Appendix

## Introduction

In this appendix, we introduce three methods to cope with individual heterogeneity in capture-recapture models. First, we present multistate models in which heterogeneity is measured on individuals using states. Then, we illustrate models with individual random effects and finite mixtures that can help in dealing with hidden heterogeneity. We refer to the paper for a formal presentation of these models and a list of references using them. Throughout this appendix, we use R to simulate data and program Mark is called from R using package RMark to fit models. We do our best to ensure reproducibility. Note that program E-SURGE could be used instead, or the Bayesian approach using Jags.

## Multistate models

In this section, we aim at illustrating how not accounting for individual heterogeneity may obscure the detection of life-history tradeoffs. In details, we consider two states for the individuals of our fake population, non-breeding (NB) and breeding (B). To mimic individual heterogeneity, we simulate a bunch of good individuals with survival and and a bunch of bad individuals with survival and . Overall, the cost of breeding on survival should be detected only in bad individuals after accounting for individual heterogeneity through quality. For each group of bad vs. good individuals, we consider the same detection probability , the same transition probabilities between breeding states and , and 100 newly marked individuals for each group in each year of the 6-year study.

### Data simulation

Using R code from [Kéry and Schaub (2012)](http://www.vogelwarte.ch/de/projekte/publikationen/bpa/) book (chapter 9), we first define a function to simulate multistate capture-recapture data:

# Define function to simulate multistate capture-recapture data  
simul.ms <- function(PSI.STATE, PSI.OBS, marked, unobservable = NA){  
 # Unobservable: number of state that is unobservable  
 n.occasions <- dim(PSI.STATE)[4] + 1  
 CH <- CH.TRUE <- matrix(NA, ncol = n.occasions, nrow = sum(marked))  
 # Define a vector with the occasion of marking  
 mark.occ <- matrix(0, ncol = dim(PSI.STATE)[1], nrow = sum(marked))  
 g <- colSums(marked)  
 for (s in 1:dim(PSI.STATE)[1]){  
 if (g[s]==0) next # To avoid error message if nothing to replace  
 mark.occ[(cumsum(g[1:s])-g[s]+1)[s]:cumsum(g[1:s])[s],s] <-  
 rep(1:n.occasions, marked[1:n.occasions,s])  
 } #s  
 for (i in 1:sum(marked)){  
 for (s in 1:dim(PSI.STATE)[1]){  
 if (mark.occ[i,s]==0) next  
 first <- mark.occ[i,s]  
 CH[i,first] <- s  
 CH.TRUE[i,first] <- s  
 } #s  
 for (t in (first+1):n.occasions){  
 # Multinomial trials for state transitions  
 if (first==n.occasions) next  
 state <- which(rmultinom(1, 1, PSI.STATE[CH.TRUE[i,t-1],,i,t-1])==1)  
 CH.TRUE[i,t] <- state  
 # Multinomial trials for observation process  
 event <- which(rmultinom(1, 1, PSI.OBS[CH.TRUE[i,t],,i,t-1])==1)  
 CH[i,t] <- event  
 } #t  
 } #i  
 # Replace the NA and the highest state number (dead) in the file by 0  
 CH[is.na(CH)] <- 0  
 CH[CH==dim(PSI.STATE)[1]] <- 0  
 CH[CH==unobservable] <- 0  
 id <- numeric(0)  
 for (i in 1:dim(CH)[1]){  
 z <- min(which(CH[i,]!=0))  
 ifelse(z==dim(CH)[2], id <- c(id,i), id <- c(id))  
 }  
 return(list(CH=CH[-id,], CH.TRUE=CH.TRUE[-id,]))  
# CH: capture histories to be used  
# CH.TRUE: capture histories with perfect observation  
}

Second, we use this function to simulate the two datasets of good and bad individuals:

set.seed(1) # for reproducibility  
p = 0.9  
R = 100  
#------------------------------  
#---- good quality individuals  
#------------------------------  
# Define mean survival, transitions, recapture, as well as number of occasions, states, observations and released individuals  
phiA <- 0.7  
phiB <- 0.8  
psiAB <- 0.8  
psiBA <- 0.3  
pA <- p  
pB <- p  
n.occasions <- 6  
n.states <- 3  
n.obs <- 3  
marked <- matrix(NA, ncol = n.states, nrow = n.occasions)  
marked[,1] <- rep(R, n.occasions)  
marked[,2] <- rep(R, n.occasions)  
marked[,3] <- rep(0, n.occasions)  
# Define matrices with survival, transition and recapture probabilities  
# 1. State process matrix  
totrel <- sum(marked)\*(n.occasions-1)  
PSI.STATE <- array(NA, dim=c(n.states, n.states, totrel, n.occasions-1))  
for (i in 1:totrel){  
 for (t in 1:(n.occasions-1)){  
 PSI.STATE[,,i,t] <- matrix(c(  
 phiA\*(1-psiAB), phiA\*psiAB, 1-phiA,  
 phiB\*psiBA, phiB\*(1-psiBA), 1-phiB,  
 0, 0, 1 ), nrow = n.states, byrow = TRUE)  
 } #t  
} #i  
# 2.Observation process matrix  
PSI.OBS <- array(NA, dim=c(n.states, n.obs, totrel, n.occasions-1))  
for (i in 1:totrel){  
 for (t in 1:(n.occasions-1)){  
 PSI.OBS[,,i,t] <- matrix(c(  
 pA, 0, 1-pA,  
 0, pB, 1-pB,  
 0, 0, 1 ), nrow = n.states, byrow = TRUE)  
 } #t  
} #i  
  
# Execute function  
sim <- simul.ms(PSI.STATE, PSI.OBS, marked)  
CH <- sim$CH  
his1 = CH[!apply(CH,1,sum)==0,] # remove lines of 0s  
  
#------------------------------  
#---- bad quality individuals  
#------------------------------  
# Define mean survival, transitions, recapture, as well as number of occasions, states, observations and released individuals  
phiA <- 0.7  
phiB <- 0.6  
psiAB <- 0.8  
psiBA <- 0.3  
pA <- p  
pB <- p  
n.occasions <- 6  
n.states <- 3  
n.obs <- 3  
marked <- matrix(NA, ncol = n.states, nrow = n.occasions)  
marked[,1] <- rep(R, n.occasions)  
marked[,2] <- rep(R, n.occasions)  
marked[,3] <- rep(0, n.occasions)  
# Define matrices with survival, transition and recapture probabilities  
# 1. State process matrix  
totrel <- sum(marked)\*(n.occasions-1)  
PSI.STATE <- array(NA, dim=c(n.states, n.states, totrel, n.occasions-1))  
for (i in 1:totrel){  
 for (t in 1:(n.occasions-1)){  
 PSI.STATE[,,i,t] <- matrix(c(  
 phiA\*(1-psiAB), phiA\*psiAB, 1-phiA,  
 phiB\*psiBA, phiB\*(1-psiBA), 1-phiB,  
 0, 0, 1 ), nrow = n.states, byrow = TRUE)  
 } #t  
} #i  
# 2.Observation process matrix  
PSI.OBS <- array(NA, dim=c(n.states, n.obs, totrel, n.occasions-1))  
for (i in 1:totrel){  
 for (t in 1:(n.occasions-1)){  
 PSI.OBS[,,i,t] <- matrix(c(  
 pA, 0, 1-pA,  
 0, pB, 1-pB,  
 0, 0, 1 ), nrow = n.states, byrow = TRUE)  
 } #t  
} #i  
  
# Execute function  
sim <- simul.ms(PSI.STATE, PSI.OBS, marked)  
CH <- sim$CH  
his2 = CH[!apply(CH,1,sum)==0,] # remove lines of 0s

Last, we pool these two datasets together:

his = rbind(his1,his2)   
head(his) # display first lines

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 1 2 1 0 2 2  
## [2,] 1 0 0 0 0 0  
## [3,] 1 0 0 0 0 0  
## [4,] 1 1 0 0 0 0  
## [5,] 1 0 0 0 0 0  
## [6,] 1 2 2 2 2 0

tail(his) # display last lines

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1995,] 0 0 0 0 2 2  
## [1996,] 0 0 0 0 2 0  
## [1997,] 0 0 0 0 2 2  
## [1998,] 0 0 0 0 2 2  
## [1999,] 0 0 0 0 2 0  
## [2000,] 0 0 0 0 2 1

### Model fitting

First, we format the data we've just simulated so that these can be used with RMark (check out [these notes](https://sites.google.com/site/workshoponcmr/) by Mike Conroy for more details):

k = ncol(his) # nb of capture occasions  
n = nrow(his) # nb of individuals  
out = array(dim=n)  
for (i in 1:n){  
 y = (his[i,] > 0) \* his[i,]  
 out[i] = paste(y,collapse="")  
}  
capt.hist = data.frame(ch = out)

