1

Fitting multivariate state-space models with covariates

A MARSS model with covariate effects in both the process and observation components is written as:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{B}_t\mathbf{x}_{t-1} + \mathbf{u}_t + \mathbf{C}_t\mathbf{c}_t + \mathbf{w}_t, \text{ where } \mathbf{w}_t \sim \text{MVN}(0, \mathbf{Q}_t) \\ \mathbf{y}_t &= \mathbf{Z}_t\mathbf{x}_t + \mathbf{a}_t + \mathbf{D}_t\mathbf{d}_t + \mathbf{v}_t, \text{ where } \mathbf{v}_t \sim \text{MVN}(0, \mathbf{R}_t) \end{aligned} \tag{1.1}$$

where $\mathbf{c}_t$ is the $p \times 1$ vector of covariates (e.g., temperature, rainfall) which affect the states and $\mathbf{d}_t$ is a $q \times 1$ vector of covariates (potentially the same as $\mathbf{c}_t$), which affect the observations. $\mathbf{C}_t$ is an $m \times p$ matrix of coefficients relating the effects of $\mathbf{c}_t$ to the $m \times 1$ state vector $\mathbf{x}_t$, and $\mathbf{D}_t$ is an $n \times q$ matrix of coefficients relating the effects of $\mathbf{d}_t$ to the $n \times 1$ observation vector $\mathbf{y}_t$.

With the `MARSS()` function, one can fit this model by passing in `model$c` and/or `model$d` in the `model` argument as a $p \times T$ or $q \times T$ matrix, respectively. The form for $\mathbf{C}_t$ and $\mathbf{D}_t$ is similarly specified by passing in `model$C` and/or `model$D`. $\mathbf{C}$ and $\mathbf{D}$ are matrices and are specified as 2-dimensional matrices as you would other parameter matrices.

1.1 Examples using plankton data

Here we show some examples using the Lake Washington plankton data set and covariates in that dataset. We use the 10 years of data from 1965-1974 (Figure 1.1), a decade with particularly high green and bluegreen algae levels. We use the transformed plankton dataset which has 0s replaced with NAs. Below, we set up the data and z-score the data. The original data were already z-scored, but we changed the mean when we subsampled the years so need to z-score again.

```
fulldat = lakeWAplanktonTrans
years = fulldat[,"Year"]>=1965 & fulldat[,"Year"]<1975
dat = t(fulldat[years,c("Greens", "Bluegreens")])
```

```
the.mean = apply(dat,1,mean,na.rm=TRUE)
the.sigma = sqrt(apply(dat,1,var,na.rm=TRUE))
dat = (dat-the.mean)*(1/the.sigma)
```

Next we set up the covariate data, temperature and total phosphorous. We z-score the covariates to standardize and remove the mean.

```
covariates = rbind(
   Temp = fulldat[years,"Temp"],
   TP = fulldat[years,"TP"])
# z.score the covariates
the.mean = apply(covariates,1,mean,na.rm=TRUE)
the.sigma = sqrt(apply(covariates,1,var,na.rm=TRUE))
covariates = (covariates-the.mean)*(1/the.sigma)
```

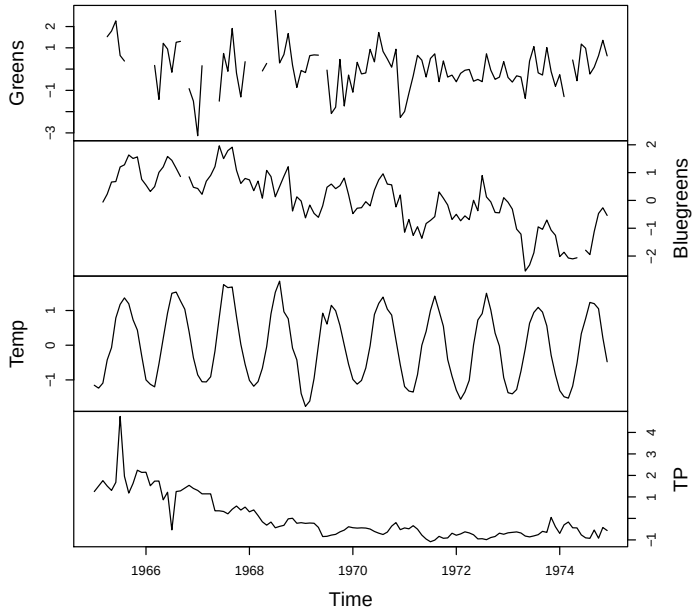

Fig. 1.1. Time series of Green and Bluegreen algae abundances in Lake Washington along with the temperature and total phosporous covariates.

