1

Linear regression models in matrix form

This chapter shows how to write linear regression models in matrix form. The purpose is to get you comfortable writing multivariate linear models in different matrix forms before we start working with time-series versions of these models. Each matrix form is an equivalent model for the data, but written in different forms. You do not need to worry which form is better or worse at this point. Simply get comfortable writing multivariate linear models in different matrix forms.

We will work with the `stackloss` dataset available in R. The `stackloss` dataset consists of 21 observations on the efficiency of a plant that produces nitric acid as a function of three explanatory variables: air flow, water temperature and acid concentration. We are going to use just the first 4 datapoints so that it is easier to write the matrices, but the concepts extend to as many datapoints as you have

```
data(stackloss)
dat = stackloss[1:4,] #subsetted first 4 rows
dat
```

```
  Air.Flow Water.Temp Acid.Conc. stack.loss
1       80         27         89         42
2       80         27         88         37
3       75         25         90         37
4       62         24         87         28
```

We will start by regressing stack loss against air flow. In R using the `lm` function this is

```
require(stats)
lm(stack.loss ~ Air.Flow, data=dat)
```

This fits the following model for the i-th measurment:

$$stack.loss_i = \alpha + \beta air_i + e_i, \text{ where } e_i \sim \mathrm{N}(0, \sigma^2) \tag{1.1}$$

We will write the model for all the measurements together in two different ways, Form 1 and Form 2.

1.1 Form 1

In this form, we have the explanatory variables in a matrix on the left of our parameter matrix:

$$\begin{bmatrix} stack.loss_1 \\ stack.loss_2 \\ stack.loss_3 \\ stack.loss_4 \end{bmatrix} = \begin{bmatrix} 1 & air_1 \\ 1 & air_2 \\ 1 & air_3 \\ 1 & air_4 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} \tag{1.2}$$

You should work through the matrix algebra to make sure you understand why Equation 1.2 is Equation 1.1 for all the i data points together.

We can write the first line of Equation 1.2 succinctly as

$$\mathbf{y} = \mathbf{Z}\mathbf{x} + \mathbf{e} \tag{1.3}$$

where $\mathbf{x}$ are our parameters, $\mathbf{y}$ are our response variables, and $\mathbf{Z}$ are our explanatory variables (with a 1 column for the intercept). The `lm()` function uses Form 1, and we can recover the $\mathbf{Z}$ matrix for Form 1 by using the `model.matrix()` function on the output from a `lm` call:

```
fit=lm(stack.loss ~ Air.Flow, data=dat)
Z=model.matrix(fit)
Z[1:4,]
```

```
  (Intercept) Air.Flow
1           1       80
2           1       80
3           1       75
4           1       62
```

1.1.1 Solving for the parameters*

You will not need to know how to solve linear matrix equations for this course. This section just shows you what the `lm` function is doing to give you the parameters.

Notice that $\mathbf{Z}$ is not a square matrix and its inverse does not exist but the inverse of $\mathbf{Z}^\top\mathbf{Z}$ exists—if this is a solveable problem. We can go through the following steps to solve for $\mathbf{x}$, our parameters α and β.

$$
\begin{gathered}
\text{Start with } \mathbf{y} = \mathbf{Z}\mathbf{x} + \mathbf{e} \text{ and multiply by } \mathbf{Z}^\top \text{ on the left to get} \\
\mathbf{Z}^\top \mathbf{y} = \mathbf{Z}^\top \mathbf{Z}\mathbf{x} + \mathbf{Z}^\top \mathbf{e} \\
\text{Multiply that by } (\mathbf{Z}^\top \mathbf{Z})^{-1} \text{ on the left} \\
(\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{y} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{Z}\mathbf{x} + (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{e} \\
(\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{Z} \text{ equals the identity matrix} \\
(\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{y} = \mathbf{x} + (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{e} \\
\text{Move the } \mathbf{x} \text{ to the right by itself} \\
(\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{y} - (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{e} = \mathbf{x}
\end{gathered}
\tag{1.4}
$$

Let's assume our errors, the $\mathbf{e}$, are i.i.d. which means that

$$
\mathbf{e} \sim \text{MVN}\left(0, \begin{bmatrix} \sigma^2 & 0 & 0 & 0 \\ 0 & \sigma^2 & 0 & 0 \\ 0 & 0 & \sigma^2 & 0 \\ 0 & 0 & 0 & \sigma^2 \end{bmatrix}\right) \tag{1.5}
$$

This equation means "$\mathbf{e}$ is drawn from a multivariate normal distribution with a variance-covariance matrix that is diagonal with equal variances." Under that assumption, the expected value of $(\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{e}$ is zero. So we can solve for $\mathbf{x}$ as

$$
\mathbf{x} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{y}
$$

Let's try that with R and compare to what you get with `lm`:

```
y=matrix(dat$stack.loss, ncol=1)
Z=cbind(1,dat$Air.Flow) #or use model.matrix() to get Z
solve(t(Z)%*%Z)%*%t(Z)%*%y
```

```
            [,1]
[1,] -11.6159170
[2,]   0.6412918
```

```
coef(lm(stack.loss ~ Air.Flow, data=dat))
```

```
(Intercept)    Air.Flow
-11.6159170   0.6412918
```

As you see, you get the same values.

1.1.2 Form 1 with multiple explanatory variables

We can easily extend Form 1 to multiple explanatory variables. Let's say we wanted to fit this model:

$$
stack.loss_i = \alpha + \beta_1 air_i + \beta_2 water_i + \beta_3 acid_i + e_i \tag{1.6}
$$

With `lm`, we can fit this with

```
fit1.mult=lm(stack.loss ~ Air.Flow + Water.Temp + Acid.Conc., data=dat)
```

Written in matrix form (Form 1), this is

$$\begin{bmatrix} stack.loss_1 \\ stack.loss_2 \\ stack.loss_3 \\ stack.loss_4 \end{bmatrix} = \begin{bmatrix} 1 & air_1 & water_1 & acid_1 \\ 1 & air_2 & water_2 & acid_2 \\ 1 & air_3 & water_3 & acid_3 \\ 1 & air_4 & water_4 & acid_4 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} \tag{1.7}$$

Now $\mathbf{Z}$ is a matrix with 4 columns and $\mathbf{x}$ is a column vector with 4 rows. We can show the $\mathbf{Z}$ matrix again directly from our `lm` fit:

```
Z=model.matrix(fit1.mult)
Z
```

```
  (Intercept) Air.Flow Water.Temp Acid.Conc.
1           1       80         27         89
2           1       80         27         88
3           1       75         25         90
4           1       62         24         87
attr(,"assign")
[1] 0 1 2 3
```

We can solve for $\mathbf{x}$ just like before and compare to what we get with `lm`:

```
y=matrix(dat$stack.loss, ncol=1)
Z=cbind(1,dat$Air.Flow, dat$Water.Temp, dat$Acid.Conc)
#or Z=model.matrix(fit2)
solve(t(Z)%*%Z)%*%t(Z)%*%y
```

```
            [,1]
[1,] -524.904762
[2,]   -1.047619
[3,]    7.619048
[4,]    5.000000
```

```
coef(fit1.mult)
```

```
(Intercept)    Air.Flow  Water.Temp  Acid.Conc.
-524.904762   -1.047619    7.619048    5.000000
```

Take a look at the $\mathbf{Z}$ we made in R. It looks exactly like what is in our model written in matrix form (Equation 1.7).

