Fitting Cormack-Jolly-Seber Models with Individual Heterogeneity to Capture-Recapture Data with ADMB and R2admb

Jeff Laake

December 2, 2012

Cormack-Jolly-Seber models are often used to estimate survival from release-recapture data. Some details are provided in `https://github.com/downloads/jlaake/ADMB-Examples/CJSFixedEffects.pdf`. This is the abreviated version which outlines a model which includes individual heterogeneity through a random effect. Let ω_i be the capture history for the i^{th} animal which is a vector of m values $(\omega_{i1}, \omega_{i2}, ..., \omega_{im})$ where ω_{ij} is 1 if animal i was intially released or recaptured on occasion j and 0 otherwise. Let f_i be the occasion the animal was first released and l_i the last occasion the animal was seen. The CJS log-likelihood is:

$$lnL = \sum_{i=1}^{n} ln(Pr(\omega_i))$$

where ignoring loss on captures

$$Pr(\omega_i) = \sum_{d=l_i}^{m} \prod_{j=f_i+1}^{d} p_{ij}^{\omega_{ij}}(1-p_{ij})^{(1-\omega_{ij})} \left(\prod_{j=f_i}^{d-1} \phi_{ij} \right) (1-\phi_{id}) \quad ,$$

which depends on the parameters p_{ij} for capture-probability and ϕ_{ij} for survival. Consider a model in which both capture and survival are subject to individual heterogeneity which was proposed by Gimeinez and Choquet (2010). I'll use a logit link for both parameters:

$$log(\phi_{ij}/(1-\phi_{ij})) = \mu_\phi + \varepsilon_{\phi i}$$

$$log(p_{ij}/(1-p_{ij})) = \mu_p + \varepsilon_{pi}$$

where ε_{pi} are iid $N(0, \sigma_p^2)$ and $\varepsilon_{\phi i}$ are iid $N(0, \sigma_\phi^2)$ and the two sources of error are independent. Then the log-likelihood is

$$lnL(\mu_p, \mu_\phi, \sigma_p^2, \sigma_\phi^2, p|\omega_1, \omega_2, ..., \omega_n) = \sum_{i=1}^{n} \log \left[\int_{\varepsilon_p} \int_{\varepsilon_\phi} Pr(\omega_i|\varepsilon_p, \varepsilon_\phi) d\varepsilon_p d\varepsilon_\phi \right]$$

It is also possible to create models with both temporal and individual heterogeneity. Consider a model with temporal and invididual heterogeneity in survival but for simplicity constant capture probability. Let

$$log(\phi_{ij}/(1 - \phi_{ij})) = \mu + \varepsilon_i + \gamma_j$$

where ε_i are iid $N(0, \sigma_\varepsilon^2)$ and γ_i are iid $N(0, \sigma_\gamma^2)$ and the two sources of error are independent. Then the likelihood is

$$L(\mu, \sigma_\varepsilon^2, \sigma_\gamma^2, p|\omega_1, \omega_2, ..., \omega_n) = \int \cdots \int_{\gamma_1, \ldots \gamma_K} \left[\prod_{i=1}^{n} \int_{\varepsilon} Pr(\omega_i|\gamma, \varepsilon) d\varepsilon \right] d\gamma_1 ... d\gamma_K \quad .$$

Here I provide an example with individual heterogeneity only and fit the same model with the implementation of the Gimeinez and Choquet (2010) model in MARK (White and Burnham, 1999)using the RMark interface (Laake and Rexstad, 2008). The example uses admbcjsre.tpl file shown at the end of this document. The structure is similar to the fixed effects CJS TPL file (admbcjs.tpl as described in `https://github.com/downloads/jlaake/ADMB-Examples/CJSFixedEffects.pdf`) but the differences are as follows:

- definition of random_effects_vectors, pu and phiu, for p_{ij} and ϕ_{ij} and parameters for the standard deviations of the random effect distributions (psigeps and phisigeps respectively)
- the likelihood contribution for each animal is computed in a SEPARABLE_FUNCTION ll_i and the PROCEDURE_SECTION is a single loop over i=1,n that calls ll_i.
- the likelihood includes standardized normal components for each random effect

