CHAPTER 6

Adding constraints: MARK and linear models

Up until now, we've used **MARK** to build relatively simple models. We've seen how to use **MARK** to help with model selection, and how the process of model selection can be viewed in an analysis of variance context, by comparing models with and without grouping factors (Chapter 4).

However, suppose, for example, you want to build a model where annual variation in survival is 'related' to some weather variable. How would you construct a model where survival was constrained to be a function of 'weather'? The concept of 'constraints', and how to use them to apply linear models to **MARK**, is one of the most important and powerful extensions of what we have covered so far.

What do we mean by 'constraint'? Here, we are referring to a mathematical constraint – 'forcing' **MARK** to estimate survival and encounter probabilities after imposing one or more linear constraints on the structure of the underlying model. While the concept is simple, the ability to construct linear models gives you considerable flexibility in addressing a very large number of questions and hypotheses with your data; if you can conceive of a linear model (ANOVA, ANCOVA, multiple regression, etc.), you can apply it to mark-encounter data using **MARK**. The only thing you'll need to know is how to construct the 'linear constraint' – the *linear model*. This is the subject of the present chapter.

6.1. A (brief) review of linear models

If you have a prior background in linear models, then much of the following material will be familiar. Our purpose is to provide a 'minimum-sufficient level' of background. If you are new to linear models, we urge you supplement your reading of this chapter by having a look at one of the many good textbooks on this subject. McCullagh & Nelder (1989) and Dobson & Barnett (2008) are particularly good.

The basic idea underlying linear models can be stated quite simply: the response variable in many statistical analyses can be expressed as a linear regression function of 1 or more other factors. In fact, any ANOVA-type design can be analyzed using linear regression models (although interpretation of interactions is sometimes complex). In general, for data collected from marked individuals, the 'response variable' is often a probability or proportion (e.g., survival or encounter probability), which must be transformed prior to analysis using a linear models approach (we'll get to that in a moment). For the moment, assume the response variable has been suitably transformed.

We begin by demonstrating this relationship between 'regression' and 'ANOVA' by means of a simple example. Consider data from a study of a physically small tropical species of bat, where we're interested in knowing if the wing length is on average larger in males than in females (we'll assume for the moment that all of the individuals in the sample were the same chronological age). Let's suppose we measure 7 male and 7 female bats, and analyze our data using a normal single-classification ANOVA.

Here are the wing length data (in inches):

```
  male  7.2  7.1  9.1  7.2  7.3  7.2  7.5
female  9.8  8.5  8.7  8.6  8.4  7.7  8.2
```

First, the results from a 'standard ANOVA' (as you might generate using some statistical analysis software), which indicate a nominally significant difference in wing length between males and females:

```
Source   df   SS     MS     F     P
SEX      1    3.806  3.806  8.33  0.0137
Error    12   5.485  0.457
Total    13   9.292
```

However, what if our statistics package was limited only to a regression subroutine? Could we have analyzed our data using a linear regression model, instead of ANOVA, and arrived at the same result?

The answer is, indeed, 'yes, we can'. What we do is simply take the classification factor (`SEX`) and 'code' it as a '`0`' or '`1`' indicator ('dummy') variable (we'll see why in just a moment). For example, let '`0`' represent females, and '`1`' represent males. Thus, every individual in our data set is assigned a '`0`' or a '`1`', depending upon their gender. Let's call this dummy variable '`SEX`'. Now, all we need to do is regress our response variable (wing length) on the dummy variable for `SEX`.

Here are the results of such a regression analysis:

```
Source   df   SS     MS     F     P
SEX      1    3.806  3.806  8.33  0.0137
Error    12   5.485  0.457
Total    13   9.292
```

No, it's not a typo – they are in fact the exact same results as shown previously for the classical ANOVA. The two approaches are equivalent, yielding identical results. How can this be? The answer lies in the structure of the models actually being tested. Let's look at things a bit more formally.

In general, a linear model can be expressed in matrix (or, matrix-vector) form as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\mathbf{y}$ is a vector of responses (i.e., a vector of the response variables), $\boldsymbol{\beta}$ is a vector of parameters (e.g., the intercept and 1 or more 'slopes'), $\mathbf{X}$ is a matrix with either '`0`' or '`1`' elements, or values of 'independent' variables, and $\boldsymbol{\epsilon}$ is a vector of random error terms.

In cases of analysis of variation of the response variable among different levels of one or more classification (i.e., 'treatment' or 'factor') levels, there is a parameter β in the vector $\boldsymbol{\beta}$ to represent each level of a factor. The elements of $\mathbf{X}$ (which is generally referred to as the *design matrix* – discussed below) are chosen to exclude or include the appropriate parameters for each observation. These elements are often referred to as either 'dummy' or 'indicator' variables ('indicator' generally being used when only '`1`' or '`0`' are used as the coding variables).

The following simple example will make this clear, and will illustrate the underlying connection between a linear regression model and analysis of variation (ANOVA). Suppose you have collected data on the scutum width of male and female individuals of some insect species. You are interested in whether a difference in mean scutum width between the sexes is larger than would be expected by random chance. Typically, you might use a single-classification (Model I) ANOVA for this sort of analysis.

Recall that for this sort of analysis, any single variate Y (in this case, $Y = f$[wing length]), can be decomposed as:

$$Y_{ij} = \mu + \alpha_i + \epsilon_{ij}.$$

In other words, each individual variate Y_{ij} is the sum of the global mean (μ), the deviation of the individual from that mean due to the 'classification' factor (sex; α_i), and the random error term (ϵ_{ij}) In this example, with 2 levels of the classification factor (i.e., males and females), we would be testing for differences of the type $(\alpha_1 - \alpha_2)$. If $(\alpha_1 - \alpha_2) = 0$ (the null hypothesis), then we would conclude no significant group effect (i.e., no significant difference in group means between the sexes).

How could we use linear regression to approach the same analysis? In a regression analysis, each individual variate Y_i would be decomposed as:

$$Y_i = \beta_1 + \beta_2 x_i + \epsilon_i.$$

In this case, each variate Y_i is the sum of the product of the slope (β_2) and the variable x, the intercept (β_1), and a random error term (ϵ_i). In this case, the hypothesis being tested is whether or not the estimate of the slope is significantly different from 0 ($\mathbf{H_o}$: $\beta_2 = 0$).

However, what is the variable 'x'? In fact, this is the key to understanding the connection between the regression model and the ANOVA analysis. In the regression formulation, x represents a coding ('dummy') variable specifying male or female (i.e., sex, the classification variable in the ANOVA analysis). The coding variable takes on the value of '0' or '1' (you might use '0' for females, '1' for males). We regress the response variable Y (scutum width) on the coding variable for sex. If the slope (β_2) is not different from 0, then we interpret this as evidence that the numerical value of the coding variable does not significantly influence variation in our data. Put another way, if the slope does not differ from 0, then this indicates no significant difference between the sexes. This is entirely analogous to test of the $(\alpha_1 - \alpha_2)$ hypothesis in the ANOVA analysis.

Recall that we can express a linear model in vector-matrix form as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\mathbf{y}$ is a vector of responses (i.e., a vector of the response variables), $\boldsymbol{\beta}$ is a vector of parameters (e.g., the intercept and 1 or more 'slopes'), $\mathbf{X}$ is a matrix with either '0' or '1' elements, or values of 'independent' variables, and $\boldsymbol{\epsilon}$ is a vector of random error terms. For our present example, the design matrix $\mathbf{X}$ consists of 2 columns of '0' and '1' dummy variables (the first column corresponding to the intercept, β_1, and the second column corresponding to dummy variable coding for a given sex, β_2).

Given K individuals in each sex (although a balanced design is not required), $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ can be written as

$$\left[\begin{array}{c} Y_{11} \\ Y_{12} \\ \vdots \\ Y_{1K} \\ \hline Y_{21} \\ Y_{22} \\ \vdots \\ Y_{2K} \end{array}\right] = \left[\begin{array}{cc} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ \hline 1 & 1 \\ 1 & 1 \\ \vdots & \vdots \\ 1 & 1 \end{array}\right] \left[\begin{array}{c} \beta_1 \\ \beta_2 \end{array}\right] + \left[\begin{array}{c} \epsilon_{11} \\ \epsilon_{12} \\ \vdots \\ \epsilon_{1K} \\ \hline \epsilon_{21} \\ \epsilon_{22} \\ \vdots \\ \epsilon_{2K} \end{array}\right].$$

In fact, in this case, if we used '1' to code for males, and '0' to code for females, then the intercept (β_1) would represent the estimate for female survival (since if the dummy variable is '0', then all that remains in the model is the intercept, and the random error term). The β_2 term actually reflects (male survival - female survival), such that $\beta_1 + \beta_2 = (\texttt{female}) + (\texttt{male} - \texttt{female}) = \texttt{male}$ survival. The structure of the design matrix is discussed in more detail in the next section.

It is perhaps worth noting that models of the form '$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$' are called linear models because the non-error part of the expression $\mathbf{X}\boldsymbol{\beta}$ is a *linear* combination of the parameters (and not specifically because of the relationship of ANOVA to linear regression). **MARK** uses this general linear models approach as the basis for all of the analysis (data) types available.

begin sidebar

matrix approach to linear regression & ANOVA: simple introduction

Here, we provide a *very* simple example of a matrix approach to linear regression (and, by extension, to linear models in general). For deeper understanding, you should consult one of the several very good 'linear models' textbooks which give *much* fuller treatments of the subject.

Consider the linear model, say of individual (i) with mass (Y_i) relative to sex (X_i, where $X = 0$ or $X = 1$ for female or male, respectively), measured with Gaussian (normally) distributed random variation (ϵ_i) about the mean. We'll assume the following 'fake' data:

	mass (Y)			
male ($X = 1$)	11	12	11	14
female ($X = 0$)	8	11	12	10

The mean mass for males ($\bar{x}_m = 12$) is larger than the mean mass for females ($\bar{x}_f = 10.25$) – the usual question being, is the difference between the two larger than expected due to random chance?

Here we adopt a linear models approach to answering this question. As a first step, we write the relationship between `mass` and `sex` in linear model form as:

$$Y_i = \beta_1 + \beta_2 X_i + \epsilon_i.$$

The null hypothesis of 'no difference between sexes' can be expressed formally in terms of the β term for sex; i.e., $\mathbf{H_0}: \beta_2 = 0$, where X_i is the 'dummy' indicator for sex (male or female). The technical problem then is estimating the β_i coefficients in the linear model. To do this, first define a vector $\mathbf{y}$ for all the Y_i, a matrix $\mathbf{X}$ for a vector of 1s and all the X_i, a vector $\boldsymbol{\epsilon}$ for all the ϵ_i, and further define a vector $\boldsymbol{\beta}$ for the coefficients β_1 and β_2.

Then (for our 'fake' data set) we get

$$\mathbf{Y} = \begin{bmatrix} 11 \\ 12 \\ 11 \\ 14 \\ \hline 8 \\ 11 \\ 12 \\ 10 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ \hline 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{13} \\ \epsilon_{14} \\ \hline \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{23} \\ \epsilon_{24} \end{bmatrix} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

Note that the matrix $\mathbf{X}$ is referred to as the *design matrix*. The construction of the design matrix is fundamental to using linear models in **MARK**, as we will cover in considerable detail later in this chapter. So, to derive estimates of the β_i coefficients, we need to find a vector $\boldsymbol{\beta}$ such that $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$.

Is this possible? The answer is clearly 'no', because that would require the points to lie exactly on a straight line. A more modest (and tractable) question is: can we find a vector $\hat{\boldsymbol{\beta}}$ such that $\mathbf{X}\hat{\boldsymbol{\beta}}$ is in a

sense 'as close to $\mathbf{y}$ as possible'? The answer is 'yes'. The task is to find $\hat{\boldsymbol{\beta}}$ such that the length of the vector $\boldsymbol{\epsilon} = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$ is as small as possible (i.e., $\boldsymbol{\epsilon} \to 0$).

How do we get there from here? Fairly easily. First, we note that what we're trying to do is solve for $\boldsymbol{\beta}$ in the linear model. The first step is to let $\boldsymbol{\epsilon} = 0$ (such that it drops out of the equation – this should make sense, if you keep in mind that what we're trying to do is to find $\hat{\boldsymbol{\beta}}$ such that the length of the vector $\boldsymbol{\epsilon}$ is, in effect, 0). This leaves us with

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}.$$

Then, a few steps of algebra to solve for the vector $\boldsymbol{\beta}$:

$$\begin{aligned}
\mathbf{y} &= \mathbf{X}\boldsymbol{\beta} \\
\mathbf{X}^{\mathsf{T}}\mathbf{y} &= \mathbf{X}^{\mathsf{T}}\mathbf{X}\boldsymbol{\beta} \\
(\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y} &= (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{X}\boldsymbol{\beta} \\
(\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y} &= \boldsymbol{\beta} \\
\hat{\boldsymbol{\beta}} &= (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y}.
\end{aligned}$$

In words, we multiply both sides of the initial equation by the transpose of $\mathbf{X}$ to get the cross-product $\mathbf{X}^{\mathsf{T}}\mathbf{X}$, which is a square matrix (*note*: the square matrix $(\mathbf{X}^{\mathsf{T}}\mathbf{X})$ is called the *pseudo inverse* of $\mathbf{X}$. We cannot use the true matrix inverse of $\mathbf{X}$ (i.e., $\mathbf{X}^{-1}$) because it generally does not exist as $\mathbf{X}$ is not generally a square matrix; $m \neq n$). We then find the inverse of this cross-product matrix and multiply both sides by that. This allows us to cancel out the term involving $\mathbf{X}$ on the right-hand side of the equation, allowing us to find an estimate of $\boldsymbol{\beta}$, which we call $\hat{\boldsymbol{\beta}}$, in terms of the original data.

It is worth noting that we could also approach this problem using the more familiar method of *least squares*. Recall that least squares involves minimizing the sum of the squared residuals between the observed and expected values. More formally, we want to minimize the Euclidean norm squared of the residual $(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$. That is, the quantity

$$\left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 = \left([Y_1 - (\mathbf{X}\boldsymbol{\beta})_1]\right)^2 + \left([Y_2 - (\mathbf{X}\boldsymbol{\beta})_2]\right)^2 + \cdots + \left([Y_i - (\mathbf{X}\boldsymbol{\beta})_n]\right)^2,$$

where $(\mathbf{X}\boldsymbol{\beta})_i$ denotes the ith component of the vector $(\mathbf{X}\boldsymbol{\beta})$.

We could also rewrite this as

$$\begin{aligned}
\left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 &= \left([Y_1 - (\mathbf{X}\boldsymbol{\beta})_1]\right)^2 + \left([Y_2 - (\mathbf{X}\boldsymbol{\beta})_2]\right)^2 + \cdots + \left(\left[Y_i - (\mathbf{X}\boldsymbol{\beta})_n\right]\right)^2 \\
&= \sum_{i=1}^{n} \left(Y_i - (\beta_1 + \beta_2 x_i)\right)^2.
\end{aligned}$$

You might recall (from some linear algebra class you might have taken) that for some vector $\boldsymbol{\theta}$

$$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{\theta} = \begin{bmatrix}\theta_1 & \theta_2 & \cdots & \theta_n\end{bmatrix}\begin{bmatrix}\theta_1 \\ \theta_2 \\ \vdots \\ \theta_n\end{bmatrix} = \theta_1^2 + \theta_2^2 + \cdots + \theta_n^2 = \sum_i^n \theta_i^2.$$

Thus, if $\boldsymbol{\theta} = \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)$, then we can write

$$\begin{aligned}
\left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|^2 &= \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^{\mathsf{T}}\left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) \\
&= \mathbf{y}^{\mathsf{T}}\mathbf{y} - 2\boldsymbol{\beta}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{y} + \boldsymbol{\beta}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{X}\boldsymbol{\beta}.
\end{aligned}$$

All that's left is to differentiate this expression with respect to $\boldsymbol{\beta}$, set to 0, and solve. Let

$$S = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^{\mathsf{T}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

Thus,

$$\frac{\partial S}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^{\mathsf{T}}\mathbf{y} + 2\mathbf{X}^{\mathsf{T}}\mathbf{X}\boldsymbol{\beta} = 0$$

$$\mathbf{X}^{\mathsf{T}}\mathbf{Y} = \mathbf{X}^{\mathsf{T}}\mathbf{X}\boldsymbol{\beta}$$

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y}.$$

Note this resulting algebraic solution is *identical* to that obtained earlier.

In fact, we could show that both solutions are equivalent to the MLE estimates for $\boldsymbol{\beta}$ (the Gaussian linear model is nice in the sense that the parameter estimates – namely the solution to the linear set of equations, the least squares estimate, and the maximum likelihood estimate – are all the same). For our 'fake' data:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{Y}$$

$$= \begin{bmatrix} 10.25 \\ 1.75 \end{bmatrix}.$$

Thus, our estimates for the intercept and slope are $\hat{\beta}_1 = 10.25$ and $\hat{\beta}_2 = 1.75$, respectively.

We would next estimate the error variance for $\hat{\beta}_1$ and $\hat{\beta}_2$. First, we derive an estimate of the variance-covariance matrix for the vector $\boldsymbol{\beta}$ estimates as

$$\text{var}(\hat{\boldsymbol{\beta}}) = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\sigma_e^2.$$

We can estimate σ_e^2 from the residual sums of squares (RSS) as

$$\text{RSS} = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^{\mathsf{T}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

If the model estimates p parameters, then the estimate of σ_e^2 is simply $\text{RSS}/(N - p)$ where N is the number of data points. Thus,

$$\text{Var}(\hat{\boldsymbol{\beta}}) = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\frac{\text{RSS}}{(N-p)}$$

$$= (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\frac{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^{\mathsf{T}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}{(N-p)}.$$

So, for our 'fake' data (where $N = 8$ and $p = 2$), and our vector $\hat{\boldsymbol{\beta}}$,

$$\text{RSS} = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^{\mathsf{T}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

$$= 14.75,$$

and thus

$$\text{Var}(\hat{\boldsymbol{\beta}}) = \frac{(\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^{\mathsf{T}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}{(N-p)}$$

$$= \begin{bmatrix} 0.6146 & -0.6146 \\ -0.6146 & 1.2292 \end{bmatrix}.$$

From this, we can calculate $\widehat{\text{SE}}(\hat{\beta}_1) = \sqrt{0.6146} = 0.7840$, and $\widehat{\text{SE}}(\hat{\beta}_2) = \sqrt{1.2292} = 1.1087$. And, since a 95% CI for $\hat{\beta}_2$ (approximately $\hat{\beta}_2 \pm 2\text{SE}$; [−0.4674, 3.9674]) clearly includes 0, we would conclude no significant `sex` effect at a nominal $\alpha = 0.05$ level.

Our 'hand calculated' estimates of slope and intercept, and variances for both parameters, are identical to the values returned by fitting the linear model in any statistical software package (below):

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	10.25000	0.78395	13.07	<.0001
sex	1	1.75000	1.10868	1.58	0.1655

end sidebar

6.2. Linear models and the 'design matrix': the basics

In program **MARK**, the default design matrix for a given model is determined by the parameter structure of the model you are trying to fit (number of groups, and the number and structure of the parameters; i.e., the PIMs). This design matrix is then modified in various ways to examine the relative fit of different models to the data. In order to understand this process, it is essential that you understand how the design matrix is constructed.

We'll introduce the concept of a design matrix by means of an example. Suppose you are doing a 'typical' ANOVA on data with a single classification factor (say, 'treatment'). Suppose that there are 4 levels for this factor (perhaps a control, and 3 different levels of the 'treatment'). You want to test the hypothesis that there is no heterogeneity among 'treatment' levels ($\mathbf{H_o}$: $\mu_1 = \mu_2 = \mu_3 = \mu_4$). Recall from the preceding discussion that this problem can be formulated as an applied linear regression problem using '0/1 dummy variable' coding for the different levels of the 'treatment'.

Recall the previous example (above) which had 1 'treatment' or classification factor (`sex`), with 2 levels (male and female). The corresponding regression model was

$$Y_i = \beta_1 + \beta_2 x_i + \epsilon_i,$$

where x represented a coding variable specifying male or female (i.e., `sex`, the classification variable in the ANOVA analysis). The coding variable took on the value of '`0`' (for females) or '`1`' (for males).

What would the regression model look like for our present example, with 4 levels of the treatment factor instead of 2? How can we use a simple '`0`' or '`1`' dummy variable coding scheme (which clearly has only 2 'levels') to accommodate a treatment factor with 4 levels? The key is to consider the answer to the following question: if x_i can take on 1 of 2 values (`0` or `1`), then how many values of x_i do we need to specify k levels of the classification variable (i.e., the treatment variable)? If you think about it for a moment, you should realize that the answer is $k - 1$ (which, of course, corresponds to the familiar '$k - 1$ degrees of freedom' for a single-classification ANOVA).

Thus, for the present example, x_1, x_2 and x_3 could be:

$$x_1 = \begin{cases} \mathbf{1} \text{ if trt } \mathbf{1} \\ \mathbf{0} \text{ if other} \end{cases} \qquad x_2 = \begin{cases} \mathbf{1} \text{ if trt } \mathbf{2} \\ \mathbf{0} \text{ if other} \end{cases} \qquad x_3 = \begin{cases} \mathbf{1} \text{ if trt } \mathbf{3} \\ \mathbf{0} \text{ if other} \end{cases}$$

Clearly, when the coefficients for x_1, x_2 and x_3 are all **0**, then the treatment level must be **4** ('other').

Thus, our regression equation for this example would be:

$$Y_i = \beta_1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 x_3 + \epsilon_i.$$

In this case, β_1 is the intercept, while β_2, β_3 and β_4 correspond to the slopes for each of the levels of the treatment factor. Since there are 4 levels of the treatment, 3 slopes are needed to code 4 levels of the treatment, because 1 of the levels of the treatment corresponds to the case where all 3 slopes are 0. Parameters β_2, β_3 and β_4 refer to treatment levels 1, 2, and 3, respectively. If $x_1 = x_2 = x_3$, then β_1 refers to treatment level **4**. In other words, the intercept corresponds to treatment level **4**.

begin sidebar

why is level 4 the intercept?

Choosing the intercept to specify treatment **4** was entirely arbitrary. We could for example have used any other level of the treatment as the intercept, and adjusted the coding for the remaining levels according. For example, we could have used level 1 of the treatment as 'other' (i.e., the intercept), as follows:

$$x_1 = \begin{cases} \mathbf{1} \text{ if trt } \mathbf{2} \\ \mathbf{0} \text{ if other} \end{cases} \qquad x_2 = \begin{cases} \mathbf{1} \text{ if trt } \mathbf{3} \\ \mathbf{0} \text{ if other} \end{cases} \qquad x_3 = \begin{cases} \mathbf{1} \text{ if trt } \mathbf{4} \\ \mathbf{0} \text{ if other} \end{cases}$$

In this case, when the coefficients for x_1, x_2 and x_3 are all **0**, then the treatment level must be **1** ('other'). Our regression equation would stay the same

$$Y_i = \beta_1 + \beta_2 x_1 + + \beta_3 x_2 + \beta_4 x_3 + \epsilon_i,$$

but now, parameters β_2, β_3 and β_4 refer to treatment levels **2**, **3**, and **4**, respectively. If $x_1 = x_2 = x_3$, then β_1 refers to treatment level **1**.

What is important to note here is that in either case, one of the levels is specified by the intercept (i.e., β_1). This level is referred to as the '*control*' or '*reference*' level. In this design, then, the other levels ($\beta_2 \rightarrow \beta_4$) are 'offsets' from this reference (control) level (i.e., the other β terms represent the magnitude that a particular level of the treatment differs from the control).

We will discuss this and related issues in much more detail later.

end sidebar

From this step, it is fairly straightforward to derive the design matrix (so-called because it fully represents the design of the analysis). The design matrix is simply a matrix showing the structure of the 'dummy' coding variables in the analysis. Because there are 4 parameters being estimated in the equation (β_1, β_2, β_3 and β_4), each corresponding to the 4 levels of the main effect, then the design matrix will be a (4×4) square matrix.

To help construct the design matrix, we can decompose the general regression equation for this analysis (above) into n regression equations, where n is the number of parameters in the regression equation (i.e., the number of levels of the main effect; $n = 4$).

treatment	*equation*
1	$Y_i = \beta_1(\mathbf{1}) + \beta_2(\mathbf{1}) + \beta_3(\mathbf{0}) + \beta_4(\mathbf{0})$
2	$Y_i = \beta_1(\mathbf{1}) + \beta_2(\mathbf{0}) + \beta_3(\mathbf{1}) + \beta_4(\mathbf{0})$
3	$Y_i = \beta_1(\mathbf{1}) + \beta_2(\mathbf{0}) + \beta_3(\mathbf{10}) + \beta_4(\mathbf{1})$
4	$Y_i = \beta_1(\mathbf{1}) + \beta_2(\mathbf{0}) + \beta_3(\mathbf{0}) + \beta_4(\mathbf{0})$

The design matrix **X** (below) is simply the matrix of the coefficient multipliers in these equations:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

While this seems logical enough, there are, in fact, a number of alternative parameterizations of the design matrix, each of which yields the same 'model fit', but which have different interpretations.

For example, all 6 of the following design matrices ($\mathbf{X}_1 \rightarrow \mathbf{X}_6$) give equivalent model fits for our example problem:

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{X}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{X}_3 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{X}_4 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{X}_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{X}_6 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & -1 & -1 & -1 \end{bmatrix}$$

$\mathbf{X}_1$ (above) is the design matrix we derived previously; we estimate an intercept term for the last 'treatment' level (4), and then an additional 'treatment' effect for 'treatment' levels 1, 2 and 3. Matrices $\mathbf{X}_2 \rightarrow \mathbf{X}_4$ are based on the same underlying idea, except that the intercept specifies a different 'reference' level in each case (see preceding `-sidebar-`). For example, in $\mathbf{X}_2$, the intercept corresponds to treatment level 1. In $\mathbf{X}_3$, the intercept corresponds to treatment level 2. And, in $\mathbf{X}_4$, the intercept corresponds to treatment level 3.

The matrix $\mathbf{X}_5$ is an *identity* design matrix. Here, each row corresponds to a parameter, and each column corresponds to a parameter. Thus, each parameter represents a treatment estimate directly, not as an 'offset' (deviation) from the 'control' or 'reference' (i.e., the intercept).

In matrix $\mathbf{X}_6$, we estimate a mean parameter among treatment levels, and then an 'offset' for each of the 4 levels; the first column corresponds to the mean treatment value, and the remaining columns provide the treatment effects.

We'll consider these different design matrices later in the chapter. Note that the choice of the structure of the design matrix doesn't affect the *real* estimates of the parameters (φ, or p, for example, on the real probability scale) – but it does change how estimates of the individual slope parameters (i.e., the β estimates) in the linear model are interpreted. We will see many examples of this later in the chapter.

Perhaps the most important thing to remember in considering design matrices is that the number of rows corresponds to the number of parameters in your PIMs, whereas the number of columns corresponds to the number of these parameters you are trying to individually estimate. As we will see in the next section, this distinction becomes important when fitting models where parameters are constrained to be functions of 1 or more effects.

Finally, a more complex example, using 2 groups (say, males and females), with multiple levels of a treatment within group (i.e., within sex). This example is clearly analogous to a 2-way ANOVA, with 2 main 'effects' (treatment, and sex). Again, assume there are 4 possible treatment levels. The response

variable Y can be decomposed as:

$$Y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk},$$

where α_i is the sex (group) effect, β_j is the treatment effect, and $(\alpha\beta)_{ij}$ is the interaction of the two.

The corresponding regression equation would be:

$$\begin{aligned} Y_{ij} = {} & \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{t}_1) + \beta_4(\texttt{t}_2) + \beta_5(\texttt{t}_3) \\ & + \beta_6(\texttt{SEX.t}_1) + \beta_7(\texttt{SEX.t}_2) + \beta_8(\texttt{SEX.t}_3) + \epsilon. \end{aligned}$$

If we derive the design matrix directly from this expression, then we see that we have 8 rows: 2 levels for `SEX` (male or female) multiplied by 4 treatment levels within sex (remember, $(n-1) = 3$ columns). The design matrix $\mathbf{X}$ (shown below) would also have 8 columns, corresponding to the intercept, the `SEX` (group effect), and the treatment and interaction terms, respectively

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The first column represents the intercept, the second column the group (`SEX`) effect (`1`=male, `0`=female; i.e., the additive effect of males-females), columns 3-5 represent the treatment effect ($t_1 \rightarrow t_3$), and columns 6-8 represent the interactions of `SEX` (male) and treatment. Why male, and not female? It depends on the coding – in this case, we're using '`0`' to represent females, and thus the interaction columns have non-zero elements for males only.

Suppose, for example, rather than the full model (with interactions), you wanted to fit the additive model consisting simply of the 2 main effects (no interaction term):

$$Y_{ijk} = \mu + \alpha_i + \beta_j + \epsilon_{ijk},$$

which, in regression form, is

$$Y_{ij} = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{t}_1) + \beta_4(\texttt{t}_2) + \beta_5(\texttt{t}_3) + \epsilon.$$

Using the design matrix $\mathbf{X}$ (above), this is easily accomplished by simply deleting the columns corresponding to the interaction terms:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Got it? As we work through this chapter, we'll come back to the concept of a 'linear model' and the 'design matrix' with considerable frequency, but hopefully you have the basic idea. In the examples we will explore in this chapter, you will learn the basic steps of creating these linear 'dummy variable' models, design matrices, and how to use them with **MARK** to test a variety of hypotheses.

The only thing we now need to consider is – how can we use 'regression models' for analysis of mark-encounter data, since both survival and encounter are not 'normal' response variables – normal in the sense that they are both constrained to be values from $0 \rightarrow 1$? If you simply regressed 'live = `1`/dead = `0`' or 'seen = `1`, not seen = `0`' on some set of explanatory variables $\mathbf{x}$, it is quite conceivable that for some values of $\mathbf{x}$ the estimates value of survival or encounter would be > 1 or < 0, which clearly can't be correct! However, we clearly want to be able to bring the full power of ANOVA-type analyses to bear on capture-encounter studies.

As mentioned earlier in this chapter, the way around this problem is to transform the probability of survival or encounter, such that the transformed probabilities have been mapped from $[0, 1]$ to $[-\infty, +\infty]$, which is of course the 'assumption' for normal linear regression models. To accomplish this, **MARK** uses a 'link function' (see the following `-sidebar-` for more general background on link functions). In fact, **MARK** allows you to choose among a number of different link functions (some of which are more appropriate for certain types of analyses than others).

The default link function is the sin link, which has very good properties for analyses that use what is known as the 'identity matrix' (much more on this matrix in a minute). For models which don't use the identity matrix (such as constrained models), the logit link function is preferred (this is also discussed later on in this chapter). Using these transformed probabilities, we can use linear regression models analogous to the one we just considered in the bat wing length example introduced at the start of this chapter.

We will now consider a simple example in detail, based on live encounter data from the European dipper, to demonstrate how linear models are constructed using **MARK**.

begin sidebar

What is a 'link function'?

In the context of analysis of data from marked individuals, a 'link function' is a transformation of probability such that the transformed probability is mapped from $[0, 1]$ to $[-\infty, +\infty]$. For example, suppose you want to express a dichotomous (i.e., binary) response variable Y (e.g., survival or encounter) as a function of 1 or more explanatory variables. Let $Y = 1$ if alive or present; otherwise $Y = 0$. Let $\mathbf{x}$ be a vector of explanatory variables, and $p = \Pr(Y = 1 \mid x)$ is the probability of the response variable you want to model. We can construct a linear function of this probability by using a certain type of transform of the probability, p.

For example, the logit transformation (one of several transformation or link functions you can use with **MARK**) is given as:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \beta_1 + \beta_2 x,$$

where β_1 is the intercept, and $\boldsymbol{\beta}_2$ is the vector of slope parameters. Since $\theta = \ln(p/(1-p))$ has inverse $p = e^{\theta}/(1+e^{\theta}) = 1/(1+e^{-\theta})$, then the back-transformed estimate of $\hat{p}$ (i.e., back-transformed to the $[0, 1]$ probability scale) is

$$\hat{p} = \frac{e^{\hat{\beta}_1 + \hat{\beta}_2 x}}{1 + e^{\hat{\beta}_1 + \hat{\beta}_2 x}} = \frac{1}{1 + e^{-\hat{\beta}_1 - \hat{\beta}_2 x}}.$$

In other words, we can express the probability of the event (survival or encounter) as a linear function of a vector of explanatory variables. The logit (or logistic) model is a special case of a more general class

of linear models where a function $f = f(m)$ of the mean of any arbitrary response variable is assumed to be linearly related to the vector of explanatory variables. The function f is the 'link' between the random component of the model (the response variable) and the fixed component (the explanatory variables). For this reason, the function $f(m)$ is often referred to as a 'link function'.

MARK allows you to choose among a number of different link functions (we will discuss the various link functions later in this chapter), some of which are more appropriate for certain types of analysis than others. **MARK** estimates the intercept and vector of the slope parameters, using the specified link, and then reconstitutes the values of the parameter from the values of the explanatory variables, x. **MARK** does this in 2 steps: (1) first, **MARK** reconstitutes estimates of the parameter from $\hat{\beta}_1$, $\hat{\beta}_2$ and x, and then (2) **MARK** computes values of the parameter from f using the back transform f^{-1}. There are several examples of this in the text.

end sidebar

6.3. The European dipper – the effects of flooding

We return to our analysis of the European dipper data set. Now, we will examine the effects of a specific climatic event (flood conditions on the breeding ground) on survival and encounter estimates.

As you may recall from Chapter 3 and Chapter 4, this data set involves 7 occasions of mark-encounter, of both male and female individuals. In those earlier chapters, we focussed on the males exclusively. In this chapter, we'll reanalyze this data set including both males and females. Each year in the study was characterized by the presence or absence of flood conditions. Are years with high (or low) values of either survival or encounter (or both) associated with years when there was a flood? Does the relationship between flood and either survival or encounter differ significantly between male and female dippers? In order to address these questions, we will use the following 'logic sequence':

step 1 - *is there support for an interaction of sex and the covariate (flood) on variation in either survival or encounter?*

step 2 - *if there is no strong support for such an interaction, then is there evidence supporting a difference between the sexes in survival?*

step 3 - *in the absence of an interaction between sex and flood, is there any evidence supporting a linear relationship between survival (or encounter) and the covariate (flood)?*

This is the same sequence of steps used in analysis of covariance (ANCOVA). This is a very basic (and hopefully familiar) analytical design in statistical analysis, and we will demonstrate that the very same approach can be used to analyze variation in survival or encounter. Before we begin, let's recast our analysis in terms of linear models. For the moment, let's use the simplified expression of linear models used in earlier chapters. Our basic model, including our classification variable (`SEX`) is

$$\varphi \text{ (or } p) = \texttt{SEX + FLOOD + SEX.FLOOD + error}.$$

One thing you might ask at this point is – why isn't `TIME` included in the model? The answer lies in the fact that when we talk about constraints, we are speaking about 'applying' a constraint to a particular starting model.

For example, we could start with the standard CJS model, with time-dependence of both survival and encounter probabilities. Then, we could apply a specific constraint to this model. In this example, we replace the 'random' effect of time in the CJS model by the 'specific' temporal effect of `FLOOD`. `FLOOD` is a particular case of time-dependence, because years with the same flood condition will share the same survival value. Thus, models with the `FLOOD` factor are 'nested' in the corresponding CJS model with the 'random' `TIME` factor. Of course, `FLOOD`, just like `TIME`, can be crossed (interaction) with `SEX`.

Our first step in this analysis is to test for the significance of the interaction term: `SEX.FLOOD`. If the interaction term is not significant, we can proceed to test for the significance of the other factors.

How do we test for significance of the interaction term? Clearly, we test the model with the interaction against the same model without the interaction – using either relative model support (the QAIC approach), or (if you prefer) a LRT. The difference in fit of these two models is a 'test' of the interaction term.

$$\begin{array}{rl} & \varphi \text{ (or } p) = \texttt{SEX + FLOOD + SEX.FLOOD + error} \\ \textit{versus} & \varphi \text{ (or } p) = \texttt{SEX + FLOOD + error} \\ \hline & \texttt{SEX.FLOOD} \end{array}$$

How do we do this? The basic mechanics are the same as were described in earlier chapters – we use **MARK** to fit the 2 models we want to compare. But, in this case, there is a subtle difference; although the `SEX` term clearly corresponds to the 2 groups on our analysis, how do we incorporate the information specified by the `FLOOD` variable? In other words, how do we 'constrain' our estimates for either sex to be a linear function of flood? What about the design matrix?

OK – here we go – step by step...

Step 1 – reading in the data

Start **MARK**, and create a new project. The data file for this example is the full dipper data set (**`ed.inp`**). Select it, and label the 2 groups '`males`' and '`females`'. [Reminder, you are expected to know or remember which columns in the `.INP` file correspond to which groups]. There are 7 occasions in the dipper data set.

Step 2 – identify the parameters you want to constrain

In any typical analysis, your next step would be to decide on your starting, underlying model. For example, your starting model might include simple time-dependence in both parameters. Remember, this fully time-dependent model is the default starting model for **MARK**.

How do you decide on the structure for the starting model? By using the techniques discussed in the preceding chapters, and the GOF procedures outlined in Chapter 5. Remember – you apply a constraint to a particular underlying (or starting) model – if this model doesn't adequately fit the data, then applying a constraint will not yield a particularly powerful test.

For the moment, let's assume that the model $\{\varphi_{g*t}\, p_{g*t}\}$ (i.e., time and group effects for both survival and encounter) is a good (and valid) starting model. Once you've determined the starting model, you need to determine the parameter indexing that **MARK** will use. For the dipper data set, we have 2 groups, and 7 occasions.

Thus, the PIMs for this model would look like the following:

survival

males

1	2	3	4	5	6
	2	3	4	5	6
		3	4	5	6
			4	5	6
				5	6
					6

females

7	8	9	10	11	12
	8	9	10	11	12
		9	10	11	12
			10	11	12
				11	12
					12

encounter

13	14	15	16	17	18
	14	15	16	17	18
		15	16	17	18
			16	17	18
				17	18
					18

males

19	20	21	22	23	24
	20	21	22	23	24
		21	22	23	24
			22	23	24
				23	24
					24

females

Remember, this is the default model that **MARK** will start with, so there is no need to modify the PIMs at this stage. A GOF test (using the median-$\hat{c}$ – see chapter 5) yields a $\hat{c}$ value of 1.183 (remember, your estimate of $\hat{c}$ might differ somewhat from this value, since the actual estimate of $\hat{c}$ will depend upon the number of simulations you run). Adjust the $\hat{c}$ in the results browser from 1.000 (the default) to 1.183 (remember, this means we're now changing from AIC_c to $QAIC_c$).

This model represents our 'starting point'. Note that there are 24 *structural* parameters – 6 survival parameters for each sex (12 total), and 6 encounter parameters for each sex (12 total). However, recall that there are a couple of non-estimable β-terms here, one for males and females, respectively, so 22 total *estimable* parameters (10 survival, 10 encounter, 2 β-terms). Go ahead and fit this model to the data. Call it '`phi(g*t)p(g*t) - PIM`' (we've added the PIM label so that when we look at the model in the browser, we'll know that this model was constructed using PIMs only).

Results Browser: Live Recaptures (CJS) c-hat = 1.1830000

Model	QAICc	Delta QAICc	QAICc Weight	Model Likelihood	No. Par.	QDeviance
{phi(g*t)p(g*t) - PIM}	599.3016	0.0000	1.00000	1.0000	22	60.4176

Now, we'll fit a 'constrained model' to these data. For the moment, we're concentrating on survival in our analysis of these data, so the parameters we're interested in are in 'survival' matrices, the survival PIMs. Thus, we want to constrain 12 parameters: 6 survival estimates for males, and 6 for females. How do we do this? In other words, we want to make the probability of survival a linear function of other factors. In this particular example, the 'other factors' are `SEX`, and `FLOOD`.

Step 3 – defining the model structure of the linear constraint (modifying the design matrix)

As suggested earlier, linear models are specified via a 'design matrix'. In fact, this is precisely what we'll do in **MARK** – specify a particular design matrix, corresponding to the particular linear model we want to apply to the data. Again, recall that the name 'design matrix' reflects what it does – it is a matrix containing the dummy variable structure which 'codes' the design of the linear model you are fitting to the data. What would the design matrix corresponding to model

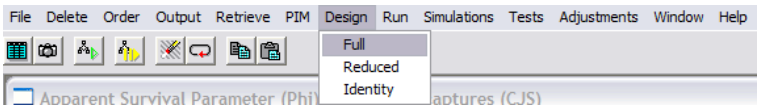
$$\varphi \text{ (or } p) = \texttt{SEX} + \texttt{FLOOD} + \texttt{SEX.FLOOD} + \texttt{error}$$

look like? As a first step, we generally rewrite this model expression more formally. If we consider `FLOOD` as a simple binary variable (a year is either '`flood`' or '`non flood`'), then the corresponding linear model equation would be:

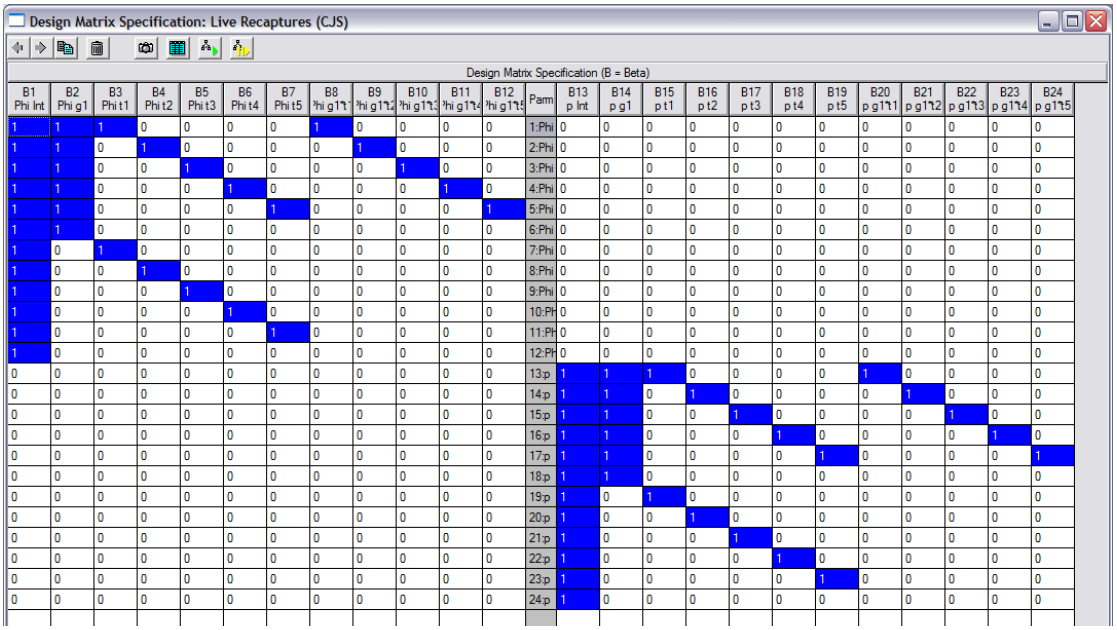
$$Y_{ij} = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{FLOOD}) + \beta_4(\texttt{SEX.FLOOD}) + \epsilon.$$

Recall that each column in the design matrix corresponds to each 'β' term in the model. In the model, we have 4 different 'β' terms – the intercept, the term for `SEX`, the term for `FLOOD`, and the term

corresponding to the (SEX.FLOOD) interaction. So, the design matrix will have 4 columns. Now we need to create, or modify, the design matrix for this model in **MARK**. You may have noticed that there is a menu in **MARK** called '**Design**'. If you pull down the '**Design**' menu, you'll see that you are presented with several options: **full**, **reduced**, and **identity**.



