CHAPTER 11

Individual covariates

In most of the analyses we've looked at so far in this book, we've partitioned variation in one or more parameters among different levels of what are commonly referred to as 'classification' factors. For example, comparing survival probabilities between male and female individuals (where 'sex' is the classification factor), good and poor breeding colonies (where 'colony' is the classification factor), among age-classes, and so on.

However, in many cases, there may be one or more factors which you might think are important determinants of variation among parameters which do not have natural 'classification' levels. For example, consider body size. It is often hypothesized that survival of individuals may be significantly influenced by individual differences in body size. While it is possible to take individuals and classify them as 'large', 'medium' or 'small' (based on some criterion), such classifications are artificial, and arbitrary. For a continuous covariate such as body size, there are an infinite number of possible classification levels you might create. And, your results may depend upon how many classification levels for body size (or some other continuous factor) you use, and exactly where these levels fall.

As such, it would be preferable to use the real, continuous values for body size in your analysis – each individual in the data set has a particular body size, so you want to constrain the estimates of the various parameters in your model to be functions of one or more continuous individual covariates. The use of the word 'covariate' might tweak some memory cells – think 'analysis of covariance' (ANCOVA), which looks at the influence of one or more continuous covariates on some response variable, conditional on one or more classification variables. For example, suppose you have measured the resting pulse rate for male and female children in a given classroom. You believe that pulse rate is influenced by the sex of the individual, and their body weight. So, you might set up a linear model where `SEX` is entered as a classification variable (with 2 levels: male and female), and `WEIGHT` is entered as a continuous linear covariate. You might also include an interaction term between `SEX` and `WEIGHT`.

In analysis of data from marked individuals, you essentially do much the same thing. Of course, there are a couple of 'extra steps' in the process, but essentially, you use the same mechanics for model building and model selection we've already considered elsewhere in the book. The major differences concern: data formatting, modifying the design matrix, and reconstituting parameter estimates. We will introduce the basic ideas with a series of worked examples.

Before we begin, though, it is important that you fully understand the semantic and functional distinction between an '*individual covariate*' (a covariate that applies to that *individual*; e.g., body size at birth), and an '*environmental*' or '*group*' covariate (a covariate which applies to *all individuals* encountered at a particular casion or over a particular interval; e.g., weather). Further, we need to distinguish between a '*fixed*' individual covariate (i.e., a covariate that does not change over time), and a '*time-varying*' covariate (which is self-explanatory).

11.1. ML estimation and individual covariates

Conceptually, the idea behind modeling survival or recapture (or any other parameter) as a function of an individual covariate isn't particularly difficult. It stems from the realization that it is possible to write the likelihood as a product of individual 'contributions' to the overall likelihood. Consider the following example. Suppose you have 8 individuals, which you mark and release. You go out next year, and find 3 of them alive (we'll ignore issues of encounter probability and so forth for the moment). We know from Chapter 1 that the maximum likelihood estimate of the survival probability S is simply $(3/8) = 0.375$. More formally, the (binomial) likelihood of observing 3 survivors out of 8 individuals marked and released is given as (where $Y = 3$, and $N = 8$):

$$\mathcal{L}(S \mid \text{data}) = \binom{N}{Y} S^Y (1-S)^{N-Y}.$$

Or, dropping the binomial probability term (which is a constant, and not a function of the parameter – see Chapter 1):

$$\mathcal{L}(S \mid \text{data}) \propto S^Y (1-S)^{N-Y}.$$

If we let $Q = (1 - S)$, then we could re-write this likelihood as

$$\mathcal{L}(S \mid \text{data}) \propto S^Y Q^{N-Y} = S^3 Q^5.$$

We could rewrite this likelihood expression as

$$\mathcal{L}(S \mid \text{data}) \propto S^3 Q^5 = (S.S.S).(Q.Q.Q.Q.Q) = \prod_{i=1}^{3} S_i \prod_{i=4}^{8} Q_i.$$

Alternatively, we might define a variable a, which we use to indicate whether or not the animal is found alive ($a = 1$) or dead ($a = 0$). Thus, we could write the likelihood for the i^{th} individual as

$$\mathcal{L}(S \mid N, \{a_1, a_2, \ldots, a_8\}) \propto \prod_{i=1}^{8} S^{a_i} Q^{(1-a_i)}.$$

Try it and confirm this is correct. Let S = the MLE = 0.375. Then, $(0.375)^3(1-0.375)^5 = 0.00503$, which is equivalent to

$$\begin{aligned}
&(0.375)^1(0.625)^{(1-1)}(0.375)^1(0.625)^{(1-1)}(0.375)^1(0.625)^{(1-1)} \\
&\times(0.375)^0(0.625)^{(1-0)}(0.375)^0(0.625)^{(1-0)}(0.375)^0(0.625)^{(1-0)}(0.375)^0(0.625)^{(1-0)}(0.375)^0(0.625)^{(1-0)} \\
&= (0.05273) \times (0.09537) \\
&= 0.00503.
\end{aligned}$$

In each of these 3 forms of the likelihood the individual 'fate' has its own probability term (and the likelihood is simply the product of these individual probabilities). Written in this way there is a straightforward and perhaps somewhat obvious way to introduce individual covariates into the likelihood. All we need to do to model the survival probability of the individuals is to express the survival probability of each individual S_i as some function of an individual covariate X_i.

For example, we could use

$$S_i = \frac{e^{\beta_1+\beta_2(X_i)}}{1+e^{(\beta_1+\beta_2(X_i))}} \equiv \frac{1}{1+e^{-(\beta_1+\beta_2(X_i))}},$$

with logit link function

$$\ln\left(\frac{S_i}{1-S_i}\right) = \beta_1 + \beta_2(X_i).$$

Then, we simply substitute this expression for S_i into

$$\mathcal{L}\left(S \mid N, \{a_1, a_2, \ldots, a_8\}\right) \propto \prod_{i=1}^{8} S^{a_i} Q^{(1-a_i)}.$$

Written this way, the MLE's for the β_1 and β_2 (intercept and slope, respectively) become the focus of the estimation.

Pretty slick, eh? Well, it is, with one caveat. The likelihood expression gets 'really ugly' to write down. It becomes a very long, cumbersome expression (which fortunately **MARK** handles for us), and because of the way it is constructed, numerically deriving the estimates takes somewhat longer than it does when the likelihood is not constructed from individuals. Also, there are a couple of things to keep in mind. First, it is important to realize that the survival probabilities are replaced by a logistic submodel of the individual covariate(s). Conceptually, then, every animal i has its own survival probability, and this may be related to the covariate. During the analysis, the covariate of the i^{th} animal must correspond to the survival probability of that animal. **MARK** handles this, and it is this sort of 'book-keeping' that slows down the estimation (relative to analyses that don't include individual covariates).

OK – enough background. Let's look at some examples, and how you handle individual covariates in **MARK**.

11.2. Example 1 – normalizing selection on body weight

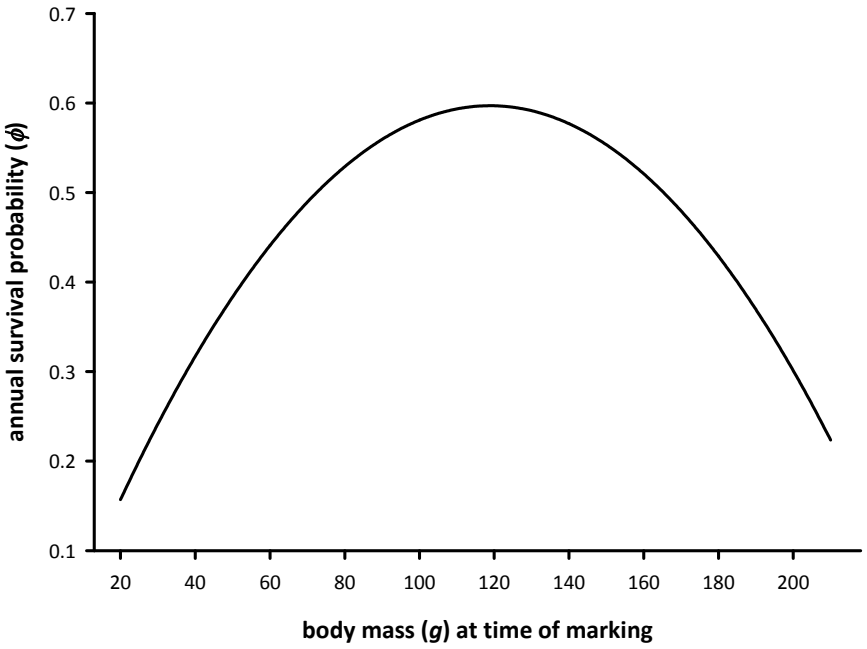
Consider the following example. You believe that the survival probability of some small bird is a function of the mass of the bird at the time it was marked. However, you believe that there might be normalizing selection on body mass, such that there is a penalty for being either 'too light' or 'too heavy', relative to some 'optimal' body mass.

Now, a key assumption – we're going to assume that survival probability for each individual bird is potentially influenced by the mass of the bird at the time it was first marked and released. Now, you might be saying to yourself 'hmmm, but body mass is likely to change from year to year?'. True – and this is an important point to keep in mind – we assume that the individual covariate (in this case, body mass) is 'fixed' over the lifetime of the individual bird. We will consider using 'temporally variable covariates' later on. For now, we will assume that the mass of the bird when it is marked and released is the important factor.

We simulated some capture-recapture data, according to the following function relating survival probability (φ) to body mass (`mass`), according to the following equation:

$$\varphi = -0.039 + 0.0107(\texttt{mass}) - 0.000045(\texttt{mass}^2).$$

To help visualize how survival varies as a function of body mass, based on this equation, consider the following figure:



We see that survival first rises with increasing body mass, then eventually declines – this represents 'normalizing' selection, since survival is 'maximized' for birds that are neither too heavy nor too light (right about now, some of the hard core evolutionary ecologists among you may be rolling your eyes, but it is a reasonable simplification. . .).

We simulated data for 8 occasions, 500 newly marked birds per release cohort (i.e., per year). We also made our life simple (for this example) by assuming that survival probability does not vary as a function of time, only body mass. We set recapture probability to be 0.7 for all birds, whereas survival probability was set as a function of a randomly generated body mass (with mean of 110 mass units). We'll deal with the complications of time-variation in a later example.

Here is a 'piece' of the simulated data set (contained in **`indcov1.inp`**):

```
11111111  1  120.71  14570.24;
11111110  1   86.26   7440.76;
11111110  1  118.23  13978.42;
11111110  1   72.98   5325.47;
11111110  1  101.52  10305.69;
```

Several things to note. First, and perhaps obviously, in order to use individual covariate data, you must include the encounter history for each individual in the data file – you can't summarize your data by calculating the frequency of each encounter history as you may have done earlier (see Chapter 2 for the basic concepts if you're unsure). Each line of the `.INP` file contains an individual encounter history. The encounter history is followed immediately by a single digit '1', to indicate that the frequency of this individual history is 1 (or, that each line of data in the `.INP` file corresponds to 1 individual).

What about the next 2 columns? Consider the following line from the data file:

```
11111111  1  120.71  14570.24;
```

The values 120.71 and 14,570.24 refer to the mass of this individual bird (i.e., `mass` in the equation), and the square of the mass (i.e., `mass`^{2} in the equation = 14,570.24 = 120.71^2). Now, in this example, we've 'hard-coded' the value of the square of body mass right in the `.INP` file. While this may, on occasion, be convenient, we'll see later on that there are situations where you don't want to do this, where it will be preferable to let **MARK** 'handle the calculation of the covariate functions (squaring mass, in this case) for you'.

So, for each bird, we have the encounter history, the number '1' to indicate 1 bird per history, and then one or more columns of 'covariates' – these are the individual values for each bird – in this example, corresponding to mass and the square of the mass, respectively.

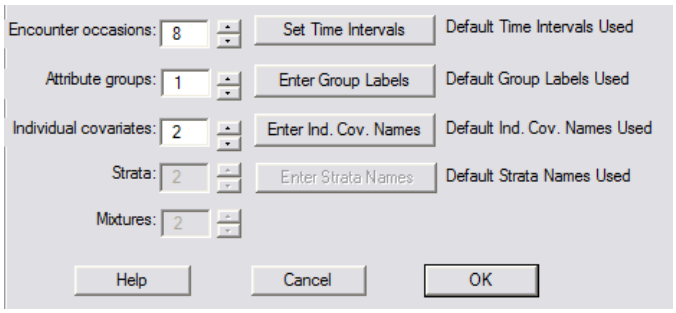
Finally, what about missing values? Suppose you have individual covariate data for some, but not all of the individuals in your data set. Well, unfortunately, there is no simple way to handle missing values. You can either (i) use the mean value of the covariate, calculated among all the other individuals in the data set, in place of the missing value, or (ii) discard the individual from the data set. Or, alternatively, you can discretize the covariates, and use a multi-state approach. The general problem of missing covariates, time-varying covariates and so forth is discussed later in this chapter (section 11.6).

That's about it really, as far as data formatting goes. The next step involves bring these data into **MARK**, and specifying which covariates you want to use in your analyses, and how.

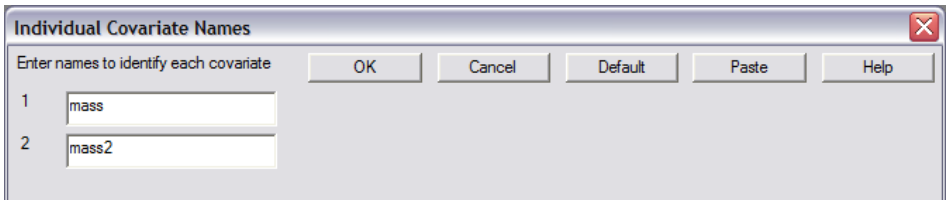
11.2.1. Specifying covariate data in MARK

Start program **MARK**, and begin a new project – '**`recaptures only`**'. We will use the live encounter data contained in **`indcov1.inp`** – 8 occasions, 'standard' mark-recapture 'LLLLL' format. The encounter data for each individual are accompanied by 2 individual covariates for each individual, which we'll call `mass` (for mass) and `mass2` (for mass^2).

At this point, we need to 'tell' **MARK** we have 2 individual covariates (below):



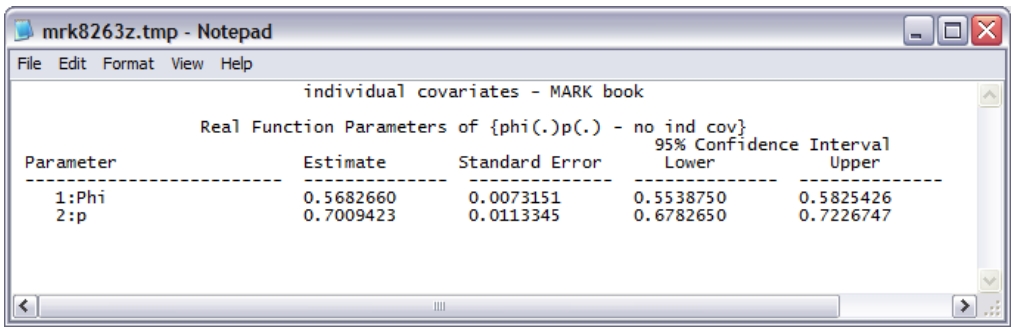
Next, we want to give the covariates some 'meaningful' names, so we click the '**`Enter Ind. Cov. Names`**' button. We'll use `mass` and `mass2` to refer to body mass and body mass-squared, respectively (shown at the top of the next page). That's it! From here on, we refer to the covariates in our analyses by using the assigned labels `mass` and `mass2`.



11.2.2. Executing the analysis

In this example, we simulated data with a constant survival and recapture probability over time. Thus, for our starting model, we will modify the model structure to reflect this – in other words, we'll start by fitting model $\{\varphi_{.}\ p_{.}\}$. Go ahead and set up this model using your preferred method (by either modifying the PIMs directly, or modifying the PIM chart), and run it. When you run **MARK**, you'll notice that it seems to take a bit longer to start the analysis. This is a result of the fact that this is a fairly large simulated data set, and that you are not using summary encounter histories – because we've told **MARK** that the data file contains individual covariates, **MARK** will build the likelihood piece by piece – or, rather, individual by individual. This process takes somewhat longer than building the likelihood from data summarized over individuals.

Add the results to the browser. Let's have a look at the 2 reconstituted parameter estimates:



Start with parameter 2 – the recapture probability. The estimate of 0.7009 is very close to the 'true' value of $p = 0.70$ used in simulating the data (not surprising they should be so close given the size of the data set). What about the first parameter estimate – $\hat{\varphi} = 0.568$? This is the estimate of the apparent survival probability assuming (i) no time variation, and (ii) all individuals are the same. Clearly, it is this second assumption which is most important here, since we know (in this case) that all individuals in this data set are **not** the same – there is heterogeneity among individuals in survival probability, as a function of individual differences in body mass.

Thus, we expect that a model which accounts for this heterogeneity will fit significantly better than a model which ignores it. Where does the value of 0.568 come from? Remember that the actual probability of survival was set in the simulation to be a function of body mass:

$$\varphi = -0.039 + 0.0107(\texttt{mass}) - 0.000045(\texttt{mass}^2).$$

The data were simulated using a normal distribution with mean 110 mass units, and a standard deviation of 25. Thus, the value of 0.568 is the mean survival probability expected given the normal distribution of body mass values, and the function relating survival to body mass. However, if you put

the value of '110' into this equation, you get an estimate of survival of $\hat{\varphi} = 0.594$, which is somewhat different from the reported value of $\hat{\varphi} = 0.568$. Why? Because what **MARK** is reporting is the mean survival of *the data set as a whole*: if you were to take *all* of the mass data in the input file, run each individual value for mass through the preceding equation, and take the mean of all of the generated values of φ, you would get an estimate of $\hat{\varphi} = 0.566$, which is ~identical to the value reported by **MARK**.*

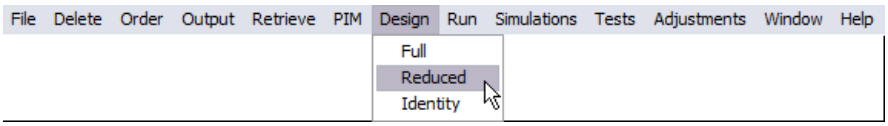
But, back to the question at hand – as suggested, we expect a model which incorporates individual covariates (body mass) to fit better than a model which ignores these differences. How do we go about fitting models with covariates? Simple – we include the individual covariate(s) in the design matrix.

In fact, including individual covariates in the design matrix is often straightforward. For our present example, we're effectively performing a multiple regression analysis. We want to take our starting model $\{\varphi_{.}\ p_{.}\}$ and constrain the estimates of survival to be functions of body mass, and (if we believe that normalizing selection is operating), the square of body mass. These were the 2 covariates contained in the input file (`mass` and `mass2`, respectively).

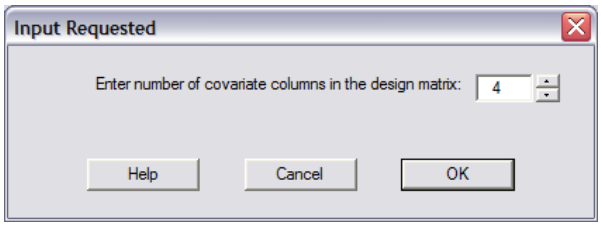
To fit a model with both `mass` and `mass2`, we need to modify the design matrix for our starting model. We can do this in several ways, but as a test of your understanding of the design matrix (discussed at length in Chapter 6), we'll consider it the following way. Our starting model is model $\{\varphi_{.}\ p_{.}\}$. One parameter for survival and recapture probability, respectively. Thus, the starting design matrix will be a (2×2) matrix. We want to modify this starting model to now include terms for `mass` and `mass2`. We want to constrain survival probability to be a function of both of these covariates.

Remembering what you know about linear models and design matrices, you should recall that this means an intercept term, and one term ('slope') for `mass` and `mass2`, respectively. Thus, 3 terms in total, or, more specifically, 3 columns in the design matrix for survival, and 1 column for the recapture probability.

Let's look at how to do this. Select '**`design matrix | reduced`**'.

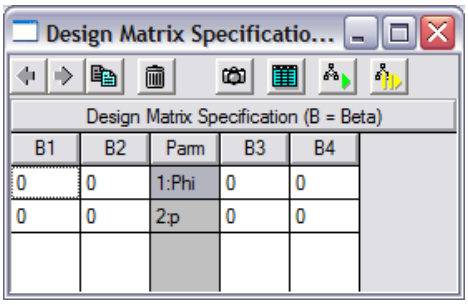


This will spawn a window asking you to specify the number of covariate columns you want. Translation – how many **total** columns do you want in your design matrix. As noted above, we want 4 columns – 3 to specify the survival parameter, and 1 to specify the encounter probability (since this is the parameter structure specified by the PIMs we created when we started). So, enter '4'.

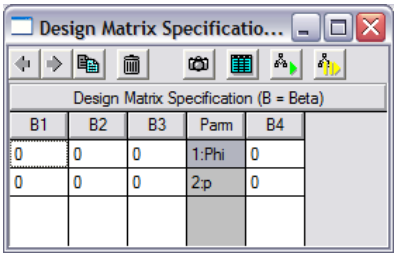


* This issue relates to *Jensen's inequality*, which says that the expected value of the function is not (in general) equal to the function of the expected value, $E[f(x)] \neq f(E[x])$. If you take the average of the data and then apply the function to it, you'll get a different (usually wrong, i.e., not what you meant) answer than if you apply the function to each data value first and then take the average of the values. This is exactly what has happened here – **MARK** reports mean survival, and not survival of the mean.

Once you have entered the number of covariate columns you want in the design matrix, and clicked the '**OK**' button, you'll be presented with an 'empty' (4×2) design matrix.

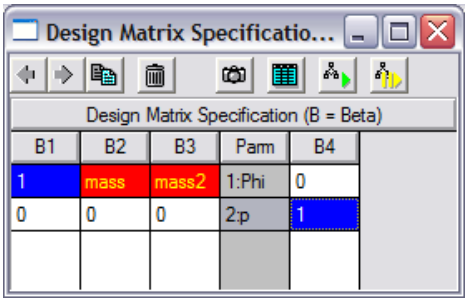


To start with, let's move the grey '**Parm**' column one column to the right, just to make things a bit clearer.



Now, all we need to do is add the appropriate values to the appropriate cells of the design matrix. If you remember any of the details from Chapter 6, you might at this moment be thinking in terms of '0' and '1' dummy variables. Well, you're not far off. We do more or less the same thing here, with one twist – we use the names of the covariates explicitly, rather than dummy variables, for those columns corresponding to the covariates.

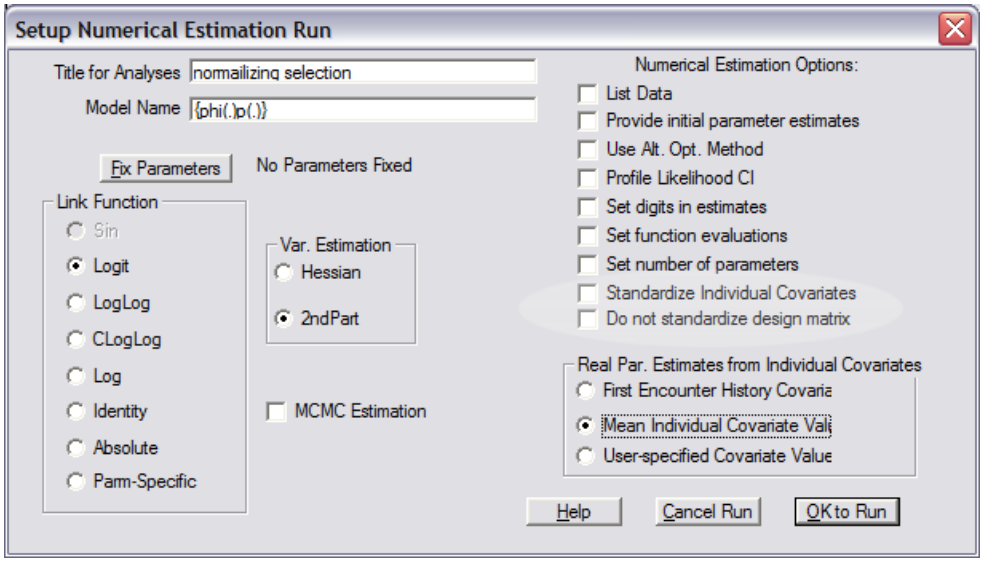
Let's start with the probability of survival. We have 3 columns in the design matrix to specify survival: 1 for the intercept, and 1 each for the covariates `mass` and `mass2`, respectively. For the intercept, we enter a '1' in the first cell of the first column. However, for the 2 covariate columns (columns 2 and 3), we enter the labels we assigned to the covariates, `mass` and `mass2`. For the recapture parameter, we simply enter a '1' in the lower right-hand corner. The completed design matrix for our model is shown below:



That's it! Go ahead and run this model. When you click on the '**Run**' icon, you'll be presented with

the '**Setup Numerical Estimation Run**' window. We need to give our model a title. We'll use '`phi(mass mass2)p(.)`' for the model specified by this design matrix. Again, notice that the sin link is no longer available – recall from Chapter 6 that the sin link is available only when the identity design matrix is used. The new 'default' is the logit link. We'll go ahead and use this particular link function.

Now, before we run the model, the first 'complication' of modeling individual covariates. On the right hand side of the '**Setup Numerical Estimation Run**' window, you'll notice a list of various options. Two of these options refer to 'standardizing' – the first, refers to standardizing the individual covariates. The second, specifies that you do not want to standardize the design matrix. These two '**standardization**' check boxes are followed by a nested list of suboptions (which have to do with how the real parameter estimates from the individual covariates are presented – more on this later).



The first check box (standardize individual covariates) essentially causes **MARK** to '*z-transform*' your individual covariates. In other words, take the value of the covariate for some individual, subtract from it the mean value of that covariate (calculated over all individuals), and divide by the standard deviation of the distribution of that covariate (again, calculated over all individuals). The end result is a distribution for the transformed covariate which has a mean of 0.0, and a standard deviation of 1.0, with individual transformed values ranging from approximately $(-3 \rightarrow +3)$ (depending on the distribution of the individual data). One reason to standardize individual covariates in this way is to make all of your covariates have the same mean and variance, which can be useful for some purposes.

Another reason is as an *ad hoc* method for accommodating any missing values in your data – if you use the z-transform standardization, the mean of the covariates over all individuals is 0, and thus missing data could simply be coded with 0 (which, again, is the mean of the transformed distribution). If you compute the mean of the non-missing values of an individual covariate, and then scale the non-missing values to have a mean of zero, the missing values can be included in the analysis as zero values, and will not affect the slope of the estimated β. However, this 'trick' is not advisable for a covariate with a large percentage of missing values because you will have little to no power. [The issue of 'missing values' is treated more generally in a later section of this chapter.] While these seem fairly reasonable and innocuous reasons to use this standardization option, there are several reasons to be very careful when using this option, as discussed in the following `-sidebar-`. In fact, it is because of some of these complications that the default condition for this standardization option is 'off'.

What about the second option – '**Do not standardize (the) design matrix**'? As noted in the **MARK** help file, it is often helpful to *scale* the values of the covariates to ensure that the numerical optimization algorithm finds the correct parameter estimates. The current version of **MARK** defaults to scaling your covariate data for you automatically (without you even being aware of it). This 'automatic scaling' is done by determining the maximum absolute value of the covariates, and then dividing each covariate by this value. This results in each column scaled to between -1 and 1. This internal scaling is purely for purposes of ensuring the success of the numerical optimization – the parameter values reported by **MARK** (i.e., in the output that you see) are 'back-transformed' to the original scale. There *may* be reasons you don't want **MARK** to perform this 'internal standardization' – if so, you simply check the '**Do not standardize (the) design matrix**' button.

begin sidebar

when to standardize – careful!

While using the z-transform standardization on your individual covariates may appear reasonable, or at the least, innocuous, you do need to think carefully about when, and how, to standardize individual covariates. For example, when you specify a model with a common intercept but 2 or more slopes for the individual covariate, and instruct **MARK** to standardize the individual covariate, you will get a different value of the deviance than from the model run with unstandardized individual covariates.