Below is some R code that simulates data with random effects in both p_{ij} and ϕ_{ij} and then compiles and runs the ADMB program and the MARK equivalent on the same data.

```
> # Number of histories
> n=500
> # Number of occasions
> nocc=10
> # Matrix of survival events - fixed survival
> alive=matrix(rbinom(n*(nocc-1),1,.9),ncol=nocc-1)
> # Cummulative survival events (once dead stays dead)
> alive=t(apply(alive,1,cumprod))
> # Individual variation in capture probability
> p=plogis(rnorm(n,0,1))
> # Capture events
> seen=matrix(rbinom(n*(nocc-1),1,rep(p,each=nocc-1)),byrow=T,ncol=nocc-1)
> # Construct capture history matrix for a single release cohort
> chmat=cbind(rep(1,n),seen*alive)
> ch=apply(chmat,1,paste,collapse="")
> # Create dataframe with ch values
> test=data.frame(ch=ch,stringsAsFactors=FALSE)
> # detach RMark if attached and attach marked package
> if(length(grep("RMark",.packages()))!=0)detach("package:RMark")
> library(marked)
> # process data frame but don't accumulate same capture histories
> test.proc=process.data(test,model="cjs",begin.time=1,accumulate=F)
> # create design data
> test.ddl=make.design.data(test.proc)
> # Fit model
> admb.mod=crm(test.proc,test.ddl,use.admb=TRUE,re=TRUE,
+               compile=TRUE,extra.args="-gh 15")

Computing initial parameter estimates

Elapsed time in minutes:  3.7282

> # Show results
> admb.mod

crm Model Summary

Npar :  4
-2lnL:  4609.4
```

```
AIC  :  4617.4

Beta
                  Estimate           se           lcl          ucl
Phi.(Intercept)  2.2182100 8.065957e-02     2.0601172    2.3763028
p.(Intercept)   -0.0690131 8.846265e-02    -0.2423999    0.1043737
sigma_phisigeps -8.2733303 1.159656e+03 -2281.1996743 2264.6530136
sigma_psigeps    0.0780586 9.691427e-02    -0.1118934    0.2680106

> # Now fit the same model with MARK via RMark
> # detach marked and attach RMark
> detach("package:marked")
> library(RMark)
> # process same dataframe for RMark
> test.proc=process.data(test,model="CJSRandom",begin.time=1)
> # make design data (different format from marked)
> test.ddl=make.design.data(test.proc)
> # fit model
> mark.mod=mark(test.proc,test.ddl,output=FALSE)

Note: only  3  parameters counted of  4  specified parameters
AICc and parameter count have been adjusted upward

> # show results
> summary(mark.mod,brief=TRUE)

Output summary for CJSRandom model
Name : sigmaphi(~1)Phi(~1)sigmap(~1)p(~1)

Npar :  4  (unadjusted=3)
-2lnL:  4609.381
AICc :  4617.403  (unadjusted=4615.3942)

Beta
                      estimate          se           lcl           ucl
sigmaphi:(Intercept) -7.6548524 635.1367700 -1252.5229000 1237.2132000
Phi:(Intercept)       2.2185146   0.0806559     2.0604290    2.3766002
sigmap:(Intercept)    0.0790210   0.0970480    -0.1111930    0.2692351
p:(Intercept)        -0.0692704   0.0883149    -0.2423675    0.1038268
```

```
> # cleanup files
> cleanup(ask=FALSE)
> clean_admb("admbcjsre")
```

There are slight differences that are probably due to slight numerical differences but may deserve further exploration.

