Estimating abundance in open populations using capture-recapture models

Olivier Gimenez

August 8, 2016

Introduction

Lately, I have found myself repeating the same analyses again and again to estimate population size from capture-recapture models, and the Cormack-Jolly-Seber (CJS) model in particular. I do not intend here to provide extensive details on this model and its variants. It is just a basic attempt to put together some R code to avoid spending hours digging up in my files how to do this analysis.

I use RMark because everything can be done in R, and it's cool for reproducible research. But other pieces of software are fine too. I consider simple CJS models and models with transience. In passing, I also fit models with heterogeneity in the detection process with an individual random effect. The bootstrap is used to obtain confidence intervals.

I have ignored multi-model inference for simplicity. However the bootstrap can be used to perform model selection (e.g., Buckland et al. 1997).

The data and codes are part of a manuscript that is currently in review:

> Chiara G. Bertulli, Loreleï Guéry, Niall McGinty, Ailie Suzuki, Naomi Brannan, Tania Marques, Marianne H. Rasmussen, Olivier Gimenez (in review). Abundance estimation of photographically identified common minke whales, white-beaked dolphins and humpback whales in Icelandic coastal waters using capture-recapture methods.

Abundance estimates from a Cormack-Jolly-Seber model

First, let's load the RMark package for analysing capture-recapture data by calling MARK from R.

```
library(RMark)
```

The data come from an opportunistic monitoring through which photos were taken of Humpback whales in the Icelandic waters. More details in Chiara's paper cited above. Each row is an individual that has been detected (coded as a 1) or non-detected (coded as a 0) through photo-identification over the years in columns. Have a look to the file in your favorite text editor. Other formats are fine.

```
hw.dat = import.chdata("humpbackwhaleMaySep20062013.txt", header = F, field.names = c("ch"), field.type
summary(hw.dat)
```

```
##       ch
##  Length:195
##  Class :character
##  Mode  :character
```

```
attach(hw.dat)
```

Now it is time to build our model. We're gonna use a standard Cormack-Jolly-Seber model. I have carried out goodness-of-fit tests before and found that everything was OK.

```r
hw.proc = process.data(hw.dat, model="CJS")
hw.ddl = make.design.data(hw.proc)
```

Then we specify the effects we'd like to consider on survival and detection probabilities.

```r
# survival process
Phi.ct = list(formula=~1) # constant
Phi.time = list(formula=~time) # year effect
# detection process
p.ct = list(formula=~1) # constant
p.time = list(formula=~time) # year effect
```

Let's roll and run four models with or without a year effect!

```r
# constant survival, constant recapture
Model.1 = mark(hw.proc,hw.ddl,model.parameters=list(Phi=Phi.ct,p=p.ct),output = FALSE,delete=T)
# constant survival, time-dependent recapture
Model.2 = mark(hw.proc,hw.ddl,model.parameters=list(Phi=Phi.ct,p=p.time),output = FALSE,delete=T)
# time-dependent survival, constant recapture
Model.3 = mark(hw.proc,hw.ddl,model.parameters=list(Phi=Phi.time,p=p.ct),output = FALSE,delete=T)
# time-dependent survival, time-dependent recapture
Model.4 = mark(hw.proc,hw.ddl,model.parameters=list(Phi=Phi.time,p=p.time),output = FALSE,delete=T)
```

Let's have a look to the AIC for these models.

```r
summary(Model.1)$AICc
```

```
## [1] 317.4295
```

```r
summary(Model.2)$AICc
```

```
## [1] 311.3483
```

```r
summary(Model.3)$AICc
```

```
## [1] 313.1428
```

```r
summary(Model.4)$AICc
```

```
## [1] 322.0564
```

