Software for fitting capture heterogeneity models to estimate the size of a closed population

M. S. Ridout

Institute of Mathematics, Statistics and Actuarial Science
University of Kent
Canterbury, Kent CT2 7NF, UK
email: m.s.ridout@kent.ac.uk

Last updated: August 3, 2007 (Version 1.0)

1 Introduction and outline

These notes describe how to use R software to fit some models of capture heterogeneity to estimate N, the size of a closed population, from capture-recapture data. The models are specific forms of the class of models usually referred to as M_h in the literature and are described in two papers:

PAPER1: Morgan, B.J.T. and Ridout, M.S. (2007) A new mixture model for capture heterogeneity. *need rest of reference*

PAPER2: Morgan, B.J.T. and Ridout, M.S. (2007) Estimating N: a robust approach to capture heterogeneity. *need rest of reference*

The most general model considered is a mixture of a binomial distribution, with probability ϕ and a beta-binomial distribution with parameters μ and θ. The proportion of the mixture arising from the binomial component is γ. This model is proposed as a model of capture heterogeneity in the papers above (though it has been used previously in other applications, see Brooks *et al.*, 1997). The model generalises two commonly used heterogeneity models, the beta-binomial model (Burnham, 1972; Dorazio and Royle, 2003, 2005) and (two-component) finite mixture models (Norris and Pollock, 1996; Pledger, 2000, 2005).

The software allows four different models to be fitted:
`bin` – no heterogeneity, captures follow a binomial distribution (this is model M_0);
`betabin` – captures follow a beta-binomial distribution;
`twobin` – captures follow a mixture of two binomial distributions;
`binbbin` – captures follow a mixture of a binomial and a beta-binomial distribution.

There are important problems of identifiability in models of capture heterogeneity (Link 2003, 2006). The model here does not resolve these problems, but does offer a broad and flexible class of models within which to investigate and compare estimates of N. These issues are discussed in PAPER1 and PAPER2.

The software consists of a series of R functions contained in the file `estimateN.r`, which may be downloaded from

web address here

As with all software of this type, **the software carries no warranty of any sort** – use entirely at your own risk. In particular, these models can sometimes be difficult to fit and there is no guarantee that the reported results correspond to the global maximum of the likelihood function.

The software is intended to be easy to use. For example, the following R commands fit all four models to the skinks data set discussed in PAPER1 and PAPER2.

```
source("estimateN.r")                   # load the functions
skinks = c(56, 19, 28, 18, 24, 14, 9)   # frequencies
kskinks = 7                             # number of sampling occasions
fitallmodels(skinks, kskinks)
```

Alternatively, a function `fitonemodel` is available for fitting a specific model. This provides more control over the fitting process and the output produced. We now describe these two functions in more detail.

2 The function `fitallmodels`

The specification of this function is

```
fitallmodels <- function(freq, kmax)
```

where `freq` is an array such that `freq[k]` is the number of animals seen on k occasions and `kmax` is the total number of sampling occasions.

Typically `freq` will be of length `kmax`, but zeros at the end of the array may be omitted. For example, with the taxicab data discussed in PAPER1 and PAPER2 there were 10 sampling occasions (so `kmax`= 10) with frequencies specified as follows:

```
taxicabs = c(142, 81, 49, 7, 3, 1, 0, 0, 0, 0)
```

No taxis were observed on more than six occasions and a valid alternative specification is

```
taxicabs = c(142, 81, 49, 7, 3, 1)
```

The choice is purely for convenience (useful if there are a large number of zeros at the end of the array); there is no effect on computational efficiency, since any trailing zero frequencies are ignored in the likelihood calculations.

The program will report an error if the specified value of `kmax` is not consistent with the array `freq`. Note that to fit the most complex model requires `kmax` ≥ 5 and again the program will report an error if this condition is not met.

The output produced by the program should be largely self-explantory, but is explained in Section 4.

