Multi-state models...

Many of the first chapters in this book focussed on ‘typical’ open population mark-recapture models, where the probability of an individual being seen was defined by 2 parameters: the probability the animal survived and remained in the sample area ($p$), and the probability that the animal was encountered ($q$), conditional on being alive and in the sample area.

In this chapter, we extend this simpler paradigm by (in effect) considering a third parameter – a ‘movement’ parameter ($\psi$). We’ll defer formal definition of this parameter for a moment, since the definition changes somewhat depending on one or more assumptions. However, to foreshadow, let $\psi$ represent the probability of moving between states in which the marked individual may potentially be encountered, conditional on being alive and in that state. The fact that there may be more than a single state (i.e., more than one location, or condition, or state) is what leads to the models we describe in this chapter being referred to generally as multi-state models.

Most of this chapter is a synthesis of the basic ideas behind multi-state models, with particular focus on how to implement them in MARK. The concepts and ideas are derived from seminal work by Neil Arnason, Carl Schwarz, Cavell Brownie, Ken Pollock, Bill Kendall, Jim Hines and Jim Nichols, who have been exploring the mechanics and application of these models. Critical in this early evolution was the advent of software to fit multi-state models, most notably program MS-SURVIV, created by Jim Hines. More recently, the development of programs M-E-SURGE by Rémi Choquet, Roger Pradel and Jean-Dominique Lebreton. Multi-state models have been shown to be an extremely rich class of models, with broad applications to many important questions in evolutionary ecology, population dynamics (especially metapopulation dynamics), and conservation biology and management. At best, we hope to provide you with the essence of multi-state models as implemented in MARK, sufficient to convince you of the importance of more careful study of this class of models.

So what are multi-state models? A simple example will make the basic concepts a bit clearer. Suppose you are conducting a study of some seabird that breeds on any one of three discrete islands far offshore from any large land mass. Let the islands have the less-than-inspired names of ‘Island A’, ‘Island B’, and ‘Island C’. Each individual bird, given that it is alive and breeding, will do so on one of these three islands. You are fortunate enough to find sufficient funding so that you are able to mount capture (or resight) operations on all three islands simultaneously. On each occasion (say, each year at the end of the breeding season), you capture, mark and release unmarked individuals, and record recaptures of previously marked individuals. On each occasion, you record the fact that the marked individual was reencountered, and on which island (i.e., the state).

Let’s consider what factors will define the probability of encounter. In the ‘typical’ mark-recapture context, with one sampling location, the probability of encountering the individual in the sample was defined by the probability that it was alive and in the sampling area ($q$), and the probability of encounter
conditional on being alive and in the sample area (p).

In our ‘island’ example, though, we have more than one sampling state – we have 3 islands (A, B and C). Suppose you are working on island B. You capture an unmarked individual, individually mark and release it. You come back next year, and look for this individual. What determines whether or not you will find it? In effect, a re-reading of the definitions of the parameters for the single-state model provides a clue – the marked individual might be encountered on island B, conditional on (a) it surviving to the next occasion, and (b) it not moving to either of the other two islands. As originally described by Arnason (1972, 1973), and later by Brownie et al. (1993) and Schwarz et al. (1993), the transition probabilities (i.e., making the transition from live to dead, or from one island to another) represent what is known as a first-order Markov process. Such a process is defined as one in which the probability of making a given transition between occasion (i) and (i+1) is dependent only on the state at time (i).

Under this assumption, we can now define the parameters which jointly define the probability of encountering a marked individual in a given state on a given occasion:

\[ q_{i}^{rs} = \text{the probability that an animal alive in state } r \text{ at time } i \text{ is alive and in state } s \text{ at time } i+1 \]

\[ p_{i}^{s} = \text{the probability that a marked animal alive in state } s \text{ at time } i \text{ is recaptured or resighted at time } i. \]

As written, \( q \) reflects the joint probability of both surviving and making a transition. Let’s consider this schematically. In Fig. (10.1), we show the 3 islands, with arrows indicating the possible transitions:

Figure 10.1: Schematic representing typical multi-state model. Here, there are 3 states (A, B and C), with the arrows indicating directional movement between states over a given time interval. The probability of moving, conditional on survival, between state \( i \) and \( j \) is determined by parameter \( q^{ij} \).

Remember, the \( q \) values are the probabilities of both surviving and moving. Now, at this point some of you are probably already leaping ahead to the question ‘...is there any way to separate these two probabilities – survival and movement?’ We’ll come to that in a moment. For now, let’s stick with these 2 parameters (\( q \) and \( p \)), as defined above. What do our capture (or encounter) histories look like, and what are the associated probability statements? In fact, as discussed briefly in Chapter 2 (‘Data
Formatting’), the format of the encounter history for multi-state models is qualitatively identical to the ‘normal’ mark-recapture history – a contiguous series of variables indicating whether or not the marked individual was encountered on a particular occasion. For ‘normal’ mark-recapture, this is typically a contiguous series of ‘1’s and ‘0’s.

For multi-state models, instead of ‘1’s to indicate an encounter, we use variables (letters or numbers*) which reflect the particular state in which the individual was encountered. We continue to use ‘0’s to indicate if the individual wasn’t encountered in any of the states on a particular occasion.

For example:

<table>
<thead>
<tr>
<th>encounter history</th>
<th>interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAB0CC</td>
<td>marked on A at occasion 1, seen again on A at occasion 2, seen on B at occasion 3, not seen on any of the islands on occasion 4, seen on C at occasion 5 and occasion 6...</td>
</tr>
<tr>
<td>BABA00</td>
<td>marked on B at occasion 1, seen on A at occasion 2, returned to B on occasion 3, back to A on occasion 4, and not seen on any island at either occasion 5 or occasion 6</td>
</tr>
<tr>
<td>AAAAA</td>
<td>seen on A on occasion 1, moved to C on occasion 2, then back to A and seen on all subsequent occasions in the study</td>
</tr>
</tbody>
</table>

Of course, as we’ve seen from earlier chapters, each of these encounter histories reflects a particular realization of a probabilistic series of events. It is the relative frequency of each history in the data set which provides the basis for parameter estimation.

Consider a simpler case, with only 2 states: A and B. What does the encounter history ‘AAB’ tell us? In this case, the individual was marked and released on A, seen again on A on the next occasion, and then seen on B on the final occasion. What is the corresponding probability expression? Clearly, the organism survived from occasion 1 to occasion 2, and remained on A. It also survived from occasion 2 to occasion 3, but in the process, moved from A to B. Thus, for the encounter history ‘AAB’, the corresponding probability expression is \( \phi^{AA} p^A \phi^{AB} p^B \) (note that for convenience we do not show the subscripts corresponding to the occasions – normally we would do so. The absence of subscripting normally would indicate that the probabilities do not change through time).

What about something slightly more complex – like ‘A0B’? In this case, the individual was marked in state A on occasion 1, released, not seen in either state A or B on the second occasion, and then seen again on the third occasion in state B. What would the corresponding probability statement look like? In this case, the trick is to realize that there are 2 different ‘probability paths’ by which this encounter history could occur:

<table>
<thead>
<tr>
<th>encounter history</th>
<th>interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi^{AA} (1 - p^A) \phi^{AB} p^B )</td>
<td>survived and stayed in state A, but not seen in A on occasion 2, survived and moved from A to B and seen in B on occasion 3</td>
</tr>
<tr>
<td>( \phi^{AB} (1 - p^B) \phi^{BB} p^B )</td>
<td>survived and moved from A to B during first interval, not seen in B at occasion 2, stayed in state B and seen in B on occasion 3</td>
</tr>
</tbody>
</table>

The trick is to realize that (i) the individual clearly survives from occasion 1 to occasion 3 – it is

---

*We hope it is obvious to you that ‘0’ (zero) is not a valid variable to use to indicate a particular state. We hope the reason why is also sufficiently obvious.
simply ‘missed’ (not encountered) at occasion 2 in either state, and (ii) since we don’t know where the individual was at occasion 2 (i.e., in which state), we must accommodate both possibilities – that it either stayed in state A (where it was originally marked), or that it moved from A to B during the first interval. As such, the expected frequency of individuals with encounter history ‘A0B’ would be:

\[ R_1^A \left[ \varphi_1^{AA} (1 - p_2^A) \varphi_2^{AB} p_3^B + \varphi_1^{AB} (1 - p_2^B) \varphi_2^{BB} p_3^B \right], \]

where \( R_1^A \) is the number marked and released in state A on occasion 1.

### 10.1. Separating survival and movement

Now, while the ability to estimate the combined probability of surviving and moving is useful for some purposes, it is ultimately limiting for others. For example, suppose that the states don’t consist of physical locations (like islands), but breeding states (say, breeder and non-breeder). There is no shortage of literature on whether or not mortality selection operates on individuals as a function of their breeding status (does ‘cost of reproduction’ ring a bell, or two?). In a typical analysis of the cost of reproduction, we might want to know (1) is survival dependent upon breeding state, and (2) given that the individual survives, is breeding state at time (\( i \)) a significant determinant of breeding state at time (\( i+1 \))?

So, can we in fact separate ‘survival’ from ‘movement’? The answer is a qualified ‘yes’ – qualified, because the separation of ‘survival’ and ‘movement’ requires making a particular assumption.

Specifically

\begin{align*}
\text{If we assume that survival from time } i \rightarrow i + 1 \text{ does not depend on state at time } i+1, \\
\text{then we can write:} \\
\varphi_i^{rs} = S_i^r \psi_i^{rs}, \\
\text{where (i) } S_i^r \text{ is the probability of survival from time } i \text{ to } i + 1, \text{ given that the individual is in state } r \text{ at time } i, \text{ and (ii) } \psi_i^{rs} \text{ is the conditional probability that an animal in state } r \text{ at time } i \text{ is in state } s \text{ at time } i+1, \text{ given that the animal is alive at } i+1.
\end{align*}

Read it again – slowly. The basic idea is to ‘separate’ the two events in time – survival, and moving. Think of it this way – the individual is in state A. It survives from (i) to (i+1) with probability \( S^A \) based solely on the fact that it was in state A at time (i). Then, immediately before (i+1), it either moves to another state, or stays, with probability \( \psi^{Ax} \) (where \( x = A, B, \) or C in our example). In other words, first flip the ‘survive or not’ coin, and then conditional on survival, flip the ‘move or not’ coin. If the independence assumption is met (i.e., two separate coins – one for survival, another one for movement), then the ordering here (survive and move, or move and survive) is arbitrary.

Now, if we make these assumptions, then the sum of the joint survival/transition probabilities for a given state is equal to the survival probability for that state. In other words,

\[ \sum_s \varphi_i^{rs} = S_i^r. \]

Consider the following example – assume there are just two states: s and r. Since \( \varphi_i^{rs} = S^r \psi_i^{rs} \), and since \( \varphi_i^{rr} = S^r \psi_i^{rr} \), then \( \sum \varphi_i^{rs} = S^r \psi_i^{rr} + S^r \psi_i^{rs} = S^r ( \psi_i^{rr} + \psi_i^{rs} ) \). Since \( ( \psi_i^{rr} + \psi_i^{rs} ) = 1 \), then \( \sum \varphi_i^{rs} = S^r (1) = S^r \). The same logic is true (obviously) for \( \sum \varphi_i^{sr} = S^s \).
In the following (Fig. 10.2), we re-draw the early multi-state figure (10.1), decomposing $\varphi$ into the survival ($S$) and movement ($\psi$) parameters:

$$
S^{A} \psi^{AA} \\
S^{B} \psi^{BB} \\
S^{A} \psi^{AB} \\
S^{A} \psi^{AC} \\
S^{B} \psi^{BC} \\
S^{C} \psi^{CB} \\
S^{C} \psi^{CC} \\
S^{B} \psi^{BB} \\
S^{C} \psi^{CC}
$$

Figure 10.2: Re-parameterization of Fig. (10.1), where $\varphi^{ij}$ is partitioned as the product of survival ($S$) and movement ($\psi$).

If you’ve followed the earlier chapters on standard mark-recapture approaches, you might be thinking that ‘while this is a neat trick, the parameters are probably not separately identifiable’. In fact, they are, because of the constraint that $\sum \psi^{ij}_{-i} = 1$. In other words, the transition (movement) parameters $\psi^{ij}$ are conditional on survival – and hence, on being present in the study area. The effect of this constraint is that animals that move out of all the states in the study, i.e., move outside the study area, cause the estimates of survival to be biased in the same sense that ‘apparent survival’ is estimated. That is, emigration off (or, out of) all the states in the study results in ‘apparent survival’ being ‘true survival’ times the probability that the animal remains on the study area.

A simple example will make this clearer. Assume that 3 states are sampled: A, B, and C. As noted earlier, the encounter histories must include the information indicating which state an animal was encountered in. For 5 encounter occasions, a history such as ‘BCACC’ could result. That is, the animal was initially captured in state B, captured in state C on the second sampling occasion, captured in state A on the third occasion, captured in state C on the fourth occasion, and then again in state C on the fifth occasion. The cell probability describing this encounter history is

$$
\left[ S^{B} \psi^{BC}_{1} p^{C}_{2} \right] \times \left[ S^{C} \psi^{CA}_{2} p^{A}_{3} \right] \times \left[ S^{A} \psi^{AC}_{3} p^{C}_{4} \right] \times \left[ S^{C} \left( 1 - \psi^{CA}_{4} - \psi^{CB}_{4} \right) p^{C}_{5} \right],
$$

where encounter occasions are separated within the square brackets. Note that for the fourth interval, the probability of remaining in state C is just 1 minus the sum of the probabilities of leaving state C.

This cell probability demonstrates a key assumption of this model: survival is modeled with the survival probability for the state where the animal was captured, and then movement to a new state takes place. That is, as implemented in MARK, all mortality takes place before movement. An animal cannot move to a new state where a different survival probability applies, and then die. If it dies, it must do so on (or, as a function of) the current state. If it lives, then it can move to a new state. This assumption is critical if survival probabilities are different between the states. If survival is constant across states, then the
assumption is not important. Biologically, this assumption may not always be reasonable.

Another assumption for MS models...

Previously, we noted one of the key assumptions which we need to make in order to partition \( \varphi \) into subcomponents \( S \) and \( \psi \). Specifically, that survival from time \( i \) to \( i+1 \) does not depend on state at time \( i+1 \). Obviously, if this assumption is not met, then the estimates of \( S \) and \( \psi \) may be strongly biased.

