CHAPTER 5

Goodness of fit testing...

In the preceding chapter, we took our first look at a fundamental topic – comparing models. In particular, we considered the different 'paradigms' of AIC, or LRT, as tools to allow us to robustly 'select' among a candidate set of models. More generally, however, these approaches both rest fundamentally on issues of fit – how well does a particular model fit the data. This is a very important consideration, regardless of which 'flavor' of model selection you prefer – both AIC comparisons, and LRT, require assessment of model fit as part of the process.

In this chapter we will provide a brief introduction to this very important topic – goodness of fit testing (GOF). All of the models and approaches we have discussed so far make very specific assumptions (concerning model fit) that must be tested before using **MARK**. Thus, as a first step, you need to confirm that your starting (general) model adequately fits the data, using GOF tests. We will make frequent reference to this, directly or indirectly, at several points throughout the book.

There are a number of ways in which GOF testing can be accomplished, and a variety of GOF procedures have been implemented in several different CMR software applications. For example, programs **RELEASE**, **SURVIV**, **JOLLY**, and **JOLLYAGE** all provide GOF statistics for various models. Some applications do not provide any 'built-in' GOF testing. As a starting point, we will assert that there are two primary purposes for GOF testing.

The first, which we've already noted, is that it is a necessary first step to insure that the most general model (i.e., the model with the most parameters) in your candidate model set (see Chapter 4) adequately fits the data. Comparing the relative fit of a general model with a reduced parameter model provides good inference only if the more general model adequately fits the data.

However, suppose you construct a candidate model set, based on *a priori* hypotheses concerning the data in hand. This model set contains at least one 'general' model, containing enough parameter structure so that, you believe, it will fit the data. Suppose however, it does not – suppose you have a means of assessing GOF, and that based on this 'test' you determine that the general model does not adequately fit the data. What next?

Well, in addition to providing a simple 'yes/no' criterion for fit, the GOF testing procedure can in itself reveal interesting things about your data. While significant lack of fit of your general model to your data is in some senses a nuisance (since it means you need to carefully reconsider your candidate model set), in fact the lack of fit forces you to look at, and think about, your data more carefully than you might have otherwise – the key question becomes – *why* doesn't the model fit the data? The answers to this question can sometimes be extremely valuable in understanding your problem.

What do we mean by 'lack of fit'? Specifically, we mean that the arrangement of the data do not meet the expectations determined by the assumptions underlying the model. In the context of simple

mark-recapture, these assumptions, sometimes known as the 'CJS assumptions' are:

1. every marked animal present in the population at time (i) has the same probability of recapture (p_i)
2. every marked animal in the population immediately after time (i) has the same probability of surviving to time ($i + 1$)
3. marks are not lost or missed.
4. all samples are instantaneous, relative to the interval between occasion (i) and ($i + 1$), and each release is made immediately after the sample.

We will generally assume that assumptions 3 and 4 are met (although we note that this is not always a reasonable assumption. For example, neck collars, commonly used in studies of large waterfowl, have a significant tendency to 'fall off' over time). It is assumptions 1 and 2 which are typically the most important in terms of GOF testing.

In this chapter, we will look at GOF testing in two ways. First, we shall discuss how to do basic GOF testing in program **MARK**, using a parametric bootstrapping approach. Then, we show how to use program **MARK** to call another, vintage program (**RELEASE**) to more fully explore potential reasons for lack of fit for the CJS model only. Then, we introduce two newer approaches to estimating lack of fit. We finish by discussing how to accommodate lack of fit in your analyses.

5.1. Conceptual motivation – 'c-hat' ($\hat{c}$)

GOF testing is a diagnostic procedure for testing the assumptions underlying the model(s) we are trying to fit to the data. To accommodate (adjust for, correct for...) lack of fit, we first need some measure of how much extra binomial 'noise' (variation) we have. The magnitude of this overdispersion cannot be derived directly from the various significance tests that are available for GOF testing, and as such, we need to come up with some way to quantify the amount of overdispersion. This measure is known as a variance inflation factor, or $\hat{c}$ (phonetically, 'c-hat').

We start by introducing the concept of a *saturated model*. The saturated model is loosely defined as the model where the number of parameters equals the number of data points or data structures. As such, the fit of the model to the data is effectively 'perfect' (or, as good as it's going to get).

begin sidebar

saturated models in MARK

In the following, the method used to compute the saturated model likelihood is described for each type of data. This is the method used when no individual covariates are included in the analysis. Individual covariates cause a different method to be used for any data type.

Live Encounters Model. For the live encounters model (Cormack-Jolly-Seber model), the encounter histories within each attribute group are treated as a multinomial. Given n animals are released on occasion i, then the number observed for encounter history j [n_j] divided by n is the parameter estimate for the history. The $-2\ln(\mathcal{L})$ for the saturated model is computed as the sum of all groups and encounter histories. For each encounter history, the quantity $(n_j \times \ln[n_j/n])$ is computed, and then these values are summed across all encounter histories and groups.

Dead Encounters Model – Brownie. The method used is identical to the live encounters model. For this type of data, the saturated model can be calculated by specifying a different value in every PIM entry. The resulting $-2\ln(\mathcal{L})$ value for this model should be identical to the saturated model value.

Dead Encounters Model – Seber. For dead encounters models with S and f coding. The saturated model for this data type is the same as the usual (Brownie) dead encounters model.

Joint Live and Dead Encounters Model. The method used is identical to the live encounters model.

Known Fate Model. The known fate data type uses the (`group` × `time`) model as the saturated model. For each occasion and each group, the number of animals alive at the end of the interval divided by the number of animals alive at the start of the interval is the parameter estimate. The $-2\ln(\mathcal{L})$ value for the saturated model is the same as the $-2\ln(\mathcal{L})$ value computed for the (`group` × `time`) model.

Closed Captures Model. The saturated model for this type of data includes an additional term over the live encounters model, which is the term for the binomial coefficient portion of the likelihood for $\hat{N}$. For the saturated model, $\hat{N}$ is the number of animals known to be alive $[M_{t+1}]$, so the log of $\hat{N}!$ factorial is added to the likelihood for each group.

Robust Design Model. The saturated model for this data type is the same as the closed captures model, but each closed-captures trapping session contributes to the log likelihood.

Multi-strata Model. The saturated model for this data type is the same as for the live encounters model.

BTO Ring Recovery Model. The saturated model for this data type is the same as for the live encounters data.

Joint Live and Dead Encounters, Barker's Model. The method used is identical to the live encounters model.

Pradel and Link-Barker Models. These models assume that an animal can enter the study on any occasion, so the saturated model is computed with the parameter estimate as the number of animals with the encounter history divided by the total number of animals encountered. The same procedure is used for the Burnham Jolly-Seber model and the POPAN model, but because these data types include population size, complications result.

All Data Types with Individual Covariates. For any of the models with individual covariates, the sample size for each encounter history is 1. The saturated model then has a $-2\ln(\mathcal{L})$ value of zero. The deviance for any model with individual covariates is then just its $-2\ln(\mathcal{L})$ value.

end sidebar

Consider the following example of a logistic regression of some medical data. Suppose there is a sample of male and female cardiac patients. Interest is focussed on whether or not the amplitude (high or low) of a particular waveform in an electrocardiograph test (EKG) was a good predictor of cardiac disease (`0` = no disease, `1` = disease), and whether the predictions were influenced by the gender of the patient. Here are the data, presented as a frequency table:

female

	EKG	
disease	h	l
0	10	15
1	16	9

male

	EKG	
disease	h	l
0	12	11
1	9	17

If we use both `sex` and `EKG` and their interaction in the model, we will use up all the degrees of freedom. That is, we are fitting each cell in the contingency table with its own parameter, which constitutes a saturated model:

$$\texttt{disease} = \texttt{sex} + \texttt{EKG} + (\texttt{sex} \times \texttt{EKG})$$

If we fit this model to the data (using the logistic regression program from your favorite statistics package), $-2\ln(\mathcal{L}) = 132.604$. The AIC for this model is 140.604 ($132.604 + [2 \times 4] = 140.604$). Remember

– the saturated model is the model where the model structure is saturated with respect to the data. In other words, it is sufficiently parameterized that every data point is effectively encompassed by the model. As such, the likelihood for the saturated model is as small as it's ever going to get.

Now, in this case, the parameter values for the terms in the saturated model are all estimable. This will not generally be the case. Moreover, in many cases, the saturated model is not a plausible, or interesting general starting model. For the moment, let's pretend that is the case here. Suppose that instead of the saturated linear model proposed above, our general starting model is

$$\texttt{disease} = \texttt{sex} + \texttt{EKG}$$

If we fit this model to the data, the model likelihood is $-2\ln(\mathcal{L}) = 136.939$, with an AIC of 142.939. As expected, the fit isn't as good as the saturated model. But, the point of interest here is – how different is the fit? The numerical difference between the likelihood for the saturated model and the general model is $(136.939 - 132.604) = 4.335$. The difference in the degrees of freedom (number of estimable parameters) between the two models is 1 (the interaction term).

Now, for the **key conceptual step** – the difference in fit (deviance) between the saturated model and any other model (in this case, the general model in the candidate model set) is asymptotically χ^2 distributed (at least, in theory). In **MARK**, the *deviance* (as reported in the results browser) is defined as the difference in $-2\ln(\mathcal{L})$ between the current model and the saturated model. For our example analysis, $\chi^2_1 = 4.335$ is marginally significant ($P = 0.0373$) based on a nominal $\alpha = 0.05$ level. This suggests that the general model does not quite adequately fit the data.

So, why is this important? Well, suppose we didn't know that the general model in our model set has some lack of fit to the data (relative to the saturated model), and proceeded to compare the general model with a reduced parameter model

$$\texttt{disease} = \texttt{EKG}$$

In other words, we're comparing

$$\begin{array}{rl} & \texttt{disease} = \texttt{SEX} + \texttt{EKG} + \texttt{error} \\ \textit{versus} & \underline{\texttt{disease} = \qquad \texttt{EKG} + \texttt{error}} \\ & \qquad\qquad \texttt{SEX} \end{array}$$

which amounts to a test of the importance of SEX in determining the presence or absence of the cardiac disease. The likelihood for the reduced model (disease = EKG) is $-2\ln(\mathcal{L}) = 137.052$, with an AIC of 141.052. If we use a LRT to compare the fits of the general and reduced models, we get a test statistic of $\chi^2_1 = (137.052 - 136.939) = 0.0113$, which is clearly not 'significant' by usual statistical standards.

However, in making this comparison we've ignored the fact that our general model has marginally significant lack of fit to the data (relative to the saturated model). Does this make a difference in our analysis? In fact, the generally accepted approach to this would be to 'adjust' (correct) the likelihood of both the general model and the reduced model to account for the lack of fit between the general and saturated models.

For a correctly specified model, the χ^2 statistic (or the deviance) divided by the degrees of freedom, should be approximately equal to one. When their values are much larger than one, the assumption of simple binomial variability may not be valid and the data are said to exhibit *overdispersion*. *Underdispersion*, which results in the ratios being less than one, occurs less often in practice.

The most common approach to correcting for overdispersion in linear models is to multiply the covariance matrix by a dispersion parameter (*note*: this approach is most robust when the sample sizes

in each subpopulation in the analysis are roughly equal). In other words, we use a function of the lack of fit (typically, some function of the χ^2/df for the general model), to adjust the fit of the general and all other models in the candidate model set. For our present example, applying a χ^2/df 'correction' yields $-2\ln(\mathcal{L}) = 31.590$ for the general model, and $-2\ln(\mathcal{L}) = 31.616$ for the reduced model.

Do we need to modify the LRT in any way? In fact, the LRT, which is normally a χ^2 test between two models, is transformed into an F-test, with (df_{LRT},df_{model}) degrees of freedom:

$$F = \frac{\left(\chi^2_{LRT}/df_{LRT}\right)}{\hat{c}},$$

where

$$\hat{c} \approx \frac{\chi^2}{df} = 1.$$

For this example, no big difference in the subsequent LRT between the two models.

What about the AIC approach? Well, recall from Chapter 4 that the sample-size corrected AIC_c is estimated as

$$AIC_c = -2\ln\left(\mathcal{L}(\hat{\theta})\right) + 2K + \left(\frac{2K(K+1)}{n-K-1}\right).$$

Do we need to adjust the AIC_c for lack of fit between the general model and the saturated model? Perhaps given the preceding discussion it is not surprising that the answer is 'yes'. We have to adjust the likelihood term, yielding the *quasi-likelihood adjusted* AIC, the $QAIC_c$

$$QAIC_c = \frac{-2\ln(\mathcal{L})}{\hat{c}} + 2K + \left(\frac{2K(K+1)}{n-K-1}\right),$$

where $\hat{c}$ is the measure of the lack of fit between the general and saturated models.*

Now, since

$$\hat{c} \approx \frac{\chi^2}{df} = 1,$$

for a saturated model, then as the general model gets 'further away' from the saturated model, $\hat{c} > 1$. If $\hat{c} = 1$, then the expression for $QAIC_c$ reduces back to AIC_c (since the denominator for the likelihood term disappears). If $\hat{c} > 1$, then the contribution to the $QAIC_c$ value from the model likelihood will decline, and thus the relative penalty for a given number of parameters K will increase. Thus, as $\hat{c}$ increases, the $QAIC_c$ tends to increasingly favor models with fewer parameters.

begin sidebar

What if $\hat{c} < 1$?

What if $\hat{c} < 1$? In the preceding, we mention the case where $\hat{c} > 1$, indicating some degree of lack of fit, reflecting (in all likelihood) overdispersion in the data. Now, if instead, $\hat{c} < 1$, then we generally

* Some people feel that every model should have one additional parameter included if a value of c is estimated for the set of models. The is an option in **MARK** (under '**File | Preferences**') to automatically add 1 to K, the number of parameters estimated, for each model. However, the effect on model selection results is typically extremely small, and can result in errors in the value of K. Use at your own risk.

consider this as reflecting underdispersion. While the intuitive thing to do is to simply enter the $\hat{c}$ as estimated (discussed below), there is lack of unanimity on how to handle $\hat{c} < 1$. Some authors recommend using the estimated $\hat{c}$, regardless of whether or not it is > 1 or < 1. However, still others suggest that if $\hat{c} < 1$, then you should set $\hat{c} = 1$ (i.e., make no adjustment to various metrics). For the moment, the jury is out – all we can recommend at this stage is – if $\hat{c} > 1$, then adjust. If $\hat{c} < 1$, then set $\hat{c} = 1$, and 'hold your nose'.

end sidebar

5.2. The practical problem – estimating $\hat{c}$

In a (2002) paper, Gary White commented:

> *The Achilles' heel...in capture- recapture modeling is assessing goodness-of-fit (GOF). With the procedures presented by Burnham & Anderson (1998), quasi-likelihood approaches are used for model selection and for adjustments to the variance of the estimates to correct for over-dispersion of the capture-recapture data. An estimate of the over-dispersion parameter, c, is necessary to make these adjustments. However, no general, robust, procedures are currently available for estimating* c. *Although much of the goodness-of-fit literature concerns testing the hypothesis of lack of fit, I instead view the problem as estimation of c.*

So, the objective then becomes estimating the lack of fit of the model to our data. In other words, how to estimate $\hat{c}$? The general challenge of estimating $\hat{c}$ is the basis for a significant proportion of the remainder of this chapter.

As we'll see, there are a number of approaches that can be taken. The most obvious approach is to simply divide the model χ^2 statistic by the model degrees of freedom:

$$\hat{c} \cong \frac{\chi^2}{df}.$$

However, in many (most?) cases involving the sorts of multinomial data we analyze with **MARK**, this approach doesn't work particularly well. Although the distribution of the deviance between the general and saturated models is supposed to be asymptotically χ^2 distributed, for the type of data we're working with it frequently (perhaps generally) isn't because of sparseness in the frequency table of some proportion of the possible encounter histories. For example, for live encounter mark-recapture data, for the CJS model (Chapter 4), there are $[(2^n - 1) - 1]$ possible encounter histories for n occasions, and for typical data sets, many of the possible encounter histories are either rare or not observed at all. The asymptotic distribution of the deviance assumes that all encounter histories are sampled (which would be true if the sample were infinitely large, which is of course the underlying assumption of 'asymptopia' in the first place).

Given that the asymptotic assumptions are often (perhaps generally) violated for these sorts of data, alternative approaches are needed. Moreover, an estimate of χ^2 is not available for all models (in particular, for models where the saturated model is not defined), and there can be some non-trivial difficulties involved in the calculation of the χ^2 statistics, especially for sparse data sets. On the other hand, the advantage of using a χ^2 approach is that the frequency tabulations used in deriving the χ^2 statistic are often very useful in determining the 'sources' of lack of fit.