This behavior is because the centering effect of the standardization method affects the intercept differently depending on the value of the slope parameter. The effect is caused by the nonlinearity of the logit link function. You get the same effect if you standardize variables in a logistic regression, and run them with a common intercept. The result is that the estimates are not scale independent, but depend on how much centering is performed by subtracting the mean value. In other words, situations can arise where the real parameter estimates and the model's AIC_c differ between runs using the standardized covariates and the unstandardized covariates. This situation arises because the z-transformation affects both the slope and intercept of the model. For example, with a logit link function and the covariate x_1,

$$\begin{aligned}\text{logit}(S) &= \beta_1 + \beta_2 \left(x_1 - \bar{x_1}\right) / SD_1 \\ &= \left(\beta_1 - \beta_2 \bar{x_1} / SD_1\right) + (\beta_2 / SD_1) x_1,\end{aligned}$$

where the intercept is the quantity shown in the first set of brackets, and the second bracket is the slope. This result shows the conversion between the β parameter estimates for the standardized covariate and the β parameter estimates for the untransformed covariate, i.e., the intercept for the untransformed analysis would correspond to the quantity in the first set of brackets, and the slope for the untransformed analysis would correspond to the quantity in the second set of brackets. All well and good so far, because the model with a standardized covariate and the model with the unstandardized covariate will result in identical models with identical AIC_c values.

However, now consider the case where we have 2 groups, and want to build a model with different slope parameters for each group's individual covariate values, but a common intercept. In this example, x_1 and x_2 are considered to be the same individual covariate, each standardized to the overall mean and SD, but with values specific to group 1 (x_1) or group 2 (x_2). The *unstandardized* model would look like:

Group 1: $\text{logit}(S_1) = \beta_1 + \beta_2 x_1$

Group 2: $\text{logit}(S_2) = \beta_1 + \beta_3 x_2$

Unfortunately, when the individual covariates *are* standardized, the result is:

Group 1: $\text{logit}(S_1) = (\beta_0 - \beta_1 \bar{x}_1 / SD) + (\beta_1 / SD) x_1$

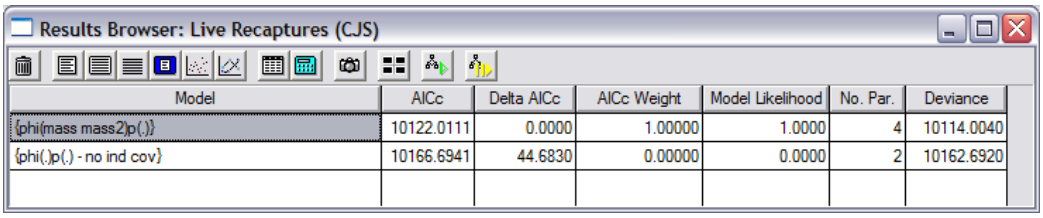
Group 2: $\text{logit}(S_2) = (\beta_0 - \beta_2 \bar{x}_2 / SD) + (\beta_2 / SD) x_2$

In this case, the intercepts for the 2 groups are no longer the same with the standardized covariates, resulting in a different model with a different AIC_c value than for the unstandardized case. This difference causes the AIC_c values for the 2 models to differ because the real parameter estimates differ between the 2 models.

An alternative to this z-transformation is to use the `product` function in the design matrix (c.f. p. 20) to multiply the individual covariate by a *scaling* value. As an example, suppose the individual covariate `Var` ranges from 100 to 900. Using the design matrix function `product(Var,0.001)` in the entries of the design matrix would result in values ranging from 0.1 to 0.9, and would result in 3 more significant digits being reported in the estimates of the β parameter for this individual covariate.

end sidebar

Acknowledging the need for caution discussed in the preceding -sidebar-, for purposes of demonstration, we'll go ahead and run our model, using the z-transformation on the covariate data (by checking the '**`Standardize Individual Covariates`**' checkbox). Add the results to the browser.



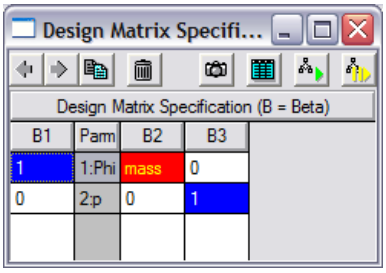
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(mass mass2)p(.)}	10122.0111	0.0000	1.00000	1.0000	4	10114.0040
{phi(.)p(.) - no ind cov}	10166.6941	44.6830	0.00000	0.0000	2	10162.6920

First, we notice right away that the model including the 2 covariates fits **much** better than the model which doesn't include them – so much so that it is clear there is effectively no support for our naïve starting model.

Do we have any evidence to support our hypothesis that there is normalizing selection on body mass? Well, to test this, we might first want to run a model which does not include the `mass2` term. Recall that it was the inclusion of this second order term which allowed for a decrease in survival with mass beyond some threshold value. How do you run the model with `mass`, but not `mass2`?

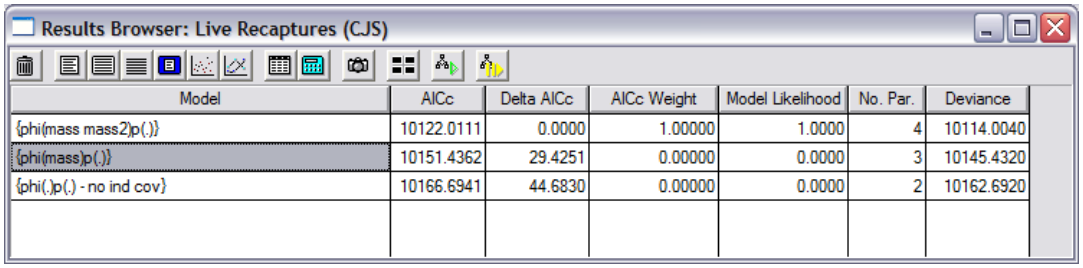
The easiest way to do this is to simply eliminate the column corresponding to `mass2` from the design matrix. So, simply bring the design matrix for the current model up on the screen (by retrieving the current model), and delete the column corresponding to `mass2` (i.e., delete column 3 from the design matrix).

The modified design matrix now looks like:



B1	Parm	B2	B3
1	1:Phi	mass	0
0	2:p	0	1

Go ahead and run this model – again using standardized covariates. Call this model '`phi(mass)p(.)`'. Add the results to the results browser (shown at the top of the next page).


Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(mass mass2)p(.)}	10122.0111	0.0000	1.00000	1.0000	4	10114.0040
{phi(mass)p(.)}	10151.4362	29.4251	0.00000	0.0000	3	10145.4320
{phi(.)p(.) - no ind cov}	10166.6941	44.6830	0.00000	0.0000	2	10162.6920

Note that the model with `mass` only (but not the second order term) fits better than our general starting model, but nowhere near as well as the model including both `mass` and `mass2` – it has essentially no support. In other words, our model with both `mass` and `mass2` is clearly the best model for these data (this is not surprising, since this is the very model we used to simulate the data in the first place!).

So, at this stage, we could say with some assurance that there is fairly strong support for the hypothesis that there is normalizing selection on body mass. However, suppose we want to actually look at the 'shape' of this function. How can we derive the function relating survival to mass, given the results from our **MARK** analysis? In fact, it's fairly easy, *if you remember the details concerning the logit transform, and how we standardized our data*.

To start, let's look at the output from **MARK** for the model including `mass` and `mass2` (shown at the top of the next page). In this case, it's easier to use the '**`full results`**' option (i.e., the option in the browser toolbar which presents all of the details of the numerical estimation). Scroll down until you come to the section shown at the top of the next page. Note that we have 3 sections of the output at this point. In the first section we see the estimated logit function parameters for the model. There are 4 β values, corresponding to the 4 columns of the design matrix (the `intercept`, `mass`, `mass2` and the encounter probability, `p`, respectively). These parameters, in fact, are what we need to specify the function relating survival to body weight.

In fact, if you think about it, only the first 3 of these logit parameters are needed – the last one refers to the encounter probability, which is not a function of body mass. What is our function? Well, it is

$$\text{logit}(\hat{\varphi}) = 0.256733 + 1.1750545(\texttt{mass}_s) - 1.0555046(\texttt{mass}_s^2).$$

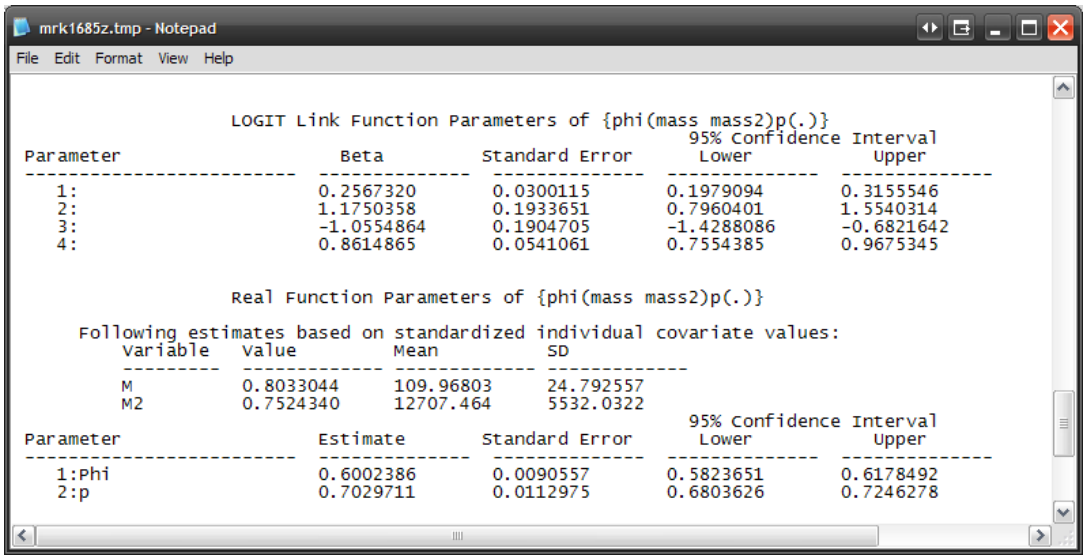
Note that for the two mass terms, we have added a small subscript '*s*' – reflecting the fact that these are 'standardized' masses. Recall that we standardized the covariates by subtracting the mean of the covariate, and dividing by the standard deviation. Thus, for each individual,

$$\text{logit}(\hat{\varphi}) = 0.256733 + 1.17505\left(\frac{\texttt{m} - \bar{\texttt{m}}}{\text{SD}_{\texttt{m}}}\right) - 1.0555\left(\frac{\texttt{m}^2 - \bar{\texttt{m}}^2}{\text{SD}_{\texttt{m}^2}}\right).$$

In this expression, `m` refers to `mass` and $\texttt{m}^2$ refers to `mass2`.

The output from **MARK** (shown at the top of the next page) actually gives you the mean and standard deviations for both covariates. For `mass`, mean = 109.97, and SD = 24.79, while for `mass2`, the mean = 12,707.46, and the SD = 5,532.03. The 'value' column shows the standardized values for `mass` and `mass2` (0.803 and 0.752) for the first individual in the data file. Let's look at an example. Suppose the mass of the bird was 110 units. Thus `mass` = 110, `mass2` = 110^2 = 12,100. Thus,

$$\text{logit}(\hat{\varphi}) = 0.2567 + 1.17505\left(\frac{110 - 109.97}{24.79}\right) - 1.0555\left(\frac{12{,}100 - 12{,}707.46}{5{,}532.03}\right) = 0.374.$$



mrk1685z.tmp - Notepad

File Edit Format View Help

LOGIT Link Function Parameters of {phi(mass mass2)p(.)}

Parameter	Beta	Standard Error	95% Confidence Interval Lower	Upper
1:	0.2567320	0.0300115	0.1979094	0.3155546
2:	1.1750358	0.1933651	0.7960401	1.5540314
3:	-1.0554864	0.1904705	-1.4288086	-0.6821642
4:	0.8614865	0.0541061	0.7554385	0.9675345

Real Function Parameters of {phi(mass mass2)p(.)}

Following estimates based on standardized individual covariate values:

Variable	Value	Mean	SD
M	0.8033044	109.96803	24.792557
M2	0.7524340	12707.464	5532.0322

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.6002386	0.0090557	0.5823651	0.6178492
2:p	0.7029711	0.0112975	0.6803626	0.7246278

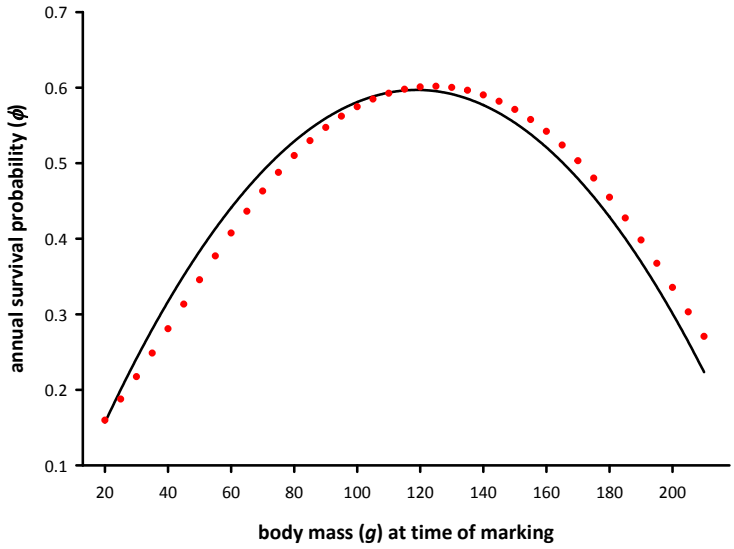
So, if $\text{logit}(\hat{\varphi}) = 0.374$, then how do we get the reconstituted values for survival? Recall that

$$\text{logit}(\theta) = \log\left(\frac{\theta}{1-\theta}\right) = \alpha + \beta x,$$

and

$$\theta = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}.$$

Thus, if $\text{logit}(\hat{\varphi}) = 0.374$, then the reconstituted estimate of φ, transformed back from the logit scale is $e^{0.374}/(1+e^{0.374}) = 0.592$. Thus, for an individual weighing 110 units, expected annual survival probability is 0.592. How well does the estimated function match with the 'true' function used to simulate the data? Let's plot the observed versus expected values:



As you can see from the plot, the fit between the values expected given the 'true' function (solid black line) and those based on the function estimated from **MARK** (red dots) are quite close, as they should be. The slight deviation between the two is simply because the simulated data are simply one realization of the stochastic process governed by the underlying survival and recapture parameters.

Note: in the preceding, we've described the mechanics of reconstituting the parameter estimate – this basically involves back-transforming from the logit scale to the normal $[0, 1]$ probability scale. What about reconstituting the variance, or SE of the estimate, on the normal scale? This is somewhat more complicated. As briefly introduced in Chapter 6, reconstituting the sampling variance on the real probability scale involves use of something known as the '*Delta method*'. The Delta method, and its application to reconstituting estimates of sampling variance is discussed at length in Appendix B.

begin sidebar

AIC, BIC – example of the difference

Back in Chapter 4, we briefly introduced two different information theoretic criteria which can be used to assist in model selection, the AIC (which we've made primary use of), and the BIC. Recall that we briefly discussed the differences between the two – noting that (in broad, simplified terms), the AIC_c has a tendency to pick overly complex models – especially if the 'true' model structure is complex, whereas the BIC has a tendency to pick overly simple models when the reverse is true.

We can demonstrate these differences by contrasting the results of model selection using AIC or BIC for our analysis of the normalizing selection data. To highlight differences between the two, we'll consider the following 4 models: $\{\varphi_{.}\, p_{.}\}$, $\{\varphi_{(mass)}\, p_{.}\}$, $\{\varphi_{(mass,mass^2)}\, p_{.}\}$, and $\varphi_{(mass,mass^2,mass^3)}\, p_{.}\}$. Recall that the *true* model used to generate the simulated data was model $\{\varphi_{(mass,mass^2)}\, p_{.}\}$. So, our candidate model set consists of two models which are simpler than the 'true' model, and one model that is more complex than the 'true' model.

Here are the results from fitting the model set to the data, using AIC_c as the model selection criterion:

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(mass mass2 mass3)p(.)}	10118.4916	0.0000	0.85318	1.0000	5	10108.4810
{phi(mass mass2)p(.)}	10122.0111	3.5195	0.14682	0.1721	4	10114.0040
{phi(m)}	10151.4362	32.9446	0.00000	0.0000	3	10145.4320
{phi(.)p(.) - no covar}	10166.6941	48.2025	0.00000	0.0000	2	10162.6920

Note that although model $\{\varphi_{(mass,mass^2)}\, p_{.}\}$ is the true generating model, it was not the most parsimonious model using AIC_c – in fact, it was 5-6 times less well supported than was a more complex model $\{\varphi_{(mass,mass^2,mass^3)}\, p_{.}\}$.

What happens if we use BIC as our model selection criterion? (Remember this can be accomplished by changing **MARK**'s preferences; '**`File | Preferences`**'). If you look at the results browser at the top of the next page, you'll see that the BIC selected what we know to be the 'true' model $\{\varphi_{(mass,mass^2)}\, p_{.}\}$ – the next best model $\{\varphi_{(mass,mass^2,mass^3)}\, p_{.}\}$ was 5-6 times less well supported than was the most parsimonious model.

Results Browser: Live Recaptures (CJS)

Model	BIC	Delta BIC	BIC Weight	Model Likelihood	No. Par.	Deviance
{phi(mass mass2)p(.)}	10148.5793	0.0000	0.82640	1.0000	4	10114.0040
{phi(mass mass2 mass3)p(.)}	10151.7001	3.1208	0.17359	0.2101	5	10108.4810
{phi(m)}	10171.3635	22.7842	0.00001	0.0000	3	10145.4320
{phi(.)p(.) - no covar}	10179.9797	31.4004	0.00000	0.0000	2	10162.6920

So, is this an example of BIC 'doing better' when the true model is relatively simple? Or is the fact that the BIC picked the right model an artifact of the inclusion of the right model in the candidate model set (a point of some contention in the larger discussion)? Our point here is not to make conclusions one way or the other. Rather, it is merely to demonstrate the fact that different model selection criterion can yield quite different results (conclusions) – so much so (at least on occasion) that it will be worth you spending some time thinking hard about the general question, and reading the pertinent literature. Particularly good starting points are Burnham & Anderson (2004) and Link & Barker (2006).

end sidebar

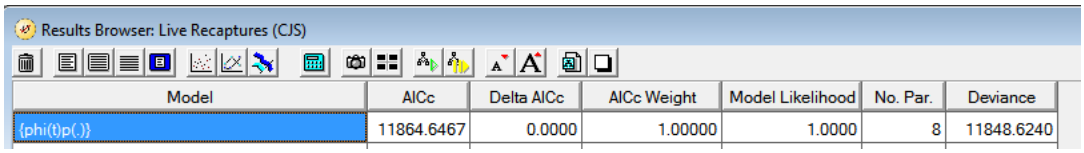
11.3. A more complex example – time variation

In the preceding example, we made life simple by simulating some data where there was no variation in either survival or recapture rates over time. In this example, we'll consider the more complicated problem of handling data where there is potential variation in survival over time.

We'll use the same approach as before, except this time we will simulate some data where survival probability is a complex function of both mass and cohort. In this case, we simulated a data set having normalizing selection in early cohorts, with a progressive shift towards diversifying selection in later cohorts. Arguably, this is a rather 'artificial' example, but it will suffice to demonstrate some of the considerations involved in using **MARK** to handle temporal variation in the relationship between estimates of one or more parameters and one or more individual covariates.

The data for this example are contained in **indcov2.inp**. We simulated 8 occasions, and assumed a constant recapture rate ($p = 0.7$) for all individuals in all years. The data file contains 2 covariates – `mass` and `mass2` (as in the previous example). As with the first example, we start by creating a new project, and importing the **indcov2.inp** data file. Label the two covariates `mass` and `mass2` (respectively).

We will start by fitting model $\{\varphi_t\, p_.\}$, since this is structurally consistent with the data, and will provide a reasonable starting point for comparisons with subsequent models. Go ahead and add the results of this model to the results browser.



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(t)p(.)}	11864.6467	0.0000	1.00000	1.0000	8	11848.6240

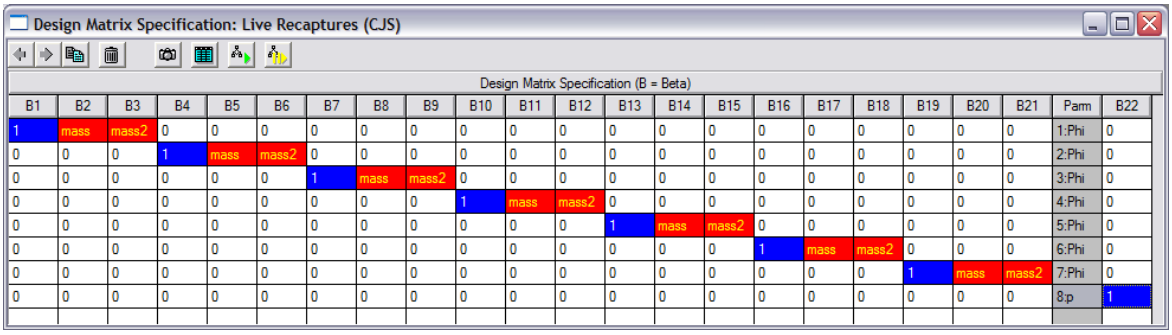
Now, to fit models with both individual covariates, and time variation in the relationship between survival and the covariates, we need to think a bit more carefully than in our first example. If you understood the first example, you might realize that to do this, we need to modify the design matrix. However, how we do this will depend on what hypothesis we want to test. For example, we might believe that the relationship between survival and mass changes with each time interval. Alternatively, we might suppose there is a common intercept, but different slopes for each interval. It is important to consider carefully what hypothesis you want to test before proceeding.

We'll start with the hypothesis that the relationship between survival and mass changes with each time interval. With a bit of thought, you might guess how to construct this design matrix. In the previous example, we used 3 columns to specify this relationship – representing the intercept, `mass` and `mass2`, respectively. However, in the first example, we assumed that this model was constant over all years. So, what do we do if we believe the relationship varies from year to year? Easy, we simply have 3 columns for each interval in the design matrix for survival (with 1 additional column at the end for

the constant recapture probability). So, 7 intervals = 21 columns for survival, plus 1 column for the recapture probability. How many rows? Remembering from Chapter 6, 8 rows total – 7 rows for the 7 survival intervals, and 1 for the constant recapture rate.

So, let's go ahead and construct the design matrix for this model, using the '**`Design | Reduced`**' menu option we discussed previously. We'll start simply, using a DM based on the basic structure of the *identity* matrix – recall that for an identity DM, each row corresponds to a 'time-specific regression model', since each row has its own intercept (see Chapter 6). Or, put another way, each interval 'has its own multiple regression line – separate intercept, separate slope(s) – relating survival to `mass` and `mass2`'.

This matrix (shown below) is sufficiently big such that it's rather difficult to see the entire structure at once.


Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	B21	Parm	B22
1	mass	mass2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1:Phi	0
0	0	0	1	mass	mass2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2:Phi	0
0	0	0	0	0	0	1	mass	mass2	0	0	0	0	0	0	0	0	0	0	0	0	3:Phi	0
0	0	0	0	0	0	0	0	0	1	mass	mass2	0	0	0	0	0	0	0	0	0	4:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	1	mass	mass2	0	0	0	0	0	0	5:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	mass	mass2	0	0	0	6:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	mass	mass2	7:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8:p	1

To help you visualize it, let's look at just a small piece of this design matrix:

B1	B2	B3	B4	B5	B6	B7	B8	B9	
1	mass	mass2	0	0	0	0	0	0	0
0	0	0	1	mass	mass2	0	0	0	0
0	0	0	0	0	0	1	mass	mass2	0

As you can see, for each survival interval, we have 3 columns – 1 intercept, and 1 column each for `mass` and `mass2`, respectively. So, the columns `B1`, `B2` and `B3` correspond to interval 1, `B4`, `B5` and `B6` for interval 2, and so on. You simply do this for each of the 7 survival intervals. The bottom right-hand cell of the matrix (shown on the preceding page) contains a single '`1`' for the constant encounter probability. Call this model '`phi(t * mass mass2)p(.) - separate intcpt`', and run it – remember to standardize the covariates before running the model. Add the results to the browser.

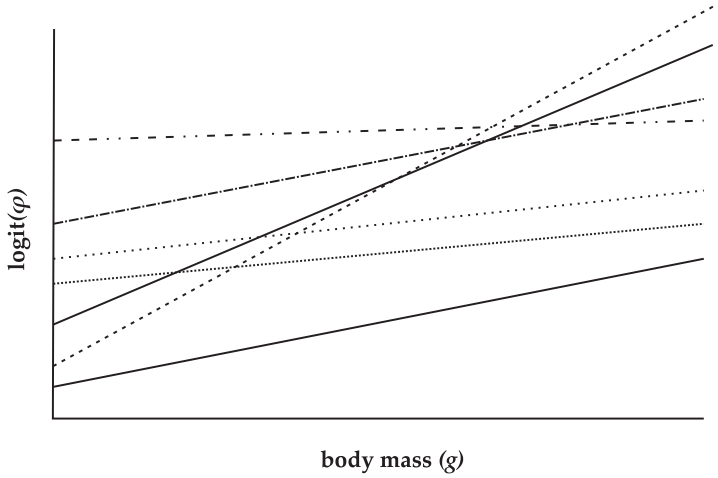
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(t * mass mass2 - separate intcpt)p(.)}	11809.5367	0.0000	1.00000	1.0000	22	11765.3770
{phi(t)p(.)}	11864.6467	55.1100	0.00000	0.0000	8	11848.6240

Again, note that the model constrained to be a function of `mass` and `mass2` fits much better than our naïve starting model. Not surprising, since the data were simulated under the assumption that survival varies as a function of `mass` and `mass2`, and that the function relating survival to both covariates changes over time (i.e., we just fit the true model to the data).

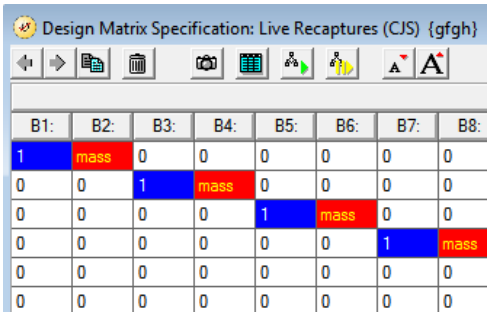
Of course, in practice, we don't know what the true model is, so we fit a set of approximating models. How do we construct those models if the include one or more individual covariates? In the following, we discuss various ways to construct design matrices – in principle, we use the same ideas and mechanics introduced in Chapter 6. However, the design matrices 'look somewhat different' when they include one or more individual covariates.