1.2 Observation-error only model

We can estimate the effect of the covariates using a process-error only model, an observation-error only model, or a model with both types of error. An observation-error only model is a multivariate regression, and we will start here so you see the relationship of MARSS model to more familiar linear regression models.

In a standard multivariate linear regression, we only have an observation model with independent errors (*i.e.*, the state process does not appear in the model):

$$\mathbf{y}_t = \mathbf{a} + \mathbf{D}\mathbf{d}_t + \mathbf{v}_t, \text{ where } \mathbf{v}_t \sim \text{MVN}(0, \mathbf{R}) \tag{1.2}$$

The elements in **a** are the intercepts and those in **D** are the slopes (effects). We have dropped the t subscript on **a** and **D** because these will be modeled as time-constant. Writing this out for the two plankton and the two covariates we get:

$$\begin{bmatrix} y_g \\ y_{bg} \end{bmatrix}_t = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} \beta_{\text{g,temp}} & \beta_{\text{g,tp}} \\ \beta_{\text{bg,temp}} & \beta_{\text{bg,tp}} \end{bmatrix} \begin{bmatrix} \text{temp} \\ \text{tp} \end{bmatrix}_{t-1} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}_t \tag{1.3}$$

Let's fit this model with MARSS. The **x** part of the model is irrelevant so we want to fix the parameters in that part of the model. We won't set $\mathbf{B} = 0$ or $\mathbf{Z} = 0$ since that might cause numerical issues for the Kalman filter. Instead we fix them as identity matrices and fix $\mathbf{x}_0 = 0$ so that $\mathbf{x}_t = 0$ for all t.

```
Q = U = x0 = "zero"; B = Z = "identity"
d = covariates
A = "zero"
D = "unconstrained"
y = dat # to show relationship between dat & the equation
model.list = list(B=B,U=U,Q=Q,Z=Z,A=A,D=D,d=d,x0=x0)
kem = MARSS(y, model=model.list)
```

```
Success! algorithm run for 15 iterations. abstol and log-log tests passed.
Alert: conv.test.slope.tol is 0.5.
Test with smaller values (<0.1) to ensure convergence.

MARSS fit is
Estimation method: kem
Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
Algorithm ran 15 (=minit) iterations and convergence was reached.
Log-likelihood: -276.4287
AIC: 562.8573   AICc: 563.1351

                   Estimate
R.diag                0.706
D.(Greens,Temp)       0.367
```

```
D.(Bluegreens,Temp)    0.392
D.(Greens,TP)          0.058
D.(Bluegreens,TP)      0.535

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

We set `A="zero"` because the data and covariates have been demeaned. Of course, one can do multiple regression in R using, say, `lm()`, and that would be much, much faster. The EM algorithm is over-kill here, but it is shown so that you see how a standard multivariate linear regression model is written as a MARSS model in matrix form.

1.3 Process-error only model

Now let's model the data as an autoregressive process observed without error, and incorporate the covariates into the process model. Note that this is much different from typical linear regression models. The **x** part represents our model of the data (in this case plankton species). How is this different from the autoregressive observation errors? Well, we are modeling our data as autoregressive so data at $t-1$ affects the data at t. Population abundances are inherently autoregressive so this model is a bit closer to the underlying mechanism generating the data. Here is our new process model for plankton abundance.

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{C}\mathbf{c}_t + \mathbf{w}_t, \text{ where } \mathbf{w}_t \sim \text{MVN}(0, \mathbf{Q}) \tag{1.4}$$

We can fit this as follows:

```
R = A = U = "zero"; B = Z = "identity"
Q = "equalvarcov"
C = "unconstrained"
model.list = list(B=B,U=U,Q=Q,Z=Z,A=A,R=R,C=C,c=covariates)
kem = MARSS(dat, model=model.list)
```

```
Success! algorithm run for 15 iterations. abstol and log-log tests passed.
Alert: conv.test.slope.tol is 0.5.
Test with smaller values (<0.1) to ensure convergence.