1.1.3 When does Form 1 arise?

This form of writing a regression model will come up when you work with dynamic linear models (DLMs). With DLMs, you will be fitting models of the form $\mathbf{y}_t = \mathbf{Z}_t\mathbf{x}_t + \mathbf{e}_t$. In these models you have multiple $\mathbf{y}$ at regular time points and you allow your regression parameters, the $\mathbf{x}$, to evolve through time as a random walk.

1.1.4 Form 1b: The transpose of Form 1

We could also write Form 1 as follows:

$$\begin{bmatrix} stack.loss_1 & stack.loss_2 & stack.loss_3 & stack.loss_4 \end{bmatrix} = \begin{bmatrix} \alpha & \beta_1 & \beta_2 & \beta_3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ air_1 & air_2 & air_3 & air_4 \\ wind_1 & wind_2 & wind_3 & wind_4 \\ acid_1 & acid_2 & acid_3 & acid_4 \end{bmatrix} + \begin{bmatrix} e_1 & e_2 & e_3 & e_4 \end{bmatrix} \tag{1.8}$$

This is just the transpose of Form 1. Work through the matrix algebra to make sure you understand why Equation 1.8 is Equation 1.1 for all the i data points together and why it is equal to the transpose of Equation 1.2. You'll need the relationship $(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top$.

Let's write Equation 1.8 as $\mathbf{y} = \mathbf{Dd}$, where $\mathbf{D}$ contains our parameters. Then we can solve for $\mathbf{D}$ following the steps in Equation 1.4 but multiplying from the right instead of from the left. See if you can work through the steps to show that $\mathbf{d} = \mathbf{y}\mathbf{d}^\top(\mathbf{d}\mathbf{d}^\top)^{-1}$.

```
y=matrix(dat$stack.loss, nrow=1)
d=rbind(1, dat$Air.Flow, dat$Water.Temp, dat$Acid.Conc)
y%*%t(d)%*%solve(d%*%t(d))
```

```
          [,1]      [,2]     [,3] [,4]
[1,] -524.9048 -1.047619 7.619048    5
```

```
coef(fit1.mult)
```

```
(Intercept)    Air.Flow  Water.Temp  Acid.Conc.
-524.904762   -1.047619    7.619048    5.000000
```

1.2 Form 2

In this form, we have the explanatory variables in a matrix on the right of our parameter matrix as in Form 1b but we arrange everything a little differently:

$$\begin{bmatrix} stack.loss_1 \\ stack.loss_2 \\ stack.loss_3 \\ stack.loss_4 \end{bmatrix} = \begin{bmatrix} \alpha & \beta & 0 & 0 & 0 \\ \alpha & 0 & \beta & 0 & 0 \\ \alpha & 0 & 0 & \beta & 0 \\ \alpha & 0 & 0 & 0 & \beta \end{bmatrix} \begin{bmatrix} 1 \\ air_1 \\ air_2 \\ air_3 \\ air_4 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} \tag{1.9}$$

Work through the matrix algebra to make sure you understand why Equation 1.9 is the same as Equation 1.1 for all the i data points together.

We will write Form 2 succinctly as

$$\mathbf{y} = \mathbf{Zx} + \mathbf{e} \tag{1.10}$$

1.2.1 Form 2 with multiple explanatory variables

The $\mathbf{x}$ is a column vector of the explanatory variables. If we have more explanatory variables, we add them to the column vector at the bottom. So if we had air flow, water temperature and acid concentration as explanatory variables, $\mathbf{x}$ looks like

$$\begin{bmatrix} 1 \\ air_1 \\ air_2 \\ air_3 \\ air_4 \\ water_1 \\ water_2 \\ water_3 \\ water_4 \\ acid_1 \\ acid_2 \\ acid_3 \\ acid_4 \end{bmatrix} \tag{1.11}$$

Add columns to the $\mathbf{Z}$ matrix for each new variable.

$$\begin{bmatrix} \alpha & \beta_1 & 0 & 0 & 0 & \beta_2 & 0 & 0 & 0 & \beta_3 & 0 & 0 & 0 \\ \alpha & 0 & \beta_1 & 0 & 0 & 0 & \beta_2 & 0 & 0 & 0 & \beta_3 & 0 & 0 \\ \alpha & 0 & 0 & \beta_1 & 0 & 0 & 0 & \beta_2 & 0 & 0 & 0 & \beta_3 & 0 \\ \alpha & 0 & 0 & 0 & \beta_1 & 0 & 0 & 0 & \beta_2 & 0 & 0 & 0 & \beta_3 \end{bmatrix} \tag{1.12}$$

The number of rows of $\mathbf{Z}$ is always n, the number of rows of $\mathbf{y}$, because the number of rows on the left and right of the equal sign must match. The number of columns in $\mathbf{Z}$ is determined by the size of $\mathbf{x}$. If there is an intercept, there is a 1 in $\mathbf{x}$. Then each explanatory variable (like air flow and wind) appears n times. So if the number of explanatory variables is k, the number of columns in $\mathbf{Z}$ is $1+k\times n$ if there is an intercept term and $k\times n$ if there is not.

1.2.2 When does Form 2 arise?

Form 2 is similar to how multivariate time-series models are typically written for reading by humans (on a whiteboard or paper). In these models, we see equations like this:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}_t = \begin{bmatrix} \beta_a & \beta_b \\ \beta_a & 0.1 \\ \beta_b & \beta_a \\ 0 & \beta_a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_t + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}_t \tag{1.13}$$

In this case, $\mathbf{y}_t$ is the set of 4 observations at time t and $\mathbf{x}_t$ is the set of 2 explanatory variables at time t. The $\mathbf{Z}$ is showing how we are modeling the

effects of x_1 and x_2 on the ys. Notice that the effects are not consistent across the x and y. This model would not be possible to fit with `lm` but will be easy to fit with MARSS (and MARSS-Bayes).

1.2.3 Solving for the parameters for Form 2*

You can just skim this section if you want but make sure you carefully look at the code in 1.2.4. You will need to adapt that for the homework. Though you will not need any of the math discussed here for the course, this section will help you practice matrix multiplication and will introduce you to 'permutation' matrices which will be handy in many other contexts.

To solve for α and β, we need our parameters in a column matrix like so $\left[\begin{smallmatrix}\alpha\\ \beta\end{smallmatrix}\right]$. We do this by rewritting $\mathbf{Zx}$ in Equation 1.10 in 'vec' form: if $\mathbf{Z}$ is a $n \times m$ matrix and $\mathbf{x}$ is a matrix with 1 column and m rows, then $\mathbf{Zx} = (\mathbf{x}^\top \otimes \mathbf{I}_n)\,\mathrm{vec}(\mathbf{Z})$. The symbol $\otimes$ means Kronecker product and just ignore it since you'll never see it again in our course (or google 'kronecker product' if you are curious). The "vec" of a matrix is that matrix rearranged as a single column:

$$\mathrm{vec}\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} = \begin{bmatrix}1\\ 3\\ 2\\ 4\end{bmatrix} \tag{1.14}$$

Notice how you just take each column one by one and stack them under each other. In R, the vec is

```
A=matrix(1:6,nrow=2,byrow=TRUE)
vecA = matrix(A,ncol=1)
```

$\mathbf{I}_n$ is a $n \times n$ identity matrix, a diagonal matrix with all 0s on the off-diagonals and all 1s on the diagonal. In R, this is simply `diag(n)`.