11.4. The DM & individual covariates – some elaborations

Suppose you want to fit a model with different intercepts and different slopes for each year. In other words, the same model we just built. Start by considering what such a model means. In the following figure, each line represents the relationship (which we assume here is strictly linear) between the parameter, φ, and the individual covariate, `mass`, for each of the 7 years in the study (i.e., separate slope and intercept for each year):



As we've already seen (above), you could accomplish this by adding an 'intercept' and 'slope' parameter(s) to each row for the parameter in question (i.e., using a identity-like structure, have a 'separate regression' for each interval). So, for a simple linear model of survival as a function of `mass`, we could use could use something like the following:



B1:	B2:	B3:	B4:	B5:	B6:	B7:	B8:
1	mass	0	0	0	0	0	0
0	0	1	mass	0	0	0	0
0	0	0	0	1	mass	0	0
0	0	0	0	0	0	1	mass
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

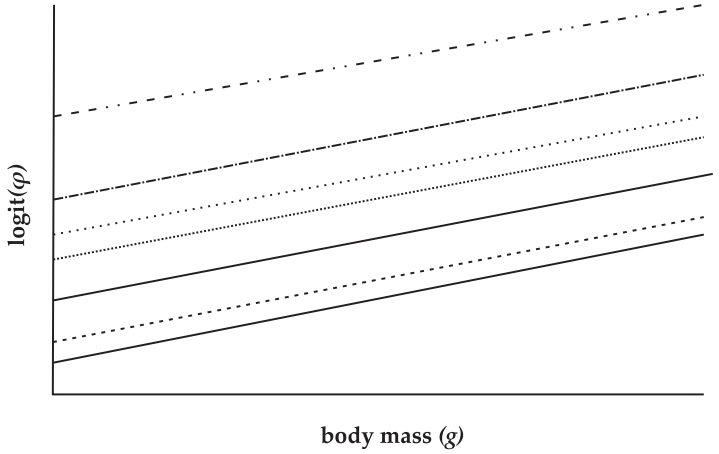
However, a more flexible (and, in some senses, 'familiar') way to model this would have been to use:

B1 intcp	B2 m	B3 t1	B4 t2	B5 t3	B6 t4	B7 t5	B8 t6	B9 m.t1	B10 m.t2	B11 m.t3	B12 m.t4	B13 m.t5	B14 m.t6	Parm	B15 p
1	mass	1	0	0	0	0	0	mass	0	0	0	0	0	1:Phi	0
1	mass	0	1	0	0	0	0	0	mass	0	0	0	0	2:Phi	0
1	mass	0	0	1	0	0	0	0	0	mass	0	0	0	3:Phi	0
1	mass	0	0	0	1	0	0	0	0	0	mass	0	0	4:Phi	0
1	mass	0	0	0	0	1	0	0	0	0	0	mass	0	5:Phi	0
1	mass	0	0	0	0	0	1	0	0	0	0	0	mass	6:Phi	0
1	mass	0	0	0	0	0	0	0	0	0	0	0	0	7:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	8:p	1

In other words, a column of '1's for the intercept, a column for the covariate (`mass`), and then the columns of dummy variables corresponding to each of the time intervals (`t1` → `t6`), and then columns reflecting the interaction of the covariate and time. You might recognize this as the same analysis of covariance (ANCOVA) design you saw back in Chapter 6. If you take this design matrix, and run it, you'll see that you get exactly the same results as you did with the design matrix we used initially – each leads to time-specific estimates of the slope and intercept.

So, if they both yield the 'same results', why even consider this more formal design matrix? As we noted in Chapter 6, the biggest advantage is that using this more complete (formal) design matrix allows you to test some models which aren't possible using the first approach.

For example, consider the additive model – where we have different intercepts, but a common slope among years:



In other words, testing model

$$\varphi = \texttt{time} + \texttt{mass},$$

as opposed to the first model which included the (`time.mass`) interaction (i.e., where the slopes and intercepts vary among years):

$$\varphi = \texttt{time} + \texttt{mass} + \texttt{time.mass}.$$

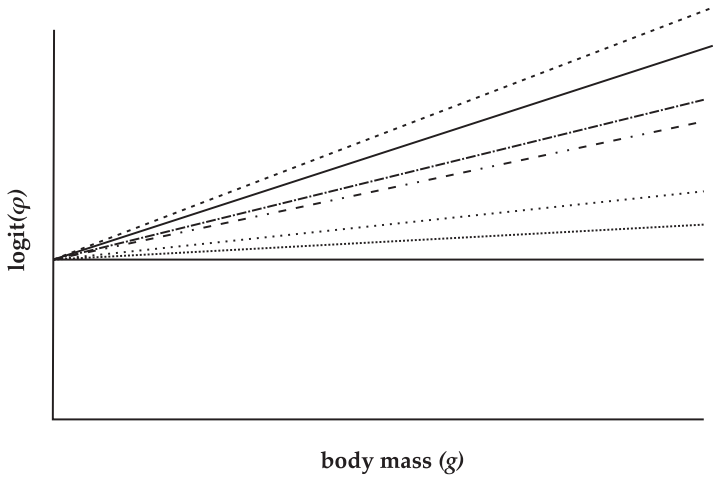
As we discussed in Chapter 6, this sort of additive model can only be fit using this formal design-matrix approach.

So, to fit this model – where we have different intercepts, but a common slope among years – we simply delete the interaction columns. It's that simple!

Here is the reduced design matrix:

B1: intcp	B2: m	B3: t1	B4: t2	B5: t3	B6: t4	B7: t5	B8: t6	Parm	B9: p
1	mass	1	0	0	0	0	0	1:Phi	0
1	mass	0	1	0	0	0	0	2:Phi	0
1	mass	0	0	1	0	0	0	3:Phi	0
1	mass	0	0	0	1	0	0	4:Phi	0
1	mass	0	0	0	0	1	0	5:Phi	0
1	mass	0	0	0	0	0	1	6:Phi	0
1	mass	0	0	0	0	0	0	7:Phi	0
0	0	0	0	0	0	0	0	8:p	1

If instead you wanted a common intercept for all years, but different slopes for `mass` for each year,



then the DM would look like:

B1 intcp	B2 m.t1	B3 m.t2	B4 m.t3	B5 m.t4	B6 m.t5	B7 m.t6	B8 p	Parm
1	mass	0	0	0	0	0	0	1:Phi
1	0	mass	0	0	0	0	0	2:Phi
1	0	0	mass	0	0	0	0	3:Phi
1	0	0	0	mass	0	0	0	4:Phi
1	0	0	0	0	mass	0	0	5:Phi
1	0	0	0	0	0	mass	0	6:Phi
1	0	0	0	0	0	0	0	7:Phi
0	0	0	0	0	0	0	1	8:p

Now that you have the general idea, let's consider constructing a set of models to test various (made-up) hypotheses concerning the encounter data in **`indcov2.inp`**.

We'll suppose that we're interested in fluctuating selection for survival as a function of body mass. Meaning, we suspect that survival varies as a function of body mass (in a potentially non-linear way), and that the pattern of variation varies over time. So, we'll consider a set of models where we fit both first- and second-order polynomial of survival as a function of mass (i.e., survival = f(mass), and survival=f(mass+mass2)), with and without variation in that function over time. We'll start with the most general model – survival as a second-order function of mass, with time variation in the slope and the intercept of that function: $\{\varphi_{time.(m+m^2)}\, p.\}$.

In fact, we built precisely this model in the preceding section, but, using the following design matrix, with a separate intercept for each time interval:

Design Matrix Specification (B = Beta)

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	B21	Parm	B22
1	mass	mass2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1:Phi	0
0	0	0	1	mass	mass2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2:Phi	0
0	0	0	0	0	0	1	mass	mass2	0	0	0	0	0	0	0	0	0	0	0	0	3:Phi	0
0	0	0	0	0	0	0	0	0	1	mass	mass2	0	0	0	0	0	0	0	0	0	4:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	1	mass	mass2	0	0	0	0	0	0	5:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	mass	mass2	0	0	0	6:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	mass	mass2	7:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8:p	1

Here, we'll build the exact same model, but using a common intercept for all time intervals. If you followed what we did earlier in this section, you should have a pretty good guess what it might look like. We know from above that we need 21 columns for survival.

Here is the DM:

B1 int	B2 m	B3 m2	B4 t1	B5 t2	B6 t3	B7 t4	B8 t5	B9 t6	B10 m.t1	B11 m.t2	B12 m.t3	B13 m.t4	B14 m.t5	B15 m.t6	B16 m2.t1	B17 m2.t2	B18 m2.t3	B19 m2.t4	B20 m2.t5	B21 m2.t6	Parm	B22 p
1	mass	mass2	1	0	0	0	0	0	mass	0	0	0	0	0	mass2	0	0	0	0	0	1:Phi	0
1	mass	mass2	0	1	0	0	0	0	0	mass	0	0	0	0	0	mass2	0	0	0	0	2:Phi	0
1	mass	mass2	0	0	1	0	0	0	0	0	mass	0	0	0	0	0	mass2	0	0	0	3:Phi	0
1	mass	mass2	0	0	0	1	0	0	0	0	0	mass	0	0	0	0	0	mass2	0	0	4:Phi	0
1	mass	mass2	0	0	0	0	1	0	0	0	0	0	mass	0	0	0	0	0	mass2	0	5:Phi	0
1	mass	mass2	0	0	0	0	0	1	0	0	0	0	0	mass	0	0	0	0	0	mass2	6:Phi	0
1	mass	mass2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8:p	1

The models are entirely equivalent – in terms of fit, and reconstituted parameter estimates. So is there an advantage of one over the other (i.e., common intercept, versus separate intercepts)? The common intercept approach makes it easier to fit models with specific types of constraint – for example, additive models. On the other hand, interpreting interval-specific intercepts and slopes from the DM built using separate intercepts is somewhat more straightforward than when using a common intercept.

For example, if you look at the parameter (β) estimates from the 'separate intercept' approach, you will see that they correspond to what we expected (given the model under which the data were simulated): in the early cohorts the sign of the slope for mass is positive, and for mass2 is negative – consistent with normalizing selection. In later cohorts, the signs are consistent with increasingly disruptive selection. In contrast, to figure out what is going on when you use a 'common intercept' approach, where each estimated slope is interpreted relative to a reference level (by default, the final time interval), requires more work.

This distinction between the 'separate intercept' approach (which in effect amounts to using an identity matrix), and the 'common intercept' approach (where the slopes reflect variation of levels

of a factor – say, time – relative to a reference level of that factor) were introduced in Chapter 6. We'll consider a more direct way to 'parse out the pattern' – by graphing the relationships directly – later in this chapter.

For the moment, we'll continue building models using the 'common intercept'-based DM as our starting structure. Let's now consider a model that does not have time variation in the relationship between survival and body mass. All we need to do is modify our general DM (with the common intercept for all time intervals), by eliminating the time columns, and the columns showing the interaction of mass with time:

B1	B2	Parm	B3	B4
1	mass	1:Phi	mass2	0
1	mass	2:Phi	mass2	0
1	mass	3:Phi	mass2	0
1	mass	4:Phi	mass2	0
1	mass	5:Phi	mass2	0
1	mass	6:Phi	mass2	0
1	mass	7:Phi	mass2	0
0	0	8:p	0	1

Finally, suppose you want to test the hypothesis that there is a common intercept for each year, but a different slope. How would you modify the design matrix for our general model to reflect this? Well, by now you might have guessed – you simply have 1 column for an intercept for all 7 intervals, and then multiple columns for the `mass` and `mass2` terms for each interval:

B1: int	B2: m.t1	B3: m.t2	B4: m.t3	B5: m.t4	B6: m.t5	B7: m.t6	B8: m2.t1	B9: m2.t2	B10: m2.t3	B11: m2.t4	B12: m2.t5	B13: m2.t6	Parm	B14: p
1	mass	0	0	0	0	0	mass2	0	0	0	0	0	1:Phi	0
1	0	mass	0	0	0	0	0	mass2	0	0	0	0	2:Phi	0
1	0	0	mass	0	0	0	0	0	mass2	0	0	0	3:Phi	0
1	0	0	0	mass	0	0	0	0	0	mass2	0	0	4:Phi	0
1	0	0	0	0	mass	0	0	0	0	0	mass2	0	5:Phi	0
1	0	0	0	0	0	mass	0	0	0	0	0	mass2	6:Phi	0
1	0	0	0	0	0	0	0	0	0	0	0	0	7:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	8:p	1

which you might now realize is entirely equivalent to

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	Parm	B16
1	mass	mass2	0	0	0	0	0	0	0	0	0	0	0	0	1:Phi	0
1	0	0	mass	mass2	0	0	0	0	0	0	0	0	0	0	2:Phi	0
1	0	0	0	0	mass	mass2	0	0	0	0	0	0	0	0	3:Phi	0
1	0	0	0	0	0	0	mass	mass2	0	0	0	0	0	0	4:Phi	0
1	0	0	0	0	0	0	0	0	mass	mass2	0	0	0	0	5:Phi	0
1	0	0	0	0	0	0	0	0	0	0	mass	mass2	0	0	6:Phi	0
1	0	0	0	0	0	0	0	0	0	0	0	0	mass	mass2	7:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8:p	1

It is worth noting that when you specify a model with a common intercept but 2 or more slopes for the individual covariate, and standardize the individual covariate, you will get a different value of

the deviance than from the model run with unstandardized individual covariates. This is because the centering effect of the standardization method affects the intercept differently depending on the value of the slope parameter. The effect is caused by the nonlinearity of the logit link function. You get the same effect if you standardize variables in a logistic regression, and run them with a common intercept. The result is that the estimates are not scale independent, but depend on how much centering is performed by subtracting the mean value.

begin sidebar

Design Matrix Functions

A number of special functions are allowed as entries in the design matrix: `add`, `product`, `power`, `min`, `max`, `log`, `exp`, `eq` (equal to), `gt` (greater than), `ge` (greater than or equal to), `lt` (less than), and `le` (less than or equal to). These names can be either upper- or lower-case. You should not include blanks within these function specifications to allow **MARK** to properly retrieve models with these functions in their design matrix.

As shown below, these functions can be nested to create quite complicated expressions, which may require setting a larger value of the design matrix cell size (something you can specify by changing **MARK**'s preferences – '**`File | Preferences`**').

1. `add` and `product` functions

These two functions require 2 arguments. The `add` function adds the 2 arguments together, whereas the product function multiplies the 2 arguments. The arguments for both functions must be one of the 3 types allowed: numeric constant, an individual covariate, or another function call.

The following design matrix demonstrates the functionality of these 2 functions, where `wt` is an individual covariate.

```
1   1      1       wt   product(1,wt)   product(wt,wt)
1   1      2       wt   product(2,wt)   product(wt,wt)
1   1      3       wt   product(3,wt)   product(wt,wt)
1   0   add(0,1)   wt   product(1,wt)   product(wt,wt)
1   0   add(1,1)   wt   product(2,wt)   product(wt,wt)
1   0   add(1,2)   wt   product(3,wt)   product(wt,wt)
```

The use of the `add` function in column 3 is just to demonstrate examples; it would not be used in a normal application. In each case, a continuous variable is created by adding constant values. The results are the values 1, 2, and 3, in rows 4, 5, and 6, respectively.

Column 5 of the design matrix demonstrates creating an interaction between an individual covariate and another column (the first 3 rows) or a constant and an individual covariate (the last 3 rows). Column 6 of the design matrix demonstrates creating a quadratic effect for an individual covariate. Note that if the 2 arguments were different individual covariates, an interaction effect between 2 individual covariates would be created in column 6.

2. IF functions: `eq` (equal to), `gt` (greater than), `ge` (greater than or equal to), `lt` (less than), `le` (less than or equal to)

These five functions require 2 arguments. The `eq`, `gt`, `ge`, `lt`, and `le` functions will return a zero if the operation is false and a one if the operation is true. For each of these functions, 2 arguments (`x1` and `x2`) are compared based on the function.

For example, `eq(x1,x2)` returns 1 if `x1` equals `x2`, and zero otherwise; `gt(x1,x2)` returns 1 if `x1` is greater than `x2`, zero otherwise; and `le(x1,x2)` returns 1 if `x1` is less than or equal to `x2`, zero otherwise. The arguments for these functions must be one of the 3 types allowed: numeric constant, column variable, or an individual covariate.

The following design matrix demonstrates the functionality of both the add function and the IF function (eq), where age is an individual covariate.

```
1   add(0,age)   eq(0,add(0,age))
1   add(1,age)   eq(0,add(1,age))
1   add(2,age)   eq(0,add(2,age))
1   add(3,age)   eq(0,add(3,age))
1   add(4,age)   eq(0,add(4,age))
1   add(5,age)   eq(0,add(5,age))
```

In this particular example, the individual covariate age corresponds to the number of days before a bird fledges from its nest (fledge day 0) and subsequently enters the study. Suppose an individual fledges from its nest during the fourth survival period. Its encounter history (LDLD format) would consist of '00 00 00 10' and the individual would have −3 as its age covariate because the individual did not fledge from its nest until the fourth survival period. A bird that did not fledge from its nest until survival period 20 would have −19 as its age covariate. Think of the use of negative numbers as an accounting technique to help identify when the individual fledges.

Column 2 of the design matrix demonstrates the use of the add function to create a continuous age covariate for each individual by adding a constant to age. The value returned in the first row of the second column is −3 (0 + (−3) = −3). The value returned in the second row of the second column is −2 (1 + (−3) = −2). The value returned in the fourth row of the second column is zero and corresponds to fledge day 0 (3 + (−3) = 0). The value returned in the fifth row of the second column is one and corresponds to fledge day 1. Thus, column 2 is producing a trend effect of age on survival, with the intercept of the trend model being age zero. A trend model therefore models a constant rate of change with age on the logit scale, so that each increase in age results in a constant change in survival, either positive or negative depending on the sign of β_2.

Now, suppose that survival is thought to be different on the first day that a bird fledges, i.e., the first day that the bird enters the encounter history. To model survival as a function of fledge day 0, use the eq function to create the necessary dummy variable. This is demonstrated in the third column. The eq function returns a value of one only when the statement is true, which only occurs on the first day the bird is fledged. Recall that the value for age of this individual is −3; therefore, the add function column will return a value of −3 (0 + (−3) = −3) in the first row. The eq function in the third column would return a value of zero because age (−3) is not equal to zero. The eq function in the third column, fourth row would return a value of one because age (0) is equal to (0). Note this will only be true for row four for this particular individual; all other rows return a value of zero because they are false. Thus, the eq function will produce a dummy variable allowing for a different survival probability on the first day after fledging from the trend model for age which applies thereafter.

Note that the eq function in this example is using the same results of the add function from the preceding column, and illustrates the nesting of functions.

3. power function

This function requires 2 arguments (x,y). The first argument is raised to the power of the second argument; i.e., the result is x^y. As an example, to create a squared term of the individual covariate length, you would use power(length,2). To create a cubic term, power(length,3). So, in our normalizing selection example (first example of this chapter), we did not need to explicitly include $mass^2$ in the .INP file – we could have used power(mass,2) to accomplish the same thing.

4. min/max functions

The min function returns the minimum of the 2 arguments, whereas the max function returns the maximum of the 2 arguments. These functions allow the creation of thresholds with individual covariates. So, with the individual covariate length, the function min(5,length) would use the value of length when the variable is < 5, but replace length with the value 5 for all lengths > 5. Similarly, max(3,length) would replace all lengths < 3 with the value 3.

5. log, exp functions

These functions are equivalent to the natural logarithm function and the exponential function. Each only requires one argument. So, for the individual covariate `length` = 2, `log(length)` returns 0.693147181, and `exp(length)` returns 7.389056099.

Example

These functions are useful for constructing a design matrix when using the nest survival analysis (Chapter 18). Here, the `add` and `ge` functions are demonstrated. Stage-specific survival (egg or nestling) could be estimated only if nests were aged and frequent nest checks were done to assess stage of failure.

```
1   add(0,age)    GE(add(0,age),15)     product(add(0,age),GE(add(0,age),15))
1   add(1,age)    GE(add(1,age),15)     product(add(1,age),GE(add(1,age),15))
1   add(2,age)    GE(add(2,age),15)     product(add(2,age),GE(add(2,age),15))
1   add(3,age)    GE(add(3,age),15)     product(add(3,age),GE(add(3,age),15))
1   add(4,age)    GE(add(4,age),15)     product(add(4,age),GE(add(4,age),15))
1   add(5,age)    GE(add(5,age),15)     product(add(5,age),GE(add(5,age),15))
1   add(6,age)    GE(add(6,age),15)     product(add(6,age),GE(add(6,age),15))
1   add(7,age)    GE(add(7,age),15)     product(add(7,age),GE(add(7,age),15))
1   add(8,age)    GE(add(8,age),15)     product(add(8,age),GE(add(8,age),15))
1   add(9,age)    GE(add(9,age),15)     product(add(9,age),GE(add(9,age),15))
1   add(10,age)   GE(add(10,age),15)    product(add(10,age),GE(add(10,age),15))
1   add(11,age)   GE(add(11,age),15)    product(add(11,age),GE(add(11,age),15))
1   add(12,age)   GE(add(12,age),15)    product(add(12,age),GE(add(12,age),15))
1   add(13,age)   GE(add(13,age),15)    product(add(13,age),GE(add(13,age),15))
1   add(14,age)   GE(add(14,age),15)    product(add(14,age),GE(add(14,age),15))
1   add(15,age)   GE(add(15,age),15)    product(add(15,age),GE(add(15,age),15))
1   add(16,age)   GE(add(16,age),15)    product(add(16,age),GE(add(16,age),15))
1   add(17,age)   GE(add(17,age),15)    product(add(17,age),GE(add(17,age),15))
1   add(18,age)   GE(add(18,age),15)    product(add(18,age),GE(add(18,age),15))
```

In this particular example, the `age` covariate corresponds to the day that the first egg was laid in a nest (nest day 0). Suppose a nest is initiated during the fourth survival period. Its encounter history (`LDLD` format) would consist of `00 00 00 10` and the nest would have −3 as its age covariate because the first egg was not laid in the nest until the fourth survival period.

Column 2 of the design matrix demonstrates the use of the add function to create a continuous `age` covariate for each nest. The value returned in the first row of the second column is −3. The value returned in the second row of the second column is −2. The value returned in the fourth row of the second column is a zero and corresponds to the initiation of egg laying. The value returned in the fifth row of the second column is one (the nest is one day old).

To model survival as a function of stage, use the `ge` function to quickly create the necessary dummy variable. This is demonstrated in third column. The value of 15 is used in this example because it corresponds to the number of days before a nest will hatch young birds. Day 0 begins with the laying of the first egg, so values of 0 → 14 correspond to the egg stage. Values of 15 → 23 correspond to the nestling stage. The `ge` function will return a value of one (nestling stage) only when the statement is true.

Because the value of `age` for this nest is −3, the add function column returns a value of -3 (since 0 + −3 = −3) for the first row. The `ge` function (third column) returns a value of zero because the statement is false; age (−3) is not greater than or equal to 15. A value of one appears for the first time in row 19; here, the `add` function returns a value of 15 (since 18 + (−3) = 15). The `ge` function returns a value of one because the statement is true; `add(18,age)` results in 15 which is greater than or equal to 15.

The fourth column produces an age slope variable that will be zero until the bird reaches 15 days of age, and then becomes equal to the bird's age. The result is that the age trend model of survival now changes to a different intercept and slope once the bird hatches.

Some useful tricks

An easy way to prepare these complicated sets of functions is to use Excel to prepare the values and then paste them into the design matrix. The following illustrates how to used the concatenate function in Excel to concatenate together a column and a closing ')' to create a complicated column of functions that duplicate the above example.

```
A   B                                C                                D

1   =concatenate("add(age,",A2,")")  =concatenate("GE(",B2,",15)")  =concatenate("product(",B2,",",C2,")")
2   =concatenate("add(age,",A3,")")  =concatenate("GE(",B3,",15)")  =concatenate("product(",B3,",",C3,")")
3   =concatenate("add(age,",A4,")")  =concatenate("GE(",B4,",15)")  =concatenate("product(",B4,",",C4,")")
...
```

Other details

The design matrix values can have up to 60 characters, and unlimited nesting of functions (within the 60 character limit). As an example, the following is a very complicated way of computing a value of 1:

```
log(exp(log(exp(product(max(0,1),min(1,5))))))
```

Before the design matrix is submitted to the numerical optimizer, each entry in the design matrix is checked for a valid function name at the outermost level of nesting, plus that the number of '(' matches the number of ')'.

In previous versions of **MARK**, the design matrix functions were allowed to reference a value in one of the preceding columns. This capability was removed when the ability to nest functions was installed. No flexibility was lost with the removal of the '`Colxx`' capability, and a considerable increase in versatility was obtained with the nested design matrix function calls. As shown in the Excel 'Tricks' example above, the ability to use values from other columns is still available. The '`Colxx`' capability was also a very error prone method in that a column could be inserted ahead of the column being referenced, and the entire model would become nonsense without the user realizing that a mistake had been made. Therefore, the '`Colxx`' capability was removed.

end sidebar

11.5. Plotting + individual covariates

In the first example presented in this chapter, we considered the relationship between survival and individual body mass, under the hypothesis that there was strong 'normalizing selection' on mass – i.e., that the relationship between survival and mass was quadratic. We found that a quadratic model

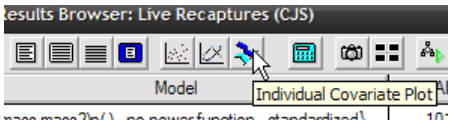
$$\text{logit}(\hat{\varphi}) = 0.256733 + 1.1750545(\texttt{mass}_s) - 1.0555046(\texttt{mass}_s^2),$$

had good support in the data. We discussed briefly the mechanics of reconstituting the estimates of survival on the normal probability scale – the complication is that you need to generate a reconstituted value for each plausible value of the covariate(s) in the model. In fact, this is not particularly challenging for simple models such as this. Because the linear model consists of a covariate (`mass`) plus a function of the covariate ($\texttt{mass}^2$), it is relatively trivial to code this into a spreadsheet and generate a basic plot of predicted survival values over a range of values for `mass`. In fact, this is effectively what was done to generate the plot of predicted versus observed values we saw earlier (example on p. 14).

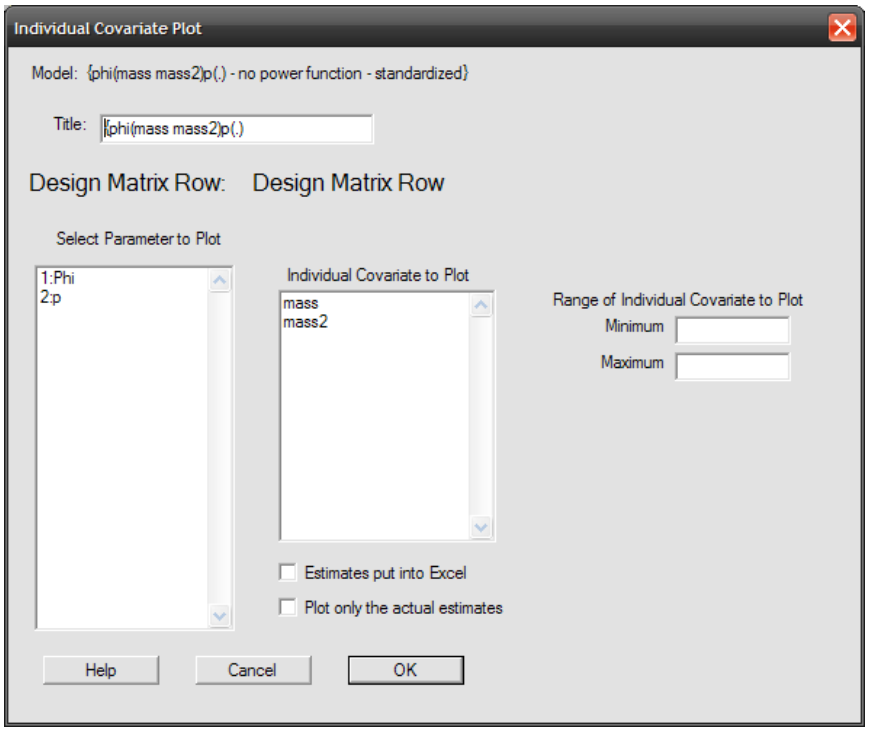
But, there are no confidence bounds on the predicted value function. The calculation of 95% CI for this function requires use of the *Delta method* – although not overly difficult to apply (the Delta method is discussed at length in Appendix B), it can be cumbersome and time consuming to program.

Fortunately, **MARK** has a plotting tool that make it convenient to generate a plot of predicted values from models with individual covariates, which includes the estimated 95% CI. **MARK** also makes it possible to output the data (including the data corresponding to the 95% CI) to a spreadsheet.

Let's demonstrate this for the analysis we previously completed on the normalizing selection data in **`indcov1.inp`**. Open up the `.DBF` file corresponding to those results, and retrieve the most parsimonious model from the model set we fit to those data $\{\varphi_{mass+mass^2}\ p_.\}$. Then, click on the '**`Individual Covariate Plot`**' icon in the main **MARK** toolbar:



This will bring up a new window which will allow you to specify key attributes of the plot:

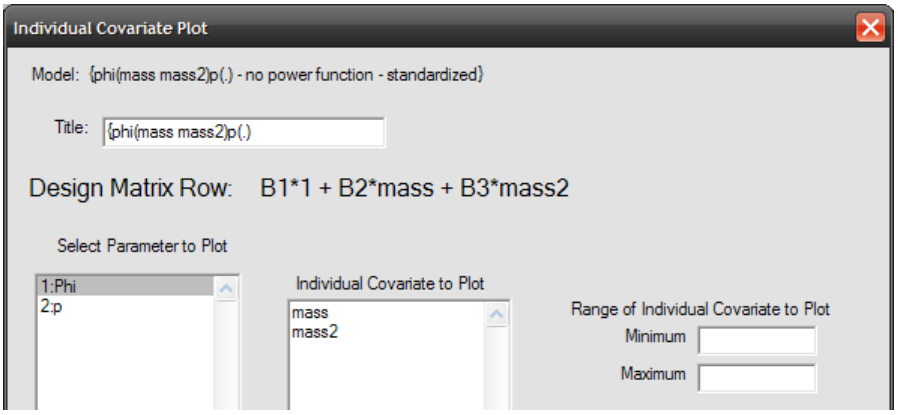


Notice that the title of the currently active model is already inserted in the title box. Next, are two boxes where you specify (i) which parameter you want to plot, and (ii) which individual covariate you want to plot. In our model, there are 2 different individual covariates – `mass` and `mass2`.