Although the preceding assumption is well-known (and usually mentioned at least once in most papers using MS approaches), there is another assumption which has not received as much attention:

\begin{quote}
**MS models assume that all individuals make the transitions at the same time (relative to the start or end of the time interval), or if not, that the distribution of the transition times is known.**
\end{quote}

The consequences of violating this assumption are addressed in Joe & Pollock (2002).

---

### 10.2. A worked example: cost of breeding analysis

Consider the following example. You are studying a single cohort of individually marked adult deer, for 8 years (i.e., you mark a sample of deer on the first occasion, and then simply follow them for 7 more years). On each occasion, the breeding status of the deer can be determined without error (all individuals can be assigned either breeder or non-breeder status). You want to examine the possibility that survival is influenced by breeding status.

This example is analogous to an important paper published by Nichols et al. (1994) on estimating breeding proportions and costs of reproduction with capture-recapture data. Normally, we might consider breeding status as a ‘trait’, but clearly this is an annually variable phenotypic trait for most organisms. As such, the classic approach of subdividing the sample along trait-lines and looking for differences among the trait groups will not work here. We need another approach. In fact, the multi-state models are just such an approach – we model the movement between breeding states just as we would model movement among physically discrete states.

To demonstrate the point without the complications of ‘messy real world data’, we’ve simulated a data set for this ‘virtual deer’ population (DEER.INP), using the parameter values tabulated at the top of the next page. There are 500 total individuals to start in the simulated data – 250 in the breeding state, and 250 in the non-breeding state. We assume that both states are ‘observable’ – meaning, that \( p > 0 \) in either state (although the detection probability might differ between the states). The assumption that both states are observable is an important one we consider later in this chapter.

If you look carefully at the parameter values (on the next page), you’ll see we’re creating a data set where it is ‘costly’ to breed – the survival from \( (i) \) to \( (i+1) \) for individuals in the breeding state at \( (i) \) is lower than for non-breeders at \( (i) \). However, the probability of switching states (moving from one state to the opposite state) is higher for non-breeders than for breeders (i.e., individuals aren’t likely to stay non-breeders for very long).
Begin a new project in MARK. For the multi-state analysis of the ‘virtual deer’, we want to select ‘Multi-strata Recaptures only’ (about half-way down the list of different data types). Once you select the multi-strata option, you may have noticed that the option to ‘Enter State Names’ has now become active (lower-right corner of the specification window). This will become important in a minute.

Next, enter a title for the project (say, ‘Analysis of virtual deer’), and then select the file (DEER.INP). Note that the encounter histories in DEER.INP look virtually identical to the histories we used for typical mark-recapture analysis – a contiguous string 8 characters long (i.e., 8 occasions), followed by the frequency of individuals having that particular history. But remember – rather than ‘1’s and ‘0’s, we now have ‘N’s, ‘B’s and ‘0’s. In this case, the ‘N’ value represents individuals in the non-breeding state at a particular occasion, and the ‘B’ values are for breeding individuals. The ‘0’s represent occasions when the individual was not seen. Thus, the history ‘NBNN0B0B’ indicates an individual marked as a non-breeder on the first occasion, seen as a breeder on the second occasion, seen as a non-breeder on occasions 3 to 4, not seen at all on occasion 5, and then seen as a breeder for the final two occasions. The use of ‘N’ and ‘B’ in this example is entirely arbitrary – you can use anything you want to indicate state (numbers 1 → 9, letters). The only condition is that it can be only one character wide (e.g., ‘N’ for non-breeders, not ‘NB’).

Next, tell MARK that DEER.INP has 8 occasions.∗ Finally, we need to tell MARK how the states are coded in the input file. To do this, click on the ‘Enter State Names’ button we referred to earlier (MARK defaults to 2 states, so we don’t need to change anything there).

This will cause MARK to spawn a new window which lets you set the labels (codes) for the different states, and their respective names.

---

∗ Here, time intervals are assumed to be equal, 1.0. However, if you have unequal intervals in a multi-state analysis, you need to be very careful. See section 10.6.
Once you’ve entered the appropriate codes, and state names, click ‘OK’, which will bring you back to the Specification window. Once you’re sure everything in this window is correct, click ‘OK’.

As with our standard mark-recapture analysis in MARK, what you’ll see first is the PIM for the survival parameters for Group 1 (in this example, we have only one group). But, within a group, you might have 2 or more states. If you look at the PIM chart, you should recognize that the PIM reflects a time-dependent structure for survival for individuals in breeding state.

To get a quick sense of the way MARK lays out the parameters for this model, let’s look at the PIM chart (by clicking the ‘PIM Chart’ button in the PIM itself):

Clearly, there are 6 parameters involved for each state. Right away this should tell you something. Six parameters means that MARK is making the assumption that survival is dependent only on the state at occasion \((i)\), and is not influenced by the state entered at occasion \((i+1)\). In other words, we’re using the identity we introduced earlier.

\[
\varphi_{i}^{rs} = S'_{i} \psi_{i}^{rs}.
\]

Recall that this identity is true only under this stated assumption. The fact that MARK defaults to...
this assumption becomes important later on. Thus, each of the ‘blue boxes’ in the PIM chart refers to (respectively, going from the lower-left to the upper-right) $S^B, S^N, p^B, p^N, \psi^{BN}$ and $\psi^{NB}$. Examination of the horizontal axis of the PIM chart shows that the current model has time-dependence for each parameter, and that there are 42 total parameters. Even though we know that for these simulated data the parameters are constant through time (no time-dependence), let’s pretend we’re approaching these data naïvely, and go ahead and run this model (call it ‘$S(g.t)p(g.t)\psi(g.t)$’, where the ‘g’ refers to group – or state, breeder or non-breeder in this case).

The first thing you might notice, especially if you’re using a computer of ‘average’ processing power, is how much longer this model takes to run than analysis of typical mark-recapture data. The reason is fairly straightforward – the more parameters, the longer it takes to reach the solution (although the increase in time taken does not scale as a simple linear function of the number of parameters). We have 3 parameters ($S, p$ and $\psi$), so it takes longer than models with only 2 (say, $\varphi$ and $p$). Once the estimation is complete, add the results to the browser.

Before we look at the results of this analysis, let’s run another model – ‘$S(g)p(g)\psi(g)$’ – constancy for all parameters, but allowing for possible differences among groups (breeding states). Obviously, the first thing we need to do is modify the parameter structure. As you may have gathered from earlier chapters, this is most easily done using the PIM chart. Recall from earlier chapters that we could modify the parameter structure from within the PIM chart (at least for certain models) by simply right clicking on each of the ‘blue boxes’ on the PIM chart. In this case, you could move the cursor over each of the ‘blue boxes’ and right-click with the mouse. This causes a menu to pop up which lets you select among various parameter structures. One of the options is ‘Constant’. By selecting the ‘Constant’ option for each blue box in turn, we could build our model. Then, to eliminate the ‘gaps’ between the ‘blue boxes’ (i.e., to make the parameter index values contiguous), you could either drag each box manually, or right-click anywhere in the PIM chart, and select ‘Renumber no overlap’. This will cause the PIM chart to reformat without any gaps between any of the blue boxes.

While this works, in this case, for this particular model (where the structure is the same for all 6 blue boxes), there is a much faster way – simply select the ‘Initial | All | Constant’ menu option from within the PIM chart. If you do this, your PIM chart will be quickly reformatted to the model we’re after, which looks like the following:
Note that, superficially, this looks identical to the PIM chart we saw for the fully time-dependent model discussed earlier, but if you look carefully at the horizontal axis, you’ll see it now has many fewer parameters – 6 (instead of 42). Go ahead and run this model, and add the results to the browser.

Inspecting the results browser shows us immediately that the model with constant parameter values is a much more parsimonious model of these data than is the fully time-dependent model. Before we go much further, let’s have a look at the real parameter estimates for the constant model – \( S(g)p(g)\psi(g) \) – more formally, model \( \{S^g, p^g, \psi^g\} \).

We can see that the estimates are quite close to the parameters used to simulate the data set. Since the constant model is clearly much better supported than the fully-time-dependent model, let’s delete the time-dependent results from the browser (by highlighting the time-dependent model and then clicking on the trash can icon in the toolbar of the browser window). We will use the constant model \( \{S^g, p^g, \psi^g\} \) as the general model in our candidate model set. We’re interested in examining two questions: (1) is there a difference in survival among breeders and non-breeders, and (2) does the probability of transition between breeding states depend on breeding state at occasion \( i \)?

Let the following represent the candidate model set:

\[
\{S^g, p^g, \psi^g\}, \{S, p^g, \psi^g\}, \{S^g, p^g, \psi\}, \{S, p^g, \psi\}
\]

In other words, (1) constant over time, but with group \( g \); breeding state) differences for all parameters, (2) equal survival between breeding states, but differences in recapture and movement, (3) differences in survival and recapture, but no differences in transitions probabilities between breeding states, and (4) differences in recapture probability among breeding states only.

At this point, you should be able to run the 3 new models fairly easily – it should take you only a few seconds to construct each model using the PIM chart approach. The results for all 4 models in the candidate model set are shown at the top of the next page. We see that the model with differences in both survival and movement rates between breeders and non-breeders is 10-times better supported
by the data than the next best model, where survival is the same between breeding states, and both recapture and movement probability differ \(0.915/0.085 = 10.8\).

Let’s re-examine the estimates from our best model, \(\{S, p, \psi\} \). While survival is clearly estimated for each state separately, the question is, which ‘movement’ parameter gets estimated? For example, among individuals in breeding state (B) at time \((i)\), they can either move to the other state (with probability \(\psi^{BN}\)), or stay in the breeding state (\(\psi^{BB}\)). Which one do we estimate?

Well, at this point, we take advantage of the logical necessity that the sum of \(\psi^{BN}\) and \(\psi^{BB}\) must equal 1.0 (i.e., an animal in a given state, must either remain in that state or move to another state, conditional on remaining alive). As such, one of the movement parameters is redundant (if you know the value of one, you know the other as 1 minus the first one). As such, any one of the movement parameters for a given state could be omitted. For example, our estimate for \(\psi^{NB} = 0.8274\). Thus, \(\psi^{NN} = 1 - 0.8274 = 0.1726\), which is fairly close to the parameter used in the simulation (0.2).

Fortunately, MARK gives you some flexibility as to what movement parameters are estimated – the default is to estimate the probability of moving from one state to another (i.e., \(\psi^{ij}\); probability of moving from \(i\) to \(j\)). However, you might instead want to estimate the probability of remaining within a state (\(\psi^{ii}\); probability of moving from \(i\) to \(i\)). You can ‘tell’ MARK to estimate \(\psi^{ii}\) simply by changing the definition of the PIM. Returning back to our previous deer example, we can simply retrieve a model from the browser, and then select ‘Change PIM definition’ from the PIM menu, and run it (or you can change the PIM structure before running the model).

Go ahead and try it – doing so will bring up a window showing you the non-default transitions that are available – in this case, there is only one other possibility (i.e., the non-default \(\psi^{ii}\)).

Now, you need to be a bit careful here – read the text at the top of this window carefully. It’s asking you to tell MARK which of the transitions you want to estimate by subtraction. For example, MARK defaults to estimating the transition \(\psi^{ij}\). So, you’d normally (by default) have to derive \(\psi^{ij}\) by subtraction. So, if you want MARK to estimate \(\psi^{ii}\), you need to tell it you want to estimate \(\psi^{ii}\) by subtraction. So, in our example, with two states (N and B), MARK defaults to estimating \(\psi^{NB}\) and \(\psi^{BN}\). So, if you want MARK to estimate (for example) \(\psi^{BB}\), you need to tell MARK you want to estimate \(\psi^{BB}\) by subtraction, as shown at the top of the next page:
If you run this model, you’ll see that it gives you the estimate of $\psi^{BB}$ you want. And, changing the specification of which transitions are estimated does not (and should not) change anything else about fitting the model – the model deviance, and AIC value, should be unchanged.

There are several potential advantages to being able to specify which transition parameters are estimated. First, the optimization routine in MARK is known to work better if the parameters are not close to the boundaries (i.e., not close to 0.0 or 1.0). Second, you are likely not to want estimates of the estimated parameters to be ‘too big’, because if their sum is > 1, then the remaining probability is estimated as < 0. The idea, then, is to pick transition probabilities that are likely to be small, giving you the best chance that the remainder transition probability will be > 0 (although as we will see, we can circumvent this particular problem by specifying a different link function – the multinomial logit link). Third, being able to specify which movement parameter you want to use gives you the ability to build specific constrained models. For example, suppose you are working with a 3-island system, and wanted to assess whether the probability of returning to a given island (i.e., philopatry) was equal for the various islands, but did not want to assume that the probabilities of movement to other islands was also equal. You could do this by constraining the probability of remaining on a given island. Note that for a 2-state model, setting the probabilities of leaving for the other state equal is also setting the probability of remaining equal. With > 2 states, this is not true.

However, it is important to remember that you can change the PIM definition only in terms of the ‘recipient’ state. The ‘donor’ state at the time of the transition is fixed, whereas the ‘recipient’ state is dynamic, since it is the outcome of a probabilistic process determined by the parameter $\psi$. You have one $\psi$ parameter for movements from each state, not movements to a given state. So, for our 2 state ‘breeder’ (B) or ‘non-breeder’ (N) example, we could change the PIM definitions to (say) $\psi^{NB}$ and $\psi^{BB}$, or $\psi^{BN}$ and $\psi^{NN}$. In either case, we’re estimating the probabilities of moving into a common ‘recipient state’, each as a function of the different ‘donor’ states. Note that the sum of $\psi^{NB} + \psi^{BB}$ (for example) does not necessarily equal 1. In contrast, $\psi^{BN} + \psi^{BB} = 1$. As noted earlier, since the movement is conditional on survival, then the sum of all movement probabilities from a given state must equal 1 (if you’re alive at the end of the interval, you must be in one of the available states, with probability 1).