To show how we solve for α and β, let's use an example with only 3 data points so Equation 1.9 becomes:

$$\begin{bmatrix}stack.loss_1\\ stack.loss_2\\ stack.loss_3\end{bmatrix} = \begin{bmatrix}\alpha & \beta & 0 & 0\\ \alpha & 0 & \beta & 0\\ \alpha & 0 & 0 & \beta\end{bmatrix} \begin{bmatrix}1\\ air_1\\ air_2\\ air_3\end{bmatrix} + \begin{bmatrix}e_1\\ e_2\\ e_3\end{bmatrix} \tag{1.15}$$

Using $\mathbf{Zx} = (\mathbf{x}^\top \otimes \mathbf{I}_n)\,\mathrm{vec}(\mathbf{Z})$, this means

$$\begin{bmatrix} \alpha & \beta & 0 & 0 \\ \alpha & 0 & \beta & 0 \\ \alpha & 0 & 0 & \beta \end{bmatrix} \begin{bmatrix} 1 \\ air_1 \\ air_2 \\ air_3 \end{bmatrix} = \left(\begin{bmatrix} 1 & air_1 & air_2 & air_3 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} \alpha \\ \alpha \\ \alpha \\ \beta \\ 0 \\ 0 \\ 0 \\ \beta \\ 0 \\ 0 \\ 0 \\ \beta \end{bmatrix} \tag{1.16}$$

We need to rewrite the vec($\mathbf{Z}$) as a 'permutation' matrix times $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$:

$$\begin{bmatrix} \alpha \\ \alpha \\ \alpha \\ \beta \\ 0 \\ 0 \\ 0 \\ \beta \\ 0 \\ 0 \\ 0 \\ \beta \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \mathbf{Pp} \tag{1.17}$$

where $\mathbf{P}$ is the permutation matrix and $\mathbf{p} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$. Thus,

$$\mathbf{y} = \mathbf{Zx} + \mathbf{e} = (\mathbf{x}^\top \otimes \mathbf{I}_n)\mathbf{P} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \mathbf{Mp} + \mathbf{e} \tag{1.18}$$

where $\mathbf{M} = (\mathbf{x}^\top \otimes \mathbf{I}_n)\mathbf{P}$. We can solve for $\mathbf{p}$, the parameters, using

$$(\mathbf{M}^\top \mathbf{M})^{-1} \mathbf{M}^\top \mathbf{y}$$

as before.

1.2.4 Code to solve for parameters in Form 2

In the homework, you will use the R code in this section to solve for the parameters in Form 2. Later when you are fitting multivariate time-series models, you will not solve for parameters this way but you will need to both construct $\mathbf{Z}$ matrices in R and read $\mathbf{Z}$ matrices. The homework will give you practice creating $\mathbf{Z}$ matrices in R.

```
#make your y and x matrices
y=matrix(dat$stack.loss, ncol=1)
x=matrix(c(1,dat$Air.Flow),ncol=1)
#make the Z matrix
require(MARSS)
n=nrow(dat) #number of rows in our data file
k=1
#Z has n rows and 1 col for intercept, and n cols for the n air data points
#a list matrix allows us to combine "characters" and numbers
Z=matrix(list(0),n,k*n+1)
Z[,1]="alpha"
diag(Z[1:n,1+1:n])="beta"
#this function creates that permutation matrix for you
P=MARSS:::convert.model.mat(Z)$free[,,1]
M=kronecker(t(x),diag(n))%*%P
solve(t(M)%*%M)%*%t(M)%*%y
```

```
             [,1]
alpha -11.6159170
beta    0.6412918
```

```
coef(lm(dat$stack.loss ~ dat$Air.Flow))
```

```
(Intercept) dat$Air.Flow
-11.6159170    0.6412918
```

Go through this code line by line at the R command line. Look at `Z`. It is a list matrix that allows you to combine numbers (the 0s) with character string (names of parameters). Look at the permutation matrix `P`. Try `MARSS:::convert.model.mat(Z)$free` and see that it returns a 3D matrix, which is why the `[,,1]` appears (to get us a 2D matrix). To use more data points, you can redefine `dat` to say `dat=stackloss` to use all 21 data points.

1.3 Groups of intercepts

Let's say that the odd numbered plants are in the north and the even numbered are in the south. We want to include this as a factor in our model that affects the intercept. Let's go back to just having air flow be our explanatory variable. Now if the plant is in the north our model is

$$stack.loss_i = \alpha_n + \beta air_i + e_i, \text{ where } e_i \sim \mathrm{N}(0, \sigma^2) \tag{1.19}$$

If the plant is in the south, our model is

$$stack.loss_i = \alpha_s + \beta air_i + e_i, \text{ where } e_i \sim \mathrm{N}(0, \sigma^2) \tag{1.20}$$

We'll add north/south as a factor called 'reg' (region) to our dataframe:

```
dat = cbind(dat, reg=rep(c("n","s"),n)[1:n])
dat
```

```
  Air.Flow Water.Temp Acid.Conc. stack.loss reg
1       80         27         89         42   n
2       80         27         88         37   s
3       75         25         90         37   n
4       62         24         87         28   s
```

And we can easily fit this model with `lm`.

```
fit2 = lm(stack.loss ~ -1 + Air.Flow + reg, data=dat)
coef(fit2)
```

```
 Air.Flow       regn       regs
0.5358166 -2.0257880 -5.5429799
```

The -1 is added to the `lm` call to get rid of α. We just want the α_n and α_s intercepts coming from our regions.

1.3.1 North/South intercepts in Form 1

Written in matrix form, Form 1 for this model is

$$\begin{bmatrix} stack.loss_1 \\ stack.loss_2 \\ stack.loss_3 \\ stack.loss_4 \end{bmatrix} = \begin{bmatrix} air_1 & 1 & 0 \\ air_2 & 0 & 1 \\ air_3 & 1 & 0 \\ air_4 & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta \\ \alpha_n \\ \alpha_s \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} \tag{1.21}$$

Notice that odd plants get α_n and even plants get α_s. Use `model.matrix()` to see that this is the $\mathbf{Z}$ matrix that `lm` formed. Notice the matrix output by `model.matrix` looks exactly like $\mathbf{Z}$ in Equation 1.21.

```
Z=model.matrix(fit2)
Z[1:4,]
```

```
  Air.Flow regn regs
1       80    1    0
2       80    0    1
3       75    1    0
4       62    0    1
```

We can solve for the parameters using $\mathbf{x} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{y}$ as we did for Form 1 before by adding on the 1s and 0s columns we see in the $\mathbf{Z}$ matrix in Equation 1.21. We could build this $\mathbf{Z}$ using the following R code:

```
Z=cbind(dat$Air.Flow,c(1,0,1,0),c(0,1,0,1))
colnames(Z)=c("beta","regn","regs")
```