A mixed-effects model with individual heterogeneity can be created by specifying a formula for p and ϕ. Below is an example using some simulated data with a fixed sex effect as well as individual variation in p.

```
> # Number of histories
> n=500
> # Number of occasions
> nocc=10
> # Matrix of survival events - fixed survival
> alive=matrix(rbinom(n*(nocc-1),1,.9),ncol=nocc-1)
> # Cummulative survival events (once dead stays dead)
> alive=t(apply(alive,1,cumprod))
> # Individual variation in capture probability - with fixed sex effect
> p=plogis(rnorm(n,0,1)+0.5*c(rep(0,n/2),rep(1,n/2)))
> # Capture events
> seen=matrix(rbinom(n*(nocc-1),1,rep(p,each=nocc-1)),byrow=T,ncol=nocc-1)
> # Construct capture history matrix for a single release cohort
> chmat=cbind(rep(1,n),seen*alive)
> ch=apply(chmat,1,paste,collapse="")
> # Create dataframe with ch values
> test=data.frame(ch=ch,sex=c(rep(0,n/2),rep(1,n/2)),stringsAsFactors=FALSE)
> # detach RMark if attached and attach marked package
> if(length(grep("RMark",.packages()))!=0)detach("package:RMark")
> library(marked)
> # process data frame but don't accumulate same capture histories
> test.proc=process.data(test,model="cjs",begin.time=1,accumulate=F)
> # create design data
> test.ddl=make.design.data(test.proc)
> # Fit model
> admb.mod=crm(test.proc,test.ddl,
+              model.parameters=list(p=list(formula=~sex)),
+              use.admb=TRUE,re=TRUE,compile=TRUE,extra.args="-gh 15")
```

```
Computing initial parameter estimates

Elapsed time in minutes:  2.7915

> # Show results
> admb.mod

crm Model Summary

Npar :  5
-2lnL:  4734.5
AIC  :  4744.5

Beta
                   Estimate         se         lcl       ucl
Phi.(Intercept)  2.37821000 0.10082553  2.18059196 2.5758280
p.(Intercept)    0.02145630 0.11461941 -0.20319774 0.2461103
p.sex            0.36696200 0.15958039  0.05418443 0.6797396
sigma_phisigeps -0.41062785 0.55561540 -1.49963404 0.6783783
sigma_psigeps    0.08738925 0.09466288 -0.09815000 0.2729285

> # Now fit the same model with MARK via RMark
> # detach marked and attach RMark
> detach("package:marked")
> library(RMark)
> # process same dataframe for RMark
> test.proc=process.data(test,model="CJSRandom",begin.time=1)
> # make design data (different format from marked)
> test.ddl=make.design.data(test.proc)
> # fit model
> mark.mod=mark(test.proc,test.ddl,
+          model.parameters=list(p=list(formula=~sex)),
+                  output=FALSE)
> # show results
> summary(mark.mod,brief=TRUE)

Output summary for CJSRandom model
Name : sigmaphi(~1)Phi(~1)sigmap(~1)p(~sex)
```

```
Npar :  5
-2lnL:  4734.495
AICc :  4744.524

Beta
                       estimate        se        lcl       ucl
sigmaphi:(Intercept) -0.4112628 0.5564621 -1.5019285 0.6794030
Phi:(Intercept)       2.3782444 0.1008287  2.1806202 2.5758685
sigmap:(Intercept)    0.0877046 0.0948733 -0.0982470 0.2736563
p:(Intercept)         0.0213636 0.1143676 -0.2027968 0.2455240
p:sex                 0.3669492 0.1595785  0.0541754 0.6797230

> # cleanup files
> cleanup(ask=FALSE)
> clean_admb("admbcjsre")
```

References

Gimeinez, O. and Choquet, R. (2010). Individual heterogeneity in studies on marked animals using numerical integration: capture-recapture mixed models. *Ecology*, 91:951–957.

Laake, J. and Rexstad, E. (2008). RMark – an alternative approach to building linear models in MARK. In Cooch, E. and White, G. C., editors, *Program MARK: A Gentle Introduction*.