In the following we'll discuss two broadly different approaches for estimating $\hat{c}$. The first approach we'll describe, using program **RELEASE**, provides estimates of $\hat{c}$ for CJS live-encounter data using a

contingency table (i.e., χ^2) approach. However, this is not generalizable to other data types, so other approaches are required.

The second approach we'll discuss (the bootstrap, and median-$\hat{c}$) uses simulation and resampling to generate the estimate of $\hat{c}$. Rather than assuming that the distribution of the model deviance is in fact χ^2 distributed (since it generally isn't for typical '**MARK** data', as noted above), the bootstrap and median-$\hat{c}$ approaches generate the distribution of model deviances, given the data, and compare the observed value against this generated distribution. The disadvantage of the bootstrap and median-$\hat{c}$ approaches (beyond some technical issues) is that both merely estimate $\hat{c}$. While this is useful (in a practical sense), it reveals nothing about the underlying sources of lack of fit. In a similar vein, we'll also introduce an approach (the Fletcher-$\hat{c}$) that is somewhat similar to the approach based on program **RELEASE**, but is more general, and like the bootstrap and median-$\hat{c}$ approaches, provides no guidance as to the causes for the lack of fit.

Each approach has different strengths and weaknesses, so a good understanding of each of these procedures is important to assessing model fit using **MARK**.

5.3. Program RELEASE – details, details...

For testing the fit of the data to a fully-time-dependent CJS model, program **RELEASE** (introduced in the Burnham *et al*. 1987 monograph) has been the *de facto* standard approach for many years. In the following, we describe the use of **RELEASE** for GOF testing (and estimation of $\hat{c}$). We will discuss the use of **RELEASE** to generate specific GOF statistics, and give some broad suggestions for how to interpret lack-of-fit (from both a statistical and biological point of view), and what remedies are available. Note, **RELEASE** is primarily intended for standard live-recapture models, although it can be tweaked to handle some dead recovery models as well. While this may not be appropriate for your particular analysis (e.g., if you're working with telemetry, for example), there is still value in understanding how **RELEASE** works, since the principles underlying it are important for all analyses, not just standard live-recapture studies.

5.4. Program Release – TEST 2 & TEST 3

Program **RELEASE** generates 3 standard 'tests', which are given the absolutely uninformative names '**TEST 1**', '**TEST 2**', and '**TEST 3**'. The latter 2 tests, **TEST 2** and **TEST 3**, together provide the GOF statistics for the reference model (the time-dependent CJS model). **TEST 1** is an omnibus test that is generated **only** if you are comparing groups, and tests the following hypothesis under model $\{\varphi_{g*t}\, p_{g*t}\}$:

$\mathbf{H}_o$: all parameters φ_i and p_i have the same value across treatment groups (i.e., there is no difference in survival (φ_i) or recapture (p_i) considered simultaneously among groups).

$\mathbf{H}_a$: at least some values for either φ_i or p_i (or both) differ among groups.

The big advantage of using **MARK** or one of the other applications available for CMR analysis, is that you can separately model differences in either survival or recapture rate independently. **TEST 1** does not do this – it only tests for an 'overall' difference among groups. Since this severely limits its utility, we will not discuss use of **TEST 1**. In fact, we actively discourage it's use, since it is possible to do far more sophisticated analysis if you have capture histories from individually marked animals (although **TEST 1** may still be of use when complete capture histories are not available – see the 'blue book' for use of **RELEASE** and **TEST 1** under alternative capture protocols). While **TEST 1** may be of limited use, **TEST 2** and **TEST 3** together are quite useful for testing the GOF of the standard time-dependent CJS

(Cormack-Jolly-Seber) model to the data (this model was first presented in detail in Chapter 4).

What do we mean by 'lack of fit'? As noted previously, we mean that the arrangement of the data do not meet the expectations determined by the assumptions underlying the model. These assumptions, which we also noted earlier in this chapter, sometimes known as the CJS assumptions are:

1. Every marked animal present in the population at time (i) has the same probability of recapture (p_i)
2. Every marked animal in the population immediately after time (i) has the same probability of surviving to time ($i + 1$)
3. Marks are not lost or missed.
4. All samples are instantaneous, relative to the interval between occasion (i) and ($i + 1$), and each release is made immediately after the sample.

For now, we will assume that assumptions 3 and 4 are met. It is assumptions 1 and 2 which are typically the most important in terms of GOF testing. In fact, **TEST 2** and **TEST 3** in **RELEASE**, as well as the GOF tests in other software, directly test for violations of these two assumptions (in one form or another).

Let's expand somewhat on assumptions 1 and 2. Assumption 1 says that all marked animals in the population have the same chances of being captured at any time (i). What would be the basis for violating this assumption? Well, suppose that animals of a particular age or size are more (or less) likely to be captured than animals of a different age or size? Or, suppose that animals which go through the process of being captured at occasion (i) are more (or less) likely to be captured on a later occasion than animals who were marked at some other occasion? Or, what if some marked individuals temporarily leave the sample area (temporary emigration)? Or what if animals always exist in 'pairs'? For estimation of survival in open populations, marked animals have the same probability of recapture. For estimation of population size (abundance), both marked and unmarked animals must have the same probability of capture.

What about assumption 2? Assumption 2 says that, among the marked individuals in the population, all animals have the same probability of surviving, regardless of when they were marked. In other words, animals marked at occasion ($i-1$) have the same probability of surviving from (i) to ($i+1$) as do animals marked on occasion (i). When might this assumption be violated? One possibility is that individuals caught early in a study are more (or less) prone to mortality than individuals caught later in the study. Or, perhaps you are marking young individuals. An individual captured and marked as an offspring at ($i-1$) will be older, or larger, or possibly of a different breeding status, at occasion (i), while offspring marked at occasion (i) are just that, offspring. As such, the offspring marked at ($i-1$) may show different survival from (i) to ($i + 1$) than offspring marked at (i), since the former individuals are older, or larger, or somehow 'different' from individuals marked at (i).

For both **TEST 2** and **TEST 3** we have noted several reasons why either **TEST 2** or **TEST 3** might be violated. The examples noted are by no means an all-inclusive list – there are many other ways in which either or both tests could be violated. While violation of the underlying model assumptions has a specific statistical consequence (which we will deal with shortly), it may also serve to point out something interesting biologically. For example, suppose all animals are not equally likely to be captured at any occasion. We might ask 'why? Does this reveal something interesting about the biology?'.

We'll approach GOF testing in 2 steps. First, we'll describe the 'mechanics' of how to run **RELEASE** to generate the **TEST 2** and **TEST 3** results. Then, we'll discuss the mechanics of how these two tests are constructed, and how to interpret them.

5.4.1. Running RELEASE

Running **RELEASE** from within **MARK** is easy. Running it as a standalone application is also fairly straightforward – more on this in a moment. For now, we will restrict our discussion to running **RELEASE** from within **MARK**, although there may be a few instances where it may become necessary to run **RELEASE** as a stand-alone application.

To run **RELEASE** from within **MARK**, simply pull down the '**Tests**' menu, and select '**Program RELEASE GOF**'. This option is available only if you selected '**Recaptures**' as the data type. That's it. **RELEASE** will run, and the results will be output into a Notepad window.

At the top of this output file there will be some information concerning recent updates to the **RELEASE** program, and some statement concerning limits to the program (maximum number of groups, or occasions). Then, you will see a listing of the individual capture histories in your data set, plus a summary tabulation of these histories known as the *reduced m-array*. The *m*-array contains summary information concerning numbers of individuals released at each occasion, and when (and how many) of them were captured at subsequent occasions. The reduced *m*-array will be discussed in more detail later. These *m*-array tabulations are then followed by the **TEST 2** and **TEST 3** results for each group (respectively), followed in turn by the summary statistics.

TEST 2

TEST 2 deals only with those animals known to be alive between (i) and ($i+1$). This means we need individuals marked at or before occasion (i), and individuals captured at or later than ($i+1$). If they were alive at (i), and captured at or later than ($i+1$), then they must have been alive in the interval from occasion (i) to ($i+1$).

In other words, 'is the probability of being seen at occasion ($i+1$) a function of whether or not you were seen at occasion (i), given that you survived from (i) to ($i+1$)?'. Under assumption 1 of the CJS assumptions, all marked animals should be equally 'detectable' at occasion ($i+1$) independent of whether or not they were captured at occasion (i). **TEST2.C** has the following general form: of those marked individuals surviving from (i) to ($i+1$), some were seen at ($i+1$), while some were seen after ($i+1$). Of those not seen at ($i+1$), but seen later, does 'when' they were seen differ as a function of whether or not they were captured at occasion (i)?

In other words:

	when seen again?					
seen at (i)	($i+1$)	($i+2$)	($i+3$)	($i+4$)	...	($i+k$)
no	f	f	f	f	f	f
yes	f	f	f	f	f	f

So, **TEST2** asks 'of those marked animals not seen at ($i+1$), but known to be alive at ($i+1$) (since they were captured after $i+1$), does when they were next seen ($i+2$, $i+3$...) depend on whether or not they were seen at (i)?'. Again, we see that **TEST2.C** deals with capture heterogeneity. For most data sets, pooling results in a (2×2) matrix.

TEST2 (in general) is sensitive to short-term capture effects, or non-random temporary emigration. It highlights failure of the homogeneity assumption (assumption 1), among animals and between occasions. In practice, **TEST 2** is perhaps most useful for testing the basic assumption of 'equal catchability' of marked animals. In other words, we might loosely refer to **TEST 2** as the 'recapture test'.

TEST 3

In general, **TEST 3** tests the assumption that all marked animals alive at (i) have the same probability of surviving to (i + 1) – the second CJS assumption.

TEST 3 asks: 'of those individuals seen at occasion (i), how many were ever seen again, and when?'. Some of the individuals seen at occasion (i) were seen for the first time at that occasion, while others had been previously seen (marked). Does whether or not they were ever seen again depend on this conditioning? The first part of **TEST 3**, known as **TEST3.SR**, is shown in the following contingency table:

seen before (i)	**seen again**	**not seen again**
no	f	f
yes	f	f

In other words, does the probability that an individual known to be alive at occasion (i) is ever seen again depend on whether it was marked at or before occasion (i)? If there is only a single release cohort, then 'seen before i?' becomes 'seen before i, excluding initial release?'.

TEST3.SR is what is presented for **TEST 3** in the version of **RELEASE** bundled with **MARK**. There is also a **TEST3.Sm**, which asks '...among those animals seen again, does **when** they were seen depend on whether they were marked on or before occasion (i)?'. **TEST3.Sm** is depicted in the following contingency table:

	when seen again?					
seen before (i)	(i + 1)	(i + 2)	(i + 3)	(i + 4)	...	(i + k)
no	f	f	f	f	f	f
yes	f	f	f	f	f	f

If there is only a single release cohort, then 'seen before i?' become 'seen before i, excluding initial release?'. So, in a very loose sense, **TEST 2** deals with 'recapture problems', while **TEST 3** deals with 'survival problems' (although there is no formal reason to make this distinction – it is motivated by our practical experience using **RELEASE**). If you think about it, these tables should make some intuitive sense: if assumptions 1 and 2 are met, then there should be no difference among individuals if or when they were next seen conditional on whether or not they were seen on or before occasion (i).

Let's consider a simple example of GOF testing with **RELEASE**. We simulated a small data set – 6 sampling occasions, 350 newly marked individuals released alive at each occasion. First, let's look at something call the *reduced m-array* table **RELEASE** generates as the default (the other m-array presentation you can generate running **RELEASE** as a stand-alone application is the *full m-array* – this will be discussed later). Examination of the m-array (in either form) will give you some idea as to 'where the numbers come from' in the **TEST 2** and **TEST 3** contingency tables.

The reduced m-array for the simulated data is shown at the top of the next page. The main elements of interest are the $\mathbf{R}_i$, $m_{i,j}$, and r_i values. The $\mathbf{R}_i$ values are the number of individuals in total released on each occasion. For example, $\mathbf{R}_1 = 350$ equals 350 individuals released on the first occasion – all newly marked. At the second occasion ($\mathbf{R}_2$), we released a total of 428 individuals – 350 newly marked individuals, plus 78 individuals from the first release which were captured alive at the second occasion. The $m_{i,j}$ values are the number of individuals from a given release event which are captured for the first

Observed Recaptures for Group 1
Group 1

i	R(i)	m(i,j)					r(i)
		j= 2	3	4	5	6	
1	350	78	41	26	12	0	157
2	428		170	92	36	4	302
3	561			269	99	17	385
4	737				358	44	402
5	855					332	332
m(j)		78	211	387	505	397	
z(j)		79	170	168	65	0	

Sums for the above Groups

m.	0	78	211	387	505	397
z.	0	79	170	168	65	0
R.	350	428	561	737	855	
r.	157	302	385	402	332	

time at a particular occasion. For example, $m_{1,2} = 78$. In other words, 78 of the original 350 individuals marked and released at occasion 1 (i.e., $\mathbf{R}_1$) were recaptured for the first time at occasion 2. At the third occasion ($m_{1,3}$), 41 individuals marked and released at occasion 1 were recaptured for the first time, and so on.

The r_i values are the total number of individuals captured from a given batch release see below). For example, from the original $\mathbf{R}_1 = 350$ individuals, a total of 157 were recaptured *at least once* over the next 5 capture occasions. Neither the $m_{i,j}$, or r_i values distinguish between newly marked or re-released individuals – they are simply subtotals of all the individuals released at a given occasion. As we'll see shortly, this limits the usefulness of the reduced *m*-array.

begin sidebar

Batch release??

What do we mean by a 'batch release'? Historically, a *cohort* referred to a group of animals released at the same occasion – whether newly marked or not. However, when using **MARK**, we refer to a cohort as all animals marked at the some occasion. In this context, an animal does not change cohorts – it is a 'fixed' characteristic of each marked individual. In the **RELEASE** context, cohort changes with each capture occasion. To prevent confusion, we use the term 'release batch', or simply 'batch', to refer to all individuals (marked and unmarked) released on a given occasion.

end sidebar

Following the reduced *m*-array are the results for **TEST 3**. Since there are 5 recapture occasions there are as many as 7 total **TEST 3** tables (4 for **TEST3.SR** and 3 for **TEST3.Sm**). Let's consider just one of these tables – the first **TEST3.SR** table, for individuals captured on occasion 2 (top of the next page). Why is this the first table? Well, recall what **TEST3.SR** compares – seen before versus not seen before – obviously, at occasion 1, **no** animals were seen before. Thus, we start at occasion 2.

Look closely at the table. Note that the table starts with a restatement of what is being tabulated – here, 'goodness of fit test of seen before versus not seen before against seen again versus not seen again by capture occasions'. You will also see comments concerning which group is being tested, and possibly something concerning the 'control' group. By default, if you're working with only one group, **RELEASE** assumes that it is a 'control group' in a 'control vs. treatments' experiments.

Note the labeling: **TEST3.SR**2. The '**TEST.3SR**' part is obvious, the '2' simply refers to the second occasion (so, **TEST3.SR3** for the third occasion, **TEST3.SR4** for the fourth occasion, and so on). At occasion 2, a total of 428 individuals were released. Of these, 78 had been seen before, and 350 were

```
Goodness of fit test of seen before versus not seen before
against seen again versus not seen again by capture occasions.

                        Test for Group 1
                            Group 1

        TEST 3.SR2: Animals captured on occasion 2

                    +------+------+
                  O|  52  |  26  |  78
                  E|  55.0|  23.0|
                  C|   0.2|   0.4|
                    +------+------+
                  O| 250  | 100  | 350
                  E| 247.0| 103.0|
                  C|   0.0|   0.1|
                    +------+------+
                      302    126    428
          Chi-square=0.6963 (df=1) P=0.4040
```

newly marked individuals. In the first row of the contingency table, we see that of the 78 individuals seen before, a total of 52 (or 67%) of these individuals were ever seen again. In the second row of the table, we see that of the 350 newly marked individuals, a total of 250 (or 71%) were ever seen again. Where did the numbers 52 and 250 come from? Can we tell from the reduced *m*-array? Unfortunately, the answer is 'no'. Why? Because the reduced *m*-array does not 'keep track' of the fates of individuals depending on when they were marked. For this, you need a different type of *m*-array, known as the *full m*-array. To generate the full *m*-array, you need to run **RELEASE** as a standalone application, and modify a specific control element to generate the full *m*-array.

5.4.2. Running RELEASE as a standalone application

To run **RELEASE** as a stand-alone application, you first need to make a simple modification to the `INP` file containing the encounter history data. You simply need to add a single line to the top of the `INP` file. The 'additional' line is the `PROC CHMATRIX` statement. Here is the minimal `PROC CHMATRIX` statement for our example data set:

```
PROC CHMATRIX OCCASIONS=6 GROUPS=1;
```

The `PROC CHMATRIX` statement must include (at least) the `GROUPS` and `OCCASIONS` statements. However, there are a number of other options which can also be applied to this procedure. One of these options is `DETAIL` – as its name implies, the `DETAIL` option provides 'detailed' information about something. The `DETAIL` option is the default in the version of **RELEASE** which comes with **MARK**.