Why is this distinction important? It is important because being aware of it gives you some additional flexibility to test various hypotheses. For example, suppose for our ‘cost of breeding’ analysis we wanted to test a model where the probability of breeding next time step is potentially a function of whether or not you breed this time step. In other words, you might be interested in constructing a null model where $\psi^{BN} = \psi^{BB}$. In other words, a model where the probability of breeding next year is random with respect to breeding state this year. The problem is, the two parameters are constructed based on a common ‘donor’ state (breeder, B), which is not possible by simply changing the PIM definition. But, if you remember that $\psi^{BN} + \psi^{BB} = 1$, then if $\psi^{BN} = \psi^{BB}$, then $\psi^{BN} = \psi^{BB} = 0.5$. Which of course is the expected probability for either transition of the movement is strictly random. So, you simply need to build a model where you first fix the estimate of either $\psi^{BN}$ or $\psi^{BB}$ (whichever one you’ve defined in your PIM) to be 0.5. Alternatively, you might interested in whether or not breeding next year is a function of not breeding this year You would exactly the same logic – you build a model where you first fix the estimate of either $\psi^{NN}$ or $\psi^{NB}$ to be 0.5. As noted above, for a 2-state model, setting the
probabilities of leaving for the other state equal is also setting the probability of remaining equal. Also as noted, this is not true with > 2 states, at which point, things get somewhat more complicated.

What about \( \psi^S \)? Recall that as originally described, the multi-state models focussed on estimation of the ‘combined’ probability of survival and movement. **MARK** assumes that survival is dependent only on state at time \((i)\) – this allows **MARK** to separate \( \psi \) into its component parts \( S \) and \( \psi \). Can we estimate \( \psi \) using **MARK**? Yes, but only by hand. Consider the results of our analysis so far. \( \hat{\psi}^{BN} \) is estimated as 0.4263. \( S^B \) is estimated as 0.6997. Thus, \( \hat{\psi}^{BN} \) is estimated as \( \hat{\psi}^{BN} = S^B \hat{\psi}^{BN} = 0.2983 \). Given the standard errors for both \( S^B \) and \( \psi^{BN} \) from **MARK**, it is possible to derive standard errors for \( \hat{\psi}^{BN} \) using the Delta method, or by using the sample chains from an MCMC analysis (Appendix B).

Are there further limitations imposed by **MARK**? In fact, there may be one more, stemming from the underlying ‘assumption’ **MARK** defaults to – the assumption that survival depends only on state at time \((i)\). While this is perhaps reasonable in many ‘real world’ situations, what if it isn’t? Nichols and colleagues have extensively explored models where the transition probabilities depend on state both at time \((i)\) and \((i-1)\). While the recapture parameters remain the same, they introduced a new transition parameter:

\[
\rho_{i-1,i} = \text{the probability that an animal alive in state } r \text{ at time } i-1 \text{ and state } s \text{ at time } i \text{ is in state } t \text{ at time } i+1.
\]

They referred to this as a ‘memory model’ (technically, it is a second order Markov model), suggesting that the ‘history’ of events experienced by the marked individual leading up to its state at time \((i)\) might influence the transition probability between \((i)\) and \((i+1)\). These ‘memory’ models were coded into MS-SURVIV*, and allow for testing hypotheses that the transition \( \rho \) is first-order Markovian (i.e., dependent only on state at time \((i)\) versus those in which the transition \( \rho \) is dependent on the state at both \((i)\) and \((i-1)\) (i.e., second-order Markovian).

At present, **MARK** is unable to handle ‘memory’ models, in part since they clearly violate the assumption **MARK** makes that survival is dependent only on state at time \((i)\). These models may prove increasingly important tools to explore questions concerning life-history decisions over the lifetime of the organism. Much theory exists suggesting that the optimal decisions at age \( x \) (e.g., breed or not, emigrate or not) are likely to reflect the sequence of decisions experienced (or made) at age \( < x \). However, such memory models are extremely ‘data hungry’, and much work remains to be done to develop extensions of such models to relevant biological questions.

But while this is a limitation of **MARK** when compared to MS-SURVIV, for Markovian models, **MARK** adds significant flexibility for many models, particularly through use of the design matrix. In the next section, we shall explore examples showing how we can use the design matrix to constrain the estimates of survival and movement.

### 10.3. States as ‘groups’ – multi-state models and the DM

In the preceding, we fit a series of ‘dot’ models to the data – there was no need to build a design matrix (DM) for the models in our candidate model set. But, it is important to understand how the DM is built for multi-state models. It is really not much different from what you’ve already seen – all you need to do is remember that states are, in effect, treated like groups. But, with a catch you need to be aware of.

Consider the deer data we just analyzed, and consider fitting a fully time-dependent model, for all parameters \((S, p \text{ and } \psi)\). If we set the PIM structure to \(\{S^1_p^1\psi^1\} \), and then pull up the design matrix using ‘Design matrix | Full’, this is what we see (if we ‘zoom in’ in on the part of the DM coding for

---

* MS-SURVIV is developed and maintained by Jim Hines, USGS-Patuxent Wildlife Research Center.
If you look closely, you’ll see that columns 1 \(\rightarrow\) 7 correspond to survival for breeding individuals, while columns 8 \(\rightarrow\) 14 correspond to survival for non-breeding individuals. What is important to note here is that each parameter has a separate intercept. In other words, \textit{MARK} is treating the same parameter \((S)\) for different levels of the state (breeding, non-breeding) as if in fact they were separate parameters.

While there is nothing wrong with this in terms of the reconstituted parameter values, it does limit the models you can build. For example, you would not be able to construct an additive model in any obvious way, since there are no \textit{explicit} interaction columns. In fact, the interaction is \textit{implicit} in the fact that the two parameters do not share a common intercept. So, in order to have more flexibility, we generally choose to re-code the DM such that parameters share a common intercept across levels of the state variable. To demonstrate this, we first go ahead and run this model (using the default ‘fully time-dependent’ DM with a separate intercept for each parameter/state combination. The reported model deviance is 680.054.

Now, how would we modify the DM with a common intercept? Simple:

Here, we have a column for the common intercept, a single column for state (since there are 2 states, we need only a single column), and then 6 columns for time, and 6 additional columns for \textit{state.time} interactions (for a total of 14 columns, identical to the number of columns in the original DM, and equal variation survival, \(S\)).
to the number of parameters specified in the PIMs for survival, $S$). If you run this modified DM (shown below), the resulting deviance is 680.054, identical to the value reported for the original DM.

Here is another DM example, again using the deer data. Suppose we decided to fit model $\{S_i^b p_i^b \psi_i^b\}$ – in other words, time varying survival as a function of breeding state ($g$), with constant encounter and movement probabilities which differ between breeding states. First, build the model using PIMs – here is the PIM chart corresponding to our model:

![PIM Chart](image)

Run this model, and add the results to the browser. Now, let’s build this same model using a design matrix approach. Based on the PIM chart, we see that we have 18 structural parameters in the model. We select ‘Design | Reduced’, and specify 18 covariate columns in the design matrix. We will follow the convention introduced above, and treat ‘state’ as a grouping or classification factor. In other words, we’ll use a common intercept for modeling differences in survival among states. In this example, we have 2 levels of ‘state’, so we need one column to code for it.

Here is the design matrix corresponding to our model:

![Design Matrix](image)
Again, this structure is identical to what you have seen before for a single classification factor with 2 levels of the factor. Run this model (label it with DM), and add the results to the browser:

<table>
<thead>
<tr>
<th>Model</th>
<th>AICc</th>
<th>Delta AICc</th>
<th>AICc Weight</th>
<th>Model Likelihood</th>
<th>No. Par.</th>
<th>Deviance</th>
<th>-2Log(L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S</td>
<td>g</td>
<td>p</td>
<td>o</td>
<td>n</td>
<td>g) - DM)</td>
<td>3819.0683</td>
<td>0.0000</td>
</tr>
<tr>
<td>(S</td>
<td>g</td>
<td>p</td>
<td>o</td>
<td>n</td>
<td>g) - PIM)</td>
<td>3821.1249</td>
<td>2.0566</td>
</tr>
</tbody>
</table>

Note that the models have different AIC values. Is our design matrix wrong? No! Look at the deviances. Note that they are exactly the same. If the deviances are the same, then the models are the same.

So, why the difference in the AIC values? Remember, the AIC is calculated as a function of the fit (deviance), and the number of parameters. If the AIC values differ, but the fit is the same (i.e., same model deviances), then it is the number of estimated parameters which differ. You see in the browser that this is indeed the case – the model built with the PIM chart (which defaults to using the sin link function) shows 18 estimated parameters, whereas the model we just built using the DM (which defaults to the logit link function) reports only 17 parameters. This explains why the two models have different AIC values. You can confirm this was indeed the problem by re-running the model you built with PIMs, but first specifying the logit link, instead of the default sin link. If you do so, you’ll see that the PIM model run using the logit link reports 17 parameters. Meaning, the problem (difference) is due to the link function, not ‘errors’ in your DM.

What about time-varying movement parameters, $\psi_i$? Let’s consider model $\{S^E, p^S, \psi^S\}$ – time variation in survival and movement, but constant encounter probability.

Here is the PIM chart corresponding to this model.

What is the structure of the design matrix corresponding this parameter structure?
We covered the construction of the DM for the survival and encounter parameters for a multi-state design earlier. The DM for these 2 parameters should look like:

For the movement parameters, $\psi$, the structure is essentially the same as what we used for the survival parameter:

Note that it is not necessary to use a common intercept, but it can be convenient to do so for models where you may want to create a structural relationship between the parameters.

### 10.3.1. A simple metapopulation model – size, distance & quality

In this example, we go back to the hypothetical model we considered right at the beginning – 3 islands with colonies of a particular species of sea-bird, with the potential for exchange among some or all of the islands. For this example, we’ll add some complexity to the model, by introducing a number of factors which might potentially influence any of the 3 parameters, either individually or together –
factors which are fairly representative of ‘real-world’ data.\footnote{In fact, our example is qualitatively quite similar to a classical study of a metapopulation of roseate terns (Spendelow et al. 1995 \textit{Ecology} 76: 2415-2428).}

In our example, we’ll vary 2 main factors: (1) the size of individual islands, and (2) the spacing (distance) among the islands. Here is a graphical representation of our ‘island system’:

We see that the islands are clearly not equally spaced: as drawn, island B is closer to island A than it is to island C. And, the islands are not the same size: island A is the largest, followed by island C, with island B the smallest. The islands might differ in terms of some characteristic (say, some limiting resource). Sometimes this might scale with the size of the island. For our example, we’ll simplify somewhat: we’ll assume that island A has the highest ‘quality’, while island B and island C have equivalent quality. As indicated, all transitions among islands are possible. The question we have then is – are there differences in survival, movement probability or recapture probability among the 3 islands? Further, might any differences correlate with differences in spacing, size or quality of the islands?

Based on this ‘metapopulation structure’, we simulated a 6 occasion study, with capture, mark and release occurring simultaneously on all three islands in all years. In other words, in this example, we’re not simply following a single marked cohort through time, but are releasing recaptures and newly marked birds on each occasion. We simulated 250 newly marked individuals on each island at each occasion. The encounter data are contained in the file ISLAND.INP.

Rather than tell you \textit{a priori} what the parameter values were in the simulation, let’s see how well we can do by building a candidate model set, and using Akaike weights to select the best model. Based on our description of the system, we have good reason to expect that island quality might influence survival. Further, distance among islands, and differences in island size, might influence movement probabilities.

Of course, we might also hypothesize that island size could influence both survival and movement if we invoked density-dependent effects (which will tend to lead to departures from the ideal free distribution based on simple differences in quality). For now, let’s say that, based on earlier studies of this species, we have no evidence for density-dependent effects.

Next, assume there is a fixed number of investigators on each island capturing and releasing the birds. Assume also that, all other things being equal, colony size is proportional to the size of the island. Thus, since island A is the largest, you might anticipate that for a constant level of capture effort, that recapture probability should be smaller on island A than on (say) island B, which is the smallest of the three islands.

Finally, we assume that environmental conditions during the 6 years of the study have been near-constant.
Based on these *a priori* hypotheses, we construct our candidate model set (shown below). Given the detail of our background knowledge, and our insight about these birds, we start with a general model which allows for ‘group’ differences (i.e., differences among islands), but one that is constant over time. We include several plausible reduced parameter models starting from this general model.

<table>
<thead>
<tr>
<th>model</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S^g p^g \psi^g$</td>
<td>general model – all groups (states) different – constant over time</td>
</tr>
<tr>
<td>$S p^g \psi^g$</td>
<td>constant survival – group difference in recapture and movement</td>
</tr>
<tr>
<td>$S^g p^g \psi^g$</td>
<td>constant recapture – group differences in survival and movement</td>
</tr>
<tr>
<td>$S^g p^g \psi^g$</td>
<td>constant movement (inter-island all equal) – group differences in survival and recapture</td>
</tr>
<tr>
<td>$S^{quality} p^g \psi^g$</td>
<td>survival constrained to be a function of island quality – group differences in recapture and movement</td>
</tr>
<tr>
<td>$S^g p^{size} \psi^g$</td>
<td>recapture constrained as a function of island size – group differences in survival and movement</td>
</tr>
<tr>
<td>$S^g p^{distance} \psi^g$</td>
<td>inter-island movement a function of inter-island distance – group differences in survival and recapture</td>
</tr>
<tr>
<td>$S p^{size} \psi^{distance}$</td>
<td>recapture a function of island size, movement a function of inter-island distance – constant survival</td>
</tr>
<tr>
<td>$S^{quality} p^{size} \psi^{distance}$</td>
<td>survival a function of island quality, recapture a function if island size, and inter-island movement a function of inter-island distance</td>
</tr>
</tbody>
</table>

Open up the PIM chart, and change the default (time-dependent) parameter structure to one that has group (i.e., island) differences, but that is constant over time for each parameter, as shown below:
Go ahead and run the model, and add the results to the results browser. The AICc for this model is 19703.54, with 12 estimated parameters. Since this is our general model, let’s have a preliminary look at the parameter values. To make it easier to relate the estimates to the model, we’ll add the estimates to our ‘model diagram’, shown below:

Clearly, there is some heterogeneity among islands for recapture and movement probability, but not survival. The question is, are the apparent differences in recapture or movement significant, and does the pattern of variation correlate with one or more of the covariates in our model(s)? Since the 2nd through 4th models in the model set (preceding page) are straightforward (hopefully!), we’ll skip the mechanics of setting them up and running them, and go ahead and consider the results:

The best model \( \{ S^S, p^S, \psi^S \} \) is not appreciably better supported by the data than is the next best model \( \{ S^S, p^S, \psi^S \} \). As such, we could only say that these two models are probably equally likely. So, our tentative conclusion at this point is that there is some evidence of equivalence (in some senses) of survival among islands. The movement probabilities clearly differ. This would seem to be concordant with our casual inspection of the estimates from the most general model (shown in the schematic diagram, above).

What about GOF (goodness of fit)? Normally, at this stage, we’d be thinking about fit of our general model – in part for the purposes of deriving an estimate of \( \hat{\xi} \). We discuss GOF testing for multi-state models at the end of this chapter – for this example, we’ll assume the general model fits the data, and leave \( \hat{\xi} \) at the default value of 1.0.