So, first question – which one to plot? If you look back at the figure at the bottom of p. 13, you'll see that we're interest in plotting 'survival' versus 'mass'. So, if our goal is to essentially replicate these plots, with the addition of 95% CI, using this individual covariate plot tool in **MARK**, it would seem to make sense that we should specify `mass` as the covariate we want to plot.

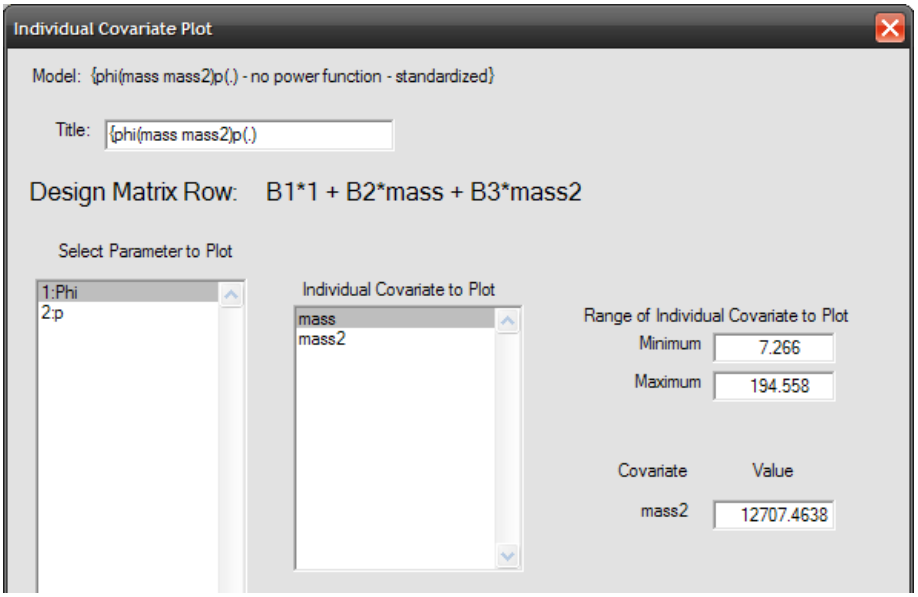
Finally, two boxes which allow us to specify the numerical range of the individual covariate to plot. Also notice the check box you can check if you want to output the various estimates that go into the plot output to a spreadsheet.

OK – seems easy enough. Let's start by clicking on the survival parameter '**Phi**'.



As soon as we do so, the window 'updates', and now presents you with the '**Design Matrix Row**'. For this example, the DM has only 2 rows, so what is presented is in fact the linear model itself.

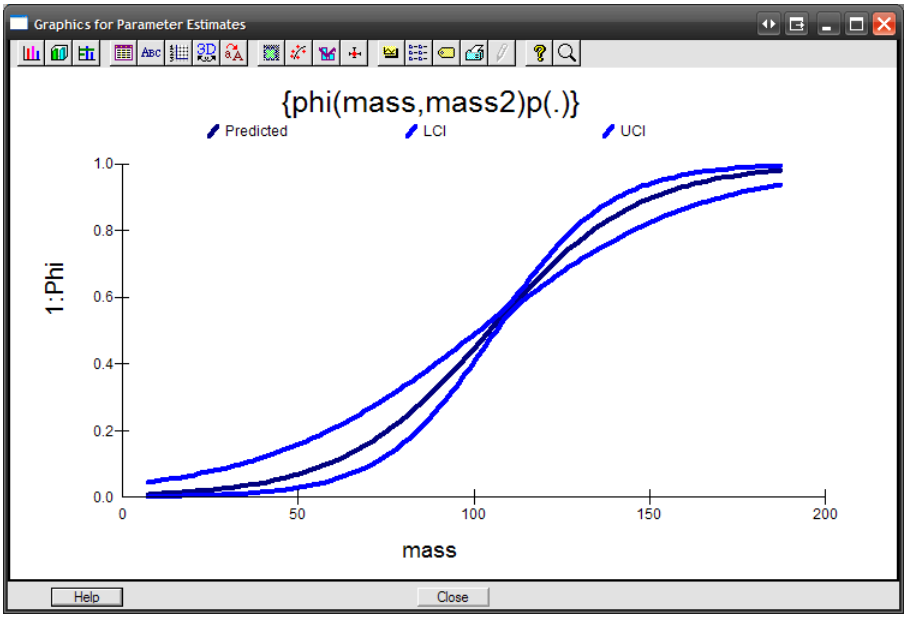
Next, we click on 'mass' to specify that as the individual covariate we want to plot. The window immediately updates – and spawns a new box in the process.



As you can see, the range of covariate values has been updated showing the maximum and minimum values that are actually in the .INP file. You can change these manually as you see fit (usual caveats about extrapolating a plot outside the range of the data apply).

Now, what about the new box – showing mass2 set to 12,707.4638? First, you might recognize the number 12,707.4638 as the square of the mean mass of all individuals in the sample. But, why is a box for mass2 there in the first place? It's there because the linear model that **MARK** is going to plot has 2 covariates – mass and mass2.

OK – so what does **MARK** actually plot? Well, if you click the '**OK**' button, **MARK** responds with



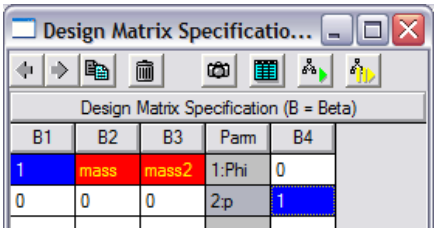
which doesn't look remotely like the quadratic curve we were expecting. What is actually being plotted? Well, if you think about it for a moment, it should be clear that **MARK** is plotting the functional relationship between survival and `mass`, holding the value of `mass2` constant at the mean value! Different values of `mass2` would yield different plots.

So, **MARK** isn't doing anything wrong – it's simply plotting what you told it to plot. **MARK** generates a 2-D plot between some parameter and one covariate. If there are other covariates in the model, then it needs to know what to do with them. Clearly, if there were only 2 covariates in the model, you could construct a 3-D plot (the two covariates on the x- and y-axes, and the parameter on the z-axis), but what if you had > 2 covariates? If would be difficult to program **MARK** to accommodate all permutations in the plot specification window, so it defaults to 2-D plots, meaning (i) you plot a parameter against only one covariate, and (ii) you need to tell **MARK** what to do with the other covariates.

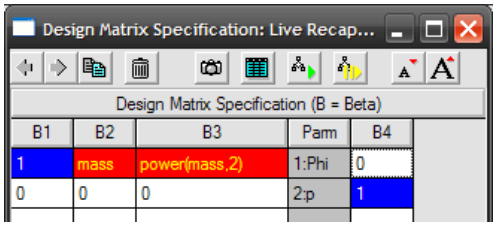
So, how do you tell **MARK** to plot survival versus `mass` and `mass2` together, as a single 2-D plot? The key is in specifying the relationship between `mass` and `mass2` explicitly – in effect, telling **MARK** that `mass2` is in fact just (`mass` × `mass`). **MARK** doesn't 'know' that the second covariate (`mass2`) is a simple function of the first (mass). **MARK** doesn't know this because you haven't told **MARK** that this is the case. In your DM, you simply entered `mass` and `mass2` as label names for the covariates, which were in fact 'hard-coded' in the `.INP` file. You (the user) know what they represent, but all **MARK** sees are two different covariates with two different labels.

So, if you can't pass this information to **MARK** in the plot specification window, where can you do so? *Hint*: what was the subject of the last `-sidebar-` presented several pages back? Looking back, you'll see that we introduced a series of 'design matrix functions', which included `power` and `product`. In our current analysis, we coded for `mass` and `mass2` explicitly in the DM by entering the labels corresponding to the `mass` and `mass2` covariates, which were hard-coded into the `.INP` file. As such, we know what the covariates represent, but **MARK** doesn't – it only knows the label names.

But, what if instead of

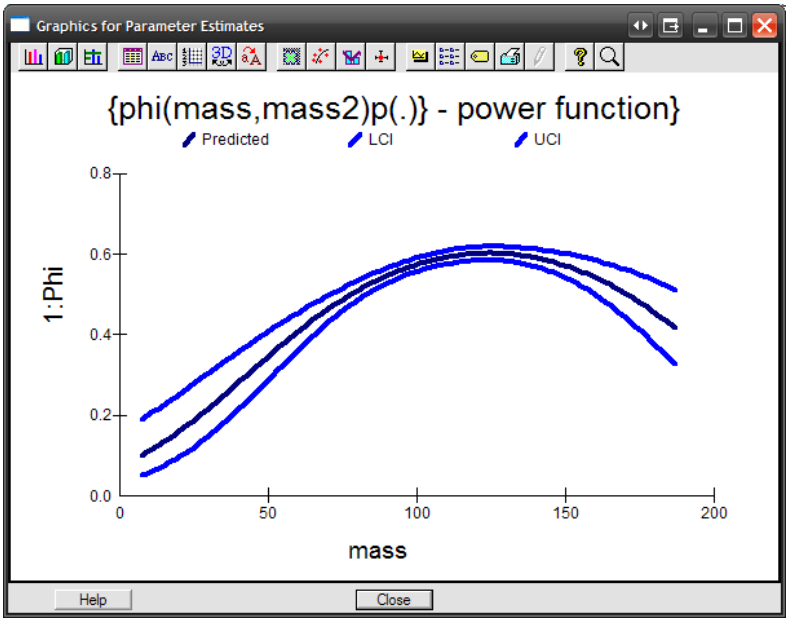


we used



Look closely at this second DM – notice that we've used the `power` function. Recall that the `power` function has two arguments – the first argument (`mass`, in this example) is raised to the power of the second argument (2, in this case). Now, we have explicitly coded (i.e., told **MARK**) that the second covariate is a power function of the first covariate. And because **MARK** now 'knows this', it 'knows what to plot, and how'.

Run this model, and add the results to the browser. As expected, the results are identical to what we saw when we ran this model using the hard-coded `mass2` in the `.INP` file. But, more importantly, when we plot this model, we get exactly what we were looking for:



Note that there are two other options in the '**Individual Covariate Plot**' specification window: you can (i) output the estimates into Excel, or (ii) plot only the actual estimates (meaning, plot only the reconstituted estimates for the parameter for the actual covariates in the input file – the estimates are presented without their estimated SE).

Beyond the mechanics of plotting individual covariate functions, which is clearly part of the intent of this section, this example also demonstrates one of the 'hidden' advantages of using the DM functions to handle coding any functional relationships you might have among your covariates. Not only does this save you from having to do those calculations by hand while you construct the `.INP` file, they also provide a convenient mechanism to make those functional relationships 'known' to **MARK**.

Plotting model averaged models with covariates is possible in **MARK** (see section 11.8), and using **RMark** (see Appendix C – discussion of the `covariate.predictions` function).

begin sidebar

plotting 'environmental' covariates as 'individual' covariates

In Chapter 6, section 6.8.2, we considered the plotting of the functional relationship between some parameter of interest and a particular 'environmental' covariate. One of the things noted in Chapter 6 was the lack of a direct option in **MARK** to plot this functional relationship.

But, we can, in fact, generate exactly the plot we're looking for, within **MARK**, by using a 'trick' that involves individual covariates. The 'trick' is to get **MARK** to treat environmental covariates as individual covariates, and then use the individual covariate plotting capabilities in **MARK** that we introduced in the preceding section.

The basic idea is actually quite simple – if you remember the difference between an 'environmental' and 'individual' covariate. They key is the idea that an 'environmental covariate' is a covariate that applies to all individuals. So, how do we use individual covariates to model/plot environmental covariates? Easy – you simply add the value of the environmental covariate to each individual in the `.INP` file, as if it were an individual covariate.

We'll demonstrate this using the dipper data (what else?). Assume that we believe that annual apparent survival, φ is a function of some measure of rainfall. The dipper data consists of live capture data over 7 occasions (6 intervals).

Here are the 'rain data' we'll use in our model.

```
Interval  1    2    3    4    5    6
rain      1   10    8   15    3    6
```

For this demonstration, we'll use the full dipper data (**ed.inp**) – 7 occasions, 2 attribute groups (males and females). The first step involves entering the environmental covariate data into the `.INP` file, such that each value of the environmental covariate (rain) will be a time-specific individual covariate, with the values of those covariates repeated for all of the individuals in the data set.

The easiest way to explain is this be demonstration. First, here are the top few lines of the full dipper encounter history file (which consists of 294 individuals). There are 2 frequency columns after the encounter history – the first column corresponds to males, while the second corresponds to females.

The first few lines of the `.INP` file happen to be for male individuals.

```
1111110 1 0;
1111000 1 0;
1100000 1 0;
```

Now, all we need to do is enter the environmental covariates as a set of time-specific individual covariates.

Here is what the modified `.INP` file will look like (**`ed_mod.inp`**) – again, we'll only show the first few lines of the file:

```
1111110 1 0  1  10  8  15   3   6;
1111000 1 0  1  10  8  15   3   6;
1100000 1 0  1  10  8  15   3   6;
```

OK, now that we have this modified `.INP` file, start a new project in **MARK** start a new project – 7 occasions, 2 attribute groups (males and females), and 6 individual covariates, which we'll refer to as {`r1`,`r2`,...,`r6`}, corresponding to time interval 1, time interval 2, and so on.

We'll start by fitting model $\{\varphi_t\ p_.\}$ – in other words, no sex differences in φ, but φ allowed to vary over time, t. Encounter probability, p, is constant over time, with no sex differences.

To make our lives simpler, we'll build the underlying parameter structure for our starting model using the following PIM chart (we'll assume that by now you know how to do this). Then, we'll build the DM corresponding to this PIM structure – again, this should all be familiar territory:

B1 phi intc	B2 t1	B3 t2	B4 t3	B5 t4	B6 t5	Parm	B7 p
1	1	0	0	0	0	1:Phi	0
1	0	1	0	0	0	2:Phi	0
1	0	0	1	0	0	3:Phi	0
1	0	0	0	1	0	4:Phi	0
1	0	0	0	0	1	5:Phi	0
1	0	0	0	0	0	6:Phi	0
0	0	0	0	0	0	7:p	1

Go ahead and run this model, and add the results to the browser.

Next, we want to modify the DM to constraint φ to be a linear function of `rain`. Recall from Chapter 6 that all we need to is (i) eliminate the time columns from the DM, and (ii) insert a column containing the values for the environmental covariate, `rain`. The modified DM is shown at the top of the next page.

B1 phi intc	B2 rain	Parm	B3 p
1	1	1:Phi	0
1	10	2:Phi	0
1	8	3:Phi	0
1	15	4:Phi	0
1	3	5:Phi	0
1	6	6:Phi	0
0	0	7:p	1

Go ahead and run the model, and add the results to the browser:

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(rain)p(.)}	672.7933	0.0000	0.64619	1.0000	3	666.7364
{phi(t)p(.)} - DM}	673.9980	1.2047	0.35381	0.5475	7	659.7301

If we look at the β estimates, we see that the linear model for apparent survival is

$$\text{logit}(\varphi) = 0.3027129 + (-0.0076410)(\texttt{rain}).$$

So, as rain increases, apparent survival decreases, since the estimate for the coefficient for the '`rain`' covariate is negative.

But, now, we'd like to plot this relationship, using **MARK**. To do this, we're first going to duplicate model $\{\varphi_{rain}\ p_{.}\}$, but this time using our individual covariates corresponding to the environmental covariates – recall that we named them {`r1`, `r2`,..., `r6`} when we set up the specifications for the analysis.

How do we modify the DM to use these individual covariates? Easy – simply remember that each covariate is *time-specific*. In other words, `r1` corresponds to interval 1, `r2` corresponds to interval 2, and so on. Keeping this in mind, then here is what our modified DM will look like:

B1: phi int	B2: t1	Parm	B3: p
1	r1	1:Phi	0
1	r2	2:Phi	0
1	r3	3:Phi	0
1	r4	4:Phi	0
1	r5	5:Phi	0
1	r6	6:Phi	0
0	0	7:p	1

Go ahead and run this model – let's name it '`phi(ind rain cov)p(.)`'. Let's have a look at the browser:

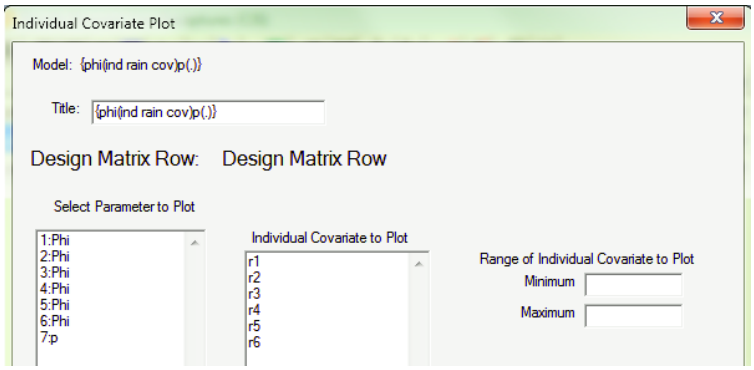
Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(rain)p(.)}	672.7933	0.0000	0.39254	1.0000	3	666.7364
{phi(ind rain cov)p(.)}	672.7933	0.0000	0.39254	1.0000	3	666.7364
{phi(t)p(.)}	673.9980	1.2047	0.21492	0.5475	7	659.7301

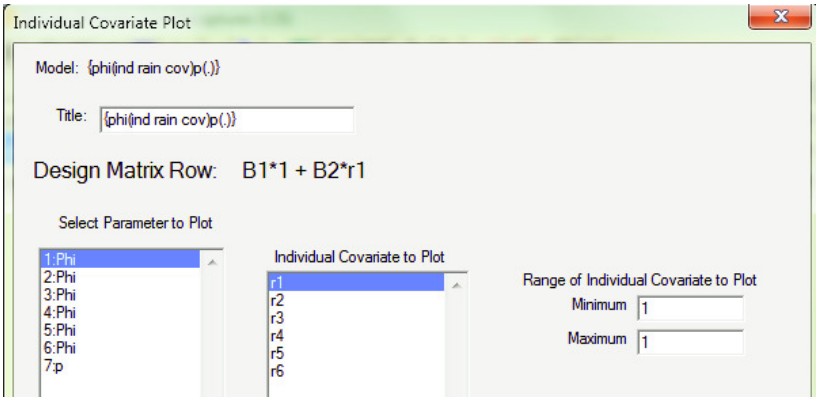
We see that the model deviance for model '`phi(rain)p(.)`' – built using the environmental covariates 'the usual way', and the deviance of model '`phi(ind rain cov)p(.)`', are identical. If you compare reconstituted parameter estimates between the two models, they're also the same.

Simply put, the 2 models are equivalent, in all but one important way. Because model '`phi(ind rain cov)p(.)`' was built using individual covariates, we can use the individual covariate plotting capabilities in **MARK** to plot the functional relationship – and the uncertainty in that relationship – between the parameter (in this case, φ), and the covariate (`rain`).

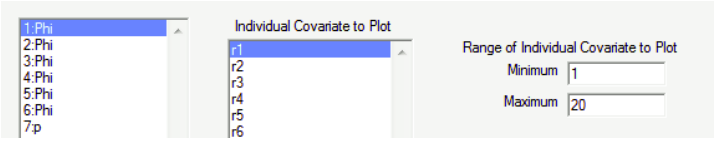
To generate the plot, simply click the '**`Individual covariate`**' plot icon in the toolbar, which will bring up the following window:



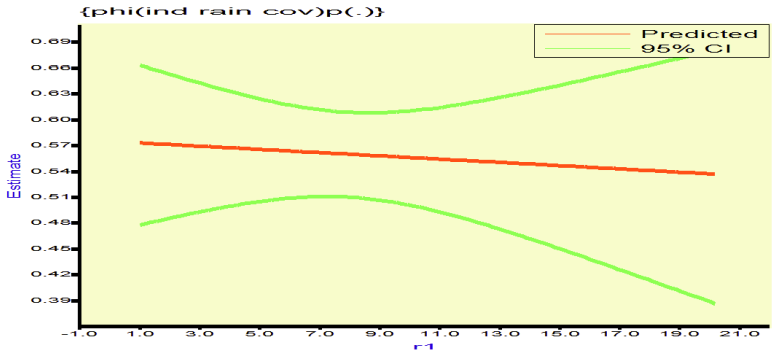
All you need to do is pick any one of the 6 parameters to plot (`1:Phi`, `2:Phi`,...), and (this is important) the correct (matching) individual covariate. For example, parameter '`1:Phi`' corresponds to the first interval, which corresponds to time-specific individual covariate '`r1`'. Parameter '`2:Phi`' corresponds to covariate '`r2`', and so on. It doesn't matter which parameter you pick, but it does matter that you pick the appropriate covariate it matches to. For present purposes, we'll select '`1:Phi`' and '`r1`'.



Now, notice that on the far right-hand side, the range for '`r1`' is shown as 1 for the minimum, and 1 for the maximum. That is because '`r1`' corresponds to the rain covariate for the first interval, which was 1. Needless to say, if we don't adjust the range, the plot won't be particularly interesting. Let's change the range to 1 for the minimum, and 20 for the maximum:



All that remains is to generate the plot. Click the '**OK**' button and we get exactly the plot we're after – the basic function, and the uncertainty represented by the 95% CI:



But, the Dipper data set contains encounter data for both males, and females. Why does the plot contain only a single line? In other words, how do you generate a separate plot for both males and females? In fact, this is covered in section 11.8.2.

end sidebar

11.6. Missing covariates, time-varying covariates, and other complications...

Several strategies for handling missing individual covariates are available. Probably the best option is to code missing individual covariate values with the mean of the variable for the population measured. Replacing the missing value with the average means that the mean of the observed values will not change, although the variance will be slightly smaller because all missing values will be exactly equal to the mean and hence not variable.

The easiest way to accomplish this in **MARK** is to use the 'standardize covariates' option – if you compute the mean of the non-missing values of an individual covariate, and then scale the non-missing values to have a mean of zero, the missing values can be included in the analysis as zero values, and will not affect the value of the estimated β term. (*note*: we don't advise this trick for a covariate with a large percentage of missing values because you have no power, but this approach does work for a 'small' number of missing values).

If you have lots of missing values, another option is to code the animals into 2 groups, where all the missing values are in one group. Then, you can use both groups to estimate a common parameter, and only apply the individual covariate to one group. This approach can be tricky, so think through what you are doing before you try it.

What about covariates that vary through time? In all our examples so far, we've made the assumption that the covariate is a constant 'fixed' value over the sample period. But, clearly, this will often (perhaps generally) not be the case. For example, consider body mass. For perhaps most species, body mass typically changes dynamically over time, and if we believe that body mass influences survival or some other parameter, then we might want to constrain our estimates to be functions of a dynamically changing covariate, rather than a static one (typically measured at the time the individual was initially captured and marked). You can handle time-varying covariates in one of a couple of ways.

First, you **can** include time-varying individual covariates in **MARK** files, but you must have a value for every animal on every occasion, even if the animal is not captured. Typically, you can impute these values if they are missing (not observed), but be sure to recognize what this imputation might do to your estimates. As demonstrated in the preceding `-sidebar-`, you implement time-varying individual covariates just like any other individual covariate, except that you have to have a different name for each covariate corresponding to each time period. '

For example, suppose you have a known fate model (which we'll cover in chapter 17) with 5 occasions, and you have estimated the parasite load for each animal at the beginning of each of the 5 occasions. The 5 values for each animal are contained in the variables `v1`, `v2`, `v3`, `v4`, and `v5`.

A design matrix that would estimate the effect of the parasite load assuming that the effect is constant across time would be:

```
1 v1
1 v2
1 v3
1 v4
1 v5
```

The second β estimate is the slope parameter associated with the time-varying individual covariates. Note that you do not want to standardize these individual covariates, because standardizing them will cause them to no longer relate to one another on the same scale (making a common slope parameter nonsensical). Each would have a different scale after standardizing. If you need to standardize the covariates, you must do so before the values are included in a **MARK** encounter histories input file, and

you must use a common mean and standard deviation across the entire set of variables and observations.

The following design matrix would build a model where you assume the effect of parasite load is different for each interval, but with the same survival probability for animals with no parasites (i.e., the same intercept).

```
1 v1  0  0  0  0
1  0 v2  0  0  0
1  0  0 v3  0  0
1  0  0  0 v4  0
1  0  0  0  0 v5
```

The following model would allow different survival probabilities for each interval (i.e., time-specific survival), but assumes the same impact of parasites on survival on the logit scale (assuming that a logit link function is used). In other words, same slope, different intercept for each interval:

```
1 1 0 0 0 v1
1 0 1 0 0 v2
1 0 0 1 0 v3
1 0 0 0 1 v4
1 0 0 0 0 v5
```

Finally, a DM like the one shown below would allow a completely different survival probability and parasite effect for each occasion:

```
1  1  0  0  0 v1  0  0  0  0
1  0  1  0  0  0 v2  0  0  0
1  0  0  1  0  0  0 v3  0  0
1  0  0  0  1  0  0  0 v4  0
1  0  0  0  0  0  0  0  0 v5
```

which is equivalent to specifying a separate function for each interval – this is perhaps illustrated in a 'more obvious' fashion in the following DM, which is equivalent to the one above (although interpretation of the β terms is clearly different).

```
1 v1  0  0  0  0  0  0  0  0
0  0  1 v2  0  0  0  0  0  0
0  0  0  0  1 v3  0  0  0  0
0  0  0  0  0  0  1 v4  0  0
0  0  0  0  0  0  0  0  1 v5
```

Alternatively, you can 'discretize' the covariate, and use a multi-state model (chapter 10) to model transitions as a function of the covariate 'class' the individual is in. For example, suppose you believe that survival from time (i) to ($i + 1$) is strongly influenced by the size of the organism at time (i). Now, size is clearly a continuously distributed trait. But, perhaps you might reasonably classify each marked individual as either 'large', 'average', or 'small' size. Then, each individual at each occasion is classified into one of these 3 different size classes, and you use a multi-state approach to estimate the probability of surviving as a function of being in a particular size class. If the covariate is not measured (typically, if the individual is not captured), then the missing value is accounted for explicitly by including the encounter probability p in the model. Moreover, you would also be able to look at the relationship between survival as a function of size, and the probability of moving among size classes.

Sounds reasonable, but you need to consider a couple of things. First, in applying this approach, you are discretizing a continuous distribution, and how many discrete categories you use, and how you decide to partition them (e.g., what criterion you use to define a 'large' versus 'small' individual), may strongly influence the results you get. However, when there are a large number of missing covariate values, or if discretizing seems 'reasonable', this is a robust and easily implemented approach. Second, you might need to be a bit 'clever' in setting up your design matrix to account for trends (relationships) among states, as we'll see in the following worked example.

Finally, and perhaps more elegantly, Simon Bonner, building on a solution first proposed by Catchpole *et al*. (2008), has described the use of a 'trinomial' mark-live encounter-dead recovery to handle missing covariates, without requiring the arbitrary discretization of the covariates required by the multi-state approach as just described (Bonner 2013). The 'trinomial approach' is based on re-structuring the likelihood for each individual, by modeling only the events that follow the release of each marked individual. The resulting likelihood depends only on the observed values of the covariate. This approach is both clever, and practically useful, but also reinforces some of the models and ideas presented in earlier chapters (in particular, the live encounter-dead recovery models introduced in Chapter 9).

In the following, we discuss examples of both the multi-state and trinomial approaches to the problem of time-varying individual covariates when $p < 1$.

11.6.1. Continuous individual covariates & multi-state models...

Let's work through an example – not only to demonstrate an application of multi-state modeling to this sort of problem (giving you a chance to practice what you learned in Chapter 10), but also to force you to think deeply (yet again) about the building of a design matrix.

