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](http://www.phidot.org/software/mark/docs/book/pdf/app_3.pdf) 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 finite mixtures and an individual random effect. The bootstrap is used to obtain confidence intervals. I also illustrate multi-model inference and use the bootstrap to perform model selection (Buckland et al. 1997).

## 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)

Each row is an individual that has been detected (coded as a 1) or non-detected (coded as a 0) over the years in columns. Have a look to the file dataset1.txt in your favorite text editor. Other formats are fine.

hw.dat = import.chdata("dataset1.txt", header = F, field.names = c("ch"), field.types = NULL)  
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.

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.

# 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!

# 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.

summary(Model.1)$AICc

## [1] 317.4295

summary(Model.2)$AICc

## [1] 311.3483

summary(Model.3)$AICc

## [1] 313.1428

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 0.5234884 0.0565077 0.4133879 0.6313524

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 0.6814075 0.2413577 0.1948435 0.9497570  
## p g1 c1 a2 t3 0.1515677 0.1041699 0.0352270 0.4663918  
## p g1 c1 a3 t4 0.4246960 0.1541836 0.1764801 0.7177504  
## p g1 c1 a4 t5 0.2866319 0.1204953 0.1123644 0.5605063  
## p g1 c1 a5 t6 0.3746890 0.1538311 0.1419705 0.6845395  
## p g1 c1 a6 t7 0.4332827 0.1215465 0.2246677 0.6685709  
## p g1 c1 a7 t8 0.8383462 0.1685125 0.3119204 0.9834243

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 58.86564 45.35434 93.41080 122.32199 90.65467

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 = 50  
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(process.data(pseudo),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)]  
}

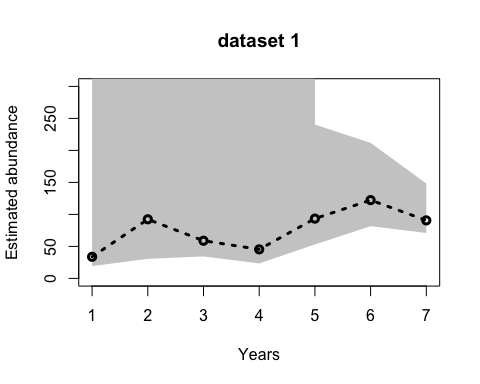
Now we can get confidence intervals:

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]  
## 2.5% 19.22829 3.056743e+01 34.36395 2.340005e+01 53.05576 81.58606  
## 97.5% 113.86445 3.484317e+11 219.90659 1.749008e+09 240.40987 211.72946  
## [,7]  
## 2.5% 71.0000  
## 97.5% 148.2417

A plot:

plot(1:(nb\_years-1),estim\_abundance, col="black", type="n", pch=21, xlab="Years", lty=3, ylab="Estimated abundance", main="dataset 1",lwd=3,ylim=c(0,300))  
polygon(c(rev(1:(nb\_years-1)), 1:(nb\_years-1)), c(rev(ci\_hw[2,]), ci\_hw[1,]), col = 'grey80', border = NA)  
lines(1:(nb\_years-1), estim\_abundance, col="black",lty=3,type='o',lwd=3,pch=21)



## What if transience occurs?

We now analyse another dataset dataset2.txt.

library(RMark)  
mw.dat = import.chdata("dataset2.txt", header = F, field.names = c("ch"), field.types = NULL)  
summary(mw.dat)

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

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.

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 = FALSE)

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.

# 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!

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

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

summary(Model.1)$AICc

## [1] 480.6304

summary(Model.2)$AICc

## [1] 486.8038

summary(Model.3)$AICc

## [1] 485.1968

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.

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)](https://dl.dropboxusercontent.com/u/23160641/my-pubs/Madonetal2012MMS.pdf). 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 have been 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", right = FALSE)  
  
# 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,delete=T)  
 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

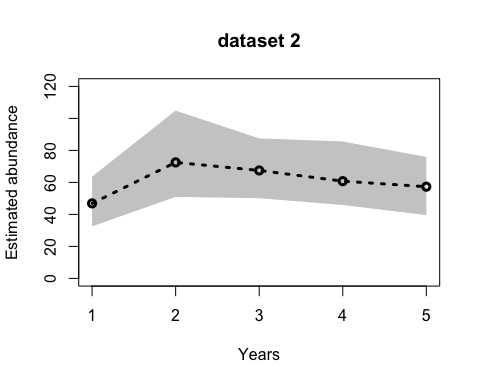
## [1] 46.88134 72.49815 67.46543 60.77500 57.31773