Then we fit a multistate model: we assume that survival depends on the breeding states, transition probabilities are constant over time, as well as the detection probability:

# load RMark package  
library(RMark)

## This is RMark 2.2.0

# Process data  
mstrata.processed=process.data(capt.hist,model="Multistrata")  
  
# Create default design data  
mstrata.ddl=make.design.data(mstrata.processed)  
  
# Define survival probability  
S.stratum=list(formula=~stratum) # survival depends on states  
  
# Define detection probability  
p.dot=list(formula=~1) # constant over time, does not depend on states  
  
# Define transition probs  
Psi.s=list(formula=~-1+stratum:tostratum)  
  
# Run model with state effect on survival  
mstrata.mod = mark(mstrata.processed,mstrata.ddl,model.parameters=list(S=S.stratum,p=p.dot,Psi=Psi.s),output = FALSE,delete=T)  
mstrata.mod$results$real[c(1:4,19),1:4]

## estimate se lcl ucl  
## S s1 g1 c1 a0 o1 t1 0.6920953 0.0128058 0.6664453 0.7166116  
## S s2 g1 c1 a0 o1 t1 0.6998620 0.0100857 0.6797302 0.7192511  
## p s1 g1 c1 a1 o1 t2 0.9004132 0.0079210 0.8837763 0.9148979  
## Psi s1 to2 g1 c1 a0 o1 t1 0.7776552 0.0135237 0.7500268 0.8030318  
## Psi s2 to1 g1 c1 a0 o1 t1 0.3083454 0.0118114 0.2856886 0.3319640

Run same model without state effect on survival:

S.dot=list(formula=~1) # survival does not depend on states  
m.mod = mark(mstrata.processed,mstrata.ddl,model.parameters=list(S=S.dot,p=p.dot,Psi=Psi.s),output = FALSE,delete=T)  
m.mod$results$real[c(1:3,18),1:4]

## estimate se lcl ucl  
## S s1 g1 c1 a0 o1 t1 0.6968449 0.0078024 0.6813379 0.7119163  
## p s1 g1 c1 a1 o1 t2 0.9003744 0.0079243 0.8837306 0.9148652  
## Psi s1 to2 g1 c1 a0 o1 t1 0.7776517 0.0135229 0.7500250 0.8030271  
## Psi s2 to1 g1 c1 a0 o1 t1 0.3083577 0.0118130 0.2856980 0.3319794

Compare AICc:

m.mod$results$AICc

## [1] 8838.698

mstrata.mod$results$AICc

## [1] 8840.482

Sounds like the difference in survival of breeding vs. non-breeding individuals is hard to detect.

Let's add individual heterogeneity through an individual covariate for bad vs. good individuals:

capt.hist$quality=c(rep('good',nrow(his1)),rep('bad',nrow(his2)))  
head(capt.hist)

## ch quality  
## 1 121022 good  
## 2 100000 good  
## 3 100000 good  
## 4 110000 good  
## 5 100000 good  
## 6 122220 good

tail(capt.hist)

## ch quality  
## 1995 000022 bad  
## 1996 000020 bad  
## 1997 000022 bad  
## 1998 000022 bad  
## 1999 000020 bad  
## 2000 000021 bad

Now we fit again the two models from above, including the effect of individual heterogeneity.

# Process data  
mstrata.processed=process.data(capt.hist,model="Multistrata",groups = 'quality')

## Warning in process.data(capt.hist, model = "Multistrata", groups = "quality"):   
## quality is not a factor variable. Coercing to factor.

# Create default design data  
mstrata.ddl=make.design.data(mstrata.processed)  
  
# define survival function of both states and quality  
S.covstrata=list(formula=~quality\*stratum)  
S.cov=list(formula=~quality)  
  
# Run model with state effect on survival  
mcovstrata.mod = mark(mstrata.processed,mstrata.ddl,model.parameters=list(S=S.covstrata,p=p.dot,Psi=Psi.s),output = FALSE,delete=T)  
mcovstrata.mod$results$real[c(1:6,21),1:4]