For convenience, we will say that `model 2` is the model best supported by the data, the one with constant survival probability and time-dependent recapture probability. Multi-model selection would be more appropriate here. Let's have a look to the parameter estimates: survival, then recapture probabilities estimates.

```
phitable = get.real(Model.2,"Phi", se= TRUE)
# names(phitable)
phitable[c("estimate","se","lcl","ucl")][1,]
```

```
##                    estimate         se       lcl       ucl
## Phi g1 c1 a0 t1 0.5234883 0.0565078 0.4133877 0.6313525
```

```
ptable = get.real(Model.2,"p", se= TRUE)
ptable[c("estimate","se","lcl","ucl")][1:7,]
```

```
##                  estimate         se       lcl       ucl
## p g1 c1 a1 t2 0.6814076 0.2413587 0.1948420 0.9497575
## p g1 c1 a2 t3 0.1515677 0.1041699 0.0352270 0.4663918
## p g1 c1 a3 t4 0.4246961 0.1541836 0.1764801 0.7177504
## p g1 c1 a4 t5 0.2866319 0.1204953 0.1123644 0.5605063
## p g1 c1 a5 t6 0.3746889 0.1538311 0.1419705 0.6845395
## p g1 c1 a6 t7 0.4332827 0.1215465 0.2246677 0.6685710
## p g1 c1 a7 t8 0.8383463 0.1685128 0.3119192 0.9834245
```

Now it's easy to estimate abundance estimates by calculating the ratios of the number of individuals detected at each occasion over the corresponding estimate of recapture probability. Note that we estimate **re**capture probabilities, so that we cannot estimate abundance on the first occasion.

```
# calculate the nb of recaptured individiduals / occasion
obs = gregexpr("1", hw.dat$ch)
n_obs = summary(as.factor(unlist(obs)))
estim_abundance = n_obs[-1]/ptable$estimate[1:7]
estim_abundance
```

```
##         2         3         4         5         6         7         8
##  33.75366  92.36796  58.86562  45.35434  93.41083 122.32199  90.65466
```

We use a boostrap approach to get an idea of the uncertainty surrounding these estimates, in particular to obtain the confidence intervals.

We first define the number of bootstrap iterations (10 here for the sake of illustration, should be 500 instead, or even 1000 if the computational burden is not too heavy), the number of capture occasions and format the dataset in which we'd like to resample (with replacement). This is non-parametric bootstrap (and alternative is parametric bootstrap where data are simulated using the model estimates). We also define a matrix `popsize` in which we will store the results, and we define the seed for simulations (to be able to replicate the results).

```
nb_bootstrap = 10
nb_years = 8
target = data.frame(hw.dat,stringsAsFactors=F)
popsize = matrix(NA,nb_bootstrap, nb_years-1)
set.seed(5)
pseudo = target # initialization
```

Finally, we define the model structure and the effects on parameter (same for all bootstrap samples).

```
# define model structure
hw.proc = process.data(pseudo, model="CJS")
hw.ddl = make.design.data(hw.proc)
# define parameter structure
phi.ct = list(formula=~1)
p.time = list(formula=~time)
```

Let's run the bootstrap now:

```
for (k in 1:nb_bootstrap){
  # resample in the original dataset with replacement
  pseudo$ch = sample(target$ch, replace=T)
  # fit model with Mark
  res = mark(hw.proc,hw.ddl,model.parameters=list(Phi=phi.ct,p=p.time),delete=TRUE,output=FALSE)
  # get recapture prob estimates
  ptable = get.real(res,"p", se= TRUE)
  # calculate the nb of recaptured individiduals / occasion
  allobs = gregexpr("1", pseudo$ch)
  n = summary(as.factor(unlist(allobs)))
  popsize[k,] <- n[-1]/ptable$estimate[1:(nb_years-1)]
}
```