The 'something' is in fact detailed information concerning **TEST 2** and **TEST 3**. When the `DETAIL` option is in effect, **RELEASE** provides the individual contingency tables (including observed and expected frequencies) upon which they are based (discussed below), as well as the summary statistics for all batches pooled. If you have a data set with a large number of occasions, this can generate a very large amount of output.

The opposite to the `DETAIL` option is the `SUMMARY` option, which forces **RELEASE** to print only the summary **TEST 2** and **TEST 3** results for each batch separately and all batches pooled.

You choose either the `DETAIL` or `SUMMARY` option as follows:

```
PROC CHMATRIX OCCASIONS=6 GROUPS=1 DETAIL;
```

To use the `SUMMARY` option (instead of `DETAIL`), you would type

```
PROC CHMATRIX OCCASIONS=6 GROUPS=1 SUMMARY;
```

To generate a full *m*-array (below) you would simply write:

```
PROC CHMATRIX OCCASIONS=6 GROUPS=1 DETAIL FULLM;
```

How do you run **RELEASE**? Simply shell out to DOS, and type:

```
REL_32 I=<INPUT FILE> O=<OUTPUT FILE> <enter>
```

If nothing happens, it probably means that **REL_32** isn't in the PATH on your computer. Make sure it is, and try again. If our **RELEASE** file is called TEST.REL, and we want our results to be written to a file called TEST.LST, then we would type:

```
REL_32 I=TEST.REL O=TEST.LST  <enter>
```

The output would be in file TEST.LST, which you could examine using your favorite text editor. Now, for the present, we're interested in considering the full *m*-array. Assume that we've successfully added the appropriate PROC CHMATRIX statement to the INP file for our simulated data, and successfully run **RELEASE**. In the output, we see something that looks quite different than the simple, reduced *m*-array. This is the full *m*-array, and is shown below:

```
        Full m(i,j) array for Group 1
        Control Group
       Release-Recapture Data
Release ---------------------------------------------------        R(i)-
 i      1        2        3        4        5        6     r(i)  r(i)
-------------------------------------------------------------------------

 1 {1}   350   78( 0)   41( 0)   26( 0)   12( 0)    0( 0)   157   193

 2         {01}   350  141( 0)   77( 0)   28( 0)    4( 0)   250   100
 2         {11}    78   29( 0)   15( 0)    8( 0)    0( 0)    52    26

 3                {001}   350  162( 0)   62( 0)   12( 0)   236   114
 3                {101}    41   17( 0)   12( 0)    1( 0)    30    11
 3                {011}   141   71( 0)   19( 0)    4( 0)    94    47
 3                {111}    29   19( 0)    6( 0)    0( 0)    25     4

 4                       {0001}   350  172( 0)   24( 0)   196   154
 4                       {1001}    26   11( 0)    3( 0)    14    12
 4                       {0101}    77   42( 0)    3( 0)    45    32
 4                       {1101}    15    5( 0)    3( 0)     8     7
 4                       {0011}   162   71( 0)    6( 0)    77    85
...+....1....+....2....+....3....+....4....+....5....+....6....+....7.]..+....8..
 4                       {1011}    17   11( 0)    1( 0)    12     5
 4                       {0111}    71   38( 0)    4( 0)    42    29
 4                       {1111}    19    8( 0)    0( 0)     8    11

 5                              {00001}   350  144( 0)   144   206
 5                              {10001}    12    5( 0)     5     7
 5                              {01001}    28    7( 0)     7    21
 5                              {11001}     8    5( 0)     5     3
 5                              {00101}    62   26( 0)    26    36
 5                              {10101}    12    3( 0)     3     9
 5                              {01101}    19    6( 0)     6    13
 5                              {11101}     6    2( 0)     2     4
 5                              {00011}   172   68( 0)    68   104
 5                              {10011}    11    3( 0)     3     8
 5                              {01011}    42   14( 0)    14    28
 5                              {11011}     5    2( 0)     2     3
 5                              {00111}    71   25( 0)    25    46
 5                              {10111}    11    5( 0)     5     6
 5                              {01111}    38   14( 0)    14    24
 5                              {11111}     8    3( 0)     3     5
```

As you can readily see, the full *m*-array contains **much** more information than the reduced *m*-array. In fact, it contains the entire data set! If you have the full *m*-array, you have all the information you need to run a CMR analysis. If you look closely at the full *m*-array, you'll see why.

Let's concentrate on the information needed to generate **TEST3.SR2**. From the preceding page, recall that of the 78 individuals (i) initially marked at occasion 1, that (ii) were also captured and re-released at occasion 2, 52 were seen again at some later occasion. What would the encounter history of these 78 individuals be? – obviously '11' – marked at the first occasion, and recaptured at the second occasion. The '11' encounter history is represented as {11} in the full *m*-array. Find this encounter history in the 3rd line. To the right, you will see the number 78, indicating that there were 78 such individuals. To the right of this value are the totals, by capture occasion, of individuals from this group of 78 ever seen again. For example, 29 of this 78 were seen again for the first time at occasion 3, 15 were seen for the first time at occasion 4, and so on. In total, of the 78 {11} individuals released, a total of 52 were seen again. You should now be able to see where the values in the **TEST3.SR2** table came from.

Now, consider the 'statistical results'. Although the proportions seen again appear to differ between the two groups (68% for previously marked vs 71% for the newly marked), they are not statistically different ($\chi^2_1 = 0.696$, P=0.412). What are the other 2 numbers in each of the cells? Well, if you look down the left side of the table you'll get a hint – note the 3 letters 'O', 'E' and 'C'. 'O' = the observed frequencies, 'E' = the expected frequencies (under the null hypothesis of the test), and 'C' represents the contribution to the overall table χ^2 value (summing the 'C' values for all four cells yield 0.696. The 'C' values are simply $(O-E)^2/E$). So, for individuals released at the second occasion, there is no significant difference in 'survival' between newly marked and previously marked individuals.

Following the last table (**TEST3.SR5** – individuals released at occasion 5), **RELEASE** prints a simple cumulative result for **TEST3.SR** – which is simply the sum of the individual χ^2 values for each of the individual **TEST3.SRn** results. In this case, $\chi^2 = 2.41$, df = 3, $P = 0.492$. What if **TEST.3SR** had been significant? As we will see shortly, examination of the individual tables is essential to determine the possible cause of lack of fit. In this case, since we have no good 'biological explanation' for **TEST3.SR3** (obviously, since this is a simulated data set!), we accept the general lack of significance of the other tables, and conclude that there is no evidence over all occasions that 'survival' differs between newly marked and previously marked individuals.

Now let's look at **TEST3.Sm2** (i.e., **TEST3.Sm** for occasion 2). Recall that this test focuses on 'of those individuals seen again, when were they seen again, and does when they were seen differ among previously and newly marked individuals?'. As with **TEST3.SR**, there is a contingency table for each of the batches, starting with the second occasion, and ending with occasion 4.

Why not occasion 5? Well, think about what **TEST3.Sm** is doing – it is comparing when individuals are seen again (as opposed to are they seen again). At occasion 5, any individual if seen again must have been seen again at the last occasion (6), since there are no other occasions! So, it doesn't make much sense to create **TEST3.Sm** for the penultimate occasion. Let's consider **TEST3.Sm2** – the second occasion.

```
Goodness of fit test of seen before versus not seen before
against when next seen again by capture occasions.

                        Test for Group 1
                            Group 1

TEST 3.Sm2: Animals captured on occasion 2

                  +------+------+
                O| 141  | 109  | 250
                E| 140.7| 109.3|
                C|   0.0|   0.0|
                  +------+------+
                O|  29  |  23  |  52
                E|  29.3|  22.7|
                C|   0.0|   0.0|
                  +------+------+
                    170    132    302
          Chi-square=0.0070 (df=1) P=0.9335
            Fisher's Exact Test P=1.0000
```

At occasion 2, a total of 428 individuals were released – 78 that had been seen before, and 350 newly

marked individuals. Of these 428 individuals, 302 were seen again. From the **TEST3.Sm2** table (above), 250 of this 302 were newly marked individuals, and 52 were previously marked. You should be able to determine where these totals come from, using the full *m*-array (shown a page or so back).

However, we're now faced with a different puzzle – why only two columns? If **TEST3.Sm** considers 'when' individuals were seen again, then unless all individuals seen again were seen on only the next two occasions, then there should be more than two columns.

Look at the full *m*-array (on the preceding page). We see that of the 428 individuals marked and released at occasion 2, 350 were newly marked and released (the {01} individuals), while 78 were previously marked at occasion 1, and released (the {11} individuals). Of the 350 {01} individuals, 141 were seen again for the first time at occasion 3, 77 were seen again for the first time at occasion 4, and so on. Among the 78 {01} individuals, 29 were seen again for the first time at occasion 3, 15 were seen again for the first time at occasion 4, and so on.

Thus, if we were to construct our own **TEST3.Sm2** table, it would look like:

TEST3.Sm2	**when seen again?**			
seen at (2)	**(3)**	**(4)**	**(5)**	**(6)**
{01}	141	77	28	4
{11}	29	15	8	0

So why doesn't the **RELEASE** table for **TEST3.Sm2** look like this? It doesn't, because **RELEASE** is 'smart' enough to look at the 'true' table (above) and realize that the data are too sparsely distributed for the later occasions for a contingency test to be meaningful. **RELEASE** has simply pooled cells, collapsing the (2×4) table to a (2×2) table.

Now consider **TEST 2**, starting with **TEST2.C**. Recall that in **TEST2.C**, we are 'using' individuals that are known to have survived from (i) to ($i + 1$). **TEST2.Ct** tests if the probability of being seen at occasion ($i + 1$) is a function of whether or not the individual was seen at occasion (i), conditional on surviving from (i) to ($i + 1$). **TEST 2** differs from **TEST 3** somewhat in that we are not considering when an individual was marked, but rather on when it was recaptured. The result for **TEST.2C2** is shown below:

```
Goodness of fit test of recaptures partitioned by rows.

                    Test for Group 1
                        Group 1

TEST 2.C2: Test of row 1 vs. row 2

        +------+------+------+
O|  41   |  26  |  12  |  79
E|  43.8|  24.5|  10.8|
C|   0.2|   0.1|   0.1|
        +------+------+------+
O| 170   |  92  |  40  | 302
E| 167.2|  93.5|  41.2|
C|   0.0|   0.0|   0.0|
        +------+------+------+
          211    118     52    381
Chi-square=0.5129 (df=2) P=0.7738
```

Once each of the component tests **TEST 3** and **TEST 2** are finished, **RELEASE** presents you with a convenient tabulation of all of the individual **TEST 3** and **TEST 2** results. It also gives you some indication as to whether or not there was sufficient data in a given test for you to be able to 'trust' the result. Using our simulated data, we have no significant **TEST 2** or **TEST 3** result. Thus, the overall GOF result (**TEST 2** + **TEST 3** = 6.34) is also not significant ($P = 0.898$). This is perhaps not surprising,

since we set up the simulation so that the data **would** follow the CJS assumptions! Our purpose here was simply to introduce **TEST 2** and **TEST 3**.

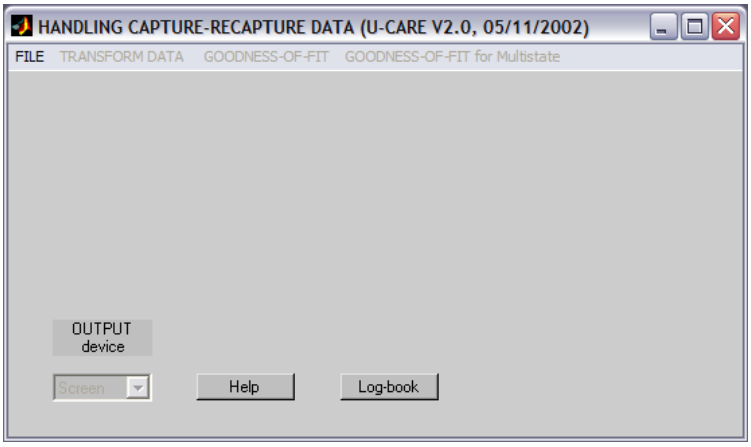
One thing you might be asking yourself at this point is 'since **RELEASE** gives me these nice summary tables, why do I need so much detail?'. The answer – if your data *do* fit the CJS model, then you clearly don't. But if your data don't fit the model (i.e., if any of the tests is rejected), then the only chance you have of trying to figure out what is going on is to look at the individual contingency tables. We got some sense of this when we looked at **TEST3.SR** in our simulated data set – one of the batches had results quite different from the other batches, leading to a near-significant **TEST3.SR** result overall. Further, even if the 4 tests are accepted (no significant differences) you should remember that these tests are for simple heterogeneity – they do not test specifically for systematic differences. Again, the only clue you might have is by looking at the individual tables.

5.5. Enhancements to RELEASE – program U-CARE

Recently, Rémi Choquet, Roger Pradel, and Olivier Gimenez (Choquet *et al*. 2009) have developed a program (known as **U-CARE**, for **U**nified **Ca**pture-**Re**capture) which provides several enhancements to program **RELEASE**. **U-CARE** provides goodness-of-fit tests for single-site models, as well as tests for the multistate **J**olly**M**o**V**e (JMV) and Arnason-Schwarz (AS) models (Pradel *et al*. 2003, 2005 – for discussion of the use of **U-CARE** for GOF testing for multi-state models, see the last section(s) of Chapter 10). Here, we concentrate on using **U-CARE** for GOF testing for single-state models only.

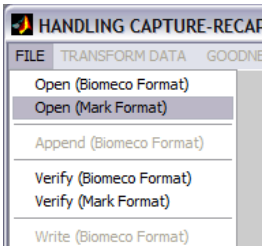
U-CARE contains several tests which are similar to those found in **RELEASE**, but in many cases using slightly different strategies for pooling sparse contingency tables (and thus, the results may differ slightly from those from **RELEASE** – we'll see an example of this shortly). More importantly, however, **U-CARE** incorporates specific 'directional' tests for transience (Pradel *et al*. 1997) and trap-dependence (trap-happiness or trap shyness; Pradel 1993) derived from the contingency tables used in the GOF tests in **RELEASE**. Forthcoming versions of **U-CARE** are anticipated to incorporate further specialized tests and appropriate treatment of sparse data together with indications on the recommended model from which model selection can start.

At present, **U-CARE** cannot be run from within **MARK**, and so must be run separately, as a stand-alone program. When you start **U-CARE**, you will be presented with two windows : one, a 'black DOS window' (which is where evidence of the numerical estimations can be seen – you may have already noticed that **MARK** uses a similar 'command window' during its numerical estimations), and the main 'graphical' front-end to **U-CARE** – clearly, it's pretty 'minimalist':



Initially, only one menu is available: the '**File**' menu. As you might expect, this is where you tell **U-CARE** where to find the data you want to perform a GOF test on.

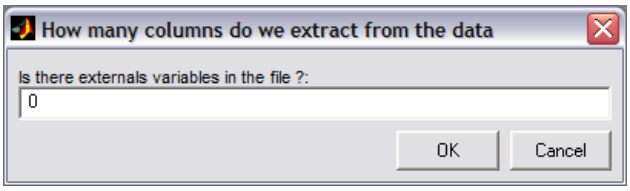
However, if you access the '**File**' menu,



you will see a number of options: you can open encounter history files in one of two formats: a format used in program **SURGE** (and derivatives) known as **Biomeco**, and the one we're familiar with, the **MARK** format (the distinctions between the formats are minor – in fact, **U-CARE** provides you the utility function of being able to read in data in one format, verify it, and write it out in another format.

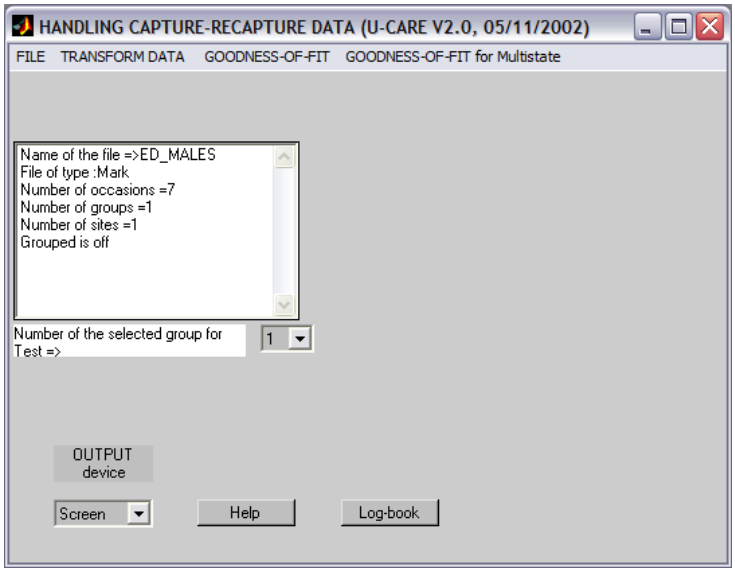
To demonstrate using **U-CARE**, let's test the fit of a familiar data set – the European dippers. We'll focus for the moment on the males only (i.e., a single group). This file is called **ed_males.inp**. We simply select this file using the '**Open (MARK Format)**' file option in **U-CARE**.