Or just use `model.matrix()`. This will save time when models are more complex.

```
Z=model.matrix(fit2)
Z[1:4,]
```

```
  Air.Flow regn regs
1       80    1    0
2       80    0    1
3       75    1    0
4       62    0    1
```

Now we can solve for the parameters:

```
y=matrix(dat$stack.loss, ncol=1)
solve(t(Z)%*%Z)%*%t(Z)%*%y
```

```
               [,1]
Air.Flow  0.5358166
regn     -2.0257880
regs     -5.5429799
```

Compare to the output from `lm` and you will see it is the same.

```
coef(fit2)
```

```
 Air.Flow       regn       regs
0.5358166 -2.0257880 -5.5429799
```

1.3.2 North/South intercepts in Form 2

We would write this model in Form 2 as

$$\begin{bmatrix} stack.loss_1 \\ stack.loss_2 \\ stack.loss_3 \\ stack.loss_4 \end{bmatrix} = \begin{bmatrix} \alpha_n & \beta & 0 & 0 & 0 \\ \alpha_s & 0 & \beta & 0 & 0 \\ \alpha_n & 0 & 0 & \beta & 0 \\ \alpha_s & 0 & 0 & 0 & \beta \end{bmatrix} \begin{bmatrix} 1 \\ air_1 \\ air_2 \\ air_3 \\ air_4 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} = \mathbf{Zx} + \mathbf{e} \tag{1.22}$$

To estimate the parameters, we need to be able to write a list matrix that looks like **Z** in Equation 1.22. We can use the same code we used in Section 1.2.4 with **Z** changed to look like that in Equation 1.22.

```
y=matrix(dat$stack.loss, ncol=1)
x=matrix(c(1,dat$Air.Flow),ncol=1)
n=nrow(dat)
k=1
#list matrix allows us to combine numbers and character strings
Z=matrix(list(0),n,k*n+1)
```

```
Z[seq(1,n,2),1]="alphanorth"
Z[seq(2,n,2),1]="alphasouth"
diag(Z[1:n,1+1:n])="beta"
P=MARSS:::convert.model.mat(Z)$free[,,1]
M=kronecker(t(x),diag(n))%*%P
solve(t(M)%*%M)%*%t(M)%*%y
```

```
                [,1]
alphanorth -2.0257880
alphasouth -5.5429799
beta        0.5358166
```

Make sure you understand the code used to form the **Z** matrix. Also notice that `class(Z[1,3])="numeric"` while `class(Z[1,2])="character"`. This is important. `0` in R is a number while `"0"` would be a character (the name of a parameter).

1.4 Groups of β's

Now let's say that the plants have different owners, Sue and Aneesh, and we want to have β for the air flow effect vary by owner. If the plant is in the north and owned by Sue, the model is

$$stack.loss_i = \alpha_n + \beta_s air_i + e_i, \text{ where } e_i \sim \text{N}(0, \sigma^2) \tag{1.23}$$

If it is in the south and owned by Aneesh, the model is

$$stack.loss_i = \alpha_s + \beta_a air_i + e_i, \text{ where } e_i \sim \text{N}(0, \sigma^2) \tag{1.24}$$

You get the idea.

Now we need to add an operator variable as a factor in our stackloss dataframe. Plants 1,3 are run by Sue and plants 2,4 are run by Aneesh.

```
dat = cbind(dat, owner=c("s","a"))
dat
```

```
  Air.Flow Water.Temp Acid.Conc. stack.loss reg owner
1       80         27         89         42   n     s
2       80         27         88         37   s     a
3       75         25         90         37   n     s
4       62         24         87         28   s     a
```

Since the operator names can be replicated the length of our data set, R fills in the operator colmun by replicating our string of operator names to the right length, conveniently (or alarmingly).

We can easily fit this model with `lm` using the ":" notation.

```
coef(lm(stack.loss ~ -1 + Air.Flow:owner + reg, data=dat))
```

```
          regn            regs Air.Flow:ownera
         -38.0            -3.0             0.5
Air.Flow:owners
           1.0
```

Notice that we have 4 datapoints and are estimating 4 parameters. We are not going to be able to estimate any more parameters than data points. If we want to estimate any more, we'll need to use the fuller stackflow dataset (which has 21 data points).

1.4.1 Owner β's in Form 1

Written in Form 1, this model is

$$\begin{bmatrix} stack.loss_1 \\ stack.loss_2 \\ stack.loss_3 \\ stack.loss_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & air_1 \\ 0 & 1 & air_2 & 0 \\ 1 & 0 & 0 & air_3 \\ 0 & 1 & air_4 & 0 \end{bmatrix} \begin{bmatrix} \alpha_n \\ \alpha_s \\ \beta_a \\ \beta_s \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} = \mathbf{Zx} + \mathbf{e} \tag{1.25}$$

The air data have been written to the right of the 1s and 0s for north/south intercepts because that is how `lm` writes this model in Form 1 and I want to duplicate that (for teaching purposes). Also the β's are ordered to be alphabetical because `lm` writes the **Z** matrix like that.

Now our model is more complicated and using `model.matrix` to get our **Z** saves us a lot tedious matrix building.

```
fit3=lm(stack.loss ~ -1 + Air.Flow:owner + reg, data=dat)
Z=model.matrix(fit3)
Z[1:4,]
```

```
  regn regs Air.Flow:ownera Air.Flow:owners
1    1    0               0              80
2    0    1              80               0
3    1    0               0              75
4    0    1              62               0
```

Notice the matrix output by `model.matrix` looks exactly like **Z** in Equation 1.25 (ignore the attributes info). Now we can solve for the parameters:

```
y=matrix(dat$stack.loss, ncol=1)
solve(t(Z)%*%Z)%*%t(Z)%*%y
```

```
                 [,1]
regn            -38.0
regs             -3.0
Air.Flow:ownera   0.5
Air.Flow:owners   1.0
```

Compare to the output from `lm` and you will see it is the same.

1.4.2 Owner β's in Form 2

To write this model in Form 2, we just add subscripts to the β's in our Form 2 **Z** matrix:

$$\begin{bmatrix} stack.loss_1 \\ stack.loss_2 \\ stack.loss_3 \\ stack.loss_4 \end{bmatrix} = \begin{bmatrix} \alpha_n & \beta_s & 0 & 0 & 0 \\ \alpha_s & 0 & \beta_a & 0 & 0 \\ \alpha_n & 0 & 0 & \beta_s & 0 \\ \alpha_s & 0 & 0 & 0 & \beta_a \end{bmatrix} \begin{bmatrix} 1 \\ air_1 \\ air_2 \\ air_3 \\ air_4 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} = \mathbf{Zx} + \mathbf{e} \tag{1.26}$$

To estimate the parameters, we change the β's in our **Z** list matrix to have owner designations:

```
y=matrix(dat$stack.loss, ncol=1)
x=matrix(c(1,dat$Air.Flow),ncol=1)
n=nrow(dat)
k=1
Z=matrix(list(0),n,k*n+1)
Z[seq(1,n,2),1]="alpha.n"
Z[seq(2,n,2),1]="alpha.s"
diag(Z[1:n,1+1:n])=rep(c("beta.s","beta.a"),n)[1:n]
P=MARSS:::convert.model.mat(Z)$free[,,1]
M=kronecker(t(x),diag(n))%*%P
solve(t(M)%*%M)%*%t(M)%*%y
```

```
        [,1]
alpha.n -38.0
alpha.s  -3.0
beta.s    1.0
beta.a    0.5
```

The parameters estimates are the same, though β's are given in reversed order simply due to the way `convert.model.mat` is ordering the columns in Form 2's **Z**.