Understanding the distinction between the various options in the '**Design**' menu will take a few steps, so for the moment, we'll start by selecting the '**Design | Full**' menu option from the '**Design**' menu. Once you select '**Full**', a new window will pop up on the **MARK** desktop, looking like the following:



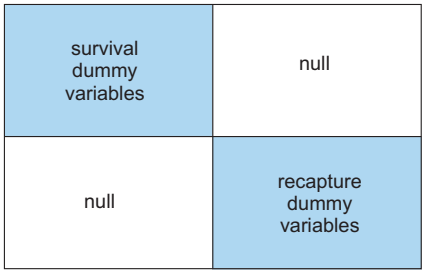
B1 Phi Int	B2 Phi g1	B3 Phi t1	B4 Phi t2	B5 Phi t3	B6 Phi t4	B7 Phi t5	B8 Phi g1t1	B9 Phi g1t2	B10 Phi g1t3	B11 Phi g1t4	B12 Phi g1t5	Parm	B13 p Int	B14 p g1	B15 p t1	B16 p t2	B17 p t3	B18 p t4	B19 p t5	B20 p g1t1	B21 p g1t2	B22 p g1t3	B23 p g1t4	B24 p g1t5
1	1	1	0	0	0	0	1	0	0	0	0	1:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	1	0	0	0	2:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	1	0	0	3:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	1	0	4:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0	0	0	0	1	5:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	6:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	7:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	8:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	9:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	10:PH	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	11:PH	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	12:PH	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	13:p	1	1	1	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	14:p	1	1	0	1	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	15:p	1	1	0	0	1	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	16:p	1	1	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	17:p	1	1	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	18:p	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19:p	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	20:p	1	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	21:p	1	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	22:p	1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	23:p	1	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	24:p	1	0	0	0	0	0	0	0	0	0	0	0

This new window is the '**Design Matrix Specification**'.* Here we are using the full dipper data, consisting of 2 groups (males, females), and 7 sampling occasions). For larger data sets (more groups, more occasions), you may only see parts of it, so you need to get familiar with the basic layout. First, and most obviously, the matrix is split into a series of columns. In this example, there are in fact 25 columns, 1 each for each of the 24 'potentially' estimable parameters (remember, this is the CJS model, with 12 survival and 12 encounter parameters), and 1 'parameter label' column (the grey one in the middle of the matrix, labeled 'Parm'). At the top of each of the columns representing the 24 parameters, you'll see headers like 'B1', 'B2' and so forth. Recall that 'B' stands for 'β' – thus, the columns 'B1', 'B2', 'B3' refer to model parameters 'β_1', 'β_2' and 'β_3', respectively.

How many rows in the design matrix? You might guess 24. You would be correct! The design matrix – the full design matrix – for the CJS model for this data set is (in effect) a (24×24) matrix (we'll ignore the parameter column for the moment).

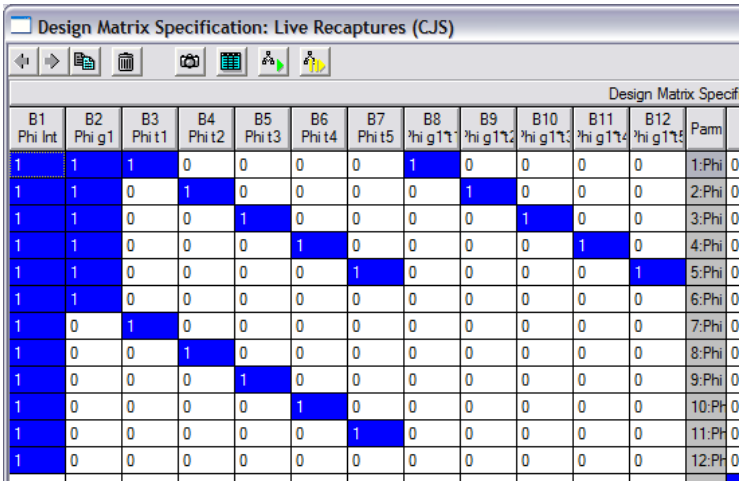
* This is where it helps to have a big monitor!

If you look at the design matrix for this model carefully, you'll see that it has the following general structure:



In the upper-left and lower-right quadrants, we have the 'dummy variable coding' for the survival and encounter probabilities, respectively. In the upper-right and lower-left quadrants, we have what we'll refer to as the 'null' coding.

What are these codings? Let's look at the upper-left quadrant first – the dummy coding for the survival parameters. This quadrant is shown below:



First – how big (number of rows, number of columns) is the quadrant? Since the full matrix is (in effect) (24×24), then $1/4$ of this is a (12×12) matrix. Thus, the upper-left quadrant are the first 12 columns and rows (going from left to right, top to bottom). In this quadrant (pictured at the top of this page), we see a column of 12 '1's (the column labeled '`B1 - Phi Int`'), a second column of 6 '1's followed by 6 '`0`'s, then a series of columns with '1's going down along a diagonal. (*Note*: the column shows a label of '`B1 - Phi Int`'. **MARK** provides the indicated column labels by default for the full design matrix corresponding to a fully time-dependent model structure, as specified by the PIMs. For other models, the default column labels are simply `B`*n* (e.g., `B1`, `B2`, `B3`...). To change the label for a particular column in the design matrix, simply right-click within any cell in a particular column. This will spawn a series of options you can select from – one of which is to label the column. Meaningful (to you) column labeling is very useful until you've developed some experience with design matrices – the labels will help you keep track of things.)

Remember, what is pictured here is the part of the design matrix which corresponds to the survival parameters for the 'full' general model - model $\{\varphi_{g*t}\, p_{g*t}\}$ (where g = `SEX`). In other words, the basic CJS

model for 2 groups. This is what '**Full**' refers to when you select that option from the '**Design**' menu - '**Full**' means the fully time-dependent model, as specified by the PIMs.

Second, the actual structure of this part of the design matrix reflects the linear model corresponding to model $\{\varphi_{g*t}\}$ (remember, we're only considering the survival part of the design matrix for now). Now, given that there are 7 occasions, and 2 groups (males and females), the linear model corresponding to model φ_{g*t} is (wait for it. . .)

$$Y_{ij} = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{t}_1) + \beta_4(\texttt{t}_2) + \beta_5(\texttt{t}_3) + \beta_6(\texttt{t}_4) + \beta_7(\texttt{t}_5) + \beta_8(\texttt{SEX.t}_1) + \beta_9(\texttt{SEX.t}_2) + \beta_{10}(\texttt{SEX.t}_3) + \beta_{11}(\texttt{SEX.t}_4) + \beta_{12}(\texttt{SEX.t}_5) + \epsilon.$$

Pretty cumbersome looking, but not too hard if you go through it slowly. One term for the intercept (β_1), one term for the 'group' effect (i.e., sex, β_2), 5 terms for the 6 time intervals over which we hope to estimate survival (β_3 to β_7), and then 5 terms for the interaction of sex and time (β_8 to β_{12}) – a total of 12 parameters. Thus, 12 columns for this part of the design matrix, just as we observe in the preceding figure. Remember – even though there are 6 time intervals for survival (7 occasions = 6 intervals), we need only 5 columns – 5 parameters – to code them. So, for 6 intervals (which is analogous to 6 levels of a 'time' treatment), you need $(6 - 1) = 5$ parameters. This is precisely where the '$n - 1$' degrees of freedom bit comes from in ANOVA (and which is almost never explained to you in your basic stats class – they simply teach you 'rules' like '$n - 1$ degrees of freedom. . .' without explaining why. Now you know why! – it relates to how the linear model is set up and reflects the number of columns needed to code for a 'treatment' in the design matrix).

However, note that the columns of the design matrix are labeled '`B1`' to '`B12`'. **MARK** defaults to labeling the first column (the 'intercept' column) as '`B1`', which seems counter to the (fairly) standard linear models convention of using β_0 for the intercept. While you can always change it in the **MARK** preferences if you want, we will adopt the convention of using β_1 for the intercept. Doing so makes it much easier to relate the terms of the linear model to specific columns in the design matrix (column `B1` in the design matrix corresponds to β_1 in the linear model, column `B2` corresponds to β_2, and so on).

OK, so that's the basic structure – 12 columns, and 12 rows (12×12): 12 columns for the 12 parameters of the linear model, and 12 rows for the (2 groups $\times$ 6 occasions). Remember – the number of columns in the design matrix represents the number of parameters we want to estimate, while the number of rows in the design matrix reflects the number of parameters in the underlying model (i.e., the parameter indexing specified in the PIMs). If the number of columns is $<$ the number of rows, then this implies that we are applying a *constraint* to the underlying model structure.

What about the actual dummy variable coding? Well, the intercept column should be straightforward – it's all '`1`'s. Next – the column coding for '`SEX`'. Here it is arbitrary which dummy variable you use to code for '`males`' and for '`females`' (there is a fairly common convention for using '`1`'s for males, and '`0`'s for females, but it makes absolutely no difference at all). Again, there are 2 levels of this effect – 2 sexes. So, we need 1 column of dummy variables to code for '`SEX`' (that old '$n - 1$ degrees of freedom' thing again). So far, so good – and hopefully, pretty easy stuff.

What about time? Well, if you remember the introduction to linear models and the design matrix from earlier in this chapter, you realize that what **MARK** does is use a row of all '`0`'s to code for the last time interval, and then '`1`'s along the diagonal to code for the preceding intervals. The choice of using '`0`'s first, then the diagonal, is entirely arbitrary (you could, for example, use a diagonal set of '`1`'s, with the last row being all '`0`'s – makes little difference to the overall model fit, or the reconstituted estimates) – it is a **MARK** default. Note that this pattern is repeated twice – once for the males, and once for the females. Look closely – make sure you really do get it. Finally, the interaction terms. Pretty easy – just multiply the '`SEX`' column by the '`TIME`' columns to generate the '`SEX.TIME`' interaction columns. Got it? Good!

Now, what about encounters? So far, we've been talking only about modeling survival. Well, since the general model is $\{\varphi_{g*t}\ p_{g*t}\}$, then the structure for the design matrix for encounters should be identical to the one for survival, except it is located in the lower right quadrant of the design matrix – the encounter part of the design matrix is pictured below. Before we proceed, what about the 2 'null' quadrants? Well, since 'null' generally refers to '0', you might suspect that the 'null' quadrants are filled entirely with '0's. You would be correct.

13:p	1	1	1	0	0	0	0	1	0	0	0	0
14:p	1	1	0	1	0	0	0	0	1	0	0	0
15:p	1	1	0	0	1	0	0	0	0	1	0	0
16:p	1	1	0	0	0	1	0	0	0	0	1	0
17:p	1	1	0	0	0	0	1	0	0	0	0	1
18:p	1	1	0	0	0	0	0	0	0	0	0	0
19:p	1	0	1	0	0	0	0	0	0	0	0	0
20:p	1	0	0	1	0	0	0	0	0	0	0	0
21:p	1	0	0	0	1	0	0	0	0	0	0	0
22:p	1	0	0	0	0	1	0	0	0	0	0	0
23:p	1	0	0	0	0	0	1	0	0	0	0	0
24:p	1	0	0	0	0	0	0	0	0	0	0	0

begin sidebar

changing the default reference level

In the preceding, we've used the default coding system in **MARK** to specify the '**full**' design matrix. The default uses the last time interval or occasion as the reference (i.e., the last time interval or occasion is represented by the intercept). As will be discussed later in this chapter (p. 77), there may be reasons why you don't want to use the last interval or occasion as the intercept – in particular, if it involves confounded parameters. For example, in a fully time-dependent CJS model, the final φ and p estimates are confounded. In such cases, it may make sense to change the default reference level to (say) the first interval or occasion. **MARK** makes it easy to do so – simply access '**File | Preferences**', and check the box '**For time effects in the using matrix, make the first row the reference value**'. Recall that the first row corresponds to the first interval or occasion.

end sidebar

Back to the problem at hand – we want to constrain the survival estimates – the first 12 parameters (rows 1 → 6 for the males, and rows 7 → 12 for the females). In **MARK**, you constrain parameters by 'modifying' the design matrix. For our present example, we want to constrain survival to be a function of SEX, FLOOD, with a SEX.FLOOD interaction. Recapture probability we'll leave as is – with simple SEX and TIME differences, with a SEX.TIME interaction. Simple designs tend to be very easy, more complex designs obviously less so. So, we really need to consider only the structure of the design matrix for survival.

In fact, all you really need to do is write out the linear model equation for survival. In this case, it would be:

$$Y_{ij} = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{FLOOD}) + \beta_4(\texttt{SEX.FLOOD}) + \epsilon.$$

A term for the intercept (β_1), a term for the sex effect (β_2), a term for the flood effect (β_3 – remember that flood is a simple binary state variable – either 'flood' or 'no flood'), and a term for the interaction of the two (β_4). So, the survival part of the design matrix would consist of 4 columns, corresponding to this linear model. The number of rows? Again, the number of rows is the product of the number of

time intervals specified in the PIMs, and the number of groups (so, $6 \times 2 = 12$ rows).

We already know how to code the intercept term, and the SEX term. What about the FLOOD term? Well, since flood state is a binary variable, we can use '1' to indicate a flood year, and '0' to indicate no flood. In this study, the flood occurred during the 2nd and 3rd breeding seasons only.

Thus, the design matrix for the survival parameters will be – for males:

INTERCEPT	SEX	FLOOD	SEX.FLOOD
1	1	0	0
1	1	1	1
1	1	1	1
1	1	0	0
1	1	0	0
1	1	0	0

and for females

INTERCEPT	SEX	FLOOD	SEX.FLOOD
1	0	0	0
1	0	1	0
1	0	1	0
1	0	0	0
1	0	0	0
1	0	0	0

Note that since the SEX column is now all '0', the interaction column will also be a column of '0's, regardless of what is in the FLOOD column. Thus, putting the two sexes together, the design matrix for survival would be:

INTERCEPT	SEX	FLOOD	SEX.FLOOD
1	1	0	0
1	1	1	1
1	1	1	1
1	1	0	0
1	1	0	0
1	1	0	0
1	0	0	0
1	0	1	0
1	0	1	0
1	0	0	0
1	0	0	0
1	0	0	0

Got it? Now, all that's left is to translate this design matrix into **MARK**. There are a couple of ways to do this, but since we have the 'full design matrix' open already, we'll go ahead and modify it.

Now, recall that in the unmodified full design matrix, we have 12 columns for the survival parameters, and 12 columns for the encounter parameters (we're ignoring the encounter parameters for the time being). So, we want to reduce the number of columns for the design matrix for survival from 12, to 4.

It is this reduction that leads us to say that a model where survival is constrained to be a function of SEX and FLOOD is a 'reduced parameter model'. It is reduced because the number of columns (i.e., the number of parameters) is reduced, relative to the starting model.

Now, **MARK** makes it easy to manipulate the design matrix. But, for the moment, we'll do it 'manually', without some of the 'way cool and nifty' shortcuts that **MARK** offers. After some practice, you'll probably skip the manual approach, but then...the manual approach almost always works, even if it takes a bit longer. Basically, we want to keep the column corresponding to the intercept (the B1 column in the matrix **MARK** presents). We also want to keep the SEX column (column B2). Then, we want 2 columns: one for the flood dummy variable, and one for the interaction. The simplest approach to doing this is to manually edit the cells in columns 3 and 4 of the existing design matrix, entering the dummy variable coding for FLOOD, and the SEX.FLOOD interaction, as described earlier. All that's left after that is to delete the other 8 columns, which are no longer needed (there are lots of ways to delete or add columns to the design matrix – note the various menus specifically for this purpose. You can also right-click your way to the required structure). The final design matrix, showing both survival and encounters, is shown below:

File UnDo ReDo AddCol DelCol FillCol Appearance Run PIM Browse Window Help

Design Matrix Specification (B = Beta)

B1	B2	B3	B4	Parm	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16
1	1	0	0	1:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	2:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	3:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	4:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	5:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	6:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	7:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	8:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	9:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	10:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	11:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	12:Phi	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	13:p	1	1	1	0	0	0	0	1	0	0	0	0
0	0	0	0	14:p	1	1	0	1	0	0	0	0	1	0	0	0
0	0	0	0	15:p	1	1	0	0	1	0	0	0	0	1	0	0
0	0	0	0	16:p	1	1	0	0	0	1	0	0	0	0	1	0
0	0	0	0	17:p	1	1	0	0	0	0	1	0	0	0	0	1
0	0	0	0	18:p	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	19:p	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	20:p	1	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	21:p	1	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	22:p	1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	23:p	1	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	24:p	1	0	0	0	0	0	0	0	0	0	0	0

Again, we now have 4 columns coding for the survival parameters, and 12 columns for the encounter parameters. Study this design matrix carefully – make sure you understand how it is constructed, and (obviously) why it has the structure it does.

Note: You may have noticed the grey-shaded column in the various design matrices we've examined thus far. This is a handy little feature of the presentation of the design matrix in **MARK**, which helps you remember which rows of the design matrix correspond to which parameters. This 'guide column'

(for lack of a better name) can be dragged left or right within the design matrix – this is convenient since you can drag it to a point which conveniently separates the survival and encounter parameters to the left or right of the guide column, respectively (as we've shown in this example).

6.3.1. Design matrix options: full, reduced, and identity

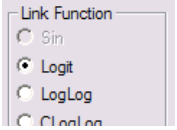
For the preceding example, we started by generating the 'full design matrix', corresponding to the general time-dependent model $\{\varphi_{g*t}\, p_{g*t}\}$. We did this by selecting the '**Full**' option from the '**design matrix**' menu. You might have noticed two other options – one for a '**reduced**' design matrix, and the other for an '**identity**' design matrix. The distinctions among the three options are:

1. The '**full**' design matrix corresponds to a fully '`group` × `time`' model. If the underlying PIM structure is fully '`group` × `time`', then selecting the design matrix option '**full**' from the menu will generate the standard 'group × time' design matrix, based on intercept (reference) coding – **MARK** defaults to using the last time interval as the reference cell (this is discussed elsewhere). If you select the '**full**' design matrix option when the underlying PIM structure is not fully '`group` × `time`', then **MARK** will respond with an error box. The '**full**' design matrix option applies **only** if the underlying PIM structure, is fully 'group × time' – if the PIM structure is **anything** else, then you need to use the '**reduced**' option (see 2, below).
2. If either (i) the underlying PIM structure does not represent a fully '`group` × `time`' model (e.g., if the PIM structure represents a reduced parameter structure – fewer parameters than the '`group` × `time`' structure, or more parameters than a 'group × time' structure – for example, for a TSM model; Chapter 7), or (ii) you want to build a design matrix which constrains the underlying PIM structure (e.g., applying a constraint to a 'group × time' PIM structure where time might be constrained to be a linear function of some environmental covariate), then you would select the '**reduced**' design matrix option (note: the term 'reduced' is perhaps somewhat misleading, since in fact you would use it for a model with more parameters than a '`group` × `time`' model – the word '**reduced**' would suggest fewer parameters).

   The '**reduced**' option is the option you use whenever you want to construct a design matrix which does not correspond to a full '`group` × `time`' model. When you select '**reduced**', **MARK** presents you with a popup window which asks you to specify the number of columns you want in the design matrix, **MARK** defaults to the number of columns represented in the PIMs (i.e., if the PIMs specify n parameters, then the default design matrix will have n columns). If you want to start with fewer columns, you simply change the value in the popup window.
3. The '**identity**' matrix is the default design matrix in **MARK** – The '**identity**' option results in the number of columns in the matrix equaling the number of rows, with the diagonal filled with ones and the rest of the matrix zeros. This is an identity matrix, and provides no constraints on the parameters. The identity matrix can be selected for any model, regardless of the underlying PIM structure.

6.4. Running the model: details of the output

Go ahead and run this model – call it '`Phi(sex*flood)p(sex*time)`'. Now, before you submit this model, something important to notice (image to the right) – the sin link is no longer the default link function. In fact, as you can see, the sin link is not even available. The default (and generally preferred) link function when you change the design matrix from the default identity matrix is now the logit link.



begin sidebar

available link functions in MARK, and...why no sin link with a design matrix?

MARK currently supports 8 different link functions: 4 which constrain parameters to the interval [0, 1] (logit, log-log, complementary log-log, and the default sin link), and 4 which either do not restrict the parameters to be in the [0, 1] interval (identity and log), or which are constrained versions of the logit transform (cumulative and multinomial logit; these will be introduced later).

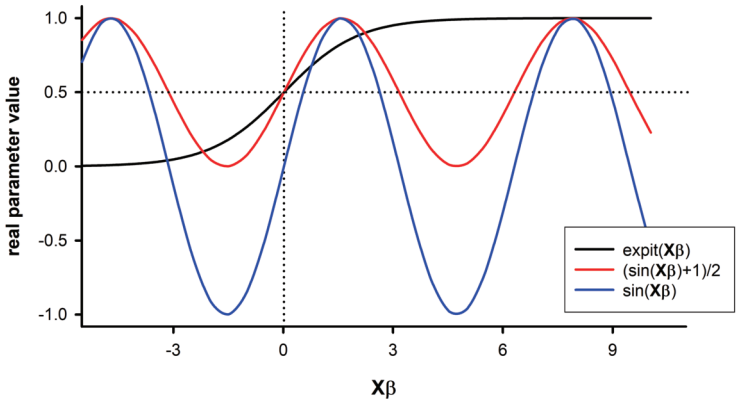
The following tabulates many of the more commonly used link functions and back-transformations in **MARK**, presented assuming a simple linear function $\theta = \beta_1 + \beta_2(x)$, where θ is some parameter bounded [0, 1] (e.g., encounter probability).

link	function	back-transform
sin	$\arcsin(2\theta - 1) = \beta_1 + \beta_2(x)$	$\theta = \left[\sin\left(\beta_1 + \beta_2(x)\right) + 1\right]/2$
logistic	$\log\left(\frac{\theta}{1-\theta}\right) = \beta_1 + \beta_2(x)$	$\theta = \frac{\exp^{\left(\beta_1+\beta_2(x)\right)}}{1+\exp^{\left(\beta_1+\beta_2(x)\right)}}$ (*)
log	$\ln(\theta) = \beta_1 + \beta_2(x)$	$\theta = \exp\left(\beta_1 + \beta_2(x)\right)$
log-log	$\log\left(-\log(\theta)\right) = \beta_1 + \beta_2(x)$	$\theta = \exp\left(-\exp(\beta_1 + \beta_2(x)\right)$
complementary log-log	$\log\left(-\log(1-\theta)\right) = \beta_1 + \beta_2(x)$	$\theta = 1 - \exp\left[-\exp\left(\beta_1 + \beta_2(x)\right)\right]$
identity	$\theta = \beta_1 + \beta_2(x)$	$\theta = \beta_1 + \beta_2(x)$

*Note the equivalence of the following: $\frac{e^{X\beta}}{1+e^{X\beta}} = \frac{1}{1+e^{-X\beta}}$

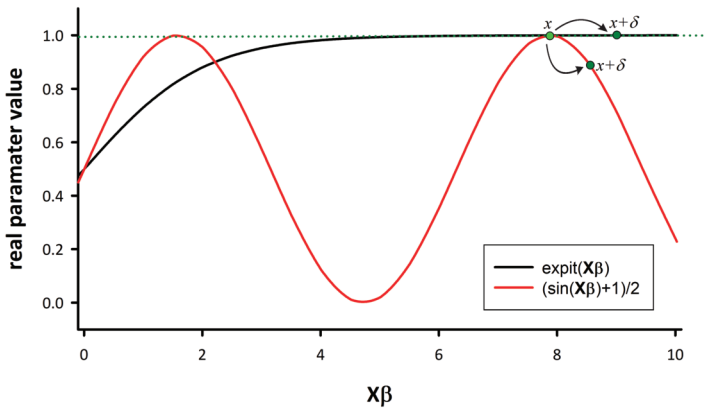
Although the *logit* link function is a familiar, commonly used link function used for statistical analysis of [0, 1] bounded parameters, it can present some problems during the numerical optimization for parameter estimates for β that approach $\pm\infty$ (i.e., as the *real* parameter values approach either 0 or 1 – so-called 'boundary estimates'). As a result, the *sin* link, which tends to perform better for 'boundary estimates', is the default in **MARK**.

Why the difference in performance between the two link functions? To understand the difference, first consider the following figure where we plot parameter values back-transformed to the real probability scale, for both the sin and logit link functions:



In the plot, notice that the 'standard' sin transform (blue line) oscillates between [+1, −1], whereas we want a function that is bound on the interval [0, 1]. To achieve this, **MARK** transforms the sin link by first adding 1 (changing the scale from [+1, −1] to [+2, 0]), and then dividing by 2, transforming it to the [0, 1] interval (the red line in the plot).* Both the original and transformed sin link functions oscillate, whereas the back-transform from the logit scale (black line; the back-transform is usually known as *expit*, the exponential function '`exp`' being the opposite of the '`log`' transformation used in the logit link) is monotonic. This difference (oscillatory versus monotonic) has important implications, and as a result, the logit and sin links have both advantages and disadvantages that are important to understand.

First, why is the sin link the *default* link function in **MARK**? The simple explanation is that the sin link generally 'performs better' (than the logit link) at handling parameters that are estimated near the 0 or 1 boundaries. Have a look at the following plot:



Here we see that for some values of the covariate (represented by $\mathbf{X}\boldsymbol{\beta}$ along the x-axis), the logit link asymptotically approaches 1 (the same thing applies in reverse for the 0 boundary). In fact, over a fair range along the x-axis (say, for values of $\mathbf{X}\boldsymbol{\beta} \geq 6$), it is probably impossible to differentiate between two values of $\mathbf{X}\boldsymbol{\beta}$ in this region of the curve. This is important because **MARK** counts parameters by looking at the slope (partial derivative) of the likelihood function at the putative maximum (this point is called x in the plot). **MARK** does this numerically by evaluating the derivative as

$$\lim_{\delta \to 0} \frac{f(x+\delta) - f(x)}{\delta},$$

where δ is a (very) small deviation away from the value x, on the function $f(x)$ (in this case, $f(x)$ is the likelihood function). If the derivative evaluates to something other than 0, then the parameter has changed the 'curvature' of the (likelihood) function, and thus, the parameter is counted as a distinct parameter. If the function is 'flat' (as the logit function is), then **MARK** has a difficult time detecting such a change in slope, and as a result, may not correctly count the parameter. More formally (as detailed in the Addendum to Chapter 4), both the first and second derivatives of the function approach 0. As a result, determining the rank of the information matrix (a key step in counting parameters) is not a reliable method for determining the number of parameters estimated when using the logit link.

In contrast, because the sin link does not asymptote for any range of values of $\mathbf{X}\boldsymbol{\beta}$, then the numerical evaluation for the slope (derivative) between x and $x + \delta$ will generally $\neq 0$, even if very close to either

* The use of the sin link was introduced by Box (1966) – the original reference used $\sin(\mathbf{X}\boldsymbol{\beta})^2$, whereas **MARK** uses $(\sin(\mathbf{X}\boldsymbol{\beta})+1)/2$ to ensure the back-transformed real probability estimate is on the [0, 1] interval. In fact, it might be better named the *arcsin link*, since estimation is on that transformed scale, and is back-transformed using the sin function (see table on the preceding page), but 'sin link' is somewhat traditional, and is the name used in **MARK**.

the 0 or 1 boundaries. As a result, the rank of the information matrix is a reliable estimator of the number of parameters estimated – the sin link will generally correctly estimate (and count) parameters that fall near either boundary. Thus, the default link function in **MARK** is the sin link.

But, the sin link is 'greyed out'...

So, if the sin link has this advantage 'at the boundaries', why is sin link 'not available' (i.e., is 'greyed out') when the design matrix is modified (i.e., when the design matrix is not a default identity matrix)?

The reason, in fact, reflects the 'downside' of the sin link being *cyclical* (oscillatory), and not *monotonic*. The sin link oscillates between 0 and 1 (under the transformation used in **MARK**), with a period of 2π. This is important because it means that under the sin link, any two covariate values that differ by 2π would yield exactly the same parameter estimate (i.e., $\sin(x)$, $\sin(x + 2\pi)$, and $\sin(x + 4\pi)$ all result in the same value, given x), which might not make much biological sense (in other words, it is unlikely that the biological relationship between some covariate and some parameter oscillates). On the other hand, it is probably much more reasonable to imagine that the parameter changes as a linear, monotonic function of the convariate (e.g., as covariate increases, parameter increases, over the observed range of the covariate), which is what the logit link allows.

As a result, the sin link should only be used with design matrices that are identity matrices, or when only one column in each row has a value not equal to zero, because the sin link will reflect around the parameter boundary, and not enforce monotonic relationships (as discussed above). In other words, when the range of the covariate and/or the parameter estimates cause the link function to straddle 0, the logit link is better for non-identity design matrices.

Thus, although the sin link is the best link function to enforce parameter values in the [0, 1] interval and yet obtain correct estimates of the number of parameters estimated, you need to be careful: in fact, because the sin link is not monotonic, the sin link is simply not available whenever you manipulate the design matrix (i.e., **MARK** protects you from potential error if you try to use the sin link).

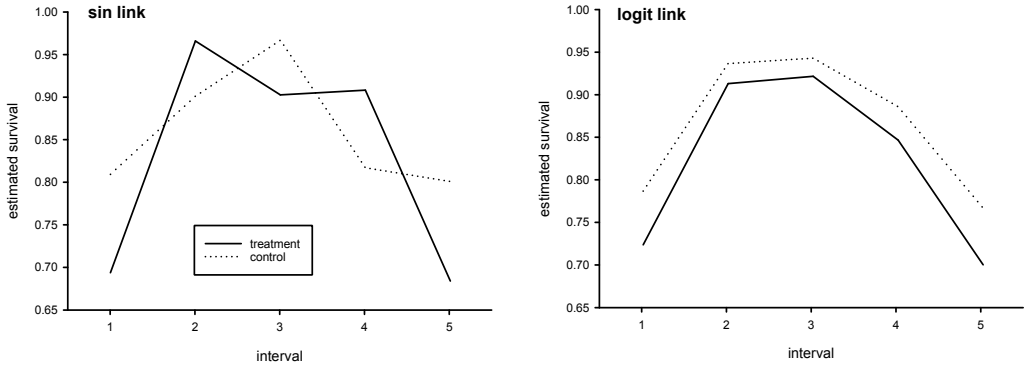
Here is a worked example demonstrating the problem. Consider the survival of some organism deliberately exposed to a potentially toxic chemical. The data set (**sin_example.inp**) consists of 6 sampling occasions, and 2 groups (treatment and control). Analysis of this data set provides a very interesting example of the misbehavior of the sin link function*. Fit the model $\{\varphi_{t+g}\, p_t\}$ using a sin link. **MARK** will not let you do this directly – you'll need to use a '**parameter-specific**' link, which we'll discuss later, and examine the parameter estimates:

	Sin Link		Logit Link	
	Treatment	Control	Treatment	Control
Interval	Estimate	Estimate	Estimate	Estimate
1	0.693804	0.809271	0.723796	0.786824
2	0.966087	0.901265	0.913274	0.936837
3	0.902766	0.966995	0.921747	0.943152
4	0.908336	0.817021	0.846637	0.886047
5	0.684043	0.800919	0.700348	0.767003

As illustrated at the top of the next page, the differences in survival of the treatment and control groups are not consistent between the two link functions. Using the sin link, the control group is not consistently larger than the treatment group, or vice versa. In contrast, note how this constraint is enforced with the logit link function, i.e., the control estimate is always greater than the treatment estimate. The survival probabilities are parallel on the logit scale, but not on the sin scale.

How can this be? The answer again lies in the fact that the logit function is monotonic, whereas the sin function is not. For this example, the biological interpretation of the sin model is nonsense, yet, interestingly, this model provides the minimum AIC by 3 units. Further, the treatment effect would appear to be significant with this model. So, if you use the sin link with a DM, be careful, especially for design matrices where a row in the matrix has non-zero values in more than 1 column.

* When it misbehaves – it is generally not easy to predict when it will.



Some important additional details...

Among the 6 standard link functions (excluding the cumulative and multinomial logit links), the log and identity link functions do not constrain the probability to the interval [0, 1], which can occasionally cause numerical problems when optimizing the likelihood. The log link does constrain real parameter values >1 (which is clearly useful for parameters which are naturally ≥ 0). For the log and identity link functions used with an identity design matrix, **MARK** uses the sin link function to obtain initial estimates for parameters, then transforms the estimates to the parameter space of the log and identity link functions when they are requested.

end sidebar

Go ahead and run this model, and add the results to the results browser. The $QAIC_c$ for this model (given the value for $\hat{c}$ we specified at the outset) is 584.77, which is approximately 15 less than our starting, general model. Thus, based on these 2 models, we would conclude that our 'constrained' (i.e., reduced parameter) model is **many** times better supported by the data than was our general starting model. However, note that our constrained model is reported to have 15 estimated parameters. And yet, if you look at the design matrix, you'll see that we have 16 columns, meaning 16 parameters.

So why does **MARK** only report 15, and not 16? **MARK** reports the numbers of parameters that it *can* estimate, given the data, not the number of parameters that are *theoretically* available to be estimated. As discussed in the preceding `-sidebar-`, the sin link generally does 'better' when dealing with parameters near the boundaries – meaning that it will often 'succeed' at estimating a few more of these 'problem' parameters than the logit link. However, here, where we've modified the design matrix, we need to use the logit link. If you look at the parameter estimates for the constrained model, you see that the encounter probability for males for the third occasion (reconstituted parameter 14) is estimated at 1.0, with a standard error of 0, which is usually diagnostic of a problem. As such, **MARK** doesn't 'count it' in the parameter total. So, **MARK** reports only 15 parameters, not 16. As a result, you need to manually adjust the number of parameters to reflect the 'missing' parameter. You do this with the '**`Adjustments`**' menu. Simply change the number of parameters for this model from 15 to 16.

Now, let's look at the estimates for survival (shown at the top of the next page). Couple of things to notice. First, the parameters 1 → 6 correspond to males, 7 → 12 to females. This is what we specified in the PIMs for the underlying model, to which you applied the constraint reflected in the linear model. Now, remember that there was a flood over the second and third intervals only. This is seen in the estimates – for males, for example, survival in the 2 flood years is 0.4725, while in the non-flood years, survival is 0.5970. For females, the survival during the 2 flood years is 0.4537, while for the non-flood years, it is 0.6403.

dipper analysis
Standard Error and Confidence Intervals Corrected for c-hat = 1.1830000

Real Function Parameters of {phi(sex+flood+sex*flood)p(sex+time+sex*time)

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.5970433	0.0570863	0.4820290	0.7022920
2:Phi	0.4724823	0.0698962	0.3407771	0.6081324
3:Phi	0.4724823	0.0698962	0.3407771	0.6081324
4:Phi	0.5970433	0.0570863	0.4820290	0.7022920
5:Phi	0.5970433	0.0570863	0.4820290	0.7022920
6:Phi	0.5970433	0.0570863	0.4820290	0.7022920
7:Phi	0.6403604	0.0576376	0.5215850	0.7441143
8:Phi	0.4536926	0.0640348	0.3335576	0.5794739
9:Phi	0.4536926	0.0640348	0.3335576	0.5794739
10:Phi	0.6403604	0.0576376	0.5215850	0.7441143
11:Phi	0.6403604	0.0576376	0.5215850	0.7441143
12:Phi	0.6403604	0.0576376	0.5215850	0.7441143

Where do these 'estimates' come from? Notice that these are labeled as '**Real Function Parameters**' – what does that mean? The answer is found if you look in the '**full output**'. Scroll down until you get to the sections showing the **link function parameter estimates**, and the reconstituted **real function estimates**. These are the key 'bits' we need. What are these two types of estimates, and why are they important?

In brief, the '**link function parameters**' are the estimates of the 'slope parameters' in the linear model, on the link function scale. Remember, we're fitting what is in effect a regression model to the data, a regression model that contains estimated parameters – these are the β values referred to earlier. In this example, they are *logit link* function parameter estimates. The real function parameters are the estimates of the survival and encounter parameters themselves, calculated from the regression model on the transformed scale, back-transformed to the real probability scale.

6.5. Reconstituting parameter values

Let's look at the link function parameter estimates a bit more closely. Recall that for the constrained model, 15 parameters were estimated. Can we confirm this by looking at the link function parameter estimates? You can look at the '**Beta**' estimates by clicking on model '`Phi(SEX*FLOOD)p(SEX*time)`' in the browser (to make it active), and then clicking on the third icon from the left in the browser toolbar.

This will open up the editor showing the β-estimates (i.e., the estimates of the intercept and slopes):

Parameter	Beta	Standard Error	95% Confidence Interval Lower	Upper
1:Phi Int	0.5769292	0.2301023	0.1259286	1.0279298
2:sex	-0.1837705	0.3170832	-0.8052536	0.4377125
3:flood	-0.7626908	0.3391790	-1.4274816	-0.0979000
4:sex.flood	0.2593500	0.4845990	-0.6904641	1.2091640
5:p Int	1.0885012	0.6904232	-0.2647282	2.4417306
6:p g1	2.5547736	6.1004028	-9.4020161	14.511563
7:p t1	-0.1122512	1.1841032	-2.4330934	2.2085911
8:p t2	0.7270215	1.1959260	-1.6169934	3.0710365
9:p t3	1.3091620	1.2610077	-1.1624131	3.7807370
10:p t4	0.9080853	0.9696158	-0.9923617	2.8085323
11:p t5	1.4251404	1.0766764	-0.6851454	3.5354262
12:p g1*t1	-2.2989366	6.1921984	-14.435646	9.8377726
13:p g1*t2	12.150812	1917.2767	-3745.7116	3770.0133
14:p g1*t3	-2.5850169	6.3181526	-14.968596	9.7985625
15:p g1*t4	-1.9295379	6.1796697	-14.041691	10.182615
16:p g1*t5	-2.4613900	6.0362762	-14.292492	9.3697117

We see that there are 16 β-estimates, but that only 15 of them are actually estimable (note the standard error for parameter 13). This is why **MARK** reports 15 estimable parameters. No confounded β-terms here (more on why in a minute). So we see again that the number of estimable parameters **MARK** reports in the browser corresponds to the number of estimable 'slopes' in the linear model.

But let's explore what these 'link function parameter estimates' actually mean. This will make explicitly clear what link functions are all about. Let's look at the 'link function parameter estimates' from model {SEX FLOOD SEX.FLOOD} model (shown at the bottom of the preceding page). The first 4 parameter estimates are:

Parameter	*term*	$\hat{\beta}$
1	INTERCEPT	0.576929
2	SEX	−0.183767
3	FLOOD	−0.762691
4	SEX.FLOOD	0.259347

These values are the β coefficients ('slopes') for each of the terms in our linear model, estimated on the logit link scale: one each for the INTERCEPT, SEX, FLOOD, and the SEX.FLOOD interaction (parameters $1 \rightarrow 4$, respectively, as shown above). It is through using these β estimates that we 'reconstitute' our estimates for survival.

A simple example will make this clear. Suppose you were given the equation $Y = 3.2x + 4$. If you were then given some value x, you could interpolate what the value of Y will be (on average, if the equation is a regression line). For example, if $x = 4$, then $Y = 16.8$. The same thing applies in our constraint analysis. We now have an equation (linear model) of the form:

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

Now, from this equation, and given estimates of each of the regression coefficients $\hat{\beta}_i$, we can 'reconstitute' estimates of survival for any value of SEX, FLOOD and the interaction (SEX.FLOOD).

To make sure you really understand what is happening, let's consider how the reconstituted estimate for male survival over the fifth interval (i.e., $\hat{\varphi}_5$) is obtained. We must first compute the estimate of survival on the logit scale using the linear formula noted above, where the values of $\beta_1, \beta_2, \beta_3$ and β_4 are parameters ('beta') 1, 2, 3 and 4 (respectively).

For males, SEX is coded '1'. As the fifth interval is a 'non-flood' year, FLOOD is '0', and thus the interaction term (SEX.FLOOD) is also '0' (since $(1 \times 0) = 0$).

Therefore,

$$\begin{aligned}\text{logit}(\hat{\varphi}_5) &= \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{FLOOD}) + \beta_4(\texttt{SEX.FLOOD}) \\ &= 0.576929 + (-0.18377) \times (1) + (-0.76269) \times (0) + (0.25935) \times (0) \\ &= 0.393159.\end{aligned}$$

The reciprocal of the *logit* transform (usually referred to as *expit*) is

$$\text{expit}(\hat{\varphi}_5) = \frac{e^{\text{logit}(\varphi_5)}}{1 + e^{\text{logit}(\varphi_5)}}.$$

Thus, the 'reconstituted' value is

$$\hat{\varphi}_5 = \frac{e^{0.393159}}{1 + e^{0.393159}}$$
$$= 0.5970429.$$

This is the result reported by **MARK** (up to the level of the rounding error).

begin sidebar

reconstituting standard error of estimate

In the preceding, we saw how we can 'back-transform' from the estimate of β on the logit scale to an estimate of some parameter θ (e.g., φ or p) on the probability scale (which is bounded on the interval $[0, 1]$). But, we're clearly also interested in an estimate of the variance (precision) of our estimate, on both scales. Your first thought might be to simply back-transform from the link function (in our example, the logit link), to the probability scale, just as we did above. But, does this work?

Consider the male dipper data. Using the logit link, we fit model $\{\varphi_{\cdot}\ p_{\cdot}\}$ to the data – no time-dependence for either parameter. Let's consider only the estimate for $\hat{\varphi}$. The estimate for β for φ from model $\{\varphi_{\cdot}\ p_{\cdot}\}$ is 0.2648275. Thus, our estimate of $\hat{\varphi}$ on the real probability scale is

$$\hat{\varphi} = \frac{e^{0.2648275}}{1 + e^{0.2648275}} = \frac{1.303206}{2.303206} = 0.5658226,$$

which is what **MARK** reports (to within rounding error).

But, what about the variance? Well, if we look at the β estimates, **MARK** reports that the standard error for the estimate of β corresponding to survival is 0.1446688. If we simply back-transform this estimate of the SE from the logit scale to the probability scale, we get

$$\hat{\varphi} = \frac{e^{0.1446688}}{1 + e^{0.1446688}} = \frac{1.155657}{2.155657} = 0.5361043.$$

However, **MARK** reports that the standard error for φ is 0.0355404, which isn't even remotely close to our back-transformed value of 0.5361043.

What has happened? Well, remember (from Chapter 1) that the variance for a parameter is estimated from the likelihood based on the rate of change in the likelihood at the MLE for that parameter (i.e., the second derivative of the likelihood evaluated at the MLE). As such, you can't simply back-transform from the SE on the logit scale to the probability scale, since the different scalings influence the shape of the likelihood surface, and thus the estimate of the SE.

To get around this problem, we can make use of something called the *Delta method*. The Delta method is particularly handy for approximating the variance of transformed variables (and clearly, that is what we are dealing with here). The details underlying the Delta method are beyond our scope at this point (the Delta method is treated in some detail in Appendix B); here we simply demonstrate the application for the purpose of estimating the variance of the back-transformed parameter.

For example, suppose one has an MLE $\hat{\gamma}$ and an estimate of $\widehat{\text{var}}(\hat{\gamma})$, but makes the transformation,

$$\hat{\theta} = e^{\hat{\gamma}^2}.$$

Then, using the Delta method, we can write

$$\widehat{\text{var}}(\hat{\theta}) \approx \left(\frac{\partial \hat{\theta}}{\partial \hat{\gamma}}\right)^2 \times \widehat{\text{var}}(\hat{\gamma}).$$

So, all we need to do is differentiate the transformation function for θ with respect to γ, which yields $2\gamma.e^{\gamma^2}$. We would simply substitute this derivative into our expression for the variance, yielding

$$\widehat{\text{var}}(\hat{\theta}) \approx \left(2\hat{\gamma}.e^{\hat{\gamma}^2}\right)^2 \times \widehat{\text{var}}(\hat{\gamma}).$$

Given values for $\hat{\gamma}$, and $\widehat{\text{var}}(\hat{\gamma})$, you could easily derive the estimate for $\widehat{\text{var}}(\hat{\theta})$.