White, G. C. and Burnham, K. P. (1999). Program MARK: survival estimation from populations of marked animals. *Bird Study*, 46:120–139.

```
// Cormack-Jolly-Seber model with individual random effects for survival (Phi) and capture probability (p)
// Each capture history must represent a single animal
// Jeff Laake; 15 Nov 2012
DATA_SECTION
    init_int n;                         // number of capture histories
    init_int m;                         // number of capture occasions
    init_imatrix ch(1,n,1,m);           // capture history matrix
    init_ivector frst(1,n);             // occasion first seen for each history
    init_ivector lst(1,n);              // occasion last seen for each history
    init_ivector loc(1,n);              // 0 or 1, 1 if lost on capture at last event
    init_matrix tint(1,n,1,m-1);        // time interval between occasions for each history-interval
    init_int kphi;                      // number of columns in the design matrix for Phi - survival
    int nrows;                          // number of entries in design matrix m-1 values for each of n histories
    !! nrows=n*(m-1);
    init_matrix phidm(1,nrows,1,kphi);  // design matrix for Phi
    init_int kp;                        // number of columns in the design matrix for p - capture probability
    init_matrix pdm(1,nrows,1,kp);      // design matrix for p
    init_int K;                         // number of fixed Phi values
    init_matrix PhiF(1,K,1,2);          // Phi fixed matrix with index in first column and value in second column
    init_int L;                         // number of fixed p values
    init_matrix pF(1,L,1,2);            // p fixed matrix with index in first column and value in second column

PARAMETER_SECTION
    init_vector phibeta(1,kphi,1);        // parameter vector for Phi
    init_vector pbeta(1,kp,1);            // parameter vector for p
    init_number phisigeps(2);
                                        // sigma for random effect in phi;
    init_number psigeps(2);
                                        // sigma for random effect in p;
    objective_function_value f;         // objective function - negative log-likelihood
    random_effects_vector phiu(1,n,2);    // random effect for scale
    random_effects_vector pu(1,n,2);      // random effect for scale

TOP_OF_MAIN_SECTION
  arrmblsize=5000000;                   // not sure how to set this as a function of problem size

PROCEDURE_SECTION
    int i;                              // index over observations
    for(i=1;i<=n;i++)                   // loop over capture histories - one per animal
    {
          ll_i(i,phisigeps,psigeps,phiu(i),pu(i),phibeta,pbeta);
    }

SEPARABLE_FUNCTION void ll_i(const int i, const dvariable& phisigeps,const dvariable& psigeps,const dvariable& phiu,const dvariable& pu,const dvar_v
    dvar_vector phi(1,m);               // temp vector for Phis for each occasion for a single history
    dvar_vector p(1,m-1);               // temp vector for Phis for each occasion for a single history
    dvar_vector phicumprod(1,m);        // cummulative survival probability across occasions
    dvar_vector cump(1,m);              // cummulative probability of being seen across occasions
    dvariable pch;                      // probability of capture history
    int i1,i2,j;                        // miscellaneous ints
    dvariable phieps=phiu*exp(phisigeps);   // scaled re for phi
    dvariable peps=pu*exp(psigeps);         // scaled re for p
                                                            //
    phi=0;                                                  // set all phi values to 0
    phicumprod=1.0;                                         // set cummulative survival to 1
    cump=1.0;                                               // set cummulative p to 1
    i1=(m-1)*(i-1);                                         // compute beginning index in design matrix
    for(j=frst(i)+1;j<=m;j++)                               // loop over occasions from frst to m
    {
       i2=i1+j;                                             // increment index in design matrix
```

```
   phi(j-1)=1/(1+exp(-phidm(i2-1)*phibeta-phieps));              // compute phi for the interval
   if(tint(i,j-1)!=1)
      phi(j-1)=pow(phi(j-1),tint(i,j-1));                        // adjust phi for the time interval length
   p(j-1)=1/(1+exp(-pdm(i2-1)*pbeta-peps));                      // compute p for the occasion
   phicumprod(j)=phicumprod(j-1)*phi(j-1);                       // compute cummulative survival
   cump(j)=cump(j-1)*((1-p(j-1))*(1-ch(i,j))+p(j-1)*ch(i,j));    // compute cummulative capture probability
}
pch=0.0;                                                         // initialize prob of the capture history to 0
for(j=lst(i);j<=((1-loc(i))*m+loc(i)*lst(i));j++)                // loop over last occasion to m unless loss on capture
{                                                                //  to compute prob of the capture history
   i2=i1+j;                                                      // index occasion
   if(loc(i)==1)
      pch=pch+cump(j)*phicumprod(j);                             // probability of history given possible last occasion alive
   else
      pch=pch+cump(j)*phicumprod(j)*(1-phi(j));                  // probability of history given possible last occasion alive
}
f-= log(pch+.0000000000000001);                                  // sum log-likelihood log(pr(ch))
f -= -0.5*square(pu)-0.9189385332046727;                         // normal random effect distr; constant is log(sqrt(2*pi))
f -= -0.5*square(phiu)-0.9189385332046727;                       // normal random effect distr; constant is log(sqrt(2*pi))
```