Once you've selected the **MARK** input file, **U-CARE** responds with a small 'pop-up' window which is asking you (in effect) if there are any external covariates in the file (see Chapter 2). In this case, with the male dipper data, there are no covariates included in the data set, so **U-CARE** informs you that, as far as it can tell, there are 0 covariates:



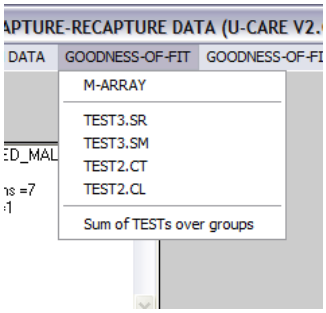
If this is correct (which it is in this case), simply click the '**OK**' button to proceed. You'll then be presented with 2 new windows: one window shows the encounter histories themselves (if they look 'strange' – specifically, if you're wondering why the columns are separated by spaces – not to worry. This is `Biomeco` format, and is functionally equivalent to **MARK**'s input format in how **U-CARE** processes the data).

The other window (shown at the top of the next page) is the main **U-CARE** window, but with many more options now available, plus a window showing you some details about the file you just read in. Note that **U-CARE** assumes that the number of occasions in the data file is the number of occasions you want to test GOF over. In **MARK**, recall that you must 'tell **MARK**' how many occasions there are (or, that you want to use).



If you pull down each of the menus in turn, you'll see that there are a **lot** of options in **U-CARE**. The '**`Transform Data`**' menu provides a set of convenient ways in which to split or pool data (e.g., pooling multiple strata into a single stratum), according to various criterion, reverse the encounter histories, and so forth.

The other two menu options are clearly relevant for GOF testing. There is a GOF menu, and then one specific to multi-state models. For the moment, since our example data have only one 'state' (multi-state models is something we cover is some detail in Chapter 10), we'll consider only the '**`Goodness-of-Fit`**' menu. If you access this menu, you'll see several options.

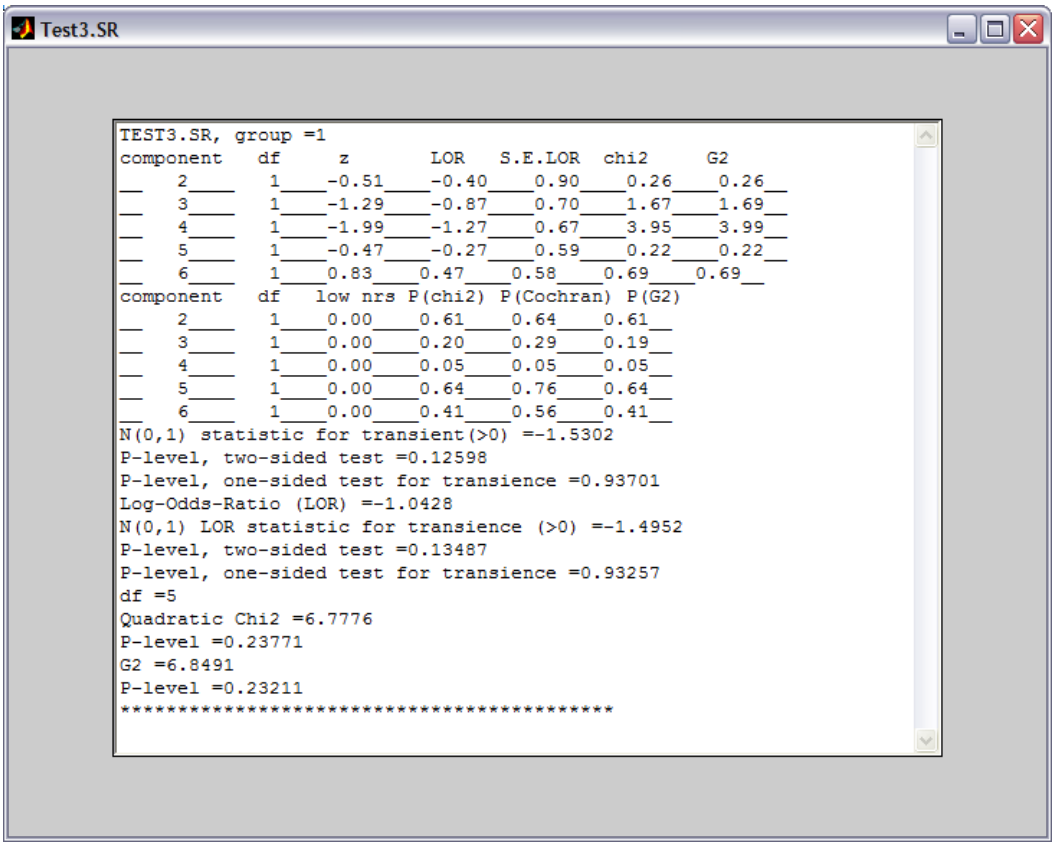


The first ('**`M-ARRAY`**') allows you to generate the reduced *m*-array for your data. Recall that the reduced *m*-array is a summary table, and does not represent all of the details concerning the encounter histories, which are contained in the full *m*-array. The *m*-array is useful for 'visual diagnosis' of some aspects of your data.

Next are 4 component tests: two for '**Test 3**', and two for '**Test 2**'. The use of '**Test 3**' and '**Test 2**' indicates clearly that **U-CARE** is built on the underlying principles (and code base) of program **RELEASE**. In some cases, the tests are identical (for example, **TEST3.SR**). In other cases, they are somewhat different (e.g., there is no **TEST2.CL** in the version of **RELEASE** distributed with **MARK**). More on these individual component tests in a moment. Finally, there is an option (at the bottom of the menu) to sum the tests over groups. This option basically gives you the summary results of the individual component tests, in

a single output.

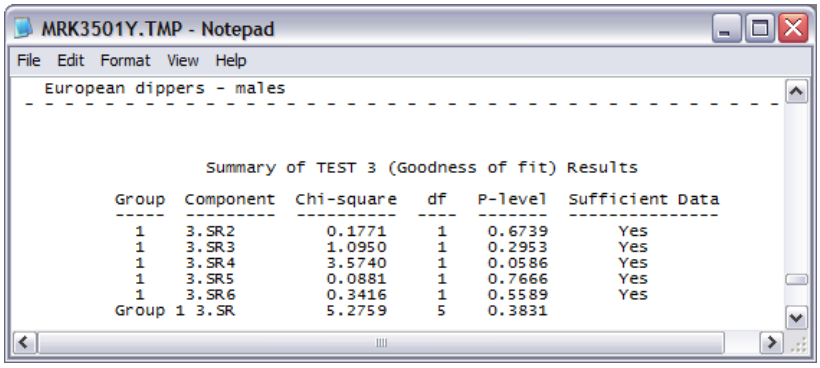
To explore the individual component tests in **U-CARE**, let's proceed to do the GOF testing on the male dippers. We'll start with **TEST3.SR**. Recall from the earlier discussion of program **RELEASE** that **TEST3.SR** tests the hypothesis that there is no difference among previously and newly marked individuals captured at time (i) in the probability of being recaptured at some later time > i (i.e., that whether or not an animal is ever encountered again is not a function of whether or not it is newly marked). If you select **TEST3.SR** from the menu, **U-CARE** will respond with a window showing the contributions of each cohort to the overall χ^2 statistic for this test:



One of the first things we notice from the output for **TEST3.SR** (and all the other tests, which will get to shortly) is that **U-CARE** provides a fair number more 'statistical bits' than you find in the output from program **RELEASE**. For example, you'll recall from our preceding discussion of program **RELEASE** that by careful examination of the individual contingency tables of **TEST3.SR**, you might be able to 'visually' detect systematic departures from expectation in the contingency tables, which might be consistent with transient effects (or age effects). However, **U-CARE** formalizes this level of analysis (while also making it much simpler), by providing a test specific for 'transience' (or, perhaps more accurately, directionality).

In fact, **U-CARE** gives you 2 different approaches to this statistic (the second one based on a log-odds-ratio), as well as both a two-tailed and one-tailed significance test. **U-CARE** also provides two test statistics for overall heterogeneity (the quadratic and likelihood-based G test). The table-like material at the top of the output is the contribution of each cohort to the various statistics (the additivity of the various statistics is quite useful, since it can help you identify particular cohorts which might be having undue influence on the overall fit).

How do these results compare to those from **RELEASE**? Recall we mentioned in passing that **U-CARE** uses a slightly different pooling algorithm than does **RELEASE**, and as such, there will be occasions where **U-CARE** and **RELEASE** give slightly different answers. Here are the results from **TEST3.SR** from program **RELEASE**.



We see that the overall heterogeneity χ^2 statistic from **RELEASE** (which is based on a Pearson statistic) is 5.2759, with 5 df. Based on a two-tailed test, the calculated probability is 0.3831. From **U-CARE**, there are two test statistics: 6.7776 and 6.8491, both with the same degree of freedom (5). Both of these values are somewhat higher than the value from **RELEASE**. These differences come from differences in how pooling in sparse cohort-specific contingency tables is handled between the two programs. You can get a sense of this by comparing the contributions from each cohort to the overall χ^2 statistic between the two programs. Note that the differences are quite striking in this example: many of the cohorts have very sparse data.

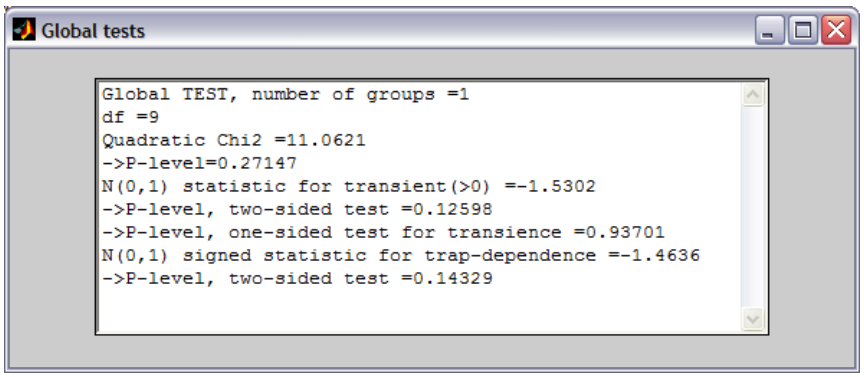
What about the other component tests? In **RELEASE**, there is a companion test for **TEST3.SR**, referred to as **TEST3.Sm** (recall that **TEST3.Sm** tests the hypothesis that there is no difference in the expected time of first recapture between the 'new' and 'old' individuals captured at occasion i and seen again at least once). This test is also available in **U-CARE**.

However, there are some notable differences between **MARK** and **U-CARE** when it comes to **TEST2**. In **MARK**, there is only a single **TEST2** provided (**TEST2.C**), whereas in **U-CARE**, **TEST2** is divided into two component tests: **TEST2.CT**, and **TEST2.CL**. **TEST2.CT** tests the hypothesis that there is no difference in the probability of being recaptured at $(i+1)$ between those captured and not captured at occasion i, conditional on presence at both occasions. **TEST2** differs from **TEST3** somewhat in that we are not considering when an individual was marked, but rather on when it was recaptured. The **TEST2.C** in **MARK** is equivalent to **TEST2.CT** in **U-CARE**. But, what about **TEST2.CL**, which is presented in **U-CARE**? **TEST2.CL**, based on the contingency table where individuals are classified on whether or not they were captured before (or at) occasion (i), and after (or at) occasion $(i+2)$ (and thus known to be alive at both (i), and $(i+1)$). The null hypothesis being tested in **TEST2.CL** is that there is no difference in the expected time of next recapture between the individuals captured and not captured at occasion i conditional on presence at both occasions (i) and $(i+2)$. To date, this test has no 'simple' interpretation, but it is a component test of the overall **TEST2** fit statistic.

5.5.1. RELEASE & U-CARE – estimating ĉ

OK, so now we have several **TEST3** and **TEST2** component statistics. What do we need these for? Well, clearly one of our major motivations is assessing fit, and (more mechanically) deriving an estimate of the $\hat{c}$ value we'll use to adjust for lack of fit. Using either **U-CARE**, or **RELEASE**, one estimate for $\hat{c}$ is to

take the overall χ^2 (sum of the **TEST 2** and **TEST 3** component tests), and divide by the overall degrees of freedom. If we use **RELEASE**, we see that the overall **TEST 3** statistic is 5.276 (for **TEST3.SR**), and 0.000 (for **TEST3.SM**), for an overall **TEST 3** χ^2 = 5.276. For **TEST 2**, there is only one value reported in **RELEASE**: **TEST2.CT** χ^2 = 4.284. Thus, the overall model χ^2 = (5.276 + 4.284) = 9.56, with 9 df. The probability of a χ^2 value this large, with 9 df, is reported as 0.3873. Thus, the estimate for $\hat{c}$, based on the results from **RELEASE**, is (9.56/9) = 1.06, which is close to 1.0. From **U-CARE**, we can get the 'overall' statistics quite easily, simply by selecting '**Sum of tests over groups**'. When we do so, we get the following output:



The overall test statistic is reported as 11.0621, with 9 df, yielding an estimate of $\hat{c}$ = (11.062/9) = 1.23, which is somewhat larger than the value calculated from the **RELEASE** output. But, note that **U-CARE** also provides some further diagnostics: specifically, tests of transience, and trap-dependence. In this case, for the male dipper data, there is no compelling evidence for either transience, or trap-dependence.

What else can we use the component tests for? As described above, we've used the sum of **TEST3** and **TEST2** test statistics, divided by model df, to derive an estimate of $\hat{c}$. But, remember that the model we're testing here is the fully time-dependent CJS model (i.e., $\{\varphi_t\, p_t\}$). But, what do we do if the time-dependent CJS model isn't our general model – what if we want to 'start' with some other model? As it turns out, the components of **TEST 3** and **TEST 2** are still useful in assessing fit, and providing estimates of $\hat{c}$, for a variety of models. The following table indicates some of the ways in which components can be used in this way. Note that we haven't covered some of these models yet (but will in later chapters).

components used	*model*	*detail*
TEST3.SR+TEST3.SM+TEST2.CT+TEST2.CL	$\varphi_t\, p_t$	fully time-dependent CJS model
TEST3.SM+TEST2.CT+TEST2.CL	$\varphi_{t/t}\, p_t$	two age-classes for survival, time-dependence in both age-classes (also know as the 'transience' models in some references)
TEST3.SR+TEST3.SM+TEST2.CL	$\varphi_t\, p_{t*m}$	immediate trap-dependence in re-capture rate (see Pradel 1993)

Thus, using **RELEASE**, and **U-CARE**, goodness-of-fit tests are available readily for 3 models – which, as it turns out, are often the starting points for many single-site recapture analyses. Among them, $\{\varphi_t\, p_t\}$, which makes the assumptions that survival and capture probabilities are solely time-dependent, is the most restrictive because it does not permit survival to differ between newly marked and previously marked animals contrary to $\{\varphi_{t/t}\, p_t\}$, nor capture to differ between animals captured at the previous

occasion and those not captured then, contrary to model $\{\varphi_t\, p_{m*t}\}$. In fact, $\{\varphi_t\, p_t\}$ is nested within each of the two other models. Models $\{\varphi_t\, p_{m*t}\}$ and $\{\varphi_{t/t}\, p_t\}$ are not directly related (i.e., are not nested).