Can we improve our understanding by constraining the general model to be a function of one or more of the ‘potentially relevant’ covariates? At this point we need to modify the design matrix to constrain the general model. Click once on model \( \{ S^S, p^S, \psi^S \} \) in the browser – to make it the active model. Then, right-click this model, and select the ‘Retrieve’ option. Since we want to constrain model \( \{ S^S, p^S, \psi^S \} \), we want the design matrix to initially reflect its parameter structure.

Before we actually look at the design matrix, what would the linear model look like for the model you just retrieved? Remember, the retrieved model was \( \{ S^S, p^S, \psi^S \} \). Since ‘group’ = ‘island’, then there are 3 levels of group (i.e., 3 islands). Thus, we need \((3 - 1) = 2\) columns, plus a column for the intercept, to code for the various group effects for survival and recapture. The structure of the design matrix for the \( S \) and \( p \) parameters is (hopefully) straightforward – a column for the intercept, followed by 2 columns...
of dummy variables, ‘1 0’ for island A, ‘0 1’ for island B, and ‘0 0’ for island C. For this model, we’ll use the same basic linear structure for both parameters (S and p). Remember, these codings are arbitrary – we could have just as easily used ‘1 0’ for A, ‘1 1’ for B, and ‘0 1’ for C. The important point is that what is required is 2 columns for the coding, and that the coding is based on 0 or 1 dummy variables.

What about the ψ (movement) parameters? A little trickier, since there are several equivalent coding schemes which would accomplish the same thing. Note that there are 3 groups of movement parameters – those involving movements from island A, those involving movements from island B, and those involving movements from island C – 2 parameters for each group. Thus, following the standard linear models paradigm, we could also use 1 intercept column, and 1 column to indicate which of the 2 transitions within each of the 3 movement groups. For example, as shown below, for the parameters ψAB and ψAC, we could have a column of intercept, followed by a column with a ‘0’ indicating movement from A to B, and a ‘1’ indicating the A to C movement.

The design matrix for all 3 parameters is shown below:

<table>
<thead>
<tr>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>Form</th>
<th>B7</th>
<th>B8</th>
<th>B9</th>
<th>B10</th>
<th>B11</th>
<th>B12</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 SA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Try running this design matrix – the estimates are identical to the estimates for the model you created initially simply by modifying the PIMs.

Now that we have built our general model using the design matrix, let’s build the next model in the set, model \( \{S_{quality} \psi AB \psi AC\} \). For model \( \{S_{quality} \psi AB \psi AC\} \), we want to constrain survival to be a function of island ‘quality’. Recall that in this example, island A is believed to be of better quality than island B or C, but that island B and C are believed to be of equal quality. Thus, we need a single column for the intercept, and a single column coding for quality. We’ll let ‘1’ represent ‘good quality’, and ‘0’ represent ‘poor quality’.

Thus, the design matrix corresponding to the survival parameters would look like:

Go ahead and run this model, and add the results to the browser. Before we examine the results
of this model, let’s go ahead and fit the next 2 models in the list – model \(\{S, p, \psi\}\) and model \(\{S, p, \psi_{distance}\}\).

Since we need some ‘numbers’ to represent island size and inter-island distance, we used the following values for each, respectively:

<table>
<thead>
<tr>
<th>island size</th>
<th>island A</th>
<th>island B</th>
<th>island C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 10</td>
<td>A 0</td>
<td>A 5</td>
<td>A 12</td>
</tr>
<tr>
<td>B 3</td>
<td>B 0</td>
<td>B 7</td>
<td>B 0</td>
</tr>
<tr>
<td>C 6</td>
<td>C 0</td>
<td>C 0</td>
<td>C 0</td>
</tr>
</tbody>
</table>

Since models \(\{S, p, \psi\}\) and \(\{S, p, \psi_{distance}\}\) are both structurally similar to model \(\{S_{\text{quality}}, p, \psi\}\), you should be able to quickly see how to modify the design matrix (basically, as we just did, but for different parameters).

For example, consider model \(\{S, p, \psi\}\), where we want to constrain the probability of recapture to be a function of the size of the island (where it might be reasonable to assume that the bigger the island, the lower the recapture probability for a given marked individual, all other things being equal). To fit this model, you simply need to modify the part of the design matrix corresponding to the recapture probabilities.

Given the ‘island size data’ in the preceding table, here is the design matrix for model \(\{S, p, \psi\}\) (only that part of the design matrix corresponding to the survival and recapture parameters is shown).

Pretty straightforward. Potentially the only tricky one is model \(\{S, p, \psi_{distance}\}\). Again, the key is to look closely at the coding for the movement parameters, \(\psi\). If you’re constraining \(\psi\) to be a function of the distance, then you would replace the ‘1’ and ‘0’ dummy variables in the design matrix with the actual distance values themselves (remember: the ‘0’ and ‘1’ coding treated islands as levels of a classification factor, while using the distances directly is considering them as linear covariates). Sounds reasonable.

However, this is a good example of a problem that is in fact a bit trickier than it might seem at first. For example, you need to decide if you want the probability of moving from A to C (\(\psi_{AC}\)) to be the same as the probability of moving from C to A (\(\psi_{CA}\)), since clearly the distance is the same between the same two islands, regardless of the direction you’re moving. Is this a reasonable constraint?

Let’s assume we want to allow the movement probabilities to differ, even among ‘complementary’ transitions (i.e., we’ll let \(\psi_{AC}\) differ from \(\psi_{CA}\)). How would we set that up? Well, the most flexible way would be to categorize each of the movement transitions according to the donor island. For example, treating movements from island A as one group, from island B as one group, and so on. Since there
are 3 island groups, then 1 intercept column, and 2 columns of dummy variables to code for island. Then, a single covariate column coding for the linear distance among islands. Finally, 2 columns for the interaction of ‘island group’ with linear distance.

The relevant portion of the design matrix we need for this model is shown below. The values of the covariates are the inter-island distance values listed in the table at the top of this page:

Now, it is important here to understand what we’ve done in the design matrix. The intercept is in the first column, the dummy variables for ‘island grouping’ are in columns two and three, and the linear covariate (distances among islands) is in column four. The interaction of ‘island’ and ‘distance’ is shown in columns five and six. Remember, the interaction means that the estimate of movement rate varies as a unique function of island and distance among islands. Go ahead and run the model corresponding to this design matrix, and add the results to the browser. See if you can build all the candidate models listed earlier at the start of this example.

begin sidebar

The multinomial logit link and MS models

Logically, the transitions from a given state must logically sum to 1.0. However, for reasons related to how MARK numerically calculates the estimates of the various parameters, this logical constraint isn’t always met – in other words, the sum is occasionally > 1, which is clearly not logically feasible. This tends to happen (if it happens at all) if some of the transitions are close to the [0, 1] boundaries.

One solution is to change the link function MARK uses – from the sin or logit link, to what is known as the multinomial logit link function (MLogit). We will introduce the MLogit link here with respect to multi-state models, but it is also frequently used in POPAN (J-S) models (Chapter 12), and open robust design models (Chapter 15).

The multinomial logit works as follows. Assume that each of the transition parameters from state A have their own $\beta$ value, so that A to B is $\beta_1$, A to C is $\beta_2$, and A to D is $\beta_3$. To constrain these 3 parameters to sum to $\leq 1$, the multinomial logit link works as follows:

$$
\psi^{AB} = \frac{e^{\beta_1}}{1 + e^{\beta_1} + e^{\beta_2} + e^{\beta_3}}
$$

$$
\psi^{AC} = \frac{e^{\beta_2}}{1 + e^{\beta_1} + e^{\beta_2} + e^{\beta_3}}
$$

$$
\psi^{AD} = \frac{e^{\beta_3}}{1 + e^{\beta_1} + e^{\beta_2} + e^{\beta_3}}
$$

To create this set of links, you need to tell MARK to use a Mlogit link. You do this by first selecting the ‘Parm-Specific’ link function from list of link options on the ‘Run’ window:

Chapter 10. Multi-state models...
When you hit the ‘OK to run’ button, you’re presented with a second window, which allows you to specify the link function for each parameter in your model (this window was first described in Chapter 6 when we introduced the cumulative logit link).

For example, in the deer example, for model \( \{ S^b \psi^b \psi^s \} \) you have 6 parameters, so the relevant part of the window looks like:

![Image of window with link function selections]

Here, we’ve selected \( \text{MLogit}(1) \) for ‘\( \Psi \text{ B to N} \)’, and \( \text{MLogit}(2) \) for ‘\( \Psi \text{ N to B} \)’. The number inside the parentheses is a simple indexing for state (2 states – index 1 and index 2). So, if we had 3 states (A, B and C), we’d need 3 levels of indexing. Thus, for example, if we use \( \text{MLogit}(1) \) for all of the A state transitions, we might use \( \text{MLogit}(2) \) for state B transitions, and \( \text{MLogit}(3) \) for state C transitions. For each set of parameters where you want the constraint that the parameters sum to \( \leq 1 \), you must specify a \( \text{MLogit}(x) \) function, where \( x \) represents the set number of the MLogit link function.

Here is an example involving > 2 states – based on the example input file MSSURV.INP, with 4 occasions and 3 states (A, B and C). For a model with full time-dependence in all parameters (including the movement parameters \( \psi^{ij} \)), we would use the following MLogit specification:

![Image of window with MLogit specifications]

It is important to understand the ‘pattern’. For 4 occasions, there are 3 intervals. So, there would be 3 parameters for \( \psi^{AB} \) (i.e., \( \psi^{AB}_1, \psi^{AB}_2, \text{ and } \psi^{AB}_3 \)). Same for \( \psi^{AC}, \psi^{BA}, \text{ and so on. } \) Consider state A for occasion 1. Given 3 states, then there are 3 possible transitions: \( A \rightarrow A, A \rightarrow B, A \rightarrow C \). Since the default in \text{MARK} is to estimate the probability of remaining in a state by subtraction (i.e., \( \hat{\psi}^{AA} = 1 - \hat{\psi}^{AB} - \hat{\psi}^{AC} \)), then there are only 2 parameters for individuals in state A at occasion 1. We apply the same MLogit link (\( \text{MLogit}(1) \)) to those two parameters. For occasion 2 for state A, we use \( \text{MLogit}(2) \), and so on. The key is (1) remembering that the index \( x \) in \( \text{MLogit}(x) \) refers to a set of parameters that you want to constrain, and (2) for time-dependent models, you need to keep track of which sets of parameters correspond to which sampling intervals.

Still not clear? Here is a final demonstration, unrelated to MS models, but using the familiar male Dipper data. Recall that there are 7 sample occasions for the Dipper data – so 6 intervals. Suppose for some reason you wanted the sum of the estimates \( \psi_1 + \psi_2 = 1, \psi_3 + \psi_4 = 1 \), and \( \psi_5 + \psi_6 = 1 \).
All you need to do to enforce these constraints using the MLogit is start with a model where apparent survival ($\phi$) is time-dependent, then specify the 'Parm-Specific' option from the 'Setup Numerical Estimation Run' window. Since there are 3 sets of parameters we want to constraint (i.e., $\phi_1$ and $\phi_2$, $\phi_3$ and $\phi_4$, and $\phi_5$ and $\phi_6$), then we use indexing 1 $\rightarrow$ 3 when we specify the MLogit link for each successive pair of parameters (MLogit(1) for $\phi_1$ and $\phi_2$, MLogit(2) for $\phi_3$ and $\phi_4$, and MLogit(3) for $\phi_5$ and $\phi_6$):

![Specify Link Values](image1)

Looking at the reconstituted estimates on the real probability scale:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:Ph1</td>
<td>0.5949028</td>
<td>0.0872560</td>
<td>0.4193647 - 0.7494399</td>
</tr>
<tr>
<td>2:Ph1</td>
<td>0.409981</td>
<td>0.0872539</td>
<td>0.2308090 - 0.5809332</td>
</tr>
<tr>
<td>3:Ph1</td>
<td>0.4439275</td>
<td>0.0633644</td>
<td>0.3289607 - 0.5589306</td>
</tr>
<tr>
<td>4:Ph1</td>
<td>0.5516725</td>
<td>0.0632644</td>
<td>0.3496994 - 0.6701393</td>
</tr>
<tr>
<td>5:Ph1</td>
<td>0.4892334</td>
<td>0.0532551</td>
<td>0.3868384 - 0.5925990</td>
</tr>
<tr>
<td>6:Ph1</td>
<td>0.5107465</td>
<td>0.0532551</td>
<td>0.4074010 - 0.6131815</td>
</tr>
</tbody>
</table>

we see that $\hat{\phi}_1 + \hat{\phi}_2 = 0.5949 + 0.4099 = 1.0000$, $\hat{\phi}_3 + \hat{\phi}_4 = 0.4439 + 0.5517 = 1.0000$, and $\hat{\phi}_5 + \hat{\phi}_6 = 0.4893 + 0.5107 = 1.0000$, as expected.

While constructing the MLogit link is straightforward, you need to be careful. Consider the following set of parameters in a PIM for the survival probability for the male Dipper data:

![Apparent Survival Parameter](image2)
10.3.1. A simple metapopulation model – size, distance & quality

The parameter-specific link would be selected in the ‘Setup Numerical Estimation Run’ window, and the MLogit(1) link would be applied to parameters 1 → 6 to force these 6 estimates to sum to \( \leq 1 \).

But suppose that instead you wanted to force all of the 6 survival probabilities to be the same, and have the sum of all 6 be \( \leq 1 \)? You might be tempted to specify a PIM such as

(i.e., simply use the same index value for all the parameters in the PIM, which would result in the same estimate for each interval), and apply the MLogit link to parameter 1, but that would be incorrect. Changing the PIM and selecting the MLogit link for parameter 1 would result in parameter 1 alone summing to \( \leq 1 \) (i.e., just like a logit link), but would not force the sum of the 6 values of parameter 1 to sum to \( \leq 1 \). Go ahead and try it for yourself – you’ll see that whether or not you use the MLogit or logit link, parameter 1 is estimated (for a model with time-varying encounter probability) as 0.5561. Clearly, \((6 \times 0.5561) \gg 1\).