The program uses multiple random starts in an attempt to find the global maximum of the likelihood function. This is discussed in more detail in Section 5. The `fitallmodels` function uses fixed numbers of starting values of 5, 5, 50 and 50 for the `bin`, `betabin`, `twobin` and `binbbin` models, respectively. If you want to use larger or smaller values, you need to use the `fitonemodel` function described next.

3 The function `fitonemodel`

The specification of this function is rather more involved, because it provides several options. However, most options have default settings and the function remains quite easy to use, as illustrated by the examples in the next section. The full specification is

```
fitonemodel <- function(freq, kmax, model="bin", initparam=NULL,
                  nrandstart=100, plotprofile=FALSE, logn0values=NULL,
                  confpercent=95, nrandprof=10)
```

The `freq` and `kmax` arguments are as described above. The `model` argument should be set to one of `"bin"`, `"betabin"`, `"twobin"` or `"binbbin"`; the default is `"bin"`.

The argument `initparam` provides a means of specifying initial parameter values. However, it is not intended for routine use, and you will need to look at the function `unpack` in the file `estimateN.txt` to see how parameters are specified (the specification varies from model to model). In normal use, this argument should be left unset, in which case the function works from a number of random starting points, specified by the argument `nrandstart`. The model is fitted from each possible starting value, using the Nelder-Mead simplex algorithm, and the best fit overall is reported. Details of how the random starting points are chosen are given below in Section 5.

It is also possible to plot the profile-log-likelihood for n_0 by setting `plotprofile=TRUE`. Here n_0 denotes the number of unseen animals (*not* the total population size N). Typically, the profile is extremely asymmetric about the estimate of n_0, and so in the profile plot, the profile is plotted against $\log(n_0)$. By default, the profile is plotted at 25 equally-spaced points in the range

$$[\log(\hat{n}_0) - 4.605, \log(\hat{n}_0) + 4.605]$$

which is $[\hat{n}_0/100, 100\hat{n}_0]$ on the log-scale. Alternatively, you can supply a set of evaluation points using the option `logn0values`, for example

```
    logn0values = seq(lowvalue, highvalue, length.out=25)
```

for suitably chosen values of `lowvalue` and `highvalue`. See the next section for an example. If you supply values in this way, there is no need to set `plotprofile=TRUE` as well.

Whether or not the profile is plotted, the program calculates a profile-likelihood interval for n_0. The number of observed individuals is then added to the limits, so that the interval that appears in the output is for the total population size, N. The confidence level is determined by the argument `confpercent`; by default 95% intervals are produced. Quite often it may transpire that the lower confidence limit is simply the observed number of individuals and/or the upper limit is infinite. This is discussed in more detail in the next section.

Calculating the profile-log-likelihood function for n_0, whether for plotting or for finding confidence limits, requires optimisation with respect to the remaining parameters and again it can be important use multiple random starting values. However, it is seldom necessary to use as many starting values as in the full model optimisation. The argument `nrandprof` can be used to specify the number required; the default is 20. You should increase this, for example, if the profile appears bumpy, see example below.

As the algorithm runs from random starting points, results will not be reproducible unless you set the random number seed. All the results here were produced by using the command

```
    set.seed(12345)
```

immediately before calling `fitonemodel` or `fitallmodels`.

4 Description of output

The commands:

```
skinks = c(56, 19, 28, 18, 24, 14, 9)
kskinks = 7
fitonemodel( skinks, kskinks, model="twobin", nrandstart=25)
```

produce the following output (see below if it looks different):

```
    *** TWO-BINOMIAL MODEL ***
  Minimised negative log-likelihood value = 23.0394

  Parameter estimates
  Binomial p1 = 0.13 (logit = -1.901 , SE = 0.4524 )
  Binomial p2 = 0.6586 (logit = 0.657 , SE = 0.175 )
  Proportion p1 = 0.3982 (logit = -0.413 , SE = 0.1878 )
  Number of unseen individuals = 48.9 (log = 3.89 , SE = 0.57 )
  Number of individuals seen = 168

  Estimated total population (N)= 216.9
  95 % profile limits for N
  lower = 185.9 upper = 332.5

  Correlation matrix of TRANSFORMED parameters
  (First parameter is log(n0).  Other parameters
  are in the order given above)
              [,1]       [,2]        [,3]         [,4]
  [1,]   1.0000000  -0.7009116  -0.94259344   -0.15762503
  [2,]  -0.7009116   1.0000000   0.78128365   -0.22239173
  [3,]  -0.9425934   0.7812837   1.00000000   -0.01396201
  [4,]  -0.1576250  -0.2223917  -0.01396201    1.00000000

      freq   Fitted
 [1,]   56    52.26
 [2,]   19    26.79
 [3,]   28    17.52
 [4,]   18    23.54
 [5,]   24    26.34
 [6,]   14    16.89
 [7,]    9     4.65
```