MARSS fit is
Estimation method: kem
Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
Algorithm ran 15 (=minit) iterations and convergence was reached.
Log-likelihood: -285.0732
AIC: 586.1465   AICc: 586.8225
```

```
                      Estimate
Q.diag                  0.7269
Q.offdiag              -0.0210
x0.X.Greens            -0.5189
x0.X.Bluegreens        -0.2431
C.(X.Greens,Temp)      -0.0434
C.(X.Bluegreens,Temp)   0.0988
C.(X.Greens,TP)        -0.0589
C.(X.Bluegreens,TP)     0.0104

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

Now, it looks like temperature has a strong negative effect on algae? Also our log-likelihood dropped a lot. Well, the data do not look at all like a random walk model (*i.e.*, where $\mathbf{B} = 1$), which we can see from the plot of the data (Figure 1.1). The data are fluctuating about some mean so let's switch to a better autoregressive model—a mean-reverting model. To do this, we will allow the diagonal elements of $\mathbf{B}$ to be something other than 1.

```
model.list$B = "diagonal and unequal"
kem = MARSS(dat, model=model.list)
```

```
Success! algorithm run for 15 iterations. abstol and log-log tests passed.
Alert: conv.test.slope.tol is 0.5.
Test with smaller values (<0.1) to ensure convergence.

MARSS fit is
Estimation method: kem
Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
Algorithm ran 15 (=minit) iterations and convergence was reached.
Log-likelihood: -236.6106
AIC: 493.2211   AICc: 494.2638

                             Estimate
B.(X.Greens,X.Greens)          0.1981
B.(X.Bluegreens,X.Bluegreens)  0.7672
Q.diag                         0.4899
Q.offdiag                     -0.0221
x0.X.Greens                   -1.2915
x0.X.Bluegreens               -0.4179
C.(X.Greens,Temp)              0.2844
C.(X.Bluegreens,Temp)          0.1655
C.(X.Greens,TP)                0.0332
C.(X.Bluegreens,TP)            0.1340
```

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

Notice that the log-likelihood goes up quite a bit, which means that the mean-reverting model fits the data much better.

With this model, we are estimating $\mathbf{x}_0$. If we set `model$tinitx=1`, we will get a error message that $\mathbf{R}$ diagonals are equal to 0 and we need to fix `x0`. Because $\mathbf{R} = 0$, if we set the initial states at $t = 1$, then they are fully determined by the data.

```
x0 = dat[,1,drop=FALSE]
model.list$tinitx = 1
model.list$x0 = x0
kem = MARSS(dat, model=model.list)
```

```
Success! algorithm run for 15 iterations. abstol and log-log tests passed.
Alert: conv.test.slope.tol is 0.5.
Test with smaller values (<0.1) to ensure convergence.

MARSS fit is
Estimation method: kem
Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
Algorithm ran 15 (=minit) iterations and convergence was reached.
Log-likelihood: -235.4827
AIC: 486.9653   AICc: 487.6414

                               Estimate
B.(X.Greens,X.Greens)            0.1980
B.(X.Bluegreens,X.Bluegreens)    0.7671
Q.diag                           0.4944
Q.offdiag                       -0.0223
C.(X.Greens,Temp)                0.2844
C.(X.Bluegreens,Temp)            0.1655
C.(X.Greens,TP)                  0.0332
C.(X.Bluegreens,TP)              0.1340

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

1.4 Both process- and observation-error

Here is an example where we have both process and observation error but the covariates only affect the process:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{B}\mathbf{x}_{t-1} + \mathbf{C}_t\mathbf{c}_t + \mathbf{w}_t, \text{ where } \mathbf{w}_t \sim \text{MVN}(0, \mathbf{Q}) \\ \mathbf{y}_t &= \mathbf{x}_{t-1} + \mathbf{v}_t, \text{ where } \mathbf{v}_t \sim \text{MVN}(0, \mathbf{R}), \end{aligned} \tag{1.5}$$

x is the true algae abundances and **y** is the observation of the **x**'s.

Let's say we knew that the observation variance on the algae measurements was about 0.16 and we wanted to include that known value in the model. To do that, we can simply add **R** to the model list from the process-error only model in the last example.