Consider a situation where we believe there is strong *directional* selection on (say) body size, where larger individuals have higher survival than do smaller individuals. Suppose we have categorized individuals as 'small' (**S**), 'medium' (**M**) and 'large' (**L**). For this example, we simulated a 6-occasion data set (`ms_directional.inp`) according to the parameter values for 'size-specific survival' tabulated at the top of the next page. If you look closely, you'll see that within each interval, the difference in the latent survival probability used in the simulation differs by a constant multiplicative factor such that there is a linear increase in survival with size.

	interval				
state	1	2	3	4	5
S	0.500	0.700	0.600	0.700	0.700
M	0.525	0.749	0.624	0.749	0.749
L	0.551	0.801	0.649	0.801	0.801

However, if you look even more closely, you'll note that the rate of this increase in survival with size is not constant over intervals. So, imagine that for each time interval, you calculate the slope of the relationship between survival and size. This slope should show heterogeneity among intervals (i.e., the strength of directional selection on size varies over time).

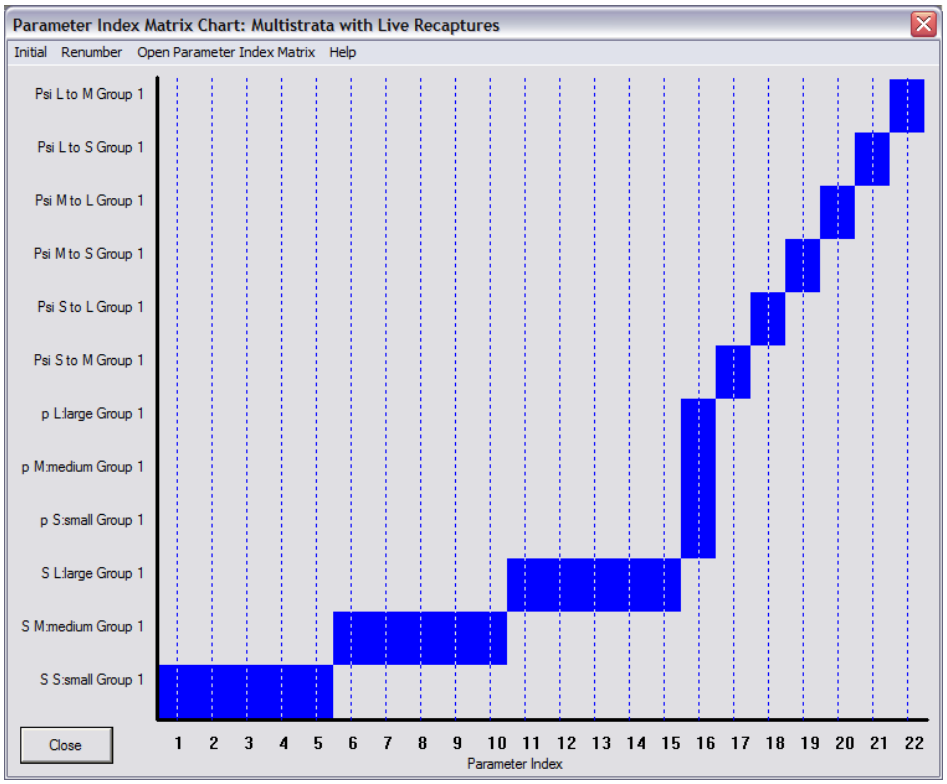
To make things 'fun' (i.e., more realistic) we'll also specify some size-specific transition parameters:

		from		
		S	**M**	**L**
	S	0.7	0.0	0.0
to	**M**	0.2	0.8	0.0
	L	0.1	0.2	1.0

So, small (**S**) and medium (**M**) individuals can stay in the same size class or grow over a given interval, but individuals cannot get smaller. We'll assume that the encounter probability for all size classes and all intervals is the same; however, to make this even more realistic, we'll assume that $p = 0.7$ for all size classes – since $p < 1$, then we have 'missing covariates'.

So, start **MARK**, and begin a new '**multi-state**' analysis: select the **ms_directional.inp** file, and specify 6 occasions, and 3 states: `S`, `M`, `L`. We'll start with a general model with time-dependence in survival, among states, and among time intervals. We'll make the encounter parameter p constant among states and over time, and will make ψ constant within state.

This general structure is reflected in the following PIM chart:



Now, before we run this model, we have to consider if there are any parameters we need to fix (due to logical constraints). As noted earlier, some of the transitions are not possible; specifically, $\psi^{MS} = \psi^{LM} = \psi^{LS} = 0$. Thus, looking at our PIM chart, we see that this corresponds to setting parameters 19, 21 and 22 to 0.

Go ahead and fix these parameters in the numerical estimation setup window. Call this model '`s(state*time)p(.)psi(state)`', run it, and add the results to the browser. If we look at the estimates, we'll see that, by and large, the values are consistent with the underlying model structure.

OK – on to the 'clever' design matrix we alluded to before. The model we just fit is a naïve model, as far as our underlying hypothesis is concerned – it is a model which simply allows the estimates for survival to vary among states, and over time. In essence, a simple heterogeneity model. By itself, this is not particularly interesting, although it is arguably a reasonable null model.

But, we're interested in a particular *a priori* hypothesis: specifically, that survival increases with size. We may also suspect that the strength (magnitude) of this directional selection favoring larger sized

individuals varies over time. So, what we want to fit is a model where, within a given interval, survival is constrained to be a linear function of size (i.e., follow a trend), and that the slope of this trend may vary over time.

So, here's the tricky bit – in effect, we're now going to treat each time interval as a group, and ask if the slope of a relationship between survival and size varies among levels of this group (i.e., among time intervals). So, we need to figure out how to do two things: (1) build a design matrix where each time interval is a group, and (2) within a time interval, have survival constrained to follow a trend with size among states (i.e., an ordinal constraint on survival with increasing size). How do we do this?

Well, with a bit of thought, you might see your way to the solution. First, start by writing out the linear model. We know we need an intercept (β_1). There are 6 occasions, so 5 time intervals, meaning we need 4 columns in the design matrix to code for the `TIME` grouping ($\beta_2 \rightarrow \beta_5$). Next, we want to impose a `TREND` over states. Recall from Chapter 6 how we handled trends: a single column consisting of an ordinal series. So, for `TREND`, one column (β_6). Next, the interaction term of `TIME` and `TREND` - $(4 \times 1) = 4$ columns for the interaction terms ($\beta_7 \rightarrow \beta_{10}$). So, for the survival parameter,

$$\begin{aligned} S = {} & \beta_1 \\ & + \beta_2(\texttt{T1}) + \beta_3(\texttt{T2}) + \beta_4(\texttt{T3}) + \beta_5(\texttt{T4}) \\ & + \beta_6(\texttt{TREND}) \\ & + \beta_7(\texttt{T1.TREND}) + \beta_8(\texttt{T2.TREND}) + \beta_9(\texttt{T3.TREND}) + \beta_{10}(\texttt{T4.TREND}) \end{aligned}$$

Now, encounter probability p is constant among states and over time, so one column (β_{11}) for that parameter. For the ψ parameters, one for each of the estimated transitions. Remember that if there are n states that there are $n(n-1)$ estimated transitions, then for 3 size states, $3(3-1) = 6$ transitions, meaning 6 columns ($\beta_{12} \rightarrow \beta_{17}$). So, in total, our design matrix should have 17 columns. So, we tell **MARK** we want to build a ' reduced design matrix, with 17 columns. **MARK** will then respond by giving us a 'blank' design matrix with 17 columns.

Starting the process of specifying our design matrix is easy enough: a column of 15 '1's for the intercept. Then, looking back at our linear model, we see that we next want to code for the 5 `TIME` intervals: 4 columns ($\beta_1 \rightarrow \beta_4$). We use the same coding scheme we're familiar with – all we want to do is make sure the dummy-variable structure unambiguously indicates the time interval:

Design Matrix Specification (B = Beta)

B1 int	B2 T1	B3 T2	B4 T3	B5 T4	B6	B7	B8	B9	Parm
1	1	0	0	0	0	0	0	0	1:S S:small
1	0	1	0	0	0	0	0	0	2:S S:small
1	0	0	1	0	0	0	0	0	3:S S:small
1	0	0	0	1	0	0	0	0	4:S S:small
1	0	0	0	0	0	0	0	0	5:S S:small
1	1	0	0	0	0	0	0	0	6:S M:medium
1	0	1	0	0	0	0	0	0	7:S M:medium
1	0	0	1	0	0	0	0	0	8:S M:medium
1	0	0	0	1	0	0	0	0	9:S M:medium
1	0	0	0	0	0	0	0	0	10:S M:mediur
1	1	0	0	0	0	0	0	0	11:S L:large
1	0	1	0	0	0	0	0	0	12:S L:large
1	0	0	1	0	0	0	0	0	13:S L:large
1	0	0	0	1	0	0	0	0	14:S L:large
1	0	0	0	0	0	0	0	0	15:S L:large

So far, so good. Now, for the 'hard part'. We now need to code for TREND. But, remember, here, we're not coding for TREND over TIME, but rather, TREND over states *within* TIME. You might remember that if we have 3 levels we want to constrain some estimate to follow a trend over, then we can use the ordinal sequence 1, 2, and 3 as the TREND covariate (check the relevant sections of Chapter 6 if you're unsure here). But, where do we put these TREND coding variables?

The key is remembering – TREND *among* states *within* TIME interval. So, here is how we code TREND for this model:

Design Matrix Specification (B = Beta)

B1 int	B2 T1	B3 T2	B4 T3	B5 T4	B6	B7	B8	B9	Parm
1	1	0	0	0	1	0	0	0	1:S S:small
1	0	1	0	0	1	0	0	0	2:S S:small
1	0	0	1	0	1	0	0	0	3:S S:small
1	0	0	0	1	1	0	0	0	4:S S:small
1	0	0	0	0	1	0	0	0	5:S S:small
1	1	0	0	0	2	0	0	0	6:S M:medium
1	0	1	0	0	2	0	0	0	7:S M:medium
1	0	0	1	0	2	0	0	0	8:S M:medium
1	0	0	0	1	2	0	0	0	9:S M:medium
1	0	0	0	0	2	0	0	0	10:S M:mediu
1	1	0	0	0	3	0	0	0	11:S L:large
1	0	1	0	0	3	0	0	0	12:S L:large
1	0	0	1	0	3	0	0	0	13:S L:large
1	0	0	0	1	3	0	0	0	14:S L:large

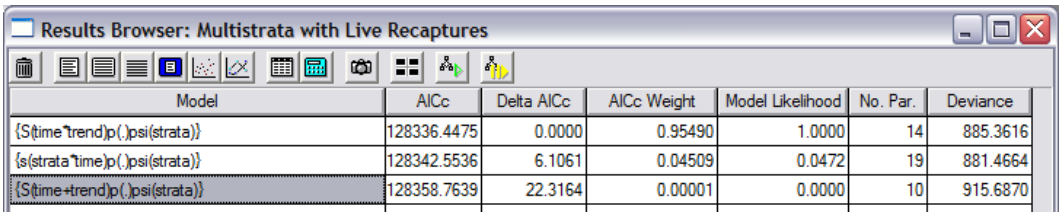
Remember, TREND *among* states *within* TIME interval. So, for the first interval for the 3 states, corresponding to rows 1, 6, and 11, respectively, we enter 1, 2 and 3. Similarly, for the second interval for the 3 states, corresponding to rows 2, 7, and 12, respectively, we again enter 1, 2 and 3, and so on for each of the intervals. Think about this – remember, TIME *is a grouping variable for this model*.

After all this, the interaction terms (and the encounter and transition parameters) are straightforward (the full design matrix is shown below):

Design Matrix Specification (B = Beta)

B2 T1	B3 T2	B4 T3	B5 T4	B6 trend	B7 T1.TR	B8 T2.TR	B9 T3.TR	B10 T4.TR	Parm	B11	B12	B13	B14	B15	B16	B17
1	0	0	0	1	1	0	0	0	1:S S:small	0	0	0	0	0	0	0
0	1	0	0	1	0	1	0	0	2:S S:small	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1	0	3:S S:small	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1	4:S S:small	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	5:S S:small	0	0	0	0	0	0	0
1	0	0	0	2	2	0	0	0	6:S M:medium	0	0	0	0	0	0	0
0	1	0	0	2	0	2	0	0	7:S M:medium	0	0	0	0	0	0	0
0	0	1	0	2	0	0	2	0	8:S M:medium	0	0	0	0	0	0	0
0	0	0	1	2	0	0	0	2	9:S M:medium	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0	10:S M:medium	0	0	0	0	0	0	0
1	0	0	0	3	3	0	0	0	11:S L:large	0	0	0	0	0	0	0
0	1	0	0	3	0	3	0	0	12:S L:large	0	0	0	0	0	0	0
0	0	1	0	3	0	0	3	0	13:S L:large	0	0	0	0	0	0	0
0	0	0	1	3	0	0	0	3	14:S L:large	0	0	0	0	0	0	0
0	0	0	0	3	0	0	0	0	15:S L:large	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	16:p S:small	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	17:Psi S to M	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	18:Psi S to L	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	19:Psi M to S	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	20:Psi M to L	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	21:Psi L to S	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	22:Psi L to M	0	0	0	0	0	0	1

Go ahead and run this model – call it '`s(time*trend)p(.)psi(state)`', where the `time*trend` part indicates an interaction of the trend among `TIME` intervals. Run the model, and add the results to the browser. Then, build the additive model (by deleting the interaction columns from the design matrix) – call this model '`s(time+trend)p(.)psi(state)`', and again, add the results to the browser.


Results Browser: Multistrata with Live Recaptures

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(time*trend)p(.)psi(strata)}	128336.4475	0.0000	0.95490	1.0000	14	885.3616
{s(strata*time)p(.)psi(strata)}	128342.5536	6.1061	0.04509	0.0472	19	881.4664
{S(time+trend)p(.)psi(strata)}	128358.7639	22.3164	0.00001	0.0000	10	915.6870

We see clearly that our model constraining survival to show a trend among states, with full interaction among time intervals, is by far the best supported model. Of course, this isn't surprising, since the data were simulated under this model.

So – a fairly complex example of using a multi-state approach to handle covariates which vary through time. And, yet another example of why it is important to have a significant level of comfort with design matrices – unless you do, you won't be able to build the 'fancy models' you'd like to.

11.6.2. the 'trinomial likelihood' approach...

Here, we analyze the simulated example data that accompany Bonner (2013). As noted earlier, the trinomial approach conditions only on these events that occur after occasions when the individual is live captured (encountered) and released (in other words, if you live encounter an individual, and presumably measure the covariate, what happens in the interval immediately following the encounter-live release event).

Following Catchpole *et al*. (2008), there are 3 possible fates in the interval following a live encounter at occasion i: (i) the individual is not captured alive on the next occasion, $(i+1)$, nor is it recovered dead over the interval following the release, from $i \rightarrow i+1$) (we will code this as `0`); (ii) the individual is encountered alive at the next occasion, $(i+1)$ (we will code this as a `1`); (iii) the individual is recovered dead over the interval from $(i \rightarrow i+1)$ following the live encounter-release at occasion i (we will code this as a `2`).

Consider the following coded encounter history:

'`11012`'

This individual was first encountered live at occasion 1, presumably marked and live released. It was then subsequently live encountered at occasion 2, released live, not encountered (in any way) either at occasion 3, or over the interval from $2 \rightarrow 3$, was encountered live at occasion 4, and then encountered as a dead recovery over the interval from $4 \rightarrow 5$.

Now, suppose that at each live encounter (occasions 1, 2 and 4) we measured some covariate, and that the standardized covariate values were 1.62, −1.26, and −0.28, respectively. In a standard **MARK** input file, the encounter history containing the covariate data would look like the following (here, we use a 'dot' to indicate the missing value – occasions when the individual wasn't live encountered, and covariate not measured as a result):

```
11012  1  1.62  -1.26  .  -0.28  .;
```

The 'trinomial approach' requires some re-formatting of these encounter data, transforming into a 'live-encounter-dead recovery history', for each of the intervals following a live encounter. First, recall from Chapter 9 that the live encounter-dead recovery data are code using the LDLD format. In other words, for each 'year' (interval) following a live sampling event, the marked individual could be encountered live, or dead (or both). The 'LD' represents encountered live (L) or dead (D). So, part of the process is to take each interval after a live encounter event, for time (i) to $(i+1)$, and transform it into a single LDLD pair. The first LD in the pair corresponds to the sampling event at (i) and the interval from $(i) \rightarrow (i+1)$), and the second LD pair corresponds to occasion $(i+1)$.

The following table will make it clear what we're about to do, as a first step. Consider an individual live encountered at time i. As noted above, there are 3 possible fates over the interval from $(i) \rightarrow (i+1)$: 0 (not encountered in any way), 1 (live encountered at $i+1$), or 2 (dead recovery over the interval from $(i) \rightarrow (i+1)$). So, the translations from the 'event' (0, 1 or 2) to the LDLD pair are:

event		*LD coding*
0	⟶	1000
1	⟶	1010
2	⟶	1100

OK, that's step 1 – understanding the translation from event to LDLD. Next, we have to 'split apart' the individual encounter history ('11012') into 3 separate LDLD histories, one corresponding to each of the 3 occasions where this individual was live encountered. Each of these 3 separate LDLD histories will be entered as a separate line in the reformatted encounter history file.

But, how do we handle 'time' in the formatting? If we simply generate an LDLD pairs for each live encounter event, how do we keep track of when the encounter events actually occur? Solution – we code each encounter time as a 'group'. So, for our example history ('11012'), we see that there are 5 live encounter occasions, so, 4 'sampling intervals'. In other words, we'll need to code 4 'time groups'. Note that for our example history, there were live encounters at occasions 1, 2 and 4.*

Following from Table 1 in Bonner (2013), here is ours example encounter history, recoded into LDLD pairs, using 'time groups' to indicate the sampling occasion/interval. Remember – each live encounter event is coded as an LDLD pair, as a separate line in the reformatted encounter history file. For our example, 3 live encounter events, 3 separate LDLD coded lines in the reformatted file (the 'time group' coding is shown in green font):

event history		*LD coded history*	*'time group'*
11012	⟶	1010 1 0 0 0	(live encounter occasion 1, 'time group' 1)
		1000 0 1 0 0	(live encounter occasion 2, 'time group' 2)
		1100 0 0 0 1	(live encounter occasion 4, 'time group' 4)

Look closely – make sure you see exactly what we've done here, and how the 'time group' coding indicates the interval after each live encounter event.

Finally, the individual covariates (being ultimately the point of the exercise). Recall that for this individual, the covariate values of 1.62, −1.26 and −0.28 were obtained at each of the 3 live encounter events (occasions 1, 2 and 4). Since each live encounter event for an individual is treated as an individual

* Also, don't get tripped up here thinking about 'time' in the way that we do when coding it in the design matrix. In that case, for (k) intervals, you need $(k-1)$ columns to uniquely code for each interval, using 0/1 dummy variables. Here we are using 'groups' to indicate explicitly which 'time interval' the encounter occurs in – we have a separate column for each 'time group', just like we might have (say) a separate column for males and females for a 'gender group'.

record (line) in the file, we simply tack on the covariate value for each live encounter event at the end of the corresponding line in the file (as shown below – covariate in blue font):

event history		*LD coded history*				
11012	⟶	1010	1	0	0	0 1.62;
		1000	0	1	0	0 -1.26;
		1100	0	0	0	1 -0.28;

Here are a couple of more examples – its very important you see the connection between the original encounter (event) history, and the reformatted histories:

event history	*covariates*	*LD coded history*				
11100	0.48, −0.45, 0.00, ·, ·	1010	1	0	0	0 0.48;
		1010	0	1	0	0 -0.45;
		1000	0	0	1	0 0.00;
10102	0.34, ·, −2.61, ·, ·	1010	1	0	0	0 0.48;
		1010	0	1	0	0 -0.45;
		1000	0	0	1	0 0.00;

Note that *all* of the reformatted LD coded histories are 4 characters wide (i.e., a single LDLD pair), no matter how many occasions there are in the original event history (in our examples, above, there were 5 sampling occasions). And, also note that the maximum number of entries (lines) for an individual in the reformatted file is the number of sampling intervals, if the individual was encountered at each sampling occasion. And finally, note that we're coding only the interval after the live encounter occasion – this means that the second D of the LDL**D** pair (indicated here in blue, **bold** font) has to be fixed to 0 (since this second D refers to the *second* interval; this becomes important when we actually analyze data in **MARK**).

Once you have your brain wrapped around what the re-formatting is doing, there is the pesky problem of 'mechanics'. It is probably obvious that you really don't want to do the re-formatting by hand, and probably also obvious you don't want to spend much time writing your own code to reformat your data. Fortunately, Simon Bonner has written an **R** package (**tRiMark**), which will handle the bulk of the re-formatting for you. We won't describe it here – consult the package documentation on your own.

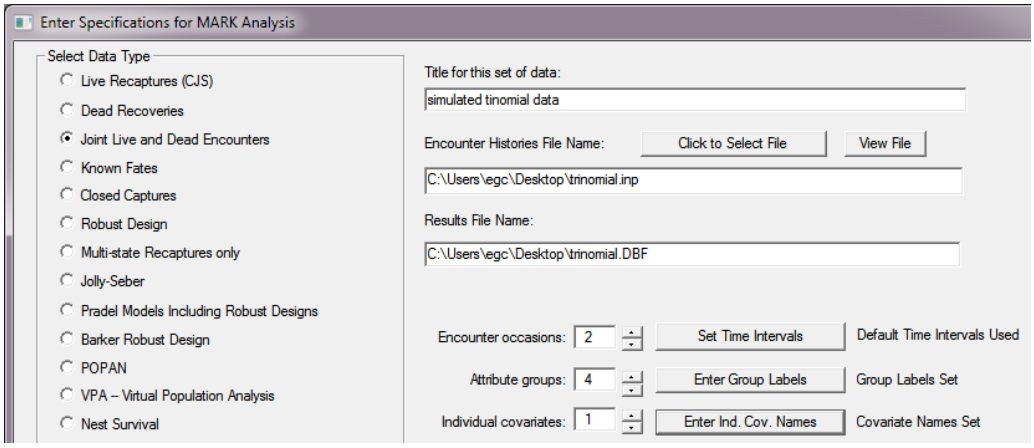
The re-formatted data we'll analyze are contained in **trinomial.inp** (see Bonner (2013) for the details). The original, pre-formatted data consisted of encounter data over 5 sample occasions, so 4 intervals, and thus 4 'time group' frequency columns in the input file. Here are the first few lines of **trinomial.inp**:

```
1010 1 0 0 0 0.24 ;
1000 0 1 0 0 -0.03 ;
1010 1 0 0 0 0.48 ;
1010 0 1 0 0 -0.45 ;
1000 0 0 1 0 0 ;
1010 1 0 0 0 0.22 ;
```

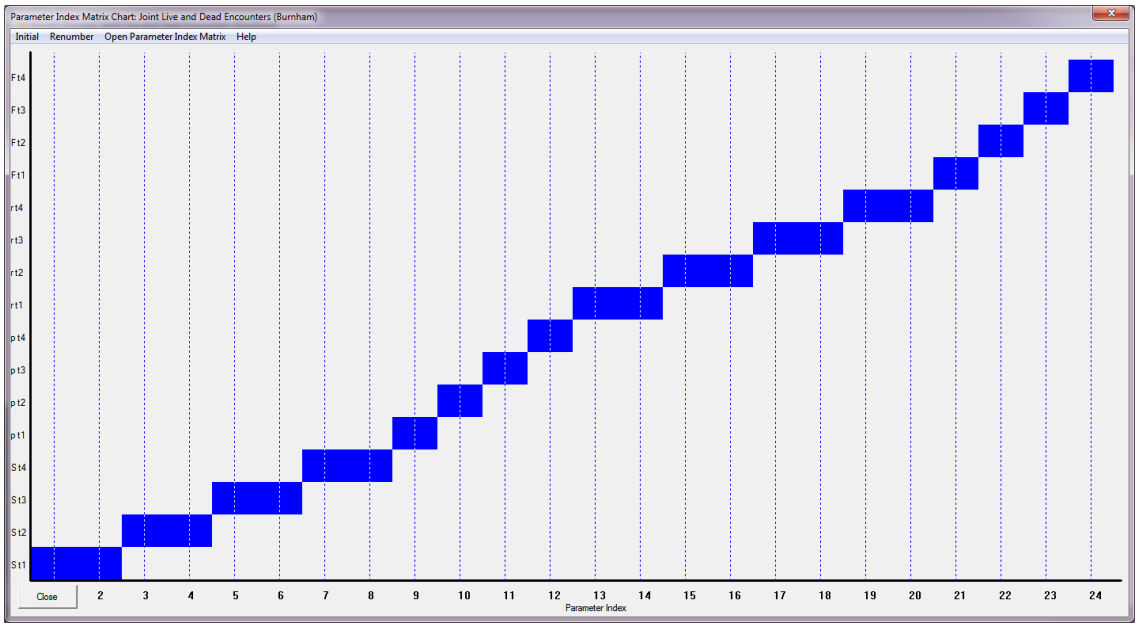
The data were simulated under a true generating model where survival was a linear function of a single continuous covariate, with no temporal variation in the relationship (i.e., variation in survival is a function of variation in the covariate only). There was temporal variation for both live encounter and dead recovery probabilities. The true parameter values can be found in Table 2 in Bonner (2013).

OK, we're finally ready to start or analysis in **MARK**. To begin, we need specify that we're using a '**`Joint Live and Dead Encounters | Burnham`**' data type (Chapter 9; as discussed in Bonner (2013), the trinomial approach could be extended for other permutations of the 'live encounter-dead recovery' data type, but we'll focus on the simple Burnham model here).

Now, a bit of thought needed. We need to 'tell' **MARK** the number of encounter occasions, the number of groups, and the number of covariates. For the trinomial analysis, where the data are re-formatted to `LDLD` pairs, the number of occasions for each record (line) in the input file is 2. The number of groups is 4 (corresponding to each of the 4 intervals in the 5 occasion study). We'll call them `t1`, `t2`, `t3` and `t4`, respectively. And finally, there is (for this example) a single covariate (which we'll simply call `cov1`):



Here is the default PIM structure:



Pretty densely packed, but if you look closely, note that there are two parameters for survival (S) and recovery (r), for each of the 4 'time groups', whereas there is only a single parameter for live encounter,

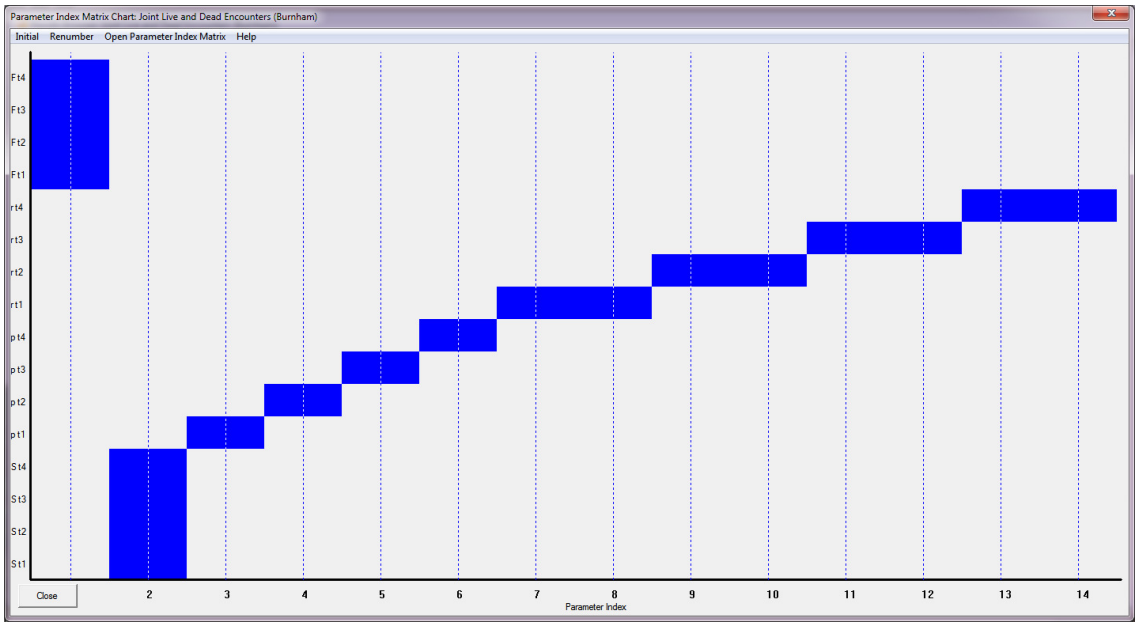
p, and fidelity, F, for each of the same'time-groups'.