## estimate se lcl ucl  
## S s1 gbad c1 a0 o1 t1 0.6986444 0.0186714 0.6608382 0.7339331  
## S s2 gbad c1 a0 o1 t1 0.5782697 0.0157664 0.5471001 0.6088280  
## S s1 ggood c1 a0 o1 t1 0.6874006 0.0174323 0.6522602 0.7205116  
## S s2 ggood c1 a0 o1 t1 0.8002455 0.0121662 0.7753300 0.8230286  
## p s1 gbad c1 a1 o1 t2 0.9000171 0.0079117 0.8834062 0.9144906  
## Psi s1 to2 gbad c1 a0 o1 t1 0.7776836 0.0135261 0.7500500 0.8030643  
## Psi s2 to1 gbad c1 a0 o1 t1 0.3081264 0.0118024 0.2854873 0.3317273

Same model without state effect on survival:

mcov.mod = mark(mstrata.processed,mstrata.ddl,model.parameters=list(S=S.cov,p=p.dot,Psi=Psi.s),output = FALSE,delete=T)  
mcov.mod$results$real[c(1:4,19),1:4]

## estimate se lcl ucl  
## S s1 gbad c1 a0 o1 t1 0.6262137 0.0118487 0.6027153 0.6491313  
## S s1 ggood c1 a0 o1 t1 0.7571266 0.0098936 0.7372136 0.7759892  
## p s1 gbad c1 a1 o1 t2 0.8998853 0.0079319 0.8832309 0.9143946  
## Psi s1 to2 gbad c1 a0 o1 t1 0.7776518 0.0135229 0.7500250 0.8030271  
## Psi s2 to1 gbad c1 a0 o1 t1 0.3083578 0.0118130 0.2856980 0.3319794

Compare AICc:

mcovstrata.mod$results$AICc # quality and state on survival

## [1] 8720.129

mstrata.mod$results$AICc # state on survival

## [1] 8840.482

mcov.mod$results$AICc # quality on survival

## [1] 8766.522

m.mod$results$AICc # constant survival

## [1] 8838.698

Clearly, the inclusion of quality improves the AICc. Also, the model with a difference in survival between breeders and non-breeders is better supported by the data when individual heterogeneity is accounted for.

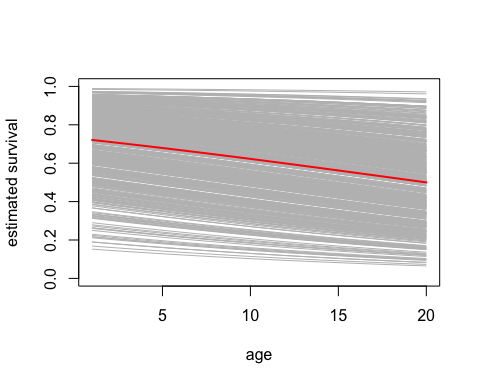
# Models with individual random effects

Here, we aim at illustrating how not accounting for individual heterogeneity may obscure the detection of senescence in survival. More specifically, we consider a single cohort of 500 individuals with survival decreasing as they age over a 20-year study. We also add a frailty for each individual under the form of a normal distribution. Specifically, we specify where . We use , and . If we condition upon the random effect, survival is decreasing as age increases. Note that we consider the same detection probability for all individuals.

### Data simulation

First, we simulate survival for each individual then plot the individual trajectories (in grey) as well as survival conditional on the random effect (in red):

r = set.seed(3) # for reproducibility  
p = 0.5 # detection  
intercept\_phi = 1   
slope\_phi = -0.05  
sigmaphi = 1  
nind = 500 # nb of individuals  
nyear = 20 # duration of the study  
expit<-function(x){exp(x)/(1+exp(x))} # reciprocal logit function  
z<-data<-x<-matrix(NA,nrow=nind,ncol=nyear)  
first<-rep(1,nind)  
age = matrix(NA,nind,nyear)  
phi = matrix(NA,nind,nyear)  
# simulate age-varying survival for each individual  
for (i in 1:nind){  
 mask <- first[i]:nyear  
 age[i,mask] <- mask - first[i] + 1  
 phi[i,mask] <- expit(intercept\_phi + slope\_phi \* age[i,mask] + rnorm(1,0,sigmaphi))  
}  
plot(age[1,],phi[1,],type='l',col='grey',ylim=c(0,1),xlab='age',ylab='estimated survival')  
for (i in 2:nind){  
 lines(age[i,],phi[i,],type='l',col='grey')  
}  
lines(1:nyear,expit(intercept\_phi + slope\_phi \* 1:nyear),col='red',lwd=2)