What about the logit transform? Actually, it's no more difficult, although the derivative is a bit 'uglier'. Since

$$\hat{\varphi} = \frac{e^{\hat{\beta}}}{1 + e^{\hat{\beta}}},$$

then

$$\begin{aligned}
\widehat{\text{var}}(\hat{\varphi}) &\approx \left(\frac{\partial\hat{\varphi}}{\partial\hat{\beta}}\right)^2 \times \widehat{\text{var}}(\hat{\beta}) \\
&= \left(\frac{e^{\hat{\beta}}}{1 + e^{\hat{\beta}}} - \frac{\left(e^{\hat{\beta}}\right)^2}{1 + \left(e^{\hat{\beta}}\right)^2}\right)^2 \times \widehat{\text{var}}(\hat{\beta}) \\
&= \left(\frac{e^{\hat{\beta}}}{\left(1 + e^{\hat{\beta}}\right)^2}\right)^2 \times \widehat{\text{var}}(\hat{\beta}).
\end{aligned}$$

It is worth noting that if

$$\hat{\varphi} = \frac{e^{\hat{\beta}}}{1 + e^{\hat{\beta}}},$$

then it can be easily shown that the derivative of φ with respect to β is:

$$\hat{\varphi}(1 - \hat{\varphi}) = \frac{e^{\hat{\beta}}}{\left(1 + e^{\hat{\beta}}\right)^2}.$$

So, we could rewrite our expression for the variance of $\hat{\varphi}$ conveniently as

$$\begin{aligned}
\widehat{\text{var}}(\hat{\varphi}) &\approx \left(\frac{e^{\hat{\beta}}}{\left(1 + e^{\hat{\beta}}\right)^2}\right)^2 \times \widehat{\text{var}}\left(\hat{\beta}\right) \\
&= \left(\hat{\varphi}(1 - \hat{\varphi})\right)^2 \times \widehat{\text{var}}\left(\hat{\beta}\right).
\end{aligned}$$

From **MARK**, the estimate of the SE for $\hat{\beta}$ was 0.1446688. Thus, the estimate of $\widehat{\text{var}}(\hat{\beta})$ is calculated simply as $(0.1446688)^2 = 0.02092906$. Given the estimate of $\hat{\beta}$ of 0.2648275, we substitute into the preceding expression, which yields

$$\begin{aligned}
\widehat{\text{var}}(\hat{\varphi}) &\approx \left(\frac{e^{\hat{\beta}}}{\left(1 + e^{\hat{\beta}}\right)^2}\right)^2 \times \widehat{\text{var}}\left(\hat{\beta}\right) \\
&= (0.0603525 \times 0.02092906) = 0.001263.
\end{aligned}$$

So, the estimated SE for $\hat{\varphi}$ is $\sqrt{0.001263} = 0.0355404$, which is what is reported by **MARK** (again, within rounding error).

Note: The standard approach to calculating 95% confidence limits for some parameter θ is $\theta \pm (1.96 \times \text{SE})$. However, to guarantee that the calculated 95% CI is $[0, 1]$ bounded for parameters (like φ) that are $[0, 1]$ bounded, **MARK** first calculates the 95% CI on the logit scale, before back-transforming to the real probability scale. However, because the logit transform is not linear, the *reconstituted* 95% CI will not be symmetrical around the parameter estimate, especially for parameters estimated near the $[0, 1]$ boundaries.

end sidebar

We further distinguish between 'reconstituted' parameter estimates and 'free parameters' in the next section. In **MARK**, the output file does not distinguish between 'reconstituted parameters' and 'free parameters'. In fact, **MARK** arguably makes it a bit more difficult to see the correspondence between the number of constrained and free parameters. In **MARK**, all the parameters are printed in sequence – your only clue as to which are 'constrained' and which are 'free' is to look at the link function parameter estimates. This can be somewhat confusing for beginners. Recall that in this example, we applied the constraint to the survival estimates only. Thus, the encounter probabilities were estimated 'the normal way', although their value reflects the influence of the constrained survival estimates – this is why they are not the same as the estimates from the preceding CJS analysis. Got it?

If we follow the classical iterative modeling procedure discussed in earlier chapters, we might proceed to test reduced parameter versions of the 'flood model' by sequentially eliminating various terms in the model. For example, given the structure of the starting model, the first step would be to test for 'significance' of the interaction term. In other words, does the effect of flood on survival differ as a function of the sex of the organism? Recall that this is the prerequisite analysis in either multi-factorial ANOVA or ANCOVA.

We would do this by comparing the following models:

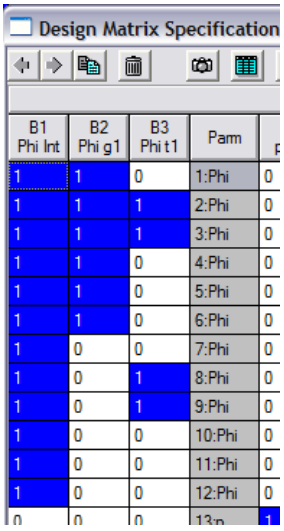
$$\begin{array}{rl} & \varphi = \texttt{SEX} + \texttt{FLOOD} + \texttt{SEX.FLOOD} + \texttt{error} \\ \textit{versus} & \varphi = \texttt{SEX} + \texttt{FLOOD} \qquad\qquad\quad + \texttt{error} \\ & \qquad\qquad\qquad\qquad\qquad\quad \overline{\qquad\qquad\qquad\quad \texttt{SEX.FLOOD}} \end{array}$$

So, from top to bottom, we see that we first fit the 'full model', with both main effects and the interaction. We then follow this by fitting the reduced model without the interaction term. This model is what we refer to as an additive model – something we'll speak more about later in the chapter. The comparison of the fit of these models is a test of the significance of the interaction term.

We've just finished fitting the full model, so we'll proceed directly to fitting the second model – without the interaction term. This analysis is very similar to what we've just done. All we need to do is modify the design matrix. Again, remember we are still applying the constraint to the underlying CJS time-dependent model. In fact, in **MARK**, this is very easy. All you need to do is drop the interaction term (in other words, delete the column containing the dummy variable coding for the interaction term – the fourth column, in our case) from the design matrix. That's it! Isn't that easy?

Go ahead and delete the interaction column from the design matrix (shown at the top of the next page), and then run this reduced parameter model. Name it '`Phi(SEX+FLOOD)p(SEX.TIME)`'. Note we change the syntax from '`SEX.FLOOD`' to '`SEX+FLOOD`' – a fairly standard convention to indication we've dropped the interaction term from the model. The newly modified design matrix (for the survival parameters) should now look like the figure shown below:

Look closely at this new model. First, instead of 3 slopes and 1 intercept, we now only have 2 slopes and one intercept. The slopes correspond to the `SEX` and `FLOOD` terms in our model, respectively. We have 1 fewer slope parameters since we eliminated the interaction term (`SEX.FLOOD`) from the model.



Since we've dropped the interaction term, how many parameters should we have? Well, if we had 16 for the model with the interaction (remember – 15 were originally reported, but we manually adjusted this 'up' to 16 – see above), then we should have (at least in theory) 15 for the model without the interaction (since the interaction term corresponds to 1 link function parameter estimate). Note from the results browser that **MARK** reports 14 parameters – again, because of the fact that **MARK** was unable to correctly estimate p_3 for males, we need to adjust the number of parameters for this model 'up', from 14 to 15.

Next, we examine the 'reconstituted' survival estimates:

dipper analysis
Standard Error and Confidence Intervals Corrected for c-hat = 1.1830000

Real Function Parameters of {phi(sex+flood)p(sex+time+sex*time) - DM}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.6021395	0.0402184	0.5213231	0.6777457
2:Phi	0.4505534	0.0567718	0.3434579	0.5624354
3:Phi	0.4505534	0.0567718	0.3434579	0.5624354
4:Phi	0.6021395	0.0402184	0.5213231	0.6777457
5:Phi	0.6021395	0.0402184	0.5213231	0.6777457
6:Phi	0.6021395	0.0402184	0.5213231	0.6777457
7:Phi	0.6237530	0.0490910	0.5238617	0.7141239
8:Phi	0.4731972	0.0541722	0.3697596	0.5789879
9:Phi	0.4731972	0.0541722	0.3697596	0.5789879
10:Phi	0.6237530	0.0490910	0.5238617	0.7141239
11:Phi	0.6237530	0.0490910	0.5238617	0.7141239
12:Phi	0.6237530	0.0490910	0.5238617	0.7141239

Again, we notice that there are effectively 2 estimates for each sex: one each for the flood or non-flood years (1 → 6 for males, 7 → 12 for females). Notice that the estimates are similar to, but not exactly the same, as the estimates with the full constraint (i.e., including the interaction term). In fact, on the logit scale, we see that the parameters parallel each other – with a constant difference between the males and females for a given year (again, *on the logit scale*).

However, the key question is, is this difference 'large enough' to be 'biologically meaningful'? If we follow the 'classical' paradigm of model testing, we can compare the relative fits of the two models, keeping track of the number of parameters difference between the 2 models. From the results browser, we see that the QAIC_c for the reduced model is 585.01, and for the full model, 586.93. Using the Akaike

weights, the model without the interaction is approximately 2.6 times as well supported as the model with the interaction term – supporting the conclusion that there is no strong support for an interaction of SEX and FLOOD.

The LRT results are consistent with this – the difference in deviance is not statistically significant by usual standards ($\chi^2_1 = 0.242, P > 0.5$). Since the interaction term is not significant, we could next proceed with testing the significance of the main effects: SEX and FLOOD. We can do so easily by using exactly the same process as we just completed above: we modify the constraint to include one of these two remaining terms, and compare the fit. However, while this allows us to test for significance of both terms, we must remember that we will not be able to use LRT to determine if the model containing SEX alone is a better model than one containing FLOOD alone.

Why? Because these are not nested models. Thus, for comparison of these 2 models, we will have to use the $QAIC_c$ comparison approach. Even if the nesting isn't obvious (if it isn't, think about it! We will reconsider 'nestedness' in the context of linear models later in this chapter; see the -sidebar- beginning on p. 41), the necessity of using $QAIC_c$ for these 2 models will be obvious when you compare the number of parameters.* In fact, comparison of non-nested models was one of the original motivations for use of the $QAIC_c$ for model selection (Lebreton *et al*. 1992).

If we were to proceed through each of the 'nested' models, we would see that the model where survival is constrained to be a function of FLOOD alone is the most parsimonious model, and is approximately 2.8 times better supported by the data than the next best model (SEX+FLOOD). No other model in the model set is adequately supported by the data. In other words, we conclude that there is no support for a 'significant' difference between the sexes, and that flooding significantly influences variation in survival.

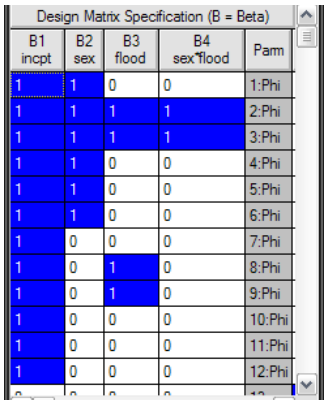
Does this mean that this is our best model overall? The short answer to this question is 'no'. What we have done is simply to test a set of hypotheses under specific conditions. What were our main conditions? The conditions in this example were the use of the CJS model structure for both survival and encounter, prior to adding the constraints. The remaining question is – would we have come up with a different result if we had made the encounter probability constant? What if we had left time-dependence in encounter probability, but used the same parameter values between the sexes? How does our current model compare to one where survival is assumed to be constant?

Where we go from here, then, very much depends upon what we're after. We have to keep in mind the various purposes of model testing. At one level, we are seeking to test specific biological hypotheses. At the other, we may also be trying to find the most parsimonious model, which will provide us with the most precise and least biased estimates for modeling purposes.

Again, our recommended strategy is to use the process of model selection over a set of candidate models to identify the most parsimonious acceptable model containing the effect(s) that you want to test, and then proceed to use LRT or $QAIC_c$ to compare this model with reduced parameter models where one or more of these terms have been eliminated. Remember, by 'acceptable' we mean a model which fits the data (Chapter 5). In fact, we can't emphasize this enough – the first step in analyzing your data must be to ensure that your most general model (fully time-dependent CJS, $\{\varphi_t\, p_t\}$, for example) adequately fits the data.

How would we derive the design matrix for the next 2 models – {SEX} and {FLOOD}? We can do this easily in **MARK**, using one of a couple of different approaches. The most intuitive approach is to simply modify the design matrix manually. Start with the design matrix for the most general model, {SEX+FLOOD+SEX.FLOOD}, shown at the top of the next page.

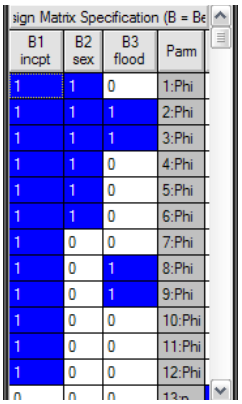
* If it isn't obvious – the two models have the same number of parameters, so the degrees of freedom for a putative LRT would be 0, which of course makes no sense.



Design Matrix Specification (B = Beta)

B1 incpt	B2 sex	B3 flood	B4 sex*flood	Parm
1	1	0	0	1:Phi
1	1	1	1	2:Phi
1	1	1	1	3:Phi
1	1	0	0	4:Phi
1	1	0	0	5:Phi
1	1	0	0	6:Phi
1	0	0	0	7:Phi
1	0	1	0	8:Phi
1	0	1	0	9:Phi
1	0	0	0	10:Phi
1	0	0	0	11:Phi
1	0	0	0	12:Phi

To create model {SEX+FLOOD}, we simply take the design matrix for {SEX+FLOOD+SEX.FLOOD} (above), and delete the column corresponding to the interaction term:



ign Matrix Specification (B = Be

B1 incpt	B2 sex	B3 flood	Parm
1	1	0	1:Phi
1	1	1	2:Phi
1	1	1	3:Phi
1	1	0	4:Phi
1	1	0	5:Phi
1	1	0	6:Phi
1	0	0	7:Phi
1	0	1	8:Phi
1	0	1	9:Phi
1	0	0	10:Phi
1	0	0	11:Phi
1	0	0	12:Phi

Using the same approach, to fit model {SEX} all we'd need to do is take the design matrix for {SEX+FLOOD} and drop the FLOOD column, leaving {SEX}. But, once we've dropped the FLOOD column, how can we go from the design matrix for {SEX} back to {FLOOD}?

Do we have to manually re-enter the appropriate dummy-variable coding? No! In **MARK**, all you need to do is click on any 'more saturated' model containing the terms you want in the results browser, and then 'retrieve' its design matrix. For example, if we want to fit model {FLOOD}, simply (a) click on the model {SEX+FLOOD} in the browser (this model is more saturated because it contains the factor of interest – FLOOD – plus one or more other terms), then (b), pull down the '**Retrieve**' menu and retrieve the '**Current model**'. This causes **MARK** to extract the design matrix for the model you've selected. Once you have this design matrix (in this case, corresponding to model {SEX+FLOOD}), all you need to do is delete the SEX column to yield the design matrix for model {FLOOD}. Pretty slick!

6.5.1. Subset models and the design matrix

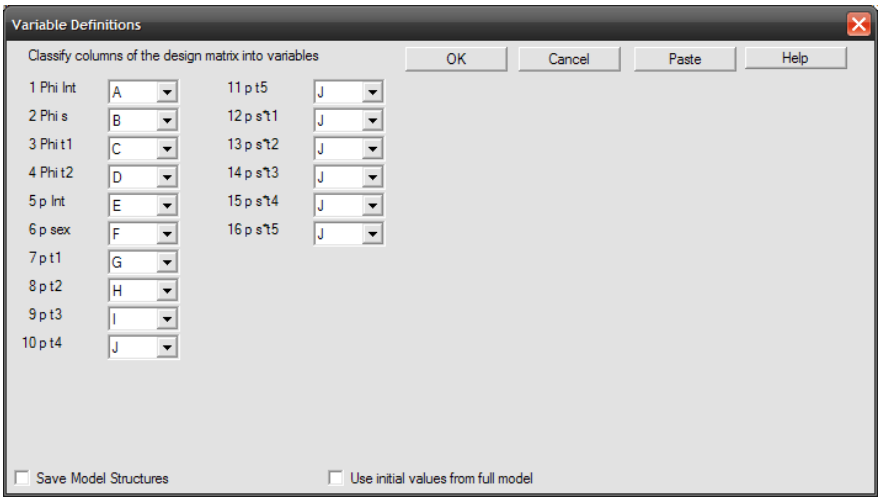
In the preceding example, we used the full design matrix from our general model $\{\varphi_{sex.flood}\, p_{sex.time}\}$, and built the reduced parameter models by deleting one or more columns from this design matrix. All of the remaining models in the candidate model set were nested within the general model.

While this is relatively straightforward to do 'by hand' for simple models, it can quickly become

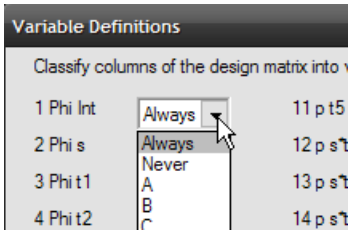
tedious for more complicated analysis, where the design matrices can be very large. **MARK** has an option referred to as 'subset models' to automate much of the process of constructing various nested models from the design matrix for some more general model.

As the first step, you first construct the design matrix for the full model which contains all the variables (i.e., columns in the design matrix of interest). Note that the full model may not have actually been run, i.e., the saved structure of the full model can be used to construct the subset models. Then, using the 'subset models' feature, you simply select the columns from the full model design matrix to be used in a nested set of models.

To start the process, in the '**`Results Browser Window`**', highlight (retrieve) the model with the design matrix that contains all the variables that you want to use in all possible subset model combinations – in this case, model $\{\varphi_{s.f}\ p_{s.t}\}$. Then, select the menu choices '**`Run | Subset of DM Models`**' to create and run all possible models. This will bring up the interactive interface window, below:



The interactive interface gives you a list of all the columns in the design matrix for the selected model. If you pull down any of the drop-down menus shown beside each parameter, you will see a set of options: '**`Always`**', '**`Never`**', followed by the letters **`A`** → **`J`**.



The 'subsetting' of different variables (columns) in the design matrix is accomplished by selecting elements from these drop-down lists for each parameter. Some of these options are relatively self-explanatory. For example, columns in the design matrix corresponding with a model 'intercept' (which we've designated in the label) we will generally keep in all models in the candidate model set, and so would be assigned the value '**`Always`**' to designate this status. For parameters (columns) which will never show up in any of the other models in the candidate model set, you would select the value '**`Never`**'.

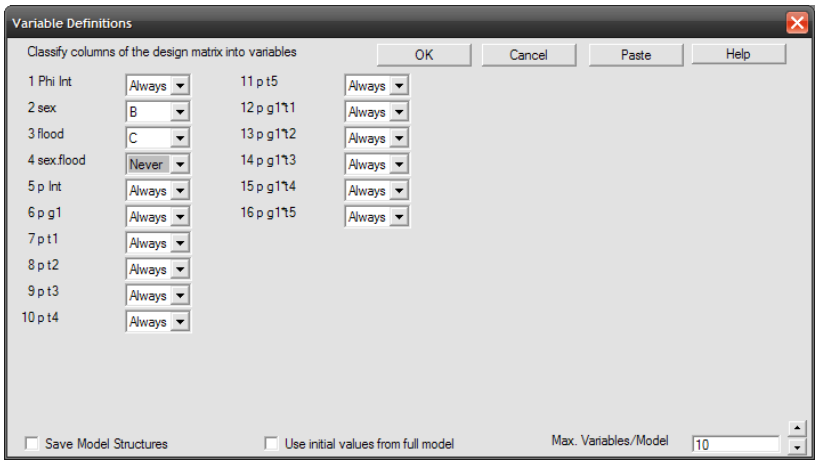
The meaning and use of the values **A** → **J** requires a bit more explanation. Columns in the models which are neither '**always present**' or '**never present**' will be given values of **A**, **B**,...,**J**, designating up to 10 variables, either singly or jointly. Currently, the upper limit is set to 10 variables, which produces 2^{10} = 1,024 possible models. If you think you need to run more that 1,024 models, we suggest you think harder about the list of variables you are considering!

The letters **A** → **J** identify how columns in the design matrix are related to each other and how they will be combined in subsets of models. So, as an example, suppose as in the dipper example you have 4 columns for apparent survival (φ) (the `intercept` column, the `sex` column, the `flood` column, and the (`sex.flood`) interaction column), and 12 columns for the encounter probability (p) (the `intercept` and `sex` columns, the 5 columns for time, and the 5 columns representing the interaction of (`sex.time`)). In our candidate model set, we focus only on a series of 4 nested models for apparent survival: $\{\varphi_{sex+flood}\}, \{\varphi_{flood}\}, \{\varphi_{sex}\}, \{\varphi_{.}\}$. Recall that our general model is $\{\varphi_{sex.flood}\}$. Thus, the structure for our encounter probability $\{p_{sex.time}\}$ occurs in every model – in other words, it is 'always' there. So, as a first step, we simply select the value '**Always**' for the encounter parameters (columns).

Now, for the apparent survival parameters. Clearly, the `intercept` is in each model, so we select the value '**Always**' for the intercept. The variables `sex` and `flood` are 'stand alone' variables – i.e., each defines a category with a single column in the design matrix. Each variable could appear alone in the model, so each is assigned its own letter. We'll use `A` for `sex`, and `B` for `flood` (it doesn't much matter which letters you use, so long as they are different).

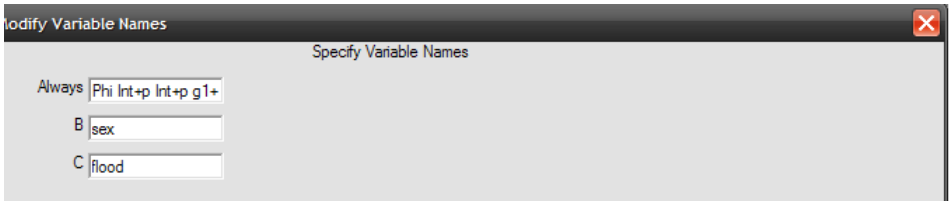
What about the interaction term (`sex.flood`)? We need to use the value '**Never**', for two reasons: first, because the interaction model cannot enter the model without the two interacting variables also included in the model (i.e., model `Y = A + B + A.B` is valid, but `Y = A.B` is not), and moreover, there is no direct way to conditionally subset columns (i.e., select `C` if only `A` and `B` are present; discussed in more detail below). But second, and perhaps most obviously, we don't need the interaction column because that column is already present in the general model we started with. Think about it for a moment.

Here (below) is what the variable definition screen should look like so far:



Now, before running the models, note the additional options which are available at the bottom of the model specification screen. Instead of running the models immediately, the model structure can be saved and then all of the models run later in batch mode. The second option is to use the β estimates from the full model as initial values for the subset models. However, these estimates may not be great, depending on the collinearity among the variables. Note that this option does not appear if the full model has not actually been run (i.e., only the saved structure is used to specify the full model).

After clicking '**OK**', another window will pop up asking if you want to change (specify) variable names:



The naming of models fit using the '**subset models**' approach defaults to listing all of the columns which were 'always' included in the model. This can often result in very long model names (as we will see). This windows gives you the opportunity of overriding the default naming convention, but that places the burden of responsibility on you to remember which columns were included in your models.

Once you hit the '**OK**' button, a little popup window will inform you that you're going to run 4 models. But, before you simply click the '**OK**' button – think a bit. What are the 4 models? If you understand what we just did (above), then you'll know that the 4 models (in terms of φ) are $\{\varphi_{s+f}\}, \{\varphi_s\}, \{\varphi f\}, \{\varphi_.\}$.

Here is the results browser, showing the general model and each of the models in the candidate model set fit by manually modifying the design matrix, and the 4 fit using the '**subset models**' approach:

Model	AICc	Deviance
{Phi Int+p Int+p g1+p t1+p t2+p t3+p t4+p t5+p g1t1+p g1t2+p g1t3+p g1t4+p g1t5+flood}	682.1585	72.7979
{flood}	682.1585	72.7979
{Phi Int+p Int+p g1+p t1+p t2+p t3+p t4+p t5+p g1t1+p g1t2+p g1t3+p g1t4+p g1t5+sex+flood}	684.2119	72.7128
{sex+flood}	684.2119	72.7128
{Phi Int+p Int+p g1+p t1+p t2+p t3+p t4+p t5+p g1t1+p g1t2+p g1t3+p g1t4+p g1t5}	684.8886	79.7738
{.}	684.8886	79.7738
{sex.flood}	686.0740	72.4262
{Phi Int+p Int+p g1+p t1+p t2+p t3+p t4+p t5+p g1t1+p g1t2+p g1t3+p g1t4+p g1t5+sex}	687.0051	79.7726
{sex}	687.0051	79.7726

As noted earlier, the models fit using the '**subset models**' approach have very long model names – the naming syntax explicitly indicates which columns were included in the model. In particular, pay attention to the right-hand side of the default model names – this is where the 'variable' columns that are included in a given model are indicated. For example, model

```
{Phi  Int+p  Int+p  g1+p  t1+p  t2+p  t3+p  t4+p  t5+p
g1*t1+p  g1*t2+p  g1*t3+p  g1*t4+p  g1*t5+sex+flood}
```

The last two terms (i.e., `sex+flood`) indicate that this model is the additive $\{\varphi_{s+f}\}$ model.

More importantly, notice that the model deviance values are identical for a given model regardless of whether or not it was generated by modifying the design matrix manually, or using the 'subset models' approach.

The preceding analysis was very simple, and the cost savings for using the 'subset models' approach might not be particularly significant in this instance. But, that will generally not be the case.

A common issue alluded to earlier occurs when you only want to include a particular column when another column is included in the model. An example would be for linear trend (`T`) and the associated quadratic trend (`TT`). As an example, suppose that there are two additional variables, age and gender.

One approach to only including 'TT' when 'T' is in the model is to do 2 sets of models. For the first set, only the 'T' variable would be used:

```
intercept   Always
age         A
gender      B
T           C
TT          Never
```

Then, a second set of models are constructed to always include 'TT' with 'T':

```
intercept   Always
age         A
gender      B
T           C
TT          C
```

Each set would produce 8 models, for a total of 16. However, the user will have 4 sets of duplicates when neither 'T' or 'TT' are included (sorting the list of models by model name may help find the duplicates):

```
intercept
intercept + age
intercept + gender
intercept + age + gender
```

A second issue is that the user never wants 2 particular variables in the model at the same time. Suppose this is the case for length and weight. Again, a simple solution is to run 2 sets of models, specifying the '**Never**' key word first for length, and then for weight. Again, some duplicate models will have to be removed.

begin sidebar

subset models and 'importance of a factor' – a mechanical shortcut

As introduced in Chapter 4 (section 4.6.3), assessment of the relative importance of variables has often been based only on the best model (e.g., often selected using a stepwise testing procedure of some sort). Variables in that best model are considered 'important', while excluded variables are considered 'not important'. Burnham & Anderson have suggested that this approach is too simplistic. Importance of a variable can be refined by making inference from all the models in the candidate set. Akaike weights are summed for all models containing predictor variable (i.e., factor) x_j, $j = 1, \ldots, R$. Denote these sums as $w_{+(j)}$. The predictor variable with the largest predictor weight, $w_{+(j)}$, is estimated to be the most important, while the variable with the smallest sum is estimated to be the least important predictor.

As suggested by Anderson & Burnham, summing support over models is regarded as superior to making inferences concerning the relative importance of variables based only on the best model. This is particularly important when the second or third best model is nearly as well supported as the best model or when all models have nearly equal support.

The robustness of the use of cumulative AIC weights appears to be strongly conditional on the 'symmetry' of the candidate models set. A 'symmetrical' model set is one which has roughly the same number of models with, and without a particular factor. While this can sometimes be difficult to accomplish, especially for models involving interaction terms, for models where the factors of interest

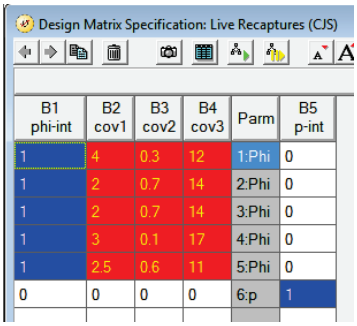
are entered into the models as independent factors (covariates), the '**Subset of DM models**' option makes generating symmetrical model sets straightforward. In additional, there is an option in **MARK** to automatically calculate the cumulative AIC weights over models built using the '**Subset of DM models**' approach.

We'll demonstrate this approach using some simulated data contained in **var-importance.inp**. The data consist of live encounter data, 6 occasions. Time-varying apparent survival φ, and constant encounter probability, p. We're interested in the relative 'importance' of 3 different environmental covariates (`cov1`, `cov2`, `cov3`).

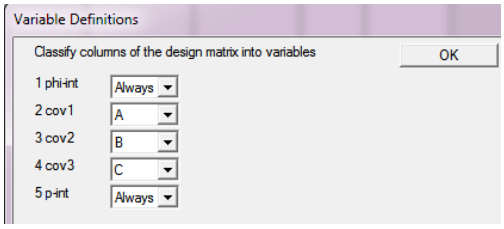
Here are the covariate values corresponding to each interval:

	1	2	3	4	5
cov1	4	2	2	3	2.5
cov2	0.3	0.7	0.7	0.1	0.6
cov3	12	14	14	17	11

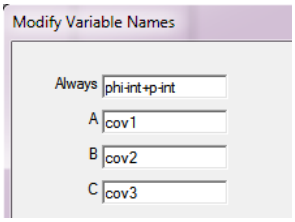
Here is the DM corresponding to the most 'general' model, containing all 3 covariates:



We'll go ahead and run this model – give it a temporary name like '`hold`', or some such. Now, we'll use the '**Subset of DM models**' option to build a model set symmetrical for all three environmental covariates. As shown below, we want to 'always' include the intercepts for φ and p. We have three covariates (`cov1`,`cov2`,`cov3`), which we'll label `A`, `B`, `C`, respectively.



We will accept the default variable names, as shown below:



Once you click the '**OK**' button, **MARK** will pop-up a window informing us that we have specified 8 models. With a bit of thought, you'll see that this set of 8 models consists of

- 3 models containing a single covariate (`cov1`, `cov2` or `cov3`),
- 3 models consisting of 2 of the factors (`cov1+cov2`, `cov1+cov3`, or `cov2+cov3`),
- 1 model consisting of all 3 factors (`cov1+cov2+cov3`)
- 1 model not containing any of the 3 factors (i.e., intercepts only)

Go ahead and run the 8 models – the results are automatically added to the results browser. Note that the 8 models in the browser match the structures we anticipated, above.

Before we interpret the results in the browser, notice that the highlighted model (which contains all 3 covariates) is redundant to the model right below it – the one we started with which we called '`hold`'. For the '**`Subset of DM models`**' option, **MARK** needs a model to 'start from' (model '`hold`', in this example), but now that we've run the models, we no longer need it.

Go ahead and delete the model named '`hold`' from the browser.

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi-int+p-int+cov2+cov3}	1320.8008	0.0000	0.29102	1.0000	4	50.8458
{phi-int+p-int+cov2}	1321.8029	1.0021	0.17633	0.6059	3	53.8692
{phi-int+p-int+cov1+cov2}	1321.8103	1.0095	0.17568	0.6037	4	51.8552
{phi-int+p-int+cov1}	1321.8183	1.0175	0.17498	0.6013	3	53.8846
{phi-int+p-int+cov1+cov2+cov3}	1322.8123	2.0115	0.10645	0.3658	5	50.8304
{phi-int+p-int+cov1+cov3}	1323.4989	2.6981	0.07552	0.2595	4	53.5439
{phi-int+p-int}	1339.7163	18.9155	0.00002	0.0001	2	73.7988
{phi-int+p-int+cov3}	1341.4111	20.6103	0.00001	0.0000	3	73.4775

The most parsimonious model in the candidate model set contains both `cov2` and `cov3`, but the support for this model is only marginally greater than for the second-best model, which contains `cov2` only. This is a good example where model selection uncertainty makes it somewhat more difficult to establish the relative importance of the 3 different covariates.

We can calculate cumulative AIC weights in **MARK** by selecting '**`Run | Variable weights`**', which outputs the weights to both the editor (shown below), and an Excel spreadsheet:

```
=====  Variable       Weight  Count
=====  -------------  ------  -----
=====  phi-int+p-int  1.0000      8
=====  cov1           0.5326      4
=====  cov2           0.7495      4
=====  cov3           0.4730      4
```

We see clearly that `cov2` is 'more important', relative to the other 2 covariates.

end sidebar

6.6. Some additional design matrix tricks

In a moment, we'll continue with the next 'analytical question' we might be interested in – are the differences between flood and non-flood significant? Is there an interaction of `flood`, `sex` and `survival`? And so on. For the moment, though, let's consider a couple of the 'mechanical' aspects of modifying the design matrix which are worth knowing. In the preceding, we modified the design matrix manually. As

you probably realize by now, **MARK** often gives you more than one way to do things (this is generally a good thing!). What could we have done other than manually editing the design matrix? A perhaps more elegant, and (with some practice) faster way, is using some of **MARK**s nifty menu options.

For example, pull down the '**FillCol**' menu. You'll see two intercept options – the '**Intercept**' itself, and something called the '**Partial Intercept**'. Since in our example we only wanted to modify 'part' of the design matrix, we would select '**Partial Intercept**'. This causes **MARK** to spawn a window asking you the number of rows in the design matrix you want to add the intercept coding to. In this example, with 12 rows (corresponding to the 12 survival parameters – 6 for males, 6 for females), we would have responded with '12'.

Anything else we could do with the '**FillCol**' menu? Well, you might notice that there is an option to specify a '**Group Effect**' item on the menu. If you select this option, another child menu will pop up, giving you several options for the kind of group effect you want to code (broadly dichotomized into discrete or continuous). Now, within the discrete or continuous groupings, you'll see options for '**partial**'.

What does this mean? Well, '**partial**' simply means we want whatever it is we're going to do to be applied only to 'part' of the design matrix. Since in our example we're only interested in the first 12 parameters, corresponding to the first 12 rows and columns, we clearly would want 'partial' – a piece of the whole matrix. If we select '**Partial Discrete**', **MARK** would immediately fill in the first column of the design matrix, with 6 '1's followed by 6 '0's. **MARK** is clever enough to remember that (a) you have 2 groups, and (b) that there are 7 occasions (and therefore 6 parameters) for each group. In fact, it 'learned' this when you filled in the model specification window for this analysis. Pretty slick, eh? At any rate, go ahead and 'play around' a bit with the various menus available for modifying the design matrix.

One last thing – remember at the beginning of our example – we started with the '**Full**' design matrix? Recall that there were 2 other options in the '**Design Matrix**' menu - '**Reduced**', and '**Identity**'. The '**Reduced**' option allows you to tell **MARK** exactly how many columns to put into the design matrix – this can be useful (and can save you some time) *if you know how many columns you need* – which you might if you've carefully thought through the linear model, and corresponding design matrix, for your analysis. The '**Identity**' matrix is simply a matrix of the same dimension as the '**Full**' design matrix, but with '1's along the diagonal. Some people prefer modifying the identity matrix, since most of the matrix elements are '0's – fewer cells to modify. Pick whichever approach works best for you.

6.7. Design matrix...or PIMs?

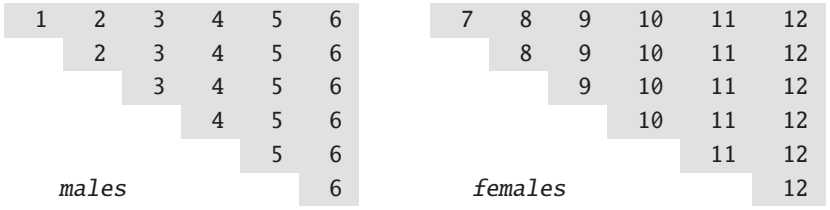
In the preceding, we fit the 'flood model' to the European dipper data by modifying the design matrix, changing it from the default identity matrix. Remember, the design matrix reflects the underlying structure of the model, which is *specified* by the PIMs. When we modify the design matrix, for a given set of PIMs, we're applying a *constraint* to the model specified by those PIMs. We are not changing the parameter structure at all (that you can only do by modifying the PIMs) – we are constraining the estimates from that parameter structure to be a function of the linear model we specify in the design matrix.

This is a very important point. So important, if fact, that we're now going to force you to think about it carefully, by pointing out that we could have fit a 'flood' model to the dipper data without using a linear model at all – simply by using a different set of PIMs!

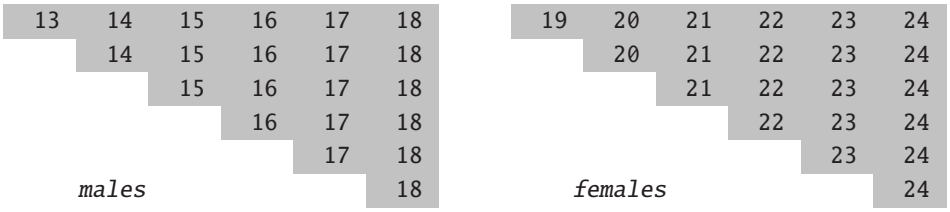
How? Recall that our original starting model for the dipper data was the fully time-dependent CJS model with two groups (the two sexes, males and females) – $\{\varphi_{s.t}\, p_{s.t}\}$.

There were 7 occasions in the data set, so the PIMs reflecting this starting model were

survival



encounter



These PIMs specified the model that we then constrained – they indicate 6 survival parameters for each sex, which we then constrained to be a linear function of flood. Remember, the actual linear model we fit initially was:

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

Now...is there any way we could have fit this model without modifying the design matrix? The answer is...'yes'. How? A couple of hints: PIM's, and the fact that 'flood' is a binary state variable.

Remember, a year is classified as either a 'flood' year, or a 'non-flood' year. In our original survival PIMs, we had 6 parameters for each sex, respectively, which allowed survival to vary among years. However, if we want survival to vary only as a function of whether or not a year is a 'flood year', then in fact we need only 2 parameters for each sex!

Thus, we could have specified model

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

using the PIMs shown below:

survival

What would the encounter PIMs look like?

Well, if we use model $\{p_{g*t}\}$ for encounter, the PIMs for the encounter parameters would look like:

encounter

5	6	7	8	9	10
	6	7	8	9	10
		7	8	9	10
			8	9	10
				9	10
males					10

11	12	13	14	15	16
	12	13	14	15	16
		13	14	15	16
			14	15	16
				15	16
females					16

Try it! You'll see that you get the same results as you did when you modified the design matrix – **IF** you specify the logit link function. In fact, one advantage of this approach is that you can use whatever link function you want. Note also that the PIMs specify 16 parameters, which is exactly the same number of parameters as there were in the design matrix approach (although only 15 of them were estimable using the logit link – if you run this PIM-based model using the sin link, all 16 parameters are estimated, and reported as estimated by **MARK**).

Pretty slick, eh? Make sure you understand this. If not, go through it again. Now, if this approach works, why not use it all the time? The reason is, because it has limited flexibility in the ability to specify certain kinds of models. You can't use this approach to build additive models (something we'll address later), or to constrain estimates to follow a trend (the next section). There are limits – but it is useful to know that you can sometimes 'get where you need to go' by modifying the PIMs directly, rather than using the design matrix. This is important both conceptually, and practically, in some cases. However, we generally recommend that you use the design matrix approach the majority of the time, since it gives you the greatest flexibility. In fact, in many cases, the design matrix is your only option – the additive (`SEX+FLOOD`) model is a good example of this. There is **no** way you could build this additive model using PIMs alone.

So, as a general recommendation, we suggest the following sequence (which we'll revisit again in this chapter, and elsewhere in the remainder of the book):

step 1 - *Build your general model using PIMs, if you can. Recall that models with multiple groups based on PIMs assume interaction effects among levels of your groups. This is often – but not always – the basis for the general model in your candidate model set,*

step 2 - *Once you have the general model constructed using PIMs, then try to build the exact same model using the design matrix approach. Run this model. You'll know your design matrix is correct if the deviance for the model fit with the design matrix approach is the same as the deviance for the same model constructed using PIMs (note: the deviances should be the same – but the number of estimated parameters **MARK** reports might differ due to differences in the link function used),*

step 3 - *Once you've successfully built your general model with the design matrix, delete the same general model you built with PIMs – you don't want two copies of the same model in your results browser. Then, build all other models in the candidate model set by manipulating the design matrix of your general model.*

These 3 steps are good, general recommendations on where to start. Once you've built your general model using the design matrix, you can quickly and easily construct reduced parameter models – including models with additive effects – simply by manipulating the design matrix. This invariably

means deleting or modifying one or more columns in the design matrix. Once you get the hang of this approach, it will become fairly automatic to you.

6.8. Constraining with 'real' covariates

In the previous sections, we've considered variation in one parameter or another over time – implicitly, we've been treating time as a 'classification' variable (or 'factor'), and looking for heterogeneity among 'time intervals' in a particular parameter. Generally, though, we're not interested in whether or not there is variation over time, but whether this variation over time corresponds to one or more 'other variables' (covariates) which we think might cause (or contribute to) the variation we observe. In other words, our interest is typically in the causes of the temporal variation, not the variation itself.

We can address this hypothesis (i.e., that variation in some parameter over time reflects variation in some covariate) by building a linear model where the parameter estimates are constrained to be linear functions of one or more covariates. This is the subject of this section – constraining parameters to be functions of 'real' variables (in the mathematical sense of real), as opposed to simple 'dummy' or other integer variables.

For example, suppose we have measured some other variable, such as total precipitation, or measures of annual food abundance, which can take on 'real' or 'fractional' values. Clearly, we might want to test the hypothesis that a model where one or both parameters are constrained to be linear functions of this type of covariate might be extremely useful. Fortunately, we have to learn nothing new in order to do this in **MARK** – all we need to do is put our 'real' covariates into our design matrix.

Consider the following example. Suppose you believe that capture rate is a function of the number of hours spent by observers in the field. This makes good intuitive sense – the more hours spent in the field, the more likely you might be to see a marked individual given that it is still alive. So, one way we might increase the precision of our estimate of encounter probability is to constrain the encounter parameters to be linear functions of number of observations hours at each occasion. Recall that parsimonious modeling of encounter probability will also influence our estimates of survival probability as well (translation: you might be more interested in estimating survival, but you won't be as successful unless you also do a good job modeling the encounter probability).

These data are unavailable for the dipper data set, so we'll 'make up' some 'observation hours' covariate data, just for purposes of illustrating this. Here are our covariate 'data':

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

Now, we simply need to construct the correct design matrix. One slight twist here is that for the first time we're going to apply a constraint to the encounter probabilities, rather than the survival probabilities.

This is no more difficult than what we've already done – all you need to do is identify the index values of the parameters you want to constrain. Recall that for the dipper data set, with 2 sexes and 7 occasions, parameters $1 \rightarrow 6$ are male survival, parameters $7 \rightarrow 12$ are female survival, parameters $13 \rightarrow 18$ are male encounter probability, and finally, parameters $19 \rightarrow 24$ are female encounter probability. Thus, if we want to constrain our encounter probabilities to be linear functions of observer hours, we're going to constrain parameters $13 \rightarrow 24$. This means that we're working in the lower-right quadrant of the design matrix.

Next, we need to decide on the model we want to test. Let's test the model where we allow the sexes to potentially differ, with full interaction.

In other words,

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

As you can see, it is exactly the same qualitative model 'structure' as in our earlier 'flood' example, with a different 'covariate' (HOURS instead of FLOOD). Given this similarity, you might guess the design matrix should look similar as well.

In fact, as shown below, it is virtually an inverted mirror image of the design matrix you used for the 'flood' analysis – now the upper-left quadrant has the time-specific coding we've seen before, and the lower-right quadrant (pictured) has the dummy variable coding for INTCPT, SEX, HOURS, and the SEX.HOURS interaction.

13:p	1	1	12.1	12.1
14:p	1	1	6.03	6.03
15:p	1	1	9.1	9.1
16:p	1	1	14.7	14.7
17:p	1	1	18.02	18.02
18:p	1	1	12.12	12.12
19:p	1	0	12.1	0
20:p	1	0	6.03	0
21:p	1	0	9.1	0
22:p	1	0	14.7	0
23:p	1	0	18.02	0
24:p	1	0	12.12	0

The only real difference is that instead of '0' or '1' to represent 'FLOOD' states, we replace that column of '0' and '1' values with the 'real' number of observer hours. And, since these are simply different levels of a single factor ('HOURS'), we need only one column to code for 'HOURS'.