If you think about it for a moment, this should make sense. For a single LDLD pair, there are in fact 2 survival 'intervals', and 2 recovery 'intervals'. On the other hand, for a single LDLD pair, where we're conditioning on a live encounter for the first L, there is only one subsequent live encounter possible, and only time step to estimate fidelity.

However, recall from above that we need to think a bit about the second LD pair – specifically, we need to remember that we'll need to fix the recovery probability for this second pair (i.e., second interval coded in the pair) to 0.

Also, with a bit more thought, you might realize (remembering from Chapter 9), that the fidelity parameter needs to be fixed to 1. We have to assume that over the LDLD pair, the individual says in the sample – because the LDLD pair is conditional on just that – the individual record in the input file is conditional on the live encounters, which can only occur if the individual is in the sample.

So, we start by simplifying the PIMs a bit, reflecting both the 'logic constraints' we noted above, and our knowledge of the true generating model for these encounter data. First, we'll drag all the fidelity parameter 'boxes' so they overlap, and then drag all of them over to the left. Since we're always going to fix this parameter to 1, it is convenient to move them all over to the left, so that we don't have to worry about the parameter indexing for fidelity changing if we change the structure for the other parameters. And, since the data were simulated assuming no temporal variation in survival, apart from variation induced by the covariate, we'll make survival constant, and overlap the boxes over all 'time groups':



All we now need to do is to remember that fidelity, F (parameter 1, above) will need to be fixed to 1, and that the second r parameter for each 'time-group' will need to be fixed to 0 for all of the models. Go ahead and fix the recovery (r) parameters to 0, and the fidelity (F) parameter to 1, and run the model (we'll call it '`starting model - PIM`'), using the logit link, and add the results to the browser.

Next, we remember that in order to consider individual covariates, we need a model based on the design matrix (DM). One (of many) possible DM corresponding to the model we just constructed using PIMs, is shown at the top of the next page.

Design Matrix Specification (B = Beta)

B1 F	B2 S	B3 p-incpt	B4 p1	B5 p2	B6 p3	B7 r-incpt	Parm	B8 tg1	B9 tg2	B10 tg3	B11 time	B12 tg1*time	B13 tg2*time	B14 tg3*time
1	0	0	0	0	0	0	1:F	0	0	0	0	0	0	0
0	1	0	0	0	0	0	2:S	0	0	0	0	0	0	0
0	0	1	1	0	0	0	3:p	0	0	0	0	0	0	0
0	0	1	0	1	0	0	4:p	0	0	0	0	0	0	0
0	0	1	0	0	1	0	5:p	0	0	0	0	0	0	0
0	0	1	0	0	0	0	6:p	0	0	0	0	0	0	0
0	0	0	0	0	0	1	7:r	1	0	0	1	1	0	0
0	0	0	0	0	0	1	8:r	1	0	0	0	0	0	0
0	0	0	0	0	0	1	9:r	0	1	0	1	0	1	0
0	0	0	0	0	0	1	10:r	0	1	0	0	0	0	0
0	0	0	0	0	0	1	11:r	0	0	1	1	0	0	1
0	0	0	0	0	0	1	12:r	0	0	1	0	0	0	0
0	0	0	0	0	0	1	13:r	0	0	0	1	0	0	0
0	0	0	0	0	0	1	14:r	0	0	0	0	0	0	0

If you run this DM-based model (we'll call it '`starting model - DM`' – remember to fix the appropriate parameters first), you'll get the same fit statistics as with the PIM-based model we ran initially (if not, you've made a mistake somewhere).

Now that we have a DM-based model, we can add our covariate (`cov1`). Since we're only interested in the relationship of the covariate to survival, the modification to the DM is straightforward (as shown below) – we can leave everything else in the DM the same:

B1: F	B2 indiv cov	B3:	B4: p-incpt	B5: p1	B6: p2	B7: p3	Parm
1	0	0	0	0	0	0	1:F
0	1	cov1	0	0	0	0	2:S
0	0	0	1	1	0	0	3:p
0	0	0	1	0	1	0	4:p
0	0	0	1	0	0	1	5:p
0	0	0	1	0	0	0	6:p
0	0	0	0	0	0	0	7:r

If we run this model (we'll call it '`starting model - covariate`'), and add the results to the browser, we see (below) that the model with the covariate does much better than the starting model without the covariate (which is not surprising, given this was the model under which they data were simulated in the first place).

Model	AICc	Delta AICc	AICc Weight	No. Par.	Deviance	-2Log(L)
{starting model - covariate}	1229.3341	0.0000	1.00000	9	1211.1758	1211.1758
{starting model - PIM}	1306.3642	77.0301	0.00000	8	1290.2377	1290.2377
{starting model - DM}	1306.3642	77.0301	0.00000	8	1290.2377	1290.2377

So, the trinomial approach, using a live encounter-dead recovery data type, is yet another way to handle temporally varying covariates when $p < 1$. Bonner (2013) is the canonical reference to this approach as implemented in **MARK**, and the larger literature on time-varying covariates.

11.7. Individual covariates as 'group' variables

Suppose you were interested in whether or not survival probability differed between male and female dippers. Having come this far in the book, you'll probably regard this as a trivial exercise – you specify the PIMs corresponding to the two sexes, perhaps construct the corresponding design matrix, and proceed to fit the models in your candidate model set. This is all fairly straightforward, and easy to implement – in part because the problem is sufficiently 'small' (meaning, only two parameters, relatively few occasions, only two groups) that the overall number, and complexity of the PIMs you construct (and the corresponding design matrix) is small. But, as we've seen, especially for 'large' problems (many parameters, many occasions, many PIMs), manipulating all the PIMs and the design matrix can become cumbersome (even given the convenience of manipulating the PIMs using the PIM chart).

Is there an option? Well, as you might guess, given that this chapter concerns the use of individual covariates, you can, for a number of categorical models, use an alternative approach based on individual covariates. Such an approach can in some cases be easier and more efficient to implement. We'll consider a couple of examples here, starting with the dippers.

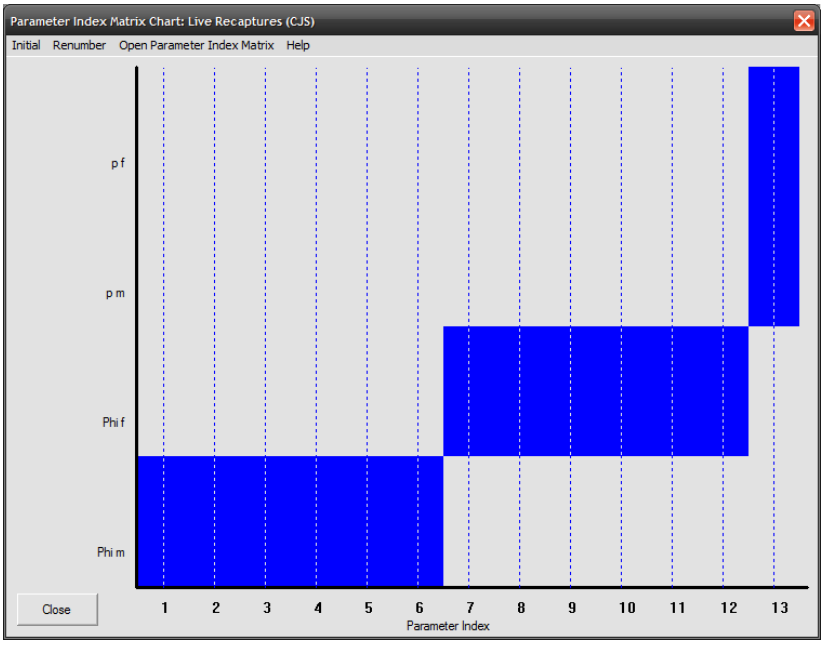
11.7.1. Individual covariates for a binary classification variable

Let's consider fitting the following 3 candidate models to the data collected for male and female dippers:

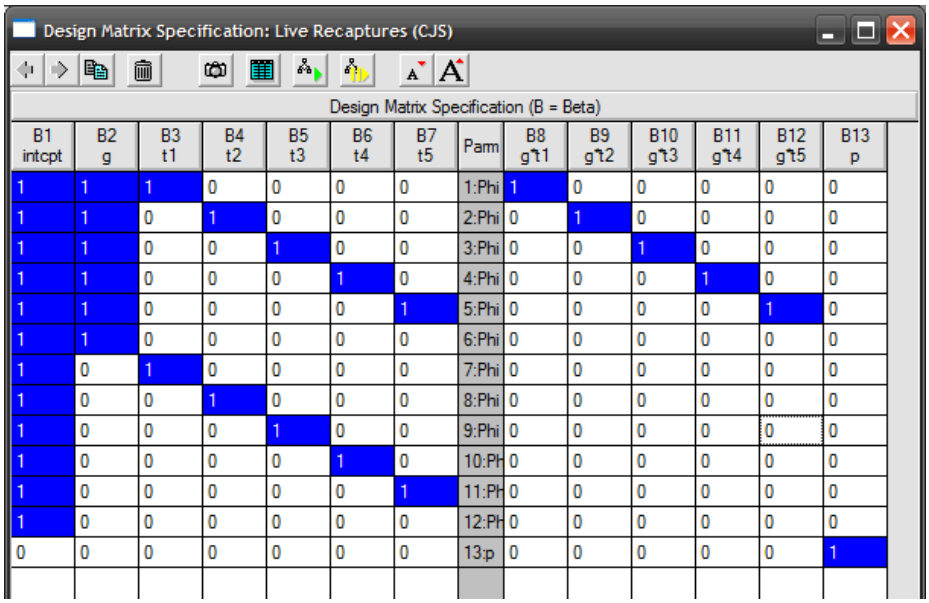
$$\{\varphi_{g*t}\, p.\}, \{\varphi_{g+t}\, p.\}, \{\varphi_{g}\, p.\},$$

where g is the 'grouping' variable – in this case, sex (male or female). Recall that the dipper data (**ed.inp**) consist of live encounter data collected over 7 encounter occasions. We specify 2 attribute groups in the data type specification window in **MARK** (which we'll label `m` and `f`, respectively), and proceed to fit the three models in the candidate model set.

To specify the underlying parameter structure for our general model $\{\varphi_{g*t}\, p.\}$, we'll use fully time-dependent PIMs for survival, and constant PIMs for the encounter probability:



with the following corresponding design matrix:



B1 intcpt	B2 g	B3 t1	B4 t2	B5 t3	B6 t4	B7 t5	Parm	B8 gt1	B9 gt2	B10 gt3	B11 gt4	B12 gt5	B13 p
1	1	1	0	0	0	0	1:Phi	1	0	0	0	0	0
1	1	0	1	0	0	0	2:Phi	0	1	0	0	0	0
1	1	0	0	1	0	0	3:Phi	0	0	1	0	0	0
1	1	0	0	0	1	0	4:Phi	0	0	0	1	0	0
1	1	0	0	0	0	1	5:Phi	0	0	0	0	1	0
1	1	0	0	0	0	0	6:Phi	0	0	0	0	0	0
1	0	1	0	0	0	0	7:Phi	0	0	0	0	0	0
1	0	0	1	0	0	0	8:Phi	0	0	0	0	0	0
1	0	0	0	1	0	0	9:Phi	0	0	0	0	0	0
1	0	0	0	0	1	0	10:Ph	0	0	0	0	0	0
1	0	0	0	0	0	1	11:Ph	0	0	0	0	0	0
1	0	0	0	0	0	0	12:Ph	0	0	0	0	0	0
0	0	0	0	0	0	0	13:p	0	0	0	0	0	1

We'll skip the details on how to modify this design matrix to specify the remaining two models in the model set (you should be pretty familiar with this by now).

The results of fitting the three models to the dipper data are shown below:

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(g)p(.) - DM}	672.7331	0.0000	0.83484	1.0000	3	84.1991
{phi(g+t)p(.) - DM}	675.9945	3.2614	0.16346	0.1958	8	77.1720
{phi(gt)p(.) - DM}	685.1244	12.3913	0.00170	0.0020	13	75.7638

Now, let's consider using an individual covariate approach to fitting the same three models to the dipper data. Our first step involves reformatting the input file. We need to reformat the input file to specify gender as an individual covariate. Much like with the design matrix, you need to consider how many covariates you need to specify group (in this case, sex). Clearly, in this case, the grouping variable is binary (has only two states), and thus we need only a single covariate to indicate group (sex).

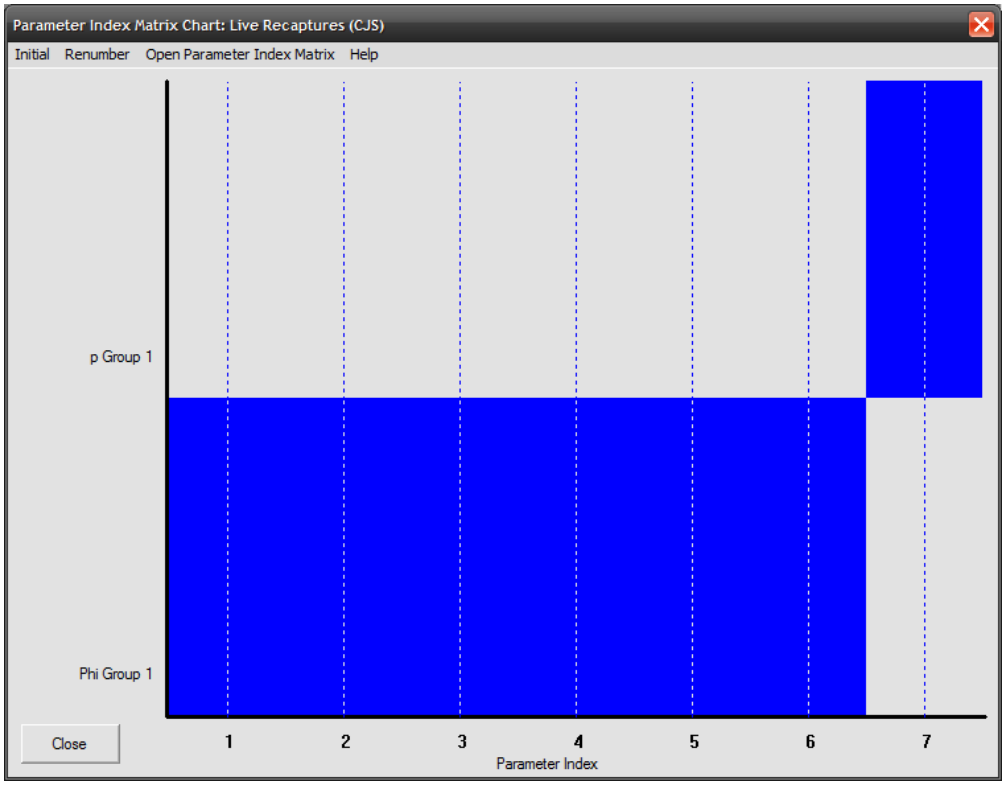
How do we reformat our data, using a single covariate to indicate sex? We'll use '1' to indicate males, and '0' to indicate females. Now, we reformat the dipper data as follows – consider the following table of different encounter histories (selected from the original **ed.inp** file in 'standard' format), which we've transformed to use an individual covariate approach:

standard	*reformatted*
`1111110 1 0;`	`1111110 1 1;`
`1111100 0 1;`	`1111100 1 0;`
`1111000 1 0;`	`1111000 1 1;`
`1111000 0 1;`	`1111000 1 0;`
`1101110 1 0;`	`1101110 1 1;`

The key is to remember that under the original 'standard' formatting, there is one column in the input file for each of the groups: two sexes, two columns following the encounter history itself. So, a '1 0' indicates male (1 in the male column, 0 in the female column), and a '0 1' indicates a female (0 in the male column, 1 in the female column). When using an individual covariates approach, you have only one column for the covariate.

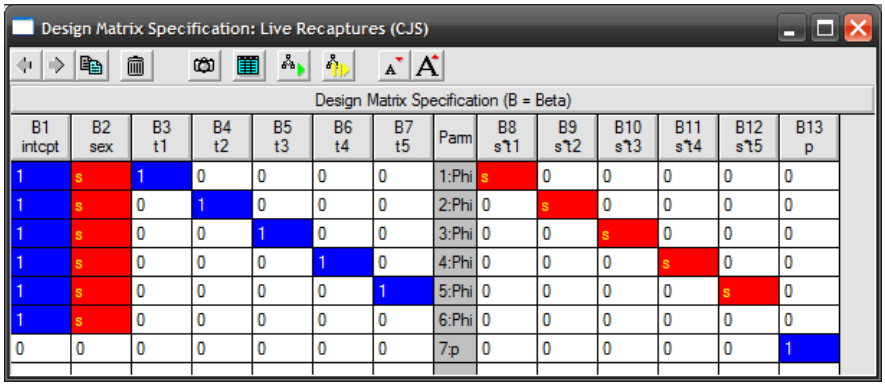
But, notice there are 2 columns after the encounter history. Why? Don't we need just 1 covariate column? Yes, but remember that we also need a column of '1's' to indicate the frequency of number of individuals with a given encounter history (and since we're working with individual covariates, each encounter history corresponds to one individual, hence the frequency column has a '1' in it for each individual history). The first column after the encounter history is the frequency, and the second column is the covariate column for group (sex). So, a male in the original file (indicated by '1 0') becomes '1 1' in the reformatted file, and a female in the original file (indicated by '0 1') becomes '1 0' in the reformatted file. The reformatted data are contained in the file **dipper_ind.inp** (we'll leave it to you to figure out an efficient way to transform your data from one format to the other).

Now, when we specify the data type in **MARK**, we do not indicate 2 attribute groups, but instead change the default number of individual covariates from 0 to 1. We'll call this covariate s (for sex). If we make the encounter probability constant, the corresponding PIM chart should look like the one pictured below:



Note that there are now only 6 parameters in the PIM chart for survival, instead of the 12 parameters specified in the PIM chart of our general model using the standard input format. Obviously, we're going to need to make up the difference somehow. In fact, you may have already guessed – by entering the individual covariates into the design matrix.

For our general model $\{\varphi_{g*t}\, p_{.}\}$, here is the corresponding design matrix using individual covariates:



Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

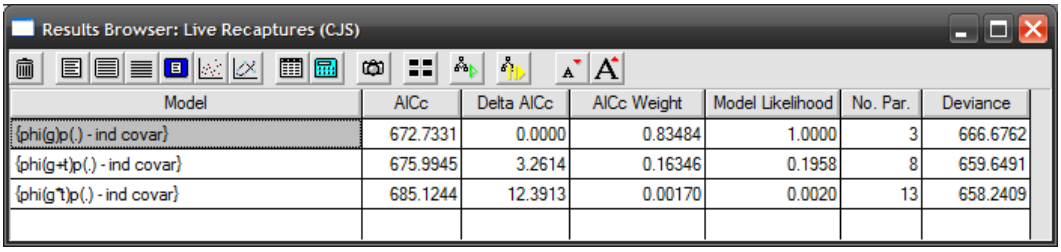
B1 intcpt	B2 sex	B3 t1	B4 t2	B5 t3	B6 t4	B7 t5	Parm	B8 s*t1	B9 s*t2	B10 s*t3	B11 s*t4	B12 s*t5	B13 p
1	s	1	0	0	0	0	1:Phi	s	0	0	0	0	0
1	s	0	1	0	0	0	2:Phi	0	s	0	0	0	0
1	s	0	0	1	0	0	3:Phi	0	0	s	0	0	0
1	s	0	0	0	1	0	4:Phi	0	0	0	s	0	0
1	s	0	0	0	0	1	5:Phi	0	0	0	0	s	0
1	s	0	0	0	0	0	6:Phi	0	0	0	0	0	0
0	0	0	0	0	0	0	7:p	0	0	0	0	0	1

We see that it has 13 columns, corresponding to 13 estimable parameters – we know from our initial analysis that this model does indeed have 13 estimable parameters. From this design matrix, we can build the other two models in the candidate model set:

$$\{\varphi_{g+t}\, p_{.}\}, \{\varphi_{g} p_{.}\},$$

simply by deleting the appropriate columns from the design matrix (e.g., for the additive model $\{\varphi_{g+t}\, p_{.}\}$, we simply delete the interaction columns $8 \rightarrow 12$).

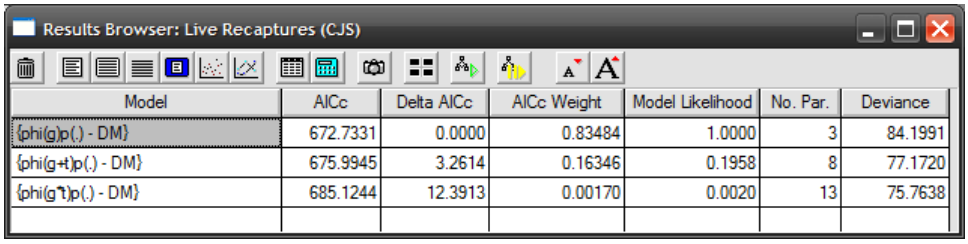
Here are the model fits for the 3 models, built using the individual covariates approach:



Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(g)p(.) - ind covar}	672.7331	0.0000	0.83484	1.0000	3	666.6762
{phi(g+t)p(.) - ind covar}	675.9945	3.2614	0.16346	0.1958	8	659.6491
{phi(g*t)p(.) - ind covar}	685.1244	12.3913	0.00170	0.0020	13	658.2409

Compare them with the results obtained using the standard approach where sex was treated as an 'attribute group':



Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(g)p(.) - DM}	672.7331	0.0000	0.83484	1.0000	3	84.1991
{phi(g+t)p(.) - DM}	675.9945	3.2614	0.16346	0.1958	8	77.1720
{phi(g*t)p(.) - DM}	685.1244	12.3913	0.00170	0.0020	13	75.7638

We see that the model AIC_c values, and the number of parameters, are identical between the two. However, the deviances are different. Does this indicate a problem? No – not if you think about it for

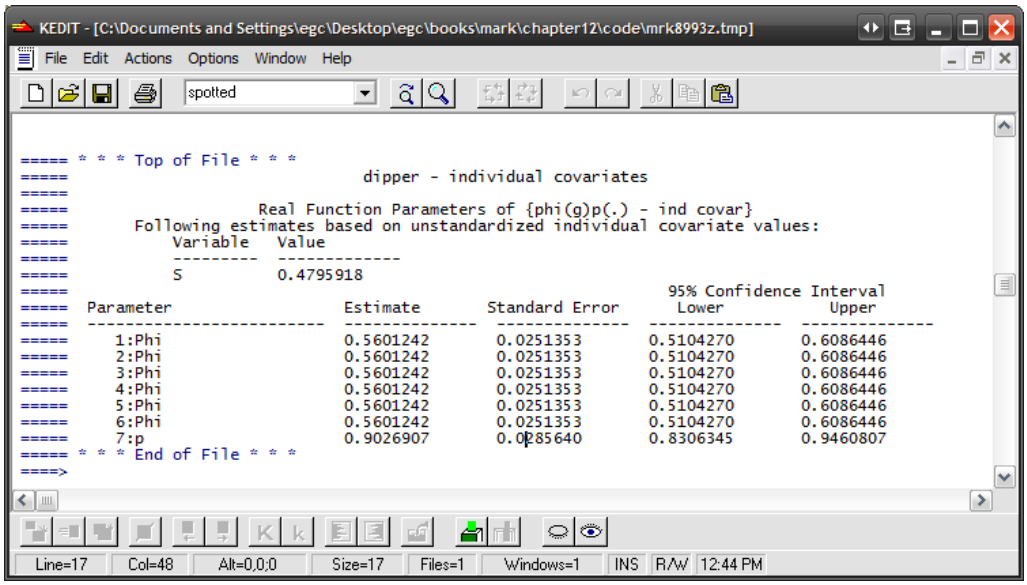
a moment. If the AIC_c values and the number of parameters are the same, then the likelihoods for the models are also the same (since the AIC_c is simply a function of the sum of the likelihood and the number of parameters – if two out of the three are the same between the different analyses, then so must the third (likelihood) be the same). In fact, if you look closely at the deviances, you'll see that the *difference* between the deviances – which is related to the likelihood (as discussed elsewhere) – is identical. For example, (666.6762 – 659.6491) = (84.1991 – 77.1720) = 7.0271.

So, the results are identical, regardless of the approach taken (attribute groups versus individual covariates coding for groups). And, it is pretty clear that the number of PIMs and the design matrix for the analysis using individual covariates is smaller (easier to handle, potentially less prone to errors) when using individual covariates. As such, is there any reason *not* to use the individual covariate approach to handling groups?

There are at least two possible reasons why you might not want to use the individual covariate approach for coding groups. First, as discussed earlier in this chapter, execution time generally increases for models involving individual covariates. For very large, complex data sets, this can be a significant issue.

Second, and perhaps more important, while the individual covariate approach might simplify aspects of building the models, in fact it complicates derivation (reconstitution) of group-specific parameter estimates. For example, take estimates of φ from our simplest model, $\{\varphi_g p_.\}$. Using the standard attribute group approach, the estimates **MARK** reports for male and female survival are $\hat{\varphi}_m = 0.5702637$ and $\hat{\varphi}_f = 0.5507352$, respectively.

What does **MARK** report for the estimates for this model fit using the individual covariates approach?

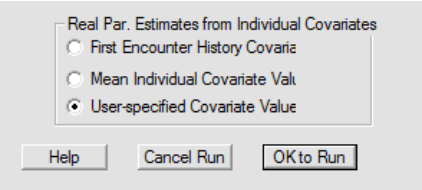


Clearly, the reported estimates using individual covariates appear to be quite different. But, are they? What does the value $\hat{\varphi} = 0.5601242$ represent? What about the value 0.4795918 reported for the sex covariate, s? How can we reconstitute separate estimates of apparent survival for both males and females?

The key is remembering that this analysis is based on *individual* covariates. Recall that **MARK** defaults to reporting the parameter estimates for the mean value of the covariate. In this case, the sex covariate is 1 (indicating male) or 0 (indicating female). If the sex-ratio of the sample was exactly 50:50, then

the mean value of the covariate would be 0.5. In fact, in the dipper data set, 47.96% of the individuals are male. Does that number look familiar? It should – it is the value of 0.4796 reported (above) as the average value of the covariate. And, the estimates of $\hat{\varphi}$ are the reconstituted values of survival for an average individual. Thus, the value of 0.5601242 is essentially identical (to within rounding error) to the weighted average of $\hat{\varphi}_m = 0.5702637$ and $\hat{\varphi}_f = 0.5507352$, which we obtained from the analysis using attribute groups ($[0.4796 \times 0.5702] + [0.5204 \times 0.5507]) = 0.5601$) – here, the weights are the frequencies of males and females in the sample (i.e., the sex ratio of the sample).

OK – fine, but that still doesn't answer the practical question of how to reconstitute separate survival estimates for males and females? The 'brute-force' approach is to use a '**`user-specified covariate value`**', when you setup the numerical estimation. You do this by checking the appropriate radio button:



Now, when you click the '**`OK to run`**' button, **MARK** will ask you to specify the individual covariate value for that model – in this case, either a 1 (for male) or 0 (for female). If we enter a '`1`', run the model, and then look at the reconstituted parameter estimates, **MARK** shows $\hat{\varphi} = 0.5703$, which is exactly what we expected for males. Similarly, if instead we enter a '`0`' for the covariate value, **MARK** shows $\hat{\varphi} = 0.5507$ for females, again, precisely matching the estimate from the model fit using attribute groups.