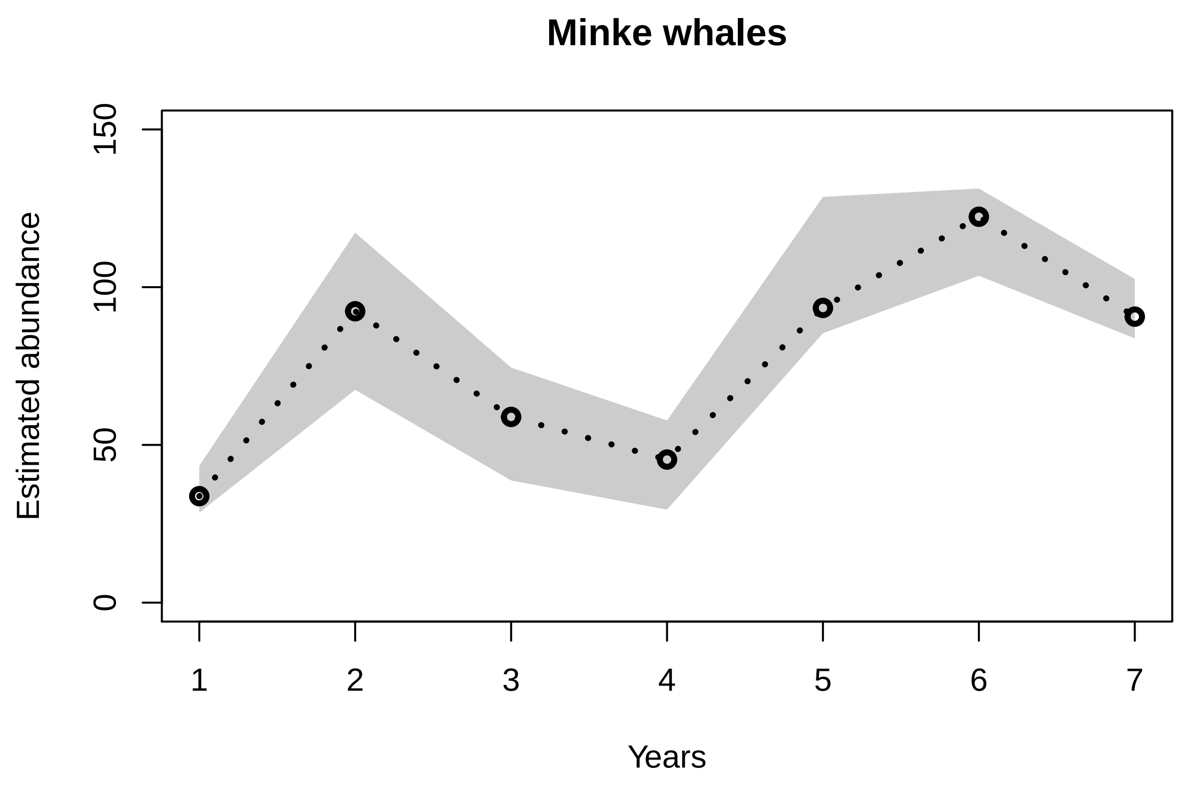
Now we can get confidence intervals:

```
ci_hw = apply(popsize,2,quantile,probs=c(2.5/100,97.5/100),na.rm=T)
ci_hw
```

```
##            [,1]      [,2]     [,3]     [,4]      [,5]     [,6]      [,7]
## 2.5%  28.54385  67.46159 38.73358 29.48032  85.40419 103.6275  83.76609
## 97.5% 43.36612 117.27433 74.52388 57.73954 128.64006 131.2653 102.58290
```

A plot:

```
plot(1:(nb_years-1),estim_abundance, col="black", type="n", pch=21, xlab="Years", lty=3, ylab="Estimate
polygon(c(rev(1:(nb_years-1)), 1:(nb_years-1)), c(rev(ci_hw[2,]), ci_hw[1,]), col = 'grey80', border = 
lines(1:(nb_years-1), estim_abundance, col="black",lty=3,type='o',lwd=3,pch=21)
```



What if transience occurs?

We now analyse data on Minke whales.

```r
library(RMark)
mw.dat = import.chdata("minkewhalesAprAug20082013.txt", header = F, field.names = c("ch"), field.types =
summary(mw.dat)
```

```
##       ch
##  Length:191
##  Class :character
##  Mode  :character
```

```r
attach(mw.dat)
```

The goodness-of-fit tests showed that Test3SR was significant, hence a transient effect due to true transient individuals, an age effet or a bit of both. To account for this effect, we use a two-age class structure on survival.

```r
mw.proc = process.data(mw.dat, model = "CJS")
mw.ddl = make.design.data(mw.proc)
mw.ddl = add.design.data(mw.proc, mw.ddl,"Phi", type = "age", bins = c(0,1,6), name = "ageclass", right
```