But, what if you want the sum of the 6 estimates to be 1.0? To implement such a model, the PIM should not be changed from a time-dependent PIM (i.e., it should maintain the indexing from 1 → 6); instead, the design matrix should be used to force the same estimate for parameters 1 → 6. Then the MLogit(1) link should be specified for all 6 parameters for apparent survival, 1 → 6. The result is that now all 6 parameters have the same value (\( \hat{\phi} = 0.166 \)) for the model specified in this DM – with time-dependent encounter probability), where \( (6 \times 0.166) = 1 \).

Another example – suppose you wanted parameters 1 and 4 → 6 (non-flood years) to be the same value, and parameters 2 and 3 (flood years) to be the same value. Obviously, there are multiple ways to implement such a model in MARK – the approach you use will be determined by what constraints you want to implement. If you want to have a separate estimate of apparent survival for flood and non-flood years, and (i) have the same estimate for all flood and non-flood years, and (ii) have the estimates within a flood-type sum to 1.0, then you need to use the design matrix, and apply the MLogit link function to the appropriate parameters. If we apply MLogit(1) to flood years (parameters 1, 4 → 6), and MLogit(2) to non-flood years (parameters 2 and 3), we end up with estimates of apparent survival of 0.25 for each of the 4 flood years \((4 \times 0.25 = 1.0)\), and 0.4965 for each of the 2 flood years \((2 \times 0.4965 = 0.9930 \leq 1.0)\).

What happens if we apply a single MLogit constraint to all 6 parameters (i.e., MLogit(1) to parameters 1 → 6)? As you might expect, applying this constraint will still yield separate estimates of apparent survival for all flood and non-flood years, but now, the sum of estimates over all 6 parameters will be \( \leq 1 \). What we see if we run this model, with MLogit(1) applied to parameters 1 → 6 is \( \hat{\phi}_{\text{flood}} = 0.1857 \), and \( \hat{\phi}_{\text{non-flood}} = 0.1286 \). Summing over all estimates, \((4 \times 0.1857) + (2 \times 0.1286) = 1.0 \).

Now, final test – what if you want to have a single separate estimate for flood and non-flood years, and have them sum to 1.0? In other words, instead of generating an estimate for each year subject to the constraint, you want to generate an estimate for each flood type subject to the constraint (so, 2 estimates, not 6, with the sum of the estimates \( \leq 1 \)).

With a bit of thought, you should realize that to fit this particular model, you do, in fact, need to first modify the PIM (shown at the top of the next page), which ultimately controls the number of estimated parameters – and then apply the MLogit constraint to the 2 parameters specified in the PIM.
Here is the modified PIM – parameter 1 indicating the flood years, and parameter 2 indicating the non-flood years:

If we run this model without applying the MLogit constraint, using instead the standard logit link, we see that we obtain estimates of $\hat{\phi}_{\text{flood}} = 0.5970$, and $\hat{\phi}_{\text{non-flood}} = 0.4725$. Note that the sum of these estimates $0.5970 + 0.4725 = 1.07 > 1.0$.

Now, if we re-run the analysis, but apply the MLogit constraint to both parameters (i.e., MLogit(1) for both parameters 1 and 2), we obtain estimates of $\hat{\phi}_{\text{flood}} = 0.5728$, and $\hat{\phi}_{\text{non-flood}} = 0.4272$ – the sum of these constrained estimates is $(0.5728 + 0.4272) = 1.0$, as expected.

The key point with these examples is that the PIM cannot be used to constrain parameters if you want the entire set of parameters (i.e., over all intervals) to sum to $\leq 1$. Rather, the design matrix has to be used to make the constraints, with each of the entries in the PIM given the same MLogit(x) link.

---

10.4. Multi-state models as a unifying framework

Multi-state models offer great potential to increase our understanding of complex, structured systems – systems with multiple states, and stochastic (or probabilistic) transitions among states. MARK makes it fairly straightforward to fit some relatively complex models to the underlying multi-state structure.

This point was first noted in a paper by Lebreton, Almeras & Pradel (1999), who pointed out that
multi-state modeling does, in fact, have the potential to be a common, unified ‘framework’ under which a large variety of models can be fit – including those combining information from multiple sources. Using data from multiple types will be discussed in a later chapter – for the moment, we’ll introduce the conceptual framework described by Lebreton et al., to give you the sense of ‘how it is done’, and (with a bit of thought) how easy it is to implement.

10.4.1. Simple example (1) – CJS mark-recapture as a MS problem

We’ll start by considering a simple mark-recapture analysis, based on live-encounter data. While we already have plenty of tools to handle these sorts of data, the simplicity of this data type, and your familiarity with it, make it a good starting point. First, keep in mind that the multi-state approach considers multiple states. In a mark-recapture analysis (or any simple survival analysis), there are 2 states of interest: live, and dead.

As noted by Lebreton et al, the interesting ‘paradoxical’ issue with mark-recapture analysis is that the information on survival (or, equivalently, mortality) is not based on observations of dead animals. Some animals are in fact in the ‘dead’ state, but are never seen. So, if a ‘dead’ state animal is never seen, then clearly, the recapture probability for this state is 0. This leads quite logically to a fairly straightforward representation of the CJS model as a 2-state model (Alive=1, Dead=2), with a 0 probability of capture in the second state (i.e., when dead, or state 2, p = 0).

As such, we can define the following transition probabilities. Let \( q_i \) = probability of surviving from time \( i \) to \( i+1 \). Thus, the probability of surviving and moving from state 1 to 1 (i.e., from live \( \rightarrow \) live) is clearly \( q_i \). The probability of moving from state 1 to 2 (live \( \rightarrow \) dead) is \( 1 - q_i \). The probability of moving from dead to live is clearly 0. And the probability of moving from dead to dead is 1 (i.e., if you’re already dead, then you will be dead at the next occasion also). Now, if you’re in state 1 (live), the recapture probability is \( p \), while if you’re in state 2, the recapture probability is 0.

We can express these transitions in a 2-state transition matrix \( \Psi \), and the recapture probabilities in a 2-state vector, \( p \):

\[
\begin{bmatrix}
1 & 2 \\
p & 0
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
1 & 2 \\
\varphi & 0 \\
1 - \varphi & 1
\end{bmatrix}
\]

For the transition matrix \( \Psi \), the rows correspond to the state at time \( i+1 \) (live and dead for rows 1 and 2), and the columns correspond to the state at time \( i \) (live and dead for columns 1 and 2). Note that the matrix must be constrained to have the sum of each column be equal to 1. Using ‘Alive’ = state 1, and ‘Dead’ = state 2, a typical capture history might be ‘00110100’. No ‘2’ ever appears since that state is never observed.

To demonstrate this analysis, we simulated a basic CJS data set (CJS_MS.INP) – 8 occasions, big sample sizes, with the following parameter values:

<table>
<thead>
<tr>
<th>occasion</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \varphi )</td>
<td>0.5</td>
<td>0.85</td>
<td>0.85</td>
<td>0.65</td>
<td>0.50</td>
<td>0.60</td>
<td>0.85</td>
</tr>
<tr>
<td>( p )</td>
<td>0.45</td>
<td>0.45</td>
<td>0.55</td>
<td>0.75</td>
<td>0.75</td>
<td>0.45</td>
<td>0.55</td>
</tr>
</tbody>
</table>

So, basic time dependence in both survival and recapture probability – model \{ \( \varphi_i \) \( p_i \) \}.  

Chapter 10. Multi-state models...
Estimates from fitting this model to the data using a ‘normal’ live-encounter CJS approach are shown below:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 PHI</td>
<td>0.492435</td>
<td>0.0134292</td>
<td>0.466159 - 0.5187528</td>
</tr>
<tr>
<td>2 PHI</td>
<td>0.8373917</td>
<td>0.0123089</td>
<td>0.8118035 - 0.8601003</td>
</tr>
<tr>
<td>3 PHI</td>
<td>0.8579654</td>
<td>0.0113201</td>
<td>0.8331357 - 0.8827609</td>
</tr>
<tr>
<td>4 PHI</td>
<td>0.6528416</td>
<td>0.0096733</td>
<td>0.6336476 - 0.6720312</td>
</tr>
<tr>
<td>5 PHI</td>
<td>0.5033981</td>
<td>0.0087432</td>
<td>0.4847559 - 0.5219518</td>
</tr>
<tr>
<td>6 PHI</td>
<td>0.6045259</td>
<td>0.0126119</td>
<td>0.5795616 - 0.6298620</td>
</tr>
<tr>
<td>7 PHI</td>
<td>0.682435</td>
<td>0.0164394</td>
<td>0.6583824 - 0.7064881</td>
</tr>
<tr>
<td>8 PHI</td>
<td>0.4198697</td>
<td>0.0106304</td>
<td>0.3890055 - 0.450035</td>
</tr>
<tr>
<td>9 PHI</td>
<td>0.4447901</td>
<td>0.0109201</td>
<td>0.4141080 - 0.475470</td>
</tr>
<tr>
<td>10 PHI</td>
<td>0.5606005</td>
<td>0.0095201</td>
<td>0.5308160 - 0.590985</td>
</tr>
<tr>
<td>11 PHI</td>
<td>0.7510784</td>
<td>0.0097206</td>
<td>0.7213439 - 0.780839</td>
</tr>
<tr>
<td>12 PHI</td>
<td>0.7688848</td>
<td>0.0109886</td>
<td>0.7388833 - 0.798735</td>
</tr>
<tr>
<td>13 PHI</td>
<td>0.4555138</td>
<td>0.0114728</td>
<td>0.4331274 - 0.4780832</td>
</tr>
<tr>
<td>14 PHI</td>
<td>0.6021416</td>
<td>0.0174564</td>
<td>0.5853029 - 0.6206820</td>
</tr>
</tbody>
</table>

Now let’s analyze these data using a multi-state approach. Start MARK, and select the ‘multi-state data type’. Specify 8 occasions, and 2 states – the .IIR file uses classic ‘0’ or ‘1’ coding for a recapture data set, so change state A to 1, and label it ‘live’, and state B to 2, and label it ‘dead’:

Now, the only potentially ‘tricky’ part of the analysis. Open up the PIM chart. You’ll see that there are the 6 blue boxes – 2 for survival, 2 for recapture, and 2 for movement, for the 2 states, respectively.

Now, some thinking. Based on the preceding page, we know that recapture probability for dead individuals (state 2) is 0 (in other words, we assume that we never see dead individuals. In effect, we’re modeling movement into an ‘unobservable state’). So, right click the blue box corresponding to recapture probability for that state, and set it ‘constant’ – we will fix the parameter value later, during the numerical estimation run. Renumber it with (or without) overlap – doesn’t much matter in this case.

Next, we know that the probability of moving from dead to live is 0. So, we set this transition (from state 2 to state 1) constant – again, we will fix the value of this parameter to be 0 during the numerical estimation run. We also need to set the survival of state 2 individuals (i.e., dead individuals) constant – once dead, always dead, so the probability of surviving and staying in this state is clearly 1. Now, while this might seem a bit strange at first, remember that we are considering probabilities of survival and movement between states. Go ahead and modify the PIM chart, followed by renumbering to eliminate the ‘spaces’ between the blue boxes (alternatively, you can manually drag the boxes so they are effectively contiguous).

Now, for the last step – and one you might have to think about for a minute. What about the movement parameter from state 1 to 2 (i.e, live to dead)? Clearly this movement (from live to dead) is a logical complement to the probability of dying, which is defined as \((1 - \phi)\). So, in order for an individual to enter state 2, it must die. State 2 is clearly an ‘absorbing state’ – once entered, it can’t be left. Thus,
the probability of moving from state 1 to state 2, conditional on surviving from $i$ to $i+1$ (which is the assumption we’re making throughout) is clearly 0 in this case. If you survive from $i$ to $i+1$, then you clearly can’t move from state 1 (live) to state 2 (dead)! Read this section again – a couple of times if needed – to make sure you have the logic down. Once you’ve gotten a good grip on the idea, simply modify the PIM chart accordingly – set the parameter structure for the movement probability from state 1 (live) to state 2 (dead) to be constant.

The modified PIM chart is shown below:

![Modified PIM Chart](image)

That’s about it. Now, the only thing we need to do is fix some of the parameters. From the PIM chart, we want to set parameter 8 (the ‘survival’ probability of the individuals in the dead state) to be 1, and parameters 16, 17, and 18 (the encounter probability for dead individuals, and the movement probabilities – which are conditional on survival) to be 0. To do this, click the ‘Fix Parameters’ button, and set the parameter to the desired value (on the real probability scale):
Go ahead, run the model, and look at the reconstituted estimates:

![Image of table showing reconstituted estimates]

The estimates for survival and recapture probability are identical to what we observed earlier, using a ‘normal’ CJS approach to this analysis.

Again, at this point, you might be asking yourself – why bother with a multi-state approach to this analysis, when we could have just as easily done it (in fact, more easily) using the standard ‘recapture’ analysis? The reason is – if you understand this multi-state approach in this simple case, then it won’t be too difficult to see how it can be applied to other data types – for example, dead recovery data, our next example.

10.4.2. Simple example (2) – dead-recovery analysis as a MS problem

Lebreton, Almeras & Pradel (1999) showed that analysis of dead recoveries can be naturally reframed as multi-state capture-recapture models with two discrete states (live and dead). When framing recovery models as multi-state models, death (mortality) is represented by the transition between ‘live’ and ‘dead’, where the ‘dead’ state is then an ‘absorbing state’ (meaning, entry into the ‘dead’ state is permanent). The main challenges in implementing them in MARK are to some degree ‘conceptual’, and ‘mechanical’ (in particular, the re-formatting of the .inp file that will be required).

In multi-state models, animals move and are potentially detected (‘encountered’ in the broad sense) in one of $N$ possible states. They are parameterized in terms of an $(N \times N)$ transition matrix $\Psi$, an $(N \times 1)$ vector $p$ of capture probabilities, and a $(N \times 1)$ vector $S$ of survival probabilities. This is the $(S, \Psi, p)$ parametrization. This framework can be used for band-recovery models if one defines 2 discrete states: newly marked (say, ‘banded’) alive (state B) and newly-dead (state N).

However, for multi-state analysis of dead recovery data, building encounter histories takes a bit of thought. While marking and live re-encounters occur during discrete, short periods (i.e., at discrete ‘encounter occasions’), dead recoveries occur throughout the interval between discrete live marking/encounter occasions. of the interval between two occasions.
Consider the following diagram – dead recoveries occur between discrete sampling occasions (1, 2, 3...).