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 104.86693 87.50819 85.56881 75.93378

A plot:

plot(1:(nb\_years-1),popRmean, col="black", type="n", pch=21, xlab="Years", lty=3, ylab="Estimated abundance", main="dataset 2",lwd=3,ylim=c(0,120))  
polygon(c(rev(1:(nb\_years-1)), 1:(nb\_years-1)), c(rev(popRci[2,]), popRci[1,]), col = 'grey80', border = NA)  
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](https://github.com/oliviergimenez/delta_method).

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)](https://dl.dropboxusercontent.com/u/23160641/my-pubs/Cubaynesetal2010.pdf). 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 dataset1.txt of the first section.

library(RMark)  
hw.dat = import.chdata("dataset1.txt", header = F, field.names = c("ch"), field.types = NULL)  
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)](https://dl.dropboxusercontent.com/u/23160641/my-pubs/Gimenez%26Choquet2010Ecology.pdf). 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=sigmaphi.fixed),output = FALSE,delete=T)

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 0.1236449 0.1760596 0.6281037

sigma\_p

## estimate se lcl ucl  
## sigmap g1 a0 t1 1.864941 0.796514 0.8360445 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](https://dl.dropboxusercontent.com/u/23160641/my-pubs/Gimenez%26Choquet2010Ecology.pdf) 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 could 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,sigmaphi=sigmaphi.fixed),output = FALSE,delete=T)

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 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) 0.5129616 0.2840257 -0.0437288 1.0696519  
## sigmap:(Intercept) 0.6232292 0.4270989 -0.2138847 1.4603430  
## p:(Intercept) -0.5095925 0.5273892 -1.5432753 0.5240904

The mean value of the random effect on the recapture is p:(Intercept). If you apply the standard 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] # produced 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)](https://dl.dropboxusercontent.com/u/23160641/my-pubs/Cubaynesetal2010.pdf) who extended the appraoch developed by Shirley Pledger and colleagues to account for heterogeneity to estimate abundance. Please, have a look to this paper and its appendix for details about the methods and formulas.

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. Let's use dataset2.txt of the second section.

# load RMark package  
library(RMark)  
# read in data  
mw.dat = import.chdata("dataset2.txt", header = F, field.names = c("ch"), field.types = NULL)  
summary(mw.dat)

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

attach(mw.dat)

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

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

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

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

Let's fit that model:

model.het = mark(mw.proc,mw.ddl,model.parameters=list(Phi=phi.ct,p=p.mix,pi=pi.dot),output = FALSE,delete=T)

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)](https://dl.dropboxusercontent.com/u/23160641/my-pubs/Cubaynesetal2011MEE.pdf). So let's fit the same model with homogeneous recapture probability and compare the AIC values:

mw.proc2 = process.data(mw.dat, model="CJS")  
mw.ddl2 = make.design.data(mw.proc2)  
model.hom = mark(mw.proc2,mw.ddl2,model.parameters=list(Phi=phi.ct,p=p.ct),output = FALSE,delete=T)

Compare the AIC values:

summary(model.het)$AICc

## [1] 480.5107

summary(model.hom)$AICc

## [1] 494.9341

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

model.het$results$real

## estimate se lcl ucl fixed note  
## pi g1 a0 t1 m1 0.4161090 0.1068705 0.2313149 0.6279348   
## Phi g1 c1 a0 t1 m1 0.8031116 0.0539366 0.6764034 0.8883921   
## p g1 c1 a1 t2 m1 0.6119298 0.0762766 0.4565781 0.7474370   
## p g1 c1 a1 t2 m2 0.0511852 0.0398347 0.0106930 0.2121336

The proportion of individuals in mixture 1 is:

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

## [1] 0.416109

with detection probability:

p1 = model.het$results$real[3,1]  
p1

## [1] 0.6119298

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

p2 = model.het$results$real[4,1]  
p2

## [1] 0.0511852

Lastly, survival is:

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

## [1] 0.8031116

Now let's use the bootstrap to get confidence intervals (and median) for abundance.