1.5 Seasonal effect as a factor

Let's imagine that the data were taken consecutively in time by quarter. We want to model the seasonal effect as an intercept change. We will drop all other effects for now. If the data were collected in quarter 1, the model is

$$stack.loss_i = \alpha_1 + e_i, \text{ where } e_i \sim \mathrm{N}(0, \sigma^2) \tag{1.27}$$

If collected in quarter 2, the model is

$$stack.loss_i = \alpha_2 + e_i, \text{ where } e_i \sim \mathrm{N}(0, \sigma^2) \tag{1.28}$$

etc.

We add a column to our dataframe to account for season:

```
dat = cbind(dat, qtr=paste(rep("qtr",n),1:4,sep=""))
dat
```

```
  Air.Flow Water.Temp Acid.Conc. stack.loss reg owner  qtr
1       80         27         89         42   n     s qtr1
2       80         27         88         37   s     a qtr2
3       75         25         90         37   n     s qtr3
4       62         24         87         28   s     a qtr4
```

And we can easily fit this model with `lm`.

```
coef(lm(stack.loss ~ -1 + qtr, data=dat))
```

```
qtrqtr1 qtrqtr2 qtrqtr3 qtrqtr4
     42      37      37      28
```

The -1 is added to the `lm` call to get rid of α. We just want the α_1, α_2, etc. intercepts coming from our quarters.

For comparison look at

```
coef(lm(stack.loss ~ qtr, data=dat))
```

```
(Intercept)     qtrqtr2     qtrqtr3     qtrqtr4
         42          -5          -5         -14
```

Why does it look like that when -1 is missing from the `lm` call? Where did the intercept for quarter 1 go and why are the other intercepts so much smaller?

1.5.1 Seasonal intercepts written in Form 1

Remembering that `lm` puts models in Form 1, look at the $\mathbf{Z}$ matrix for Form 1:

```
fit4=lm(stack.loss ~ -1 + qtr, data=dat)
Z=model.matrix(fit4)
Z[1:4,]
```

```
  qtrqtr1 qtrqtr2 qtrqtr3 qtrqtr4
1       1       0       0       0
2       0       1       0       0
3       0       0       1       0
4       0       0       0       1
```

Written in Form 1, this model is

$$\begin{bmatrix} stack.loss_1 \\ stack.loss_2 \\ stack.loss_3 \\ stack.loss_4 \end{bmatrix} = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} = \mathbf{Zx} + \mathbf{e} \tag{1.29}$$

Compare to the model that `lm` is using when the intercept included. What does this model look like written in matrix form?

```
fit5=lm(stack.loss ~ qtr, data=dat)
Z=model.matrix(fit5)
Z[1:4,]
```

```
  (Intercept) qtrqtr2 qtrqtr3 qtrqtr4
1           1       0       0       0
2           1       1       0       0
3           1       0       1       0
4           1       0       0       1
```

1.5.2 Seasonal intercepts written in Form 2

We do not need to add 1s and 0s to our $\mathbf{Z}$ matrix in Form 2; we just add subscripts to our intercepts like we did when we had north-south intercepts. In this model, we do not have any explanatory variables (except intercept) so our $\mathbf{x}$ is just a 1×1 matrix:

$$\begin{bmatrix} stack.loss_1 \\ stack.loss_2 \\ stack.loss_3 \\ stack.loss_4 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} = \mathbf{Z}\mathbf{x} + \mathbf{e} \tag{1.30}$$

1.6 * Seasonal effect plus other explanatory variables

With our 4 data points, we are limited to estimating 4 parameters. Let's use the full 21 data points so we can estimate some more complex models. We'll add an owner variable and a quarter variable to the stackloss dataset.

```
data(stackloss)
fulldat=stackloss
n=nrow(fulldat)
fulldat=cbind(fulldat,
          owner=rep(c("sue","aneesh","joe"),n)[1:n],
          qtr=paste("qtr",rep(1:4,n)[1:n],sep=""),
          reg=rep(c("n","s"),n)[1:n])
```

Let's fit a model where there is only an effect of air flow, but that effect varies by owner and by quarter. We also want a different intercept for each quarter. So if datapoint i is from quarter j on a plant owned by owner k, the model is

$$stack.loss_i = \alpha_j + \beta_{j,k} air_i + e_i \tag{1.31}$$

So there there are 4×3 β's (4 quarters and 3 owners) and 4 α's (4 quarters).

With `lm`, we fit the model as:

```
fit7 = lm(stack.loss ~ -1 + qtr + Air.Flow:qtr:owner, data=fulldat)
```

Take a look at **Z** for Form 1 using `model.matrix(Z)`. It's not shown since it is large:

```
model.matrix(fit7)
```

The **x** will be

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \beta_{1,a} \\ \beta_{2,a} \\ \beta_{3,a} \\ \cdots \end{bmatrix}$$

Take a look at the model matrix that `lm` is using and make sure you understand how **Zx** produces Equation 1.31.

```
Z=model.matrix(fit7)
```

For Form 2, our **Z** size doesn't change; number of rows is n (the number data points) and number of columns is 1 (for intercept) plus the number of explanatory variables times n. So in this case, we only have one explanatory variable (air flow) so **Z** has 1+21 columns. To allow the intercept to vary by quarter, we use α_1 in the rows of **Z** where the data is from quarter 1, use α_2 where the data is from quarter 2, etc. Similarly we use the appropriate $\beta_{j,k}$ depending on the quarter and owner for that data point.

We could construct **Z**, **x** and **y** for Form 2 using

```
y=matrix(fulldat$stack.loss, ncol=1)
x=matrix(c(1,fulldat$Air.Flow),ncol=1)
n=nrow(fulldat)
k=1
Z=matrix(list(0),n,k*n+1)
#give the intercepts names based on qtr
Z[,1]=paste(fulldat$qtr)
#give the betas names based on qtr and owner
diag(Z[1:n,1+1:n])=paste("beta",fulldat$qtr,fulldat$owner,sep=".")
P=MARSS:::convert.model.mat(Z)$free[,,1]
M=kronecker(t(x),diag(n))%*%P
solve(t(M)%*%M)%*%t(M)%*%y
```

Note, the estimates are the same as for `lm` but are not listed in the same order.

Make sure to look at the **Z** and **x** for the models and that you understand why they look like they do.