Here 25 random starting points is enough to find the global optimum (or at least running with much large numbers of starting points has not found a better fit). Note, however, that there may be small differences in the last few decimal places, because the starting values are random here. Also, the two-binomial model is symmetric and the two binomial parameters may be interchanged in the output, with a resulting change in the `Proportion p1` parameter.

The program works by minimising the negative log-likelihood (excluding some constant terms). The minimised value (here 23.0394) is useful mainly for comparing with alternative models.

Parameter estimates are given next. The optimisation is actually done in terms of transformed parameters (logit or log transformation) as discussed in Section 5 . Consequently, standard errors (obtained by inverting a numerical estimate of the Hessian matrix, using the `hessian=TRUE` setting of `optim`) relate to the transformed parameters. Thus, for example, the estimated logit of the first binomial probability is -1.901, with standard error 0.4524, giving a value of `p1`=0.13. The program attempts to print estimates to sensible numbers of decimal places, but sometimes `R` imposes some additional rounding, as here where the estimates of `p1` and `p2` have different numbers of decimal places.

For the more complicated models, the likelihood surface can be complicated and standard errors may be of little value. Quite often the numerically inverted Hessian matrix is not positive definite or is highly ill-conditioned. In these cases a warning message is printed and no standard errors are produced.

The estimate of n_0 is added to the number of distinct individuals seen to give the estimate of N and a confidence interval is given, based on the profile-likelihood. Whilst the profile-likelihood necessarily decreases away from the maximum, it may not decrease sufficiently for profile log-likelihood estimates of n_0, and hence N to exist at the required confidence level. The program only checks in the range $[\hat{n}_0/100, 100\hat{n}_0]$. If the lower limit does not exist in this range the program simply gives the number of individuals seen, with a comment to this effect. If the upper limit does not exist it is given as

```
infinity (probably)
```

The 'probably' is because the search is only to $100\hat{n}_0$. But in many cases there is genuinely no finite limit. Refer to PAPER2 for a detailed discussion of this issue.

The program also prints the correlation matrix of parameter estimates (provided that the Hessian matrix is positive-definite and not too poorly conditioned). These relate to the *transformed* parameters and the order is as earlier in the output except that `logn0` is now the first parameter rather than the last.

The final output gives fitted values. These are calculated by taking the fitted model for the number of times an animal is seen, conditioning on being seen at least once and then multiplying by the total number of animals actually seen.