Since MS models are used to estimate transition probabilities among discrete states, then we clearly need to ‘discretize’ time of entry into the ‘newly dead’ state. To do this, individuals recovered between occasion \( i \) and \( i + 1 \) will be coded as entering the state ‘newly-dead’ (N) at the discrete occasion \( i + 1 \). For instance, with a study of 4 occasions, the capture history for an animal marked and released alive at occasion 1 and recovered dead between time 2 and 3 would then be ‘A000’. Note that if we were using standard LDLD coding (chapter 2, chapter 8), then this individual history would be coded ‘1000100’. And, in fact, this is the first challenge in using a MS approach to analyze dead recovery data in MARK – you first need to reformat your data such that dead recoveries are coded as entering the new state (in this case, the newly dead state N) at the end of the interval during which they were recovered. If you have some ‘programming skills’, this isn’t perhaps too difficult. But, it is necessary.

To give you a sense of what is required, consider the first few lines of a data set (recovery-LDLD.inp) coded in standard LDLD format: the data consists of 5 occasions of mark and release, and 5 intervals during which dead recoveries can occur.

```
1000000000  1649;
1001000000  74;
1000010000  61;
1100000000  134;
1000000001  40;
1000000100  42;
```

So, how do we ‘transform’ these LDLD encounter histories to MS format? The first thing we have to remember is that individuals recovered between occasion \( i \) and \( i + 1 \) will be coded as entering the state ‘newly-dead’ (N) at the discrete occasion \( i + 1 \). Sounds straightforward, but what about the final occasion/interval?

In fact, with 5 mark and release occasions, we will need to restructure our MS encounter histories to have 6 occasions – 5 true mark and release occasions (1 → 5), and one ‘virtual’ occasion (6) which represents the ‘occasion’ on which the newly dead and recovered during the interval following sampling occasion 5 are tabulated (as entering the newly dead state N):

Once you grasp the basic idea, the next steps needed to re-format the encounter history data are relatively easy. Take, for example, the LDLD history Take the LDLD encounter history ‘1001000000’. For
this history, there are 5 LD pairs (\textbf{1}001000000) – live mark and release at the first sampling occasion, dead recovery over the second interval, and then never encountered again. How do we transform this to the MS format?

The key step is to remember that we’re discretizing the interval over which dead recoveries occur, such that a dead recovery occurring over the interval \(i \rightarrow i + 1\) is coded as entering the ‘newly dead’ state (\(N\)) at \(i + 1\). The trick, then, is to correctly assign evens for a given LD pair to the appropriate single occasion in the encounter history reformatted for a multi-state analysis.

One way to ‘get events lined up’ with the correct discrete occasion is to re-write the contiguous encounter history by first separating the initial 1 from the rest of the history – recall that for any history the first ‘1’ will always represent the newly marked individuals.

So, for example,

\[ \text{‘1001000000’} \rightarrow \text{‘1 00100000’}. \]

Next, split the contiguous part of the history into consecutive pairs:

\[ \text{‘1 00100000’} \rightarrow \text{‘1 00 10 00 00 00’}. \]

Now, here is the key – each of these ‘pairs’ represent events that occur at a particular discrete occasion, with the final ‘0’ representing a virtual occasion 6 (to allow for dead recoveries after the final marking event at occasion 5 – we simply add a ‘0’ to complete the ‘pair’ at occasion 6).

So, the initial ‘1’ is the initial mark and release event at occasion 1, the first pair, ‘00’, refers to events that will be coded as if they occurred at occasion 2, then the next pair, ‘10’, refers to events that will be coded as if they occurred at occasion 3, and so on. The basic idea is represented in the following:

\begin{center}
\begin{tabular}{cccccc}
\text{‘1’} & \text{‘00’} & \text{‘10’} & \text{‘00’} & \text{‘00’} & \text{‘00’} \\
\text{1} & \text{2} & \text{3} & \text{4} & \text{5} & \text{(6)}
\end{tabular}
\end{center}

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

What next? Well, the initial ‘1’ for any history will always represent the newly marked individuals – so, we simply change the initial ‘1’ to ‘B’:

\[ \text{‘1 00 10 00 00 00’} \rightarrow \text{‘B 00 10 00 00 00’}. \]

Next, for a ‘dead recovery’ study, (i) the dead recovery only occurs once, and (ii) there can only be one event at a given occasion – and, other than the release occasion that event is the transition to ‘newly dead’ (\(N\)).

Thus, a ‘00’ pair indicates ‘no event’ (\(0\)), whereas a ‘10’ pair indicates ‘newly dead’ (\(N\)), and our encounter history would be rewritten as

\[ \text{‘B 00 10 00 00 00’} \rightarrow \text{‘B 0 N 0 0 0’}. \]

which, after removing the spaces, yields

\[ \text{‘B 0 N 0 0 0’} \rightarrow \text{‘B0N000’}. \]
Let’s try another one: consider the encounter history ’1100000000’. Here, we have both the ‘live marking event’ and the ‘dead recovery event’ occur in the same LD pair. But, the key is remembering that we associate the dead recovery with sampling occasion 2. Thus, the transformation would go 

‘1100000000’ → ‘1 10 00 00 00 00’ → ‘B N 0 0 0 0’ → ‘BN000’

Finally, what about ‘1000000000’? For this history, we have a dead recovery in the final LD pair, of the 5 LD pairs in the history. But, as noted earlier, since a ‘dead recovery event’ over interval i → i + 1 is associated with occasion i + 1, then we need to add a 6th occasion (even though it didn’t really exist in our data).

Here, the transformation would go 

‘1000000000’ → ‘1 00 00 00 10’ → ‘B 0 0 0 0 N’ → ‘B000N’

So, here (below) are the first few lines of the transformed input file (recovery-MS.inp). Make sure you see the connection between each of these transformed encounter histories and the same lines for the corresponding history in LDLD format shown a few pages back:

| B00000 | 1649; |
| B0N000 | 74; |
| B00N00 | 61; |
| BN0000 | 134; |
| B000N0 | 40; |
| B00N00 | 42; |

OK – now that we have our ‘transformed’ data file ready, what next? Well, now we move from a ‘mechanical’ consideration (data formatting) to a ‘conceptual’ one.

Here is the 2-state transition matrix \( \Psi \), and the 2-state survival and encounter vectors, \( S \) and \( p \):

\[
\Psi = \begin{bmatrix}
B & N \\
\psi_1^{11} & 1 - \psi_1^{11}
\end{bmatrix}, \quad S = \begin{bmatrix}
B \\
1 - N
\end{bmatrix}, \quad p = \begin{bmatrix}
B \\
0
\end{bmatrix}
\]

By fixing the first entry of the \( S \) vector to 1, then the elements of the first row of \( \Psi \) represent survival \( S \) (element \( \psi_1^{11} \)) and mortality \( 1 - S \) (element \( 1 - \psi_1^{11} \)), respectively. In the second row of the matrix, the transition from state \( N \rightarrow B \) is logically fixed to 0, and thus its complement (i.e., the transition from state \( N \rightarrow N \)) is by default 1.

Now, a subtle point. In theory, a third state (which we might call ‘permanently dead’) is needed because an animal can be recovered only once in the \( N \) state (i.e., the year it dies). Failing to consider such a ‘permanently dead’ state would mean that an individual entering the \( N \) state could be recovered forever, which is clearly a logical impossibility. However, because such a ‘permanently dead’ state to which individuals move immediately after death contributes no information (i.e., it is never encountered and transitions to this state are all fixed), it can be ignored. We simply need to fix the survival of the state \( N \) to 0 in the \( S \) matrix.

Now, we’re ready to proceed with the analysis. First, for purposes of comparison, we’ll start with a ‘classical’ dead recovery analysis of these data, using the Seber parameterization. We’ll fit the default time-dependent model \( \{S_t, r_t\} \) to these data. The model deviance was 5.8316. Parameter estimates are shown below:
As expected for a fully time-dependent model, the final estimates of $S$ and $r$ are confounded.

Now, let’s fit these data using a MS approach. We have to start a new project in MARK, specifying a multi-state data type with 2 states (B and N), and 6 occasions. Since we want to estimate survival (and not mortality), we want to change the default PIM definitions so that $\psi_{BD}$ is estimated by subtraction.

Next, we need to modify the PIM structure to represent the transition structure for the 3 parameters in our model. Recall that the model we’re trying to fit is $\{S_t r_t\}$.

$$
\Psi = \begin{bmatrix}
B & N \\
\psi^{11} & 1 - \psi^{11}
\end{bmatrix},
\quad
S = \begin{bmatrix}
B \\
0
\end{bmatrix},
\quad
P = \begin{bmatrix}
0 & r
\end{bmatrix}
$$

The multi-state PIM structure for this model (given these transitions) is shown at below:
As a check, notice that the PIM structure has only 2 time-dependent parameters ($\psi_{BB}^N$, representing survival, and $p_{NN}^N$, represent the recovery probability $r$) – all the other parameters are fixed constants. This makes sense, since the model we’re trying to fit $\{S, r\}$ clearly has only two parameters, both time-dependent.

Before running the model, we need to fix some parameters. From the transition structures on the preceding page, we see that we fix survival for the newly banded individuals in state B (i.e., parameter 1) to 1, and the survival for newly dead individual in state N (i.e., parameter 2) to 0. Clearly, the encounter probability for the newly banded individuals (i.e., parameter 3) is fixed to 0.

Finally, the probability of moving from state N to B (i.e., from dead to live; parameter 14) is fixed to 0. After fixing the parameters, we run the model. Model deviance is reported as 5.8316, which is identical to the deviance reported for the same model using the classical dead recovery analysis.

The parameter estimates from the MS analysis are shown below:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: B to B</td>
<td>1.000000</td>
<td>0.000000</td>
<td>Fixed</td>
</tr>
<tr>
<td>2: D to B</td>
<td>0.000000</td>
<td>0.000000</td>
<td>Fixed</td>
</tr>
<tr>
<td>3: B to B</td>
<td>0.000000</td>
<td>0.000000</td>
<td>Fixed</td>
</tr>
<tr>
<td>4: D to B</td>
<td>0.193333</td>
<td>0.033333</td>
<td>Fixed</td>
</tr>
<tr>
<td>5: D to B</td>
<td>0.415000</td>
<td>0.224773</td>
<td>Fixed</td>
</tr>
<tr>
<td>6: D to B</td>
<td>0.149724</td>
<td>0.050950</td>
<td>Fixed</td>
</tr>
<tr>
<td>7: D to B</td>
<td>0.300943</td>
<td>0.204638</td>
<td>Fixed</td>
</tr>
<tr>
<td>8: D to B</td>
<td>0.095922</td>
<td>0.000000</td>
<td>Fixed</td>
</tr>
<tr>
<td>9: B to B</td>
<td>0.651652</td>
<td>0.052985</td>
<td>Fixed</td>
</tr>
<tr>
<td>10: B to B</td>
<td>0.820390</td>
<td>0.057529</td>
<td>Fixed</td>
</tr>
<tr>
<td>11: B to B</td>
<td>0.654362</td>
<td>0.059766</td>
<td>Fixed</td>
</tr>
<tr>
<td>12: B to B</td>
<td>0.380838</td>
<td>0.092641</td>
<td>Fixed</td>
</tr>
<tr>
<td>13: B to B</td>
<td>0.295420</td>
<td>0.000000</td>
<td>Fixed</td>
</tr>
<tr>
<td>14: B to B</td>
<td>0.000000</td>
<td>0.000000</td>
<td>Fixed</td>
</tr>
</tbody>
</table>

If you compare these estimates to those from the classical dead recovery analysis (on the preceding page), you’ll see they are identical (at least for the non-confounded parameters).

Pretty slick, eh? We’ll leave it to you to figure out how to extend this approach to a joint live encounter-dead recovery analysis (see Lebreton et al. 1999 and Lebreton & Pradel 2002 for details).

10.4.3. A more complex example – recruitment probability

To really make the flexibility of this approach clear, we’ll now look at an example related to the earlier question concerning different breeding states (breeding, and non-breeding), but with a twist – here we’re going to look at recruitment, which we’ll define as the probability of moving between a non-breeder to a first time breeder. Now, unlike the analysis of breeding state we presented earlier in the chapter, but analogous to the multi-state approach to recapture analysis we just completed, we’re interested here in a ‘permanent’ state transition. In the recapture analysis, we were interested in the transition from ‘live’ to ‘dead’. The various constraints we imposed during the numerical estimation reflect the fact that the transition is permanent (once dead, always dead). Here in this example, we’re also considering a permanent state transition – from ‘non-breeder’, defined as a bird which has never bred, to a ‘breeder’, or (perhaps more accurately, a ‘recruit’) – an individual which has become a breeder (i.e., has bred at least once). Now, once an individual is a breeder, it is always a breeder. It may not breed in every year following first breeding, but it is always a breeder.

This question, and various approaches to estimation of the probability of making this transition from non-breeder to recruit, have been discussed at length by Pradel & Lebreton (1999), and references therein.
Our purpose here is merely to point out again how the multi-state approach can be used to deal with situations where there is a non-observable state. What is the non-observable state? In this case, it is the non-breeding (pre-recruitment) state (which we’ll call NB). In many species, only breeding individuals are encountered. Thus, we need to separate the effects of being a NB individual (for which \( p = 0 \)) from death. The multi-state approach is one way to tackle this problem.

Consider the following situation (as described by Pradel and Lebreton). The probability of making the permanent transition from non-breeding to breeding (i.e., the probability of recruiting) is governed by the parameter \( a \). Formally, let \( a_i \) be the probability that an as yet inexperienced individual of age \( i \) starts to breed at that age. An individual is marked as a newborn, and then each year, you go out to look for that individual. If you encounter the individual, then it has both survived, and recruited. If you don’t encounter the individual, it is either because it hasn’t survived, or that it hasn’t recruited (and is thus non-observable).

Take the encounter history ‘2011’, where ‘2’ denotes the birth date (time of initial marking), and ‘1’ denotes ‘breeding’ or ‘recruited’. Assume there are only 2 age-specific survival probabilities (\( q_j \) and \( q_a \)), a constant recapture probability \( p \), and age-specific probabilities of recruitment (\( a_1, a_2 \) and \( a_3 \)). Thus, the associated probability of this encounter history is \( q_j [(1 - a_1) a_2 + a_1 (1 - p)] q_a p, q_a p \).

Pradel and Lebreton then simulated a data set, setting \( q_j = 0.4, q_a = 0.8, p = 0.5 \). They assumed that survival did not differ among pre-recruits (non-breeders) and recruits. For age-specific recruitment probabilities, they used \( a_1 = 0.2, a_2 = 0.375, \) and \( a_3 = 1.0 \).