Then we specify the effects on survival and detection probabilities. Age is always included. We consider time-dependent variation or not on both survival and recapture probabilities.

```r
# survival process
phi.age = list(formula=~ageclass)
phi.ageptime = list(formula=~ageclass+time)
# detection process
p.ct = list(formula=~1)
p.time = list(formula=~time)
```

Let’s roll and run models with or without a year effect!

```r
Model.1 = mark(mw.proc, mw.ddl, model.parameters = list(Phi = phi.age, p = p.ct),output = FALSE,delete=
Model.2 = mark(mw.proc, mw.ddl,model.parameters = list(Phi = phi.age, p = p.time),output = FALSE,delete
Model.3 = mark(mw.proc, mw.ddl,model.parameters = list(Phi = phi.ageptime, p = p.ct),output = FALSE,dele
Model.4 = mark(mw.proc, mw.ddl,model.parameters = list(Phi = phi.ageptime, p = p.time),output = FALSE,de
```

Let’s have a look to the AIC for these models.

```r
summary(Model.1)$AICc
```

```
## [1] 480.6304
```

```r
summary(Model.2)$AICc
```

```
## [1] 486.8038
```

```r
summary(Model.3)$AICc
```

```
## [1] 485.1968
```

```r
summary(Model.4)$AICc
```

```
## [1] 493.2464
```

For simplicity here, we will say that `model 1` is the model that is best supported by the data, the one with constant survival and recapture probabilities. Let’s have a look to the parameter estimates: survival, then recapture probabilities estimates.

```r
phitable = get.real(Model.1,"Phi", se= TRUE)
# names(phitable)
phitable[c("estimate","se","lcl","ucl")][1:2,]
```

```
##                   estimate        se       lcl       ucl
## Phi g1 c1 a0 t1 0.4313735 0.0507980 0.3357811 0.5323679
## Phi g1 c1 a1 t2 0.7976824 0.0513587 0.6787706 0.8803361
```

On the first row, the survival is for both transient and resident individuals. On the second row, this is survival for resident individuals.

```
ptable = get.real(Model.1,"p", se= TRUE)
ptable[c("estimate","se","lcl","ucl")][1,]
```

```
##                  estimate        se       lcl      ucl
## p g1 c1 a1 t2 0.5353637 0.0536269 0.4302436 0.637433
```

An estimate of abundance is obtained as in the previous section:

```
# calculate the nb of recaptured individiduals / occasion
obs = gregexpr("1", mw.dat$ch)
n_obs = summary(as.factor(unlist(obs)))
estim_abundance = n_obs[-1]/ptable$estimate[1]
estim_abundance
```

```
##         2         3         4         5         6
##  72.84767 115.80912  98.99812  76.58345  78.45134
```

We use a boostrap approach to get an idea of the uncertainty surrounding these estimates, in particular to obtain the confidence intervals. The bootstrap approach was proposed by Madon et al. (2013). Roger Pradel discovered a bug in the appendix that he corrected. He also substantially simplified the code. I found a minor problem in Roger's code that I corrected. I know, version control would be great…

We first define a few quantities. See previous section for details.

```
nb_bootstrap = 10
nb_years = 6

target = data.frame(mw.dat,stringsAsFactors=F)
pseudo = target # initialization

popTot = popT = popR = matrix(NA, nb_bootstrap, nb_years-1) # abundance
tau = rep(NA, nb_bootstrap) # transient rate
det.p = rep(NA, nb_bootstrap) # recapture

set.seed(5)

# model structure
mw.proc <- process.data(mw.dat, model = "CJS")
mw.ddl <- make.design.data(mw.proc)
mw.ddl <- add.design.data(mw.proc, mw.ddl,"Phi", type = "age", bins = c(0,1,6), name = "ageclass", righ

# parameters
phiage <- list(formula=~ageclass)
p.ct <- list(formula=~1)
```