1.7 * Models with confounded parameters

Try adding region as another factor in your model along with quarter and fit with `lm`:

```
coef(lm(stack.loss ~ -1 + Air.Flow + reg + qtr, data=fulldat))
```

```
  Air.Flow       regn       regs    qtrqtr2    qtrqtr3
  1.066524 -49.024320 -44.831760  -3.066094   3.499428
   qtrqtr4
        NA
```

The estimate for quarter 1 is gone (actually it was set to 0) and the estimate for quarter 4 is NA. Look at the **Z** matrix for Form 1 and see if you can figure out the problem. Try also writing out the model for the 1st plant and you'll see what part of the problem is and why the estimate for quarter 1 is fixed at 0.

```
fit=lm(stack.loss ~ -1 + Air.Flow + reg + qtr, data=fulldat)
Z=model.matrix(fit)
```

But why is the estimate for quarter 4 equal to NA? What if the ordering of north and south regions was different, say 1 through 4 north, 5 through 8 south, 9 through 12 north, etc?

```
fulldat2=fulldat
fulldat2$reg2 = rep(c("n","n","n","n","s","s","s","s"),3)[1:21]
fit=lm(stack.loss ~ Air.Flow + reg2 + qtr, data=fulldat2)
coef(fit)
```

```
(Intercept)    Air.Flow       reg2s     qtrqtr2     qtrqtr3
-45.6158421   1.0407975  -3.5754722   0.7329027   3.0389763
    qtrqtr4
  3.6960928
```

Now an estimate for quarter 4 appears.

The problem is two-fold. First by having both region and quarter intercepts, we created models where 2 intercepts appear for one i model and we cannot estimate both. `lm` helps us out by setting one of the factor effects to 0. It will chose the first alphabetically. But as we saw with the model where odd numbered plants were north and even numbered were south, we can still have a situation where one of the intercepts is non-identifiable. `lm` helps us out by alerting us to the problem by setting one to NA.

Once you start writing your own Jags code or using MARSS, you will need to make sure that all your parameters are identifiable. If they are not, your code will simply 'chase its tail'. The code will generally take forever to converge or if you did not try different starting conditions, it may look like it converged but actually the estimates for the confounded parameters

are meaningless. So you will need to think carefully about the model you are fitting and consider if there are multiple parameters measuring the same thing (for example 2 intercept parameters).

Problems

For the homework questions, we will using part of the `airquality` data set in R. Load that as

```
library(datasets)
data(airquality)
#remove any rows with NAs omitted.
airquality=na.omit(airquality)
#make Month a factor (i.e., the Month number is a name rather than a number)
airquality$Month=as.factor(airquality$Month)
#add a region factor
airquality$region = rep(c("north","south"),60)[1:111]
#Only use 5 data points for the homework so you can show the matrices easily
homeworkdat = airquality[1:5,]
```

1.1 Using Form 1 $\mathbf{y} = \mathbf{Z}\mathbf{x} + \mathbf{e}$, write the model being fit by this command
```
fit=lm(Ozone ~ Wind + Temp, data=homeworkdat)
```

1.2 Build the $\mathbf{y}$ and $\mathbf{Z}$ matrices for the above model in R and solve for $\mathbf{x}$ (the parameters). Show that they match what you get from the first `lm()` call.

1.3 If you added -1 to your `lm` call in question 1, what changes in your model?
```
fit=lm(Ozone ~ -1 + Wind + Temp, data=homeworkdat)
```

1.4 Write the model for question 1 in Form 2. Adapt the code from subsection 1.2.4 to solve for the parameters. You will need to contruct new Z, y and x in the code.

1.5 Model the ozone data with only a region (north/south) effect:
```
fit=lm(Ozone ~ -1 + region, data=homeworkdat)
```
Write this model in Form 1b (not Form 1) show that you can solve for the parameter values you get from the `lm` call.

1.6 Write the model in question 5 in Form 2. Show that you can solve for the parameter values you get from the `lm` call. Again to do this, you adapt the code from subsection 1.2.4.

1.7* Write the model below in Form 2.
```
fit=lm(Ozone ~ Temp:region, data=homeworkdat)
```

1.8* Using the airquality dataset with 111 data points, write the model below in Form 2 and solve for the parametes by adapting code from subsection 1.2.4.
```
fit=lm(Ozone ~ -1 + Temp:region + Month, data=airquality)
```

Solutions Chapter 1

Data Set Up

```
library(datasets)
data(airquality)
#remove any rows with NAs omitted.
airquality=na.omit(airquality)
#make Month a factor (i.e., the Month number is a name rather than a number)
airquality$Month=as.factor(airquality$Month)
#add a region factor
airquality$region = rep(c("north","south"),60)[1:111]
#Only use 5 data points for the homework so you can show the matrices easily
homeworkdat = airquality[1:5,]
```

Problem 1

Using Form 1 $\mathbf{y} = \mathbf{Z}\mathbf{x} + \mathbf{e}$, write the model being fit by this command

```
fit=lm(Ozone ~ Wind + Temp, data=homeworkdat)
```

The model is $Ozone_i = \alpha + \beta_w Wind_i + \beta_t Temp_i + e_i$. Form 1 for this model is:

$$\begin{bmatrix} Ozone_1 \\ Ozone_2 \\ Ozone_3 \\ Ozone_4 \\ Ozone_5 \end{bmatrix} = \begin{bmatrix} 1 & Wind_1 & Temp_1 \\ 1 & Wind_2 & Temp_2 \\ 1 & Wind_3 & Temp_3 \\ 1 & Wind_4 & Temp_4 \\ 1 & Wind_5 & Temp_5 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta_w \\ \beta_t \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{bmatrix}$$

Problem 2

Build the y and Z matrices for the above model in R and solve for x (the parameters). Show that they match what you get from the 'lm' call.

Here are the **y** and **Z**. You can see they match the **y** and **Z** in the equation above:

```
y=matrix(homeworkdat$Ozone, ncol=1)
Z=cbind(1, homeworkdat$Wind, homeworkdat$Temp)
y
```

```
     [,1]
[1,]   41
[2,]   36
[3,]   12
[4,]   18
[5,]   23
```

```
Z
```

```
     [,1] [,2] [,3]
[1,]    1  7.4   67
[2,]    1  8.0   72
[3,]    1 12.6   74
[4,]    1 11.5   62
[5,]    1  8.6   65
```

Next we solve for **x** and show it matches what we get from 'lm'. This uses the code in section 1.1.1

```
solve(t(Z)%*%Z)%*%t(Z)%*%y
```

```
           [,1]
[1,] 56.6219201
[2,] -4.9776707
[3,]  0.2538717
```

```
coef(lm(Ozone ~ Wind + Temp, data=homeworkdat))
```

```
(Intercept)        Wind        Temp
 56.6219201  -4.9776707   0.2538717
```

Problem 3

If you added -1 to your 'lm' call in question 1, what changes in your model?

First run the 'lm' call and see what changed. The intercept (α) is dropped.

```
fit=lm(Ozone ~ -1 + Wind + Temp, data=homeworkdat)
fit
```

```
Call:
lm(formula = Ozone ~ -1 + Wind + Temp, data = homeworkdat)

Coefficients:
  Wind    Temp
-4.650   1.037
```

The model is now $Ozone_i = \beta_w Wind_i + \beta_t Temp_i + e_i$. To get rid of the α we drop the 1s column:

$$\begin{bmatrix} Ozone_1 \\ Ozone_2 \\ Ozone_3 \\ Ozone_4 \\ Ozone_5 \end{bmatrix} = \begin{bmatrix} Wind_1 & Temp_1 \\ Wind_2 & Temp_2 \\ Wind_3 & Temp_3 \\ Wind_4 & Temp_4 \\ Wind_5 & Temp_5 \end{bmatrix} \begin{bmatrix} \beta_w \\ \beta_t \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{bmatrix}$$

Problem 4, part 1

Write the model for question 1 in Form 2.