OK, that is a functional solution, but not one that is particularly elegant (it can also be cumbersome if you have multiple levels of group, or a lot of interactions between one or more grouping variables and – say – time). It also is somewhat devoid of 'thinking', which is rarely a good strategy, since not understanding what **MARK** is doing when you 'click this button' or 'that button' will catch up with you sooner or later. The key to understanding what is going on is to remember from earlier in this chapter how parameter estimates were reconstituted for a given value of one or more individual covariates. Essentially, all you need to do is calculate the value of the parameter on the logit scale (assuming you're using the default logit link), and then back-transform to the real probability scale. For model $\{\varphi_g\, p_.\}$, the linear model is

$$\begin{aligned}\text{logit}(\hat{\varphi}) &= \beta_1 + \beta_2(s) \\ &= 0.2036416 + 0.0792854(s).\end{aligned}$$

So, if the value of the covariate is 1 (for males), then

$$\begin{aligned}\text{logit}(\hat{\varphi}_m) &= \beta_1 + \beta_2(s) \\ &= 0.2036416 + 0.0792854(1) \\ &= 0.282927,\end{aligned}$$

which, when back-transformed from the logit scale to the normal probability scale,

$$\hat{\varphi}_m = \frac{e^{0.282927}}{1 + e^{0.282927}} = 0.570264,$$

which is identical (within rounding error) to the estimate for male survival **MARK** reports using either

the attribute group approach, or by specifying the value of the covariate in the numerical estimation using the individual covariate approach. The same is true for reconstituting the estimate for females.

While this is easy enough, it can get tiresome, especially if the linear model you're working with is 'big and ugly'. Even for fairly simple models like $\{\varphi_{g*t}\, p_.\}$, the linear model you need to work with can be cumbersome:

$$\begin{aligned}\text{logit}(\varphi) = \beta_1 + \beta_2(s) + \beta_3(t_1) + \beta_4(t_2) + \beta_5(t_3) + \beta_6(t_4) + \beta_7(t_5) \\ + \beta_8(s \cdot t_1) + \beta_9(s \cdot t_2) + \beta_{10}(s \cdot t_3) + \beta_{11}(s \cdot t_4) + \beta_{12}(s \cdot t_5)\end{aligned}$$

Each extra term in the equation adds to the possibility you'll make a calculation error. The complexity of the linear equation you need to work with will clearly be increased if you have > 2 levels of a grouping factor. We consider just such a situation in our final example.

11.7.2. Individual covariates for non-binary classification variables

Here, we consider the analysis of a simulated data set with 3 levels of some grouping variable (we'll call the grouping variable `colony`, and the three levels 'poor', 'fair', and 'good', reflecting the impact of some colony attribute on – say – apparent survival). The true model under which the simulated data (contained in **`cjs3grp.inp`**) were generated is model $\{\varphi_{g+t}\, p_.\}$ – additive survival differences among the 3 colonies (in fact, additive, and ordinal, such that $\varphi_g > \varphi_f > \varphi_p$, although this ordinal sequencing isn't of primary interest here). In the input file, the group columns (from left to right) indicate the `poor`, `fair` and `good` colonies, respectively. For our model set, we'll fit the same models (structurally) that we used for the dipper data used in the preceding example: $\{\varphi_{g*t}\, p_.\}$, $\{\varphi_{g+t}\, p_.\}$, $\{\varphi_g\, p_.\}$. Here are the results for the analysis of the data formatted using the attribute grouping approach:

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(g+t)p(.) - DM}	21186.9257	0.0000	0.98007	1.0000	8	170.9553
{phi(g*t)p(.) - DM}	21194.7161	7.7904	0.01993	0.0203	16	162.7104
{phi(g)p(.) - DM}	21256.8935	69.9678	0.00000	0.0000	4	248.9325

As expected, model $\{\varphi_{g+t}\, p_.\}$ has virtually all of the support in the data (it should, given that it was the true model under which the data were simulated in the first place).

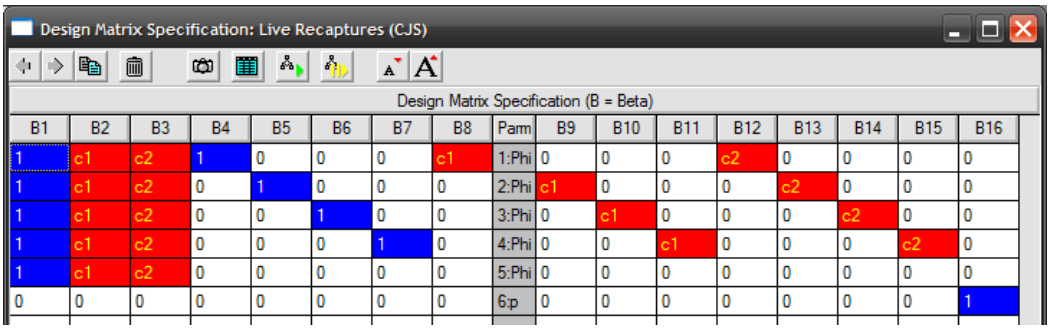
Now, let's recast this analysis in terms of individual covariates. As noted in the preceding example, we need to specify enough covariates to correctly specify group association. Your first thought might be to use a single column, with (say) a covariate value of 1, 2 or 3 to indicate a particular colony. This would work, but the model you'd be fitting would be one where you'd be constraining the estimates to following a strict ordinal trend (this is strictly analogous to how you built trend models in Chapter 6).

What if we simply want to test for heterogeneity among colonies? This, of course, is the null hypothesis of the standard analysis of variance. Since there are 3 colonies, then (perhaps not surprisingly) we need 2 columns of covariates to uniquely code for the different colonies. In effect, we're using *exactly* the same logic in constructing the covariate columns as we would in constructing corresponding columns in the design matrix. In fact, it is reasonable to describe what we're doing here – with individual covariates – as 'moving' the basic linear structure out the of the design matrix, and coding it explicitly in the input file itself.

We'll call the covariates `c1` and `c2`. For dummy coding of the colonies, we'll let '`1 0`' indicate the first (`poor`) colony, '`0 1`' indicate the second (`fair`) colony, and '`1 1`' indicate the third (`good`) colony.

So, the encounter history '`111011 1 0 0`' in the original file (indicating an individual from the poor colony) would be recoded as '`111011 1 1 0`'. Again, the first column after the encounter history after recoding is the frequency column, and is a '1' for all individuals (regardless of which colony they're in). The following two columns indicate values of the covariates `c1` and `c2`, respectively. The reformatted encounter histories are contained in **`csj3ind.inp`**.

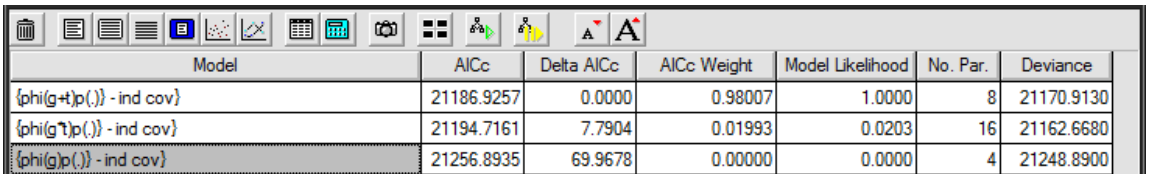
Now, when we specify the data type in **MARK**, we set the number of individual covariates to 2, and label them as `c1` and `c2`, respectively. The design matrix corresponding to the most general model in the candidate model set $\{\varphi_{g*t}\, p_{.}\}$ is shown below:



B1	B2	B3	B4	B5	B6	B7	B8	Parm	B9	B10	B11	B12	B13	B14	B15	B16
1	c1	c2	1	0	0	0	c1	1:Phi	0	0	0	c2	0	0	0	0
1	c1	c2	0	1	0	0	0	2:Phi	c1	0	0	0	c2	0	0	0
1	c1	c2	0	0	1	0	0	3:Phi	0	c1	0	0	0	c2	0	0
1	c1	c2	0	0	0	1	0	4:Phi	0	0	c1	0	0	0	c2	0
1	c1	c2	0	0	0	0	0	5:Phi	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	6:p	0	0	0	0	0	0	0	1

Column 1 is the intercept, columns 2-3 are the covariates `c1` and `c2` (respectively), columns 4 $\rightarrow$ 7 are the time intervals (6 occasions, 5 intervals), columns 8 $\rightarrow$ 12 and 13 $\rightarrow$ 17 are the interactions of the covariates `c1` and `c2` with time, respectively. Column 18 is the constant encounter probability. Go ahead and fit this model to the data – notice immediately how much longer it takes **MARK** to do the numerical estimation (again, one of the penalties in using the individual covariate approach is the increased computation time required).

Here are the results for our candidate model set:



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(g+t)p(.)} - ind cov}	21186.9257	0.0000	0.98007	1.0000	8	21170.9130
{phi(g*t)p(.)} - ind cov}	21194.7161	7.7904	0.01993	0.0203	16	21162.6680
{phi(g)p(.)} - ind cov}	21256.8935	69.9678	0.00000	0.0000	4	21248.8900

If you compared these results with those shown on the preceding page (generated using group attributes rather than individual covariates), you'll see they are identical (again, the *differences* among the model deviances are identical, even if the individual model deviances are not). Again, using individual covariates in this case seems like a reasonable 'time-savings' strategy, since the number of PIMs, and the complexity of the general design matrix, is considerably reduced relative to what you'd face if you worked directly with attribute groups in the 'standard' way.

However, as noted in our discussion of the preceding dipper analysis, there are other potential 'costs' which might temper your enthusiasm for using the individual covariate approach to coding 'attribute groups'. First, you'll need to handle reconstituting parameter estimates from what might potentially be pretty sizeable linear model (for our present example, it's sufficiently sizeable – 15 terms – that we won't write it out in full here). Second, *you* (instead of **MARK**) would have to handle the accompanying calculation of SE of the reconstituted estimates (using the Delta method – Appendix B).

However, while this is possible (albeit somewhat time consuming), what is not possible is the

derivation of the SE for the *effect size* (see Chapter 6 – section 6.12) for the difference between levels of a discrete 'attribute variable' when you've coded the 'attribute variable' using an individual covariate (e.g., '`sex`' – see section 11.7; the dipper example in subsection 11.7.1). Calculation of the SE for the 'effect size' (i.e., the difference between the estimates for different levels of the 'attribute variable') requires an estimate of the variance-covariance matrix between estimates for the different attribute levels, which is not estimable when using the individual covariate approach. Finally, generating model averaged parameter estimates from models with individual covariates is decidedly more complicated (as discussed in the next section) than for models without individual covariates.

So, while there is a clear 'up-front savings' in terms of simpler PIMs, and simpler design matrices, when using the individual covariate approach to handling attribute groups, the 'after-the-fact cost' of the number of things you'll need to do by hand (or, more typically, program into some spreadsheet) to generate parameter estimates is not insubstantial, and may be more than the hassle of dealing with lots of PIMs and big, ugly design matrices. An alternative to using individual covariates to simplify model-building is to use the **RMark** package (see Appendix C).

11.8. Model averaging and individual covariates

In chapter 4 we introduced the important topic of model averaging. If you don't remember the details, or the motivation, it might be a good idea to re-read the relevant sections. In a nutshell, the idea behind model averaging is pretty simple: there is uncertainty in our model set as to which model is 'closest to truth'. We quantify this uncertainty by means of normalized AIC_c weights – the greater the model weight, the more support in the data for a given model in that particular model set. Thus, it seems reasonable that any average parameter value must take this uncertainty into account. We do this by (in effect) weighting the estimates over all models by the corresponding model weights (strictly analogous to a weighted average that you're used to from elementary statistics).

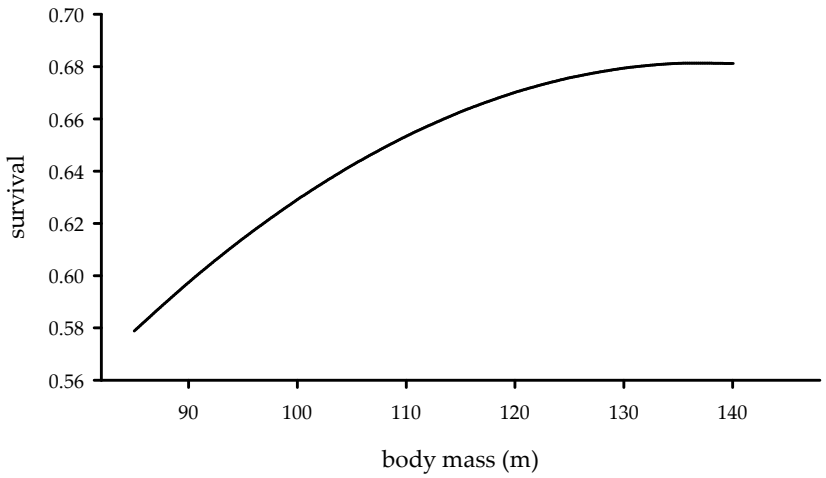
For models with individual covariates, you might guess that the situation is a bit more complex. The model averaging provides average parameter values over the models, but what you're often (perhaps generally) most interested in with individual covariates is the 'average survival probability for an organism with a value of individual covariate XYZ'. For example, suppose you've done an analysis of the relationship of body mass to survival, using individual body mass as a covariate in your analysis. Some of your models may have body mass (`mass`) included, some may have `mass`, and $\texttt{mass}^2$ (as in the first example in this chapter). What would report as the 'average survival probability for an individual with body mass *X*'?

Mechanically, what you would need to do, if doing it by hand, is take the reconstituted values of φ for each model, for a given value of the covariate, then average them using the AIC_c weights as weighting factors (for models without the covariate, the β for the covariate is, in fact, 0). This is fairly easy to do, but a bit cumbersome by hand. Moreover, you have the problem of calculating the standard errors.

Fortunately, **MARK** has a couple of options to let you handle this 'drudgery' automatically. Basically, you can either (i) specify ('define') the value of the individual covariate, and model average for that value or (ii) you can calculate (and plot) the value of the model-averaged parameter over a range of covariate values, using the individual covariate plot capability.

Consider the following example – here we've simulated a new live encounter data set (**`indcov1_avg.inp`**, 8 occasions), where survival (φ) is a function of body mass, m (over the range 85-140 mass units). The form of the relationship used in simulating the encounter data is shown in the figure at the top of the next page.

We see that the relationship between survival and body mass is non-linear – there is a tendency for survival to increase with mass, but at higher mass values the rate of change asymptotes. The data



were simulated assuming no annual variation in the relationship between survival and mass, and no temporal variation in the encounter probability.

We will start by building a candidate model set consisting of 3 models: $\{\varphi_{.}\ p_{.}\}$, $\{\varphi_{m}\ p_{.}\}$, and $\{\varphi_{m+m^2}\ p_{.}\}$. What is important to note about this model set is that we have 2 models which we anticipate will get some significant support in the data (models $\{\varphi_{m}\ p_{.}\}$, and $\{\varphi_{m+m^2}\ p_{.}\}$). We also have a model, $\{\varphi_{.}\ p_{.}\}$, which is notable because it does not contain the covariate. As we will discuss, this is an important consideration – how does 'model averaging' account for models without the individual covariate?

If we fit these 3 models to the data,

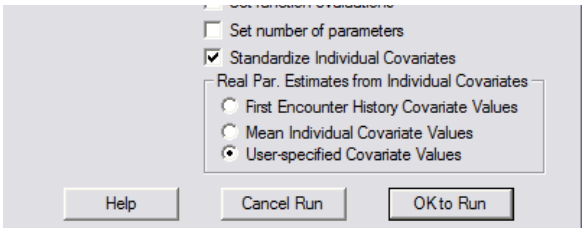
Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{phi(mass)p(.)}	11892.9138	0.0000	0.72208	1.0000	3	11886.9100	11886.9100
{phi(mass+mass^2)p(.)}	11894.8244	1.9106	0.27778	0.3847	4	11886.8180	11886.8180
{phi(.)p(.)}	11910.1349	17.2211	0.00013	0.0002	2	11906.1330	11906.1330

we see that there is relatively strong support for the model where survival is a linear function of mass, $\{\varphi_{m}\ p_{.}\}$, but there is non-negligible support for the non-linear model, $\{\varphi_{m+m^2}\ p_{.}\}$.

Now, we might for some purposes want to know what the model-averaged survival probability is for a particular mass – say, some value near the extremes of the range (a very light or very heavy individual), or perhaps the mean value. **MARK** makes it **very** easy to do this. Simply build the models, each time specifying whether you want **MARK** to provide real parameter estimates from either the first encounter record, a user-defined set of values, or the mean of the covariates.

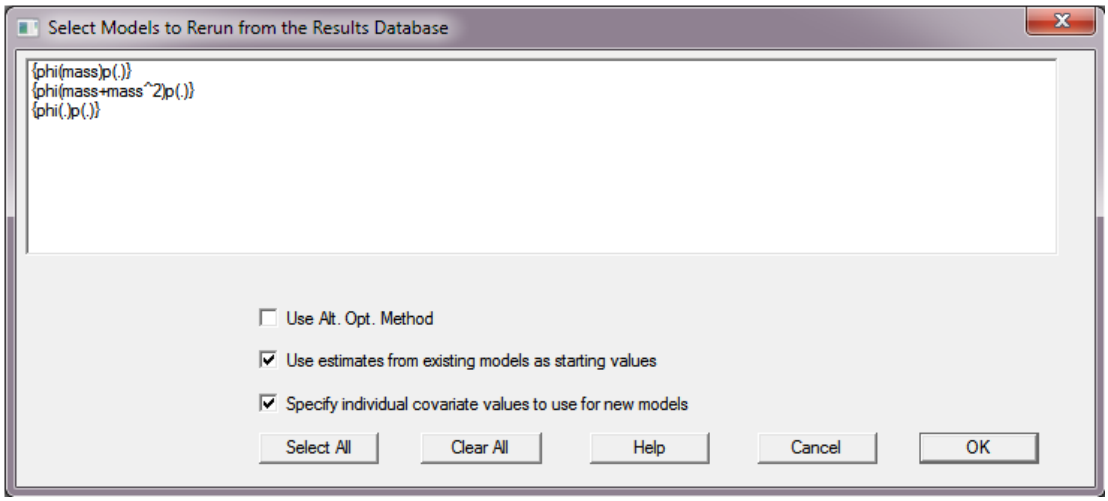
For purposes of demonstration, we'll use a user-defined covariate value (which allows us to generate a model-averaged estimate of survival for a covariate value we specify). Now, if you know you want to do this before you run your models, then fine. Simply select the model you want to re-run, and then in the '**`Setup Numerical Estimation Run`**' window, simply check the '**`user-specific covariate values`**' option box in the lower right-hand corner, as shown at the top of the next page.



If you've checked the '**user-specified covariate values**' radio button, once you click the '**OK to run**' button you'll then be presented with another small window asking you to enter the values of the covariate(s) you want to generate real parameter estimates for.

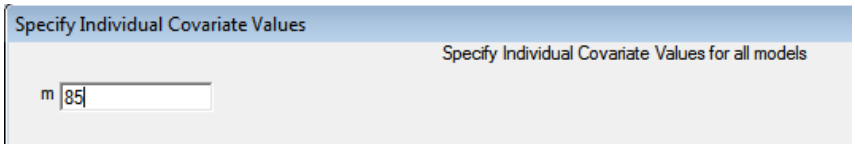
But quite often, you may run your models using the default covariate value (the mean), and then 'after the fact', decide you want to re-run the model, this time using a user-define covariate value. In fact, **MARK** makes this quite easy do. Simply select '**Run | Re-run models(s)**' from the main menu.

This will bring up the dialog window shown below:



All of the models currently in the browser are shown in the main part of this window. You select the models you want to re-run (typically, '**Select all**').

Then, to specify individual covariate values to use for re-running the models, simply check that box, as shown on the preceding page. When you click '**OK**', another window will pop up, asking you to enter the value of the covariate you want to use – say, 85 for mass (`m`):



Now, all that remains is to run the model averaging routine. For this example, using `m=85` as the value of the covariate, model averaged survival value is

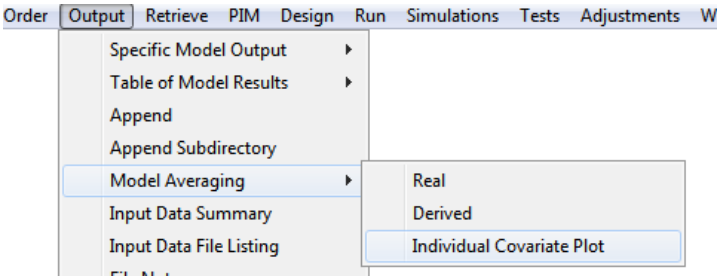
```
            Apparent Survival Parameter (Phi) Group 1 Parameter 1
Model                                    Weight    Estimate      Standard Error
---------------------------------------- -------   -------------- --------------
{phi(mass)p(.)}                         0.67479   0.5832772      0.0175797
{phi(mass+mass^2)p(.)}                  0.32509   0.5654348      0.0300501
{phi(.)p(.)}                            0.00012   0.6524177      0.0064614
---------------------------------------- -------   -------------- --------------
Weighted Average                                  0.5774853      0.0216323
Unconditional SE                                                 0.0239299
95% CI for Wgt. Ave. Est. (logit trans.) is 0.5300219 to 0.6235598
```

One conceptual issue to consider – body mass (`m`) was contained in 2 of 3 models in our candidate model set. What about the third model, $\{\varphi_{.}\ p_{.}\}$ which does not contain body mass? Well, clearly, if the covariate for a particular covariate does not show up in a model, then the β estimate for that covariate is 0, for that model. But, our interest is (typically) in model averaging real parameter estimates, not β estimates.

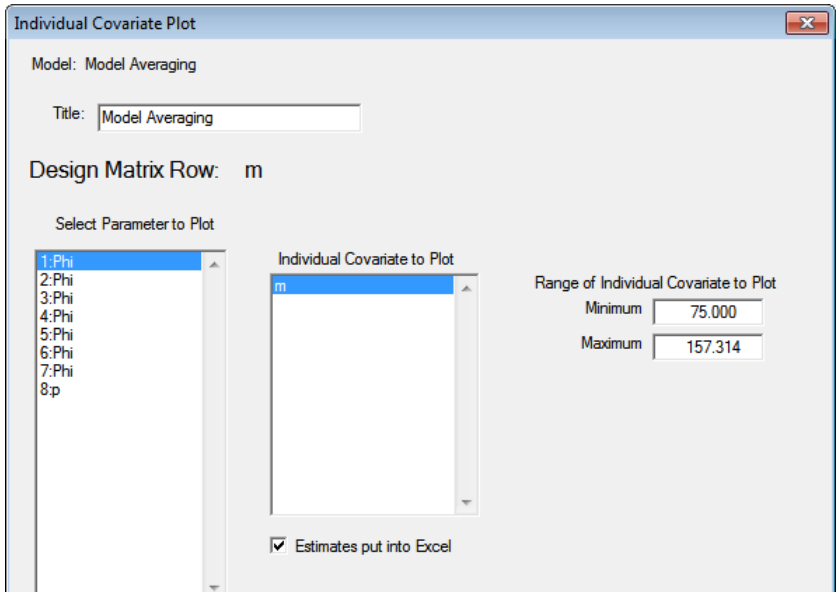
So how does **MARK** average real estimates over models including those that do not include the covariate? You can get a partial clue by looking back at the table of estimates used in the model averaging (above). Note that the reported estimate for survival for model $\{\varphi_{.}\ p_{.}\}$ is 0.6524177.

Where does this value come from? Simple – it is the estimate of survival you would get if you ignore the mass covariate (which is implicit in the model, which does not include mass), which in effects is equivalent to assuming that all individuals in the sample have the same mass – i.e., the average mass for the sample. You can confirm this for yourself by re-running all the models, and changing the user-specified model for mass. If you do this, you will see that the reported estimates of survival for models $\{\varphi_{.}\ p_{.}\}$ and $\{\varphi_{m}\ p_{.}\}$ will change, since they both include mass as a term in the model. However, the reported value for model $\{\varphi_{.}\ p_{.}\}$ will not change.

While calculating model averaged survival for specific, user-defined values of the covariate (as above) is straightforward, we're often most interested in evaluating (and visualizing) the model averaged parameter (in this example, survival) over a range of the covariate (mass). This is quite easy to do in **MARK**. Simply select '**`Output | Model Averaging | Individual Covariate Plot`**':

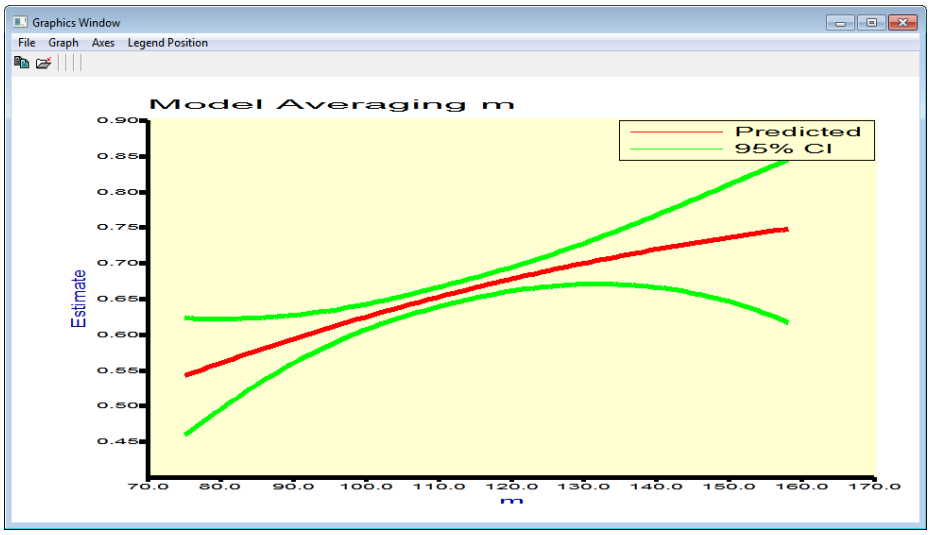


A dialog window nearly identical to the single model plot we considered earlier (section 11.5) is then opened (top of the next page), and you select the real parameter you want to plot.



However, the design matrix entry now shows the names of the individual covariates available to be plotted, because not all models in the results browser would normally use the same functional relationship between the real parameter and the individual covariate that is to be plotted. For example, some models with AIC_c weight in the results browser might not have any relationship between the covariate and the real parameter to be plotted, meaning a flat line results for this model. As with the single model plot, you select from the second list box the individual covariate to be plotted, and the range over which to plot the function. If there were other covariates included in one or more of the models in the model set, all of these other individual covariates are listed with the values used when they are included in the model for the real parameter being plotted.

For our present example, the plotted model averaged values (below) don't indicate much evidence for any non-linearity in the relationship between survival and mass (in other words, this figure doesn't look very similar to the true generating function used to generate the data used in this analysis – p. 45).



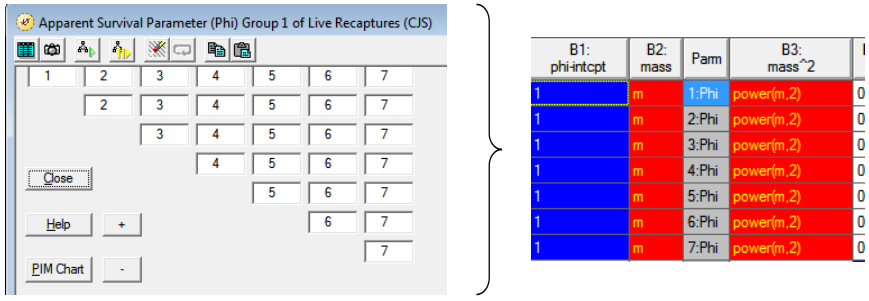
However, this plot of model averaged values is entirely consistent with the previous observation that the non-linear quadratic model in the candidate model set, $\{\varphi_{m+m^2}\, p_.\}$, did not receive appreciable support in the data. In fact, the linear model, $\{\varphi_m\, p_.\}$, had 2.6 times the support in the data as the quadratic model – and this much stronger support for the linear model is reflected in the model averaged estimates.