Now simulate the encounter histories:

for(i in 1:nind){  
 z[i,first[i]] <- x[i,first[i]] <- 1  
 for(j in (first[i]+1):nyear){  
 z[i,j]<-rbinom(1,1,phi[i,j-1]\*z[i,j-1])  
 x[i,j]<-rbinom(1,1,z[i,j]\*p)  
 }  
}  
his = x  
his[is.na(his)]=0 # remove lines with 0's

### Model fitting

First, we format the data we've just simulated so that these can be used with RMark:

k = ncol(his) # nb of capture occasions  
n = nrow(his) # nb of individuals  
out = array(dim=n)  
for (i in 1:n){  
 y = (his[i,] > 0) \* 1  
 out[i] = paste(y,collapse="")  
}  
capt.hist = data.frame(ch = out)

Now, we add age as a time-varying individual covariate to the dataset (heck out [these notes](https://sites.google.com/site/workshoponcmr/) by Mike Conroy for more details):

df = data.frame(time=c(1:(k-1)),cov=runif(k-1))   
simul.data = list(cap.data=capt.hist,cov=df)  
n.ind <- nrow(simul.data$cap.data)  
for (j in 1:k){  
 name = paste('cov',j,sep='')  
 assign(name,age[,j])  
}  
cap<-simul.data$cap.data  
# pretty ugly lines of codes to follow, happy to hear for suggestions to make this dynamic  
cap$cov1=cov1  
cap$cov2=cov2  
cap$cov3=cov3  
cap$cov4=cov4  
cap$cov5=cov5  
cap$cov6=cov6  
cap$cov7=cov7  
cap$cov8=cov8  
cap$cov9=cov9  
cap$cov10=cov10  
cap$cov11=cov11  
cap$cov12=cov12  
cap$cov13=cov13  
cap$cov14=cov14  
cap$cov15=cov15  
cap$cov16=cov16  
cap$cov17=cov17  
cap$cov18=cov18  
cap$cov19=cov19  
#head(cap)

Now we fit the model with an age effect but no individual heterogeneity to the simulated dataset:

library(RMark)  
cap.processed=process.data(cap,model="CJS")  
cap.ddl=make.design.data(cap.processed)  
Phi.cov<-list(formula=~cov)  
p.dot=list(formula=~1)  
cov.est<-mark(cap.processed,cap.ddl,model.parameters=list(Phi=Phi.cov,p=p.dot),output = FALSE,delete=T)

Having a look to the parameter estimates, it sounds like the slope of the age effect on survival is estimated positive...

cov.est$results$beta[1:2,1] # intercept and slope of the age effect

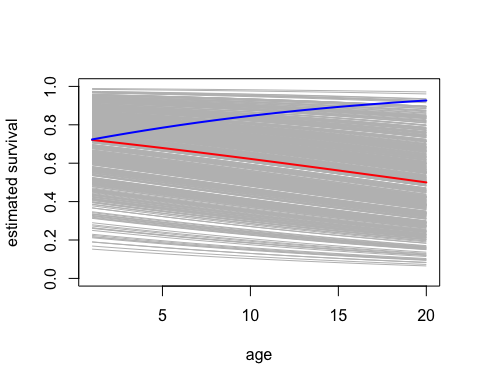
## [1] 0.8805952 0.0828273

expit(cov.est$results$beta[3,1]) # detection prob, after back-transformation

## [1] 0.496648

Which means that at the population level, whenever individual heterogeneity is ignored, then senescence (in red) is completely masked. Even worse, survival is increasing with increasing age (in blue).

plot(age[1,],phi[1,],type='l',col='grey',ylim=c(0,1),xlab='age',ylab='estimated survival')  
for (i in 2:nind){  
 lines(age[i,],phi[i,],type='l',col='grey')  
}  
lines(1:nyear,expit(intercept\_phi + slope\_phi \* 1:nyear),col='red',lwd=2)  
lines(1:nyear,expit(cov.est$results$beta[1,1] + cov.est$results$beta[2,1] \* 1:nyear),col='blue',lwd=2)



Now we fit the model with a random effect in the survival process. The model structure is specified with the model="CJSRandom" option:

cap.processed=process.data(cap,model="CJSRandom")  
cap.ddl=make.design.data(cap.processed)

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 detection to 0 to fit a model with a constant detection probability. In contrast, we let MARK estimate both parameters of the random effect for the survival probability.