However, as you'll remember from our earlier examples, if you change either of the 2 columns contained in the interaction term, you also need to change the values in the interaction term itself. The encounter portion of the design matrix (i.e., the lower-right quadrant) is shown above. Note we still have 12 rows, and 4 columns for the encounter parameter (reflecting the number of variables in the constraint). Once you have the design matrix constructed, you proceed in precisely the same fashion as you did with the 'flood' example we just covered – the only difference is that, in this example, you're concentrating on encounter probabilities, rather than survival.

begin sidebar

linear covariates and nested models: LRT revisited

Recall from Chapter 4 that the classical LRT requires that models be *nested*. What constitutes 'nested' in the case of models with one or more linear covariates? What are the options if models are not strictly nested?

In Chapter 4, we defined nested models as follows:

> **nested models**: *Two models are nested if one model can be reduced to the other model by imposing a set of linear restrictions on the vector of parameters.*

For example, consider models f and g, which we'll assume have the same functional form and error structure. For convenience, we'll express the data as deviations from their means (doing so eliminates the intercept from the linear model, since it would be estimated to be 0). These two models differ then only in terms of their regressors.

In the following

$$f: Y = \beta_1 x_1 + \epsilon_0$$
$$g: Y = \beta_1 x_1 + \beta_2 x_2 + \epsilon_1,$$

the model f is nested within model g because by imposing the linear restriction that $\beta_2 = 0$, model g becomes model f.

OK, what about the situation we're considering here – linear models with one or more linear covariates? Consider the following linear model for some parameter θ corresponding to a 5 occasion study:

$$\theta = \beta_0 + \beta_1(\texttt{interval}_1) + \beta_2(\texttt{interval}_2) + \beta_3(\texttt{interval}_3)$$

Remember: 5 occasions = 4 intervals = $(4-1) = 3$ columns of dummy variables coding for the intervals ($\beta_1 \rightarrow \beta_3$).

Here is the design matrix (DM) corresponding to this time-dependent linear model:

intcpt	β_1	β_2	β_3
1	1	0	0
1	0	1	0
1	0	0	1
1	0	0	0

Now, suppose we wanted to constrain this model such that the estimate of the parameter in the first and third intervals was equal. How would we modify the DM to achieve this constraint? The key is remembering what each β_i column represents: β_1 represents the first interval (between occasion 1 and 2), β_2 represents the second interval (between occasion 2 and 3), and so on.

So, to constrain the estimates for $\hat{\theta}$ to be the same for the first and third intervals (i.e., $\theta_1 = \theta_3$), we have to (i) eliminate one of the two columns corresponding to these intervals (either the β_1 or β_3 columns), and (ii) add a '1' dummy variable in the appropriate row to the remaining column.

For example, in the following DM:

intcpt	β_1	β_2
1	1	0
1	0	1
1	1	0
1	0	0

we have eliminated the β_3 column from the original DM, and added a dummy '1' in the 3rd row of column β_1 – recall that row 3 corresponds to interval 3. The presence of a '1' in the first and third rows in the β_1 column is what constrains $\theta_1 = \theta_3$. This is essentially the same sort of thing we did for the flood example we considered earlier in this chapter. We have constrained our time-dependent model in a particular way – using the linear constraint $\theta_1 = \theta_3$.

Similarly, what if we want to constrain $\hat{\theta}$ to be a linear function of some continuous covariate (say, rainfall). Our DM might now look like

intcpt	β_1
1	2.3
1	4
1	1.2
1	5

Here, we've constrained the estimates for each interval to be a linear function of the rainfall covariate – one β column. So, based on the criterion for 'nestedness' – where two models are nested if one model can be reduced to the other model by imposing a set of linear restrictions on the vector of parameters – these two constrained models we've just constructed are both nested within the more general time-dependent model. And, as such, these models could both be compared to the time-dependent model using an LRT.

end sidebar

6.8.1. Reconstituting estimates using real covariates

Here, we continue with our example analysis, using the full dipper data (i.e., including both males and females), where detection probability is being modeled as a linear function of the number of observation hours. For purposes of demonstrating how to reconstitute parameter estimates on the real probability scale, we'll first fit the following linear model to the dipper data:

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

As noted earlier, it is exactly the same qualitative model 'structure' as in our earlier 'flood' example, with a different 'covariate' ('HOURS' instead of 'FLOOD'). For present purposes, we'll assume simple time-dependence for survival, with no sex differences. So, our overall model is $\{\varphi_t\, p_{S+H+S.H}\}$.

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
```

After fitting our model to the data, we see that the reconstituted estimate for p for males (first group), third encounter occasion, is 0.9373487. Where did this value come from?

As with the earlier 'flood' example, we'll need the parameterized linear equation for p on the logit scale. If we look at the β estimates for p, we see that the parameterized linear model is

$$\begin{aligned}\text{logit}(\hat{p}) &= \hat{\beta}_1 + \hat{\beta}_2(\texttt{SEX}) + \hat{\beta}_3(\texttt{HOURS}) + \hat{\beta}_4(\texttt{SEX.HOURS}) \\ &= 1.4116023 + 1.4866125(\texttt{SEX}) + 0.0463456(\texttt{HOURS}) + (-0.0783096)(\texttt{SEX.HOURS}).\end{aligned}$$

The coding for males ('SEX') is '1'. The 'HOURS' covariate for the third encounter occasion is 6.03. The interaction of the two, 'SEX.HOURS', is simply (1)(6.03) = 6.03.

So,

$$\begin{aligned}\text{logit}(p_{m,3}) &= 1.4116023 + 1.4866125(1) + 0.0463456(\texttt{6.03}) + (-0.0783096)(\texttt{6.03}) \\ &= 2.70547188.\end{aligned}$$

So, back-transforming

$$\hat{p}_{m,3} = \frac{e^{2.70547188}}{1 + e^{2.70547188}} = 0.9373487,$$

which is what is reported by **MARK**.

6.8.2. Plotting the functional form – real covariates

In the preceding example, we modeled encounter probability as a linear function of the number of observation hours. But, what is the actual relationship between 'encounter probability' and 'observation hours'? If you look at the parameter estimate for $\hat{\beta}_3 = 0.0463456$, the interpretation seems easy enough – the estimated slope is positive, meaning that as the number of hours of observation increases, so does the probability of detection. Which of course, seems somewhat intuitive.

But, suppose you decide to go ahead and 'plot the relationship'. In principle, this is straightforward. First, you have to decide whether you want to plot the relationship for males, or females. Let's assume our interest is in males – recall that the dummy coding for males is '1'. Then, you simply need to derive the estimate of logit(p) for males, over a range of hours (say, from $5 \rightarrow 20$), and then back-transform from the logit scale to the real probability scale.

Given our estimated linear model,

$$\text{logit}(\hat{p}) = 1.4116023 + 1.4866125(\texttt{SEX}) + 0.0463456(\texttt{HOURS}) + (-0.0783096)(\texttt{SEX.HOURS}),$$

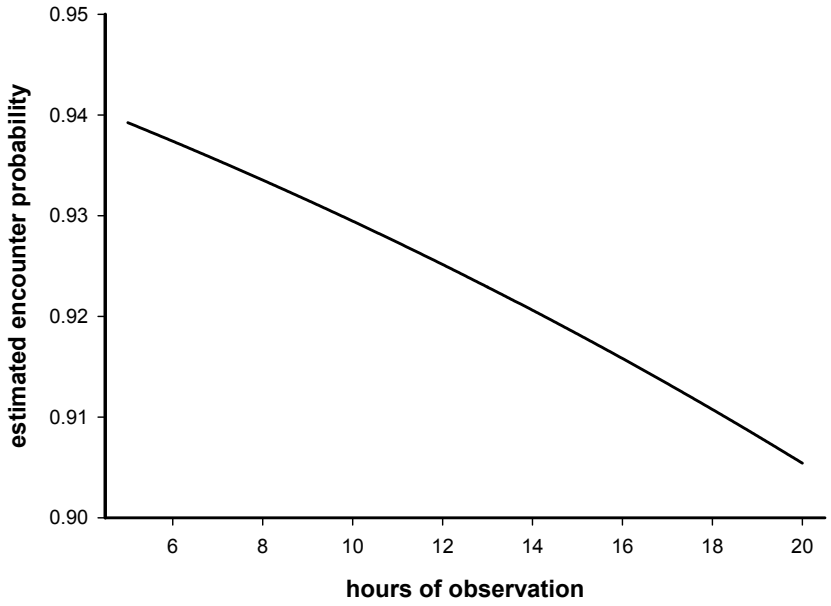
then the estimates of logit(p) for males (SEX = 1), for $5, 6, \cdots \rightarrow 20$ hours of observation, are calculated as:

```
[1] 2.738395 2.706431 2.674467 2.642503 2.610539 2.578575 2.546611 2.514647
[9] 2.482683 2.450719 2.418755 2.386791 2.354827 2.322863 2.290899 2.258935
```

which when back-transformed to the real probability scale yields

```
 [1] 0.9392546 0.9374050 0.9355031 0.9335474 0.9315368 0.9294699 0.9273455
 [8] 0.9251623 0.9229189 0.9206140 0.9182463 0.9158145 0.9133171 0.9107529
[15] 0.9081205 0.9054185
```

which, when plotted, yields a function that...



...doesn't even remotely suggest increasing encounter probability with increasing hours – in fact, it suggests the opposite of what we concluded!

So, have we made a mistake? Well, yes, in the sense that we did not fully interpret all of the β terms in our linear model:

$$\text{logit}(\hat{p}) = 1.4116023 + 1.4866125(\texttt{SEX}) + 0.0463456(\texttt{HOURS}) + (-0.0783096)(\texttt{SEX.HOURS}).$$

While the coefficient for 'HOURS' ($\hat{\beta}_3$) does clearly suggest that as hours of observation increases, encounter probability increases, look carefully at the equation – recall that 'HOURS' is also included in the interaction term, and that the estimated $\hat{\beta}_4 = -0.0783096$ for the interaction term is negative. Meaning, that depending on the SEX covariate, the relationship between logit(p) and HOURS might actually be negative, as seems to be the case when we considered males only (above).

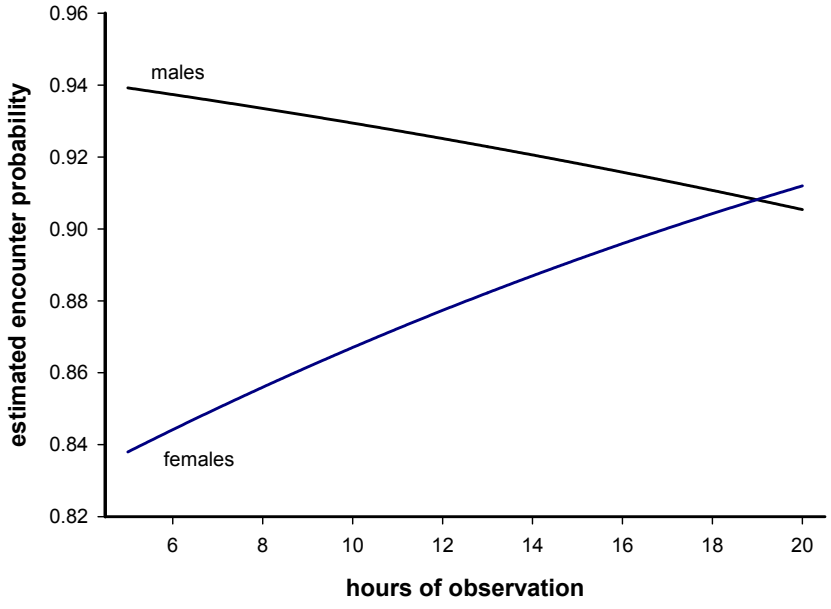
To illustrate the impact that this negative interaction term has, let's now estimate logit(p), but this time for *females*. Given our β estimates, then the estimates of logit(p) for females, for 5 $\rightarrow$ 20 hours of observation, are (for each hour increment):

```
[1] 1.643330 1.689676 1.736022 1.782367 1.828713 1.875058 1.921404 1.967750
[9] 2.014095 2.060441 2.106786 2.153132 2.199477 2.245823 2.292169 2.338514
```

which when back-transformed to the real probability scale yields

```
 [1] 0.8379876 0.8441815 0.8501810 0.8559889 0.8616083 0.8670425 0.8722949
 [8] 0.8773692 0.8822690 0.8869983 0.8915610 0.8959611 0.9002026 0.9042896
[15] 0.9082264 0.9120169
```

which, when plotted against the estimates for males, yields a function that...



...shows a classical 'interaction' – with increasing observation hours, male encounter probability ↓, while female encounter probability ↑. So – plotting your linear model is always a good idea, since it can be tricky to try to interpret the individual β terms, and embarrassing if you get it wrong (which is quite easy to do in complex models).

Now, one thing that is missing in our plot is any indication of parameter uncertainty. We recall that our estimates of each β term in our model are estimated with a SE, and thus, our reconstituted parameter estimates (in this case, on the real probability scale) are also estimated with uncertainty. How can we add confidence bands to our plots?

The short answer is, you can't. While **MARK** makes it possible to plot (or export) the β estimates, and the reconstituted parameter values for each occasion, it does not have the capability to directly generate a plot of the parameter estimates (of uncertainty in those estimates) as a function of a real covariate, over a range of values of that covariate. Meaning, **MARK** can't generate the plots we just made above, with or without confidence bands, simply by 'clicking a button or two'.

However, there are two ways you can generate the desired plot, albeit with a bit of work. One approach is to get **MARK** to treat *environmental* covariates as *individual* covariates, and then use the individual covariate plotting capabilities in **MARK** to generate the plot we want – predicted values for given values of the covariate, plus confidence bands. Using individual covariates in **MARK** is covered in Chapter 11. The specific details of using the individual covariate plotting tools in **MARK** to plot real 'environmental' covariates is covered in a `-sidebar-` in section 11.5.

The other somewhat more laborious approach is to make use of the Delta method – see the following `-sidebar-`.

begin sidebar

Calculating SE of predicted values from linear model

As briefly introduced earlier in this chapter (`-sidebar-` starting on p. 26), the Delta method is a relatively straightforward way for approximating the variance of transformed variables. The details underlying the Delta method are beyond our scope at this point (the Delta method is treated more fully in Appendix B); here we simply demonstrate the application for the purpose of estimating the variance of the prediction from a linear model.

Without proof, we can approximate the variance of some multi-variable function $\mathbf{Y}$ as

$$\widehat{\mathrm{var}}(\hat{Y}) \approx \mathbf{D}\boldsymbol{\Sigma}\mathbf{D}^{\mathsf{T}},$$

where $\mathbf{D}$ is the matrix of partial derivatives of the function $\mathbf{Y}$ with respect to each parameter, and $\boldsymbol{\Sigma}$ is the variance-covariance matrix for the parameters in the function.

In other words, to approximate the variance of some multi-variable function $\mathbf{Y}$, we (i) take the vector of partial derivatives of the function with respect to each parameter, β_i, in turn (i.e., the Jacobian), $\mathbf{D}$, (ii) right-multiply this vector by the variance-covariance matrix, $\boldsymbol{\Sigma}$, and (iii) right-multiply the resulting product by the transpose of the original vector of partial derivatives, $\mathbf{D}^{\mathsf{T}}$.

For the dipper example (above), the 'function' (i.e., the linear model fit to the data) is

$$\mathrm{logit}(\hat{p}) = \hat{\beta}_1 + \hat{\beta}_2(\mathtt{SEX}) + \hat{\beta}_3(\mathtt{HOURS}) + \hat{\beta}_4(\mathtt{SEX.HOURS}).$$

For convenience, we'll refer to this function as $\mathbf{Y}$. So, to derive an estimate of the variance for any value predicted by this function, *on the logit scale*, we first need to generate the vector of partial derivatives of the function with respect to each parameter, β_i in turn, $\mathbf{D}$:

$$\mathbf{D} = \left[\frac{\partial Y}{\partial \beta_1} \quad \frac{\partial Y}{\partial \beta_2} \quad \frac{\partial Y}{\partial \beta_3} \quad \frac{\partial Y}{\partial \beta_4}\right]$$

$$= \left[1 \quad \mathtt{SEX} \quad \mathtt{HOURS} \quad \mathtt{SEX.HOURS}\right].$$

The variance-covariance matrix among the parameters, $\mathbf{\Sigma}$, is given as

$$\mathbf{\Sigma} = \begin{bmatrix} \widehat{\mathrm{Var}}(\hat{\beta}_1) & \widehat{\mathrm{Cov}}(\hat{\beta}_1,\hat{\beta}_2) & \widehat{\mathrm{Cov}}(\hat{\beta}_1,\hat{\beta}_3) & \widehat{\mathrm{Cov}}(\hat{\beta}_1,\hat{\beta}_4) \\ \widehat{\mathrm{Cov}}(\hat{\beta}_2,\hat{\beta}_1) & \widehat{\mathrm{Var}}(\hat{\beta}_2) & \widehat{\mathrm{Cov}}(\hat{\beta}_2,\hat{\beta}_3) & \widehat{\mathrm{Cov}}(\hat{\beta}_2,\hat{\beta}_4) \\ \widehat{\mathrm{Cov}}(\hat{\beta}_3,\hat{\beta}_1) & \widehat{\mathrm{Cov}}(\hat{\beta}_3,\hat{\beta}_2) & \widehat{\mathrm{Var}}(\hat{\beta}_3) & \widehat{\mathrm{Cov}}(\hat{\beta}_3,\hat{\beta}_4) \\ \widehat{\mathrm{Cov}}(\hat{\beta}_4,\hat{\beta}_1) & \widehat{\mathrm{Cov}}(\hat{\beta}_4,\hat{\beta}_2) & \widehat{\mathrm{Cov}}(\hat{\beta}_4,\hat{\beta}_3) & \widehat{\mathrm{Var}}(\hat{\beta}_4) \end{bmatrix}.$$

After a little bit of matrix algebra, we can 'easily show' that

$$\begin{aligned} \widehat{\mathrm{var}}(\hat{Y}) &\approx \mathbf{D\Sigma D}^{\mathrm{T}} \\ &= \mathtt{SEX}\left((\mathtt{HOURS}\cdot\mathtt{SEX})\cdot\widehat{\mathrm{Var}}(\hat{\beta}_4) + \widehat{\mathrm{Cov}}(\hat{\beta}_4,\hat{\beta}_2)\cdot\mathtt{SEX} + \widehat{\mathrm{Cov}}(\hat{\beta}_4,\hat{\beta}_3)\cdot\mathtt{HOURS} + \widehat{\mathrm{Cov}}(\hat{\beta}_4,\hat{\beta}_1)\right) \\ &\quad + \mathtt{HOURS}\left(\mathtt{HOURS}\cdot\widehat{\mathrm{Var}}(\hat{\beta}_3) + \widehat{\mathrm{Cov}}(\hat{\beta}_3,\hat{\beta}_4)\cdot(\mathtt{HOURS}\cdot\mathtt{SEX}) + \widehat{\mathrm{Cov}}(\hat{\beta}_3,\hat{\beta}_2)\cdot\mathtt{SEX} + \widehat{\mathrm{Cov}}(\hat{\beta}_3,\hat{\beta}_1)\right) \\ &\quad + \mathtt{SEX}\left(\mathtt{SEX}\cdot\widehat{\mathrm{Var}}(\hat{\beta}_2) + \widehat{\mathrm{Cov}}(\hat{\beta}_2,\hat{\beta}_4)\cdot(\mathtt{HOURS}\cdot\mathtt{SEX}) + \widehat{\mathrm{Cov}}(\hat{\beta}_2,\hat{\beta}_3)\cdot\mathtt{HOURS} + \widehat{\mathrm{Cov}}(\hat{\beta}_2,\hat{\beta}_1)\right) \\ &\quad + \widehat{\mathrm{Var}}(\hat{\beta}_1) + \widehat{\mathrm{Cov}}(\hat{\beta}_1,\hat{\beta}_4)\cdot(\mathtt{HOURS}\cdot\mathtt{SEX}) + \widehat{\mathrm{Cov}}(\hat{\beta}_1,\hat{\beta}_2)\cdot\mathtt{SEX} + \widehat{\mathrm{Cov}}(\hat{\beta}_1,\hat{\beta}_3)\cdot\mathtt{HOURS}. \end{aligned}$$

Admittedly, this looks a little ugly, but most computer algebra systems (like **Maple**, **Mathematica**, **Maxima**...) handle this sort of thing very easily.

OK, now what do we do with this BUE ('big ugly equation')? Simple – we substitute in our estimates for the various parameters in this equation (i.e., $\hat{\beta}_1$, $\widehat{\mathrm{Var}}(\hat{\beta}_1)$, ...), leaving out the parameter we wish to plot predictions against. In our example, we're interested in plotting predicted encounter probabilities as a function of HOURS of observation in the field.

Let's say for the moment we're interested in the relationship between HOURS of observation and predicted detection probability, for male dippers (i.e., for SEX=1). All we do next is to substitute the following into the BUE shown above:

- $\mathtt{SEX} = 1$
- $\hat{\beta}_1 = 1.4115995$, $\hat{\beta}_2 = 1.4866175$, $\hat{\beta}_3 = 0.0463458$, $\hat{\beta}_4 = -0.0783100$
- $\mathbf{\Sigma} = \begin{bmatrix} \widehat{\mathrm{Var}}(\hat{\beta}_1) & \widehat{\mathrm{Cov}}(\hat{\beta}_1,\hat{\beta}_2) & \widehat{\mathrm{Cov}}(\hat{\beta}_1,\hat{\beta}_3) & \widehat{\mathrm{Cov}}(\hat{\beta}_1,\hat{\beta}_4) \\ \widehat{\mathrm{Cov}}(\hat{\beta}_2,\hat{\beta}_1) & \widehat{\mathrm{Var}}(\hat{\beta}_2) & \widehat{\mathrm{Cov}}(\hat{\beta}_2,\hat{\beta}_3) & \widehat{\mathrm{Cov}}(\hat{\beta}_2,\hat{\beta}_4) \\ \widehat{\mathrm{Cov}}(\hat{\beta}_3,\hat{\beta}_1) & \widehat{\mathrm{Cov}}(\hat{\beta}_3,\hat{\beta}_2) & \widehat{\mathrm{Var}}(\hat{\beta}_3) & \widehat{\mathrm{Cov}}(\hat{\beta}_3,\hat{\beta}_4) \\ \widehat{\mathrm{Cov}}(\hat{\beta}_4,\hat{\beta}_1) & \widehat{\mathrm{Cov}}(\hat{\beta}_4,\hat{\beta}_2) & \widehat{\mathrm{Cov}}(\hat{\beta}_4,\hat{\beta}_3) & \widehat{\mathrm{Var}}(\hat{\beta}_4) \end{bmatrix}$

$$= \begin{bmatrix} 1.4707909088 & -1.3489390733 & -0.1048562399 & 0.0965896636 \\ -1.3489390733 & 4.4087794743 & 0.0978171691 & -0.3077306503 \\ -0.1048562399 & 0.0978171691 & 0.0084589932 & -0.0079429315 \\ 0.0965896636 & -0.3077306503 & -0.0079429315 & 0.0237105215 \end{bmatrix}.$$

After the substitutions, the BUE is not nearly so 'big' or 'ugly':

$$\widehat{\mathrm{Var}}(\hat{Y}) \approx 0.0162836517(\mathtt{HOURS})^2 - 0.436360115(\mathtt{HOURS}) + 3.1816922365.$$

All you need to do at this point is use this equation to generate the estimated uncertainty for the encounter probability predicted for a given value of the covariate HOURS. The following **R** script demonstrates one approach for generating predicted encounter probabilities, and estimated SE and 95% CI for those predictions, for $1 \rightarrow 20$ HOURS.

```
# initialize hours vector for 1 -> 20 hours
  h <- seq(1:20);
```

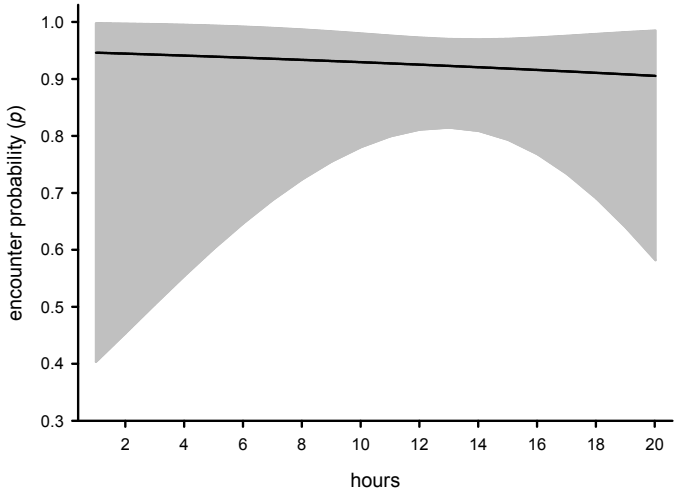
```
# generate estimates of var + SE on logit scale as a function of hours
  logit_var <- 0.0162836517*h^2-0.436360115*h+3.1816922365;
    logit_se <- sqrt(logit_var);

# generate estimated encounter probability on logit scale
  b1=1.4115995; b2=1.4866175; b3=0.0463458; b4=-0.0783100;  sex=1;

   logit_p <- b1+b2*sex+b3*hours+b4*hours*sex;
     p <- exp(logit_p)/(1+exp(logit_p));
     var <- (p*(1-p))^2*logit_var; # Delta method for var on prob scale
     se <- sqrt(var)

# now derive LCI and UCI
  uci <-exp(logit_p+1.96*logit_se)/(1+exp(logit_p+1.96*logit_se));
  lci <-exp(logit_p-1.96*logit_se)/(1+exp(logit_p-1.96*logit_se));

# put everything together
  results <- cbind(p,se,lci,uci)
  print(results);
```

Note that the SE and 95% CI are derived on the *logit* scale, and then back-transformed. This is done to guarantee that the calculated 95% CI is $[0, 1]$ bounded for parameters (like φ or p) that are $[0, 1]$ bounded. Because the logit transform is not linear, the *reconstituted* 95% CI will not be symmetrical around the parameter estimate, especially for parameters estimated near the $[0, 1]$ boundaries.

The first 4 (of 20) values generated from this script are shown below:

```
            p         se       lci       uci
[1,] 0.9461528 0.08466548 0.4035014 0.9978138
[2,] 0.9445008 0.08076792 0.4537064 0.9971406
[3,] 0.9428013 0.07662883 0.5043041 0.9962693
[4,] 0.9410530 0.07225869 0.5540990 0.9951479
```

We can use the full output from the script to plot predicted encounter probability with 95% CI to those predictions:



As mentioned earlier, the *reconstituted* 95% CI will not be symmetrical around the parameter estimate, especially for parameters estimated near the [0, 1] boundaries, as is clearly the case here. Also, we note that as the value for the covariate HOURS is much greater or lesser than the mean value (≈ 12 hours), the 95% CI gets progressively larger. This is expected as there is less 'information' at either end of the distribution of HOURS on which to base our inference, and thus, more uncertainty in our estimate.

end sidebar

6.9. A special case of 'real covariates' – linear trend

Although we are not usually interested in a simple demonstration of temporal variation in our parameter estimates (as discussed in the preceding section), there is one 'special' case where we might be. If our estimates are believed to be increasing or decreasing over time (i.e., showing a trend). We will now explore how to use **MARK** to test for 'trend' in the data – linear increase or decrease in survival or encounter.

To demonstrate the mechanics, we created a simple data set (**linear.inp**), which contains simulated data which we will use for our analysis of linear trend. There are 8 occasions in the data set. Assume that the 'simulated animals' are all marked as adults. We started with fitting the basic CJS, time-dependent model. Since the data were simulated, we can safely assume that this is an acceptable model, and fits the data (i.e., you can leave $\hat{c}$ at the default value of 1.000).

Since there is only one group, this analysis is very easy, and should take you only a few minutes with **MARK**. At this stage, we'll assume you know the steps for fitting this model, so we'll proceed directly to the results. The deviance of the CJS model fit to these data was 185.388, and the AIC_c value was 7,980.78. The estimates for both survival and encounter probabilities are tabulated below:

survival		**encounter**	
Parm	*Estimate*	*Parm*	*Estimate*
1	0.7237	8	0.5492
2	0.7022	9	0.5040
3	0.6280	10	0.5403
4	0.5836	11	0.5256
5	0.6267	12	0.4316
6	0.4586	13	0.5428
7	0.5027	14	0.5027

Note that the value of the last estimate for both survival and encounter is the same (0.5027) – remember, this is the β_8 term. Thus, for this model, we have 13 total potentially identifiable parameters. The deviance for the model was 185.39, and the AIC_c for this model is therefore 7980.78. Note that the time-specific estimates show a clear trend (not surprising, since they were simulated this way).

Now, let's proceed to see how to use **MARK** to fit a model with a linear trend – this will allow us to formally test our hypothesis that there may be a decline in survival over time. Doing this in **MARK** is very straightforward. Mechanically, we again make some simple modifications to the CJS design matrix. First, what are the index values of the survival parameters we want to constrain (i.e., constrain to be a linear function of time)? Clearly, parameters $1 \rightarrow 7$.

Now, as hinted in the last section, to fit a linear trend model, we need to modify the design matrix – how would we do this? Think back to the first example using the dipper data set – the FLOOD analysis.

Recall that in the design matrix, we had 4 columns of numbers in that file: one for the intercept, one for SEX, one for FLOOD, and one for the interaction term (SEX.FLOOD). Concentrate on the FLOOD column. We coded FLOOD as a simple binary variable: there was either a flood ('1') or there wasn't ('0'). In our present example, however, things aren't quite so simple. We are trying to build a model with a linear trend. In other words, a systematic change in survival (up or down) through time.

What do we know about a 'trend', and how does it differ from the flood example? By definition, a trend has a slope which is significantly different from 0. Given that the slope differs from 0, then on average, $y_{(i-1)} < y_i < y_{(i+1)}$ for an increasing trend with (i), and the reverse for a decreasing trend. Second, a trend is 'continuous' through time – it is not a simple binary condition, as was the case with flood. Thus, we need to code a 'trend' through time in such a way that it meets these 2 conditions.

As it turns out, this is very simple to do. To code for a linear trend, all you need to do is write a series of ordinal, evenly spaced increasing (or decreasing) numbers, 1 through n (where n is the number of occasions you want to fit the 'trend' to). You don't have to start with the number 1, but you do need to use the sequence {starting value} + 1, {starting value} + 2, and so on. So, what would the survival elements of the design matrix look like for this 8 occasion study?

Just like this:

1	1
1	2
1	3
1	4
1	5
1	6
1	7

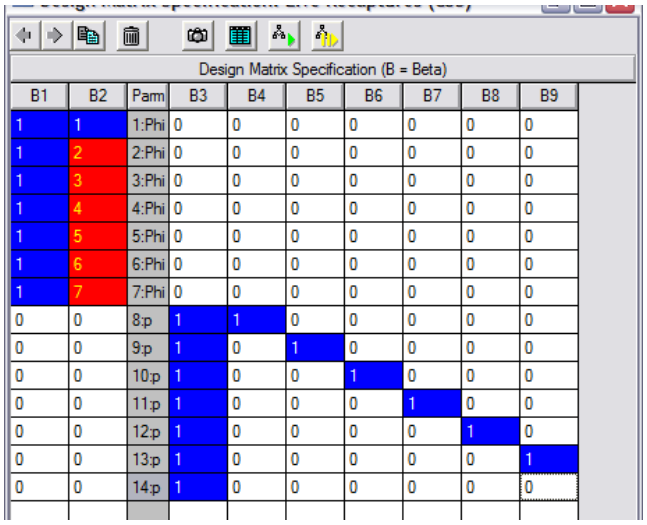
Hmmm. . .pretty strange looking (perhaps) – what do we have here? The first column corresponds to the intercept, while the second column is the dummy variable coding for the linear trend. In other words,

$$\text{logit}(\varphi) = \texttt{INTERCEPT} + \beta_1(\texttt{T}),$$

where T (commonly referred to as 'cap T') indicates a linear trend (we use a capital T for trend, to distinguish a trend model from a simple time-dependent model, which is usually indicated using a lower-case t).

So, only 2 β terms: one for the intercept, and one for the linear relationship between the response variable (φ, in this case), and the value for 'trend'. You should see the connection (at least structurally) between this linear model, and how we coded for the 'effort' covariate in the previous section. The order of the numbers (1 to 7, or 7 to 1) makes no difference – **MARK** will simply use the numbers to fit a linear trend – it will let the data determine if the trend is up or down (if any trend exists). Get it? The design matrix for this model ('Phi(linear)p(t)') is shown below at the top of the next page.

Note that this model has 9 parameters (1 for the intercept, 1 for the slope of the 'regression' of survival on time as a linear covariate, and 7 encounter parameters – no non-identifiable product β terms. Make sure you know why!). The deviance was 189.53, and the AIC_c was 7976.88. The model where survival was constrained to vary linearly with time (note we do not specify increase or decrease) is a more parsimonious model than the initial time-dependent model – in fact, it is approximately 7 times better supported by the data. This is perhaps not surprising – the data were simulated with a decline in survival!



Design Matrix Specification (B = Beta)

B1	B2	Pam	B3	B4	B5	B6	B7	B8	B9
1	1	1:Phi	0	0	0	0	0	0	0
1	2	2:Phi	0	0	0	0	0	0	0
1	3	3:Phi	0	0	0	0	0	0	0
1	4	4:Phi	0	0	0	0	0	0	0
1	5	5:Phi	0	0	0	0	0	0	0
1	6	6:Phi	0	0	0	0	0	0	0
1	7	7:Phi	0	0	0	0	0	0	0
0	0	8:p	1	1	0	0	0	0	0
0	0	9:p	1	0	1	0	0	0	0
0	0	10:p	1	0	0	1	0	0	0
0	0	11:p	1	0	0	0	1	0	0
0	0	12:p	1	0	0	0	0	1	0
0	0	13:p	1	0	0	0	0	0	1
0	0	14:p	1	0	0	0	0	0	0

Now, one subtle variation in this theme: suppose that we want to test for a non-linear trend. How would we do this? Well, there are several ways to analyze non-linear relationships. Perhaps the easiest is to use multiple regression, fitting a series of power terms to the function. For example, a comparison of the model $Y = X + X^2$ to model $Y = X$ is a formal test of the significance of the X^2 term. If the X^2 term is not significant, and if the model $Y = X$ fits the data, then we can conclude that the relationship is linear. How would we do this? In fact, it is very simple. All you need to do is add another column to your design matrix file to accommodate the X^2 term. It's that easy!

Warning: While the mechanics of what we've just demonstrated for fitting a 'trend' model appear straightforward, there is a conceptual limitation to this particular approach which you need to consider. Fitting a linear trend model using the approach we described in this section 'forces' the parameter estimates to fall precisely on a line. Clearly, this 'statistical constraint' enforces a 'biological' hypothesis which is implausible – the estimates are no more likely to fall precisely on a line than they are to be exactly the same in a 'constant over time' model. In addition, inference concerning the estimated 'slope' of the trend from a 'cap T' model is problematic – in such models, the estimated standard errors are based only on sampling variation, and would be biased low compared to a direct regression on true estimates (which is clearly not possible because the true estimates are not known).

However, an approach based on *random effects* solves this problem. In this context, random effects models assume that 'true' annual estimates do not fall exactly on any simple, smooth model; the deviation of estimates of some parameter from such models are treated as random. So, rather than assume the estimates fall precisely on a straight line (i.e., applying a simple 'cap T' model approach), the random effects regression assumes that the actual 'true' estimates vary randomly around some trend line (where the trend line represents the mean estimate if multiple samples were available from the same range of years in the data). Random effects models are covered in detail in Appendix D.

begin sidebar

Another type of 'trend': the cumulative logit link

The cumulative logit link (CLogit) function is useful for constraining a set of parameters to monotonically increase. Suppose that you desire the relationship of $S_1 \leq S_2 \leq S_3$, but do not want to enforce the relationship on the logit scale that

$$\text{logit}[S_2] - \text{logit}[S_1] = \text{logit}[S_3] - \text{logit}[S_2]$$

as a trend model (discussed in the preceding section) would do.

The CLogit link would generate this relationship as:

$$S_1 = \frac{e^{\beta_1}}{1 + e^{\beta_1} + e^{\beta_2} + e^{\beta_3}} \qquad S_2 = \frac{e^{\beta_1} + e^{\beta_2}}{1 + e^{\beta_1} + e^{\beta_2} + e^{\beta_3}} \qquad S_3 = \frac{e^{\beta_1} + e^{\beta_2} + e^{\beta_3}}{1 + e^{\beta_1} + e^{\beta_2} + e^{\beta_3}}$$

To use the CLogit link, you have to specify a separate CLogit link for each set of parameters that are to be constrained. In addition, you also have to specify the order of the parameters for the set.

For the preceding example, the link function for each of the 3 survival probabilities would be:

```
S(1): CLogit(1,1)
S(2): CLogit(1,2)
S(3): CLogit(1,3)
```

If you have a second set of parameters (in the same model) that you also want to enforce a monotonic *increase* on, say $S_4 \leq S_5 \leq S_6$, the appropriate links would be:

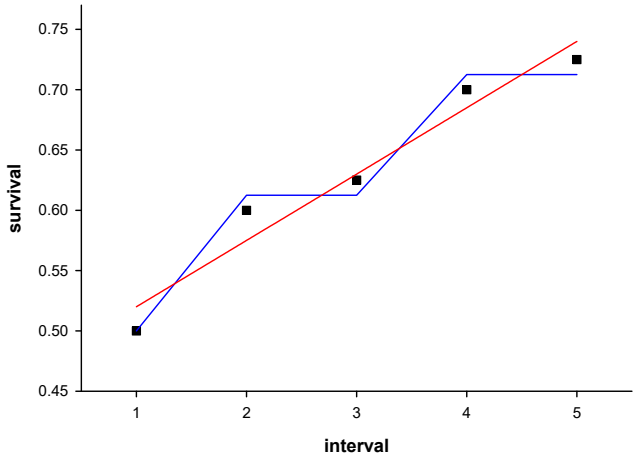
```
S(4): CLogit(2,1)
S(5): CLogit(2,2)
S(6): CLogit(2,3)
```

Note that you can force a monotonic *decrease* by reversing the order of the real parameters in the CLogit set. For example, to enforce a monotonic decrease on parameters S_1, S_2, and S_3, you would use