Let's run the bootstrap now:

```
for (k in 1:nb_bootstrap){
  # draw new sample
  pseudo$ch = sample(target$ch, replace=T)
  # calculate R and m
  firstobs = regexpr("1", pseudo$ch)
  R = summary(factor(firstobs,levels=1:nb_years))
  allobs = gregexpr("1", pseudo$ch)
  n = summary(as.factor(unlist(allobs)))
  m = n-R
  # fit model with 2 age classes on survival and constant recapture with MARK
  phiage.pct = mark(process.data(pseudo),mw.ddl,model.parameters=list(Phi=phiage,p=p.ct),output = FALSE
  tau[k] = 1 - phiage.pct$results$real[1,1] / phiage.pct$results$real[2,1] # transient rate
  det.p[k] = phiage.pct$results$real[3,1]
  # calculate abundance of residents and transients
  popR[k,] = (m[-1] + R[-1] * (1 - tau[k])) / det.p[k]
  popT[k,] = R[-1] * tau[k] / det.p[k]
}
```

Now we can calculate the abundance of residents and a confidence interval:

```
popRmean = apply(popR,2,mean) # mean resident population size
popRmean
```

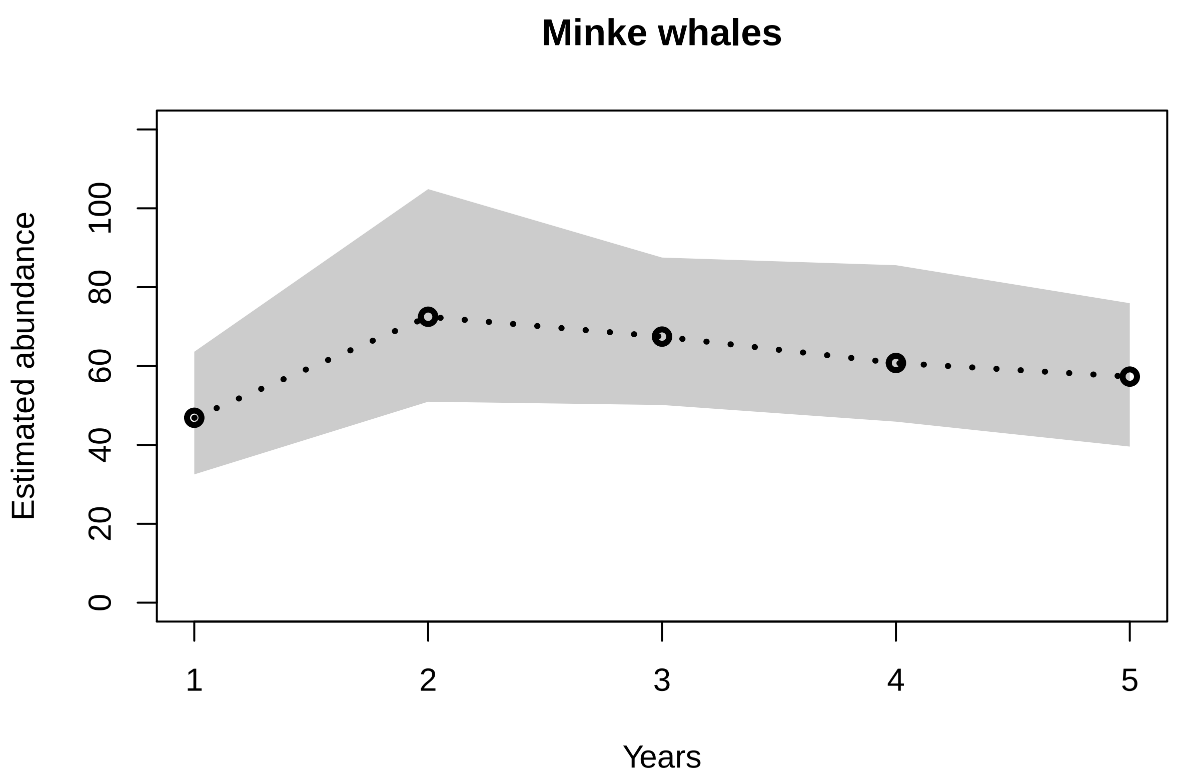
```
## [1] 46.88135 72.49815 67.46543 60.77500 57.31774
```

```
popRci = apply(popR,2,quantile,c(0.025,0.975))
popRci
```

```
##            [,1]      [,2]     [,3]     [,4]     [,5]
## 2.5%  32.52640  50.95279 50.12659 45.89512 39.58013
## 97.5% 63.62992 104.86695 87.50821 85.56883 75.93380
```