```
D = d = A = U = "zero"; Z = "identity"
B = "diagonal and unequal"
Q = "equalvarcov"
C = "unconstrained"
c=covariates
R = diag(0.16,2)
x0 = "unequal"
tinitx=1
model.list = list(B=B,U=U,Q=Q,Z=Z,A=A,R=R,D=D,d=d,C=C,c=c,x0=x0,tinitx=tinitx)
kem = MARSS(dat, model=model.list)
```

```
Success! abstol and log-log tests passed at 36 iterations.
Alert: conv.test.slope.tol is 0.5.
Test with smaller values (<0.1) to ensure convergence.

MARSS fit is
Estimation method: kem
Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
Estimation converged in 36 iterations.
Log-likelihood: -240.3694
AIC: 500.7389   AICc: 501.7815

                              Estimate
B.(X.Greens,X.Greens)          0.30848
B.(X.Bluegreens,X.Bluegreens)  0.76101
Q.diag                         0.33923
Q.offdiag                     -0.00411
x0.X.Greens                   -0.52614
x0.X.Bluegreens               -0.32836
C.(X.Greens,Temp)              0.23790
C.(X.Bluegreens,Temp)          0.16991
C.(X.Greens,TP)                0.02505
C.(X.Bluegreens,TP)            0.14183

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

Note, our estimates of the effect of temperature and total phosphorous are not that different than what you get from a simple multiple regression (our first example). This might be because the autoregressive component is small, meaning the estimated diagonals on the **B** matrix are small.

Here is an example where we have both process and observation error but the covariates only affect the observation process:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{B}\mathbf{x}_{t-1} + \mathbf{w}_t, \text{ where } \mathbf{w}_t \sim \text{MVN}(0, \mathbf{Q}) \\ \mathbf{y}_t &= \mathbf{x}_{t-1} + \mathbf{D}\mathbf{d}_t\mathbf{v}_t, \text{ where } \mathbf{v}_t \sim \text{MVN}(0, \mathbf{R}), \end{aligned} \tag{1.6}$$

x is the true algae abundances and **y** is the observation of the **x**'s.

```
C = c = A = U = "zero"; Z = "identity"
B = "diagonal and unequal"
Q = "equalvarcov"
D = "unconstrained"
d=covariates
R = diag(0.16,2)
x0 = "unequal"
tinitx=1
model.list = list(B=B,U=U,Q=Q,Z=Z,A=A,R=R,D=D,d=d,C=C,c=c,x0=x0,tinitx=tinitx)
kem = MARSS(dat, model=model.list)
```

```
Success! abstol and log-log tests passed at 45 iterations.
Alert: conv.test.slope.tol is 0.5.
Test with smaller values (<0.1) to ensure convergence.

MARSS fit is
Estimation method: kem
Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
Estimation converged in 45 iterations.
Log-likelihood: -239.5879
AIC: 499.1759   AICc: 500.2185

                             Estimate
B.(X.Greens,X.Greens)           0.428
B.(X.Bluegreens,X.Bluegreens)   0.859
Q.diag                          0.314
Q.offdiag                      -0.030
x0.X.Greens                    -0.121
x0.X.Bluegreens                -0.119
D.(Greens,Temp)                 0.373
D.(Bluegreens,Temp)             0.276
D.(Greens,TP)                   0.042
D.(Bluegreens,TP)               0.115

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

1.5 Including seasonal effects in MARSS models

Time-series data are often collected at intervals with some implicit "seasonality." For example, quarterly earnings for a business, monthly rainfall totals, or hourly air temperatures. In those cases, it is often helpful to extract any recurring seasonal patterns that might otherwise mask some of the other temporal dynamics we are interested in examining.

Here we show a few approaches for including seasonal effects using the Lake Washington plankton data, which were collected monthly. The following examples will use all five phytoplankton species from Lake Washington. First, let's set up the data.