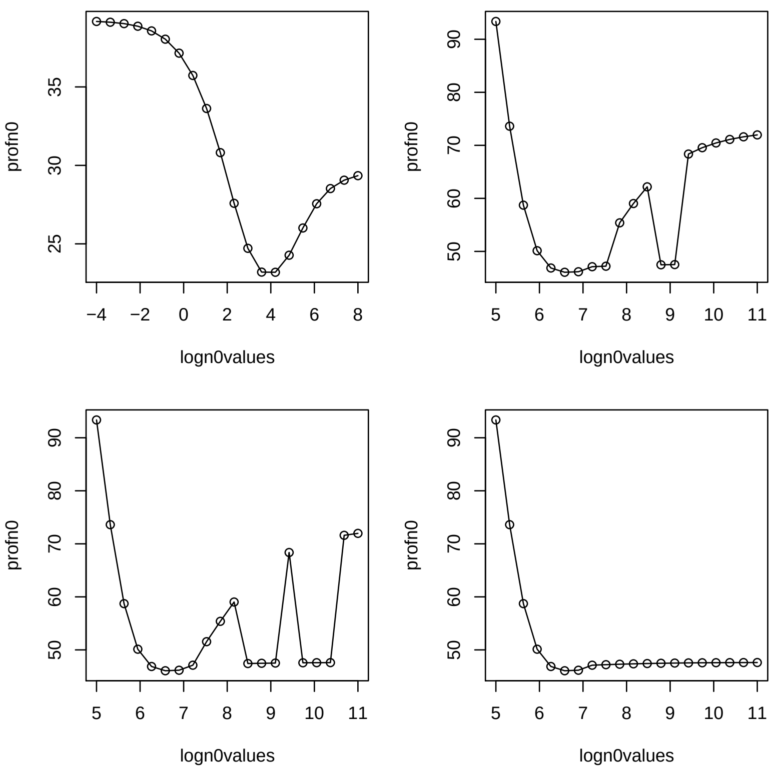
The following R commands produce the profile plots shown below:

```
par(mfrow=c(2,2), mar=c(4.5,4,1.5,1.5), oma=c(1.25,1.25,1.25,1.25))
set.seed(12345)
fitonemodel(skinks, kskinks, model="twobin", nrandstart=20,
             logn0values=seq(-4,8,length.out=20), nrandprof=10)
set.seed(12345)
fitonemodel(link3, klink3, model="binbbin", nrandstart=20,
             logn0values=seq(5,11,length.out=20), nrandprof=10)
set.seed(12345)
fitonemodel(link3, klink3, model="binbbin", nrandstart=20,
             logn0values=seq(5,11,length.out=20), nrandprof=20)
set.seed(12345)
fitonemodel(link3, klink3, model="binbbin", nrandstart=20,
             logn0values=seq(5,11,length.out=20), nrandprof=50)
```

The top-left figure is a profile log-likelihood plot for the mixture of two binomial models fitted earlier to the skinks data. Using just 10 random starting points for the profile (the default is `nrandprof=20`) appears to give a satisfactory profile.

The remaining figures are for the Link3 data set discussed in PAPER 2 and given originally by Link (2003, Table 3). The model fitted is the mixture of a binomial and a beta-binomial model. The graphs differ as a result of the different setting of the `nrandprof` option. Using `nrandprof=10` (top right) or `nrandprof=20` (bottom left) fails to give a smooth profile, but the profile is smooth with `nrandprof=50` (bottom right). Note that the smoothness may depend on the particular points at which the profile is calculated as well as on the setting of `nrandprof` and the setting of the random number seed.

This has important implications for calculated profile intervals of N. If too small a value of `nrandprof` is used, the program may generate a finite upper confidence limit, whereas the correct estimate is infinite in this example. However, setting a large value slows the program down considerably. The default setting of `nrandprof=20` is intended to be a compromise. For the 16 data sets considered in PAPER 1 and PAPER 2, only Link3 and the Golftees fail to give smooth profiles with the default settings of `nrandprof`, the default choice of points at which to plot the profile (i.e. with `logn0values` unset) and if `set.seed(12345)` is used.



5 Computational details

5.1 Optimisation method

For the two-binomial and binomial/beta-binomial mixture models of heterogeneity the likelihood surface is typically difficult to optimise. To try to find a global optimum, the Nelder-Mead simplex algorithm is run from multiple starting points using the R function `optim`. Note that `optim` produces warning messages when used for 1-dimensional optimisation, recommending instead the `optimise` function. This is relevant to profile likelihood estimation in the binomial model. Because `optim` appears to perform perfectly satisfactorily in this case, I have continued to use it and simply switched off warning messages temporarily while this option is running.

5.2 Parameterisation

The model uses four parameters `pbin`, `pbbin`, `theta` and `alphabin`. In terms of the notation in PAPER1 and PAPER2, these correspond to ϕ, μ, θ and $1 - \gamma$, respectively.

Note in particular the last of these. The remaining parameter is `n0`, the number of unseen individuals.