We will need a function to spot the first detections in the encounter histories:

firstdetection = function(x){  
 b = sort(x,index.return=T,decreasing=T)  
 res = b$ix[1]  
 return(res)}

We will also need to add spaces between the columns of the dataset:

mw.dat.spaces = matrix(as.numeric(unlist(strsplit(mw.dat$ch, ''))),nrow=nrow(mw.dat),byrow=T)

Let's run the bootstrap:

Nhet = NULL # initialization  
nb\_bootstrap = 10 # nb of bootstrap iterations (should be 500 or 1000!)  
  
for (i in 1:nb\_bootstrap){  
  
 # resample in original dataset  
 mask = sample.int(nrow(mw.dat.spaces),replace=T)  
 pseudodata = mw.dat.spaces[mask,]  
  
 # nb of ind detected per occasion  
 cijdata = apply(pseudodata,2,sum)  
  
 # first detection  
 d = apply(pseudodata,1,firstdetection)  
  
 # u (newly marked)  
 udata = NULL  
 for(kk in 1:ncol(pseudodata)) {udata[kk] = sum(d == kk)}  
  
 # m (already marked)  
 mdata = cijdata - udata  
  
 # delete first occasion  
 cij = cijdata[-1]  
 m = mdata[-1]  
 u = udata[-1]  
   
 # expected newly marked (Cubaynes et al. 2010)  
 bigU <- matrix(0,nrow=length(p1),ncol=length(u)) # here length(p1) = 1, but if time-dependent, length(p1) > 1  
 for(zz in 1:ncol(bigU)) {  
 bigU[,zz] = (1-prop) \* u[zz] / p2 + prop \* u[zz] / p1  
 }  
 # expected already marked (Cubaynes et al. 2010)  
 # M2 = u1 (pi phi1 + (1-pi) phi1)  
 # M3 = u1 (pi phi1 phi2 + (1-pi) phi1 phi2) + u2 (pi phi2 + (1-pi) phi2)  
 # M4 = u1 (pi phi1 phi2 phi3 + (1-pi) phi1 phi2 phi3) + u2 (pi phi2 phi3 + (1-pi) phi2 phi3) + u3 (pi phi3 + (1-pi) phi3)  
 # ...  
 surv <- matrix(rep(phi,length(u)),ncol=length(u),byrow=T) # to be modified if phi is time-dependent  
 bigM <- matrix(0,nrow=nrow(surv),ncol=ncol(surv))  
 for(ii in 1:nrow(bigM)) {  
 for(t in 1:ncol(bigM)) {  
 temp <- rep(NA,t)  
 for(j in 1:t) {  
 temp[j] <- u[j] \* ((1-prop) \* prod(surv[ii,1:j]) + prop \* prod(surv[ii,1:j]))  
 }  
 bigM[ii,t] = sum(temp)  
 }  
 }  
 # compute abundance estimate for current bootstrap sample  
 Nhet <- rbind(Nhet,bigU + bigM)  
}

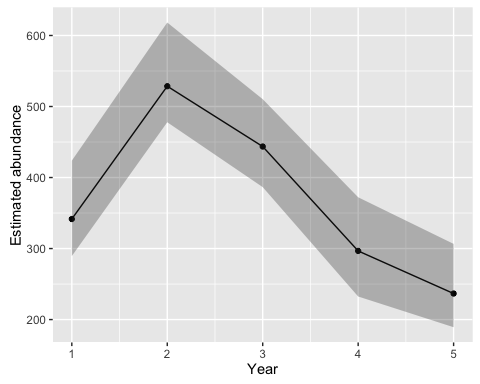
Get median and confidence interval:

res <- apply(Nhet,2,quantile,c(2.5,50,97.5)/100)  
res

## [,1] [,2] [,3] [,4] [,5]  
## 2.5% 289.3923 477.8218 386.4733 232.4835 189.2225  
## 50% 341.5989 528.5707 443.6484 296.6347 236.6584  
## 97.5% 423.7760 618.2259 510.3297 372.2116 306.5724

Let's do a nice plot:

library(ggplot2)  
mp = data.frame(year = 1:ncol(Nhet), Nhat = res[2,])  
N = nrow(mp)  
predframe <- with(mp,data.frame(year,Nhat,lwr=res[1,],upr=res[3,]))  
(p1 <- ggplot(mp, aes(year, Nhat))+  
 geom\_point()+  
 geom\_line(data=predframe)+  
 geom\_ribbon(data=predframe,aes(ymin=lwr,ymax=upr),alpha=0.3)  
 + ylab("Estimated abundance") + xlab("Year"))



## Model-averaging abundance estimates

If you remember the first section, the 4 models we fitted to dataset1 had close AICc values. In this situation, multi-model inference is recommended. Following Buckland et al. (1997), we use the bootstrap to obtain model-averaged abundance estimates.