In Form 2, $\mathbf{y} = \mathbf{Z}\mathbf{x} + \mathbf{e}$ and the explanatory variables appear in the $\mathbf{x}$ as a column vector (a matrix with one column). So $\mathbf{x}$ looks like this

$$\begin{bmatrix} 1 \\ Wind_1 \\ \dots \\ Wind_5 \\ Temp_1 \\ \dots \\ Temp_5 \end{bmatrix}$$

Once you get that, then you know that $\mathbf{Z}$ is a $5 \times (1+5+5)$ matrix. The first column of $\mathbf{Z}$ is the α. The next 5 columns of $\mathbf{Z}$ is a 5×5 diagonal matrix with β_w on the diagonal, just like in Equation 1.12. For the next explanatory variable, Temperature, we tack on another 5×5 diagonal matrix; this time with β_t on the diagonal. That's all you needed to say for homework; just to show that you figured out what the form of the $\mathbf{y}$, $\mathbf{Z}$ and $\mathbf{x}$ look like.

If you wanted to write it out in math form, you could show Z as:

$$\begin{bmatrix} \text{column of} & 5 \times 5 \text{ diagonal matrix} & 5 \times 5 \text{ diagonal matrix} \\ \alpha & \text{with } \beta_w \text{ on the diagonal} & \text{with } \beta_t \text{ on the diagonal} \end{bmatrix}$$

or

$$\begin{bmatrix} \alpha & \beta_w & \dots & 0 & \beta_t & \dots & 0 \\ & & \ddots & & & \ddots & \\ \dots & \dots & & \dots & \dots & & \dots \\ \alpha & 0 & \dots & \beta_w & 0 & \dots & \beta_t \end{bmatrix}$$

Problem 4, part 2

Solve for the parameters.

To do the 2nd part, you adapt the code from subsection 1.2.3 to solve for the parameters. You will need to contruct new **Z**, **y** and **x** in the code.

The **y** and **x** are easy:

```
y=matrix(homeworkdat$Ozone, ncol=1)
x=matrix(c(1, homeworkdat$Wind, homeworkdat$Temp), ncol=1)
```

We adapt the code for making **Z** from 1.2.3:

```
#we know that Z is a 5 x (1+5+5) matrix
#n is the number of data points
n=5
#nrows = n; what about ncol?,  ncol is 1 (alpha) + n (Wind) + n (Temp)
Z=matrix(list(0),n,1+n+n)
#the first column is alpha
Z[,1]="alpha"
#columns 2:112 are a diagonal matrix with betaw on the diagonal
diag(Z[,2:(n+1)])="betaw"
#columns 113:223 are a diagonal matrix with betat on the diagonal
diag(Z[,(n+2):(2*n+1)])="betat"
```

Now we can solve for Z:

```
require(MARSS)
P=MARSS:::convert.model.mat(Z)$free[,,1]
M=kronecker(t(x),diag(n))%*%P
solve(t(M)%*%M)%*%t(M)%*%y
```

```
            [,1]
alpha 56.6219201
betaw -4.9776707
betat  0.2538717
```

```
coef(lm(Ozone ~ Wind + Temp, data=homeworkdat))
```

```
(Intercept)        Wind        Temp
 56.6219201  -4.9776707   0.2538717
```

Problem 5, part 1

Model the ozone data with only a region effect. Write this in Form 1b.

First make sure you understand what model is being fit by the 'lm' call:

```
fit=lm(Ozone ~ -1 + region, data=homeworkdat)
fit
```

```
Call:
lm(formula = Ozone ~ -1 + region, data = homeworkdat)

Coefficients:
regionnorth  regionsouth
      25.33        27.00
```

The model is $Ozone_i = \alpha_j + e_i$ where j is the region the measurement was taken in. This is an intercept (or level) only model where the intercept is determined by the region (north or south).

We want to write that model in matrix form using Form 1b, $\mathbf{y} = \mathbf{Dd} + \mathbf{e}$. Eqn 1.21 shows you how to do this for Form 1, except we do not have explanatory variables besides region so we do not have a column with something like 'air' in it. Form 1b is the transpose of Form 1.

Matrix $\mathbf{y}$ is:

$$\mathbf{y} = \begin{bmatrix} Ozone_1 & Ozone_2 & \dots & Ozone_5 \end{bmatrix}$$

Matrix $\mathbf{D}$ is:

$$\mathbf{D} = \begin{bmatrix} \alpha_n & \alpha_s \end{bmatrix}$$

For $\mathbf{d}$, we need to know that each column of $\mathbf{d}$ is for a different data point i and tells us what region that data point is from. So $\mathbf{d}$ has 2 rows, one for each region. If there is a 1 in row 1, it means that data point came from the north. If there is a 1 in row 2, it means that data point came from the south.

Let's look at the regions

```
homeworkdat$region
```

```
[1] "north" "south" "north" "south" "north"
```

So the first measurement is from the north, next from south, then north, ... $\mathbf{Z}$ looks like this

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

The easy way to figure out $\mathbf{d}$ is to remember that Form 1b is just the transpose of Form 1, so $\mathbf{d} = \mathbf{Z}^\top$. So let R show you what $\mathbf{d}$ is:

```
t(model.matrix(fit))
```

```
            1 2 3 4 7
regionnorth 1 0 1 0 1
regionsouth 0 1 0 1 0
attr(,"assign")
[1] 1 1
attr(,"contrasts")
attr(,"contrasts")$region
[1] "contr.treatment"
```

Problem 5, part 2

Show that you can solve for the parameters.

Section 1.1.4 shows you how to solve for the parameters when the model is in form 1b. We need to make **y** and **d** in R.

```
y=matrix(homeworkdat$Ozone, nrow=1)
```

We could form **d** like so

```
ndatapoints=5
d=matrix(0,2,ndatapoints)
for(i in 1:ndatapoints) d[ifelse(airquality$region[i]=="north",1,2),i]=1
```

or this

```
d=rbind(
  as.numeric(airquality$region=="north"),
  as.numeric(airquality$region=="south")
)
```

or just use the output from our 'lm' call since R's 'lm' is forming the **d** matrix too:

```
d=t(model.matrix(fit))
```

Now we solve for the parameters using the code in section 1.1.4:

```
y%*%t(d)%*%solve(d%*%t(d))
```

```
     regionnorth regionsouth
[1,]    25.33333          27
```

```
coef(lm(Ozone ~ -1 + region, data=homeworkdat))
```

```
regionnorth regionsouth
   25.33333    27.00000
```

Problem 6, part 1

Write the model in question 5 in Form 2

The reason we had to use a matrix with 1s and 0s to tell our matrix math what α to use is that Form 1 and Form 1b have the parameters appearing once in a column vector or a row vector. So we need a matrix with 1s and 0s to say 'where' to put the αs.