Because the parameters have to satisfy constraints but the optimisations used are unconstrained, the model works in terms of transformed parameters `logit(pbin)`, `logit(pbbin)`, `log(theta)`, `logit(alphabin)` and `log(n0)`.

5.3 Choice of random starting points

The parameters `pbin`, `pbbin` and `alphabin` are constrained to lie in $[0, 1]$ and so random $U[0, 1]$ variables are used as starting values.

The remaining two parameters, `theta` and `n0` are constrained to be positive. The variance of the beta-binomial distribution is given by the expression

$$K\mu(1-\mu)\left[1+(K-1)\frac{\theta}{1+\theta}\right],$$

where K is the number of sampling occasions (`kmax` in the program). The factor $\theta/(1+\theta)$ lies in the interval $[0, 1)$ and we therefore choose a random $U[0, 1]$ variable as the starting value for $\theta/(1+\theta)$.

To choose a random starting value for n_0, we choose

$$\frac{n_0}{n_0 + n_0^{init}}$$

to have a $U[0, 1]$ distribution, where n_0^{init} is an initial estimate of n_0. Thus the generated values have equal probability of being greater than or less than n_0^{init}. For `model="bin"`, we choose n_0^{init} to be the Good-Turing estimator of the number of unseen species (Good, 1953), whilst for other models we use another coverage-based estimator recommended by Ashbridge and Goudie (2000). A further minor modification is that very large values of n_0 are excluded. Specifically if $n_0 > \exp(15) = 3269017.3$ it is replaced by this upper limit.

5.4 Error checking

The program checks for a few specific errors or inconsistencies in data input and stops the program with an appropriate error message. However, there is no attempt to check for all conceivable input errors systematically and incorrect usage may cause the program to crash.

5.5 Version number

The file estimateN.r includes a version() function, which displays the current version number:

```
> version()
[1] "This is Version 1.0 of estimateN.r (1 Aug 2007)"
```

5.6 Timing

Computer time used depends on various factors, including the number of sampling occasions, the model being fitted and the number of random starts used. However, because multiple starting points are used, it may take some time to fit models.

The software was developed on a 3.19 GHz Pentium 4 PC with 1 GB RAM, running version 2.3.1 of R. With this set up, fitting the two-binomial model to the skinks data took about 12 seconds of computing time whilst producing the four profile plots took about 310 seconds.

6 References

Ashbridge, J. and Goudie, I.B.J. (2000). Coverage-adjusted estimators for mark-recapture in heterogeneous populations. *Communications in Statistics – Simulation and Computation*, **29**, 1215-1237.

Brooks, S.P., Morgan, B.J.T., Ridout, M.S. and Pack, S.E. (1997). Finite mixture models for proportions. *Biometrics*, **53**, 1097-1115.

Burnham, K.P. (1972). Estimation of population size in multiple capture-recapture studies when capture probabilities vary among animals. PhD thesis, Oregon State University, Corvallis.

Dorazio, R.M. and Royle, J.A. (2003). Mixture models for estimating the size of a closed population when capture rates vary among individuals. *Biometrics*, **59**, 351-364.

Dorazio, R.M. and Royle, J.A. (2005). Rejoinder to "The performance of mixture models in heterogeneous closed populations". *Biometrics*, **61**, 874-876.

Good, I.J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, **40**, 237-264.

Link, W.A. (2003). Nonindentifiability of population size from capture-recapture data with heterogeneous detection probabilities. *Biometrics*, **59**, 1123-1130.

Link, W.A. (2006). Rejoinder to "On identifiability in capture-recapture models". *Biometrics*, **62**, 936-939.

Norris, J.L. and Pollock, K.H. (1996). Nonparametric mle under two closed capture-recapture models with heterogeneity. *Biometrics*, **52**, 639-649.

Pledger, S. (2000). Unified maximum likelihood estimates for closed capture-recapture models using mixtures. *Biometrics*, **56**, 434-442.

Pledger, S. (2005). The performance of mixture models in heterogeneous closed population capture-recapture. *Biometrics*, **61**, 868-873.