# mean survival  
phiage = list(formula=~cov) # covariate-dependent (age here)  
# standard deviation of the random effect on survival is to be estimated  
sigmaphi = list(formula=~1)  
# mean recapture probability  
pct = list(formula=~1)  
# standard deviation of the random effect on recapture is fixed to 0  
# in other words, no random effect on detection  
sigmap.fixed=list(formula=~1,fixed=0)

Let's roll and fit this model:

model.re = mark(cap.processed,cap.ddl,model.parameters=list(Phi=phiage,p=pct,sigmap=sigmap.fixed,sigmaphi=sigmaphi),output = FALSE,delete=T)

Let's have a look to the parameter estimates:

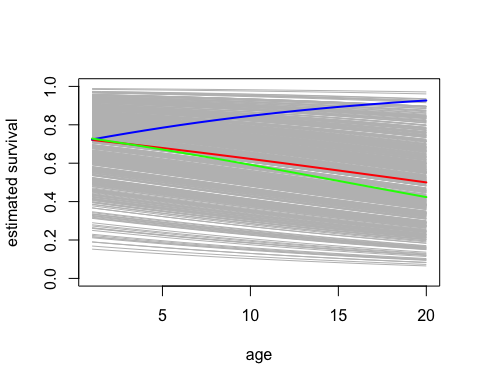
model.re$results$beta

## estimate se lcl ucl  
## sigmaphi:(Intercept) 0.1958479 0.2776931 -0.3484306 0.7401265  
## Phi:(Intercept) 1.0545663 0.1290592 0.8016101 1.3075224  
## Phi:cov -0.0681858 0.0583187 -0.1824904 0.0461189  
## p:(Intercept) 0.0161996 0.0593496 -0.1001257 0.1325248

The sandard deviation of the random effect is estimated on the log scale (I assume since we obtain a confidence interval with a negative lower bound; ask Gary and/or Jeff), hence after back-transformation, the estimate is 1.2163419. Detection probability is estimated on the logit scale, therefore, after back-transformation, we get an estimate of 0.5040498. The intercept and slope of the age-survival relationship are quite close to the values we used to simulate the data.

Now we add to our previous plot the survival as estimated when individual heterogeneity is explicitely accounted for using individual random effects (in green):

plot(age[1,],phi[1,],type='l',col='grey',ylim=c(0,1),xlab='age',ylab='estimated survival')  
for (i in 2:nind){  
 lines(age[i,],phi[i,],type='l',col='grey')  
}  
lines(1:nyear,expit(intercept\_phi + slope\_phi \* 1:nyear),col='red',lwd=2)  
lines(1:nyear,expit(cov.est$results$beta[1,1] + cov.est$results$beta[2,1] \* 1:nyear),col='blue',lwd=2)  
lines(1:nyear,expit(model.re$results$beta[2,1] + model.re$results$beta[3,1] \* 1:nyear),col='green',lwd=2)



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 unusual in that particular situation (see [Gimenez and Choquet 2010](https://dl.dropboxusercontent.com/u/23160641/my-pubs/Gimenez%26Choquet2010Ecology.pdf) for example).

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 survival probability to 0. For the sake of complexity (...), let's use the latter option:

phict = list(formula=~cov) # constant  
sigmaphi = list(formula=~1,fixed=0)  
pct = list(formula=~1)  
sigmap = list(formula=~1,fixed=0)  
model.without.re = mark(cap.processed,cap.ddl,model.parameters=list(Phi=phict,p=pct,sigmap=sigmap,sigmaphi=sigmaphi),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] 0.02211519

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

Last but not least, you might want to check that the age effect is there. To test that, we go for a model with a random effect but without the age effect:

phict = list(formula=~1)  
sigmaphi = list(formula=~1)  
pct = list(formula=~1)  
sigmap.fixed=list(formula=~1,fixed=0)  
model.noagere = mark(cap.processed,cap.ddl,model.parameters=list(Phi=phict,p=pct,sigmap=sigmap.fixed,sigmaphi=sigmaphi),output = FALSE,delete=T)

Then compare AICc:

model.re$results$AICc # age effect, with random effect

## [1] 3616.955

model.noagere$results$AICc # no age effect, with random effect

## [1] 3616.28

## Models with finite mixtures

Here, we again aim at illustrating how not accounting for individual heterogeneity may obscure the detection of senescence in survival. In contrast with the previous section, we now use finite mixtures to deal with heterogeneity. More specifically, we consider a cohort of 1000 individuals that are split into a group of robust individuals in proportion with constant high survival and a group of frail individuals with survival that senesce over the 20 years of the study according to the relationship . We use , , and . Note that we consider the same detection probability for all individuals.