In Form 2, we can have the parameters just repeat in our matrix. This is how we'd write the model on the whiteboard. We'd just have the α (intercept) column have multiple αs in it. Section 1.3.2 shows you how to have different intercepts in Form 2. The model in question 5 is just like in section 1.3.2, but without the 'air'. It just has different monthly intercepts.

The model is:

$$\begin{bmatrix} Ozone_1 \\ Ozone_2 \\ Ozone_3 \\ Ozone_4 \\ Ozone_5 \end{bmatrix} = \begin{bmatrix} \alpha_n \\ \alpha_s \\ \alpha_n \\ \alpha_s \\ \alpha_n \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{bmatrix} = \mathbf{Zx} + \mathbf{e}$$

Problem 6, part 2

Solve for the parameters Solve for the parameters

To do this we write **y**, **Z**, and **x** in R and use the code in section 1.2.3.

```
#the number of data points
n=5
y=matrix(homeworkdat$Ozone, ncol=1)
x=matrix(1)
Z=matrix(paste("alpha",homeworkdat$region, sep="."),ncol=1)
```

Then we solve for the parameters:

```
require(MARSS)
P=MARSS:::convert.model.mat(Z)$free[,,1]
M=kronecker(t(x),diag(n))%*%P
solve(t(M)%*%M)%*%t(M)%*%y
```

```
                [,1]
alpha.north 25.33333
alpha.south 27.00000
```

```
coef(lm(Ozone ~ -1 + region, data=homeworkdat))
```

```
regionnorth regionsouth
   25.33333    27.00000
```

Problem 7

Write the model below in Form 2 and solve for the parameters

```
fit=lm(Ozone ~ Temp:region, data=homeworkdat)
```

The first step is to write out what model is being fit. The model is

$$Ozone_i = \alpha + \beta_n Temp_i + e_i$$

if i is from the north and

$$Ozone_i = \alpha + \beta_s Temp_i + e_i$$

if i is from the south. So each region has the same intercept α but we are including a linear temperature effect that is different for each region.

This is just like Equation 1.26 but with one α:

$$\begin{bmatrix} Ozone_1 \\ Ozone_2 \\ Ozone_3 \\ Ozone_4 \\ Ozone_5 \end{bmatrix} = \begin{bmatrix} \alpha & \beta_n & 0 & 0 & 0 & 0 \\ \alpha & 0 & \beta_s & 0 & 0 & 0 \\ \alpha & 0 & 0 & \beta_n & 0 & 0 \\ \alpha & 0 & 0 & 0 & \beta_s & 0 \\ \alpha & 0 & 0 & 0 & 0 & \beta_n \end{bmatrix} \begin{bmatrix} 1 \\ Temp_1 \\ Temp_2 \\ Temp_3 \\ Temp_4 \\ Temp_5 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_{111} \end{bmatrix} = \mathbf{Zx} + \mathbf{e}$$

Solving for this is a matter of writing $\mathbf{y}$, $\mathbf{Z}$, and $\mathbf{x}$ in R.

```
n=5 #the number of data points
y=matrix(homeworkdat$Ozone, ncol=1)
x=matrix(c(1, homeworkdat$Temp),ncol=1)
#Set up Z; it is 5 x 6
Z=matrix(list(0),n,n+1)
#make the alpha column
Z[,1]="alpha"
#make the diagonal of columns 2:6 equal to the betas
diag(Z[,2:(1+n)])=paste("beta",homeworkdat$region, sep=".")
```

Then we solve for the parameters:

```
require(MARSS)
P=MARSS:::convert.model.mat(Z)$free[,,1]
M=kronecker(t(x),diag(n))%*%P
solve(t(M)%*%M)%*%t(M)%*%y
```

```
                 [,1]
alpha      23.86230424
beta.north  0.01510679
beta.south  0.05654090
```

```
coef(lm(Ozone ~ Temp:region, data=homeworkdat))
```

```
    (Intercept) Temp:regionnorth Temp:regionsouth
    23.86230424       0.01510679       0.05654090
```

Problem 8**

Using the airquality dataset with 111 data points, write the model below in Form 2 and solve for the parameters.

```
fit=lm(Ozone ~ -1 + Temp:region + Month, data=airquality)
```

The first step is to write out what model is being fit.

$$Ozone_i = \alpha_j + \beta_k Temp_i + e_i$$

If i is from the j-th month, the intercept is α_j. If i is from the north, $beta_k$ is β_n. If it is from the south, $beta_k$ is β_s.

Next, let's look at region and Month.

```
airquality$region[1:10]
```

```
 [1] "north" "south" "north" "south" "north" "south" "north"
 [8] "south" "north" "south"
```

```
airquality$Month
```

```
  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6
 [28] 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 [55] 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 [82] 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
[109] 9 9 9
Levels: 5 6 7 8 9
```

Region is alternating north/south and month is grouped.

So in matrix form our model looks like

$$\begin{bmatrix} Ozone_1 \\ Ozone_2 \\ \dots \\ Ozone_{111} \end{bmatrix} = \begin{bmatrix} \alpha_5 & \beta_n & 0 & \dots & 0 & 0 \\ \alpha_5 & 0 & \beta_s & \dots & 0 & 0 \\ \dots & \dots & \dots & \ddots & \dots & 0 \\ \alpha_9 & 0 & 0 & \dots & \beta_s & 0 \\ \alpha_9 & 0 & 0 & \dots & 0 & \beta_n \end{bmatrix} \begin{bmatrix} 1 \\ Temp_1 \\ Temp_2 \\ \dots \\ Temp_{111} \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_{111} \end{bmatrix} = \mathbf{Zx} + \mathbf{e}$$

Solving for this is a matter of writing $\mathbf{y}$, $\mathbf{Z}$, and $\mathbf{x}$ in R. We don't have to create character strings for our parameter names. We just use the text in our data.frame and use the `paste()` function.

```
n=111 #the number of data points
y=matrix(airquality$Ozone, ncol=1)
x=matrix(c(1, airquality$Temp),ncol=1)
#Set up Z; it is 111 x 112
Z=matrix(list(0),n,n+1)
#make the alpha column
Z[,1]=paste("alpha",airquality$Month, sep="")
#make the diagonal of columns 2:112 equal to the betas
diag(Z[,2:(1+n)])=paste("beta",airquality$region, sep=".")
```

Then we solve for the parameters:

```
require(MARSS)
P=MARSS:::convert.model.mat(Z)$free[,,1]
M=kronecker(t(x),diag(n))%*%P
solve(t(M)%*%M)%*%t(M)%*%y
```

```
                 [,1]
alpha5     -160.254743
alpha6     -187.708278
alpha7     -173.611725
alpha8     -172.152400
alpha9     -181.929911
beta.north    2.790616
beta.south    2.758379
```

```
coef(lm(Ozone ~ -1 + Temp:region + Month, data=airquality))
```

```
          Month5           Month6           Month7
    -160.254743      -187.708278      -173.611725
          Month8           Month9 Temp:regionnorth
    -172.152400      -181.929911         2.790616
Temp:regionsouth
        2.758379
```