As a consequence of this hierarchy, the goodness-of-fit test of $\{\varphi_t\, p_t\}$ involves more component tests (because more assumptions are to be checked) than the goodness-of-fit tests of $\{\varphi_t\, p_{m*t}\}$ or $\{\varphi_{t/t}\, p_t\}$. In fact, the goodness-of-fit test of $\{\varphi_t\, p_t\}$ can be decomposed into two steps, in either of two ways:

1. via $\{\varphi_{t/t}\, p_t\}$: the goodness-of-fit test of $\{\varphi_{t/t}\, p_t\}$; then, if and only if $\{\varphi_{t/t}\, p_t\}$ appears valid, the test of $\{\varphi_t\, p_t\}$ against $\{\varphi_{t/t}\, p_t\}$
2. via $\{\varphi_t\, p_{m*t}\}$: the goodness-of-fit test of $\{\varphi_t\, p_{m*t}\}$; then, if and only if $\{\varphi_t\, p_{m*t}\}$ appears valid, the test of $\{\varphi_t\, p_t\}$ against $\{\varphi_t\, p_{m*t}\}$

Thus, **RELEASE** and (especially) **U-CARE** provide very good capabilities for GOF testing for several important live-encounter 'mark-recapture' models. But, notice that the models being tested are all 'time-dependent'. While it is true that in many cases the most general model in a candidate model set (and the model for which $\hat{c}$ is estimated) is a time-dependent model, this is not always the case. What if your data are too sparse to ever support a time-dependent model? Or, what if your data don't involve live encounter data? Is there a more generic approach to GOF testing that can be used for any kind of data?

At the risk of oversimplifying, we note that GOF testing for 'typical' data from marked individuals is a form of contingency analysis – do the frequencies of individuals exhibiting particular encounter histories match those expected under a given null model, for a given number released on each occasion? You have probably already had considerable experience with some forms of GOF testing, without knowing it. For example, in some introductory class you might have had in population genetics, you might recall that deviations from Hardy-Weinberg expectations were 'established' by GOF testing – through comparison of observed frequencies of individual genotypes with those expected under the null model.

In general, the goodness-of-fit of the global model can be evaluated in a couple of ways: traditionally, by assuming that the deviance for the model is χ^2 distributed and computing a goodness-of-fit test from this statistic, and using **RELEASE** (for live recapture data only) to compute the goodness-of-fit tests provided by that program (as described previously). However, this approach is generally not valid because the assumption of the deviance being χ^2 distributed is seldom met, especially for multinomial data. Program **RELEASE**, which is only applicable to live recapture data, or dead recovery data under some simplifying assumptions, suffers from the same problem to some degree – but usually lacks statistical power to detect lack of fit because of the amount of pooling required to compute χ^2 distributed test statistics. **RELEASE** also is only really appropriate for simple variations of the time-dependent CJS model.

An alternative, and conceptually reasonable approach, is to use an approach based on 'resampling' the data. In addition to providing a basic GOF diagnostic, such approaches also enable you to 'estimate' the magnitude of the lack of fit. As we shall see, this lack of fit becomes important in assessing 'significance' of some model comparisons. In the following sections, we'll introduce two resampling-based approaches to GOF testing and estimation of $\hat{c}$ currently available in **MARK**: the bootstrap, and the median-$\hat{c}$.

5.6. MARK and bootstrapped GOF testing

As noted in the **MARK** help file, with the bootstrap procedure, the estimates of the model being evaluated for goodness of fit are used to generate data, i.e., a parametric bootstrap. In other words, the parameter estimates (survival, recapture, recovery rate...) for the model in question are used to simulate data. These simulated data exactly meet the assumptions of the model, i.e., no over-dispersion

is included, animals are totally independent, and no violations of model assumptions are included. Data are simulated based on the number of animals released at each occasion. For each release, a simulated encounter history is constructed.

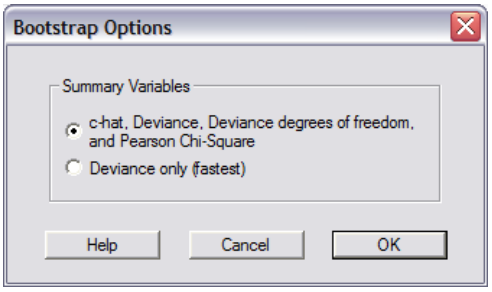
As an example, consider a live recapture data set with 3 occasions (2 survival intervals) and an animal first released at time 1. The animal starts off with an encounter history of 1, because it was released on occasion 1. Does the animal survive the interval from the release occasion until the next recapture occasion? The probability of survival is φ_1, provided from the estimates obtained with the original data. A uniform random number on the interval [0, 1] is generated, and compared to the estimate of φ_1. If the random number is less than or equal to φ_1, the animal is considered to have survived the interval. If the random value is greater than φ_1, the animal has died. Thus, the encounter history would be complete, and would be '100'. Suppose instead that the animal survives the first interval. Then, is it recaptured on the second occasion? Again, a new random number is generated, and compared to the capture probability p_2 from the parameter estimates of the model being tested. If the random value is less than p_2, the animal is considered to be captured, and the encounter history would become '110'. If not captured, the encounter history would remain '100'. Next, whether the animal survives the second survival interval is determined, again by comparing a new random value with φ_2. If the animal dies, the current encounter history is complete, and would be either '100' or '110'. If the animal lives, then a new random value is used to determine if the animal is recaptured on occasion 3 with probability p_3. If recaptured, the third occasion in the encounter history is given a '1'. If not recaptured, the third occasion is left with a zero value.

Once the encounter history is complete, it is saved for input to the numerical estimation procedure. Once encounter histories have been generated for all the animals released, the numerical estimation procedure is run to compute the deviance and its degrees of freedom. In other words, suppose there are a total of 100 individuals in your sample. Suppose you are testing the fit of model $\{\varphi_t\, p_t\}$. What **MARK** does is, for each of these hundred animals, simulate a capture (encounter) history, using the estimates from model $\{\varphi_t\, p_t\}$. **MARK** takes these 100 simulated capture histories and 'analyzes them' – fits model $\{\varphi_t\, p_t\}$ to them, outputting a model deviance, and a measure of the lack of fit, $\hat{c}$ (or, 'c-hat'), to a file. Recall from our early discussion of the AIC (Chapter 4, earlier in this chapter) that $\hat{c}$ is the *quasi-likelihood parameter*. If the model fits perfectly, $\hat{c} = 1$. c is estimated (usually) by dividing the model deviance by the model degrees of freedom. The quasi-likelihood parameter was used to adjust AIC for possible overdispersion in the data (one possible source of lack of fit). Later in this chapter, we will discuss the use of this parameter more fully. The entire process of 'simulate, estimate, output' is repeated for the number of simulations requested. When the requested number of simulations is completed, the user can access the bootstrap simulations results database to evaluate the goodness of fit of the model that was simulated.

Let's look at an example of doing this in **MARK**. We will assess the GOF of the fully time-dependent CJS model $\{\varphi_t\, p_t\}$ to the male European dipper data set. If you still have the database file from your earlier analyses of this data set go ahead and open it up in **MARK**. If not, start **MARK**, open up a new project using the male dipper data, and fit model $\{\varphi_t\, p_t\}$ to the data.

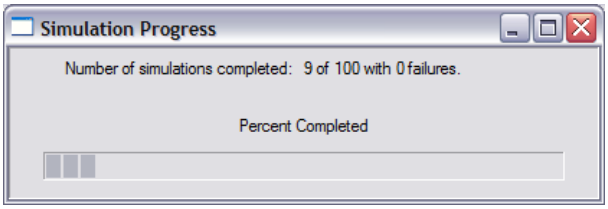
Now, to perform a bootstrapped GOF test on model $\{\varphi_t\, p_t\}$, highlight it in the results browser by clicking on the model once. Right-click with the mouse, and '**retrieve**' the model. Once you have selected the appropriate model, pull down the '**Tests**' menu, and select '**Bootstrap GOF**'.

You will then be presented with a new window (top of the next page), where you are asked if you want to output just the model deviance from the simulated data, or the deviance, deviance degrees of freedom, and the quasi-likelihood parameter c. As noted in the window, outputting the deviance alone is the fastest, but we suggest that you go for all three – it takes longer computationally, but ultimately gives you all the information you need for GOF testing, as well as a robust estimate of c which, ultimately, is necessary for adjusting the models fits in your analysis.



You will then be prompted for a name for the file into which the bootstrapped estimates are to be output. The default is `BootstrapResults.dbf`.

Finally, you will be asked to specify the number of simulations you want to make (the default is 100), and a random number seed (the default is to use the computer system clock to generate a random number seed). While using the same seed is useful on occasion to diagnose particular problems, in general, you should always use a new random number seed. Once you have entered an appropriate number of simulations (more on this below), and a random number seed, click '**OK**'. **MARK** will then spawn a little 'progress window', showing you what proportion of the requested number of simulations has been completed at a given moment.



Remember, that for each iteration, **MARK** is (1) simulating capture histories for each individual in the original sample, and (2) fitting model $\{\varphi_t\, p_t\}$ to these simulated data. As such, it will take some time to complete the task. What do we mean by 'some'? Well, this depends on (1) how big the original data set is, (2) the 'complexity' of the model being fit (i.e., the number of parameters), (3) the number of bootstrapped samples requested, and (4) the computing horsepower you have at your disposal.

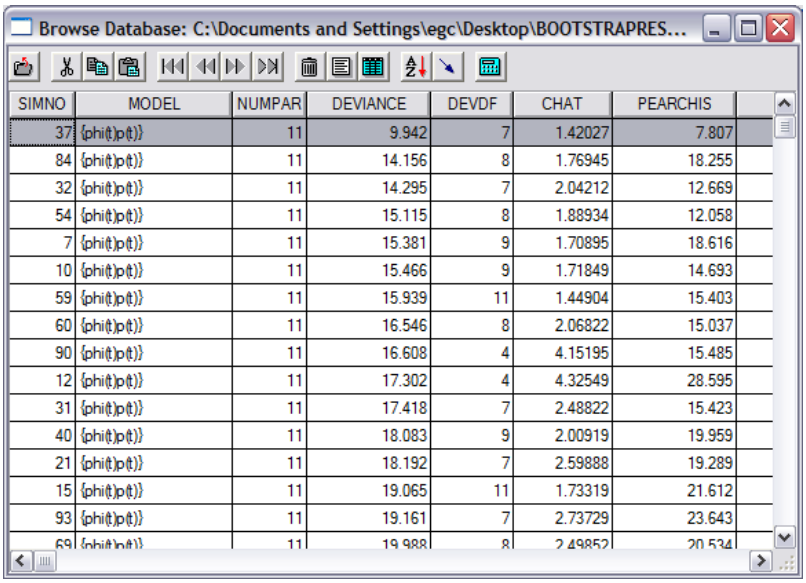
OK – now the big question: how many simulations do we need? To answer this question, you first have to understand how we use these simulated data, and the estimated deviances and quasi-likelihood parameters we derived from them. We'll start with the deviances. In essence, what we try to do with the bootstrapped values is to 'see where the observed model deviance falls on the distribution of all the deviances from the simulated data'. In other words, plot out the distribution of the deviances from the data simulated under the model in question, and look to see where the observed deviance – the deviance from the fit of the model to the original data – falls on this distribution.

Suppose for example, the deviance of the original model was 104.5, whereas the largest deviance from 1,000 simulations was only 101.2. Then, you would conclude that the possibility of observing a deviance as large as 104.5 was less than 1/1,000 (i.e., $P < 0.001$). Or, suppose that you sorted the deviances from the simulated data, from lowest to highest, and found that the 801th deviance was 104.1, and the 802nd value was 105.0. In this case, you would conclude that your observed deviance was 'reasonably likely' to be observed, with a $P < 0.198$ (198/1,000), because 198 of the simulated values exceeded the observed value.

MARK makes it easy to do this. Once your simulations are complete, pull down the '**Simulations**'

menu, and select '**`View Simulation Results`**'. You will then be asked to pick the file containing the results of the simulations (the default was '`BootstrapResults.dbf`'). Select the file.

A window will pop up that bears a fair resemblance to an Excel spreadsheet (in fact, you can read it into Excel if needed):



SIMNO	MODEL	NUMPAR	DEVIANCE	DEVDF	CHAT	PEARCHIS
37	{phi(t)p(t)}	11	9.942	7	1.42027	7.807
84	{phi(t)p(t)}	11	14.156	8	1.76945	18.255
32	{phi(t)p(t)}	11	14.295	7	2.04212	12.669
54	{phi(t)p(t)}	11	15.115	8	1.88934	12.058
7	{phi(t)p(t)}	11	15.381	9	1.70895	18.616
10	{phi(t)p(t)}	11	15.466	9	1.71849	14.693
59	{phi(t)p(t)}	11	15.939	11	1.44904	15.403
60	{phi(t)p(t)}	11	16.546	8	2.06822	15.037
90	{phi(t)p(t)}	11	16.608	4	4.15195	15.485
12	{phi(t)p(t)}	11	17.302	4	4.32549	28.595
31	{phi(t)p(t)}	11	17.418	7	2.48822	15.423
40	{phi(t)p(t)}	11	18.083	9	2.00919	19.959
21	{phi(t)p(t)}	11	18.192	7	2.59888	19.289
15	{phi(t)p(t)}	11	19.065	11	1.73319	21.612
93	{phi(t)p(t)}	11	19.161	7	2.73729	23.643
69	{phi(t)p(t)}	11	19.988	8	2.49852	20.534

In the spreadsheet, you will see the number of the simulation, the name of the model being simulated, the number of estimable parameters in the model (in this case, the number of parameters is the number determined by the rank of the variance-covariance matrix – see the addendum in Chapter 4 for technical details). Next, the model deviance, and depending upon which option you selected when you ran the simulations, the deviance degrees of freedom and the $\hat{c}$ values. To sort the data in order of ascending deviances, simply click the '**`A-Z`**' icon on the toolbar, and then select the deviances (you can sort by any or all the elements in the spreadsheet – we're only after the deviance at this stage).

First, the deviances of the simulated data can be ranked (sorted into ascending order), and the relative rank of the deviance from the original data determined. In this example, we note from the results browser that the deviance for model $\{\varphi_t\, p_t\}$ for the male dipper data is 36.401. Sorting the deviances from the simulated data, we find that the 917th deviance is 36.344, while the 918th deviance is 36.523. The rank of the sorted deviances can be determined using one of the tools on the spreadsheet toolbar.

Thus, the probability of a deviance as large or greater than the observed value of 36.401 is approximately 0.082. So, depending upon your 'comfort level' (after all, selection of an α-level is rather arbitrary), there is probably fair evidence that model $\{\varphi_t\, p_t\}$ adequately fits the male dipper data. On the other hand, some people might argue (reasonably) that $P = 0.082$ isn't particularly 'comforting', so perhaps in fact there is some evidence of lack of fit.

However, this leads us back to the following question – how many simulations do you need? In our experience, a two-stage process generally works well. Run 100 simulations, and do a rough comparison of where the observed deviance falls on the distribution of these 100 values. If the 'P-value' is > 0.2, then doing more simulations is probably a waste of time – the results are unlikely to change much (although obviously the precision of your estimate of the P-value will improve). However as the value gets closer to nominal significance (say, if the observed P-value is < 0.2), then it probably worth doing $\gg 100$ simulations (say, 500 or 1,000). Note that this is likely to take a **long** time (relatively speaking, depending on the speed of your computer).

What else can we do with these bootstrapped simulations? Well, perhaps the most useful thing we can do is to estimate the over-dispersion parameter, c.

Why? Well, recall that if the model fits the data 'perfectly', then we expect the value of $\hat{c}$ to be 1.0; $\hat{c}$ is estimated as the ratio of the model χ^2 divided by the model df. When the value of $\hat{c} > 1$, this is consistent with the interpretation that there is some degree of overdispersion. With a $P = 0.082$, perhaps we might believe that there is some marginal evidence for lack of fit of the general model to the data.

As noted in the **MARK** helpfile, two approaches are possible, based on the deviance directly, and on $\hat{c}$. For the approach based on deviance, the deviance estimate from the original data is divided by the mean of the simulated deviances to compute $\hat{c}$ for the data. The logic behind this is that the mean of the simulated deviances represents the expected value of the deviance under the null model of no violations of assumptions (i.e., perfect fit of the model to the data). Thus, $\hat{c}$ = observed deviance divided by the expected deviance provides a measure of the amount of over-dispersion in the original data.

The second approach is to divide the observed value of $\hat{c}$ from the original data by the mean of the simulated values of $\hat{c}$ from the bootstraps. Again, we use the mean of the simulated values to estimate the expected value of $\hat{c}$ under the assumption of perfect fit of the model to the data. Mean values of both $\hat{c}$ and deviance are easily obtained by simply clicking the 'calculator' icon on the toolbar of the spreadsheet containing the simulated values.

begin sidebar

Careful!

Remember, the simulation results browser allows you to derive a mean $\hat{c}$ simply by clicking a button. However, remember that this mean $\hat{c}$ value is **not** the $\hat{c}$ you need to use. Rather, if you want to use the bootstrapped estimates of $\hat{c}$'s (the 2nd of the two approaches described above), then you take the *observed* model $\hat{c}$ and divide this value by the *mean* $\hat{c}$ from the bootstraps.

end sidebar

As noted in the **MARK** helpfile, there is no good understanding at present of the relative merits of these two approaches. For the example of the male dipper data, using the observed deviance divided by the mean deviances of the simulated data yields a value of (36.401/25.673) = 1.418.