A plot:

```
plot(1:(nb_years-1),popRmean, col="black", type="n", pch=21, xlab="Years", lty=3, ylab="Estimated abunda
polygon(c(rev(1:(nb_years-1)), 1:(nb_years-1)), c(rev(popRci[2,]), popRci[1,]), col = 'grey80', border =
lines(1:(nb_years-1), popRmean, col="black",lty=3,type='o',lwd=3,pch=21)
```



Lastly, it is also possible to calculate an estimate of the transient rate along with its confidence interval using the bootstrap. Alternatively, the delta-method could be used.

```
mean(tau)
```

```
## [1] 0.4827876
```

```
quantile(tau,probs=c(2.5,97.5)/100)
```

```
##      2.5%     97.5%
## 0.3758784 0.6283435
```

Dealing with heterogeneity

As we said before, heterogeneity in the detection process may cause bias in abundance estimates (e.g., Cubaynes et al. 2010). I will consider here two ways of dealing with this issue: individual random-effect models and finite-mixture models.

Individual random effect model

Let's use the Humpback whale dataset of the first section.

```
library(RMark)
hw.dat = import.chdata("humpbackwhaleMaySep20062013.txt", header = F, field.names = c("ch"), field.types
summary(hw.dat)
```

```
##        ch
##  Length:195
##  Class :character
##  Mode  :character
```

```
attach(hw.dat)
```

We use a Cormack-Jolly-Seber model with a random effect in the detection process (Gimenez and Choquet 2010). The model structure is specified with the `model="CJSRandom"` option.

```
hw.proc <- process.data(hw.dat, model="CJSRandom")
hw.ddl <- make.design.data(hw.proc)
```

Then we specify the effects on survival and detection probabilities. By default, because we use the random structure in MARK, there is a random effect on both parameters, ie these probabilities are drawn from a normal distribution with a mean and a standard deviation. We fix the standard deviation of the random effect on survival to 0 to fit a model with a constant survival. In contrast, we let MARK estimate both parameters of the random effect for the recapture probability.

```
# mean survival
phi.ct = list(formula=~1) # constant
# standard deviation of the random effect on survival is fixed to 0
# in other words, no random effect on survival
sigmaphi.fixed=list(formula=~1,fixed=0)

# mean recapture probability
p.dot=list(formula=~1)
# standard deviation of the random effect on recapture
sigmap.dot=list(formula=~1)
```

Let's roll and fit this model.

```
model.re = mark(hw.proc,hw.ddl,model.parameters=list(Phi=phi.ct,p=p.dot,sigmap=sigmap.dot,sigmaphi=sigma
```

Let's have a look to the parameter estimates:

```
mle_p = get.real(model.re,"p", se= TRUE)[1,c('estimate','se','lcl','ucl')]
sigma_p = get.real(model.re,"sigmap", se= TRUE)[c('estimate','se','lcl','ucl')]
phi = get.real(model.re,"Phi", se= TRUE)[1,c('estimate','se','lcl','ucl')]

mle_p
```

```
##                  estimate       se       lcl       ucl
## p g1 c1 a1 t2 0.3752891 0.123645 0.1760596 0.6281039
```

```
sigma_p
```

```
##                    estimate        se       lcl     ucl
## sigmap g1 a0 t1  1.86494 0.7965141 0.8360443 4.16007
```

```
phi
```

```
##                    estimate         se       lcl       ucl
## Phi g1 c1 a0 t1 0.6255005 0.0665329 0.4890695 0.7445307
```

To test whether the random effect is significant, in other words to test the null hypothesis that the standard deviation of the random effect is null, we need to carry out a likelihood ratio test (LRT). The asymptotic behavior of the LRT statistic is a bit weird in that particular situation (see Gimenez and Choquet 2010 for details).