Simulating the encounter histories for a single cohort of 5,000 individuals, they arrived at the following histories:

<table>
<thead>
<tr>
<th>history</th>
<th>frequency</th>
<th>history</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>2111</td>
<td>32</td>
<td>2011</td>
<td>128</td>
</tr>
<tr>
<td>2110</td>
<td>48</td>
<td>2010</td>
<td>192</td>
</tr>
<tr>
<td>2101</td>
<td>32</td>
<td>2001</td>
<td>448</td>
</tr>
<tr>
<td>2100</td>
<td>88</td>
<td>2000</td>
<td>4,032</td>
</tr>
</tbody>
</table>

Let’s analyze these data using MARK. The transition matrices governing these data are:

\[
\begin{bmatrix}
1 - a_i & 0 \\
 a_i & 1
\end{bmatrix}
\begin{bmatrix}
q_j \\
q_a
\end{bmatrix}
\begin{bmatrix}
0 \\
p, p
\end{bmatrix}
\]

The procedure for analyzing these data using MARK is mechanically similar to what we just did when using the multi-state approach to analyze live recapture data. The only difference in this case is that we have to impose different constraints. The challenge is to determine constraints to impose, and whether or not they make ‘biological’ sense (as opposed to constraints that are needed out of structural necessity to make one or more parameters identifiable).

We ‘know’ from the simulated data that recapture probability is constant, so we’ll modify the PIM chart to reflect this. We also know that the probability of capturing a non-breeding pre-recruit is 0, so we set recapture probability for non-breeders to be a constant (we’ll fix it to 0) just before the numerical estimation run. We also know the probability of moving from breeder to non-breeder is 0, so we will set that parameter to be 0 before the numerical estimation. Finally, the ‘a’ parameter we’re really interested in – this corresponds to the probability of moving from non-breeder to breeder. We ‘believe’ that this is likely to be age-specific, so we use a simple time-specific parameterizations for this probability (remember, ‘age = time within cohort’, and here, we’re dealing with a single cohort).
Here is the resulting PIM chart:

![PIM Chart]

and the corresponding DM:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

We proceed to run the numerical estimation, first fixing the recapture probability for non-breeders (parameter 8 in the PIM chart) to 0, and the probability of moving from breeder back to non-breeder (parameter 9 in the PIM chart) to 0.

So far, so good – but what about the probability of moving from non-breeder to breeder (parameter \(a\))? It might seem \textit{a priori} there is no need to fix any of this parameters – after all, these are the parameters we’re interested in estimating in the first place. Anything we need to do with the survival parameters? Perhaps nothing obvious.

However, once we run this model, and look at the estimates (top of the next page), we see quickly that fixing only parameters 8 and 9 as described leads to all sorts of problems. Although numerical convergence is reached, the estimates for virtually all of the parameters are clearly wrong – anything estimated with a standard error of 0 or a 95% CI from 0 \(\rightarrow\) 1 is clearly wrong (even with simulated data!).
Nonetheless, note that the estimates themselves ‘seem’ to approximate the ‘true’ values used in the simulation rather well, except for $\varphi_{ij}$, and the final recruitment probability $a_3$. This sort of thing often implies that one (or more) parameters are not identifiable under the given set of constraints – either because the constraints are incorrect, or because you haven’t constrained enough parameters. In this case, it turns out that the latter is the cause of the problems here.*

What other constraint(s) do we need? Well, as it turns out, we need to impose 2 further constraints. First, consider the survival parameter. How can we estimate the survival probability of individuals we don’t see (i.e., the pre-breeders)? The answer is – we can’t. We need to apply a constraint, wherein we need to assume that survival of pre-breeders and breeders is the same for a given interval. We can do that simply by deleting the ‘state’ and ‘state.time’ interaction columns from the design matrix.

In addition, we also need to assume that there is some age (in this example, age 3) by which all individuals in the population that are still alive will have recruited (in other words, this involves setting $\psi_{33} = 1$, by fixing parameter 12 to 1.0 in the numerical estimation). [Note, in fact, that the data were simulated assuming $a_3 = 1$ in the first place.] This second assumption is explained in detail in the Pradel & Lebreton paper.

If we re-run the analysis, fixing parameter 8 and parameter 9 to 0, and also fixing parameter 12 to 1, we see that our estimates (shown below) are now ‘correct’.

---

* The identification of parameters that are not estimable given the structure of the model is discussed in Appendix F.
Now, clearly, the key step was assuming that there was an age after which all individuals were recruited, conditional on still being alive. This is a strong assumption, and one that makes application of this approach somewhat problematic – see Pradel & Lebreton for a full discussion. However, the point here is to demonstrate the methodology, and not to provide a full treatment of this complex problem. As noted by Pradel & Lebreton, the multi-state approach may have utility for this particular analysis, given some assumptions. But, more importantly, we again see how useful the multi-state approach can be in dealing with states which are not observable.

Estimation problems: local minima – approaches and solutions

In the preceding example, our estimates were ‘wonky’ (nonsensical) until we applied the appropriate and necessary logical constraints to the model. However, this is not always the case. Sometimes, multi-state models have ‘problems’ converging to global maxima. In fact, the multi-state model can display some heinous behavior when there are > 2 states, most notably multiple optima in the likelihood function (Lebreton & Pradel 2002). That is, depending on the starting values used to maximize the likelihood function, the solution can vary. Most of the models we have explored so far have a single maximum, and so this behavior is not encountered.

There are several approaches to handling these sort of problems, which we describe here. First, for multi-state models in particular, it is not a bad idea to first build a simple, time-constant ‘dot’ model containing all the factors of interest. We will use this model to generate ‘good starting values’ for the the more general model. Note that previously, we advocated first building the most general model in your candidate model set, for which you will then estimate \( \hat{t} \) (see chapter 5). In general, this works fine. But for multi-state models in particular, and other data types where you might be having some ‘numerical estimation problems’ for your general model, using starting values from a simpler model often helps you avoid some problems.

In some cases, even very simple models will have problems. A second approach is to make use of a different link function. In some cases, a different link function may ‘do better’ at navigating what might be a multi-modal likelihood surface. In fact, if you find fitting a given model to a data set using different link functions yields very different model deviances, then this is a reasonable indicator that you might be having problems finding the global minima with some link functions.

Finally, you might try using an alternative optimization procedure available in MARK. This procedure, based on simulated annealing, is selected by clicking the ‘alternate optimization’ checkbox on the right-hand side of the ‘run numerical estimation’ window. Simulated annealing (Goffe et al. 1994) is less computationally efficient than the default optimization algorithm in finding the maximum of the function, typically requiring many more evaluations of the likelihood to reach a solution. However, the reason for this ‘inefficiency’ is why simulated annealing is provided in MARK; periodically, the simulated annealing algorithm makes a random jump to a new parameter value, and this characteristic is what allows the algorithm more flexibility in finding the global maximum (instead of getting stuck at a local maximum).

We will demonstrate the various approaches to handling ‘estimation problems’ using an example multi-state data set with a known local minimum: 2 states (1 and 2), 7 occasions (this example was extracted from an example from Jerome Dupuis, used in a recent paper by Gimenez et al. 2005 – JABES). Here are the encounter histories:

```
202102 4; 1110101 4;
202001 4; 1010101 4;
202002 4; 1010102 4;
2201012 4; 2182011 4;
```

The true parameter values are \( S = 1.0 \) (constant over time, and the same for both states), \( p = 0.6 \) (constant over time, and the same for both states), \( \psi^{12} = 0.60 \) (constant over time), and \( \psi^{21} = 0.85 \) (constant over time). If you fit model \( \{ S, p, \psi_{12} \} \) (which is the true model) to the data in MARK, using
the default numerical optimization routine with a sin link, we get the following estimates. We see clearly that \textsc{Mark} has had some problems coming up with estimates for $\hat{\psi}$ that are even remotely close to the true values.

\begin{table}
\centering
\begin{tabular}{lllll}
\toprule
\textbf{Parameter} & \textbf{Estimate} & \textbf{Standard Error} & \textbf{95\% Confidence Interval} \\
\midrule
1: 5 1:1 & 1.0000000 & 0.0000000 & 1.0000000 & 1.0000000 \\
2: p 1:1 & 0.5833333 & 0.3352797 & 0.5123869 & 0.6502884 \\
3: P5 1 to 2 & 0.2400010 & 0.0666486 & 0.1406682 & 0.3993171 \\
4: P5 2 to 1 & 0.3419552 & 0.0735735 & 0.2213339 & 0.5009178 \\
\bottomrule
\end{tabular}
\end{table}

What has happened? Well, take a look at the likelihood profile for values of $\psi^{21}$ from 0.2 → 1.0, shown below:

![Likelihood Profile](image)

We see from the likelihood profile that there are in fact two local minima: one at approximately 0.35, and another at approximately 0.85. Now, look back at our estimate for $\psi^{21}$ generated using the default numerical optimization routine – we see that $\hat{\psi}^{21} = 0.342$, which is roughly where the first local (non-global) minima occurs. In this case, it was a case of ‘bad luck’ that \textsc{Mark} converged to this local minima – the bad luck owing to the starting values \textsc{Mark} defaults to.

If we use a starting value of 0.85 for $\psi^{21}$, and try again, we see that now \textsc{Mark} ‘correctly’ gives us the ‘right’ parameter estimates:

\begin{table}
\centering
\begin{tabular}{lllll}
\toprule
\textbf{Parameter} & \textbf{Estimate} & \textbf{Standard Error} & \textbf{95\% Confidence Interval} \\
\midrule
1: S 1:1 & 1.0000000 & 0.0000000 & 1.0000000 & 1.0000000 \\
2: p 1:1 & 0.5833333 & 0.3352797 & 0.5123869 & 0.6502884 \\
3: P5 1 to 2 & 0.5832909 & 0.0735735 & 0.4501940 & 0.7300718 \\
4: P5 2 to 1 & 0.8441960 & 0.0705638 & 0.6343620 & 0.9394203 \\
\bottomrule
\end{tabular}
\end{table}

Of course, in practice, we won’t be able to ‘cheat’ by using starting values close to the true values – since we won’t know what the true parameter values are (obviously)!
Moreover, for this particular problem, it turns out the estimates are also strongly influenced by the choice of the link function. In the preceding, we used the default sin link. What happens if instead we’d selected the logit link? In fact, for these example data, parameter estimates using the logit link (below) are very close to the true parameter values under which they data were simulated.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>95% Confidence Interval Lower</th>
<th>95% Confidence Interval Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:S 3:1</td>
<td>1.0000000</td>
<td>0.0000000</td>
<td>1.0000000</td>
<td>1.0000000</td>
</tr>
<tr>
<td>2:p 1:1</td>
<td>0.5833333</td>
<td>0.0355797</td>
<td>0.512869</td>
<td>0.650964</td>
</tr>
<tr>
<td>3:ps1 1 to 2</td>
<td>0.3993209</td>
<td>0.0738209</td>
<td>0.2501839</td>
<td>0.5484529</td>
</tr>
<tr>
<td>4:ps1 2 to 1</td>
<td>0.8441960</td>
<td>0.0709638</td>
<td>0.6543620</td>
<td>0.9394203</td>
</tr>
</tbody>
</table>

So, estimates for MS models may depend on choices of starting values, and link functions. This is clearly disconcerting. What can we do?

Well, if you’re willing to get your computer to do a bit of work for you, you can either (i) try a variety of different starting values and/or link functions, and see if there is convergence in your answers, or (ii) make use of the alternate optimization capability in MARK – based on simulated annealing. Recall that, the simulated annealing algorithm makes a periodic ‘random jump’ to a new parameter value (i.e., jumps to a different part of the likelihood surface). It is this characteristic that allows the annealing algorithm to be more likely to find the global maximum instead of a local maximum.

For our example, when we run the simulated annealing algorithm, we see that MARK gives us the correct estimates – even using the default starting values:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>95% Confidence Interval Lower</th>
<th>95% Confidence Interval Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:S 3:1</td>
<td>1.0000000</td>
<td>0.2748396-06</td>
<td>0.9999905</td>
<td>1.0000005</td>
</tr>
<tr>
<td>2:p 1:1</td>
<td>0.5833333</td>
<td>0.0355796</td>
<td>0.512869</td>
<td>0.650964</td>
</tr>
<tr>
<td>3:ps1 1 to 2</td>
<td>0.3993209</td>
<td>0.0738209</td>
<td>0.2501839</td>
<td>0.5484529</td>
</tr>
<tr>
<td>4:ps1 2 to 1</td>
<td>0.8441960</td>
<td>0.0709638</td>
<td>0.6543620</td>
<td>0.9394203</td>
</tr>
</tbody>
</table>

However, as noted earlier, simulated annealing is typically much slower than the standard numerical optimization routine MARK uses. But, if you have a strong suspicion that one or more of your parameter estimates are at the boundary, then there is some chance you might have a local minima (or several, especially if you have > 2 states), and simulated annealing might be your best option.

Is there any way you ‘predict’ when you might have such local minima? Unfortunately, there is no known simple diagnostic you can apply a priori (although there does seem to be some evidence that such local minima are more likely to occur for time-dependent models with > 2 states).

However, by making use of a numerically intensive approach known as Markov chain Monte Carlo (MCMC), we can evaluate the posterior distribution to determine whether or not there may be local minima in the likelihood for a given parameter (if the notion of MCMC, and posterior distributions, are new to you, no need to worry – these are covered in detail in Appendix E). Here, we will simply demonstrate the mechanics of using MCMC in MARK, using the preceding example where we have already determined there to be local minima in the likelihood.

To use MCMC estimation in MARK you first need to check the ‘MCMC Estimation’ check-box in the ‘numerical estimation run’ window:
Once you’ve clicked ‘OK to run’, you’ll be prompted to specify the MCMC parameters:

For the moment, we’ll simply accept the defaults (the details of the various MCMC parameters are covered in Appendix E) – these are usually sufficient to give us ‘a reasonable’ look at the posterior, from which we can often determine the presence of local minima in the likelihood for a given parameter. Note that the default file where the MCMC samples will be stored is the file MCMC.BIN (we need these samples to construct the posterior). The MCMC.BIN file is created in whatever directory contains, the .INP, .DBF, and .FPT files associated with the data set you’re working with.

Once you click the ‘OK’ button, MARK will spawn a numerical estimation window – you can ‘watch’ as MARK iterates through the MCMC samples (where the number of iterations is equal to the number of burn in samples (1,000, by default), plus the number of ‘tuning’ samples (4,000, by default), plus the number of post-burn and post-tuning samples (10,000, by default) – so, accepting the default parameters, 15,000 total samples (iterations of the Markov chain).