11.8.1. Careful! – traps to avoid when model averaging

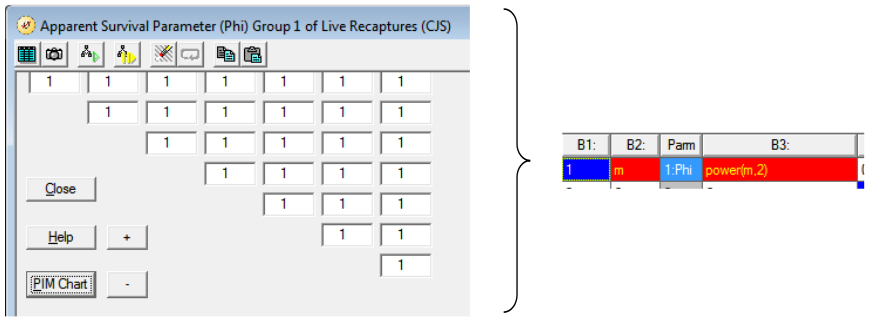
In the process of building some of your candidate models, you may have changed the definition of some of the PIMs with the '**Change PIM Definition**' menu choice. For example, consider a multi-state model (Chapter 10) – if the first transition probability from strata A is defined in some models as $\psi^{A\rightarrow A}$, and in other models as $\psi^{A\rightarrow B}$, and these real parameters are model averaged, the results may be incorrect. Thus, be sure to check the model averaging results to verify that correct parameters were selected.

Another potential 'gotcha' might arise if you want to use the '**individual covariate plot**' for modeling averaging, and if you've used different PIM structures for some of your models in your candidate model set (rather than using the same PIM structure for all your models, using the design matrix to construct reduced parameter models based on that PIM structure). For example, consider the example presented at the start of this section, based on the simulated data in **indcov_avg1.inp**. Recall that for these data, we fit the following 3 candidate models: $\{\varphi_.\, p_.\}$, $\{\varphi_m\, p_.\}$, and $\{\varphi_{m+m^2}\, p_.\}$.

However, what we didn't discuss when we initially analyzed these data is what the underlying PIM structure was. We noted that we assumed no temporal variation in φ or p. As such we could have used either of the following PIMs and corresponding DM for (say) model $\{\varphi_{m+m^2}\, p_.\}$:



which is entirely equivalent (in terms of fit to the data, and parameter estimation) to



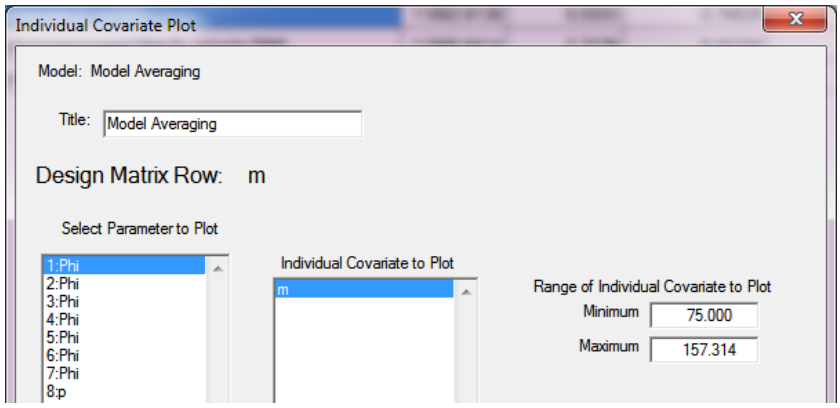
To illustrate the point, we'll refer to the first approach as being based on '`t-PIM`' (say, for 'time-based PIM'), and the second approach being based on '`simple PIM`' (no time-dependence in the PIM). We'll use the time-based PIMs for models $\{\varphi_{\cdot}\ p_{\cdot}\}$ and $\{\varphi_{m}\ p_{\cdot}\}$, and the 'simple' PIM for model $\{\varphi_{m+m^2}\ p_{\cdot}\}$.

As you can see from the browser (below), the results of fitting these models to the data are identical to what we saw before, even though we have used a different underlying PIM structure for one of the models:

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(m)p(.)} - t-PIM}	11892.9138	0.0000	0.79626	1.0000	3	11886.9100
{phi(mass+mass^2)p(.)} - simple PIM}	11895.6414	2.7276	0.20359	0.2557	4	11887.6350
{phi(.)p(.)} - t-PIM}	11910.1349	17.2211	0.00015	0.0002	2	11906.1330

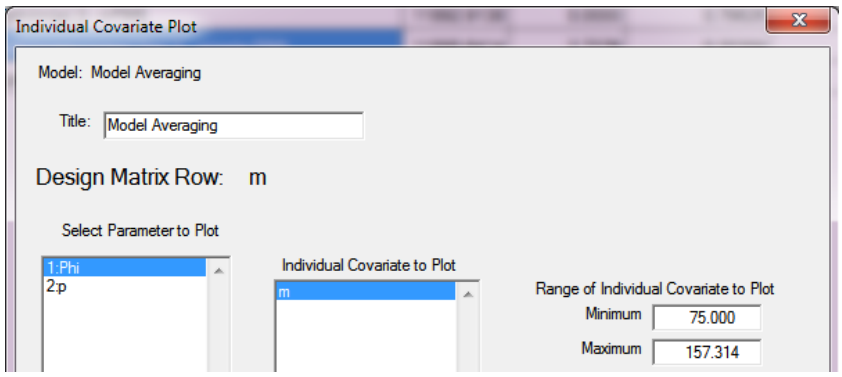
Make model $\{\varphi_{m}\ p_{\cdot}\}$ active, by selecting it in the browser, and retrieving it. Recall that this model was built with the time-based PIM.

Now, select '**`Output | Model averaging | Individual covariate plot`**'. You'll be presented with the individual plot window (below):



You'll see that you have 7 parameters for φ (labeled `1:Phi` → `7:phi`). Now, we 'know' that here, we could select any of the 7 `x:Phi`, because our DM is set up to constrain them to be equivalent.

However, if instead we made model $\{\varphi_{m+m^2}\ p_{\cdot}\}$ active, then we see the following when we select '**`Output | Model averaging | Individual covariate plot`**':



Now, we see only 1 parameter for φ, not 7, as above. Why? because we constructed model $\{\varphi_{m+m^2}\, p_.\}$ using a 'simple' PIM structure for the underlying model.

Now, in this particular case, you'll end up with the same model averaged estimates regardless of which model was 'active', but that may not always be the case (especially for complicated models where the functional relationship between the covariate(s) and the parameter vary over time). So, the general recommendation is to use a common PIM structure over all your models, and if you do want/need to use a different PIM structure for some models in your model set, be careful when model averaging.

A final trap concerns individual covariates in particular. The user can specify the values of individual covariates to be used to compute the real and derived parameter values. If different values of the individual covariate are specified for different models to be model averaged, the results will be nonsense.

Thus, be sure to use the same individual covariate values in all models to be model averaged, e.g., the mean value. The real and derived estimates can be changed to use a different individual covariate value with the '**ReGenerate Real Derived Model(s)**' option in the results browser '**Run**' menu.

11.8.2. Model averaging and environmental covariates

In chapter 6 (section 6.16), we considered model averaging across models where survival or some other parameter was constrained to be a function of one or more 'environmental covariates'. Our interest is in coming up with a way to estimate the relationship between the parameter and the covariate (similar to what was presented in the -sidebar- starting on p. 28 of this chapter), but averaged over multiple models.

As in Chapter 6, let's consider, again, the full Dipper data set, where we hypothesize that the encounter probability, p, might differ as a function of (i) the sex of the individual, (ii) the number of hours of observation by investigators in the field, with (iii) the relationship between encounter probability and hours of observation potentially differing between males and females.

Recall that our 'fake' observation hour covariates were:

```
Occasion  2     3     4     5     6     7
hours     12.1  6.03  9.1   14.7  18.02 12.12
```

Now, when we introduced this example earlier in this chapter, we fit only a single model to the data:

$$\text{logit}(p) = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{HOURS}) + \beta_4(\texttt{SEX.HOURS})$$

But, here, we acknowledge uncertainty in our candidate models, and will fit the following candidate model set to our data:

$$\begin{aligned}
&\text{model } M_1 \quad \text{logit}(p) = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{HOURS}) + \beta_4(\texttt{SEX.HOURS}),\\
&\text{model } M_2 \quad \text{logit}(p) = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{HOURS}),\\
&\text{model } M_3 \quad \text{logit}(p) = \beta_1 + \beta_2(\texttt{HOURS}),\\
&\text{model } M_4 \quad \text{logit}(p) = \beta_1 + \beta_2(\texttt{SEX}).
\end{aligned}$$

There are a couple of things to note. First, this is not intended to be an 'exhaustive, well-thought-out' candidate model set for these data. We're using these models to introduce some of the considerations for model averaging. In particular, we're using this example where encounter probability is hypothesized to be a function of a continuous environmental covariate, to force us to consider how – and what – we model average when some models include the environmental covariate (HOURS), and some don't.

Let's fit these 4 candidate models ($M_1 \rightarrow M_4$) to the full Dipper data set, treating sex as a categorical, group attribute variable. We'll build all of the models using a design matrix approach, using the encounter data in **ed.inp**. Note that models $M_2 \rightarrow M_4$ in the model set are all nested within the first model, M_1. For all 4 models, we'll assume that apparent survival, φ, varies over time, but not between males and females.

The results of fitting our 4 candidate models to the full Dipper data are shown below:

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(t)p(SEX)}	675.5036	0.0000	0.43880	1.0000	8	76.6812
{phi(t)p(HOUR)}	676.0272	0.5236	0.33773	0.7697	8	77.2047
{phi(t)p(SEX+HOUR)}	677.5253	2.0217	0.15968	0.3639	9	76.6155
{phi(t)p(SEX+HOUR+SEX*HOUR)}	679.3608	3.8572	0.06378	0.1453	10	76.3535

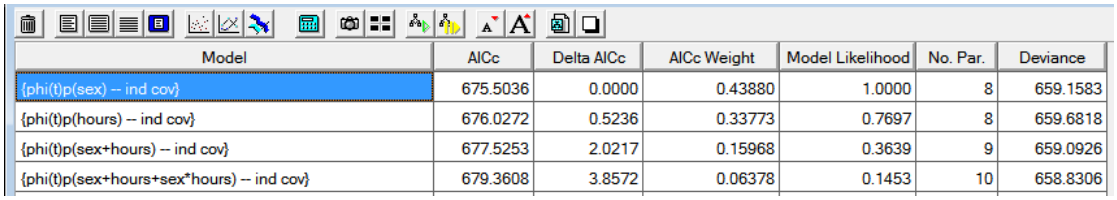
We see from the AIC_c weights that there is considerable model selection uncertainty. In fact, the ΔAIC_c values among all models is < 4.

Now, we want to fit the same candidate model set, but coding both `SEX` and `HOURS` as individual covariates. Recall from p. 28 that we code each occasions covariate value as an individual covariate. This requires reformatting the `.INP` data. Here are the top few lines of the reformatted `.INP` file (which we'll call **ed_cov.inp**):

```
      |...+....1....+....2....+....3....+....4....+....5
===== 1111110 1 1 12.1 6.03 9.1 14.7 18.02 12.12 ;
===== 1111000 1 1 12.1 6.03 9.1 14.7 18.02 12.12 ;
===== 1100000 1 1 12.1 6.03 9.1 14.7 18.02 12.12 ;
===== 1100000 1 1 12.1 6.03 9.1 14.7 18.02 12.12 ;
===== 1100000 1 1 12.1 6.03 9.1 14.7 18.02 12.12 ;
===== 1100000 1 1 12.1 6.03 9.1 14.7 18.02 12.12 ;
===== 1010000 1 1 12.1 6.03 9.1 14.7 18.02 12.12 ;
```

The first 7 columns comprise the encounter history for the individual. Column 9 is the frequency (1) for that individual. Column 11 is the coding for `SEX`, as an individual covariate (`SEX=1`, male, `SEX=0`, female), and columns 13 $\rightarrow$ 42 list the environmental covariates (`HOURS`), coded as occasion-specific individual covariates.

Now, that we've re-formatted our `.INP` file, let's fit the same 4 candidate models. We'll refer to the sex covariate as `sex`, and the environmental covariates as `h1`, `h2`, `h3`, `h4`, `h5`, and `h6`, corresponding to `HOURS` for each encounter occasion:



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(t)p(sex) -- ind cov}	675.5036	0.0000	0.43880	1.0000	8	659.1583
{phi(t)p(hours) -- ind cov}	676.0272	0.5236	0.33773	0.7697	8	659.6818
{phi(t)p(sex+hours) -- ind cov}	677.5253	2.0217	0.15968	0.3639	9	659.0926
{phi(t)p(sex+hours+sex*hours) -- ind cov}	679.3608	3.8572	0.06378	0.1453	10	658.8306

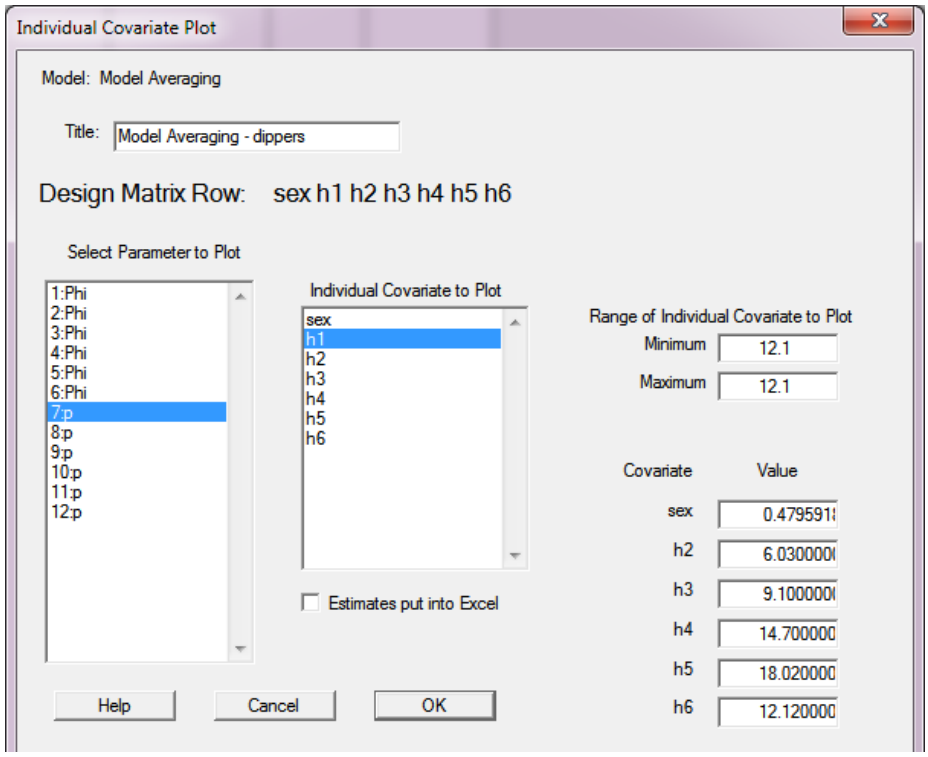
Compare these results with those shown in the browser at the top of this page. Note that the reported deviances are quite different – because the underlying likelihood structures differ, depending on whether or not you use individual covariates. However, even though the deviances differ, the relative AIC_c differences, and so on, are identical. And, if we looked at the reconstituted parameter estimates, we'd see they were also identical.

OK, so we've just confirmed that our 4 candidate models built using the individual covariates approach are 'correct', in that they match the models we built earlier, based on treating `sex` as a group attribute variable, and entering the covariate values into the DM.

Now what? Well, now we can use the model averaging (and plotting capabilities) for individual covariates in **MARK**, to generate model averaged estimates of the relationship between the parameter (in this case, encounter probability, p), and the environmental covariate, `HOURS`.

In Chapter 6, we focussed on averaging over models for `SEX=1` (males). Let's try the same thing here. Simply select '**`Output | Model Averaging | Individual Covariate Plot`**'. This will bring up the model averaging window we've seen earlier in this chapter.

Have a look what happens if we click the first encounter probability (`7:p`) and the first `HOURS` covariate (`h1`):

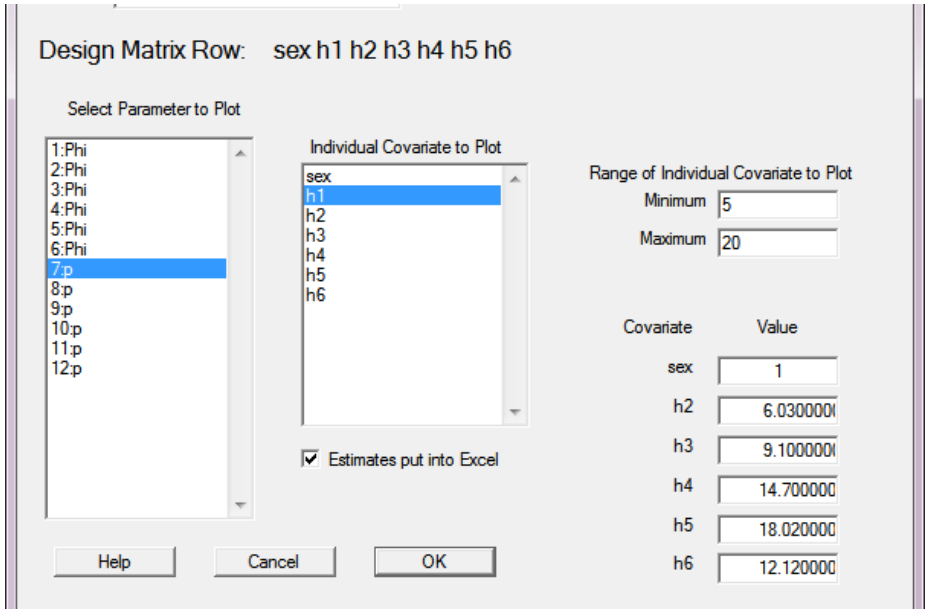


On the right-hand side, we see the range of the individual covariate we want to plot (`h1`, corresponding to encounter probability for sampling occasion 2, although it is not labeled as such). We'll change this range in a moment.

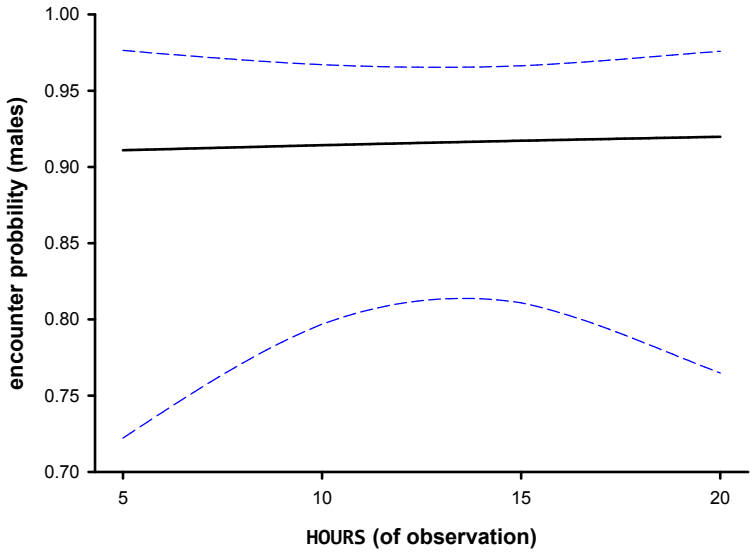
Below this are the other values of the covariates which will be 'fixed' during the averaging and plotting. Note that the `SEX` covariate is reported as 0.4795911. Where does this number come from? Remember, we coded males using `SEX` = 1, and females as `SEX` = 0. If we had an equal number of males and females in our sample, then the average coding for `SEX` would be 0.5. However, in our sample, we have slightly more females than males, and the average for `SEX` is 0.4795911 (which, in fact, is the sex-ratio for our sample).

Below the `SEX` covariate value are the values of the environmental covariate `HOURS` for each encounter occasions (`h2` = 6.03 for occasion 3, `h3` = 9.10 for occasion 4, and so on...).

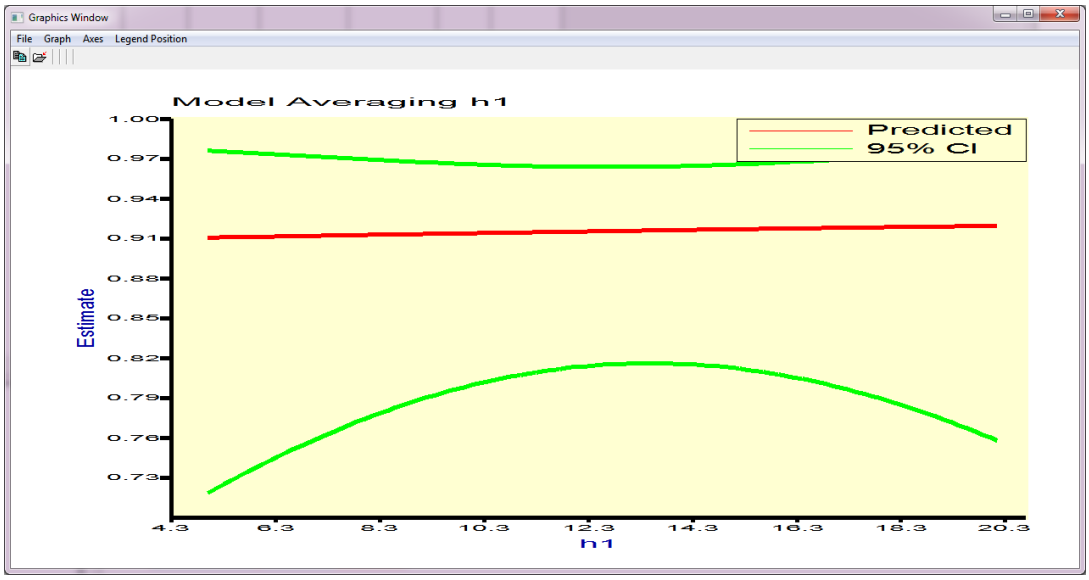
To generate the plot we're after, we'll need to modify a few things (below). First, since we are focussing on males (SEX = 1), we'll change the value of the SEX covariate to 1. In addition, we'll change the range of the individual covariate h1 we want to average over, and plot, from 12.1 → 12.1 to (say), 5 → 20. Remember, it doesn't matter which covariate you plot ($p_1, p_2, \ldots$), so long as you select the correct environmental covariate for that occasion (i.e., 7:p with h1, 8:p with h2, and so on...).



Back in Chapter 6 (section 6.16), we hand-calculated model averaged estimates for male encounter probability as function of HOURS of observation, and their associated confidence intervals, which when plotted, looked like the following:



How do the results from our 'averaging over individual covariates' compare? In fact, they are essentially identical.* Here is the plot generated by **MARK**, which is a near-perfect match to the hand-generated plot shown at the bottom of the previous page:



If you look back at section 6.16 in Chapter 6, you'll see that doing the calculation(s) 'by hand' was a lot of work. Using the individual covariate model averaging capabilities in **MARK**, demonstrated in this section, is much faster, and likely far less error-prone. The only real 'trade-off' is that to use the approach based on individual covariates, you need to re-format your `.INP` file such that everything in your analysis is coded using individual covariates (all attribute grouping variables, all environmental covariates, everything...). Depending on the scope of your data set, and the models you're fitting to those data, this can also require a fair bit of work.

11.9. GOF Testing and individual covariates

Well, now that we've seen how easy it is to handle individual covariates, now for the good news/bad news part of the chapter. The good news is that individual covariates offer significant potential for explaining some of the differences among individuals, which, as we know (see Chapter 5), is one potential source of lack of fit of the model to the data.

OK – now the bad news. At the moment, we don't have a good method for testing fit of models with individual covariates. If you try to run one of the GOF tests based on simulation or resampling – say, the median-$\hat{c}$ – you'll be presented with a pop-up warning that '*the median c-hat only works for models without individual covariates*'. The Fletcher-$\hat{c}$ isn't even printed in the full output. And so on.

For the moment, the recommended approach is to perform GOF testing on the most general model

* As discussed in Chapter 6, the back-transform of the model averaged value of logit($\hat{p}$) is not the same as the model averaged value of the back-transforms of the individual estimates of $\hat{p}$ from each model. This difference reflects Jensen's inequality. In Chapter 6, the reported and plotted model averaged estimates for the encounter probability, and associated 95% CI, were based on the model averaged value of logit($\hat{p}$), while the values **MARK** uses for the individual covariate model averaging are based on the model averaged value of the back-transforms of the individual estimates of $\hat{p}$ from each model. The difference between the two is generally very small.

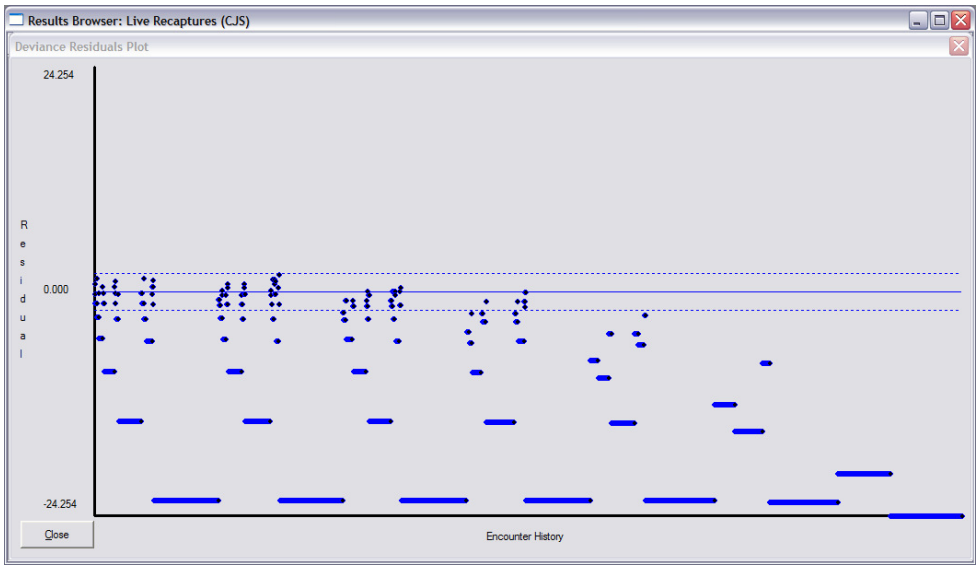
that does not include the individual covariates, and use the $\hat{c}$ value for this general model on all of the other models, even those including individual covariates. If individual covariates will serve to reduce (or at least explain) some of the variation, then this would imply that the $\hat{c}$ from the general model without the covariates is likely to be too high, and thus, the analysis using this $\hat{c}$ will be 'somewhat conservative'. So, keep this in mind...

begin sidebar

individual covariates and deviance plots

One approach to assessing the fit of a model to a particular set of data is to consider the *deviance residual plots*. While this can prove useful – in particular, to assess lack of fit because the structure of the model is not appropriate given the data (e.g., TSM models – see Chapter 7), if you try this approach for models with individual covariates, you'll quickly run into a problem.

For example, consider the deviance residual plot for the first example analysis presented in this chapter (for model $\{\varphi_{.}\, p_{.}\}$).



Clearly, something 'strange' is going on – we see fairly discrete 'clusters' of residuals, virtually all below the 0.000 line. Obviously, this is quite different than any other residual plot we've seen so far.

Why the difference? In simple terms, the reason that the residual plots change so much when an individual covariate is added is because the number of animals in each observation changes. Without individual covariates, the data are *summarized* for each unique capture history, so that variation within a history due to the individual covariate is lost. However, when the covariate is added into the model, each animal (i.e., each encounter history, even if it is the same as another history) is plotted as a separate point. The result is quite different, obviously. Without individual covariates, the binomial functions are the sample size, so animals are 'pooled'. With individual covariates, the number of animals is the sample size, each resulting in a unique residual.

In other words, the deviance residual plots for models with individual covariates are not generally interpretable.

end sidebar

11.10. Summary

That's it for Chapter 11! In this chapter, we looked at the basic mechanics of using **MARK** to fit models where one or more parameters are constrained to be functions of individual covariates. Individual covariates can be used with **any** of the models in **MARK** (not just recapture models). This is a significant increase in the flexibility of analyses you can execute with **MARK**.

11.11. References

Bonner, S. J. (2013) Implementing the trinomial mark-recapture-recovery model in Program MARK. *Methods in Ecology and Evolution*, **4**, 95-98.

Burnham, K. P., and Anderson, D. R. (2004) Multimodel inference – understanding AIC and BIC in model selection. *Sociological Methods & Research*, **33**, 261-304.

Catchpole, E. A., Morgan, B. J. T., and Tavecchia, G. (2008) A new method for analysing discrete life history data with missing covariate values. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **70**, 445-460.

Link, W. A., and Barker, R. J. (2006) Model weights and the foundations of multimodel inference. *Ecology*, **87**, 2626-2635.