We first need the deviance of the two models with and without the random effect. To get the deviance of the model without random effect, we can use the results from the first section above, or run a model with the random structure by fixing the standard deviation of the random effect on recapture probability to 0. For the sake of complexity, let's use the latter option:

```
phi.ct = list(formula=~1) # constant
sigmaphi.fixed=list(formula=~1,fixed=0)
p.dot=list(formula=~1)
sigmap.fixed=list(formula=~1,fixed=0)
model.without.re = mark(hw.proc,hw.ddl,model.parameters=list(Phi=phi.ct,p=p.dot,sigmap=sigmap.fixed,sig
```

Then we can form the LRT statistic:

```
dev_model_with_RE = model.re$results$deviance
dev_model_without_RE = model.without.re$results$deviance
LRT = dev_model_without_RE - dev_model_with_RE
```

And calculate the p-value of the test:

```
1-pchisq(LRT,1)
```

```
## [1] 0.007683643
```

The test is highly significant, we reject the null hypothesis that the standard deviation is 0, therefore there seems to be heterogeneity as detection by the random effect.

From there, one can use the bootstrap as in the first section to estimate abundance along with its confidence interval using the recapture probability estimate `mle_p`. This value was calculated for us by MARK as the inverse [reciprocal function] logit of the mean recapture probability. Have a look to the results:

```
model.re$results$beta
```

```
##                       estimate        se       lcl       ucl
## Phi:(Intercept)     0.5129615 0.2840257 -0.043729 1.0696519
## sigmap:(Intercept)  0.6232290 0.4270990 -0.213885 1.4603431
## p:(Intercept)      -0.5095923 0.5273894 -1.543276 0.5240909
```

The mean value of the random effect on the recapture is `p:(Intercept)`. If you apply the standard $1/(1 + exp(-x))$ transformation, you obtain `mle_p`.

```
mean_p = model.re$results$beta[3,1] # extract the mean value of the random effect
1/(1+exp(-mean_p)) # calculate by hand
```

```
## [1] 0.3752891
```

```
mle_p[1] # produce by MARK
```

```
##                 estimate
## p g1 c1 a1 t2 0.3752891
```

Finite mixture models

Here, we estimate abundance while accounting for heterogeneity in the detection process using a model with finite mixture (see first section above). To do so, we follow Cubaynes et al. (2010) who extended the appraoch developed by Shirley Pledger and colleagues to account for heterogeneity to estimate abundance.

For illustration, let's first fit a model with heterogeneity in the recapture probability, with time-dependent survival and constant recapture probabilities. As usual, we first load the `RMark` package and read in the data.

```
# load RMark package
library(RMark)
# read in data
hw.dat = import.chdata("humpbackwhaleMaySep20062013.txt", header = F, field.names = c("ch"), field.type
summary(hw.dat)
```

```
##       ch
##  Length:195
##  Class :character
##  Mode  :character
```

```
attach(hw.dat)
```

Then we define the model structure, by using the `model="CJSMixture"` option.

```
hw.proc = process.data(hw.dat, model="CJSMixture")
hw.ddl = make.design.data(hw.proc)
```

We also define the effect on the parameters. Time-dependent survival, two-finite mixture on the recapture probability and a constant proportion of individual in each class.

```
# survival
phi.time = list(formula=~time) # constant
# recapture
p.mix = list(formula=~mixture) # mixture
# mixture proportion
pi.dot=list(formula=~1) # constant
```

Let's fit that model:

```r
model.het = mark(hw.proc,hw.ddl,model.parameters=list(Phi=phi.time,p=p.mix,pi=pi.dot),output = FALSE,de
```

Now how to decide whether heterogeneity is important? The cool thing is that it's fine to use the AIC to compare models with/without heterogeneity (Cubaynes et al. 2012). So let's fit the same model with homogeneous recapture probability and compare the AIC values:

```r
hw.proc = process.data(hw.dat, model="CJS")
hw.ddl = make.design.data(hw.proc)
phi.time = list(formula=~time) # year effect
p.ct = list(formula=~1) # constant
model.hom = mark(hw.proc,hw.ddl,model.parameters=list(Phi=phi.time,p=p.ct),output = FALSE,delete=T)
```

Compare the AIC values:

```r
summary(model.het)$AICc
```

```
## [1] 310.6533
```

```r
summary(model.hom)$AICc
```

```
## [1] 313.1428
```