To use the second approach, we first derive the observed $\hat{c}$ value – the model deviance divided by the deviance degrees of freedom. While the model deviance can be read directly from the results browser, the deviance degrees of freedom is obtained by looking in the complete listing of the estimation results – immediately below the print-out of the conditioned S-vector (which is described in the addendum to Chapter 4) In this example, observed $\hat{c}$ is (36.401/7) = 5.20. Dividing this observed value by the mean $\hat{c}$ from the bootstrapped simulations yields (5.20/3.396) = 1.531, which is slightly higher than the value obtained dividing the observed deviance by the mean deviance.

Which one to use? Until more formal work is done, it probably makes sense to be conservative, and use the higher of the two values (better to assume worse fit than better fit – the further $\hat{c}$ is from 1, the bigger the departure from 'perfect fit').

On a practical note, because the observed deviance divided by the mean of the bootstrap deviances does not rely on estimating the number of parameters, it is typically much faster. '**Bootstrap Options**' allows you to specify that you are only interested in the deviance, and not $\hat{c}$, from the bootstrap simulations. Generally, the results are often about the same between the two approaches, but can be different when the degrees of freedom of the deviance varies a lot across the bootstrap simulations (caused by a small number of releases).

5.6.1. RELEASE versus the bootstrap

When 'true' $\hat{c}$ is 1 (i.e., no extra-binomial variation), both **RELEASE** and the bootstrap do equally well (equivalent bias, which was very low in both cases). However, when data were simulated with a 'true' $\hat{c}$ of 2 (i.e., considerable extra-binomial variation), the bootstrap was found to perform less well than did **RELEASE** (negatively biased), with the magnitude of the bias increasing with increasing numbers of occasions (White 2002).

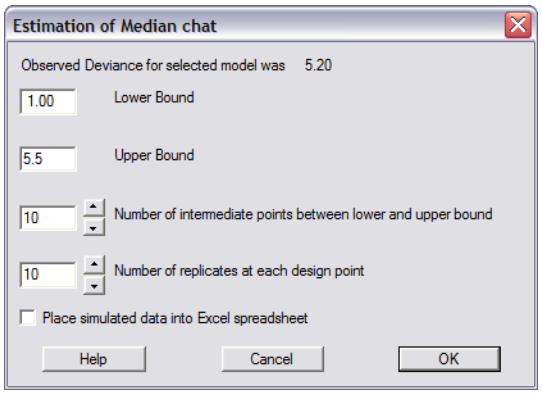
This seems to imply that **RELEASE** is your best option. Arguably, it might be for standard capture-recapture data (live encounters), but will clearly be of limited utility for data types which are not consistent with **RELEASE** (live encounter/recapture data only). So, are we stuck? Well, perhaps not entirely.

5.7. 'median $\hat{c}$' – a way forward?

A new approach (which has been implemented in **MARK**) has recently been described, and appears quite promising. As with all good ideas, it is based on a simple premise: that the best estimate of $\hat{c}$ is the value for which the observed 'deviance $\hat{c}$' value (i.e., the model deviance divided by the model degrees of freedom) falls exactly half-way in the distribution of all possible 'deviance $\hat{c}$ values', generated (simulated) under the hypothesis that a given value of c is the true value. As such, 50% of the generated 'deviance $\hat{c}$' values would be greater than the observed value, and 50% would be less than the observed value. The half-way point of such a distribution is the 'median', and thus, this new approach to GOF testing is referred to in **MARK** as the 'median-$\hat{c}$' approach.

We'll introduce this approach by means of a familiar example – the male European dipper data set we analyzed in the preceding chapter (**`ed_males.inp`**). Using program **RELEASE**, the estimate of $\hat{c}$ for our general model $\{\varphi_t\ p_t\}$ was (9.5598/9) = 1.0622. Based on a bootstrap GOF test, using 500 bootstrap samples, the estimate of $\hat{c}$ is ~ 1.53.

Now, what about this new approach – the 'median-$\hat{c}$'? To run the median GOF test, we simply select this option from the '**`Tests`**' menu. Doing so will spawn the following new window:



At the top, the observed deviance is noted: 5.20 (actually, it's the *observed deviance* $\hat{c}$: the model deviance divided by the deviance degrees of freedom: (36.401349/7) = 5.20). Next, you're presented with a lower and upper bound. The lower bound defaults to 1, since a deviance $\hat{c}$ of 1.0 indicates 'perfect' fit of the model to the data. The upper bound (5.5) is slightly larger than the observed deviance $\hat{c}$.

Next, you're asked to specify the number of intermediate 'design' points between the lower and upper bound, and the number of replicates at each 'design point'.

What do these refer to? Well, first, **MARK** is going to (ultimately) fit a regression line to some simulated data – for a series of different values of $\hat{c}$ (i.e., the number of intermediate points), simulate some data – each time you do so, output the calculated deviance $\hat{c}$ for those simulated data. The number of 'replicates' is the number of simulations you do for each value of c you simulate, between the lower and upper bound. Just like with all regressions, the more points you have between the lower and upper bound, and the greater the number of replicates at each point, the better the precision of your regression. **MARK** defaults to 10 for each, since this represents a good compromise in most cases between precision (which is always something you want to improve), and time (increasing the number of intermediate points and/or the number of replicates at each design point, will take a very long time to run for most problems – yet another reason to start agitating for a faster computer).

OK, so far, so good. But what is this 'regression' we've mentioned a couple of times? Basically, it's a *logistic regression* – a regression where the response variable is a binary state variable, suitably transformed (usually using the logit transform – hence the name logistic regression). In this case, the binary state is 'above the observed deviance $\hat{c}$' or 'below the observed deviance $\hat{c}$'. So, for each value of $\hat{c}$ in the simulation, we generate a sample of deviance $\hat{c}$'s. We count how many of these values are 'above' or 'below' the observed value, and regress this proportion on $\hat{c}$ (using the logistic regression).

Then, all we need to do is use the regression equation to figure out what value of $\hat{c}$ corresponds to the situation where the proportions of the simulated deviance $\hat{c}$'s are equal (i.e., where the number 'above' the observed value is exactly the same as the number 'below' the observed value. This, of course, is the *median* of the distribution). If the number 'above' and 'below' is the same, then this is our best estimate of $\hat{c}$, since values above or below the observed value are equally likely.

begin sidebar

median-$\hat{c}$ and logistic regressions in MARK

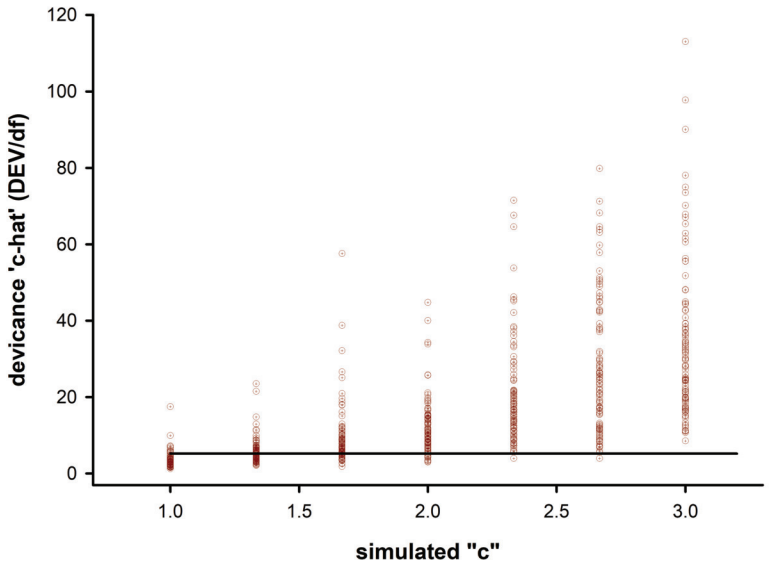
The logistic regression analysis is performed by **MARK** as a *known-fate model*; known-fate models are discussed in a later chapter. For now, it is sufficient to consider that 'fate' in this case is the 'known' proportion of c values above – or below (doesn't matter) – the observed deviance $\hat{c}$.

Output consists of the estimated value of c and a SE derived from the logistic regression analysis, with these estimates provided in a notepad or editor window preceding the known fate output. In addition, a graph of the observed proportions along with the predicted proportions based on the logistic regression model is provided. The initial dialog box where the simulation parameters are specified also has a check box to request an Excel spreadsheet to contain the simulated values. This spreadsheet is useful for additional analyses, if desired.

end sidebar

Let's try this for the male dipper data. The default upper bound is 5.5. We'll change this upper bound to 3.0, for both convenience (i.e., to save time) and for heuristic reasons (if $\hat{c}$ is > 3.0, we may have more fundamental problems; Lebreton *et al*. 1992). We set the number of intermediate points (i.e., c values) to 3 (so, 5 total design points on the function), and the number of iterations (replicates) at each 'true value' of c to 100. If you're trying this on your own, this might take some time.

A plot of the $\hat{c}$ values estimated by **MARK** for the simulations based on the male dipper data is shown at the top of the next page. Each of the open circles in the plot represents the deviance $\hat{c}$ for one of the bootstrapped replicates, at each of the 5 design points (i.e., values of c) in the simulation (in this case, 1.0, 1.5, . . . , 3.0). The solid horizontal line represents the observed deviance $\hat{c}$ for the general model (5.2002). Remember, what we're after is the value of c for which the number of open circles above the observed value is equal to the number of open circles below the observed value.



From the figure, we see clearly that this occurs somewhere in the range of $c = 1.0 \rightarrow 1.5$; for all values of $c > 1.5$, there are clearly far more values above the observed value, than below it.

In fact, if we calculate the frequency of 'above' and 'below' for each value of c, we get the following contingency table (based on 100 replicates in each column – note, your numbers may differ):

	1.0	1.5	2.0	2.5	3.0
above observed	11	66	94	99	100
below observed	89	34	6	1	0

Again, we see clearly that the 'median' will occur somewhere between $c = 1.0$ and $c = 1.5$. Applying a logistic regression to these data (where the logit transformed proportion is the response variable, and the value of c is the independent variable), we get an estimate of the intercept of $\hat{\beta}_0 = 6.7557$, and an estimate of the slope of $\hat{\beta}_1 = -4.8476$. Since we're interested in the point at which the proportion above and below is equal at 50% (i.e., 0.5), then we simply take the logistic equation, set it equal to 0.5:

$$0.5 = \frac{e^{-4.8476(c)+6.7557}}{1 + e^{-4.8476(c)+6.7557}}.$$

Solving for c (not to worry – **MARK** handles all this for you) yields our estimate of $\hat{c} = 1.3936$ (with a SE of 0.033, based on the number of intermediate points and replicates used in our simulation). That's it! So, based on the median approach, the estimate of $\hat{c}$ is 1.3936, which is intermediate between the value reported by **RELEASE**, and the bootstrapped estimate.

So, which one to use? Well, the median-$\hat{c}$ GOF approach is a 'work in progress', but preliminary assessment indicates that it appears to work well. In comparisons for the CJS data type to the **RELEASE** model, the median-$\hat{c}$ is biased high, as much as 15% in one case of $\varphi = 0.5$ with 5 occasions. However, the median-$\hat{c}$ has a much smaller standard deviation for the sampling distribution than the $\hat{c}$ estimated

by **RELEASE**. That is, the mean squared error (MSE) for the median-$\hat{c}$ is generally about $\frac{1}{2}$ of the MSE for the **RELEASE** estimator. Thus, on average, the median-$\hat{c}$ is closer to 'truth' than the **RELEASE** $\hat{c}$, even though the median-$\hat{c}$ is biased high.

One of the current limitations of the median-$\hat{c}$ goodness-of-fit procedure is that individual covariates are not allowed – but unlike the bootstrap, it can be used for most of the other models in **MARK**. Further, it can be applied to any model you want – **RELEASE** (and **U-CARE**) requires you to use the fully time-specific model as the general model, which may not always be ideal depending on your particular circumstances.

Overall, the median-$\hat{c}$ GOF approach seems promising. However, be advised that no GOF test is perfect – the median GOF approach remains a 'work in progress', and its performance compared to other GOF approaches has not been fully evaluated in all situations (e.g., multi-state models). For the moment, we'd suggest, where possible, running as many of the GOF tests as are available – each has different strengths and weaknesses. Hopefully, there will be reasonable agreement among them.

If not, then you need to decide on the choice of which $\hat{c}$ estimate to use (largest, smallest, or otherwise, recognizing that the larger the value of $\hat{c}$ used, the more support simpler models will get relative to more complicated models in your candidate model set, and that the relative scaling of AIC itself will change), and the reason for the choice should be communicated.

begin sidebar

The bootstrap and/or median-$\hat{c}$ won't give me an estimate for $\hat{c}$! Now what?

In some cases, when you run either the bootstrap or median-$\hat{c}$ goodness of fit tests, some/many/all of the simulations will have 'problems' – failure to converge, nonsense values, incorrect parameter counts, and so forth. In virtually all cases, this is not a problem with **MARK**, but, rather, with the general model you are trying to fit your data to (and which **MARK** is attempting to use to simulate data).

Remember, that the general approach to model selection involves multi-model inference over the set of candidate models, under the assumption that at least one of the models adequately fits the data. And of course, this is what you are using the GOF test to assess – the adequacy of fit of your 'general model' to the data. In many cases, your general model will be a time-dependent model in one or more of the parameters. However, you need to be aware that for some sparse data sets, a fully time-dependent model is unlikely to fit your data well – many/most of the parameters will be poorly estimated, if they are estimated at all. And, if one or more parameters can't be estimated – because of sparseness in the data – then **MARK** will not be able to successfully simulate data under that model.

The solution is to accept – grudgingly, perhaps – that your data may simply be inadequate to fitting a time-specific general model. There is nothing particularly wrong with that – you simply need to find a more reduced parameter model which satisfies your needs (i.e., which adequately fits the data). Of course, using anything other than a time-dependent model precludes use of **RELEASE** or **U-CARE** for GOF testing, but that is not a particular problem if the goal is estimation of $\hat{c}$ (and not looking in detail at the underlying sources of lack of fit).

end sidebar

Now it is **very important** that you realize that both the bootstrap and the median-$\hat{c}$ approaches we've just described assume that the source of lack of fit is simple extra-binomial noise. In fact, this is precisely how the simulations work. For example, to simulate a data set with a $\hat{c} = 2.0$, the frequency of each observed encounter history is simply doubled.

What this means is that the estimated $\hat{c}$ is robust (more or less) if and only if the primary source of lack of fit is extra-binomial. If the lack of fit is due to something else (say, structural problems in the model), then the estimated $\hat{c}$ may not be particularly useful. Some of these issues are addressed in the following section.

5.8. The Fletcher $\hat{c}$ – the 'latest and greatest'...

Estimation of overdispersion (i.e., $\hat{c}$) from the observed number of individuals associated with each possible encounter history will be tricky, due to the potentially large number of capture histories with very low expected frequencies. Neither estimates based on the deviance, nor those based on Pearson's statistic will perform well. The former will tend to be strongly negatively biased, while the latter will be less biased but much more variable. In the context of fitting a generalized linear model, David Fletcher (Fletcher 2012) and colleagues (Afroz *et al*. 2020) proposed a modification to the estimator that is based on Pearson's statistic, which is some cases may provide a robust (and much more computationally efficient) alternative to the median-$\hat{c}$.

This new estimator can also be used in the product-multinomial setting, and is given by

$$\hat{c} = \frac{\hat{c}_X}{1+\bar{s}}, \quad \text{where} \quad \bar{s} = \frac{1}{N-n}\sum_{i=1}^{N}\frac{y_i-\hat{\mu}_i}{\hat{\mu}_i} \equiv \frac{\left(\sum_{i=1}^{N} y_i/\hat{\mu}_i\right) - N}{N-n}.$$

Here, $\hat{c}_X$ is the estimator of overdispersion based on the Pearson χ^2 statistic (i.e., the Pearson χ^2 statistic divided by its df), y_i and $\hat{\mu}_i$ are the *observed* and *expected* number of individuals with encounter history i, respectively, and N and n are the total number of observable histories and number of release cohorts, respectively. One of the problems with using Pearson's statistic for sparse data is that the ith term involves dividing by $\hat{\mu}_i$, which will often be very small. The new estimator makes an allowance for this, as the ith term in the denominator also involves dividing by $\hat{\mu}_i$. Simulations suggest that this new estimator also performs better than those based on the deviance. However, there will still be many situations where this modification has non-negligible negative bias.