```
years = fulldat[,"Year"]>=1965 & fulldat[,"Year"]<1975
phytos = c("Diatoms", "Greens", "Bluegreens",
           "Unicells", "Other.algae")
dat = t(fulldat[years,phytos])
# z.score data because we changed the mean when we subsampled
the.mean = apply(dat,1,mean,na.rm=TRUE)
the.sigma = sqrt(apply(dat,1,var,na.rm=TRUE))
dat = (dat-the.mean)*(1/the.sigma)
# number of time periods/samples
TT = dim(dat)[2]
```

1.5.1 Seasonal effects as fixed factors

One common approach for estimating seasonal effects is to treat each one as a fixed factor, such that the number of parameters equals the number of "seasons" (*e.g.*, 24 hours per day, 4 quarters per year). The plankton data are collected monthly, so we will treat each month as a fixed factor. To fit a model with fixed month effects, we create a $12 \times T$ covariate matrix $\mathbf{c}$ with one row for each month (Jan, Feb, ...) and one column for each time point. We put a 1 in the January row for each column corresponding to a January time point, a 1 in the February row for each column corresponding to a February time point, and so on. All other values of $\mathbf{c}$ equal 0. The following code will create such a $\mathbf{c}$ matrix.

```
# number of "seasons" (e.g., 12 months per year)
period = 12
# first "season" (e.g., Jan = 1, July = 7)
per.1st = 1
# create factors for seasons
c.in = diag(period)
for(i in 2:(ceiling(TT/period))) {c.in = cbind(c.in,diag(period))}
# trim c.in to correct start & length
c.in = c.in[,(1:TT)+(per.1st-1)]
# better row names
rownames(c.in) = month.abb
```

Next we need to set up the form of the **C** matrix which defines any constraints we want to set on the month effects. **C** is a 5×12 matrix. Five taxon and 12 month effects. If we wanted each taxon to have the same month effect, i.e. there is a common month effect across all taxon, then we have the same value in each **C** column[1]:

```
C = matrix(month.abb,5,12,byrow=TRUE)
C
```

```
     [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9]
[1,] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep"
[2,] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep"
[3,] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep"
[4,] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep"
[5,] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep"
     [,10] [,11] [,12]
[1,] "Oct" "Nov" "Dec"
[2,] "Oct" "Nov" "Dec"
[3,] "Oct" "Nov" "Dec"
[4,] "Oct" "Nov" "Dec"
[5,] "Oct" "Nov" "Dec"
```

Notice, that **C** only has 12 values in it, the 12 common month effects. However, for this example, we will let each taxon have a different month effect thus allowing different seasonality for each taxon. For this model, we want each value in **C** to be unique:

```
C = "unconstrained"
```

Now **C** has $5 \times 12 = 60$ separate effects.

Then we set up the form for the rest of the model parameters. We make the following assumptions:

```
# Each taxon has unique density-dependence
B = "diagonal and unequal"
# Assume independent process errors
Q = "diagonal and unequal"
# We have demeaned the data & are fitting a mean-reverting model
# by estimating a diagonal B, thus
U = "zero"
# Each obs time series is associated with only one process
Z = "identity"
# The data are demeaned & fluctuate around a mean
A = "zero"
# We assume observation errors are independent, but they
# have similar variance due to similar collection methods
```

[1] `month.abb` is a R constant that gives month abbreviations in text.

```
R = "diagonal and equal"
# We are not including covariate effects in the obs equation
D = "zero"
d = "zero"
```

Now we can set up the model list for MARSS and fit the model (results are not shown since they are verbose with 60 different month effects).