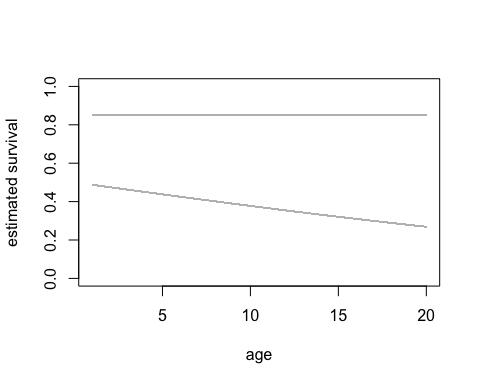
### Data simulation

First simulate data

r = set.seed(3) # for reproducibility  
p = 0.5 # detection  
prop\_class1 = 0.3 # pi  
phi\_class1 = 0.85 # survival or robust ind  
intercept\_phi\_class2 = 0 #beta\_0  
slope\_phi\_class2 = -0.05 # beta\_1  
nind = 1000 # nb of ind  
nyear = 20 # duration of the study  
expit<-function(x){exp(x)/(1+exp(x))} # reciprocal of the logit function  
z<-data<-x<-matrix(NA,nrow=nind,ncol=nyear)  
first<-rep(1,nind)  
age = matrix(NA,nind,nyear)  
phi = matrix(NA,nind,nyear)  
which\_mixture = rep(NA,nind)  
# simulate age-varying survival for each individual,   
# by first assigning them to the robust or frail class, then using the corresponding   
# survival   
for (i in 1:nind){  
 mask <- first[i]:nyear  
 age[i,mask] <- mask - first[i] + 1  
 which\_mixture[i] <- rbinom(1,1,prop\_class1) # assign ind i to a class with prob pi  
 if (which\_mixture[i] == 1){  
 phi[i,mask] <- phi\_class1 # robust  
 } else {   
 phi[i,mask] <- expit(intercept\_phi\_class2 + slope\_phi\_class2 \* age[i,mask])} # frail  
}

Represent graphically survival over time in the two classes:

plot(age[1,],phi[1,],type='l',col='grey',ylim=c(0,1),xlab='age',ylab='estimated survival')  
for (i in 2:nind){  
 lines(age[i,],phi[i,],type='l',col='grey')  
}



Now simulate the encounter histories:

for(i in 1:nind){  
 z[i,first[i]] <- x[i,first[i]] <- 1  
 for(j in (first[i]+1):nyear){  
 z[i,j]<-rbinom(1,1,phi[i,j-1]\*z[i,j-1])  
 x[i,j]<-rbinom(1,1,z[i,j]\*p)  
 }  
}  
his = x  
his[is.na(his)]=0

### Model fitting

First, we format the data we've just simulated so that these can be used with RMark:

k = ncol(his)  
n = nrow(his)  
out = array(dim=n)  
for (i in 1:n){  
 y = (his[i,] > 0) \* 1  
 out[i] = paste(y,collapse="")  
}  
capt.hist = data.frame(ch = out)

Now, we add age as a time-varying individual covariate to the dataset (check out [these notes](https://sites.google.com/site/workshoponcmr/) by Mike Conroy for more details):

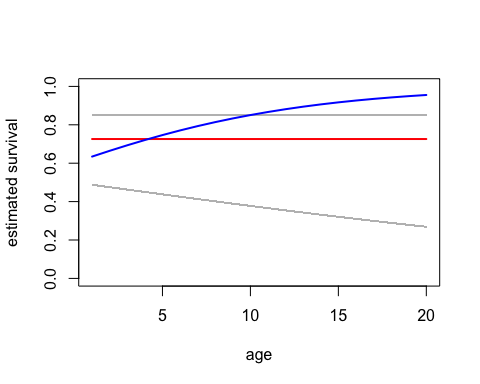
df = data.frame(time=c(1:(nyear-1)),cov=runif(nyear-1))  
simul.data = list(cap.data=capt.hist,cov=df)  
n.ind <- nrow(simul.data$cap.data)  
for (i in 1:nyear){  
 name = paste('cov',i,sep='')  
 assign(name,age[,i])  
}  
cap<-simul.data$cap.data  
# pretty ugly lines of codes to follow, happy to hear for suggestions to make this dynamic  
cap$cov1=cov1  
cap$cov2=cov2  
cap$cov3=cov3  
cap$cov4=cov4  
cap$cov5=cov5  
cap$cov6=cov6  
cap$cov7=cov7  
cap$cov8=cov8  
cap$cov9=cov9  
cap$cov10=cov10  
cap$cov11=cov11  
cap$cov12=cov12  
cap$cov13=cov13  
cap$cov14=cov14  
cap$cov15=cov15  
cap$cov16=cov16  
cap$cov17=cov17  
cap$cov18=cov18  
cap$cov19=cov19  
#head(cap,100)

