1522
```

This suggests that adding temperature or phosphorus to the model, either alone or in combination with one another, improves overall model fit. If we were truly interested in assessing the "best" model structure that includes covariates, however, we should examine all combinations of trends and structures for **R**. The model fits for the temperature-only model are shown in Fig 9.6 and they appear much better than the best model without any covariates.

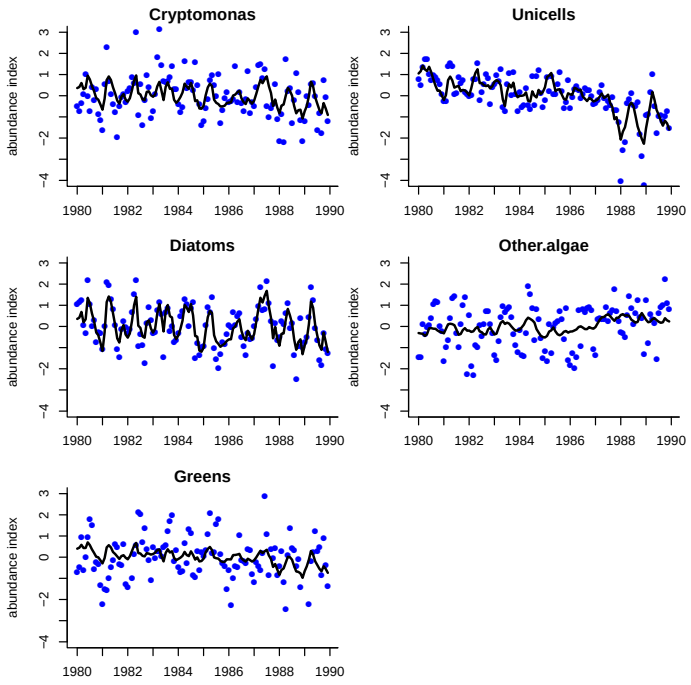

Fig. 9.6. Plot of the fits from the temperature-only model to the phytoplankton data.

9.8 Questions and further analyses

We analyzed the phytoplankton data alone. You can try analyzing the zooplankton data (type `head(plankdat)` to see the names). You can also try analyzing the phytoplankton and zooplankton together. You can also try different assumptions concerning the structure of **R**; we just tried unconstrained, diagonal and unequal, and diagonal and equal. To see all the `R` code behind the figures, type `RShowDoc("Chapter_DFA.R",package="MARSS")`. This opens a

file with all the code. Copy and paste the code into a new file, and then you can edit that code. DFA models often take an unusually long time to converge. In a real DFA, you will want to make sure to try different initial starting values (e.g., set `MCInit = TRUE`), and force the algorithm to run a long time by using `minit=x` and `maxit=(x+c)`, where `x` and `c` are something like 200 and 5000, respectively.