In **MARK**, the Fletcher-$\hat{c}$ is computed from the Pearson χ^2 statistic, $(obs-exp)^2/exp$. For each cohort of marked/released animals, the sum of the Pearson statistic is computed for all of the *observed* unique encounter histories. In addition, a term for all of the *unobserved* (i.e., observed = 0) encounter histories must be added, which is equal to the sum of their expected values (thus, the (sum of the observed) minus the (sum of the expected) for the observed histories).

The Pearson $\hat{c}$ is then computed as the χ^2-squared statistic divided by its df, which for the Pearson statistic is the total number of *possible* encounter histories minus the number of parameters estimated in the model minus 1 df for each cohort. This estimate is then corrected based on Fletcher (2012; above) to obtain the Fletcher-$\hat{c}$, as the Pearson $\hat{c}$, $\hat{c}_X$, divided by 1 plus the mean of the observed/expected values for all possible encounter histories, $\bar{s}$.

The Fletcher-$\hat{c}$ is calculated automatically for each model in the candidate model set, although use/interpretation is only appropriate for the most parameterized model in the model set. The Fletcher-$\hat{c}$, and the values used in its calculation, are printed in the full output for a given model, right above the tabulation of the β estimates.

For example, for model $\{\varphi_t\, p_t\}$ fit to the male dipper data

```
Pearson Chisquare {phi(t)p(t)} = 75.282128
Possible Encounter Histories {phi(t)p(t)} = 126
Pearson Chisquare df {phi(t)p(t)} = 109
Pearson chat {phi(t)p(t)} = 0.6906617
Sum(Observed/Expected) {phi(t)p(t)} = 86.307408
s-bar {phi(t)p(t)} = -0.3307716
Fletcher chat {phi(t)p(t)} = 1.0320269
```

Here, the Fletcher-$\hat{c}$ is calculated as 1.03203. It is interesting to note that this value is quite close to the value calculated for these data using **RELEASE**, (9.5598/9) = 1.0622.

While quite promising, several factors can cause the estimated Fletcher-$\hat{c}$ to be incorrect. First, losses on capture or dots in the encounter history will create encounter histories that are not considered in the total number of possible encounter histories. That is, the total number of possible encounter histories is based on no missing data. Second, parameter values that cause a reduction in the total number of encounter histories will bias the $\hat{c}$ estimate. Examples of such reductions are an occasion in the CJS data type with $p = 0$, or transition probabilities fixed to 0 or 1 in the multi-state data types.

5.9. What to do when the general model 'doesn't fit'?

We began this chapter by noting that model selection (whether it be by AIC, or some other procedure) is conditional on adequate fit of a general model to the data. As such, the ability to assess goodness of fit is important. As mentioned earlier, there are at least 2 classes of reasons why a model might not adequately fit the data. The first is the 'biologically interesting' one – specifically, that the structure of the general model is simply inappropriate for the data. In subsequent chapters, you'll see how we might have to radically alter the basic ultrastructure of the model to accommodate 'biological reality'. For example, if you are marking both juveniles and adults, then there is perhaps a reasonable expectation that their relative survival rates may differ, and thus, one of the assumptions of CJS (specifically assumption 2) – that every marked animal in the population immediately after time (i) has the same probability of surviving to time ($i+1$) has been violated. The second, perhaps less interesting reason is the possibility that the data are over or under-dispersed for the CJS model – extra-binomial variation. In this section, we will briefly discuss both cases.

5.9.1. Inappropriate model

Your most immediate clue to lack of fit will be a high $\hat{c}$ value. The 'challenge' will be to determine whether or not this is because you have an inappropriate model, or extra-binomial 'noise'. In some cases, this is fairly straightforward. For example, to confirm that the basic live-encounter CJS model has been rejected because the model is 'biologically unrealistic' for your data, you simply need to carefully examine the detailed **TEST 2** and **TEST 3** contingency tables in your **RELEASE** output file (or, more conveniently, look at it directly with **U-CARE**). What are you looking for? Well, in general, the thing you're looking for is a 'systematic' rejection (or bias) in the individual tables. You need to see if the failure of **TEST 2** or **TEST 3** is 'driven' by a few 'strange' batches, or is due to a 'systematic' bias. What do we mean by 'systematic' bias? Well, by 'systematic', we refer to a bias which occurs consistently at each occasion – a bias in the sense that a particular cell (or cells) in one of the test tables is consistently over or underpopulated.

An example will help make this clear. Suppose you run **RELEASE**, and find that **TEST 3** is rejected, but not **TEST 2**. You say to yourself, 'OK, recapture seems to be OK, but something is wrong with the survival assumption, under the CJS model'. You proceed to look carefully at each of the **TEST3** tables for each batch. You note that **TEST3.SR** is rejected, but that **TEST3.SM** is accepted.

Now, what does this mean? Recall that **TEST3.SR** simply asks, of those individuals seen either on or before occasion (i), what proportion were ever seen again? If **TEST3.SR** is rejected, then this suggests that there is a difference in 'survival' among individuals, depending on whether or not they were seen for the first time either on or before occasion (i). However, **TEST3.Sm** only looks at individuals who *were* seen again. Among these individuals, when they were seen again does not depend on whether or not they were seen for the first time at occasion (i).

Suppose you look at each individual **TEST3.SR** table, and find the following – a '+' indicates more individuals than expected (based on the null hypothesis of no differences between groups), and a '-' indicates fewer individuals than expected (under the same hypothesis). Since **U-CARE** provides you with the overall 'directional' test, you can make this determination very quickly. Let's say we have 10 occasions, and we find that this pattern seems to be present in the majority of them (you might use some statistical test, for example a sign test, to determine if the frequency of tables exhibiting a particular pattern occurs more often than expected by random chance). Say, 8/10 contingency tables show this pattern (which will clearly be statistically significant).

What does this suggest? Well, among individuals seen for the first time at occasion (i), significantly more are never seen again than expected, relative to individuals who had been seen before occasion (i). In other words, newly marked individuals showed a consistently lower probability of ever being seen again than previously marked individuals.

What could lead to this pattern? One possibility we suggested at the beginning of this section was age effects. Lower survival of newly marked juveniles (relative to adult survival) would lead to this pattern in **TEST3.SR**. Is this the 'only' plausible explanation? Unfortunately, no. Life would be simpler if there was only ever one possible explanation for anything, but, this is generally not the case. This example is no exception. Rejection of **TEST3.SR** could also reflect (1) a marking effect (where the act of marking causes an increase in immediate mortality), (2) presence of transients (migratory individuals leaving the sampling area shortly after marking), or (3) heterogeneity in capture rates (some individuals have low capture rates, some high).

The point here is that there may be more than one possible answer – it is at this point you'll need to use your 'biological insight' to help differentiate among the possible explanations. The other, perhaps more important point is that the presence of a consistent difference in one of the major tests (**TEST2.Ct**, **TEST2.Cm**, **TEST3.SR**, and **TEST3.Sm**) each suggest the possibility of one or more effects which violate the basic CJS assumptions. You will have to rely on your insight to help you identify possible 'biological' explanations for violation of any of these 4 tests – each of them might refer to something completely different.

What can you do if you do reject CJS? Well, the solution depends on what, exactly, 'has gone wrong'. In general, if the individual **TEST 2** or **TEST 3** results seem to show systematic deviations among occasions, the most likely solution will be to reject the CJS model as the 'correct' starting model for your analysis – it clearly doesn't fit, because the inherent assumptions aren't met by the data. In this case, where **TEST3.SR** is rejected, but the other tests are accepted, then the solution is to add age-structure to the model (this will be presented in Chapter 8).

However, simply recognizing that a 'different' starting model (say, a 2-age class model) may be more appropriate is only the first step. You still need to confirm that the data fit your 'new' model. You must go through analogous GOF tests for the 'new' starting model, just as we have done for the CJS model (as discussed earlier in this chapter).

What if your data type is one that can't be handled by **RELEASE** (which, in effect, is every data type other than live-encounter mark-recapture data)? One option is to examine *deviance residual plots*. If you click the '**`Graph Deviance Residuals`**' button on the main **MARK** toolbar, residuals are plotted against their encounter history number to assess the fit of the model. The default for the residual plot icon is deviance residuals. However, either deviance residuals or Pearson residuals are available from the '**`Output | Specific Model Output`**' menu of the results browser.

A *deviance residual* is defined as

$$\text{sign}\left(\text{obs}-\text{exp}\right)\sqrt{\frac{2\left[(\text{exp}-\text{obs})+\text{obs}\times\ln\left(\text{obs}/\text{exp}\right)\right]}{\hat{c}}},$$

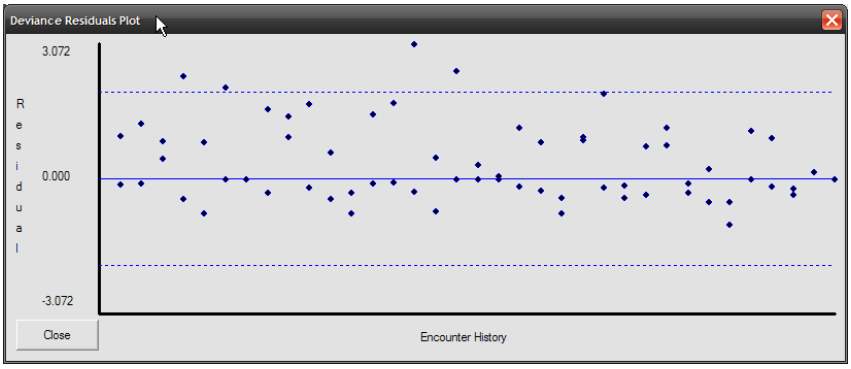
where 'sign' is the sign (plus or minus) of the value of (obs – exp).

A *Pearson residual* is defined as

$$\frac{(\text{obs} - \text{exp})}{\sqrt{(\text{exp} \times \hat{c})}}.$$

Take for example the swift analysis we introduced in Chapter 4. If we run a bootstrap analysis on our general model $\{\varphi_{c*t}\ p_{c*t}\}$, we get an estimate of $\hat{c} = 1.00$. So, pretty good fit!

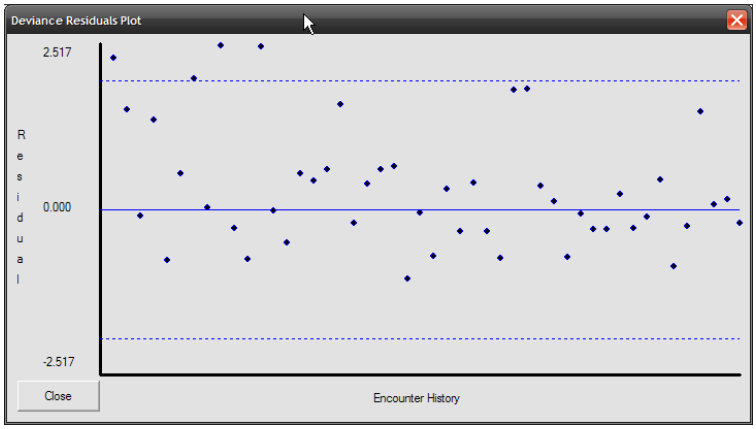
This is reflected in the deviance residual plot shown below:



If the model was a 'good-fitting model' (as suggested by our estimated $\hat{c}$), we would expect that there would be no 'trend' (non-randomness) in the pattern of the residuals - roughly half of the residuals should be above the 0.000 line, and half should be below. Further, if there is no extra-binomial variation, then most of these residuals should be fairly close to the 0.0000 line (between the two horizontal dashed lines in the preceding figure).

In the plot (above), there is little apparent trend, although most of the extreme residuals are 'on the high side' (large positive deviance – to see which observation is causing a particular residual value, you can place your mouse cursor on the plotted point for the residual, and click the left button. A description of this residual will be presented, including the group that the observation belongs to, the encounter history, the observed value, the expected value, and the residual value). However, the residuals are roughly randomly distributed above and below 0.

Here is an example of a deviance residual plot which seems to indicate lack of fit.



In the second plot, notice both a clear asymmetry in the residuals (greater proportion of positive to negative residuals) as well as some suggestion of a trend (although most residuals are positive, the magnitude of the deviation above zero seems to decline from left to right). Both suggest strongly that there is a structural problem with the fitted model. In fact, this particular plot was generated by fitting a $\{\varphi_.\}$ model structure to data where in fact φ increased linearly over time (a topic we discuss in Chapter 6). So, clearly, the fit model was not structurally appropriate, given the underlying model used to generate the data. We will re-visit the use of residual deviance plots to help evaluate lack of fit as we introduce new data types.

5.9.2. Extra-binomial variation

If there are systematic deviations in your contingency tables, or if there is some other indicator of structural causes of lack of fit (say, from a deviance residual plot), then changing the structure of the general model is the appropriate step to take next. However, what do you do, if the deviations in the contingency tables are not 'consistent' among batches – what if (say) 5/10 tables are biased in one direction, and 5/10 are biased in the other direction?

In such cases, where there seems to be no clear 'explanation' (biological or otherwise) for the violation of **TEST 2** or **TEST 3**, you then have only a few options. As you've no doubt gathered if you've read this far into this chapter, the most common 'solution' (although it is only a partial solution) is to 'adjust' your statistics to account for the 'extra noise', or (for the statistically inclined) the extra-binomial variation. Remember that the conceptual basis of all models is 'data = structure + residual variation'. In general, the structure of the residual variation is unknown, but for multinomial distributions, it is known. If the model structure is 'correct', then the variance of the residual 'noise' is 1 (where variance is defined as the expected value of the GOF χ^2 divided by its degrees of freedom). However, even if the residual variation > 1, the structural part of the model can be 'correct'. In simplest terms, if there is 'excess variation' it will show up in the model GOF χ^2 (since this value is essentially a 'residual SS'). Thus, what we need to do is 'correct' everything for the magnitude of this extra variation. To do this, we derive what is known as a variance inflation factor, $\hat{c}$. The larger the value of $\hat{c}$, the greater the amount of 'extra' variation. We have already presented this basic idea earlier in the chapter, as well as the mechanics of how to get **MARK** to adjust things.

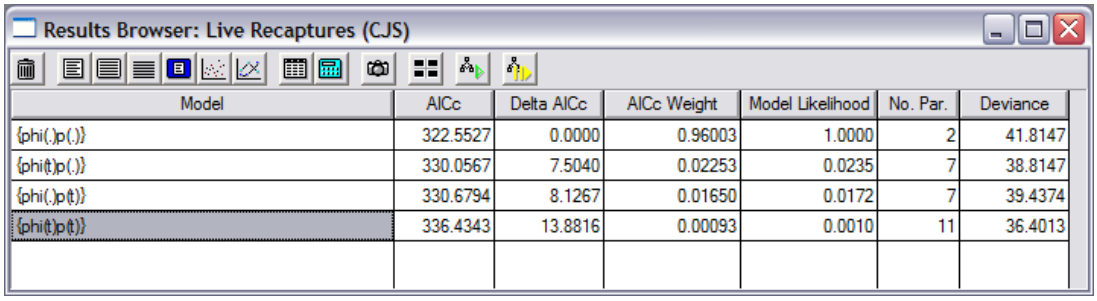
Consider, for example, the dipper data set, where we can estimate $\hat{c}$ in several different ways: using the bootstrap, median-$\hat{c}$, or by using a χ^2/df approach in **RELEASE** and **U-CARE**. Recall that from our bootstrap fit of $\{\varphi_t\, p_t\}$ to the male European dipper data set yielded an estimate of $\hat{c}$ of either 1.418, or 1.531 (depending on whether or not you calculated $\hat{c}$ using the bootstrapped distribution of deviances, or $\hat{c}$ values). The mean of these two values is 1.475. Now, in both **RELEASE** and **U-CARE**, the sum of the overall results of **TEST 2** and **TEST 3** is (in effect) the overall model χ^2. This is provided for you directly in the **RELEASE** and **U-CARE** output. For the male dipper data, **RELEASE** gives the sum of **TEST 2** + **TEST 3** = 9.5598. The model df = 9, and thus from **RELEASE**, $\hat{c}$ is estimated as (**TEST 2** + **TEST 3**)/df = (χ^2/df) = 1.0622. From **U-CARE**, (**TEST 2 + TEST 3**)/df = (χ^2/df) = (11.0621/9) = 1.229. Now, in fact, there does seem to be some variation in estimates of $\hat{c}$, depending on the method selected, with the estimates from the bootstrap approach being the highest, and that from program **RELEASE** being the lowest.

In fact, the reason (in this example) is because the male dipper data has 'a problem' – the data are somewhat sparse – so much so that many of **RELEASE** tests (especially **TEST 3**) are reported as 'invalid' (insufficient data to make the particular contingency test(s) valid). This is also reflected in the fact that one parameter (recapture rate p_3) is not particularly well-estimated (this can either be because the parameter is estimated near the boundary, or because of 'weirdness' in the data). However, these nuances notwithstanding, if you have good data (i.e., not so sparse that **RELEASE** has to do a lot of

pooling over cells in the contingency tables), then the estimate of $\hat{c}$ using the bootstrap or the median-$\hat{c}$ in **MARK** should be very close to the estimate using the (χ^2/df) approach in **RELEASE** or **U-CARE**.