Sounds like there is some heterogeneity. Let's have a look to the parameter estimates of the model with heterogeneity:

```r
model.het$results$real
```

```
##                      estimate        se          lcl       ucl fixed
## pi g1 a0 t1 m1      0.2289912 0.1284810 6.658710e-02 0.5528775
## Phi g1 c1 a0 t1 m1 0.6897862 0.2603167 1.700464e-01 0.9602101
## Phi g1 c1 a1 t2 m1 0.2760778 0.1202969 1.049174e-01 0.5537270
## Phi g1 c1 a2 t3 m1 0.9570430 0.2848274 2.821891e-05 0.9999999
## Phi g1 c1 a3 t4 m1 0.4446168 0.1441773 2.031315e-01 0.7154382
## Phi g1 c1 a4 t5 m1 0.5709730 0.1802927 2.392605e-01 0.8492054
## Phi g1 c1 a5 t6 m1 0.7454627 0.1992795 2.721192e-01 0.9582341
## Phi g1 c1 a6 t7 m1 0.9182507 0.2078496 4.705730e-02 0.9996088
## p g1 c1 a1 t2 m1   0.8878806 0.2474591 5.716670e-02 0.9990341
## p g1 c1 a1 t2 m2   0.2159634 0.0850128 9.334900e-02 0.4242670
##                       note
## pi g1 a0 t1 m1
## Phi g1 c1 a0 t1 m1
## Phi g1 c1 a1 t2 m1
## Phi g1 c1 a2 t3 m1
## Phi g1 c1 a3 t4 m1
## Phi g1 c1 a4 t5 m1
## Phi g1 c1 a5 t6 m1
## Phi g1 c1 a6 t7 m1
## p g1 c1 a1 t2 m1
## p g1 c1 a1 t2 m2
```

The proportion of individuals in mixture 1 is:

```
prop = model.het$results$real[1,1]
prop
```

```
## [1] 0.2289912
```

with detection probability:

```
p1 = model.het$results$real[9,1]
p1
```

```
## [1] 0.8878806
```

For the other mixture, the proportion is the complementary and the detection probability is:

```
p2 = model.het$results$real[10,1]
p2
```

```
## [1] 0.2159634
```

By the way, survival is:

```
phi = model.het$results$real[2:8,1]
phi
```

```
## [1] 0.6897862 0.2760778 0.9570430 0.4446168 0.5709730 0.7454627 0.9182507
```

Now let's the bootstrap to get confidence intervals (and median) for abundance. IN PROGRESS (means I'm debugging...).

To do

- multi-model inference using bootstrap à la Buckland
- add heterogeneity à la Pledger
- add complete reference for Chiara's paper once published
- add Jolly-Seber as in Karamanlidis et al. (2015)
- add robust-design as in Nina and Blaise papers.

References

- Buckland ST, Burnham KP, Augustin NH (1997). Model selection: An integral part of inference. Biometrics. 53:603–618.

- Cubaynes, S., C. Lavergne, E. Marboutin, and O. Gimenez (2012). Assessing individual heterogeneity using model selection criteria: How many mixture components in capture-recapture models? Methods in Ecology and Evolution 3: 564-573.

- Cubaynes, S. Pradel, R. Choquet, R. Duchamp, C. Gaillard, J.-M., Lebreton, J.-D., Marboutin, E., Miquel, C., Reboulet, A.-M., Poillot, C., Taberlet, P. and O. Gimenez. (2010). Importance of accounting for detection heterogeneity when estimating abundance: the case of French wolves. Conservation Biology 24:621-626.

- Gimenez, O. and R. Choquet (2010). Incorporating individual heterogeneity in studies on marked animals using numerical integration: capture-recapture mixed models. Ecology 91: 951-957.
- Karamanlidis, A.A., M. de Gabriel Hernando, L. Krambokoukis, O. Gimenez (2015). Evidence of a large carnivore population recovery: counting bears in Greece. Journal for Nature Conservation. 27: 10–17
- Madon, B., C. Garrigue, R. Pradel, O. Gimenez (2013). Transience in the humpback whale population of New Caledonia and implications for abundance estimation. Marine Mammal Science 29: 669-678.