```
S(1): CLogit(1,3)
S(2): CLogit(1,2)
S(3): CLogit(1,1)
```

You specify these link functions by selecting the '**Parm-Specific**' option as the link function, and entering the appropriate specification directly in the edit box next to the parameter name.

We'll demonstrate the use of the CLogit link by means of a worked example, based on simulated live encounter data (6 occasions, 250 individuals marked and released at each occasion), where apparent survival φ tends to increase monotonically, while the encounter probability p is constant over time (the simulated data are contained in **clogit_demo.inp**). The true values for the survival parameters are: $S_1 = 0.500$, $S_2 = 0.600$, $S_3 = 0.625$, $S_4 = 0.700$ and $S_5 = 0.725$. Note that $S_2 \cong S_3$, and $S_4 \cong S_5$. As shown in the following



even though there is an obvious tendency for survival to increase over time, the increase is clearly not strictly linear, as indicated by the square symbols in the following figure:

However, the question is whether a model which constrains the estimates to be strictly linear (red line) is a better fit to the data than one which allows for possible equality between some of the estimates (darker line).

Recall that fitting a linear trend using the design matrix forces the reconstituted estimates to fall on a perfectly straight line. So, in this example, it might seem possible (perhaps likely) that a model based on the cumulative logit link (which allows for possible equality between some of the estimates) may prove to be more parsimonious than a strictly linear trend model (even though the latter will often have fewer parameters).

First, build model $\{\varphi_t\ p_.\}$, and add the results to the browser. Note that the real parameter estimates from this model (shown below)

Real Function Parameters of {phi(t)p(.)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.5354208	0.0429860	0.4509609	0.6178968
2:Phi	0.6972772	0.0399885	0.6137379	0.7695335
3:Phi	0.5971751	0.0328316	0.5315093	0.6595326
4:Phi	0.7669756	0.0375781	0.6855091	0.8324955
5:Phi	0.7378645	0.0449388	0.6409564	0.8161203
6:p	0.5011069	0.0201832	0.4616233	0.5405768

are not particularly close to the true parameter values used to simulate the data. Given the small sample size of newly marked individuals released at each occasion, this is perhaps not surprising.

Now, let's fit two additional models: model $\{\varphi_{\text{trend}}\ p_.\}$ (simple linear trend on apparent survival), and model $\{\varphi_{\text{CLogit}}\ p_.\}$, where we use the cumulative logit link to impose an increasing, ordinal – but not strictly linear – structure on the apparent survival values.

If you followed the material covered in this section, fitting the simple linear trend model should be relatively easy. Here is the design matrix corresponding to the simple linear trend model:

Design Matrix Specification (B = Beta)

B1 Phi Int	B2 Phi t1	Parm	B3 Phi t2
1	1	1:Phi	0
1	2	2:Phi	0
1	3	3:Phi	0
1	4	4:Phi	0
1	5	5:Phi	0
0	0	6:p	1

Go ahead, fit this model, and add the results to the browser:

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(t)p(.)}	3508.3010	0.0000	0.73854	1.0000	6	39.5130
{phi(trend)p(.)}	3510.3778	2.0768	0.26146	0.3540	3	47.6229

We see clearly that model $\{\varphi_{\text{T}}\ p_.\}$ is not particularly well-supported by the data (at least, relative to the time-dependent model $\{\varphi_t\ p_.\}$).

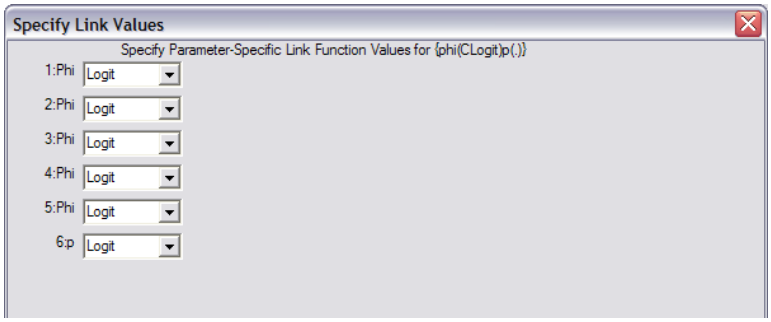
Now, let's fit model $\{\varphi_{\text{CLogit}}\, p_.\}$, where we impose an increasing, ordinal constraint on the estimates of apparent survival. In other words, we're constraining the estimates of $S_1, S_2, \cdots, S_5$ such that

$$S_1 \leq S_2 \leq S_3 \leq S_4 \leq S_5$$

Now, if you look closely at the above, you'll see that there are a very large number of possible models which would satisfy this constraint – **MARK** will effectively test all of them, and select the most parsimonious of the set.

To build this model in **MARK**, first retrieve model `phi(t)p(.)` from the browser. You can do this easily by selecting the model in the browser, right-clicking, and selecting '**`Retrieve`**'. Then, select '**`Run`**' again, and change the name of the model to `phi(CLogit)p(.)`.

Now, before clicking the '**`OK to Run`**' button, we need to specify the cumulative logit link for the 5 apparent survival parameters. To do this, you need to select the '**`Parm-specific`**' radio button option in the list of link functions. Now, when you click the '**`OK to Run`**' button, **MARK** will respond with a window where you will specify the parameter specific link functions you want to use:

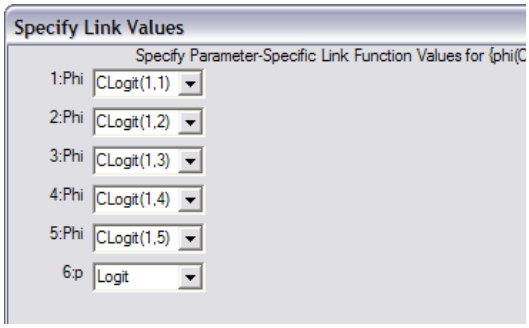


Note that in this case, **MARK** defaults to the logit link for all parameters (the default link for each parameter will either be the sin or logit link, depending on whether or not you are using an identity design matrix).

To fit our model, we need to change from the default logit link to the cumulative logit link, for parameters $1 \rightarrow 5$, corresponding to $\varphi_1 \rightarrow \varphi_5$. To enforce a monotonic increase in apparent survival, subject to the condition that

$$S_1 \leq S_2 \leq S_3 \leq S_4 \leq S_5$$

we simply specify the CLogit link for each of the survival parameters. For this example, here the completed link specification window:



Note that you *cannot* select the CLogit link function from the drop-down list of link functions like as with all of the other link functions, because you have to specify the set and the order of the parameter

within the set. Therefore, you have to *manually* enter the link function in each edit box next to the real parameter value to which it pertains.

Once you've specified the link functions, go ahead and run the model, and add the results to the browser:

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(CLogit)p(.)}	3507.4514	0.0000	0.53039	1.0000	4	42.6877
{phi(t)p(.)}	3508.3010	0.8496	0.34682	0.6539	6	39.5130
{phi(trend)p(.)}	3510.3778	2.9264	0.12278	0.2315	3	47.6229

We see clearly that the model using the cumulative logit link has considerable AIC weight in the data, relative to the other models in the model set. The estimates from this model are shown below:

mrk4065z.tmp - Notepad

File Edit Format View Help

CLogit demo

Real Function Parameters of {phi(CLogit)p(.)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.5460136	0.0435586	0.4601249	0.6292483
2:Phi	0.6422471	0.0221163	0.5978391	0.6843406
3:Phi	0.6422471	0.0221163	0.5978391	0.6843406
4:Phi	0.7477500	0.0260974	0.6932725	0.7954073
5:Phi	0.7477502	0.0260975	0.6932724	0.7954077
6:p	0.5016806	0.0198774	0.4627895	0.5405514

You see that the results follow the basic constraint specification

$$S_1 \leq S_2 \leq S_3 \leq S_4 \leq S_5$$

In this case, the most parsimonious model was one where

$$S_1 \leq S_2 = S_3 \leq S_4 = S_5$$

which seems quite reasonable given that for the true parameter values (noted a few pages back), $S_2 \cong S_3$, and $S_4 \cong S_5$.

So, how many parameters are estimated, subject to the constraint (below)?:

$$S_1 \leq S_2 \leq S_3 \leq S_4 \leq S_5$$

Given that the most parsimonious ML estimates for these simulated data subject to this constraint are

$$S_1 \leq S_2 = S_3 \leq S_4 = S_5$$

it is clear that 3 parameters are estimated.

Now, it is worth noting here that what we have done is essentially equivalent to an 'all subsets' regression – **MARK** has simply found the most parsimonious model from the set of models specified by the constraint that

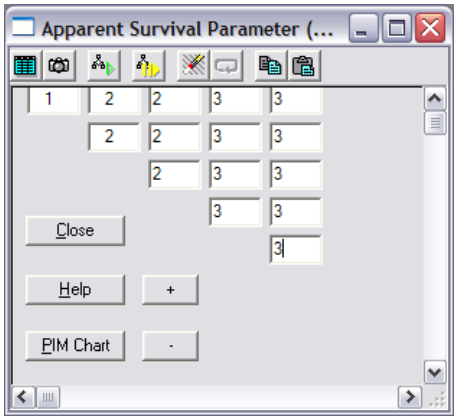
$$S_1 \leq S_2 \leq S_3 \leq S_4 \leq S_5$$

This may be potentially useful for finding a parsimonious model for improving precision of reconstituted parameter estimates, or if you believe that there is a general monotonic increase or decrease in some parameter, where you have no *a priori* expectation to the particular form of the function (other than it being monotonic in one direction or the other). In this case, the most parsimonious model was one where

$$S_1 \leq S_2 = S_3 \leq S_4 = S_5$$

was *not* motivated by a particular *a priori* hypothesis.

If in fact you had reason to include a model with this constraint in your model set (i.e., if you had an *a priori* expectation that $S_2 = S_3$ and $S_4 = S_5$, with a monotonic increase from $S_1 \rightarrow S_2 = S_3 \rightarrow S_4 = S_5$), then it would be more appropriate to (i) first construct a PIM with the basic structure $\{S_1, S_2 = S_3, S_4 = S_5\}$ (shown below) to which you then (ii) apply a cumulative logit constraint to parameters $1 \rightarrow 3$. If you try it for this example problem, you'll see that you generate *exactly* the same reconstituted parameter estimates as you did from the cumulative logit link applied to the fully time-dependent PIM.



However, by applying the cumulative logit to the PIM that reflects our *a priori* beliefs about equality of certain parameters, then we avoid the risk of having to concoct a *post hoc* 'story' (not withstanding their frequent entertainment value) to explain the particular CLogit model that **MARK** has found to be most parsimonious. The distinction here is subtle, but important.

end sidebar

6.10. More than 2 levels of a group

It is not uncommon to have more than 2 levels of a classification variable in an analysis. For example, you may have a control and 2 or more treatment groups. How would you code the design matrix for such a situation? In fact, it's easy (well, relatively), if you remember some basic principles from analysis of variance (ANOVA). The number of columns used to characterize group differences (e.g., belonging to one group or not) will always equal the number of dummy variables (coded '0' or '1') that you need to characterize group differences. For any variable that we treat as 'categorical' or 'classification' (i.e., both COLONY and TIME in the swift example) with k levels, the number of columns needed is $(k-1)$, which happens to be the numbers of degrees of freedom associated with a factor in a standard ANOVA (note – it's *not* a coincidence). In short, the number of columns (hence, the number of dummy variables) needed

equals the degrees of freedom associated with that variable. So, the minimum number of columns of dummy variables needed to specify our model are defined by the number of degrees of freedom for each factor, plus any interaction terms.

Of course, it **is** possible to specify a model with more columns than the minimum set we've just described. Would this be wrong? Not exactly – your estimates would be 'correct', but it becomes very difficult to count (separately) identifiable parameters. And since counting parameters is essential to model testing, using more columns than necessary in your design matrix to specify the model should be avoided.

Consider an example of a study over 5 occasions, where we have 3 'groups', or levels of our 'main effect'. Thus, we need $(3 - 1) = 2$ columns of 'dummy variables' to specify group association. Again, it is important to understand the logic here: we need $(n - 1) = (3 - 1) = 2$ columns, because we have an intercept in the model – the intercept codes for one level of the treatment, and the other two columns code for the remaining two levels of the treatment.

Suppose we also have a quantitative covariate (say, hours of observation). Linear terms have 1 degree of freedom, so one column of covariate values. Finally, for the interaction, we need $(3 - 1) \times (1) = 2$ columns.

Here is the design matrix – we have formatted it slightly to emphasize the 'logical connection' amongst the columns.

INTCPT	GROUP		HOURS	GROUP.HOURS	
1	0	0	1.1	0	0
1	0	0	0.2	0	0
1	0	0	3.4	0	0
1	0	0	4.1	0	0
1	0	1	1.1	0	1.1
1	0	1	0.2	0	0.2
1	0	1	3.4	0	3.4
1	0	1	4.1	0	4.1
1	1	0	1.1	1.1	0
1	1	0	0.2	0.2	0
1	1	0	3.4	3.4	0
1	1	0	4.1	4.1	0

The first column is the intercept. The next 2 columns on the left indicate group: group is identified depending upon the pair of dummy variables across these 2 columns: '`0 0`', '`0 1`', and '`1 0`'. The middle column (of the 5 total columns), is the 'covariate' column.

The last 2 columns are the interaction columns (interaction of group and covariate). Remember, the interaction term can be thought of as a 'multiplication term' – the product of the various factors contained in the interaction. Since this interaction is the interaction of 'group' (2 columns) and 'covariate' (1 column), then we have $(2 \times 1) = 2$ columns for the interaction. We simply multiply each element of the group column vectors by its corresponding element in the 'hours' column vector. Therefore, 6 columns total.

Now, if we were applying this design matrix, and we wanted to test for the significance of the interaction term, we would first run the constraint using all 6 columns in the matrix, and then a second time using only the first 4 columns – 4 because we want to drop the interaction term, which is 'stored' in columns 5 and 6.

Note that it is important to use the minimum number of columns of '0' or '1' dummy variables to characterize your groups. Why? Because each column in your matrix becomes a slope, which is an estimated parameter. Our goal is to use as few parameters as possible to specify a model. Extra columns wouldn't make your model 'wrong', but would make the counting of (separately) identifiable parameters more difficult.

It is also important to note that all that is necessary is that you use $(n-1)$ columns to code the 'group' variable (in this case, $(3-1) = 2$ columns). However, the actual 'dummy variable' coding you use to specify group is arbitrary. For example, in the preceding example, we used '0 0' for group 1, '0 1' for group 2, and '1 0' for group 3.

However, we could have used '1 1' for group one, '1 0' for group 2, and '0 1' for group 3, or any of a number of other combinations. They would all yield the same results (although, clearly, the coding in the interaction columns will change from the preceding example to reflect whatever coding you use for group columns). Try a few examples to confirm this for yourself.

6.11. > 1 classification variables: *n*-way ANOVA

Back in Chapter 2, we briefly considered the formatting of the .INP file for situations where you have > 1 classification factors. We considered an example where both males and females were sampled at each of two colonies: a good colony, and a poor colony. Thus, two classification factors: SEX and COLONY.

Recall that in the input file, we included a frequency column for each (SEX.COLONY) combination: one frequency column for females from the good colony, one frequency column for females from the poor colony, one frequency column for males from the good colony, and finally, one frequency column for males from the poor colony. We simulated a simple data set (**multi_group.inp**) including these 4 combinations.

Here, we focus on building models which have both SEX and COLONY effects. Suppose, for example, we want to build the following linear model for φ:

$$\varphi = \texttt{SEX} + \texttt{COLONY} + \texttt{TIME} + \texttt{SEX.TIME} + \texttt{COLONY.TIME} + \texttt{SEX.COLONY} + \texttt{SEX.COLONY.TIME}.$$

In other words, a 3-way ANOVA (in effect, TIME is the third classification variable in this analysis).

How do we go about building this model? Start program **MARK**, and begin a new project. Select **multi_group.inp**. Specify 5 occasions. Now, for your first challenge – how many attribute groups? You might think either 2, or 3. Well, perhaps you're comfortable enough now with **MARK** to think just two – SEX and COLONY (realizing that TIME is an implicit attribute, and doesn't need to be counted).

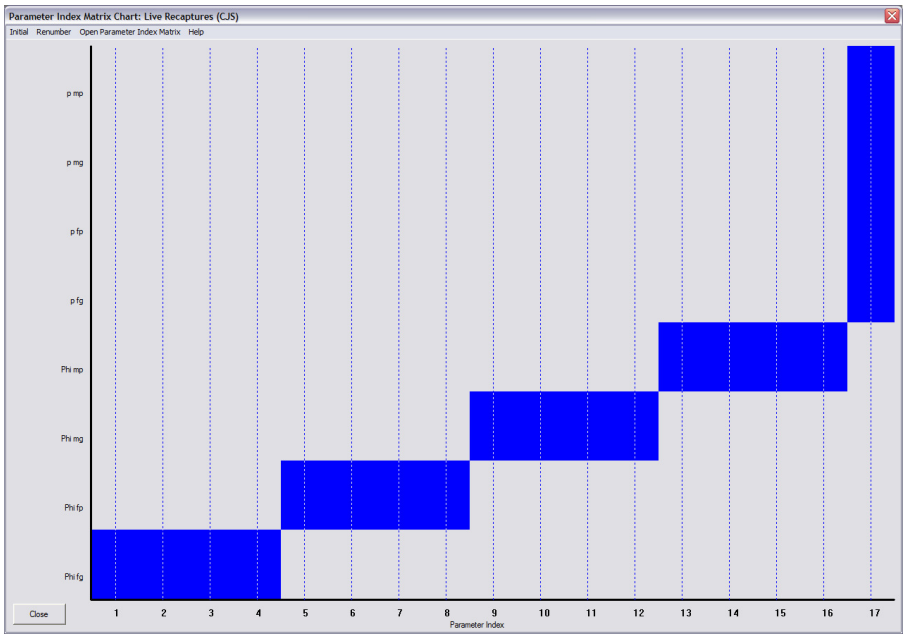
Unfortunately, you'd not be correct. In fact, there are 4 attribute groups – corresponding to each of the 4 'SEX.COLONY' combinations (i.e., the number of frequency columns in the .INP file). So, you need to tell **MARK** that there are 4 attribute groups.

Next, what to call them? In the .INP file (first few lines of which are shown below)