OK – so now what do we do with our measure of the fit of the CJS model to the male dipper data? Our estimate of the significance of departure from 'adequate fit of the model to the data' was $P = 0.08$. As noted, our estimate of $\hat{c}$ varied depending upon how it was derived: for now, we'll use $\hat{c}$=1.531 (the value from the bootstrap). Now what? Well, think a moment about what we're doing when we do the model fitting – we want to compare the *relative* fit of different models in the model set. If there is 'significant lack of fit', then intuitively this may influence our ability to select amongst the different models. Also intuitively, we'd like to adjust our model fits to compensate for the lack of fit. How do we accomplish this?

First, look at the 'original results':



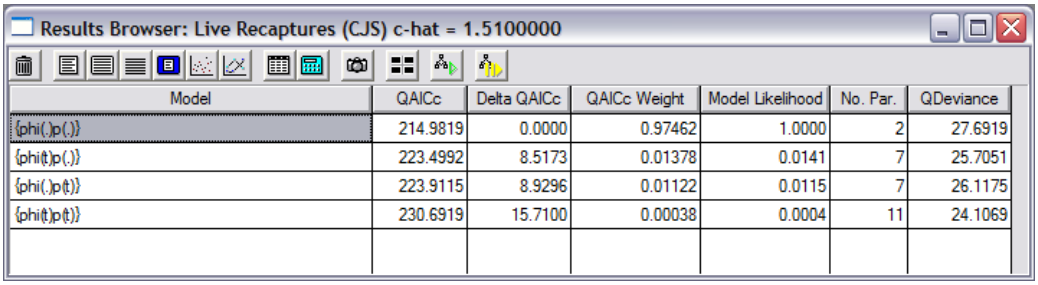
Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(.)p(.)}	322.5527	0.0000	0.96003	1.0000	2	41.8147
{phi(t)p(.)}	330.0567	7.5040	0.02253	0.0235	7	38.8147
{phi(.)p(t)}	330.6794	8.1267	0.01650	0.0172	7	39.4374
{phi(t)p(t)}	336.4343	13.8816	0.00093	0.0010	11	36.4013

These AIC$_c$ weights were calculated using the default $\hat{c}$ value of 1.0. As such, we would have concluded that the best model in the model set was ~ 43 times better supported by the data than was the next best model.

What happens if we adjust the values in the results browser using $\hat{c}$=1.51? **MARK** makes such an 'adjustment' very easy to do. Simply pull down the '**`Adjustment`**' menu, and select '**`c-hat`**'. Enter the new value for $\hat{c}$ (in this example, use 1.51). As soon as you've entered the new $\hat{c}$, the various AIC and AIC weighting values in the results browser are converted to QAIC$_c$ values (note that the column labels change in the results browser to reflect this change).

For example, consider the original AIC and AIC weight values for the 4 simplest models we applied to these data:



Results Browser: Live Recaptures (CJS) c-hat = 1.5100000

Model	QAICc	Delta QAICc	QAICc Weight	Model Likelihood	No. Par.	QDeviance
{phi(.)p(.)}	214.9819	0.0000	0.97462	1.0000	2	27.6919
{phi(t)p(.)}	223.4992	8.5173	0.01378	0.0141	7	25.7051
{phi(.)p(t)}	223.9115	8.9296	0.01122	0.0115	7	26.1175
{phi(t)p(t)}	230.6919	15.7100	0.00038	0.0004	11	24.1069

Now, we see that the degree of support for the best model has increased substantially – the best model is now > 70 times better supported than the next best model. In this case, it didn't change our final conclusions, but it is not hard to imagine how changes of this magnitude could make a significant difference in the analysis of the data.

Does anything else happen to 'our results'? The answer is 'yes', but it isn't immediately obvious – adjusting $\hat{c}$ also changes the estimated standard errors for each of the parameters in each of the models. Try it and confirm this for yourself. However, there is a subtle catch here – the estimated standard errors are changed **only** in the output of the estimates, **not** in the general output. Don't ask! :-)

5.10. How big a $\hat{c}$ is 'too big'?

When should you apply a new $\hat{c}$? Is a value of $\hat{c} = 1.51$ really different than the null, default value of 1.0? At what point is $\hat{c}$ too large to be useful? If the model fits perfectly, then $\hat{c} = 1$. What about if $\hat{c} = 2$, or $\hat{c} = 10$? Is there a point of diminishing utility? As a working 'rule of thumb', provided $\hat{c} \leq 3$, you should feel relatively safe (see Lebreton *et al*. 1992 – pp. 84-85).

However, there are a couple of fairly important 'philosophical' considerations you'll need to keep in mind. First, for some analysts, the most important thing is adjusting for fit to make the parameter estimates as robust and valid as possible. However, for others, the question of 'why' there is lack of fit is important. Consider, for example, the standard 'CJS assumptions'. In particular, the assumption that all individuals are equally likely to be caught. In truth, there may very often be considerable variation among individuals in the probability of capture – a clear violation of the CJS assumptions.

The question, then, is whether the lack of fit is due to these sorts of violations (strictly speaking, this is referred to as the problem of individual heterogeneity in capture probability), or structural problems with the general model. Clearly, by adjusting $\hat{c}$, you can accommodate lack of fit up to a certain degree, but if the $\hat{c}$ is fairly large (> 2) then this might be a good indicator suggesting a careful examination of the structure of the model in the first place is needed. So, yes, you can 'adjust' for lack of fit simply by adjusting the $\hat{c}$ value used. However, be advised that doing so 'blindly' may obscure important insights concerning your data. It is always worth thinking carefully about whether your general model is appropriate. Moreover, as $\hat{c}$ increases $\gg 1$, interpretation of some of the metrics we'll use for model selection (something we cover in later chapters) becomes more complicated.

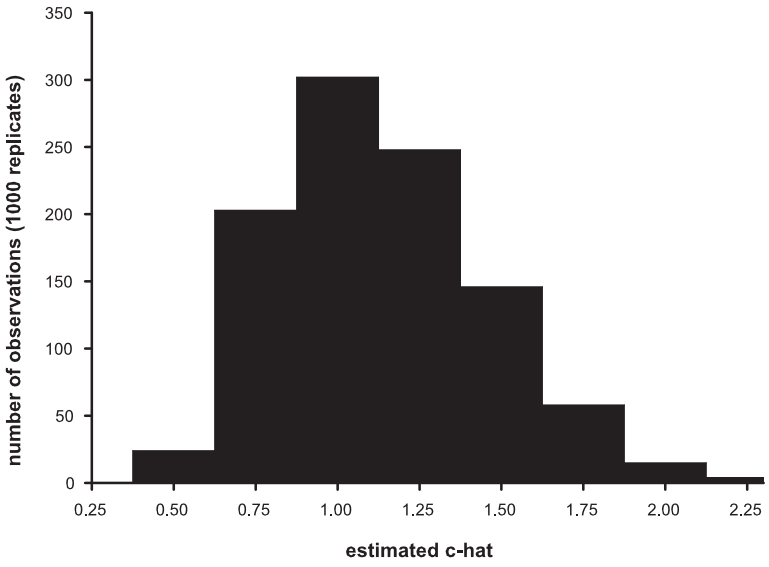
Second, as noted earlier, the bootstrap and median-$\hat{c}$ resampling approaches estimate $\hat{c}$ under the assumption that the source of the lack of fit is strictly extra-binomial 'noise' – as you might see, for example, if your sample consisted of male and female pairs, where the fate of one member of the pair was not independent of the other. The more 'behavioral structure' you have in the biology (or process) underlying your data, the more likely this is to be the case. A method to evaluate whether theoretical variance estimates are valid when the individuals are considered statistically independent, or whether variance inflation procedures are required to account for dependence among (say) siblings, is described in Appendix G. However, if the source of lack of fit is structural, and not due to extra-binomial noise, then adjusting for a $\hat{c}$ *estimated* using either of the resampling approaches may not do you much good. In such cases, your best option is to (i) work hard at trying to identify the structural problems in your model, and (ii) see how sensitive your model weights are to manual adjustments (in small increasing increments) in $\hat{c}$. This basic process is also discussed below.

Finally, and perhaps most importantly, remember that we are *estimating* c. And as with any estimate, there is uncertainty about our estimate of $\hat{c}$. As such, if you derive an estimate of c that is (say) $\hat{c} = 1.4$, there may be a significant probability that in fact the true value of c is 1.0 – but, because of the uncertainty in your estimate of c, you can't be sure.

We can demonstrate this fairly easily, using some data simulated under a true model $\{\varphi_t\, p_t\}$. We will do this using the **RELEASE** simulations capability in **MARK** (see Appendix A). We will simulate 8 occasions in the simulated data set, with 250 newly marked and released individuals on each occasion (clearly, a much bigger data set than the male dipper data). For 1,000 simulations, the mean $\hat{c}$ (estimated as the **TEST 2** + **TEST 3** χ^2 divided by the df) was 0.997, which is very close to the expected $\hat{c}$ of 1.00.

However, a couple of points to make. While the mean is very close to the expected value, there is a fair bit of variation around this value. For example, when we ran the simulation (as described), we get a variance of 0.102, with a 95% CI of [0.543, 1.57].

This is reflected in the following histogram of $\hat{c}$ estimates:



Note that there is a fair chance that for a given simulated data set, that the observed estimate of $\hat{c}$ is considerably less than, or greater than, 1.0 – even though the true value is 1.0 for these data! This is not really a problem, but one you need to be aware of: it is possible that you are fitting the correct model to the data (such that the true $\hat{c}$ is 1.0), but because each data set you collect is simply one realization of an underlying stochastic probabilistic process, there is some chance that the $\hat{c}$ you observe will differ – sometimes markedly so – from 1. We still use a quasi-likelihood adjustment to account for lack of fit, since we cannot be certain we have the correct model. (*Note*: the histogram should be approximately χ^2 distributed for the given degrees of freedom, as it appears to be for this example).

5.10.1. General recommendations

OK, some general recommendations:

1. identify a general model that, when you fit it to your data, has few or no estimability problems (i.e., all of the parameters seem adequately estimated). Note that this general model may not be a fully time-dependent model, especially if your data set is rather sparse. If there is plausible 'biological' justification for suspecting non-independence among individuals in your data, then you should evaluate whether theoretical variance estimates are valid when the individuals are considered statistically independent, or whether variance inflation procedures are required to account for dependence among (say) siblings. A formal procedure for performing such an evaluation is introduced in Appendix G.

2. estimate $\hat{c}$ for this general model, using whichever method is most robust for that particular model. While the median-$\hat{c}$ and Fletcher-$\hat{c}$ approaches show considerable promise to be generally robust approaches to estimate $\hat{c}$, for the moment it is perhaps worth generating $\hat{c}$ using all of the available approaches, and comparing them. If you have some estimates marginally above 1.0, and some marginally below 1.0, it is probably reasonable to assume the $\hat{c} \cong 1.0$. Alternatively, you could choose to be conservative, and select the largest $\hat{c}$, which provides some protection (in a sense) against selecting overly parameterized models.
3. remember that the 'estimate of c' is just that – an *estimate*. Even if the true value of c is 1.0, we use the estimate of $\hat{c}$ to account for model selection uncertainty (since we cannot be certain we have the correct model). Moreover, as noted earlier, the bootstrap and median-$\hat{c}$ resampling approaches estimate $\hat{c}$ under the assumption that the source of the lack of fit is strictly extra-binomial 'noise' – where the fate of one individual may not be independent of the other. However, if the source of lack of fit is structural, and not due to extra-binomial noise, then adjusting for a $\hat{c}$ *estimated* using either of the resampling approaches may not do you much good. In such cases, your best option is to (i) work hard at trying to identify the structural problems in your model, and (ii) see how sensitive your model weights are to manual adjustments (in small increasing increments) in $\hat{c}$ (see next point).
4. it is also worth looking qualitatively at the 'sensitivity' of your model rankings to changes in $\hat{c}$, especially for models (data types) for which no robust GOF test has been established (i.e., most open population models). Manually increase $\hat{c}$ in the results browsers from 1.0, 1.25, 1.5 and so on (up to, say, 2.0), and look to see how much the 'results' (i.e., relative support among the models in your candidate model set) changes. In many cases, your best model(s) will continue to be among those with high QAIC_c weight, even as you increase $\hat{c}$. This gives you some grounds for confidence (not much, perhaps, but some). Always remember, though, that in general, the bigger the $\hat{c}$, the more 'conservative' your model selection will be – QAIC_c will tend to favor reduced parameter models with increasing $\hat{c}$ (a look at the equation for calculating QAIC_c will show you why). This should make intuitive sense as well – if you have 'noise' (i.e., lack of fit), perhaps the best you can do is fit a simple model. In cases where the model rankings change dramatically with even small changes in $\hat{c}$, this might suggest that your data are too sparse for robust estimation, and as such, there will be real limits to the inferences you can make from your candidate model set.

This final point is related to what we might call the 'wince' statement: if you are sure your model structure is correct, and despite you're finer efforts, your $\hat{c}$ is $\gg 3$, or if your model rankings change radically with even small changes in $\hat{c}$, then you might just have to accept you don't have adequate data to do the analysis at all (or, perhaps in a more positive tone, that there are real limits to the inferences you can draw from your data). Unfortunately, your 'time, effort, and expense' are not reasonable justifications for pushing forward with an analysis if the data aren't up to it. Remember the basic credo. . .'garbage in...garbage out'.

Last words – for now (remember, GOF testing is very much a work in progress). If your data are based solely on live encounter data, then it appears that for the moment, for most data sets, using estimates of $\hat{c}$ derived from (χ^2/df) calculations (using either **RELEASE** or **U-CARE**), or the Fletcher-$\hat{c}$, is more robust (less biased) than bootstrapped or median-$\hat{c}$ estimates. But, what if you don't have live encounter data, or perhaps some mixture of live encounter with other types of encounter data (e.g., perhaps dead recovery)? For other data types (or mixtures), in some cases GOF tests have been suggested (e.g., contingency GOF testing for dead recovery data is implemented in program **MULT**), but the degree to which estimated of

$\hat{c}$ from these approaches may be biased is unknown. However, in many cases, the bootstrap or median-$\hat{c}$ can be run, which does provide some estimate of $\hat{c}$, albeit potentially biased in some cases.

5.11. Summary

That's the end of our **very** quick stroll through the problem of GOF. In this chapter, we have focussed on using program **MARK** and programs **RELEASE** and **U-CARE** to handle this task. Remember – the bootstrap, median-$\hat{c}$ and Fletcher-$\hat{c}$ approaches provide omnibus techniques for assessing GOF for **any** model, as well as providing robust estimates of $\hat{c}$ (at least, in theory – GOF testing is still very much a work in progress). **RELEASE**, which should be applied to live-recapture models only, is a good tool for examining 'where' the departures occur. Residual plots can also be quite helpful.

5.12. References

Afroz, F., Parry, M., and Fletcher, D. (2020) Estimating overdispersion in sparse multinomial data. *Biometrics*, **76**, 834-842.

Burnham, K. P., Anderson, D. R., White, G. C., Brownie, C., and Pollock, K. H. (1987) *Design and analysis methods for fish survival experiments based on release-recapture*. American Fisheries Society Monograph No. 5. Bethesda, Maryland, USA. 437pp.

Choquet, R., Lebreton, J.-D., Gimenez, O., Reboulet, A. M., and Pradel, R. (2009) U-CARE: Utilities for performing goodness of fit tests and manipulating CApture-REcapture data. *Ecography*, **32**, 1071-1074.

Fletcher, D. (2012) Estimating overdispersion when fitting a generalized linear model to sparse data. *Biometrika*, **99**, 230-237.

Lebreton, J.-D., Burnham, K. P., Clobert, J., and Anderson, D. R. (1992) Modeling survival and testing biological hypotheses using marked animals: a unified approach with case studies. *Ecological Monographs*, **62**, 67-118.

Pradel, R., Wintrebert, C., and Gimenez, O. (2003) A proposal for a goodness-of-fit test to the Arnason-Schwarz multisite capture-recapture model. *Biometrics*, **59**, 43-53.

Pradel, R., Gimenez, O., and Lebreton, J.-D. (2005) Principles and interest of GOF tests for multistate capture-recapture models. *Animal Biodiversity and Conservation*, **28**, 189-204.

White, G. C. (2002) Discussion comments on: the use of auxiliary variables in capture-recapture modeling. An overview. *Journal of Applied Statistics*, **29**, 103-106.