```
model.list = list(B=B,U=U,Q=Q,Z=Z,A=A,R=R,C=C,c=c.in,D=D,d=d)
seas.mod.1 = MARSS(dat,model=model.list,control=list(maxit=1500))
# Get the estimated seasonal effects
# rows are taxa, cols are seasonal effects
seas.1 = coef(seas.mod.1,type="matrix")$C
rownames(seas.1) = phytos
colnames(seas.1) = month.abb
```

The top panel in Figure 1.2 shows the estimated seasonal effects for this model. Note that if we had set U="unequal", we would need to set one of the columns of **C** to zero because the model would be under-determined (infinite number of solutions). If we substracted the mean January abundance off each time series, we could set the January column in **C** to 0 and get rid of 5 estimated effects.

1.5.2 Seasonal effects as a polynomial

The fixed factor approach required estimating 60 effects. Another approach is to model the month effect as a 3^{rd}-order (or higher) polynomial: $a + b \times m + c \times m^2 + d \times m^3$ where m is the month number. This approach has less flexibility but requires only 20 estimated parameters (*i.e.*, 4 regression parameters times 5 taxa). To do so, we create a $4 \times T$ covariate matrix **c** with the rows corresponding to 1, m, m^2, and m^3, and the columns again corresponding to the time points. Here is how to set up this matrix:

```
# number of "seasons" (e.g., 12 months per year)
period = 12
# first "season" (e.g., Jan = 1, July = 7)
per.1st = 1
# order of polynomial
poly.order = 3
# create polynomials of months
month.cov = matrix(1,1,period)
for(i in 1:poly.order) {month.cov = rbind(month.cov,(1:12)^i)}
# our c matrix is month.cov replicated once for each year
c.m.poly = matrix(month.cov, poly.order+1, TT+period, byrow=FALSE)
# trim c.in to correct start & length
c.m.poly = c.m.poly[,(1:TT)+(per.1st-1)]
# Everything else remains the same as in the previous example
```

```
model.list = list(B=B,U=U,Q=Q,Z=Z,A=A,R=R,C=C,c=c.m.poly,D=D,d=d)
seas.mod.2 = MARSS(dat, model=model.list, control=list(maxit=1500))
```

The effect of month m for taxon i is $a_i + b_i \times m + c_i \times m^2 + d_i \times m^3$, where a_i, b_i, c_i and d_i are in the i-th row of $\mathbf{C}$. We can now calculate the matrix of seasonal effects as follows, where each row is a taxon and each column is a month:

```
C.2 = coef(seas.mod.2,type="matrix")$C
seas.2 = C.2 %*% month.cov
rownames(seas.2) = phytos
colnames(seas.2) = month.abb
```

The middle panel in Figure 1.2 shows the estimated seasonal effects for this polynomial model.

1.5.3 Seasonal effects as a Fourier series

The factor approach required estimating 60 effects, and the 3rd order polynomial model was an improvement at only 20 parameters. A third option is to use a discrete Fourier series, which is combination of sine and cosine waves; it would require only 10 parameters. Specifically, the effect of month m on taxon i is $a_i \times \cos(2\pi m/p) + b_i \times \sin(2\pi m/p)$, where p is the period (*e.g.*, 12 months, 4 quarters), and a_i and b_i are contained in the i-th row of $\mathbf{C}$.

We begin by defining the $2 \times T$ seasonal covariate matrix $\mathbf{c}$ as a combination of 1 cosine and 1 sine wave:

```
cos.t = cos(2 * pi * seq(TT) / period)
sin.t = sin(2 * pi * seq(TT) / period)
c.Four = rbind(cos.t,sin.t)
```

Everything else remains the same and we can fit this model as follows:

```
model.list = list(B=B,U=U,Q=Q,Z=Z,A=A,R=R,C=C,c=c.Four,D=D,d=d)
seas.mod.3 = MARSS(dat, model=model.list, control=list(maxit=1500))
```

We make our seasonal effect matrix as follows:

```
C.3 = coef(seas.mod.3, type="matrix")$C
# The time series of net seasonal effects
seas.3 = C.3 %*% c.Four[,1:period]
rownames(seas.3) = phytos
colnames(seas.3) = month.abb
```

The bottom panel in Figure 1.2 shows the estimated seasonal effects for this seasonal-effects model based on a discrete Fourier series.

Rather than rely on our eyes to judge model fits, we should formally assess which of the 3 approaches offers the most parsimonious fit to the data. Here is a table of AICc values for the 3 models:

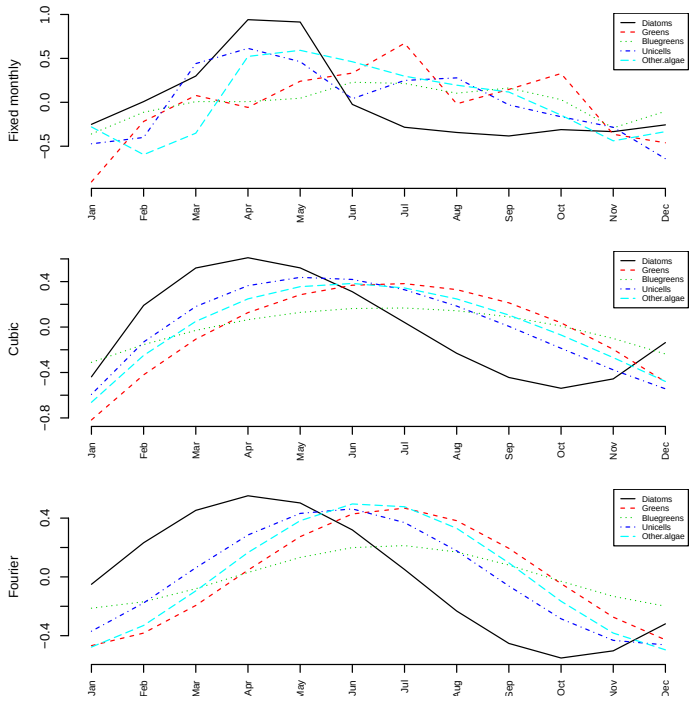

Fig. 1.2. Estimated monthly effects for the three approaches to estimating seasonal effects. Top panel: each month modelled as a separate fixed effect for each taxon (60 parameters); Middle panel: monthly effects modelled as a 3rd order polynomial (20 parameters); Bottom panel: monthly effects modelled as a discrete Fourier series (10 parameters).

```
data.frame(Model=c("Fixed", "Cubic", "Fourier"),
           AICc=round(c(seas.mod.1$AICc,
                        seas.mod.2$AICc,
                        seas.mod.3$AICc),1))
```

```
    Model   AICc
1   Fixed 1188.4
2   Cubic 1144.9
3 Fourier 1127.4
```

The model selection results indicate that the model with monthly seasonal effects estimated via the discrete Fourier sequence is the best of the 3 models. Its AICc value is much lower than either the polynomial or fixed-effects models.

1.6 Model diagnostics

We will examine some basic model diagnostics for these three approaches by looking at plots of the model residuals and their autocorrelation functions (ACFs) for all five taxa using the following code:

```
for(i in 1:3) {
  dev.new()
  modn = paste("seas.mod",i,sep=".")
  for(j in 1:5) {
    plot.ts(residuals(get(modn))$model.residuals[j,],
      ylab="Residual", main=phytos[j])
    abline(h=0, lty="dashed")
    acf(residuals(get(modn))$model.residuals[j,])
    }
  }
```