```
/* multiple attribute group demonstration (chapter 6
   fem-good, fem-poor, male-good, male-poor. 5 occasions */
00010        150   145   171   167;
00011        200   205   179   183;
00100        213   198   131    77;
00101         14    26    32    50;
00110         54    59    95   101;
00111         69    67    92   122;
01000         93    64    74    51;
```

we see that the first two frequency columns are for females sampled at the good and poor colonies, respectively, and the last two frequency columns are for males, sampled at the good and poor colonies, respectively. So, let's label the 4 attribute groups as `FG`, `FP`, `MG`, and `MP`, respectively.

Now, let's build our model – to simplify, let's assume that the encounter probability p is the same for both sexes and both colonies, and constant over time. Here is the PIM chart for this model:



Make sure you see the connection between the PIM chart, and the model we're trying to build. Go ahead and run the model – call it '`phi(S.C.T)p(.) - PIM`'. We add the `PIM` label to the model name to remind us that the model was built using the PIM chart.

The real estimates from this model are shown below:

Real Function Parameters of {phi(S.C.T)p(.) - PIM}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.4640517	0.0286232	0.4086345	0.5203712
2:Phi	0.8718165	0.0249614	0.8144706	0.9133235
3:Phi	0.4526297	0.0205268	0.4127911	0.4930846
4:Phi	0.7648547	0.0272867	0.7072604	0.8140961
5:Phi	0.5411491	0.0282499	0.4854991	0.5957912
6:Phi	0.9522418	0.0217737	0.8863770	0.9807552
7:Phi	0.4989976	0.0200902	0.4597083	0.5382993
8:Phi	0.7742726	0.0261616	0.7189396	0.8214180
9:Phi	0.5087660	0.0284463	0.4531330	0.5641826
10:Phi	0.8675647	0.0211225	0.8204310	0.9037776
11:Phi	0.7401697	0.0209846	0.6969862	0.7791490
12:Phi	0.6864606	0.0238298	0.6379828	0.7311821
13:Phi	0.5481091	0.0279462	0.4929724	0.6020895
14:Phi	0.9065044	0.0179716	0.8648389	0.9362723
15:Phi	0.8475677	0.0189659	0.8065778	0.8811510
16:Phi	0.7312073	0.0227423	0.6843772	0.7733897
17:p	0.7359362	0.0081473	0.7196611	0.7515926

Now, we want to try to construct this model using the design matrix approach (since we want to be able to use the flexibility of the design matrix to build models we can't build with the PIMs).

First – how many columns should the design matrix have? Well, if you 'cheat' and look at the results browser, you might guess 17 – one column for each of the estimated parameters. But, we want to confirm our hunch – we do this by writing out the linear model in full.

Remember, SEX has two levels (male and female), so 1 column for sex. Similarly, COLONY has two levels (good and poor), so again, 1 column for COLONY. There are 5 occasions, so 4 TIME intervals, and thus we need 3 columns to code for TIME.

Finally, we need to code for the various interaction terms as well. Remember, there are interactions of SEX.TIME, COLONY.TIME, SEX.COLONY and SEX.COLONY.TIME.

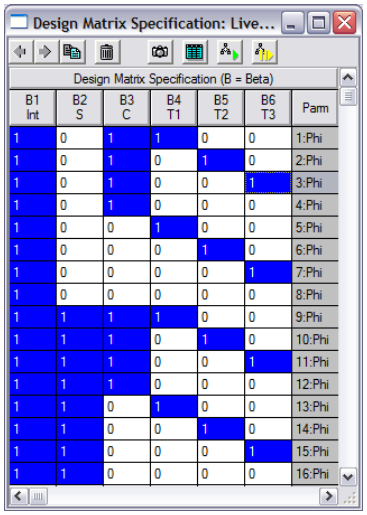
Here is the linear model:

$$\begin{aligned}\text{logit}(\varphi) = \beta_1 &+ \beta_2(\texttt{SEX}) + \beta_3(\texttt{COLONY}) \\ &+ \beta_4(\texttt{T}_1) + \beta_5(\texttt{T}_2) + \beta_6(\texttt{T}_3) \\ &+ \beta_7(\texttt{S.T}_1) + \beta_8(\texttt{S.T}_2) + \beta_9(\texttt{S.T}_3) \\ &+ \beta_{10}(\texttt{C.T}_1) + \beta_{11}(\texttt{C.T}_2) + \beta_{12}(\texttt{C.T}_3) \\ &+ \beta_{13}(\texttt{S.C}) \\ &+ \beta_{14}(\texttt{S.C.T}_1) + \beta_{15}(\texttt{S.C.T}_2) + \beta_{16}(\texttt{S.C.T}_3).\end{aligned}$$

So, we see that there are 16 β terms for φ, plus 1 β term for the constant encounter probability p, for a total of 17 columns (looks like our guess was correct). Now, let's build the design matrix.

We'll start by adding the INTCP, SEX, COLONY and TIME columns to the design matrix. As you no doubt by now realize, there is no 'hard rule' for how you code the various effects in the design matrix – as long as the coding, and the number of columns, are consistent with the model you're trying to fit, and the data in the .INP file.

Here is one possible dummy variable coding for these terms:



B1 Int	B2 S	B3 C	B4 T1	B5 T2	B6 T3	Parm
1	0	1	1	0	0	1:Phi
1	0	1	0	1	0	2:Phi
1	0	1	0	0	1	3:Phi
1	0	1	0	0	0	4:Phi
1	0	0	1	0	0	5:Phi
1	0	0	0	1	0	6:Phi
1	0	0	0	0	1	7:Phi
1	0	0	0	0	0	8:Phi
1	1	1	1	0	0	9:Phi
1	1	1	0	1	0	10:Phi
1	1	1	0	0	1	11:Phi
1	1	1	0	0	0	12:Phi
1	1	0	1	0	0	13:Phi
1	1	0	0	1	0	14:Phi
1	1	0	0	0	1	15:Phi
1	1	0	0	0	0	16:Phi

Look closely. The INCTP is coded by a column of 16 '1's. There are 4 intervals, and 4 attribute groups, so 16 underlying φ parameters. Next, the SEX column (labeled S). We'll let '1' represent males, and '0' represent females. Since the first 2 frequency columns in the .INP file represent females, then the first 8 elements of the SEX column are 0's, followed by 8 1's for the males. Next, the COLONY column (labeled

C). Now, remember that in the `.INP` file, the frequency columns were 'good' and 'poor' colonies for the females, followed by 'good' and 'poor' colonies for the males. Here, we've used a '1' to indicate the 'good' colony, and a '0' to represent the 'poor' colony, alternating for each sex. Next, the 3 columns coding for `TIME` (labeled `T1`, `T2` and `T3`), in the standard way (using reference cell coding) – here, we've set the final time interval (between occasion 4 and 5) as the reference interval.

What about interactions? Well, let's start with the easy ones: `S.T`, `C.T`, and `S.C`. Look closely at the following figure:

Design Matrix Specification (B = Beta)

B1 Int	B2 S	B3 C	B4 T1	B5 T2	B6 T3	B7 S.T1	B8 S.T2	B9 S.T3	B10 C.T1	B11 C.T2	B12 C.T3	B13 S.C	Parm
1	0	1	1	0	0	0	0	0	1	0	0	0	1:Phi
1	0	1	0	1	0	0	0	0	0	1	0	0	2:Phi
1	0	1	0	0	1	0	0	0	0	0	1	0	3:Phi
1	0	1	0	0	0	0	0	0	0	0	0	0	4:Phi
1	0	0	1	0	0	0	0	0	0	0	0	0	5:Phi
1	0	0	0	1	0	0	0	0	0	0	0	0	6:Phi
1	0	0	0	0	1	0	0	0	0	0	0	0	7:Phi
1	0	0	0	0	0	0	0	0	0	0	0	0	8:Phi
1	1	1	1	0	0	1	0	0	1	0	0	1	9:Phi
1	1	1	0	1	0	0	1	0	0	1	0	1	10:Phi
1	1	1	0	0	1	0	0	1	0	0	1	1	11:Phi
1	1	1	0	0	0	0	0	0	0	0	0	1	12:Phi
1	1	0	1	0	0	1	0	0	0	0	0	0	13:Phi
1	1	0	0	1	0	0	1	0	0	0	0	0	14:Phi
1	1	0	0	0	1	0	0	1	0	0	0	0	15:Phi
1	1	0	0	0	0	0	0	0	0	0	0	0	16:Phi

For convenience, we've labeled the columns in the design matrix so you can see which columns correspond to which interaction terms: columns 7 → 9 correspond to the `SEX.TIME` interactions, columns 10 → 12 correspond to the `COLONY.TIME` interactions, and column 13 corresponds to the `SEX.COLONY` interaction.

Finally, the `S.C.T` interaction, indicated in columns 14 → 16 in the following:

Design Matrix Specification (B = Beta)

B1 Int	B2 S	B3 C	B4 T1	B5 T2	B6 T3	B7 S.T1	B8 S.T2	B9 S.T3	B10 C.T1	B11 C.T2	B12 C.T3	B13 S.C	B14 S.C.T1	B15 S.C.T2	B16 S.C.T3	Parm	B17 p
1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1:Phi	0
1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	2:Phi	0
1	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	3:Phi	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	4:Phi	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5:Phi	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	6:Phi	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	7:Phi	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8:Phi	0
1	1	1	1	0	0	1	0	0	1	0	0	1	1	0	0	9:Phi	0
1	1	1	0	1	0	0	1	0	0	1	0	1	0	1	0	10:Phi	0
1	1	1	0	0	1	0	0	1	0	0	1	1	0	0	1	11:Phi	0
1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	12:Phi	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	13:Phi	0
1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	14:Phi	0
1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	15:Phi	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17:p	1

The element in the lower right-hand corner of the completed design matrix codes for the constant encounter probability, p.

Now, go ahead and run this model, and label it '`phi(S.C.T)p(.) - DM`', with DM indicating it was constructed using a design matrix, and add the results to the browser (top of the next page).

As expected, the two models yield identical results: the number of parameters and the model deviance

{phi(S.C.T)p(.) - PIM}	15449.2015	0.0000	0.50000	1.0000	17	100.8314
{phi(S.C.T)p(.) - DM}	15449.2015	0.0000	0.50000	1.0000	17	100.8314

is the same, regardless of whether or not the model was built using the PIMs, or via the design matrix.

As a final check, though, we want to look at the real estimates (below) for our parameters with those from the model constructed using the design matrix. We see that the results are identical, as expected:

multiple groups - MARK book

Real Function Parameters of {phi(S.C.T)p(.) - DM}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.4640515	0.0286231	0.4086343	0.5203709
2:Phi	0.8718169	0.0249613	0.8144710	0.9133238
3:Phi	0.4526300	0.0205268	0.4127914	0.4930849
4:Phi	0.7648542	0.0272864	0.7072608	0.8140951
5:Phi	0.5411491	0.0282498	0.4854992	0.5957911
6:Phi	0.9522426	0.0217730	0.8863802	0.9807552
7:Phi	0.4989973	0.0200901	0.4597081	0.5382989
8:Phi	0.7742729	0.0261610	0.7189413	0.8214173
9:Phi	0.5087658	0.0284463	0.4531329	0.5641825
10:Phi	0.8675647	0.0211225	0.8204310	0.9037776
11:Phi	0.7401698	0.0209846	0.6969863	0.7791491
12:Phi	0.6864606	0.0238297	0.6379830	0.7311818
13:Phi	0.5481091	0.0279462	0.4929725	0.6020895
14:Phi	0.9065043	0.0179715	0.8648389	0.9362722
15:Phi	0.8475676	0.0189659	0.8065778	0.8811509
16:Phi	0.7312077	0.0227421	0.6843781	0.7733898
17:p	0.7359361	0.0081473	0.7196610	0.7515924

Handling > 1 classification variable can sometimes be a bit tricky – **but** most of the challenges are 'book-keeping'. Just remember that the dummy variable coding in the design matrix needs to be consistent with both the linear model you're trying to fit, and the structure of the `.INP` file.

6.12. Time + Group – building additive models

Most newcomers to **MARK** find building design matrices the most daunting part of the 'learning curve'. However, if you've understood everything we've covered up to now, you should find building design matrices for additive models very straightforward. What do we mean by an 'additive model'? As you may remember from earlier chapters, an additive model is one where we express variation in survival or encounter as a function of the additive contributions of 2 or more factors.

Recall our comparison of the 'good' and 'poor' swift colonies (Chapter 4). Our linear model was represented by

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{COLONY}) + \beta_3(\texttt{TIME}) + \beta_4(\texttt{COLONY.TIME}).$$

In this model, we have two 'factors' – `COLONY` ('good' and 'poor') and `TIME` ($\varphi_1, \varphi_2 \ldots \varphi_7$). Each 'time interval' is considered as a different level of the `TIME` factor. In this case, we are treating time as a 'categorical' variable, as opposed to a quantitative linear covariate as we did in the 'trend' example presented earlier.

You may have noted that this is, in fact, the default **MARK** CJS model with 2 groups – as long as you tell **MARK** to use the same parameter structure between groups, but let the parameter values differ (i.e., same qualitative structure between the PIMs, but different indexing), then **MARK** uses the 'full model', with both factors (`COLONY` and `TIME`), and the interaction term (`COLONY.TIME`). When we run **MARK**, but

set the PIMs for the 2 groups to be the same (same structure, same index values), we are testing model

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{TIME}).$$

The difference between these two models is, in fact, the effect of `COLONY + COLONY.TIME`, not just `COLONY` alone. As noted in Chapter 4, we are, in fact, leaving out the 'intermediate model':

$$\text{logit}(\varphi) = \beta_1 + \beta_2\texttt{COLONY} + \beta_3\texttt{TIME}.$$

Here, we are considering the additive effects of the 2 factors – hence, we refer to it as an 'additive' model. To fit this model we simply need to take our design matrix for the fully time-dependent model

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{COLONY}) + \beta_3(\texttt{TIME}) + \beta_4(\texttt{COLONY.TIME}),$$

and delete the interaction columns! It's that easy! In fact, we already saw this earlier when we analyzed the dipper data with the flood model. Go back to the swift analysis, pull up the design matrix for the full model $\{\varphi_{g*t}\, p_{g*t}\}$, and simply delete the columns corresponding to the interaction of group and time (for survival).

The design matrix (the upper-left quadrant for survival) should look like the following:

B1 Phi Int	B2 Phi g1	B3 Phi t1	B4 Phi t2	B5 Phi t3	B6 Phi t4	B7 Phi t5	B8 Phi t6	Parm	
1	1	1	0	0	0	0	0	1:Phi	0
1	1	0	1	0	0	0	0	2:Phi	0
1	1	0	0	1	0	0	0	3:Phi	0
1	1	0	0	0	1	0	0	4:Phi	0
1	1	0	0	0	0	1	0	5:Phi	0
1	1	0	0	0	0	0	1	6:Phi	0
1	1	0	0	0	0	0	0	7:Phi	0
1	0	1	0	0	0	0	0	8:Phi	0
1	0	0	1	0	0	0	0	9:Phi	0
1	0	0	0	1	0	0	0	10:Phi	0
1	0	0	0	0	1	0	0	11:Phi	0
1	0	0	0	0	0	1	0	12:Phi	0
1	0	0	0	0	0	0	1	13:Phi	0
1	0	0	0	0	0	0	0	14:Phi	0
0	0	0	0	0	0	0	0	15:p	1

What **MARK** does with this design matrix is to estimate a coefficient ('slope') for each dummy variable. This coefficient tells us how any one particular level differs from the baseline level (i.e., the last time interval). We can put all the coefficients together in one regression equation (for `COLONY` and `TIME`):

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{COLONY}) + \beta_3(\texttt{t}_1) + \beta_4(\texttt{t}_2) + \beta_5(\texttt{t}_3) + \beta_6(\texttt{t}_4) + \beta_7(\texttt{t}_5) + \beta_8(\texttt{t}_6).$$

In this expression, β_1 tells us how much, on average, survival in the 'good' colony differs from survival in the 'poor' colony. Note that when `COLONY` = 0 (say, for the 'poor' colony), the β_2 term drops out of the regression equation, resulting in our estimating survival in the 'good' colony. Each of the remaining β-terms specifies how much survival in one year period (average over both colonies) differs from the baseline year. The greater the magnitude of a particular β the greater that year's survival probability differs from the baseline year, and the greater the statistical significance of a particular β, the greater the contribution to the overall significance of the `TIME` factor.

It should be easy to see that, for example, for the 'poor' colony (COLONY = 0) in year 3, the β_2 term drops out of the equation, and we are left with

$$\text{logit}(\varphi) = \beta_1 + \beta_5,$$

because $t_3 = 1$, and all other t values ($t_1, t_2, \ldots, t_6$) are equal to zero. Thus, β_5 tells us how much survival in year 3 differed from the baseline year (year 7).

Similarly, the equation for year 5 in the 'good' colony (COLONY = 1) would be

$$\text{logit}(\varphi) = \beta_1 + \beta_2 + \beta_7.$$

One last thing to consider. Counting parameters for the additive model (model $\{\varphi_{g+t}\, p_{g*t}\}$) is not quite as simple as for other models. First, we'll consider how to count the number of potentially available parameters in an additive model. The easiest way to do it is to remember what we're after – we're testing a model where there is time variation in survival for both colonies, but that the difference between colonies is due to a constant, additive, component. This additive component is simply 1 more parameter (like estimating a constant). This should not be surprising, especially if you think of the additive model in the ANCOVA example we dealt with earlier – in the Lebreton *et al*. (1992) monograph, they give you an explicit 'hint' when they use the word 'parallelism'. If any pair of lines in an $X - Y$ plane are parallel then for any value X, the two lines differ in Y by some constant amount. It is the constant 'difference' between the lines which constitutes one of the estimable parameters.

So, for our present example, simply count the number of parameters you would expect for 1 colony alone, for the underlying model (CJS – 13 parameters for either colony alone). Then, add 1 for the additivity (i.e., the 'constant difference') in survival (you would add 2 if you had additivity in survival and encounter simultaneously).

Now, the tricky part (potentially) – if survival in one colony is simply survival in the other colony plus a constant, then the survival probability is identifiable (by linear interpolation) for all intervals in the second colony, and thus all of the encounter probabilities are estimable (no confounding of terminal p and φ parameters). So, since there are 7 encounters, we add 7, bringing our total to (13 + 1 + 7) = 21 total parameters.

Still don't get it? Here's another way to think of it. First, in this example, we are constraining the CJS model by colony. What was estimable in this starting model will remain estimable in the same model with a constraint – in this case, the additive model. Thus, if some parameters are not identifiable in the additive model, they must be those from the last time interval: $\varphi_{7,g}$ and $\varphi_{7,p}$ and $p_{8,g}$ and $p_{8,p}$ (where 'g' = good, and 'p' = poor). Let's focus our attention on them. In the unconstrained CJS model, we could identify the products $\beta_{9,g} = (\varphi_{7,g} p_{8,g})$ and $\beta_{9,p} = (\varphi_{7,p} p_{8,p})$.

In essence, we had 2 equations and 4 unknowns. If we pick some value for (say) $\varphi_{7,g}$, then we can solve for $p_{8,g}$. Similarly, we could pick an arbitrary value for $\varphi_{7,p}$ and solve for $p_{8,p}$. Thus, we have 2 arbitrary parameters to discount from the initial total of 28 parameters in the model – in other words, (28 – 2) = 26 identifiable parameters. Of course, we knew this already – we simply want to confirm that this approach yields the same results.

Now, let's apply this same line of reasoning to the additive model case. We still have the same 2 equations as before, plus 1 new one; $\varphi_{7,g} = \varphi_{7,p} + c$ (from the constraint). 'c' is a known constant, because 'c' is common to all intervals. Therefore, if we pick a value for $\varphi_{7,g}$ then $\varphi_{7,p}$ is known, and thus also both $p_{8,g}$ and $p_{8,p}$. Thus, we have just one arbitrary parameter to discount from the total number of parameters originally included in the additive model; for survival, 7 slopes + 1 intercept = 8, and for encounter, 14. Thus, (8 + 14) = 22 – 1 (the arbitrary parameter) = 21 identifiable parameters. However, if we run the additive model, we find that **MARK** has estimated only 19 parameters. It has 1 intercept,

and 7 slopes, but of the encounter parameters, 2 are not identifiable – so (1 + 7 + 12) = (20 − 1) (the arbitrary parameter) = 19.

In fact, Lebreton *et al*. (1992) analyzed a large number of different models for this data set (see Table 14). They found that the most parsimonious model has constant survival within colony, but different colony values, and additivity ('parallelism') in encounter probabilities, $\{\varphi_c\, p_{c+t}\}$.

Before we leave this chapter, it is important to keep in mind that the 'parallelism' we have been discussing refers only to the logit scale – i.e., it is not *linear* parallelism, but *logit* parallelism. Thus, if you plot the reconstituted values from an additive model, they may not 'look' to be parallel, but on the logit scale, they indeed are.

In addition, you need to think carefully about the 'biological realism' of additive models. Suppose we had found additivity of survival between colonies for the swift data. What would this mean? It would mean that whatever factor made the survival differ overall between the colonies had a constant additive effect over time. This would imply that there is some 'real difference', possibly genetic or age, between the two colonies such that, although both of them are subject to fluctuations over time, one colony always does relatively better (or worse) than the other (and by a constant amount on the logit scale). Is this 'biologically plausible'? Perhaps. Or, perhaps not. But, even if an additive model is not 'biologically plausible', it represents a model that is 'intermediate' between a model with no group (colony) effects (`time` only), and one with full (`group` × `time`) interactions. Such a an 'intermediate' model might be parsimonious, even if not 'biologically plausible'. You will need to justify for yourself if 'additive' models should be included in your candidate model set.

6.13. Linear models and 'effect size': a test of your understanding...

Recall that in chapter 4, we introduced the question of 'significance' and 'effect size'. We considered the 'swift analysis', and asked the question: is there a difference in survival between the colonies (good colony versus poor colony), and is it 'significant'? In addressing these questions, we considered the matter of 'effect size'. In the swift analysis, we concluded that there was good support for the contention that there is a difference in survival between the two colonies, based on relative AIC model weights. The remaining questions were – how big is this difference, and is the difference 'biologically meaningful'? As we note in chapter 4, the first question relates to 'effect size' – we consider colony as an 'effect', strictly analogous to an 'effect' in ANOVA. The 'effect size' is the estimate of the magnitude of the difference in survival between the two colonies. Further, since the effect size is 'estimated', it will have an associated uncertainty which we can specify in terms of a confidence interval (CI).

The key question then becomes

> *'what are the plausible bounds on the true effect size, and are biologically important effects contained within these bounds?'*

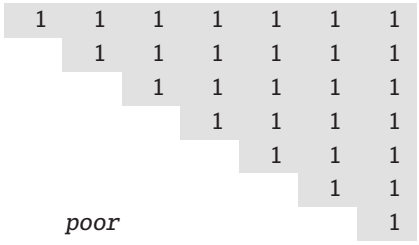
In chapter 4, we concentrated on simple interpretation of effect size. Here, we explore this a little more deeply, introducing some of the considerations involved in estimating effect size in the first place. This exercise will also serve as a test of your basic understanding of linear models (the subject of this chapter).

We'll consider a situation similar to the 'swift analysis' we introduced earlier. We'll imagine there are 2 colonies (`good` and `poor`), and that survival over a given interval in the `poor` colony is 10% lower than survival in the `good` colony over that same interval (warning: keep awake for scaling issues here!). We simulated some data, 8 occasions, 150 newly marked birds released in each colony on each occasion. We assumed a constant survival probability over time for each colony: 0.80 for the `poor` colony, and

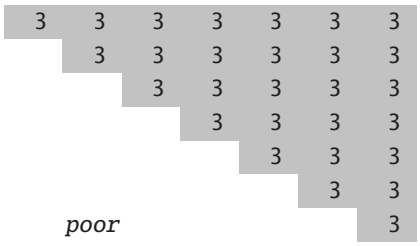
(0.80 + 10%) = 0.88 for the good colony. We also assumed a constant encounter probability of 0.75 for both colonies. The simulated data are contained in **effect_size.inp** (where the first frequency column corresponds to the poor colony, and the second frequency column corresponds to the good colony). Again, 2 groups (poor and good) and 8 occasions.

While you could fit this model by (i) building the fully time-dependent model $\{\varphi_{g*t}\, p_{g*t}\}$ using PIMs, (ii) constructing the corresponding design matrix, and then (iii) reducing the design matrix by eliminating the columns involving TIME for φ, and the columns involving both TIME and COLONY for p, for this demonstration it's easier to simply start with a PIM structure that corresponds to our underlying model, $\{\varphi_g\, p_.\}$.

Here are the PIMs for the survival



and encounter probabilities:

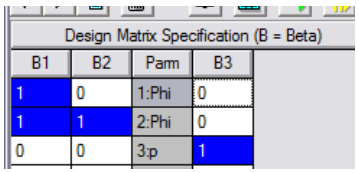


For this analysis, we're primarily interested in estimating the 'effect size' – effect of colony on survival. How do we do that? The answer in this case is fairly easy if you consider the linear model corresponding to this particular analysis. We have 2 groups (COLONY), with no variation over time. Thus, our linear model for survival would be:

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{COLONY}).$$

If you've read this far, and understood the theory behind linear models, you'll recall that β_1 and β_2 together code for the colony effect. How this coding works depends on the design matrix used.

For example, if you use the following design matrix coding

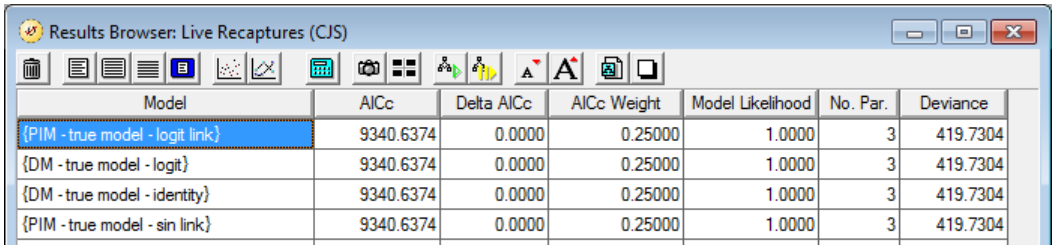


then how do we interpret the intercept (β_1) and the first slope (β_2)? If the poor colony is coded in the 2nd

column (B2 column) of the design matrix as '0', and we use '1' for the good colony, then if the colony is poor, the intercept gives the survival for the poor colony (since the β_2 term drops out of the equation).

So, if the intercept is the estimate for the poor colony, then when the dummy variable is '1' (specifying a good colony), then $\beta_1 + \beta_2 = (\text{poor}) + (\text{good-poor}) = \text{good}$. Clearly – the β_2 value is the *effect* of colony – it is the degree to which estimated survival for the poor colony differs from estimated survival for the good colony. In other words, the estimate for β_2 is the estimate of the effect size.

How do we actually get an estimate of the effect size on the familiar probability scale (i.e., in the range $[0, 1]$)? In fact, we can do this in a couple of ways. Let's start by using the identity link function. Recall that in many (perhaps most) cases, we fit models using either the logit or sin link functions. For now, though, let's re-run our model, using the identity link, which you select during the numerical estimation part of the run. Our general starting model will be $\{\varphi_{colony}\, p_.\}$. Go ahead and run it using either the default sin link, or the logit link. Then, run the model a second time using the identity link.



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{PIM - true model - logit link}	9340.6374	0.0000	0.25000	1.0000	3	419.7304
{DM - true model - logit}	9340.6374	0.0000	0.25000	1.0000	3	419.7304
{DM - true model - identity}	9340.6374	0.0000	0.25000	1.0000	3	419.7304
{PIM - true model - sin link}	9340.6374	0.0000	0.25000	1.0000	3	419.7304

Note for these data that the model fit is identical regardless of whether you use the logit, sin or identity link (although by now you probably appreciate that this is not always the case).

OK, on to the next step. We want the estimate for β_2 – the slope for the colony term in the design matrix, which we understand to be the effect size – in this case, the difference between the good and poor colonies. All you need to do is look at the '`beta estimates`' for the model with the identity link. The estimated value for $\hat{\beta}_2 = 0.0845106$, with a 95% CI of [0.0635335, 0.10548780]. For completeness, the estimate of the intercept is $\hat{\beta}_1 = 0.7897399$. So, the linear model is

$$\begin{aligned} \text{'survival'} = \text{logit}(\hat{\varphi}) &= \hat{\beta}_1 + \hat{\beta}_2(\text{'colony'}) \\ &= 0.7897399 + 0.0845106(\text{'colony'}). \end{aligned}$$

Now, what do these numbers tell us? Well, recall that the estimate for $\hat{\beta}_1$ is the estimate of survival for the poor colony. Look back at the parameter values used in the simulation for the poor colony: the 'true' value for the survival probability for the good colony is 0.8 – our estimated value for the intercept, $\hat{\beta}_1 = 0.79$, is very close. What about effect size? Well a 10% difference in survival corresponds to an absolute difference of 0.08 (since 0.8 is 10% smaller than 0.88, the true survival probability of the good colony). Since β_1 corresponds to the poor colony, then β_2 is the deviation ('effect') of the good colony on survival – in this case, $\hat{\beta}_2 = 0.0845106$ (the positive sign indicating an increase in survival for the good colony, relative to the poor colony). The estimate of $\hat{\beta}_2 = 0.0845106$ is quite close to the true difference (effect size) of 0.08 (the true difference clearly falls within the 95% CI for the estimate). Recall that these are simulated data, so we anticipate some difference between estimated and 'true' parameter values.

So, the estimate of the effect of colony on survival is $\hat{\beta}_2 = 0.0845106$. Is this 'significant'? The 95% CI for the estimate of the effect size ranges approximately from 0.064 to 0.105. Since this 95% CI doesn't bound 0, then we might conclude that there is a '*statistically* significant' effect of colony on survival. What about '*biologically* significant' – isn't that more relevant than 'statistical' significance?

Here is where 'biological insight' comes into play. Whether or not the effect estimated in this study

is 'significant or not' depends on *your* thoughts on what is or is not 'biologically significant'. Suppose, for example, that you decide *a priori* that a difference in survival between the colonies of ≥ 10% to be 'biologically significant', then in this case, our results would be considered as 'biologically inconclusive' (despite being 'statistically significant'), since the 95% CI includes values < 10%. If instead we believed that a difference in survival of 5% was 'biologically important', then we could conclude with 95% confidence that in this case there was a biologically significant result, since the 95% CI do not include this value (since the lower CI is > 5%).

Some additional comments. In the preceding, we used the identity link. We did so for convenience – the identity link gave us an estimate of the absolute value of the effect size directly, on the $[0, 1]$ probability scale we're typically interested in. But, what if the numerical estimation 'doesn't work' with the identity link (typically, because of difficulties with convergence)? Can we use the logit or sin link to get estimates of the effect size, and the standard error?

The answer is 'yes', but it does require a bit more work. Let's run the analysis of the simulated data using the logit link. The estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$ on the logit scale are 1.3233584 and 0.6157168, respectively. Remembering that the linear model we're fitting is

$$\text{logit}(\hat{\varphi}) = \hat{\beta}_1 + \hat{\beta}_2(\texttt{colony}),$$

then since β_1 = (poor), and $\beta_1 + \beta_2$ = (poor + *effect* of good), we can write

$$\begin{aligned}\text{effect of 'good'} &= \left(\frac{e^{\hat{\beta}_1+\hat{\beta}_2(1)}}{1+e^{\hat{\beta}_1+\hat{\beta}_2(1)}}\right) - \left(\frac{e^{\hat{\beta}_1}}{1+e^{\hat{\beta}_1}}\right) \\ &= 0.8742505 - 0.789740 \\ &= 0.0845106,\end{aligned}$$

which is exactly the same value as the one estimated (for β_2) using the identity link.

What about the SE of the estimated effect size? As noted earlier, the discussion of effect size is really a discussion of whether or not the confidence limits on the effect size bound a difference that we think is 'biologically significant'. From the identity link analysis, we know that the SE and confidence limits to our effect size are 0.0107 and [0.0635, 0.1055], respectively.

We can derive the same values using the estimates from the logit link analysis, but it requires a few more steps. First, recall that the variance of a difference (of say two parameters θ_i and θ_j) is

$$\text{var}(\theta_i - \theta_j) = \text{var}(\hat{\theta}_i) + \text{var}(\hat{\theta}_j) - 2\,\text{cov}(\hat{\theta}_i, \hat{\theta}_j).$$

Thus, the estimated SE for the difference (i.e., effect size) between the 'good' and 'poor' colony is

$$\sqrt{\text{var}(\texttt{good}) + \text{var}(\texttt{poor}) - 2\text{cov}(\texttt{good},\texttt{poor})},$$

where the variance and covariances of the two estimates can be output directly in **MARK**.

To do this, select the appropriate model in the results browser (in this case, the logit link model). Then, select '**Output | Specific Model Output | Variance-Covariance matrices | Real Estimates**', and output the values to a Dbase file (to maintain full numerical precision). The variance-covariance values for the estimates of interest (i.e., good and poor) on the real probability scale are: $\widehat{\text{Var}}$(poor) = 0.00007377, $\widehat{\text{Var}}$(good) = 0.00004534, and $\widehat{\text{cov}}$(poor,good) = 0.0000022834.

Thus, our estimate of the SE for the effect size is

$$\sqrt{0.00004534 + 0.00007377 - 2(0.0000022834)} = 0.0107,$$

which is the same as the estimate using the identity link (to within rounding error). The estimated 95% CI would then be (effect size ± 2SE), or $0.0846 \pm 2(0.0107) = [0.0632, 0.1060]$, which is virtually identical to the estimated 95% CI derived using the identity link (again, to within rounding error).

begin sidebar

Variance of a difference – say what??

Where does this formula for the variance of a sum, or a difference, come from? Well, if $A = X_1 + X_2$ for example, then

$$\begin{aligned}
s_A^2 &= \frac{1}{n}\sum(A - \bar{A})^2 = \frac{1}{n}\sum\left[(X_1 + X_2) - \frac{1}{n}(X_1 + X_2)\right]^2 \\
&= \frac{1}{n}\sum\left[(X_1 + X_2) - \frac{1}{n}\sum(X_1) + \frac{1}{n}\sum(X_2)\right]^2 \\
&= \frac{1}{n}\sum\left[(X_1 + X_2) - \bar{X}_1 - \bar{X}_2\right]^2 \\
&= \frac{1}{n}\sum\left[(X_1 - \bar{X}_1) + (X_2 - \bar{X}_2)\right]^2 \\
&= \frac{1}{n}\sum\left[x_1 + x_2\right]^2 \\
&= \frac{1}{n}\sum\left[x_1^2 + x_2^2 + 2x_1x_2\right] \\
&= s_1^2 + s_2^2 + 2s_{12} \\
&= \mathrm{var}(X_1) + \mathrm{var}(X_2) + 2\mathrm{cov}(X_1, X_2).
\end{aligned}$$

Similarly, if $D = (X_1 - X_2)$ (i.e., a difference rather than a sum), then

$$s_D^2 = \mathrm{var}(X_1) + \mathrm{var}(X_2) - 2\mathrm{cov}(X_1, X_2).$$

Now, a more intuitive explanation. You may recall that the variance of a sum is equivalent to the sum of the variances, *if the elements are independent* (you should be able to prove this to yourself fairly easily).

We can write this as

$$\mathrm{var}\left(\sum \hat{x}_i\right) = \sum \mathrm{var}(\hat{x}_i)\,.$$

But, if there is any sampling covariance (i.e., if the terms are dependent), then we write

$$\mathrm{var}\left(\sum \hat{x}_i\right) = \sum \mathrm{var}(\hat{x}_i) + \sum_i \sum_j \mathrm{cov}(\hat{x}_i, \hat{x}_j).$$

Thus, 'intuitively', given two values x_i and x_j, we write

$$\mathrm{var}(\hat{x}_i + \hat{x}_j) = \mathrm{var}(\hat{x}_i) + \mathrm{var}(\hat{x}_j) + 2\,\mathrm{cov}(\hat{x}_i, \hat{x}_j),$$

or, equivalently for a difference,

$$\mathrm{var}(\hat{x}_i - \hat{x}_j) = \mathrm{var}(\hat{x}_i) + \mathrm{var}(\hat{x}_j) - 2\,\mathrm{cov}(\hat{x}_i, \hat{x}_j).$$

And, if that wasn't enough, we can also derive this expression using the *Delta method* (Appendix B).

For example, if $D = (X_1 - X_2)$ (i.e., the difference between X_1 and X_2), then we can show that if the variances of the X_i are small (and this is an important assumption), then to first-order

$$\text{var}(\mathbf{Y}) = \mathbf{D}\Sigma\mathbf{D}^{\mathsf{T}},$$

where $\mathbf{D}$ is a row-vector of the partial derivative of the function (in this case, $D = X_1 - X_2$) with respect to each variable (i.e., X_1 and X_2, respectively), $\mathbf{D}^{\mathsf{T}}$ is the column-vector transpose of $\mathbf{D}$, and Σ is the variance-covariance matrix of X_1 and X_2. The concepts underlying this expression are presented in detail in Appendix B.

Thus,

$$\begin{aligned}
\text{var}(\mathbf{D}) &= \mathbf{D}\Sigma\mathbf{D}^{\mathsf{T}} \\
&= \begin{bmatrix} \frac{\partial D}{\partial X_1} & \frac{\partial D}{\partial X_2} \end{bmatrix} \begin{bmatrix} \text{var}(X_1) & \text{cov}(X_1, X_2) \\ \text{cov}(X_2, X_1) & \text{var}(X_2) \end{bmatrix} \begin{bmatrix} \frac{\partial D}{\partial X_1} \\ \frac{\partial D}{\partial X_2} \end{bmatrix} \\
&= [1 \quad -1] \begin{bmatrix} \text{var}(X_1) & \text{cov}(X_1, X_2) \\ \text{cov}(X_2, X_1) & \text{var}(X_2) \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\
&= \text{var}(X_1) + \text{var}(X_2) - 2\,\text{cov}(X_1, X_2),
\end{aligned}$$

which is identical to the expression we introduced on the preceding page.

end sidebar

Now, let's consider a slightly more complex example – same basic scenario, but now with 3 colonies, instead of 2. Again, we simulate a data set (**effect_size3.inp**) with 3 colonies. Assuming constancy of survival over time for all 3 colonies, but let the survival differ among the colonies: for colony 1, 0.65, for colony 2, survival is 10% higher (i.e., 0.715), and in colony 3, survival is 15% higher than in colony 1 (i.e., 0.7475). Thus, colony 3 has a survival probability that is 4.55% higher than colony 2. Please note the scale – we're speaking of differences in terms of percentages – not arithmetic differences.

So, for example, is a 10% difference in survival between colony 1 and colony 2 biologically meaningful? You need to think carefully about scaling, since a 10% increase in survival from a reference value of 0.5 (to 0.55) is different (arithmetically) than a 10% increase from a reference value of (say) 0.7 (to 0.77). The arithmetic difference is 0.05 in the first case, and 0.07 in the second case.

However, the effect size we're working with (as in the preceding example) is on the 'real' scale – it is not proportional (or, in terms of percent differences). So, for the present example, the effect (difference) between colony 1 and colony 2 is (0.715−0.650) = 0.065, between colony 1 and colony 3 is (0.7475−0.65) = 0.0975, and between colony 2 and colony 3 is (0.7475 − 0.715) = 0.0325. We set $p = 0.75$, and released 500 individuals on each occasion, for 8 occasions.

Let's see if we can derive estimates for the effect sizes. We'll save ourselves a few steps by simply going ahead and fitting the true model, which is $\{\varphi_{colony}\, p.\}$. If we do this using the PIM chart approach (the quickest way to fit this model), we get estimates of survival of 0.6600 for colony 1, 0.7211 for colony 2, and 0.7447 for colony 3, which are all fairly close to the 'true' values specified in the simulation. So, if we wanted to quickly derive estimates of the effect size, we have to do no more than pull out our calculator, and take the pair-wise differences among these 3 values: so the effect size between colony 1 and colony 2 is (0.7211 − 0.6600) = 0.0611, between colony 1 and colony 3 is (0.7447 − 0.6600) = 0.0847, and between colony 2 and colony 3 is (0.7447 − 0.7211) = 0.0236. Again, these are all close to the 'true' differences expected given the values using in the simulations.

How would we get these estimates in a more 'elegant' fashion (i.e., other than by simply calculating the arithmetic difference)? Well, first, we need to specify the model using a design matrix. But, as we've seen earlier, there are a number of ways in which such a design matrix could be constructed. In this case, we're interested in looking at the magnitude of differences among levels of an effect. This is most easily done using an intercept-based model (remember the preceding example where the effect size was measured as the relative difference between the intercept term and the colony factor term in the linear model).

In this example, we have 3 levels of the 'colony' effect, so we need 2 columns of dummy variables to code for this. Thus, our linear model would have an intercept term, plus 2 other terms to specify the colony. This much should be familiar (given that you've made it this far in the chapter). However, how should you code the different colonies? The following 3 design matrices (for the survival parameter; for now, we'll ignore encounter rate) are all equivalent in terms of the 'results' (i.e., the colony-specific estimates of survival), but have different interpretations:

β_1	β_2	β_3
1	0	0
1	1	0
1	0	1

β_1	β_2	β_3
1	1	0
1	0	0
1	0	1

β_1	β_2	β_3
1	1	0
1	0	1
1	0	0

The differences among these design matrices have to do with what colony is used as the 'reference' or 'control' colony. In the left-most matrix, the design matrix corresponds to a coding scheme wherein if both β_2 and β_3 are 0, then the intercept corresponds to colony 1. Why? Well, here you need to remember that in the `.INP` file, colony 1 was the first group, colony 2 was the second group, and colony 3 was the third group. In effect, each row in the design matrix corresponds to each of the different colonies. Thus, if both β_2 and β_3 are 0, then the intercept corresponds to colony 1, and as such, both colony 2 and colony 3 are then estimated 'relative' to colony 1 (remember the basic structure of the linear model: intercept + effect terms). Thus, in this case, colony 1 (given by the intercept) is the 'reference' colony – in fact, this sort of design matrix coding is often referred to as 'reference cell' coding, since whichever level of a given treatment is coded by 'all zeros' is the 'reference' (or control) level of the treatment. Got it? If so, then you should see that for the middle matrix, where the row representing colony 2 (the second row) is the reference colony. Finally, in the right-most matrix, colony 3 is the reference colony. Make sure you see this. Note that there is an interaction of the design matrices, and the order in which groups are presented in the `.INP` file.

Why do we care in this case? We care because the effect size in the linear model is calculated relative to the reference level – in this example, relative to the reference colony. Let's see how this works. We'll start by fitting the data to our model, using a design matrix with colony 1 designated as the reference colony (i.e., making use of the structure in the left-most of the 3 example matrices noted above).

Since by now you can probably do this fairly easily, we'll jump right to the 'results'. Using the logit link (the default link whenever the design matrix is modified from the identity matrix), the 'beta' estimates are: $\hat{\beta}_1 = 0.6635$, $\hat{\beta}_2 = 0.2863$, and $\hat{\beta}_3 = 0.4070$. The signs of the estimates indicate that the 'effect' for colony 2 and colony 3 are both positive with respect to colony 1 (the reference colony given our design matrix). Thus, the estimate of survival for both colony 2 and colony 3 are both expected to be bigger than for colony 1 (which is exactly what we expect).

What about the estimates of effect size? Well, here, we need to be somewhat careful. First, recall that if both β_2 and β_3 are 0, then the intercept refers to colony 1. Thus, the effect size between colony 2 and colony 1 is 'colony 2' - 'colony 1' = $(\beta_1 + \beta_2) - (\beta_1)$. Why does $(\beta_1 + \beta_2)$ correspond to colony 2? Note that there is no β_3 term – indicating that $\beta_3 = 0$. Thus, if $\beta_3 = 0$, but both β_1 and β_2 are not 0, then this refers to colony 2.

OK, so the estimate of the effect size for the difference between colony 2 and colony 1, back-transformed from the logit scale, is

$$\frac{e^{\hat{\beta}_1+\hat{\beta}_2}}{1+e^{\hat{\beta}_1+\hat{\beta}_2}} - \frac{e^{\hat{\beta}_1}}{1+e^{\hat{\beta}_1}} = (0.7211 - 0.6600)$$
$$= 0.0610,$$

which is precisely what we calculated 'by hand' from the real estimates of survival for both colonies. We could do the same thing to calculate the difference between colony 1 and colony 3 (try it as a test of your understanding – the effect size (difference) should be 0.0942).

But, what about the difference between colony 2 and colony 3? Well, there are a couple of approaches. First, you could change the design matrix, setting colony 2 or colony 3 as the reference, and then using the same approach as just described, except that you have a different reference colony.

But, in fact, you don't need to go to all that trouble; simply remember that colony 2 is $(\beta_1 + \beta_2)$ and that colony 3 is $(\beta_1 + \beta_3)$. Thus, the difference between colony 2 and colony 3 is

$$\frac{e^{\hat{\beta}_1+\hat{\beta}_2}}{1+e^{\hat{\beta}_1+\hat{\beta}_2}} - \frac{e^{\hat{\beta}_1+\hat{\beta}_3}}{1+e^{\hat{\beta}_1+\hat{\beta}_3}} = (0.2553 - 0.2789)$$
$$= -0.0236.$$

So, estimated survival for colony 2 is −0.0236 lower than estimated survival for colony 3 – exactly what we 'calculated by hand' using the real estimates of survival for colonies 2 and 3. So, obviously, you can get the estimate of 'effect size' right from the 'real estimates', without going through all the effort of getting the β estimates, and back-transforming the equation (as we have done here).

But, what about the SE for the estimates of effect size? Again, since we're dealing with the variance (or SE) of a difference (in this case, between any pair of colonies), we simply output the variance-covariance values for the real estimates. For this example, the variance for β_1 is 0.00004, for β_2 is 0.00003, and for β_3 is 0.00003. Since each of the colonies is 'independent' of each other, we don't anticipate any sampling covariance – this is what we see: Cov(colony 1, colony 2) = Cov(colony 1, colony 3) = Cov(colony 2, colony 3) = 0. Thus, given the variances, the SE for the difference between colony 1 and colony 2 (for example) is

$$\sqrt{(0.00004 + 0.00003 - 0)} = 0.0084.$$

And thus, the approximate 95% CI for the effect (difference) between colony 1 and colony 2 is [0.0442, 0.0778]. In fact, if you fit these data using the identity link, you'll see that this estimated SE matches that given by the identity link almost exactly (remember – while the identity link is generates estimates of the effect size and the SE directly on the normal [0, 1] probability scale, not all data sets can be fit using the identity link – usually due to convergence issues. In such cases, a transformation – like a logit or sin transform – is required).

So, we see that we can fairly easily come up with estimates of the effect size, and the SE of these estimates. We leave it to you to tackle the more difficult (at least conceptually) question of what constitutes a 'biologically meaningful' effect size. However, that 'debate' notwithstanding, we suggest that you routinely consider – and report – effect size where possible.

6.13.1. Linear models: β estimates and odds ratios

The β terms in the linear model are interpretable as the natural log of the *odds ratios*. The odds of 'success' (e.g., survival, movement) is the ratio of the probability of 'success' (say, θ, where θ is some $[0, 1]$ bounded parameter) to the probability of 'failure' (given by the complement, $1 - \theta$). Note we assume binary states – success or failure (live or die, move or stay...).

In other words, the 'log-odds of success' is given by

$$\ln\left(\frac{\theta}{1-\theta}\right),$$

which you should by now recognize is the logit transform of θ.*

The *log-odds ratio* between (say) two levels of some classification (treatment) variable, would be

$$\ln\left(\frac{\theta_1/(1-\theta_1)}{\theta_2/(1-\theta_2)}\right).$$

As written, we see that the odds ratio is a way of comparing whether the probability of a certain event is the same for two groups. A log-odds ratio of 0 implies that the event is equally likely in both groups. A log-odds ratio > 0 implies that the event is more likely in the first group. Conversely, a log-odds ratio < 0 implies that the event is more likely in the second group.

Consider the following example – males and females marked with a radio-telemetry device, which allows us to know the fate of the marked individual with certainty (known-fate analysis is covered in detail in chapter 17). Suppose we mark 60 males, and 45 females. Over the sampling interval, 9 males and 11 females died. Thus, the estimated survival for males is $\hat{S}_m = (51/60) = 0.85$, and for females is $\hat{S}_f = (34/45) = 0.756$, with corresponding log-odds of a male surviving the interval is $\ln(0.85/0.15) = 1.735$ (i.e., the odds of a male surviving the interval is 1.735 to 1), with the log-odds of a female surviving the interval given as $\ln(0.756/0.244) = 1.131$.

The difference in the odds ratio of survival between the two sexes is

$$\begin{aligned}\ln\left(\frac{\hat{S}_m/(1-\hat{S}_m)}{\hat{S}_f/(1-\hat{S}_f)}\right) &= \ln\left(\frac{0.85/0.15}{0.756/0.244}\right)\\ &= \ln(1.829)\\ &= 0.604\end{aligned}$$

Since the log-odds ratio is > 0, then the odds of survival is greater for males than for females.

OK, fine, but how do log-odds ratios connect to the β estimates in a linear model? Let's revisit the 'good colony' versus 'poor colony' analysis we introduced earlier in this section. Recall that we simulated data where survival in the 'good' colony was 0.88, and 0.8 in the 'poor' colony (a 10% difference). Assuming a time-invariant model (which was the true model under which the data were simulated), this corresponds to an *expected* log-odds of survival in the 'good' colony of $\ln(0.88/0.12) = 1.992$, and an *expected* log-odds of survival in the 'poor' colony of $\ln(0.80/0.20) = 1.386$.

Recall that we fit the following linear model to those simulated data:

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{COLONY}).$$

* In fact, the *logit* transform is so named because it transforms probabilities into log odds: log odds → logit; 'log it'.

The actual *estimates* for $\hat{\beta}_1$ and $\hat{\beta}_2$ were 1.3234 and 0.6157, respectively. Since β_1 = (poor), then the *estimated* log-odds for survival in the 'poor' colony is simply 1.3234 (remember, this parameter is estimated on the logit scale), which is fairly close to the expected value of 1.386. For the 'good' colony, the *estimated* log-odds ratio for survival is $(\hat{\beta}_1 + \hat{\beta}_2)$ = (1.3234 + 0.6157) = 1.9391, which again is quite close to the *expected* value of 1.992.

Final step – the *expected* log-odds difference between the two colonies is ln[(0.88/0.12)/(0.80/0.20)] = ln(1.833) = 0.606 – meaning, that for a unit change in 'colony' (whatever that might actually mean – here it means 'poor' to 'good'), the log-odds of survival increases by 0.606 (which when back-transformed from the log scale, means a change in the odds of survival of 1.833). Since β_1 = (poor), and $\beta_1 + \beta_2$ = (poor + *effect* of good), then β_2 is the effect of the 'good' colony. Notice that $\hat{\beta}_2$ = 0.6157, which is close to the expected log-odds ratio of 0.606.

Coincidence? No! The natural log of the odds ratio between the 'good' and 'poor' colonies is a measure of the difference between the two colonies – i.e., the effect size. Since the log-odds ratio in this example is >0, we conclude that the event (survival) is more likely in the 'good' colony (numerator) than in the 'poor' colony (denominator).

It can be helpful to remember that we can easily shift between 'odds' and 'probability'. Go back to the telemetry survival example introduced earlier. For males, we observed 51 individuals surviving over an interval out of 60 released at the start of the interval. Thus, $\hat{S}_m$ = (51/60) = 0.85, while the log-odds of a male surviving the interval is ln(0.85/0.15) = 1.735 (i.e., the log-odds of a male surviving the interval is 1.735 to 1 – when back-transformed from the log scale, this corresponds to $5.\dot{6}$ to 1). So, given the odds of survival of $5.\dot{6}$: 1, we can transform this into a probability as $(5.\dot{6}/(5.\dot{6} + 1)) = 0.85$.

In the preceding examples, we considered effect sizes and odds ratios for discrete levels of a factor (e.g., 'good' versus 'poor' colony). What about interpreting β coefficients for continuous covariates? In fact, there is nothing particularly new to consider – the β estimates give you information about change in log-odds for a unit change in the particular covariate.

But, what about models with interaction terms? Let's re-visit the example introduced in section 6.8 – encounter probability of European dippers was hypothesized to vary as a function of the number of hours of observation in the field, and that the relationship between hours of observation and the encounter probability might differ between males and females. For our analysis, we used the following 'fake' observation data:

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

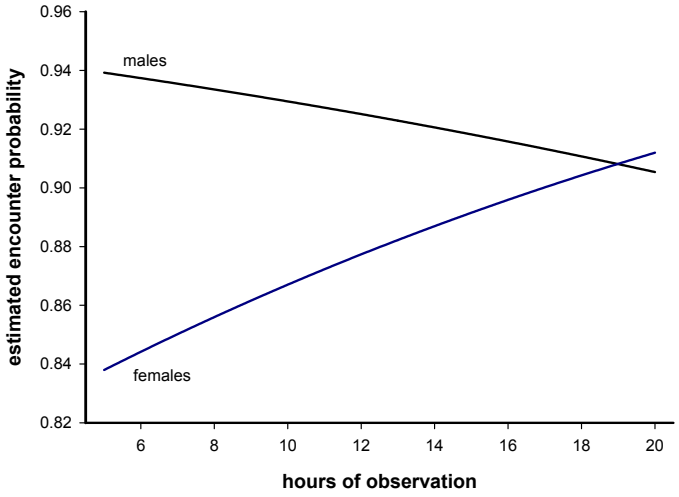
We assumed that survival varied over time, but did not differ between the sexes, φ_t. For the encounter probability, we fitted the following linear model:

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

which yielded the following estimates

$$\text{logit}(\hat{p}_i) = 1.4115996 + 1.4866142(\texttt{SEX}) + 0.0463413(\texttt{HOURS}) - 0.0783021(\texttt{SEX.HOURS}).$$

What is of note, here, is the interaction between SEX and HOURS. As shown in the following figure (top of the next page), the encounter probability decreases with increasing hours of observation for males, but increases for females (before you start trying to come up with some clever 'biological explanation' for this result, remember the observation hours covariate data are 'fake').



What is important to remember about interactions such as shown above is that is makes consideration of the 'significance' of main effects difficult at best – whether encounter probability is higher or lower for (say) males is a function of the value of the HOURS covariate.

But, suppose you were interested in the odds ratio between the sexes for some specific value of HOURS. Or, flipped around, suppose you were interested in the change in odds for a given sex, given a unit change in hours of observation – how would you handle the calculations?

In fact, it really isn't much more difficult that what we've already looked at. First, remember that the odds ratio reflects change in odds for a unit change in some variable. Let's consider the case where we're interested in a change in odds of detection for a particular sex, given a unit change in the number of hours of observation.

Recall that the log-odds ratio is given as

$$\ln\left(\frac{\theta_1/(1-\theta_1)}{\theta_2/(1-\theta_2)}\right).$$

We'll modify this expression slightly, using $p(\mathtt{H})$ to indicate the probability of encounter, p, as a function of the number of hours of observation, H. Let $p(\mathtt{H})$ be the probability of encounter given observation hours H, and let $p(\mathtt{H+1})$ be the probability of encounter given (H+1) hours of observation (i.e., a 1 unit change in HOURS).

Thus, we can write:

$$\ln\left(\frac{p(\mathtt{H+1})/(1-p(\mathtt{H+1}))}{p(\mathtt{H})/(1-p(\mathtt{H}))}\right).$$

Next, recall that the log-odds for some parameter θ is simply the logit transform for θ:

$$\ln\left(\frac{\theta}{1-\theta}\right).$$

Thus, we can write out the numerator and denominator terms of the log-odds ratio expression (above)

in terms of the linear model corresponding to both.

In other words, for the numerator,

$$\ln\Big(p(\texttt{H+1})/\big(1-p(\texttt{H+1})\big)\Big) = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{H+1}) + \beta_4(\texttt{SEX.(H+1)}),$$

while for the denominator

$$\ln\Big(p(\texttt{H})/\big(1-p(\texttt{H})\big)\Big) = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{H}) + \beta_4(\texttt{SEX.H}).$$

Remembering that we can write the log of a fraction as the difference of the log of the numerator and denominator,

$$\ln\left(\frac{p(\texttt{H+1})/\big(1-p(\texttt{H+1})\big)}{p(\texttt{H})/\big(1-p(\texttt{H})\big)}\right) = \ln\left[p(\texttt{H+1})/\big(1-p(\texttt{H+1})\big)\right] - \ln\left[p(\texttt{H})/\big(1-p(\texttt{H})\big)\right],$$

then given the linear model expressions for the numerator and denominator (above), we can show (with a bit of algebra) that the log-odds ratio given a unit change in hours of observation is

$$\begin{aligned}
&\ln\left[p(\texttt{H+1})/\big(1-p(\texttt{H+1})\big)\right] - \ln\left[p(\texttt{H})/\big(1-p(\texttt{H})\big)\right] \\
&= \left[\cancel{\beta_1} + \cancel{\beta_2(\texttt{SEX})} + \beta_3(\texttt{H+1}) + \beta_4(\texttt{SEX.(H+1)})\right] - \left[\cancel{\beta_1} + \cancel{\beta_2(\texttt{SEX})} + \beta_3(\texttt{H}) + \beta_4(\texttt{SEX.H})\right] \\
&= \beta_3(\texttt{H+1}) + \beta_4(\texttt{SEX.(H+1)}) - \beta_3(\texttt{H}) - \beta_4(\texttt{SEX.H}) \\
&= \cancel{\beta_3(\texttt{H})} + \beta_3 + \cancel{\beta_4(\texttt{SEX.H})} + \beta_4(\texttt{SEX}) - \cancel{\beta_3(\texttt{H})} - \cancel{\beta_4(\texttt{SEX.H})} \\
&= \beta_3 + \beta_4(\texttt{SEX}) \\
&= 0.0463413 - 0.07830219(\texttt{SEX}).
\end{aligned}$$

So, for males (SEX=1), the log-odds for the probability of encounter is calculated simply as

$$0.0463413 - 0.07830219(1) = -0.0319608.$$

Negative log-odds, indicating that the log-odds decrease – slightly – with each unit increase in the hours of observation, as expected given the figure on p. 76. If we back-transform from the log scale, $e^{-0.0319608} \approx 0.969$, i.e., the odds of detection of a male with an additional hour of observation are 0.969 : 1, which is less than 1 : 1, so...decreasing odds.

For females, (SEX=0),

$$0.0463413 - 0.07830219(0) = 0.0463413.$$

In other words, positive log odds, indicating that the log-odds for detecting a female increase – again slightly – with each unit increase in the hours of observation, consistent with the figure shown on the previous page.

Finally, if you wanted to express the uncertainty in your estimate of the log-odds, you can do this relatively easily by applying the Delta method. From the preceding, recall that the function relating the change in the log-odds of detection with increasing hours of observation, for a given sex, was given as

$$\begin{aligned}
\ln(\widehat{OR}) &= \ln\left[p(\texttt{H+1})/\big(1-p(\texttt{H+1})\big)\right] - \ln\left[p(\texttt{H})/\big(1-p(\texttt{H})\big)\right] \\
&= \hat{\beta}_3 + \hat{\beta}_4(\texttt{SEX}).
\end{aligned}$$

In other words, the RHS is the sum of 2 correlated random variables, $\hat{\beta}_3$ and $\hat{\beta}_4$, where the correlation structure is determined by the variance-covariance matrix for these 2 parameters (i.e., random variables).

Earlier, we showed that the estimated variance of a sum of correlated random variables is given as

$$s_A^2 = \mathrm{var}(X_1) + \mathrm{var}(X_2) + 2\,\mathrm{cov}(X_1, X_2).$$

So, for the present example,

$$\widehat{\mathrm{Var}}(\ln(OR)) = \widehat{\mathrm{Var}}(\hat{\beta}_3) + \widehat{\mathrm{Var}}(\hat{\beta}_4) + 2\,\mathrm{Cov}(\hat{\beta}_3, \hat{\beta}_4),$$

where the needed estimates of the parameter variances and covariances can be output directly from **MARK**.

begin sidebar

AIC, *P*-values and effect size – a tautology?

Hmmm...you might suspect that you smell a tautology here (it's either that, or the aroma of your frying brain cells). Up until now, we've considered the use of AIC as a robust means of model selection – part of the motivation being to avoid the use (and abuse) of classical '*P*-value' approaches to doing science.

While there are very good motivations for doing this (see relevant sections of the Burnham & Anderson text, and any number of other discussions of the issue), one of the motivations was to force us (biologists, analysts) to focus our attention more on the 'biological significance' of the effect size, rather than on some nominal '*P*-value'. We all know that it is not hard to generate a situation where we can show 'statistical significance' that has no biological meaning (this commonly occurs with very large data sets). So, we consider the effect size – the absolute magnitude of the difference.

Recall that our purpose is to consider

> *'what are the plausible bounds on the true effect size, and are biologically important effects contained within these bounds?'*

The potential problem is with the word 'plausible'. In theory, we are supposed to decide, *a priori*, on what the biologically plausible bounds are. And yet, in practice, to determine whether or not a calculated effect size falls within those bounds is based on assessing the confidence bounds estimated for the effect size, relative to the plausibility bounds designated by the scientist.

Herein is the potential tautology – we use 'biological insight' to *a priori* determine what we think a plausible (or biologically meaningful) effect should be, and yet we typically end up relying on use of 95% CI to test whether or not the effect size is plausible. And what do we base the 95% CI on? You got it – a nominal $\alpha = 0.05$ level.

Remember, that it is in part the arbitrariness of selecting the appropriate α level which underlies one of the several criticisms of the '*P*-value' approach. And yet, we potentially use the same underlying logic (and, arbitrariness) to derive the 95% CI for the effect size. There is perhaps good reason to wonder if we're not engaged in some sort of circular thinking here...stay tuned.

end sidebar

6.13.2. ĉ and effect size: a cautionary note

In the preceding, we illustrate the basic idea (and some of the mechanics) for estimating effect size. As we noted in Chapter 4, model selection (whether you use an information theoretic approach based on AIC, or the more traditional approach based on LRT) is conditional on adequate fit of the model

to the data. Some degree of lack of fit can be accommodated using a quasi-likelihood approach. This involved estimation of c, perhaps by using a median-$\hat{c}$, for example. However, one thing which may not be immediately obvious is the effect that using a quasi-likelihood adjustment has on model selection.

Consider, for example, what adjusting model fit using a $\hat{c} > 1$. One of the things you'll quickly notice if you progressively experiment by increasing $\hat{c}$ from 1, 1.25, 1.5, and so on, is that as you do so, the rankings of the models changes. Invariably, as $\hat{c}$ increases, models with fewer parameters take on progressively more support in the data, as indicated by the normalized AIC weights.

This should make some intuitive sense: a large value of $\hat{c}$ indicates significant lack of fit of the data to the general model. As such, increasing $\hat{c}$ is analogous to 'taking a more conservative' view of the data. The general model doesn't fit, so you tend to favor the more parsimonious model. Try it – pick a model set, and slowly, progressively, try increasing $\hat{c}$ from the default of 1. Watch closely to see how the model weights change, such that (typically) reduced parameter models get increasing amounts of weight.

What is not so intuitive is that changing $\hat{c}$ also changes the relative scaling of differences in model support. In short form, if 2 models differ in normalized AIC weights by some factor x for $\hat{c} = 1$, then this same difference x is not the same difference if $\hat{c} > 1$; it is less. For example, suppose you have 2 models, with $\hat{c} = 1.0$,where the difference in relative weight is (say) 2.5 times (in other words, one model has 2.5 times more weight than the other model). At this point, you look at effect size, and interpret the biological plausibility of this difference, given that one model has 2.5 more support in the data.

However, suppose instead that $\hat{c} = 1.5$, instead of 1. With increasing $\hat{c}$, a difference of 2.5 times support in the data is scaled differently, and must be interpreted differently, than it would be if $\hat{c} = 1$. Since increasing $\hat{c}$ increases the degree of 'conservatism' in the analysis, a difference of 2.5 times model support with $\hat{c} = 1.5$ is 'less of a difference' than the same 2.5 times difference would be with $\hat{c} = 1.0$.

Confused? Fair enough – it is rather non-intuitive, at first. Give it some thought. For those who want all the details, we suggest you have a look at Richard Royall's 1997 text on '*Statistical Evidence: A Likelihood Paradigm*' (a Chapman & Hall publication – part of the *Monographs Series on Statistics and Applied Probability*). In short – be a little cautious in how you interpret relative differences in normalized AIC weights if $\hat{c} > 1$.

6.14. Pulling all the steps together: a sequential approach

In this section, we summarize the basic sequence of steps of building design matrices, and interpreting the values of the estimate β terms (the 'slopes' of the linear model).

Broadly speaking, the sequence involves:

- **Step 1**: decide which model you want to fit
- **Step 2**: set up the PIMS for the general model
- **Step 3**: set up the linear model equation for the model you want to fit
- **Step 4**: set up the design matrix
- **Step 5**: confirm the design matrix with the corresponding linear model equations

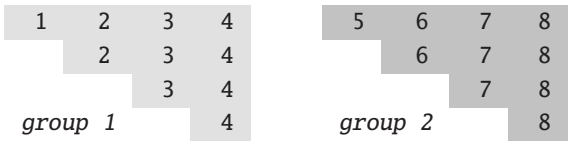
To illustrate this sequence of steps, we'll assume we have collected live mark-encounter data from 2 groups, over 4 time periods (i.e., 5 sampling occasions). To simplify the presentation somewhat, we'll focus only on the survival parameter φ (the same basic idea holds for the encounter parameter p – and any other parameters you might have in your model – equally as well).

Step 1: decide which model you want to fit.

For now, let's assume the model we're interested in is model $\{\varphi_g\}$ – in other words, a model where survival varies only as a function of the GROUP, but not TIME.

Step 2: set up PIMs for general model

For a starting model $\{\varphi_{g*t}\}$ (i.e., a model with a full interaction of GROUP and TIME) our PIMs would look like



Step 3: set up the linear model equation for model you want to fit

We want to fit model $\{\varphi_g\}$, which we write in our symbolic linear model notation as

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{GROUP}).$$

Step 4: set up the design matrix

In the following, the rows of the table correspond to the PIM index values for each time interval. The design matrix corresponding to $\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{GROUP})$ is given by the columns labeled β_1 and β_2.

			INTCP	GROUP
	PIM	Param	β_1	β_2
group 1	1	$\varphi_{g1,1}$	1	1
	2	$\varphi_{g1,2}$	1	1
	3	$\varphi_{g1,3}$	1	1
	4	$\varphi_{g1,4}$	1	1
group 2	5	$\varphi_{g2,1}$	1	0
	6	$\varphi_{g2,2}$	1	0
	7	$\varphi_{g2,3}$	1	0
	8	$\varphi_{g2,4}$	1	0

Note that as written, we have specified GROUP 2 to be the reference group (i.e., if $\beta_2 = 0$, then the intercept, β_1, corresponds to GROUP 2).

Step 5: confirm the design matrix with the equations

Here (top of the next page), we simply take the dummy coding for each GROUP and TIME combination, and substitute into the linear model equation $\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{GROUP})$.

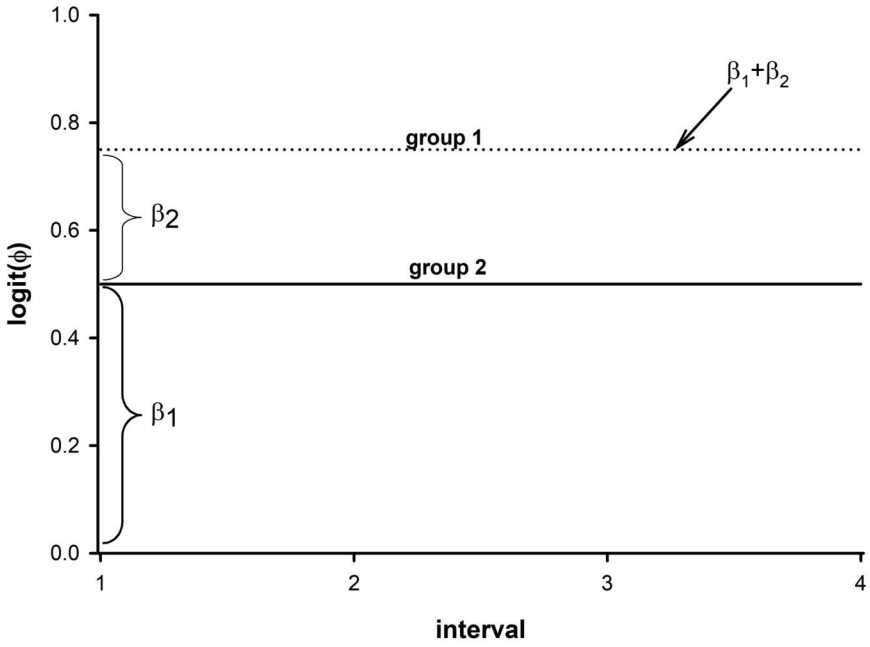
group 1	$\texttt{logit}(\varphi_{g1,1}) = \beta_1(1) + \beta_2(1)$	$\texttt{logit}(\varphi_{g1,1}) = \beta_1 + \beta_2$
	$\texttt{logit}(\varphi_{g1,2}) = \beta_1(1) + \beta_2(1)$	$\texttt{logit}(\varphi_{g1,2}) = \beta_1 + \beta_2$
	$\texttt{logit}(\varphi_{g1,3}) = \beta_1(1) + \beta_2(1)$	$\texttt{logit}(\varphi_{g1,3}) = \beta_1 + \beta_2$
	$\texttt{logit}(\varphi_{g1,4}) = \beta_1(1) + \beta_2(1)$	$\texttt{logit}(\varphi_{g1,4}) = \beta_1 + \beta_2$
group 2	$\texttt{logit}(\varphi_{g2,1}) = \beta_1(1) + \beta_2(0)$	$\texttt{logit}(\varphi_{g2,1}) = \beta_1$
	$\texttt{logit}(\varphi_{g2,2}) = \beta_1(1) + \beta_2(0)$	$\texttt{logit}(\varphi_{g2,2}) = \beta_1$
	$\texttt{logit}(\varphi_{g2,3}) = \beta_1(1) + \beta_2(0)$	$\texttt{logit}(\varphi_{g2,3}) = \beta_1$
	$\texttt{logit}(\varphi_{g2,4}) = \beta_1(1) + \beta_2(0)$	$\texttt{logit}(\varphi_{g2,4}) = \beta_1$

On the right hand side we see the result of substituting in the dummy variables. We see clearly that GROUP 2 is coded by the intercept: thus, β_2 represents the difference ('effect size') between GROUP 2 and GROUP 1.

Step 6: plot a graph to illustrate the model

Plotting a graph of the model is a very useful way to visualize (and thus, truly understand) the structure of the model, and what the individual β_i terms represent.

For this model, the graph of the β terms is shown below. Note that the meaning of the β_i terms is explicitly represented, as is the effect size (which, in this case, is the value of β_2, which is the difference between the two horizontal lines on the logit scale):



Got it? Well, let's test our understanding by trying several more scenarios.

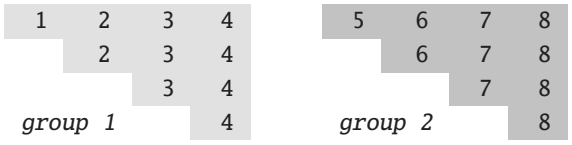
Let's take the same basic data model (2 groups, 5 occasions), and now consider fitting model φ_t – TIME variation in φ, but no GROUP effect (i.e., essentially the opposite of the model we just considered).

Step 1: decide which model you want to fit.

As noted, let's consider model $\{\varphi_t\}$ – in other words, a model where survival varies only as a function of the TIME, but not GROUP.

Step 2: set up PIMs for general model

As above, our general model would still be $\{\varphi_{g*t}\}$ (i.e., a model with a full interaction of GROUP and TIME). For this model, our PIMs would look like the following:



Step 3: set up the linear model equation for model you want to fit

We want to fit model $\{\varphi_t\}$, which we write in our symbolic linear model notation as

$$\text{logit}(\varphi) = \beta_1 + \beta_2(t_1) + \beta_3(t_2) + \beta_4(t_3).$$

Now, make sure you understand why this is the appropriate linear model. We have 5 sampling occasions, which means 4 intervals. To uniquely code the intervals, we need $(n - 1) = (4 - 1) = 3$ columns of dummy variables, or (more specifically) 3 $\beta_{i,i\neq 1}$ terms in addition to the intercept term β_1.

Step 4: set up the design matrix

In the following table the rows correspond to the PIM index values for each time interval. The design matrix corresponding to $\text{logit}(\varphi) = \beta_1 + \beta_2(t_1) + \beta_3(t_2) + \beta_4(t_3)$ is given by the columns labeled $\beta_1 \rightarrow \beta_4$. Note that as written, we have specified TIME 4 (i.e., the intervals between sampling occasion 4 and 5) to be the reference group (i.e., if $\beta_2 = \beta_3 = \beta_4 = 0$, then the intercept – β_1 – corresponds to TIME 4).

			INTCPT	TIME1	TIME2	TIME3
	PIM	Param	β_1	β_2	β_3	β_4
group 1	1	$\varphi_{g1,1}$	1	1	0	0
	2	$\varphi_{g1,2}$	1	0	1	0
	3	$\varphi_{g1,3}$	1	0	0	1
	4	$\varphi_{g1,4}$	1	0	0	0
group 2	5	$\varphi_{g2,1}$	1	1	0	0
	6	$\varphi_{g2,2}$	1	0	1	0
	7	$\varphi_{g2,3}$	1	0	0	1
	8	$\varphi_{g2,4}$	1	0	0	0

Step 5: confirm the design matrix with the equations

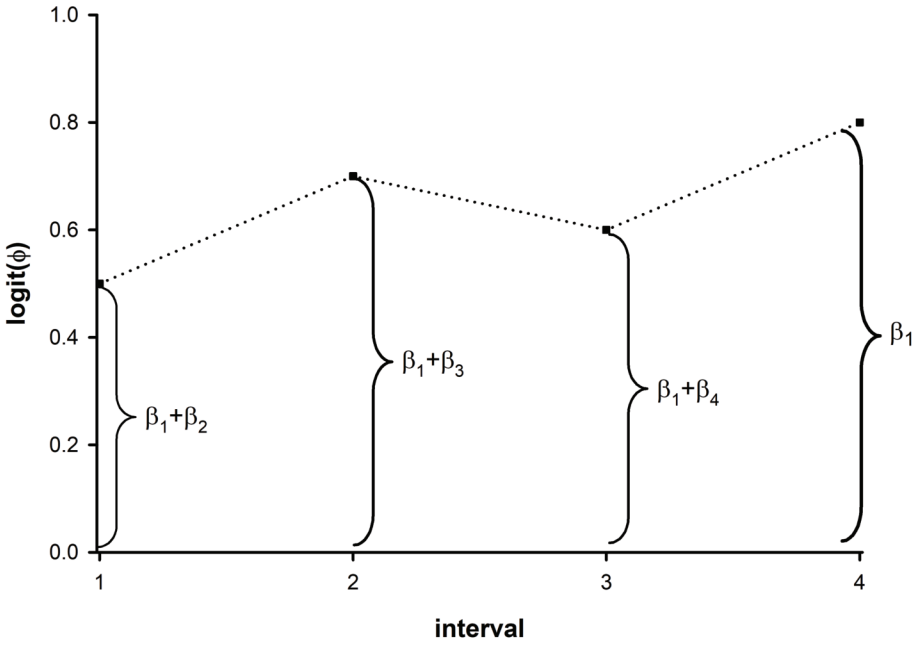
Here, we simply take the dummy coding for each GROUP and TIME combination, and substitute into the linear model equation $\text{logit}(\varphi) = \beta_1 + \beta_2(t_1) + \beta_3(t_2) + \beta_4(t_3)$.

group 1	$\texttt{logit}(\varphi_{g1,1}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(0)$	$\texttt{logit}(\varphi_{g1,1}) = \beta_1 + \beta_2$
	$\texttt{logit}(\varphi_{g1,2}) = \beta_1(1) + \beta_2(0) + \beta_3(1) + \beta_4(0)$	$\texttt{logit}(\varphi_{g1,2}) = \beta_1 + \beta_3$
	$\texttt{logit}(\varphi_{g1,3}) = \beta_1(1) + \beta_2(0) + \beta_3(0) + \beta_4(1)$	$\texttt{logit}(\varphi_{g1,3}) = \beta_1 + \beta_4$
	$\texttt{logit}(\varphi_{g1,4}) = \beta_1(1) + \beta_2(0) + \beta_3(0) + \beta_4(0)$	$\texttt{logit}(\varphi_{g1,4}) = \beta_1$
group 2	$\texttt{logit}(\varphi_{g2,1}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(0)$	$\texttt{logit}(\varphi_{g2,1}) = \beta_1 + \beta_2$
	$\texttt{logit}(\varphi_{g2,2}) = \beta_1(1) + \beta_2(0) + \beta_3(1) + \beta_4(0)$	$\texttt{logit}(\varphi_{g2,2}) = \beta_1 + \beta_3$
	$\texttt{logit}(\varphi_{g2,3}) = \beta_1(1) + \beta_2(0) + \beta_3(0) + \beta_4(1)$	$\texttt{logit}(\varphi_{g2,3}) = \beta_1 + \beta_4$
	$\texttt{logit}(\varphi_{g2,4}) = \beta_1(1) + \beta_2(0) + \beta_3(0) + \beta_4(0)$	$\texttt{logit}(\varphi_{g2,4}) = \beta_1$

On the right hand side we see the result of substituting in the dummy variables. We see clearly that TIME 4 is coded by the intercept: thus, $\beta_2 \rightarrow \beta_4$ represents the various differences ('effect sizes') between the different TIME intervals, and the final TIME interval (TIME 4).

Step 6: plot a graph to illustrate the model

Note (in the figure, top of the next page) that the meaning of the β_i terms is explicitly represented, as are the effect sizes (which, in this case, is the value of the difference $(\beta_1 + \beta_{i\neq 1})$ and β_1.



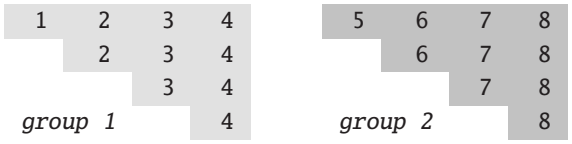
Now, for a final test, to really make sure you've got it. You've probably anticipated model $\{\varphi_{g*t}\}$. Here it is!

Step 1: decide which model you want to fit.

For our final example, let's consider model $\{\varphi_{g*t}\}$ – in other words, a model where survival varies as a function of the both GROUP and TIME, with full interaction between the two.

Step 2: set up PIMs for general model

Clearly, our general model must be $\{\varphi_{g*t}\}$ (i.e., a model with a full interaction of GROUP and TIME), since this is in fact the model we're trying to fit! For this model, our PIMs are, again



Step 3: set up the linear model equation for model you want to fit

We want to fit model $\{\varphi_{g*t}\}$, which we write in our symbolic linear model notation as

$$\begin{aligned}\text{logit}(\varphi) = {} & \beta_1 + \beta_2(\texttt{GROUP}) + \beta_3(\texttt{t}_1) + \beta_4(\texttt{t}_2) + \beta_5(\texttt{t}_3) \\ & + \beta_6(\texttt{GROUP.t}_1) + \beta_7(\texttt{GROUP.t}_2) + \beta_8(\texttt{GROUP.t}_3).\end{aligned}$$

Now, make sure you understand why this is the appropriate linear model. We have 2 groups, and 5 sampling occasions, which means 4 intervals. To uniquely code the for intervals, we need $(n-1) = (4-1) = 3$ columns of dummy variables for TIME, $(n-1) = (2-1) = 1$ column for GROUP, and $(3 \times 1) = 3$ columns for the interaction of GROUP.TIME. So, a total of $(1+3+1+3) = 8$ columns (β_i terms).

Step 4: set up the design matrix

In the following, the rows of the table correspond to the PIM index values for each time interval. The design matrix corresponding to our linear model (above) is given by the columns labeled $\beta_1 \rightarrow \beta_8$.

Note that as written, we have specified GROUP 2 during TIME 4 to be the reference group and time (i.e., if $\beta_2 = \beta_3 = \ldots \beta_8 = 0$, then the intercept – β_1 - corresponds to GROUP 2 during TIME 4).

			INTCPT	GROUP	TIME1	TIME2	TIME3	G.T1	G.T2	G.T3
	PIM	Param	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8
group 1	1	$\varphi_{g1,1}$	1	1	1	0	0	1	0	0
	2	$\varphi_{g1,2}$	1	1	0	1	0	0	1	0
	3	$\varphi_{g1,3}$	1	1	0	0	1	0	0	1
	4	$\varphi_{g1,4}$	1	1	0	0	0	0	0	0
group 2	5	$\varphi_{g2,1}$	1	0	1	0	0	0	0	0
	6	$\varphi_{g2,2}$	1	0	0	1	0	0	0	0
	7	$\varphi_{g2,3}$	1	0	0	0	1	0	0	0
	8	$\varphi_{g2,4}$	1	0	0	0	0	0	0	0

Step 5: confirm the design matrix with the equations

Here, we simply take the dummy coding for each GROUP and TIME combination, and substitute into the linear model equation, which again is

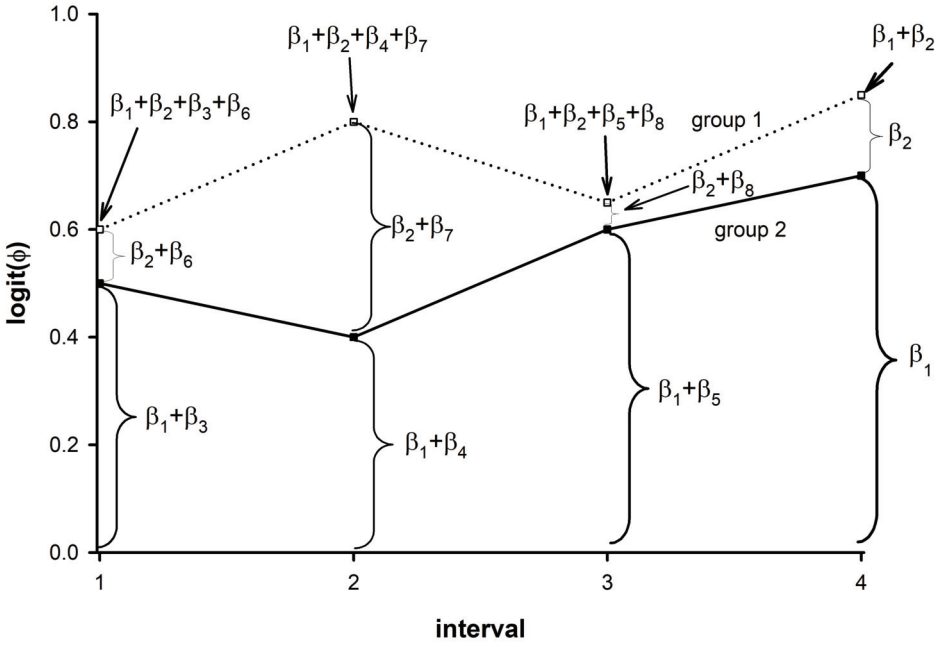
$$\begin{aligned}\text{logit}(\varphi) = {} & \beta_1 + \beta_2(\texttt{GROUP}) + \beta_3(\texttt{t}_1) + \beta_4(\texttt{t}_2) + \beta_5(\texttt{t}_3) \\ & + \beta_6(\texttt{GROUP.t}_1) + \beta_7(\texttt{GROUP.t}_2) + \beta_8(\texttt{GROUP.t}_3).\end{aligned}$$

gr 1	$\text{logit}(\varphi_{g1,1}) = \beta_1(1) + \beta_2(1) + \beta_3(1) + \beta_4(0) + \beta_5(0) + \beta_6(1) + \beta_8(0) + \beta_8(0)$	$\text{logit}(\varphi_{g1,1}) = \beta_1 + \beta_2 + \beta_3 + \beta_6$
	$\text{logit}(\varphi_{g1,2}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(1) + \beta_5(0) + \beta_6(0) + \beta_7(1) + \beta_8(0)$	$\text{logit}(\varphi_{g1,2}) = \beta_1 + \beta_2 + \beta_4 + \beta_7$
	$\text{logit}(\varphi_{g1,3}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(0) + \beta_5(1) + \beta_6(1) + \beta_7(0) + \beta_8(1)$	$\text{logit}(\varphi_{g1,3}) = \beta_1 + \beta_2 + \beta_5 + \beta_8$
	$\text{logit}(\varphi_{g1,4}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(0) + \beta_5(0) + \beta_6(0) + \beta_7(0) + \beta_8(0)$	$\text{logit}(\varphi_{g1,4}) = \beta_1 + \beta_2$
gr 2	$\text{logit}(\varphi_{g2,1}) = \beta_1(1) + \beta_2(0) + \beta_3(1) + \beta_4(0) + \beta_5(0) + \beta_6(0) + \beta_7(0) + \beta_8(0)$	$\text{logit}(\varphi_{g2,1}) = \beta_1 + \beta_3$
	$\text{logit}(\varphi_{g2,2}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(1) + \beta_5(0) + \beta_6(0) + \beta_7(0) + \beta_8(0)$	$\text{logit}(\varphi_{g2,2}) = \beta_1 + \beta_4$
	$\text{logit}(\varphi_{g2,3}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(0) + \beta_5(1) + \beta_6(1) + \beta_7(0) + \beta_8(0)$	$\text{logit}(\varphi_{g2,3}) = \beta_1 + \beta_5$
	$\text{logit}(\varphi_{g2,4}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(0) + \beta_5(0) + \beta_6(0) + \beta_7(0) + \beta_8(0)$	$\text{logit}(\varphi_{g2,4}) = \beta_1$

On the right hand side we see the result of substituting in the dummy variables. We see clearly that `GROUP` 2 at `TIME` 4 is coded by the intercept: thus, $\beta_2 \rightarrow \beta_8$ represents the various differences ('effect sizes') between the different `TIME` and `GROUP` combinations intervals, and `GROUP` 2 during `TIME` 4.

Step 6: plot a graph to illustrate the model

The figure for model $\{\varphi_{g*t}\}$ is shown below – pay close attention to all the interactions, and effects. Spend enough time here so that each of the elements of the figure make some sense to you.



Got it? Hopefully you've now got the basic idea.

Going through some version of this sequence for any model you might want to build will build understanding, and confidence. We demonstrate this is the next section, where we apply it to 'alternative design matrix' approaches for additive models.

6.14.1. Application – alternative design matrices for additive models

Back in section 6.2, we introduced the idea of the design matrix (DM), and suggested that there are frequently many different forms of the DM that are equivalent, in terms of fit of the model to the data, and the real parameters, but where the structure resulted in different interpretations of the β parameters.

For example, the β estimates from the following two design matrices have different interpretations (although they yield identical back-transformed real parameter estimates):

$$\mathbf{X}_{\text{offset}} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{X}_{\text{identity}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In the first example, $\mathbf{X}_{\text{offset}}$, β_1 represents the 'reference' (intercept), which in this case, corresponds to the final parameter estimate. The other parameters, $\beta_2 \rightarrow \beta_4$, represent the 'offset' (difference, effect) between the parameter value for that interval or occasion, and the reference. For matrix $\mathbf{X}_{\text{identity}}$, there is no reference – each β estimate represents the parameter for that interval or occasion. The real parameter estimates are the same using either DM, but the interpretation of the β parameters differs.

While most of the examples in this chapter have used 'offset coding', with a single intercept against which everything else is 'compared', it is often useful to re-structure the DM, either to generate β parameters which have a particular interpretation of interest, or, potentially, some utility given their structure. We'll demonstrate this basic idea by re-structuring the DM for an additive model (additive models were introduced earlier, in section 6.12). Here we assume that we have a simple CJS analysis, 2 groups, with 5 sampling occasions (4 intervals).

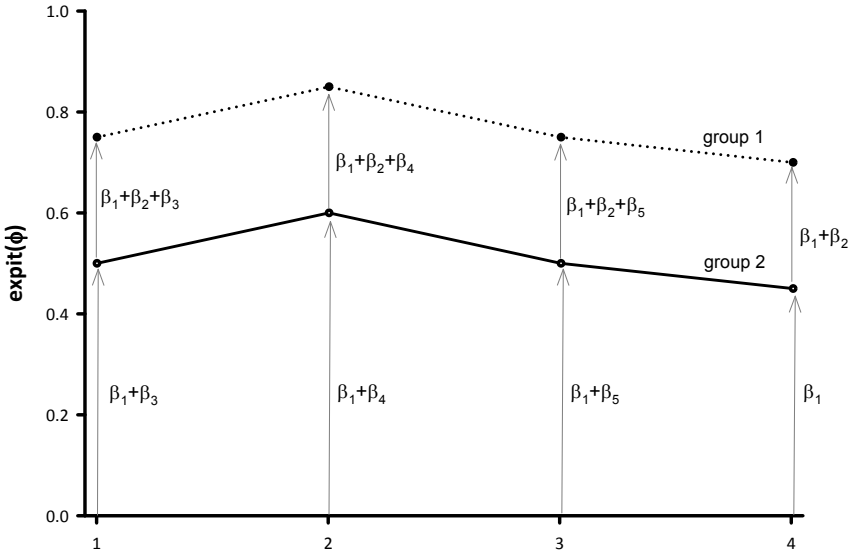
The following DM shows standard 'intercept-offset' coding for an additive model for (say) φ = `grp` + `time`:

Parm	B1: int	B2: grp	B3: t1	B4: t2	B5: t3	B6: p
1:Phi	1	1	1	0	0	0
2:Phi	1	1	0	1	0	0
3:Phi	1	1	0	0	1	0
4:Phi	1	1	0	0	0	0
5:Phi	1	0	1	0	0	0
6:Phi	1	0	0	1	0	0
7:Phi	1	0	0	0	1	0
8:Phi	1	0	0	0	0	0
9:p	0	0	0	0	0	1

The corresponding linear model is

$$\varphi = \beta_1 + \beta_2(\texttt{grp}) + \beta_3(\texttt{time}_1) + \beta_4(\texttt{time}_2) + \beta_5(\texttt{time}_3).$$

As in the preceding section, we use the plot shown at the top of the next page to illustrate the relationship between the β parameters in the linear model. As shown in the figure, each parameter (i.e., at each combination of `grp` and `time`) is estimated by the intercept (β_1) plus the 'effect' of one or more of the other β parameters. Parameter β_2 is the 'additive effect size' – the difference between the 2 groups.



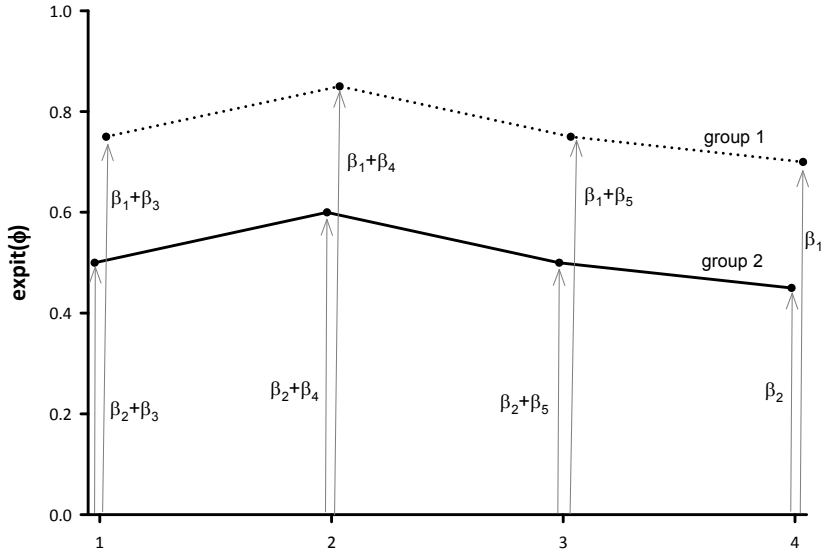
Now, instead of intercept-offset coding, let's consider two other parameterizations. First, instead of a common intercept, let's consider the case where we might be interested in each group having it's own intercept (representing, say, the final time interval for that group), with each of the time intervals representing a deviation (offset) from that reference interval, for that group.

The design matrix corresponding to this approach would be:

Parm	B1: g1	B2: g2	B3: t1	B4: t2	B5: t3	B6: p
1:Phi	1	0	1	0	0	0
2:Phi	1	0	0	1	0	0
3:Phi	1	0	0	0	1	0
4:Phi	1	0	0	0	0	0
5:Phi	0	1	1	0	0	0
6:Phi	0	1	0	1	0	0
7:Phi	0	1	0	0	1	0
8:Phi	0	1	0	0	0	0
9:p	0	0	0	0	0	1

The plot at the top of the next page shows the relationship between the β parameters in the linear model – in this case, each parameter is the group-specific intercept (β_1 for group 1, β_2 for group 2), plus the 'effect' of one or more of the other β parameters as time-specific deviation for that group, using the final interval as the 'reference'. Make sure you see the connection between this diagram and the corresponding DM (above).

You might wonder about the motivation for using such a DM. As we'll see in later chapters, it is occasionally useful to create a set of parameters that correspond to groups (`grp`) specified in your analysis (classification levels, factors). This happens naturally for `time`, which by default is modeled as a fixed effect factor, but allowing each `grp` to have its own intercept can be useful. It also means that β_1 and β_2 now represent the parameter value for each group separately, not as referenced to some common intercept. In effect, this is analogous to creating an 'identity matrix' structure, but for only one class of parameters (in this case, `grp`) – time still has the 'offset' structure, but now it is 'offset' from the



group-specific intercept.

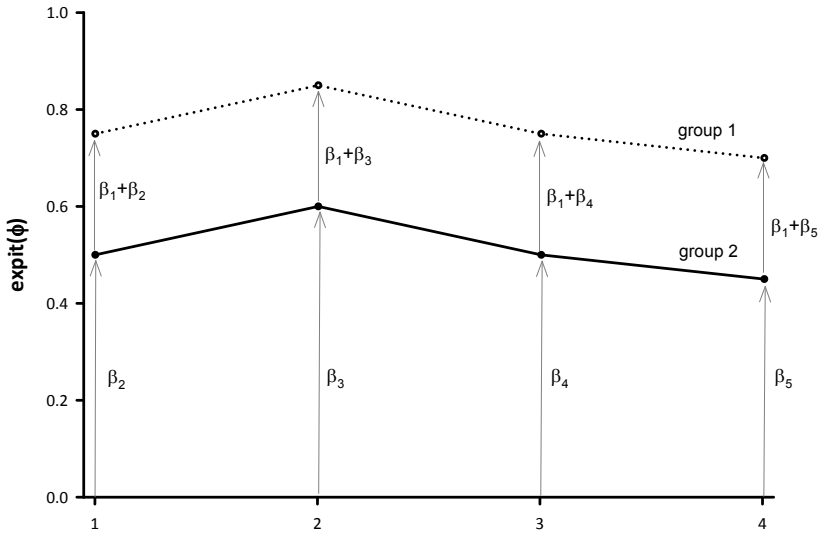
We can illustrate this idea more clearly by next considering the case where we want an identity-matrix structure for `time`, with each of the β parameters representing a deviation from that reference interval, for that group.

The design matrix achieving this objective would be:

Parm	B1: grp	B2: t1	B3: t2	B4: t3	B5: t4	B6: p
1:Phi	1	1	0	0	0	0
2:Phi	1	0	1	0	0	0
3:Phi	1	0	0	1	0	0
4:Phi	1	0	0	0	1	0
5:Phi	0	1	0	0	0	0
6:Phi	0	0	1	0	0	0
7:Phi	0	0	0	1	0	0
8:Phi	0	0	0	0	1	0
9:p	0	0	0	0	0	1

Again, we use the plot shown at the top of the next page to show the relationship between the β parameters in the linear model – in this case, parameters $\beta_2 \rightarrow \beta_5$ represent each of the 4 time intervals, respectively, whereas β_1 is the offset due to being in group 1 (here, we use group 2 as the reference).

Each of these 3 different DM would, if applied to the same data, yield identical real parameter estimates. However, the meaning/interpretation of the β parameters is different. And, in the second two examples, we've created DM where one of the parameters is 'offset-coded', while the other is 'identity' coded. These different codings can be useful for some purposes.



begin sidebar

Design matrix coding and parameter estimability

At various points in this chapter, including in the preceding section, we've noted that there are any number of ways to code the design matrix, each yielding equivalent estimates, but differing in how the individual β terms of the linear model are interpreted.

However, sometimes, there are some subtle steps to constructing a design matrix which are well worth keeping in mind. For example, consider the basic structure for the encounter part of the design matrix – up until now, we've used a reference coding scheme where we usually make the final element of the matrix the reference element.

For example, if we had the following matrix for (say) φ_t for a design with 6 time intervals

1	1	0	0	0	0
1	0	1	0	0	0
1	0	0	1	0	0
1	0	0	0	1	0
1	0	0	0	0	1
1	0	0	0	0	0

we're specifying that the last time interval is used as the reference (such that the intercept β_1 is the estimate of survival (on some scale) for this interval, and all the other β terms represent deviations from this reference.

But, suppose instead we had constructed our design matrix using

1	0	0	0	0	0
1	1	0	0	0	0
1	0	1	0	0	0
1	0	0	1	0	0
1	0	0	0	1	0
1	0	0	0	0	1

such that the estimate of survival over the first interval is now the reference value. We know from everything we've covered up until now that changing the reference coding scheme used in the design matrix will change the interpretation of the β_i terms, but does it change anything else?

The general answer is, 'no' – it shouldn't. At least, most of the time. In general, the model deviance (fit) and the reconstituted estimates should not be influenced by the choice of the coding scheme used in the design matrix. But, there are some somewhat 'pathological' cases where it might. We'll have a quick look at one example, if only to prove a point.

Consider the European dipper (yes, again) – this time, using the input file containing data from both males and females (**ed.inp**). We fit model $\{\varphi_{g*t}\, p_{g*t}\}$ to these data, using the design matrix shown below (which you'll recall is the default full '`time` × `group`' design matrix in **MARK**):

Design Matrix Specification (B = Beta)

B1 Phi Int	B2 Phi g1	B3 Phi t1	B4 Phi t2	B5 Phi t3	B6 Phi t4	B7 Phi t5	B8 Phi g1t1	B9 Phi g1t2	B10 Phi g1t3	B11 Phi g1t4	B12 Phi g1t5	Parm	B13 p Int	B14 p g1	B15 p t1	B16 p t2	B17 p t3	B18 p t4	B19 p t5	B20 p g1t1	B21 p g1t2	B22 p g1t3	B23 p g1t4	B24 p g1t5
1	1	1	0	0	0	0	1	0	0	0	0	1:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	1	0	0	0	2:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	1	0	0	3:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	1	0	4:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0	0	0	0	1	5:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	6:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	7:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	8:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	9:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	10:Ph	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	11:Ph	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	12:Ph	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	13:p	1	1	1	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	14:p	1	1	0	1	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	15:p	1	1	0	0	1	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	16:p	1	1	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	17:p	1	1	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	18:p	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19:p	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	20:p	1	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	21:p	1	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	22:p	1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	23:p	1	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	24:p	1	0	0	0	0	0	0	0	0	0	0	0

Look closely at the coding. Note that we've made the terminal time periods the reference cell. This is the default in **MARK** (which you can change by selecting the appropriate option under '**`File | Preferences`**').

OK – so what's wrong with that? Well, perhaps nothing obvious, but what do we know about this analysis: 2 groups, 7 occasions, so 24 columns in the design matrix (as shown). But, how many of the parameters (columns in the DM) are estimable?

Well, if you remember when we first introduced this data set, you might recall that the final $\varphi_6 p_7$ values for both sexes aren't separately identifiable (the so-called 'beta' parameters). So, $(24 - 2) = 22$ estimable parameters. But, if you run this model, using this design matrix and the logit link function, **MARK** reports only 21 parameters. Why? Because with the logit link, **MARK** was unable to estimate the second encounter probability for males ($p_{m,3}$).

The relevant sections of the full output for this model are shown below:

```
-0.9105111E-05 0.3694684E-04 0.5976433E-04-0.6309644E-04  0.000000
-0.1554401E-05 0.2728256E-04 0.9001948E-04-0.9354902E-04

S Vector {terminal reference}:
  133.3026      27.70245      25.51529      22.16000      20.05069
  19.20551      5.325007      4.420726      3.739595      3.270304
  3.097425      2.708648      2.502976      2.047853      1.629654
 0.8504524     0.6609147     0.4615799     0.3686566     0.3081518
 0.2072907     0.1063263E-04 0.8804990E-05 0.7287915E-06

  Threshold =  0.5000000E-06     Condition index =  0.5467199E-08

Conditioned S Vector {terminal reference}:
  1.000000      0.2078164     0.1914089     0.1662383     0.1504149
 0.1440746     0.3994677E-01 0.3316310E-01 0.2805343E-01 0.2453294E-01
 0.2323605E-01 0.2031955E-01 0.1877665E-01 0.1536244E-01 0.1222523E-01
 0.6379866E-02 0.4958005E-02 0.3462649E-02 0.2765563E-02 0.2311672E-02
 0.1555039E-02 0.7976310E-07 0.6605267E-07 0.5467199E-08

Number of Estimated Parameters {terminal reference} = 21
```

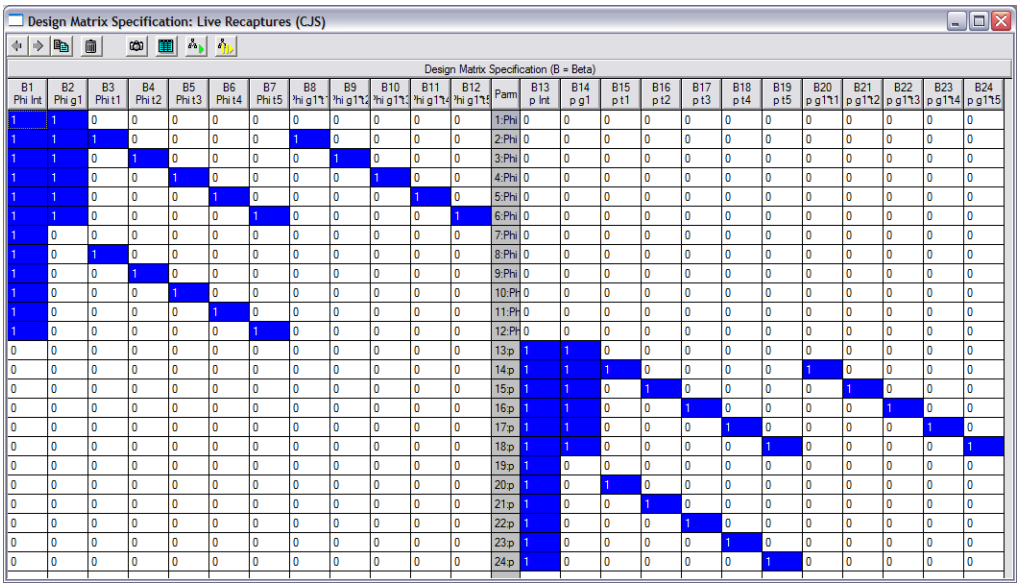
We see clearly that from the conditional **S**-vector that 3 parameters are below the threshold (24 – 3 = 21 estimated parameters, as indicated).

So, what does this have to do with how we coded the design matrix? Well, think about what this design matrix does – it makes the final time step the reference. And this might be a problem...because?

Because the final time step involves a non-identifiable β term! So, we're using for our reference a parameter which can't be separately identified anyway! Perhaps this isn't such a good idea.

What happens if instead we make the *first* time step the reference, for both parameters? [in fact, **MARK** has a '**`File | Preferences`**' option to do this for you by default, for any new design matrix you create.]

This change in coding is shown in the following design matrix:



B1 Phi Int	B2 Phi g1	B3 Phi t1	B4 Phi t2	B5 Phi t3	B6 Phi t4	B7 Phi t5	B8 Phi g1t1	B9 Phi g1t2	B10 Phi g1t3	B11 Phi g1t4	B12 Phi g1t5	Parm	B13 p Int	B14 p g1	B15 p t1	B16 p t2	B17 p t3	B18 p t4	B19 p t5	B20 p g1t1	B21 p g1t2	B22 p g1t3	B23 p g1t4	B24 p g1t5
1	1	0	0	0	0	0	0	0	0	0	0	1:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	1	0	0	0	0	2:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	1	0	0	0	3:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	1	0	0	4:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	1	0	5:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0	0	0	0	1	6:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	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
1	0	1	0	0	0	0	0	0	0	0	0	8:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	9:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	10:Ph	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	11:Ph	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	12:Ph	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	13:p	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	14:p	1	1	1	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	15:p	1	1	0	1	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	16:p	1	1	0	0	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	17:p	1	1	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	18:p	1	1	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	19:p	1	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	20:p	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	21:p	1	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	22:p	1	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	23:p	1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	24:p	1	0	0	0	0	0	1	0	0	0	0	0

Look closely, and compare it to the default coding (on the preceding page). Make sure you see the differences.

Now, does this subtle change in what we use for the reference change our results? In this case, yes – if you fit a model using this design matrix to the dipper data, **MARK** will correctly report 22 parameters. The offending parameter $p_{m,3}$ is now estimated and counted properly.

Here is the conditional *S*-vector for this modified (re-coded) model:

```
 -0.9780446E-04 0.1962163E-03 0.8807557E-04-0.6442858E-03  0.000000
 -0.1749013E-04 0.2262792E-03 0.8332759E-04-0.5849366E-03

 S Vector {initial reference}:
   130.0919       52.18201       25.64629       22.15497       19.84161
   18.86744       8.888512       7.544042       3.743907       3.287813
   3.098697       2.726529       2.432217       1.361460      0.9692126
  0.7372304      0.4456116      0.3557991      0.2610269      0.1159028
  0.3524389E-01 0.1770967E-03 0.2292490E-05 0.4812934E-06

    Threshold =  0.5000000E-06     Condition index =  0.3699643E-08

Conditioned S Vector {initial reference}:
    1.000000      0.4011167      0.1971398      0.1703026      0.1525200
   0.1450317      0.6832490E-01 0.5799012E-01 0.2877895E-01 0.2527302E-01
   0.2381930E-01 0.2095849E-01 0.1869615E-01 0.1046537E-01 0.7450218E-02
   0.5666999E-02 0.3425361E-02 0.2734984E-02 0.2006482E-02 0.8909306E-03
   0.2709155E-03 0.1361320E-05 0.1762209E-07 0.3699643E-08

  Number of Estimated Parameters {initial reference} = 22
```

Note that in the results browser (below), the two models have exactly the same model deviance – the two models have the identical likelihood and deviance values, but different conditional **S**-vectors – leading **MARK** to conclude they have a different number of estimable parameters, which they clearly should not (since they are equivalent models).

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{terminal reference}	698.2382	0.0000	0.75251	1.0000	21	71.4740
{initial reference}	700.4623	2.2241	0.24749	0.3289	22	71.4740

While we selected this 'pathological' example intentionally, the general point we want to make is that *you probably should not make a non-identifiable parameter the reference cell in your design matrix*. Doing so can have unintended results, as this example shows. As noted earlier (-sidebar-, p. 18) you can set a preference in **MARK** to use the first row (i.e., first interval or occasion) as the reference by default.

end sidebar

6.15. A final example: mean values

A final example to emphasize the power and flexibility of design matrices in **MARK**. Suppose you're working on a final report for some study of some organism. In your report, you've been asked to report the 'average' survival value over the years of the study. Now, if a model where apparent survival is constant over time (i.e., $\varphi_\cdot$) fits the data much better than the model where survival is allowed to vary over time (i.e., φ_t), then it might seem reasonable to simply report the constant survival estimate as the mean. However, suppose instead that the model with time-specific variation in survival is strongly supported – what do you do then? Well, the obvious answer might be to simply add up the individual time-specific estimates, and take the arithmetic average. Is this correct? What about the SE for this average?

In fact, the most direct (and rigorous) way to estimate the mean survival over time, and the appropriate standard error, is to fit a model with an appropriately coded design matrix. To show how this might be accomplished, assume a design matrix with 4 estimates of survival. The default in **MARK** for a time-specific estimate of survival would be a (4×4) identity matrix. Recall that for the identity matrix, each row corresponds to a real parameter, and each column corresponds to a β parameter. Thus, each β parameter represents a treatment estimate (where in this case, each level of the 'treatment' corresponds to a different time interval). Alternatively, we could use the intercept-based 'reference' coding we've used throughout most of this chapter – for this example, with 4 intervals, we would use

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Now recall from earlier in the chapter we saw that there was yet another way we could modify the design matrix:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & -1 & -1 & -1 \end{bmatrix}.$$

With this design matrix, the back-transformed β_1 provides the *mean* of the survival estimates. The estimates of β_2 through β_4 provide the time variation around the estimated mean. To see how this works, compute the mean of the rows of the matrix: $[(\beta_1 + \beta_2) + (\beta_1 + \beta_3) + (\beta_1 + \beta_4) + (\beta_1 - \beta_2 - \beta_3 - \beta_4)]/4 = 4\beta_1/4 = \beta_1$. Pretty nifty!

Now, to obtain the estimate of mean survival, the estimate of β_1 must be transformed using the link function used for the model. For example, with the logit link, we write

$$\bar{\hat{S}} = \frac{e^{\hat{\beta}_1}}{1 + e^{\hat{\beta}_1}} \equiv \frac{1}{1 + e^{-\hat{\beta}_1}}.$$

So, again we see that by use of the design matrix, we can estimate many things of interest.

begin sidebar

Caution 1: estimating the mean

Now, while the preceding discussion seems fairly straightforward, there is a small little problem involving 'bias'. It turns out that estimates of the mean are potentially biased as a result of transforming back and forth from the non-linear link function. For example, suppose S_i are 0.5, 0.6, 0.7 and 0.8. The arithmetic mean of these 4 values is 0.65. With the logit link, the transformed values are 0, 0.40547, 0.8473 and 1.38629, respectively, giving an intercept of 0.659765. Back transforming from this value gives 0.6592, not 0.6500.

Thus, all of the link functions in **MARK**, except the identity link (the only 'linear' link function), will provide slightly biased estimates of the mean value. (Note: the direction of the bias in the logit link will vary depending on whether the mean parameter value is above or below 0.5). Because the identity link is a linear function, estimates of the mean calculated using the identity link will be unbiased; however, the identity link doesn't always work. In most cases, standard errors of the estimates will typically dominate such transformation bias, so that the bias in the back-transformed estimate of the mean is frequently ignored.

Caution 2: 'dot' models and other approaches

You might wonder 'why not simply fit a time-invariant 'dot model' to derive mean values for the vital rates?'. After all, a 'dot model' yields the same value for all intervals.

Unfortunately, this approach, while simple to implement, is not strictly correct. While the estimated 'mean' from the 'dot' model will be relatively robust (i.e., fairly unbiased), the estimated SE will be negatively biased wrt to the true process variance – often by a considerable amount. Here is a simple example – we simulated a data set of live-encounter data (**`calc_mean.inp`**), 16 occasions (15 intervals; 250 newly marked individuals released at each occasion), where 'true survival' over a given interval was generated by selecting a random normal deviate drawn from $\mathcal{N}(0.7, 0.005)$ (in the generating model, encounter probability p was constant; $p = 0.35$).

Here is the set (i.e., 'sample' from the underlying random distribution) of the 15 parameter values we used for simulating survival for each of the 15 intervals:

$$\{0.636, 0.615, 0.620, 0.627, 0.644, 0.646, 0.692, 0.701, 0.720, 0.730, 0.733, 0.737, 0.746, 0.790, 0.863\}.$$

Based on these values, the true *sample* mean survival is $\bar{S} = 0.700$, with a sample variance of $\sigma^2 = 0.005$, both of which conveniently happen to match the true mean and process variance of the distribution used to generate the values in the first place (this was intentional).

We'll consider several different approaches to estimating the mean survival probability: (i) using the estimate of $\hat{\varphi}$ from a 'dot' model, $\{S_\cdot\, p_\cdot\}$, (ii) using the estimates from a design matrix for a $\{S_t\, p_\cdot\}$ model constructed such that one of the estimated β terms (the intercept) corresponds directly to the mean, (iii) a Taylor series approximation known as the 'Delta method' (introduced in detail in Appendix B),

and (iv) a variance components decomposition (the subject of random effects models and variance components analysis is considered in some detail in Appendix D).

Fitting the 'true' (generating) model, $\{S_t\ p_.\}$, to the data yielded the following estimates of survival:

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.6286787	0.0515147	0.5234903	0.7229366
2:Phi	0.6485724	0.0420425	0.5624803	0.7259768
3:Phi	0.8004159	0.0424141	0.7044357	0.8709384
4:Phi	0.7391009	0.0385135	0.6569734	0.8073320
5:Phi	0.7107562	0.0357921	0.6359482	0.7756159
6:Phi	0.7184495	0.0338199	0.6477318	0.7797970
7:Phi	0.7887287	0.0339004	0.7147438	0.8476157
8:Phi	0.7761273	0.0321467	0.7069494	0.8328365
9:Phi	0.8775703	0.0351293	0.7906370	0.9315325
10:Phi	0.7273381	0.0318916	0.6605999	0.7852198
11:Phi	0.7501455	0.0332534	0.6795370	0.8095567
12:Phi	0.7889615	0.0363677	0.7090073	0.8515470
13:Phi	0.6615078	0.0345894	0.5907941	0.7256758
14:Phi	0.7190676	0.0413522	0.6314915	0.7926644
15:Phi	0.5454265	0.0413593	0.4638761	0.6246105
16:p	0.3479615	0.0069853	0.3343987	0.3617753

The simple arithmetic mean from these estimates was $\bar{\hat{S}} = 0.7045$, fairly close to the true sample mean of $\bar{S} = 0.7$. The naïve estimate of the sample variance is 0.00733, somewhat higher than the true underlying process variation $\sigma^2 = 0.005$. As discussed in Appendix D, this is because the naïve estimate includes sampling as well as process variation (and thus will generally be larger than process variation alone).

If we next fit the data using a simple 'dot' model, $\{S_.\ p_.\}$, the estimate for survival was $\bar{\hat{S}} = 0.7041$, while the estimated SE of the was $\widehat{SE}^2 = 0.00605$. If we (naïvely) take the square-root of the SE as our estimate of the variance, we'd come up with a value (0.0778) that would appear to be strongly positively biased. So, the 'dot' model does reasonably well at estimating the mean, but seems to fail miserably at estimating the correct variance.

Next, we try fitting the data using the PIM for the true model $\{S_t\ p_.\}$, using a design matrix constructed such that one of the estimated β terms (the intercept) corresponds directly to the arithmetic mean (shown below):

Design Matrix Specification (B = Beta)

B1 Int	B2 t1	B3 t2	B4 t3	B5 t4	B6 t5	B7 t6	B8 t7	B9 t8	B10 t9	B11 t10	B12 t11	B13 t12	B14 t13	B15 t14	Parm	B16 p
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1:Phi	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2:Phi	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	3:Phi	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	4:Phi	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	5:Phi	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	6:Phi	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	7:Phi	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	8:Phi	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	9:Phi	0
1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	10:Phi	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	11:Phi	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	12:Phi	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	13:Phi	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	14:Phi	0
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	15:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16:p	1

[This particular DM is sometimes referred to as a 'sum contrast'.] When run using an identity link (for reasons discussed earlier), the estimate for the first β term in this design matrix (β_1) is the estimated mean survival probability. For our simulated data, the estimated mean is $\hat{\beta}_1 = 0.7045$, which is identical to the arithmetic mean of the individual estimates derived from the 'true' generating model, $\{\varphi_t\, p_.\}$ (0.7045, above).

What about the variance? If you look at the output, you see that the estimated *standard error* for $\hat{\beta}_1 = 0.007354$. If we (naïvely) take the square-root of the estimated SE as our estimate of the variance, we'd come up with a value (0.0858) that would appear to be quite strongly positively biased. This might make some sense to you – you'd expect the estimated variance (the sum of sampling + process) to be larger than process variance alone. More on this in a moment – for now, keep this value of $\widehat{SE} = 0.007354$ in mind.

Next, an approach based on a Taylor series approximation to the variance of a function, using what is known as the 'Delta method' (introduced – briefly – earlier in this and earlier chapters, and in detail in Appendix B). Skipping the technical details for now, we are interested in approximating the variance of a *function* of parameters. In this case, the function is the mean of the survival estimates. So, for a set of k survival estimates $\{\hat{\varphi}_1, \hat{\varphi}_2, \ldots, \hat{\varphi}_k\}$, the function (which we'll call Y) is the simple arithmetic mean of the set of estimates:

$$Y = \hat{\bar{\varphi}} = \frac{\left(\hat{\varphi}_1 + \hat{\varphi}_2 + \cdots + \hat{\varphi}_k\right)}{k}.$$

Using the Delta method, the variance for the function Y can be approximated as

$$\widehat{\mathrm{var}}(\hat{Y}) \approx \mathbf{D}\boldsymbol{\Sigma}\mathbf{D}^{\mathsf{T}},$$

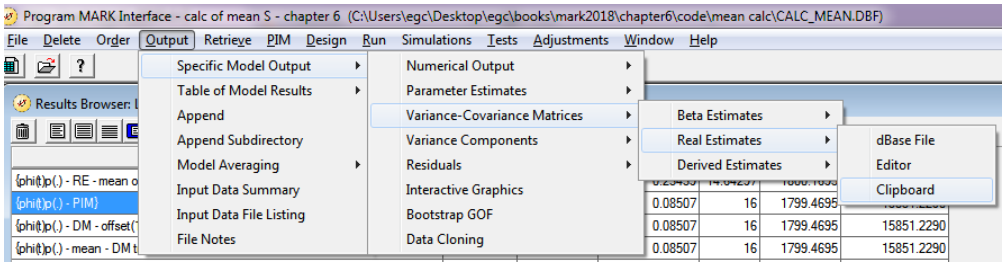
where $\mathbf{D}$ is the vector of the partial derivatives of the function with respect to each parameter in the function (i.e., the Jacobian of the function), and $\boldsymbol{\Sigma}$ is the variance-covariance matrix:

$$\widehat{\mathrm{cov}}(\hat{Y}) = \sum\!\!\widehat{}$$

$$= \begin{bmatrix} \widehat{\mathrm{var}}(\hat{\varphi}_1) & \widehat{\mathrm{cov}}(\hat{\varphi}_1, \hat{\varphi}_2) & \widehat{\mathrm{cov}}(\hat{\varphi}_1, \hat{\varphi}_3) & \cdots & \widehat{\mathrm{cov}}(\hat{\varphi}_1, \hat{\varphi}_k) \\ \widehat{\mathrm{cov}}(\hat{\varphi}_2, \hat{\varphi}_1) & \widehat{\mathrm{var}}(\hat{\varphi}_2) & \widehat{\mathrm{cov}}(\hat{\varphi}_2, \hat{\varphi}_3) & \cdots & \widehat{\mathrm{cov}}(\hat{\varphi}_2, \hat{\varphi}_k) \\ \widehat{\mathrm{cov}}(\hat{\varphi}_3, \hat{\varphi}_1) & \widehat{\mathrm{cov}}(\hat{\varphi}_3, \hat{\varphi}_2) & \ddots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ \widehat{\mathrm{cov}}(\hat{\varphi}_k, \hat{\varphi}_1) & \widehat{\mathrm{cov}}(\hat{\varphi}_k, \hat{\varphi}_2) & \cdots & \cdots & \widehat{\mathrm{var}}(\hat{\varphi}_k) \end{bmatrix}$$

So, estimates of the variances of the (real) parameter estimates along the diagonal, and covariances off the diagonal.

The variance-covariance matrix for the real parameters can be easily output from **MARK** – in the following we output it to the Windows clipboard, which we can then save to a space-delimited ASCII file which we can read into some other piece of software we might use to handle the calculations:



All thats left is 'a little bit of calculus' (i.e., deriving the Jacobian), which for this particular problem is trivial – it is a vector of 15 elements (i.e., the function, differentiated with respect to each of the 15 survival parameters in turn), which for this function, is simply:

$$\left[\frac{1}{15}\ \frac{1}{15}\ \cdots\ \frac{1}{15}\right].$$

So, using the variance-covariance matrix for the survival estimates from model $\{\varphi_t\ p_.\}$,* we estimate the variance for our function (i.e., mean survival) as:

$$\begin{aligned}\widehat{\mathrm{var}}(\hat{\bar{Y}}) &\approx \mathbf{D\Sigma D}^{\mathsf{T}} \\ &= 0.00005407449290.\end{aligned}$$

But, this value isn't even remotely close to the estimates from the previous two methods we tried. That is, until you realize that

$$\begin{aligned}\sqrt{0.00005407449290} &= 0.007353536081 \\ &\approx 0.007354.\end{aligned}$$

which is identical to the estimated standard error for $\hat{\beta}_1 = 0.007354$ from the DM-based approach we just tried (above).

In fact, what we have just derived (using either the DM-based model or the Delta method) is an estimate of the *sampling variance* of the estimates. Remember, the parameter values used to simulate the encounter data (p. 93) were simply one realization of the true underlying mean of 0.7, with true process variance of 0.005.

A simple numerical experiment will help make it clear what is going on, in terms of estimating the variance. Using the simulation capabilities in **MARK** (covered in detail in Appendix A), we simulated $R = 100, 250, 500, 1{,}000$ and 2,500 new individuals marked and released per occasion. At each occasion, we sampled from a beta distribution centered on mean of 0.7, with process variance of 0.005, i.e., $\mathcal{B}(28.7, 12.3)$, to set the true survival probability for that sampling occasion. For each simulation, we derived an estimate of $\hat{\beta}_1$, and the SE for the estimate, using model $\{\varphi_t\ p_.\}$, constrained using the DM described above.

The summary results over 1,000 simulations for each design point, for $\hat{\beta}_1$ and $\widehat{SE}(\beta_1)$ respectively, are tabulated in the following:

	$\hat{\beta}_1$			$\widehat{SE}(\beta_1)$		
R	mean	var	SD	mean	var	SD
100	0.701	0.0004405	0.020988	0.011	«0.001	0.001
250	0.700	0.0003667	0.019150	0.007	«0.001	0.000
500	0.700	0.0003638	0.019074	0.005	«0.001	0.000
1,000	0.700	0.0003388	0.018406	0.004	«0.001	0.000
2,500	0.700	0.0003293	0.018148	0.002	«0.001	0.000

Before we try to interpret the numbers in this table, recall that the true process variance is 0.005. As such, given 15 occasions, we'd expect a true process SE of $\sqrt{0.005/15} = 0.01825742$. Remember that the SE is an estimate of the SD of a sample of statistics (in this case, the mean), based on a set of replicated samples from an underlying distribution.

* **MARK** outputs the full variance-covariance matrix, over all of the parameters in the model. For this example, we need to truncate/subset the matrix to include the variances and covariances of the survival estimates only – here, we want the upper-left (15×15) sub-matrix of the full variance-covariance matrix.

For example, the following code snippet simulates 15 random beta deviates (corresponding 15 occasions), with mean 0.7 and process variances 0.005, calculates the mean, and then outputs it. This process is repeated 1,000 times. The SD of this sample of 1,000 randomly generated means is the SE of the mean:

```
  # simulate Beta for mu=0.7, sigma2=0.005
samples <- 1000; n_occas <- 15;
hold <- array(samples)

for (rep in 1:samples) { sim <- rbeta(n_occas,28.7,12.3);
                         hold[rep]=mean(sim); }

print(mean(hold));  # mean
  [1] 0.7000336

print(sqrt(var(hold)));  # SD (= SE)
  [1] 0.01827892
```

The SD of the simulated distribution (0.018279) is quite close to the expected SE given a true process variance of 0.005 and 15 occasions (0.018257). The approximation would be improved by drawing more samples.

Now, look back at the tabulated results from our **MARK** simulations, on the previous page. We see that the mean of the survival values is close to the true parametric value of 0.7, regardless of the number of releases. Perhaps not surprisingly, though, the variance (and SD) of $\hat{\beta}_1$ decreases as the number of releases increases. In addition, the mean $\widehat{SE}(\beta_1)$ decreases with increasing number of releases.

Look closely at the SD for the mean estimate $\hat{\beta}_1$ – it decreases nonlinearly with increasing number of releases, approaching an asymptote at ≈ 0.018257. That number might look familiar – it is, in fact, the expected SE given a true process variance of 0.005 and 15 occasions!

OK – so what? Well, this is important since as the number of releases R at each occasion increases, all other things being equal, then the proportion of *total* variance owing to *sampling* variation decreases, with a corresponding increase in the proportion of total variance due to *process* variance. In that case, the estimated SE for $\hat{\beta}_1$ decreases, approaching 0 as R gets very large (as we see in the table on the preceding page).

This implies that the variance of the estimates of $\hat{\beta}_1$ represents *total* variance (process + sampling), while the mean $\widehat{SE}(\beta_1)$ represents sampling variance (specifically, the square of the mean $\widehat{SE}(\beta_1)$). As such, the difference between the total variance and the sampling variance should be the process variance. For a given number of releases, and 15 occasions, we can express this difference simply as

$$\left(\mathrm{var}(\hat{\beta}_1) - \widehat{SE}(\beta_1)^2\right) \times 15.$$

So, to complete the table, by including a column of our estimate of process variance derived using this equation:

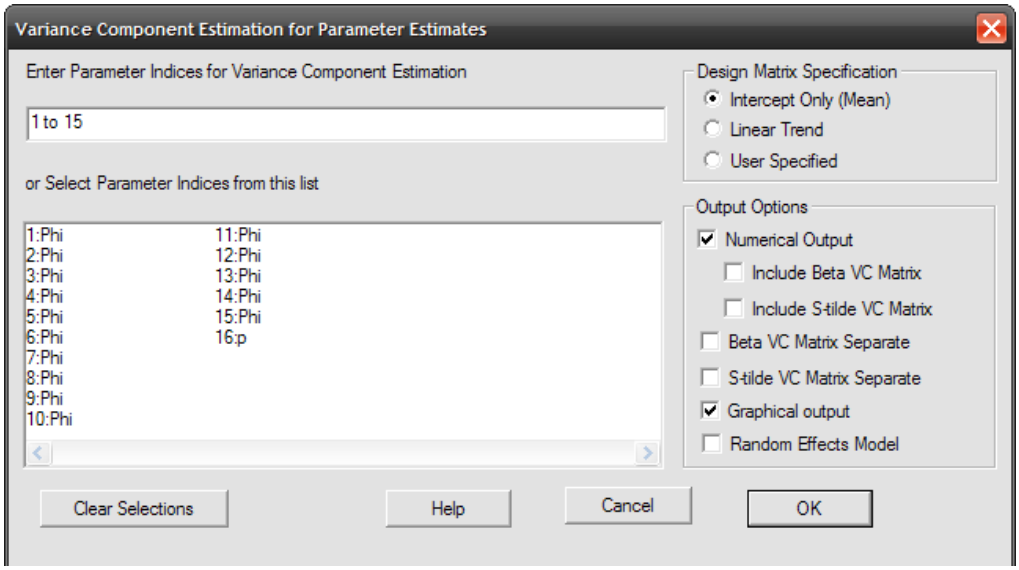
	$\hat{\beta}_1$		$\widehat{SE}(\beta_1)$	
R	mean	var	mean	$\hat{\sigma}^2_{\text{process}}$
100	0.701	0.0004405	0.011	0.004688
250	0.700	0.0003667	0.007	0.004766
500	0.700	0.0003638	0.005	0.005082
1,000	0.700	0.0003388	0.004	0.004842
2,500	0.700	0.0003293	0.002	0.004880
				$\bar{x} = 0.004856$

So, in fact, we do pretty well at estimating the process variance: rounding to 4 digits, 0.0049 is pretty close to the true value of 0.005.

However, while all this is 'interesting', the only way the SE estimated using either $\hat{\beta}_1$ from the 'sum contrast' DM, or the Delta method, would make any sense (i.e., be interpretable in terms of interest in process variance), would be if we knew the variance of the mean of the survival parameters, which we don't. In other words, you can't really get there from here.

Fear not – there is a solution, which we now introduce. To get a robust estimate of not just the mean (which the other methods outlined above would give you), but the process variance as well (which the preceding methods don't), we need to apply a variance components approach (based on a random effects intercept only model). While variance components and random effects models are introduced in more detail in Appendix D, for now we'll demonstrate the simple mechanics of deriving an estimate of the mean, and process variation using the variance components capabilities in **MARK**, based on a 'methods of moments' approach (we could also accomplish the same thing using the MCMC capabilities in **MARK** – see Appendix E). In fact, for the simple objective at hand, the mechanics are very easy.

First, retrieve model $\{\varphi_t\, p_.\}$ in the browser. Then, select the menu option '**`Output | Specific Model Output | Variance Components | Real Parameter Estimates`**'. This will spawn another window (shown below) asking you to select the parameter(s) you want to work with, plus some other options:



On the right hand side, we've selected '**`intercept only`**' (corresponding to the mean only model), plus various output options. We'll defer 'deep understanding' of what the various options in this window 'do' for now.

Once you hit the '**`OK`**' button, **MARK** will output various 'estimates' into the editor. The estimates for our example data are shown at the top of the next page. The top line gives the estimated mean ('`Beta-hat`') as 0.7286, which is close to (but slightly different than) the simple arithmetic mean. Both are very close to the true sample mean of 0.72. More important, here, is the reported `SE(Beta-hat)` of 0.01961. `SE(Beta-hat)` is the standard error of the estimated mean and includes *both* process *and* sampling variance. We use this estimated SE in the usual way to derive 95% CI to the estimated mean.

The table immediately below these two values represent the interval-specific survival estimates, their standard errors, and corresponding *shrinkage estimates* (if you don't know what a 'shrinkage' estimate is, don't worry at this stage). Skipping the technical details, these shrinkage estimates are part of the calculation(s) **MARK** uses to separate the process and sampling variance.

This table of survival estimates is followed by two 'estimates' of the *process* variance (i.e., the total variance minus the sampling variance) – the '`naive estimate of sigma^2`' (which is an estimate of

```
   Beta-hat SE(Beta-hat) Label
----------- ------------ ----------------
   0.728645     0.019611 Intercept

Par. Num     S-hat    SE(S-hat)    S-tilde  SE(S-tilde) RMSE(S-tilde)
-------- ---------- ----------- ----------- ----------- -------------
    1      0.628679    0.051515    0.644127    0.041942     0.044696
    2      0.648572    0.042043    0.657919    0.035587     0.036793
    3      0.800416    0.042414    0.788272    0.035812     0.037815
    4      0.739101    0.038514    0.740082    0.033125     0.033140
    5      0.710756    0.035792    0.712923    0.031344     0.031419
    6      0.718450    0.033820    0.720828    0.029965     0.030059
    7      0.788729    0.033900    0.784713    0.030056     0.030323
    8      0.776127    0.032147    0.778846    0.028671     0.028799
    9      0.877570    0.035129    0.864952    0.030928     0.033403
   10      0.727338    0.031892    0.733439    0.028436     0.029083
   11      0.750146    0.033253    0.750448    0.029523     0.029525
   12      0.788962    0.036368    0.781292    0.031799     0.032710
   13      0.661508    0.034589    0.669262    0.030393     0.031367
   14      0.719068    0.041352    0.711415    0.035267     0.036088
   15      0.545426    0.041359    0.568617    0.035729     0.042596

Naive estimate of sigma^2 = 0.0050871 with 95% CI (0.0020283 to 0.0149000)

Estimate of sigma^2 = 0.0052204 with 95% CI (0.0022312 to 0.0149840)
```

the total variance minus the sampling variances), and the '`estimate of sigma^2`' (from Burnham's 'moments' estimator). These are the estimates we're after – for various technical reasons (see Appendix D), the second estimate is preferred. For our simulated data, the process variance is calculated as 0.00522, which is very close to the true value 0.005.

In conclusion, to get a robust estimate of the process variance (and a less-biased estimate of the mean), you need to do a bit more work than you might have anticipated.

end sidebar

6.16. Model averaging over linear covariates

As introduced in Chapter 4, model averaging is a very important concept. Deriving model averaged estimates of different parameters explicitly accounts for model selection uncertainty. In many cases, the mechanics of model averaging are pretty straightforward.

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 effort covariates (in hours) 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 (top of the next page) to our data:

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

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 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. We'll build all of the models using a design matrix approach. 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.

Here are the results of fitting our 4 candidate models to the full dipper data. 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.

Results Browser: Live Recaptures (CJS)

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

Now, if we simply wanted to generate a 'time-' and 'sex-specific' average encounter probability, for each time interval and sex, we can use the standard model averaging routine(s) in **MARK**, as introduced in Chapter 4. **MARK** allows you to model average for each combination of classification factors in the model – for this analysis, that would include TIME (which is implicitly included in all models) and SEX (which was specified as a 'grouping' variable' when you set up the analysis.

For example, for males, we would simply select any one element in each of the 'PIM columns' (each of which corresponds to a particular time interval) in the model averaging interface, as shown below:

Model Averaging Parameter Selection

Apparent Survival Parameter (Phi) m | Apparent Survival Parameter (Phi) f | Recapture Parameter (p) m | Re

☑ 43 ☑ 44 ☑ 45 ☑ 46 ☑ 47 ☑ 48
☐ 49 ☐ 50 ☐ 51 ☐ 52 ☐ 53
☐ 54 ☐ 55 ☐ 56 ☐ 57
☐ 58 ☐ 59 ☐ 60
☐ 61 ☐ 62
☐ 63

The model averaged estimate for $\hat{p}_{2,males}$ for the first encounter occasion, for males is shown at the top of the next page. In looking at the output, we're reminded about the basic mechanics for model averaging. The actual estimate, $\hat{p}_{2,males} = 0.9152981$, is simply the average of the estimate from each of the 4 candidate models, weighted by the normalized AIC weights. This is a straightforward calculation.

Estimates only for data type Live Recaptures (CJS)

Recapture Parameter (p) m Parameter 43

Model	Weight	Estimate	Standard Error
{phi(t)p(SEX)}	0.43880	0.9227653	0.0361568
{phi(t)p(HOUR)}	0.33773	0.9007193	0.0298725
{phi(t)p(SEX+HOUR)}	0.15968	0.9217613	0.0366872
{phi(t)p(SEX+HOUR+SEX*HOUR)}	0.06378	0.9249400	0.0371166
Weighted Average		0.9152981	0.0341803
Unconditional SE			0.0358703

You might recall, though, that the calculation of the 'correct' standard error for the weighted average is somewhat more involved. The SE for each estimate from each model is called the *conditional* standard error. While it might seem intuitive to simply take the AIC weighted average of these model-specific conditional standard eros (which results in an estimate of 0.0341803), doing so ignores variation among models. As introduced in Chapter 4, the correct way to calculate the *unconditional* SE (i.e., the standard error for the average that is not conditional on a particular model) is

$$\widehat{\text{var}}(\bar{\hat{p}}) = \sum_{i=1}^{R} w_i \left[\widehat{\text{var}}(\hat{p}_i \mid M_i) + (\hat{p}_i - \bar{\hat{p}})^2 \right], \quad \text{where} \quad \bar{\hat{p}} = \sum_{i=1}^{R} w_i \hat{p}_i,$$

and the w_i are the normalized Akaike weights. The subscript i refers to the i^{th} model. The value, $\bar{p}$, is the model averaged estimate of p over R models ($i = 1, 2, \ldots, R$). While this isn't that difficult to do by hand, it isn't necessary in this case, since **MARK** handles the calculation for you. In this case, the *unconditional* SE of $\hat{p}_{m,2}$ is $\widehat{\text{SE}} = 0.0358703$.

But, what if instead of sex- and time-specific model averaged estimates for p, you instead want the model averaged estimate of p as a function of the environmental covariate, HOURS? Here is where things get more interesting. You want to derive the model averaged estimate of the encounter probability, as a function of the number of hours of observation.

You might appear to have a couple of options. First, you might be interested in the model average of the β parameter in each model for the HOURS covariate. That seems quite reasonable – the β coefficient for HOURS is the measure of the effect of a unit change in HOURS of observation on encounter probability. You average over models to generate an average of the effect of HOURS on detection probability.

But, while this seems reasonable, there are in fact a couple of problems with this approach. First, and perhaps most obviously, what do you do for models where HOURS is not a term in the model? For our present example, model M_4 does not include HOURS as a term in the model – how would we account for this model when averaging over all models in the model set?

One approach which at first seems reasonable (and somewhat clever) is to use the logical argument that 'if a model doesn't contain HOURS as a term, then the β coefficient for HOURS is logically 0'. Let's adopt this approach, and create a table of the model-specific estimates for the β coefficient, corresponding to the HOURS covariate, from each of our 4 candidate models:

model	$\hat{\beta}$	AIC weight
M_1	0.0463428	0.06378
M_2	0.0195470	0.15968
M_3	0.0169393	0.33773
M_4	0.0000000	0.43880

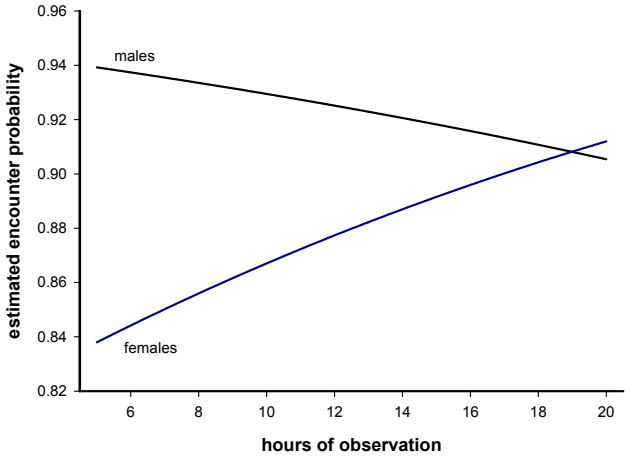
At this point, you might simply take a weighted average of the model-specific estimates for β (which for this example, would be $\bar{\hat{\beta}} = 0.01180$).

But, there are at least two important issues which we need to consider. First, recall from earlier in this chapter that the interpretation of β does not directly take into account the other terms in the model. For example, consider model M_1, which corresponds to

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

The estimate for $\hat{\beta}_3(\texttt{HOURS}) = 0.0463428$, which suggests that as the hours of observation increases, then so too does the probability of encounter.

But, what about the interaction of HOURS and SEX? Recall from section 6.8.2 that there is a strong interaction between SEX, HOURS, and encounter probability. As shown in the following figure,



encounter probability increases with HOURS, but only for females! For males, encounter probability actually declines with more HOURS of observation.

The point we're trying to make here is that considering a β estimate 'by itself', without taking the other terms of the model into account, can lead to all sorts of conclusions which may not actually reflect reality. And, as such, model averaging over those estimates is subject to the same problem.

The second potential problem concerns the standard error calculation. While β for model M_4 is logically 0, what is the SE for this 'value'? We say 'value', because it is not an 'estimate', and thus, it isn't immediately obvious how to include (or not) the conditional SE for model M_4 (which we might set to 0) into our calculation of the unconditional SE for $\bar{\beta}$.

So, in short summary, don't try to model average β's – it will very likely get you into trouble, and there are a number of technical considerations which are not solved. In fact, these are some of the reasons that **MARK** does not have an option for model averaging β terms.

But, what if you really want a model averaged estimate of the relationship between HOURS and encounter probability? In your mind you might be imagining a plot of HOURS on the horizontal axis, encounter probability, p, on the vertical axis (like the figure shown above), but averaged over all models.

In fact, you can generate exactly what you want, albeit with a bit of work. The first part of the process, deriving model averaged estimates of p as a function of the HOURS covariate is easy – you simply take the estimated linear model for each of the candidate models, and derive estimates for p as a function of

different values for the covariate HOURS. Then, simply take the average over models for a given covariate value, weighting by the model-specific normalized AIC weights.

In the following table, we show the calculated average accounter probability, $\bar{\hat{p}}$, for each of 4 different values of the HOURS covariate (5, 10, 15, and 20 hours). In order to make the calculations for this example, we either need to (i) pick one sex or the other (SEX=1, or SEX=0), or (ii) use the average value for sex (based on the proportion of males and females in the sample). In other words, we need to specify (or 'control for') the other variables in the models. We'll use SEX=1 (males) for our demonstration:

		HOURS			
model	AIC *weight*	5	10	15	20
M_1	0.0638	0.9392547	0.9294700	0.9182464	0.9054185
M_2	0.1600	0.9111449	0.9187497	0.9257566	0.9322038
M_3	0.3377	0.8894343	0.8974927	0.9050265	0.9120609
M_4	0.4388	0.9227652	0.9227652	0.9227652	0.9227652
	$\bar{\hat{p}}$	**0.91098**	**0.91429**	**0.91724**	**0.91983**

To make sure you know how the numbers in this table are generated, take the estimated linear model for model M_1:

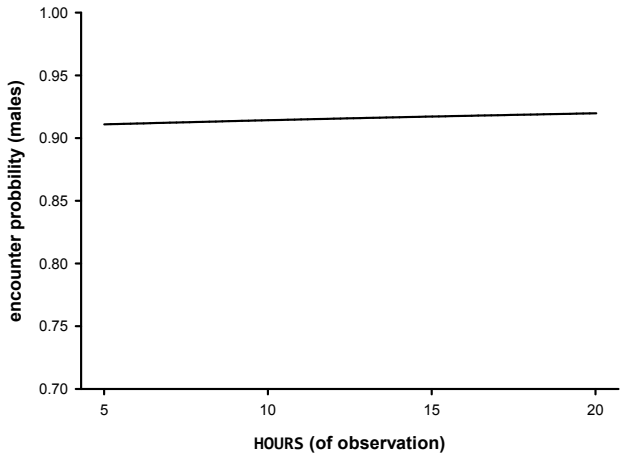
$$\begin{aligned}\text{logit}(\hat{p}) &= \hat{\beta}_1 + \hat{\beta}_2(\texttt{SEX}) + \hat{\beta}_3(\texttt{HOURS}) + \hat{\beta}_4(\texttt{SEX.HOURS}) \\ &= 1.4115973 + 1.4866197(\texttt{SEX}) + 0.0463460(\texttt{HOURS}) + (-0.0783101)(\texttt{SEX.HOURS}).\end{aligned}$$

If we substitute in SEX=1, and HOURS=5, we get the value

$$\begin{aligned}\text{logit}(\hat{p}) &= 1.4115973 + 1.4866197(1) + 0.0463460(5) + (-0.0783101)(5) \\ &= 2.738396,\end{aligned}$$

which, when back-transformed from the logit scale, yields 0.9392547, which is the value shown in the table, above. We leave it as an exercise to confirm the remaining values in the table, and weighted averages over those models.

Let's plot these model averaged values:



The figure looks a bit 'out of scale', but that's only because we're 'leaving enough room' on the vertical axis, for what comes next – the SE calculations! What we want to add to the plot now are the unconditional SE for each point on the line (which we'll calculate for the 4 points only: 5, 10, 15 and 20 HOURS. Normally, you would use more points, but our intent is only to demonstrate the mechanics).

How do we derive the unconditional SE for each model averaged estimate of p? The steps are easy in principle, but somewhat laborious in practice. First, we derive the *conditional* SE for $\hat{p}$ for different values of the covariate HOURS, for a given model, using the approach outlined in the -sidebar- back on p. 49 (if you skipped reading it before, you might need to go back and have a look at it now). Then, once you've calculate the conditional SE for each value for HOURS, for each model, you (ii) use the normalized AIC model weights to generate unconditional SE estimates using the formula noted earlier:

$$\widehat{\mathrm{var}}(\bar{\hat{p}}) = \sum_{i=1}^{R} w_i \left[\widehat{\mathrm{var}}(\hat{p}_i \mid M_i) + (\hat{p}_i - \bar{\hat{p}})^2 \right], \quad \text{where} \quad \bar{\hat{p}} = \sum_{i=1}^{R} w_i \hat{p}_i .$$

The laborious part of all this is deriving the conditional SE's for each estimate of p for a given value of HOURS, for each model. As noted on p. 49, this involves application of the Delta method, for each model in turn. In fact, because the models in our candidate model set are all nested within the most general model (M_1), this process can be automated fairly well.

Let's run through the complete calculations for HOURS = 5. As noted on p. 49, we can approximate the variance of some multi-variable function **Y** as

$$\widehat{\mathrm{var}}(\hat{Y}) \approx \mathbf{D}\boldsymbol{\Sigma}\mathbf{D}^{\mathsf{T}},$$

where **D** is the matrix of partial derivatives of the function **Y** with respect to each parameter, and **Σ** is the variance-covariance matrix for the parameters in the function.

So, all we need to do is (i) take the vector of partial derivatives of the function (i.e., the linear model) with respect to each parameter in turn (i.e., derive the Jacobian of the model with respect to the model parameters), **D**, (ii) right-multiply this vector by the variance-covariance matrix, **Σ**, and (iii) right-multiply the resulting product by the transpose of the original vector of partial derivatives, $\mathbf{D}^{\mathsf{T}}$. Step (i) involves trivial calculus, step (ii) involves extracting the variance-covariance matrix from **MARK** output for that model, and step (iii) involves a bit of linear algebra. No one step in this process is particularly difficult – there are simply lots of steps.

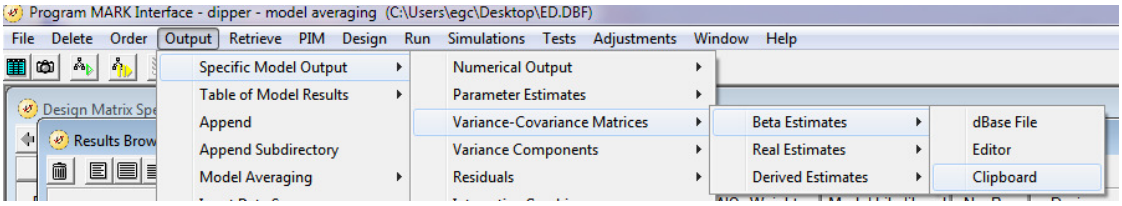
Here are the Jacobian vectors for our 4 models:

model	model structure	Jacobian
M_1	$\beta_1 + \beta_2(\mathtt{SEX}) + \beta_3(\mathtt{HOURS}) + \beta_4(\mathtt{SEX.HOURS})$	[1 SEX HOURS SEX.HOURS]
M_2	$\beta_1 + \beta_2(\mathtt{SEX}) + \beta_3(\mathtt{HOURS})$	[1 SEX HOURS]
M_3	$\beta_1 + \beta_2(\mathtt{HOURS})$	[1 HOURS]
M_4	$\beta_1 + \beta_2(\mathtt{SEX})$	[1 SEX]

You should notice immediately that the Jacobian for the linear model is simply the linear model without the β coefficients, using a '1' for the intercept, β_1. As a result, depending on your facility with a computer, you might be able to generate the Jacobian very quickly over your model set. For smaller model sets, even doing it 'by hand' should only take a few minutes.

The next step is to extract the variance-covariance matrix, $\boldsymbol{\Sigma}_i$, for the β_i coefficients, for each model M_i in the model set. This is quite straightforward to do in **MARK** – all you need to do is select the

model from the results browser, and then '**Output | Specific Model Output | Variance-Covariance Matrices | Beta Estimates**':



You may recall that your preferred options are to output to the clipboard (as show), or a dBase file (which you can open in Excel). In general, do not use the '**Editor**' output option, since outputting to the editor will truncate (round) the values in the matrix to degree that your results might be suspect.

Beyond that, the only challenge is in figuring out which rows and columns in the variance-covariance matrix you need to extract. In other words, which rows and columns correspond to the β_i coefficients in your linear model.

Consider for example, model M_2, which corresponds to

$$\text{model } M_2 \quad \text{logit}(\hat{p}) = \hat{\beta}_1 + \hat{\beta}_2(\texttt{SEX}) + \hat{\beta}_3(\texttt{HOURS})$$

We see there are 3 coefficients (β_1, β_2 and β_3) in this model, and thus the variance covariance matrix for model M_2, $\boldsymbol{\Sigma}_2$, would be a (3×3) matrix, with the variances for each β_i parameter along the diagonal, and the covariances between parameters off the diagonal (the matrix is shown at the top of the next page).

$$\boldsymbol{\Sigma}_2 = \begin{bmatrix} \widehat{\text{Var}}(\hat{\beta}_1) & \widehat{\text{Cov}}(\hat{\beta}_1, \hat{\beta}_2) & \widehat{\text{Cov}}(\hat{\beta}_1, \hat{\beta}_3) \\ \widehat{\text{Cov}}(\hat{\beta}_2, \hat{\beta}_1) & \widehat{\text{Var}}(\hat{\beta}_2) & \widehat{\text{Cov}}(\hat{\beta}_2, \hat{\beta}_3) \\ \widehat{\text{Cov}}(\hat{\beta}_3, \hat{\beta}_1) & \widehat{\text{Cov}}(\hat{\beta}_3, \hat{\beta}_2) & \widehat{\text{Var}}(\hat{\beta}_3) \end{bmatrix}.$$

However, when you output the variance-covariance from **MARK**, it outputs the matrix over all of the β terms, not just the ones you are interested in. You will need to keep track of which rows and columns correspond to the parameters you are interested in.

For model $\{\varphi_t\, p_{\text{S+H}}\}$, there are 9 total β parameters (6 for apparent survival probability, and 3 for the encounter probability). So, **MARK** outputs a (9×9) matrix. We want the variance-covariance matrix for the encounter probability parameters only, which corresponds to the (3×3) sub-matrix in the lower right-hand corner of the full matrix output by **MARK** (shaded, below):

```
04291   -0.0607603771   -0.0566947595   0.0048781146   0.0026645796
2415     0.0581297801    0.0356438243  -0.0092377126  -0.0020730364
0024     0.0555947688   -0.0089377612  -0.0004623550   0.0015674488
5473     0.0600130718    0.0503985106  -0.0040747025  -0.0025308180
3560     0.0588581256    0.0799344963  -0.0062554837  -0.0048713656
1256     0.1165972299    0.0843062150  -0.0070651030  -0.0051443046
4963     0.0843062150    1.1435035261  -0.1827007130  -0.0749623706
54837   -0.0070651030   -0.1827007130   0.4077519211   0.0016836049
13656   -0.0051443046   -0.0749623706   0.0016836049   0.0057802739

..+....8....+....9....+...10....+...11....+...12....+...13....+...14....
```

You simply need to extract this (3×3) sub-matrix, and 'paste it' in some fashion into the software application you might use for the final step, which involves the 'linear algebra' of multiplying the Jacobian and variance covariance-matrices together.

Now, at this point, you have a decision to make – the variance-covariance matrix output by **MARK** is 'numeric', whereas in the first step, we derived the Jacobian for each model 'symbolically'. While there are many available software applications that can handle mixing numeric and symbolic calculations (e.g., **Maple**, **Mathematica**, **Maxima**...), it is mechanically simpler to use one or the other (i.e., numeric, or symbolic). Since the variance-covariance matrix output by **MARK** is numeric, it is probably simplest to translate our Jacobian from 'symbolic' to 'numeric'. All this translation requires is entering the appropriate numeric value(s) into the symbolic Jacobian vector.

For our present example, we're considering males (SEX = 1) and 5 hours of observation (HOURS = 5). Thus, our translation of the Jacobian from symbolic → numeric for our 4 models would look like:

model	symbolic Jacobian	numeric Jacobian
M_1	[1 SEX HOURS SEX.HOURS]	[1 1 5 5]
M_2	[1 SEX HOURS]	[1 1 5]
M_3	[1 HOURS]	[1 5]
M_4	[1 SEX]	[1 1]

Make sure you understand how the numerically evaluated Jacobian matrices were derived.

All that's really left is to take the numeric Jacobian, $\mathbf{D}_i$, and the variance-covariance matrices, $\mathbf{\Sigma}_i$, for the different models, and 'do the linear algebra'. In other words, calculate

$$\mathbf{D}_i \mathbf{\Sigma}_i \mathbf{D}_i^{\mathsf{T}}.$$

For example, for model M_2 (SEX+HOURS), this might be implemented in an **R** script as follows:

```
# enter numeric Jacobian vector
 jac <- matrix(c(1,1,5),1,3,byrow=T);

# transpose Jacobian vector
  t_jac <- t(jac);

# enter variance-covariance matrix (cut and paste from MARK)
   vc <- matrix(c( 1.1435035261, -0.1827007130, -0.0749623706,
                  -0.1827007130,  0.4077519211,  0.0016836049,
                  -0.0749623706,  0.0016836049,  0.0057802739),3,3,byrow=T);

# multiply jac x vc matrix x transpose(jac)
   var_logit <- jac %*% vc %*%  t_jac;
   print(var_logit);
```

If we run this script, we find that the approximate variance for our estimate of $\hat{p}_{\mathtt{HOURS=5,SEX=1}}$ from model M_2, *on the logit scale*, given SEX = 1, HOURS = 5, is

$$\begin{aligned}\widehat{\mathrm{Var}} &\approx \mathbf{D}_i \mathbf{\Sigma}_i \mathbf{D}_i^{\mathsf{T}} \\ &= 0.5975732.\end{aligned}$$

with $\widehat{\text{SE}} = \sqrt{0.5975732} = 0.773029$.

All we really need to do next is (i) repeat this calculation of the *conditional* variance and SE for each of the remaining models, and then (ii) from these estimates, derive the estimate of the *unconditional* variance and standard error, over all the candidate models.

The following tabulates the *conditional* variances for $\hat{p}_{\texttt{SEX=1,HOURS=5}}$, *on the logit scale*, for all 4 candidate models:

model	AIC weight	logit($\hat{p}$)	conditional variance
M_1	0.0638	2.7383960	1.4069833
M_2	0.1600	2.3276955	0.5975763
M_3	0.3377	2.0849753	0.4738056
M_4	0.4388	2.4805252	0.2573784

Now, at this point, you could either (i) derive the unconditional variances on the logit scale, and then back-transform everything, or (ii) back-transform the conditional variances each individual model from the logit scale to the real probability scale, and then do the calculations of the unconditional variance on the real scale. While this seems almost like a semantic point, it isn't entirely so – because of Jensen's inequality. The finer points are discussed in the following -sidebar-. Since the 95% CI are derived on the logit scale, and then back-transformed, perhaps it makes sense to use the value of the model-averaged value on the logit scale. We'll adopt this convention here.

begin sidebar

Jensen's inequality – logit or probability scale?

Jensen's inequality 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.

For example, let the function of x be $f(x) = x^2$. Let the set of x be $(3, 2, 4, 6, 3)$. The mean of the set is 3.6. The function applied to the mean is $3.6^2 = 12.96$. The set of the function of x is $(9, 4, 16, 36, 9)$. The average of this set of the function of x is 14.8. So, we see that, as per Jensen's inequality, the expectation (average) of the function is different than the function of the average – in fact, as expected the mean of the function of x is greater than the function of the mean of x.

In the present context, the back-transform of the model averaged value of logit($\hat{p}$), for example, is not the same as the model averaged value of the back-transforms of the individual estimates of $\hat{p}$ from each model. In other words, if the model-averaged value for p on the logit scale is

$$\text{logit}(\bar{\hat{p}}) = w_1\text{logit}(\hat{p}_{M_1}) + w_2\text{logit}(\hat{p}_{M_2}) + w_3\text{logit}(\hat{p}_{M_3}) + w_4\text{logit}(\hat{p}_{M_4}),$$

where w_i is the normalized AIC weight for model i, and if the model averaged value for p on the normal probability scale is

$$\bar{\hat{p}} = w_1\hat{p}_{M_1} + w_2\hat{p}_{M_2} + w_3\hat{p}_{M_3} + w_4\hat{p}_{M_4},$$

then

$$\frac{e^{(\text{logit}(\bar{\hat{p}}))}}{1 + e^{(\text{logit}(\bar{\hat{p}}))}} \neq \bar{\hat{p}}.$$

Even though the SE are calculated on the logit scale before back-transforming to the real scale, because the calculation of the unconditional SE is a function of the model average of the parameter, which model averaged value you use (the back-transform of the model averaged value of logit($\hat{p}$),

versus the model averaged value of the back-transforms of the individual estimates of $\hat{p}$ from each model), will make a difference in your calculations.

While the difference between the two is generally quite small, you do need to decide which model averaged parameter to use.

end sidebar

The model averaged estimate of $\bar{\hat{p}}$, *on the logit scale*, is 2.339692.

Given

$$\widehat{\text{var}}(\bar{\hat{p}}) = \sum_{i=1}^{R} w_i \left[\widehat{\text{var}}(\hat{p}_i \mid M_i) + (\hat{p}_i - \bar{\hat{p}})^2 \right],$$

then we can show that for `SEX=1,HOURS=5`,

$$\begin{aligned}
\text{logit}\left(\widehat{\text{var}}(\bar{\hat{p}})\right) &= \sum_{i=1}^{R} w_i \left[\widehat{\text{var}}(\hat{p}_i \mid M_i) + (\hat{p}_i - \bar{\hat{p}})^2 \right] \\
&= 0.0638\left[1.40698 + (2.73840 - 2.33969)^2\right] + 0.1600\left[0.59758 + (2.32770 - 2.33969)^2\right] \\
&\quad + 0.3377\left[0.47381 + (2.08498 - 2.33969)^2\right] + 0.4388\left[0.25738 + (2.40538 - 2.33969)^2\right] \\
&= 0.499097,
\end{aligned}$$

with $\widehat{\text{logit}(\text{SE})} = \sqrt{0.499097} = 0.70647$.

Finally, we want to use our estimated variance to derive a 95% confidence interval around our estimate, and we want both the estimate and the 95% CI for the estimate on the *normal probability scale*.

```
# estimate encounter probability on logit scale - parameter estimates from MARK
  logit_avg_p <- 2.33969
  logit_var <-0.499097; logit_se <- sqrt(logit_var);

# now derive LCI and UCI on logit scale
  logit_uci <- logit_avg_p+1.96*logit_se;
  logit_lci <- logit_avg_p-1.96*logit_se;

# back-transform everything from logit -> probability scale
  p <- exp(logit_avg_p)/(1+exp(logit_avg_p));
  uci <- exp(logit_uci)/(1+exp(logit_uci));
  lci <- exp(logit_lci)/(1+exp(logit_lci));

# put everything together
  results <- cbind(p,lci,uci)
  print(results);
```

The output from this script is

```
[1,]  0.9121112 0.7221222 0.9764401
```

where the back-transformed model-averaged estimate of $\bar{\hat{p}}_{\texttt{(SEX=1,HOURS=5)}}$ on the normal probability scale is 0.9121112, with an estimated 95% CI of [0.7221222, 0.9764401].

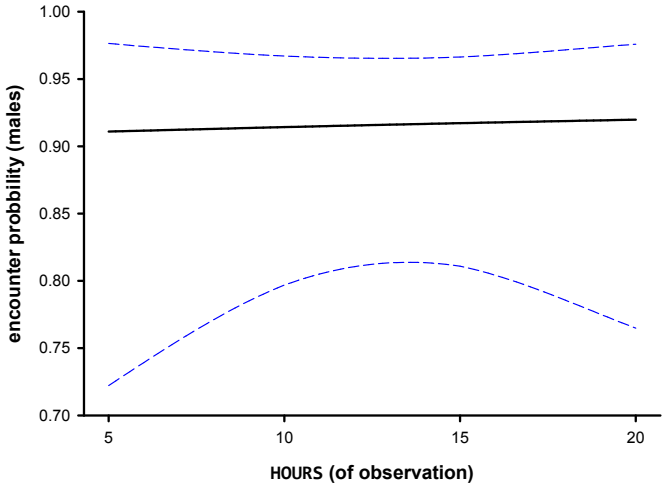
Again, note that the SE and 95% CI are first derived on the *logit* scale, and then back-transformed. This is done to guarantee that the calculated 95% CI is $[0, 1]$ bounded for parameters (like φ or p) that are $[0, 1]$ bounded.

And, a reminder that because the logit transform is not linear, the *reconstituted* 95% CI will not be symmetrical around the parameter estimate, especially for parameters estimated near the $[0, 1]$ boundaries.

Taking this approach, and replicating it for the other values of the HOURS covariate (i.e., repeat the preceding, but for HOURS = 10, HOURS = 15, and HOURS = 20), we can derive estimates of the model averaged values for p, for 5, 10, 15 and 20 observation HOURS, and their *unconditional* 95% CI. These estimates are tabulated here:

HOURS	$\bar{\hat{p}}$	$\overline{\widehat{\text{LCI}}}$	$\overline{\widehat{\text{UCI}}}$
5	0.91211	0.72212	0.97644
10	0.91480	0.79678	0.96711
15	0.91742	0.81080	0.96644
20	0.92000	0.76477	0.97598

Shown below is a plot of these model averaged estimates for male encounter probability as function of HOURS of observation, along with their associated confidence intervals (extrapolated over a more continuous range from 5 → 20 HOURS).



The calculated 95% CI will not be symmetrical around the parameter estimate – as the value for the covariate HOURS is much greater or lesser than the mean value (≈ 12 hours), the 95% CI gets progressively larger. This is expected as there is less 'information' at either end of the distribution of HOURS on which to base our inference, and thus, more uncertainty in our estimate.

Now, while this particular approach involves a lot of 'manual work' (although some facility with programming can speed things up considerably), the basic procedure can largely be automated to some extent, and executed within program **MARK**. Recall from section 6.8.2 that we mentioned that *if* we treat *environmental* covariates as *individual* covariates, then we can use the individual covariate

plotting (and model averaging) capabilities in **MARK**, to generate the model averaged values, and the confidence limits for these averaged values, as we've done 'by hand' in the preceding. The use of individual covariates in **MARK** is covered in Chapter 11.

6.17. RMark – an alternative approach to linear models

If you've made it this far, then you probably have a pretty good feel for the relationship between the design matrix and linear models in **MARK**. Good! But, by now, you may have already run into a few instances – even with our relatively simple practice examples – where you've made a typo (or several) in entering the appropriate design matrix. Or, in some cases, it may not be clear how to construct the design matrix corresponding to the linear model of interest. While you will get better with practice, it is also probably true that the 'mechanics' of designing and building design matrices in **MARK** is *the* single greatest source of 'frustration'. This is especially true for very large design matrices, which may include a large number of parameters, and complicated ultrastructural relationships within a parameter.

Jeff Laake (NOAA – National Marine Mammal Laboratory) has developed a comprehensive library of functions for the **R** statistical package called **RMark** (naturally), which allow you to build, and analyze, linear models in **MARK** – without ever having to 'get your hands dirty' with design matrices. In effect, what **RMark** does is provide a logically consistent 'natural language' (well, the **R** scripting language is becoming sufficiently familiar that it is relatively 'natural') way of building models – analogous to what you might do in **SAS** (or, in the current context, **M-SURGE**). **RMark** is a very robust, and elegant way to build a large number of complex models, quickly, and relatively easily. There is a fair learning curve (somewhat conditional on how much prior experience you may have with **R**, if any), but once you've mastered the conceptual material presented in this book, it is well worth exploring **RMark** – details and full documentation are provided in Appendix C.

6.18. Summary

We're done...at last! Chapter 6 is a **long** chapter, with many concepts and technical details. However, it is also one of the most important chapters, since it covers one of the most useful tools available with program **MARK** – linear models. It is **very** important that you understand this material, so if you're at all unsure of the material, go through it again. Your efforts will ultimately be rewarded – you'll find that using the linear models approach with **MARK** will enable you to fit a wide variety of complex analytical models, quickly and (relatively) easily.

6.19. References

Box, M. J. (1966) A comparison of several current optimization methods, and the use of transformations in constrained problems. *The Computer Journal*, **9**, 67-77.

Dobson, A. J., and Barnett, A. G. (2008) *An Introduction to Generalized Linear Models* (3rd edition), CRC Press.

Lebreton, J.-D., Burnham, K. P., Clobert, J., and Anderson, D. R. (1992) Modeling survival and testing biological hypotheses using marked animals: a unified approach with case studies. *Ecological Monographs*, **62**, 67-118.

McCullagh, P., and Nelder, J. A. (1989) *Generalized Linear Models* (2nd edition). London. Chapman & Hall.