First, load RMark and read in that data:

rm(list=ls(all=TRUE))  
library(RMark)  
hw.dat = import.chdata("dataset1.txt", header = F, field.names = c("ch"), field.types = NULL)

Define the effects on parameters:

# define parameter structure  
Phi.ct = list(formula=~1) # constant survival  
Phi.time = list(formula=~time) # year effect on survival  
p.ct = list(formula=~1) # constant detection  
p.time = list(formula=~time) # year effect on detection

Let's define a few quantities we will need later on:

nb\_bootstrap = 10 # use 500 or 1000 instead  
nb\_years = 8  
target = data.frame(hw.dat,stringsAsFactors=F)  
set.seed(5)  
pseudo = target # initialization  
# storage quantities  
outlist <- vector("list", nb\_bootstrap)  
outdata <- vector("list", nb\_bootstrap)  
outreal <- vector("list", nb\_bootstrap)  
outnhat <- vector("list", nb\_bootstrap)

Let's run the boostrap:

for (k in 1:nb\_bootstrap){  
 # resample in the original dataset with replacement  
 pseudo$ch = sample(target$ch, replace=T)  
 # define model structure  
 hw.proc = process.data(pseudo, model="CJS")  
 hw.ddl = make.design.data(hw.proc)  
 # fit all 4 models with Mark  
 # 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)  
 # gather results  
 all.models <- collect.models()  
  
 # store bootstrap sample  
 outdata[[k]] <- pseudo  
 outlist[[k]] <- all.models$model.table  
   
 # get best model  
 ind\_best\_mod = row.names(all.models$model.table)[1]  
 name\_best\_mod = paste('Model.',ind\_best\_mod,sep='')  
   
 # get recapture estimate from best model  
 # if constant, duplicate values  
 # if time-dep, take the whole vector  
 mle.p = get.real(get(name\_best\_mod),"p", se= TRUE)$estimate[1:(nb\_years-1)]  
 outreal[[k]] = mle.p  
   
 # get abundance estimate  
 allobs = gregexpr("1", pseudo$ch)  
 n = summary(as.factor(unlist(allobs)))  
 nhat = n[-1]/mle.p  
 outnhat[[k]] = nhat  
 }

Convert the list of abundance estimate in a matrix, and calculate quantiles over the bootstrap iterations:

res = matrix(unlist(outnhat),nrow=length(outnhat),byrow=T)  
apply(res,2,mean) # mean estimate

## [1] 4.944162e+01 5.465948e+10 7.457325e+01 3.399856e+01 1.195985e+02  
## [6] 1.263711e+02 1.451078e+02

apply(res,2,mean) # median estimate

## [1] 4.944162e+01 5.465948e+10 7.457325e+01 3.399856e+01 1.195985e+02  
## [6] 1.263711e+02 1.451078e+02

apply(res,2,quantile,probs=c(2.5,97.5)/100) # confidence interval

## [,1] [,2] [,3] [,4] [,5] [,6]  
## 2.5% 34.98817 2.418416e+01 42.38654 24.41150 66.82858 86.89765  
## 97.5% 78.55096 4.020420e+11 201.57190 46.91942 240.40987 197.85791  
## [,7]  
## 2.5% 77.05311  
## 97.5% 239.96159

This is not exactly what Buckland et al. (1997) advocates, see last paragraph of section 3 in this paper, but this will do for now.

## To do

* check the calculations for bigU and bigM and make them generic (what if survival is time-dependen?)
* add Jolly-Seber as in [Karamanlidis et al. (2015)](https://dl.dropboxusercontent.com/u/23160641/my-pubs/Karamanlidisetal2015-Arcturos.pdf)
* add robust-design as in papers currently in reviews (including model selection with bootstrap).

## 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.