```
i = 3; #Fourier
j = 1; #First state
par(mfrow=c(2,1))
modn = paste("seas.mod",i,sep=".")
plot.ts(residuals(get(modn))$model.residuals[j,],
    ylab="Residual", main=phytos[j])
abline(h=0, lty="dashed")
acf(residuals(get(modn))$model.residuals[j,])
```

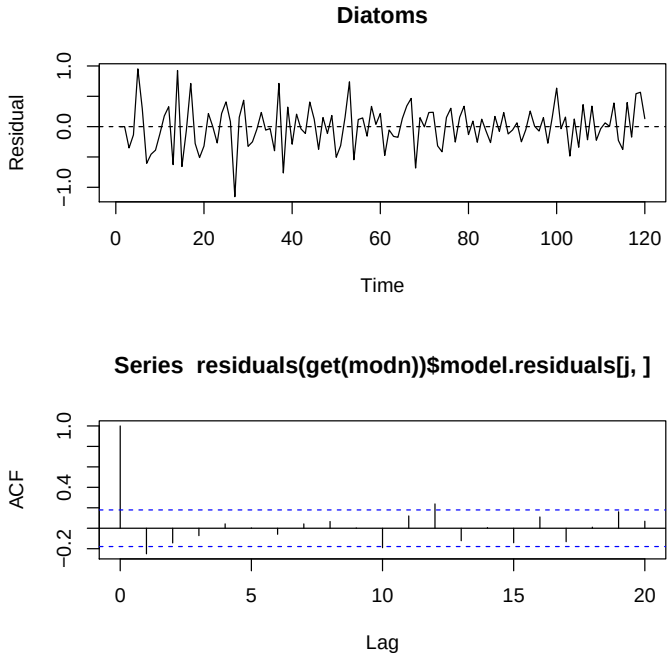

Fig. 1.3. Residuals for model with season modelled as a discrete Fourier series.

Problems

For these problems, use the following code to load in phytoplankton data, covariates, and z-score all the data. Then use `dat` and `covars` directly in your code.

```
phytos = c("Cryptomonas", "Diatoms", "Greens",
           "Unicells", "Other.algae")
yrs = lakeWAplanktonTrans[,"Year"]%in%1985:1994
dat = t(lakeWAplanktonTrans[yrs,phytos])
prec = diag(1/sqrt(apply(dat, 1, var, na.rm=TRUE)))
avg = apply(dat, 1, mean, na.rm=TRUE)
dat = prec %*% (dat-avg)
rownames(dat) = phytos
covars = rbind(Temp=lakeWAplanktonTrans[yrs,"Temp"],
               TP=lakeWAplanktonTrans[yrs,"TP"])
prec = diag(1/sqrt(apply(covars, 1, var)))
avg = apply(covars, 1, mean)
covars =  prec %*% (covars-avg)
rownames(covars) = c("Temp","TP")
```

Here are some guidelines to help you answer the questions:

- Use a MARSS model that allows for both observation and process error.
- Assume that the observation errors are independent and identically distributed. You can further assume that any process errors are independent from one another, but the variances differ by taxon.
- Assume that each group is an observation of its own process. This means `Z="identity"`.
- Use `B="diagonal and unequal"`. This implies that each of the taxa are operating under varying degrees of density-dependence, and that they do not interact with any of the other taxa.
- All the data have been de-meaned and $\mathbf{Z}$ identity, therefore use `U="zero"` and `A="zero"`.
- Include a plot of residuals versus time and acf of residuals for each question.
- Use AICc to compare models.

1.1 How does month affect the mean phytoplankton population growth rates? Show a plot of mean growth rate versus month. Estimate seasonal effects without any covariate (Temp, TP) effects.

1.2 It is likely that both temperature and total phosphorus (TP) affect phytoplankton population growth rates. Using MARSS models, evaluate which is the more important driver or if both are important. Leave out the seasonal covariates from question 1, i.e. only use Temp and TP as covariates.

1.3 Evaluate whether the effect of temperature on phytoplankton manifests itself via their underlying physiology (by affecting algal growth rates and thus abundance) or because physical changes in the water stratification makes them easier/harder to sample in some months. Leave out the seasonal covariates from question 1, i.e. only use Temp and TP as covariates.

1.4 Is there support for temperature or TP affecting all functional groups' growth rates the same, or are the effects on one taxon different from another?

1.5 Compare your results for questions 2-4 using an observation error only model, by using the lm() function.

1.6 Then compare to a process error only model using the arima() function with the xreg argument.

1.7 Compute a time-series cross-validation metric for the models and compare the results that you got using AICc for model comparison.