Once MARK has finished, it will spawn a window showing various summary statistics calculated from the posterior distribution for each of your parameters. These summary statistics (shown at the top of the next page) are presented for both the parameter estimates on the transformed scale (note: for MCMC estimation in MARK, the default settings for the priors assume you’re using the logit link. If you change to another link function, you’ll probably want to modify the priors as well), followed by the same summary statistics for the parameters transformed back to the real probability scale. The summary statistics for the 4 parameters for our example problem are shown at the top of the next page.

Of particular interest are the mean, median and modes of the posterior distributions, and the 2.5th and 97.5th percentiles of the posterior (from which we derive the 95% ‘credibility interval’ for our various parameters). Look closely at the mean, mode, and median statistics for the various parameters. Start with the encounter parameter $\psi^{12}$ (we ignore survival $S$, here, since it was fixed to 1.0). Note that for $p$, the mean, median and mode are all very close to each other.

However, this is obviously not the case for parameters $\psi^{12}$ and $\psi^{21}$. For $\psi^{12}$, for example, the mean is 0.535, the median is 0.572, and the mode is 0.600. Similarly, for $\psi^{21}$, the mean is 0.737, the median is 0.803, and the mode is 0.834. Recall that the ‘true’ parameter values were $\psi^{12} = 0.60$, and $\psi^{21} = 0.85$. 

Chapter 10. Multi-state models...
10.4.3. A more complex example – recruitment probability

Clearly, both the mean and median are not particularly robust estimators of either parameter – in both cases, the mode is the better statistic.∗

What do these differences among mean, median and mode suggest? Well, simplistically, they ‘hint’ at the possibility there may be ‘problems’ with the likelihood – specifically, multiple local minima. We can confirm our suspicion by considering two different ‘plots’ derived from the MCMC samples. First, consider a frequency histogram plotting the frequency (expressed as a percentage of the total) with which a particular parameter value was ‘visited’ during the iteration of the Markov chain.

Here is the frequency histogram for the posterior chain for $\psi^{12}$ (note: for a variety of reasons, the chain was ‘thinned’ by extracting every tenth value, prior to deriving the frequency histograms. This is commonly done to minimize the effects of serial autocorrelation in the Markov chains, although there is some debate as to whether or not this is really necessary).

∗ But we know this only because here we know the true parameter values. In fact, what we’ve just demonstrated is that MCMC is not necessarily a solution to in deriving point parameter estimates.
Here is the equivalent histogram for $\psi^{21}$:

![Histogram for $\psi^{21}$](image)

We see clearly that the posterior distributions for both $\psi^{12}$ and $\psi^{21}$ are bimodal. These plots help us understand why MARK converged on the local minimum. For example, note that the smaller, left-hand modal point for $\psi^{21}$ occurs at approximately 0.34 – 0.35, which is roughly what MARK reported (incorrectly) as the MLE for this parameter. The default starting value used by MARK (0.5) was closer to this local minimum than the global minimum for $\psi^{21}$ (which occurs at 0.85). The same explanation holds for $\psi^{12}$ as well. So, multi-modal frequency distributions of ‘visits by the Markov chain’ to parts of the parameter space indicates local minima, which MARK may (or may not) converge to (depending on the relative proximity of the starting value to a local minima).

We can also find evidence for local minima by examining the time-series of iterations of the Markov chain. Here is a plot of part of the time-series (first 5,000 iterations after burn-in and tuning) for $\psi^{12}$:

![Markov chain time-series for $\psi^{12}$](image)

We see clearly that the chain periodically ‘jumps’ to a region of the parameter space between 0.1 and 0.4, corresponding to the smaller, left-hand mode shown on the frequency histogram for $\psi^{12}$. 

---

**Chapter 10. Multi-state models...**
10.4.3. A more complex example – recruitment probability

Contrast this Markov chain with that generated for parameter $p$

For $p$, we see that the Markov chain is very ‘tight’ – indicating that the parameter will be estimated with good precision. In fact, the profile likelihood CI for $p$, estimated using the simulated annealing approach to estimating the likelihood, shows the 95% CI for $p$ to be $[0.513, -0.652]$, which is remarkably close to the MCMC-based 95% ‘credibility interval’, $[0.511, -0.648]$.

From the preceding, we can draw several conclusions:

1. > 1 minima in the likelihood for a given parameter are possible for multi-state models, especially for fully time-dependent models where there are numerous states. This can cause problems, if MARK converges on one of these local minima (thus yielding the ‘wrong’ estimate for that parameter). MARK will give you no warning when this occurs.

2. you can often test for the possibility of local minima by examining the posterior distribution for a given parameter, generated using the Markov Chain Monte Carlo (MCMC) option in MARK. Doing so for the various parameters in your general model may be a good first step prior to fitting other models. But, be warned – MCMC estimation on all of the parameters of a ‘large’ general model (i.e., time-dependence, many states) can take a long time. Moreover, there are a number of options in using the MCMC estimation routines in MARK that you may have to ‘tweak’ in the process (these are described in Appendix E), adding to the overall time it might take to fully explore the posterior distributions for various parameters.

3. an alternative approach is to either (i) try various different starting values – if they all result in convergence to the same parameter values, then you may be fortunate and not have local minima to concern yourself with. Or, alternatively, you could (ii) use the optimization routines based on simulated annealing, which are very likely to converge – eventually – to the true local minima. Again – we emphasize the word ‘eventually’, since simulated annealing can take a long time to converge.

end sidebar
10.5. GOF testing and multi-state models

What about GOF (goodness of fit)? By now, you should realize that one of the first steps in any analysis is assessing the fit of the most general model in your candidate model set to the data (introduced in Chapter 5). As part of this process, we make an estimate of $\hat{\epsilon}$ (a measure of the lack of fit of the model to the data), and use this $\hat{\epsilon}$ to ‘adjust’ the criterion we use for model selection. Here, we will describe 2 approaches to GOF testing for multi-state models.

10.5.1. Program U-CARE and GOF for time-dependent multi-state models

A few years ago, Roger Pradel and colleagues (Pradel et al. 2003) have described a method for assessing the fit of a fully time-dependent MS model to data. Although the specific details are somewhat complex, the rationale for the tests proposed by Pradel and colleagues are very similar to those underlying program RELEASE, as applied to ‘normal’ CJS data: the proposed tests essentially consider the fates of individuals seen before, or not, and whether (and when) they are seen again – but, in this case, conditional on the state in which they were previously observed.

The GOF test proposed by Pradel et al. is relevant for what they refer to as the ‘Arnason-Schwarz’ (AS) model (the AS model is in fact what we’ve been discussing in this chapter). To test the GOF of the AS model to multi-state data, Pradel et al. first describe a fully efficient goodness of fit test to what they refer to as a ‘Jolly Move’ model (JMV). Any fully efficient GOF test is based on the property that all animals present at any given time behave in the same way. This is the basic point underlying RELEASE: in Test 3, we test the assumption of ‘equivalent behaviors’ whatever their past capture history, while in Test 2, we test the ‘equivalence’ assumption whether they are currently captured or not. The same logic underlies the GOF test for the JMV model.

What is this JMV model, and how does it relate to the AS model? The JMV model (Brownie et al. 1993) differs from the typical AS model in that it permits the capture probability for time $(i+1)$ to depend on the state at periods $(i)$ and $(i+1)$ (whereas the AS model permits the encounter probability to depend only on current state, and time). Thus, the AS model is in fact a special (reduced) case of the more general JMV model. As with program RELEASE, a fully efficient GOF test for the JMV model is based on the property that all animals present at any given time on the same site behave the same.

Pradel et al. introduce 2 general tests of this multi-state ‘equivalence’ assumption: Test 3G, which assumes ‘behavioral equivalence’ of individuals released together regardless of their past capture history, and Test M, which tests ‘equivalence’ among those individuals that are eventually recaptured (on a subsequent occasion) conditional on whether or not they are encountered at the present occasion. (There are also 2 possible subcomponents for testing for transience, and memory models, but we will not discuss those here). See Pradel et al. for full details – for now, we’ll focus on the basic ideas, and the mechanics.

The first step in the GOF test proposed by Pradel et al. is to assess the fit of the fully time-dependent JMV model to the data. In many cases, the JMV model is unlikely to show significantly greater fit to the data than the AS model (since the dependence of the capture probability at time $(i+1)$ to depend on both $(i)$ and $(i+1)$ is unlikely to be observed in practice very often). As such, the GOF test for the JMV model may by a generally valid test of fit for the AS model as well. Here’s how you implement a GOF test to the general JMV model. First, you need a recent build of the program U-CARE (first described in Chapter 5). Then, you might need to modify your MARK input file – slightly. The only thing you need to do is make sure that all of your ‘state coding’ is numeric (e.g., if you use ‘B’ and ‘N’ in your input file, for example, you’ll need to change them to numbers, say 1 for ‘N’, and 2 for ‘B’...U-CARE cannot currently handle letters for state coding in the input file).
10.5.2. MS models and the median $\hat{\epsilon}$ test

We’ll demonstrate the use of U-CARE for multi-state GOF testing using simulated data (MS_GOF.INP). These data, consisting of 2 states, 8 occasions, were simulated under the true model \( \{ S_{t+1} P_{t+1} \psi \} \) – so additive time variation between the two states for survival, and encounter probability, but constant state differences in transition probability over time. We start U-CARE, and read in the input file. U-CARE will ask you to confirm a few things about the file (e.g., presence or absence of covariates). Once you’ve answered those questions, all you need to do is pull down the ‘Goodness-of-Fit for Multi-state’ menu, and select ‘Sum of multi-site tests’. Selecting this option will cause U-CARE to fit the component tests (3G and M) and the JMV to the data. Once finished, U-CARE will spawn another window, giving the results of the various tests.

We see that U-CARE reports the 2 individual component tests (3G and M). For these simulated data, U-CARE reports that both tests are accepted (Test 3G: \( \chi^2 = 33.464, df=25, P = 0.120 \), Test M: \( \chi^2 = 5.838, df=10, P = 0.829 \)).

The next line is the key. It reports the overall test of the JMV model to the data (basically, the sum of the 2 component tests 3G and M). In this case, U-CARE reports that the \( \chi^2 = 50.252 \), with df=50. The P-value is 0.463. Thus, our estimate of \( \hat{\epsilon} \) would be 50.252/50 = 1.00, which is very close to 1.0.

10.5.2. MS models and the median $\hat{\epsilon}$ test

What about the median $\hat{\epsilon}$ test introduced in Chapter 5 – can it be used for MS data? Yes, although there is limited experience with it to date. For purposes of comparison with the results from U-CARE, start MARK, and run the fully time-dependent model on these same, simulated data. Then, run the median $\hat{\epsilon}$ test – since we know these data are simulated under a reduced parameter AS model, we’ll save ourselves some time and use lower bound of 1.0, and an upper bound of 2.5 for our analysis (if you don’t remember the details of the median $\hat{\epsilon}$ GOF test, go back and look at Chapter 5). We’ll use 5 design points, with 10 replicates at each point. Using these values, MARK reported an estimate of $\hat{\epsilon}$ of 0.99, which is effectively 1.0. Again, although there is little experience to date with the median $\hat{\epsilon}$ and GOF testing of MS models, preliminary results look promising. For the moment, we suggest using U-CARE for GOF testing, especially if your general model is the fully time-dependent model. For reduced parameter general models, it is probably worth trying the median $\hat{\epsilon}$ approach.

However, be advised that running a median-$\hat{\epsilon}$ test for multi-state models with a large number of occasions, and states, can take a lot of computer time (especially if you decide to use the simulated annealing algorithm – as described earlier in this chapter). For fully time-dependent models, U-CARE is much faster. However, if your most general model which adequately fits the data is not time-dependent, then your only real option is to run the median-$\hat{\epsilon}$ GOF test. The good news (relatively speaking) is that you need only assess GOF for the most general model in your candidate model set – so you need only to go to the trouble once.
10.6. multi-state models & unequal time intervals

Various data types in MARK have state transitions – clearly, the multi-state data type that is presented in this chapter, but they also arise in robust designs (Chapter 15), Barker models (Chapter 9), and multi-season occupancy models. Any data type with state transitions suffers from the same problem when the intervals between occasions are unequal (how MARK handles unequal intervals in general was introduced earlier in Chapter 4).

To illustrate the issue, consider the case where an encounter occasion is missing in the multi-state data type. Consider the following valid MARK 5-occasion multi-state encounter history ‘A.A/zero.alt2/zero.alt2’, where the missing occasion is shown as a ‘dot’ and there are 2 states, A and B, and occasions are all 1 time unit apart. To explain this ‘dot’, several possibilities exist, namely:

\[ S_A^A \psi_1^{AA} (1 - p_1^A) S_2^A \psi_1^{AA} p_3^A \ldots \quad \text{and} \quad S_1^A \psi_1^{AB} (1 - p_2^B) S_2^B \psi_2^{BA} p_3^A \ldots \]

However, suppose that you coded the data with the ‘dot’ left out, and set the time intervals to 2, 1, and 1. That is, only 4 occasions are considered instead of 5. So the encounter history is now ‘A.A00’. Unfortunately, this approach is going to give very different results from the proper parametrization above. MARK does not generate the probabilities for the transition to state B with this parametrization. The probability of surviving from occasion 1 to occasion 2 would now be \((S_1^A)^2\), with no consideration that the animal could have moved to state B during the missing occasion. So, even the survival estimates \(S\) will be incorrect. The \(\psi\) parameters for the first interval are not comparable to the \(\psi\) parameters for the second and third intervals because they represent different time scales.

Internally, within MARK, the time interval correction on \(S\) remains, but all time interval corrections from \(\psi\) have been removed. The motivating logic is that when time intervals are ‘ragged’, e.g., 1.1, 0.9, 1.05, 0.95, it may still make sense to apply a correction to \(S\). However, this correction is inappropriate for \(\psi\), and may even be questionable for \(S\). So, ‘user beware’!

10.7. Summary

In this chapter, we have considerably expanded the range of ‘underlying’ models we can fit to mark-recapture data – in particular, we’ve added a ‘movement’ parameter. We have also seen how to apply constraints to these models as easily as we did with CJS models. In fact, we’ve seen how we can apply movement models to other data types (live encounters, dead recoveries...), which highlights one of the singular strengths of MS models – the ability to combine sources of information in a natural and relatively intuitive framework.

10.8. References

Biological, and Environmental Statistics, 10, 184-196.