Now let's fit two models assuming homogeneity, first one with constant survival probability, second one with an age effect:

library(RMark)  
phi.ct = list(formula=~1) # constant survival  
phi.age = list(formula=~cov) # age-dependent survival  
p.ct = list(formula=~1) # constant recapture  
dat.proc = process.data(cap, model="CJS")  
dat.ddl = make.design.data(dat.proc)  
model.hom.phi = mark(dat.proc,dat.ddl,model.parameters=list(Phi=phi.ct,p=p.ct),output = FALSE,delete=T)  
model.hom.phi.age = mark(dat.proc,dat.ddl,model.parameters=list(Phi=phi.age,p=p.ct),output = FALSE,delete=T)

Graphically, we have the estimate from the model with constant survival (in red) vs. age-varying survival (in blue):

plot(age[1,],phi[1,],type='l',col='grey',ylim=c(0,1),xlab='age',ylab='estimated survival')  
for (i in 2:nind){  
 lines(age[i,],phi[i,],type='l',col='grey')  
}  
lines(1:nyear,rep(model.hom.phi$results$real[1,1],nyear),lwd=2,col='red') # add survival from constant model  
lines(1:nyear,expit(model.hom.phi.age$results$beta[1,1]+model.hom.phi.age$results$beta[2,1]\*(1:nyear)),lwd=2,col='blue') # add survival from age model



Again, as in the previous section, it's striking to see that survival is increasing when age increases if individual heterogeneity is ignored. In other words, senescence is masked.

Now let's fit a model with heterogeneity in the survival probability, with constant parameters over time. We define the model structure, by using the model="CJSMixture" option:

# load RMark package  
library(RMark)  
dat.proc2 = process.data(cap, model="CJSMixture")  
dat.ddl2 = make.design.data(dat.proc2)

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

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

Let's fit that model:

model.het = mark(dat.proc2,dat.ddl2,model.parameters=list(Phi=phi.mix,p=p.ct,pi=pi.dot),output = FALSE,delete=T)

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 0.7485328 0.0438346 0.6534786 0.8245147   
## Phi g1 c1 a0 t1 m1 0.5198987 0.0283694 0.4642779 0.5750306   
## Phi g1 c1 a0 t1 m2 0.8753679 0.0148680 0.8431916 0.9017114   
## p g1 c1 a1 t2 m1 0.5071017 0.0125425 0.4825210 0.5316482

The proportion of individuals in mixture 1 is:

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

## [1] 0.7485328

with survival probability:

phi1 = model.het$results$real[2,1]  
phi1

## [1] 0.5198987

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

phi2 = model.het$results$real[3,1]  
phi2

## [1] 0.8753679

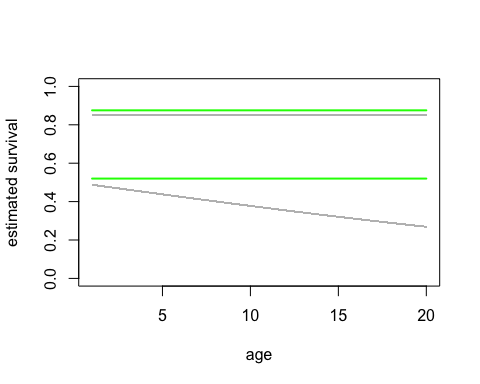
Lastly, recapture probability is:

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

## [1] 0.5071017

Let's have a look graphically:

plot(age[1,],phi[1,],type='l',col='grey',ylim=c(0,1),xlab='age',ylab='estimated survival')  
for (i in 2:nind){  
 lines(age[i,],phi[i,],type='l',col='grey')  
}  
lines(1:nyear,rep(phi1,nyear),lwd=2,col='green') # add survival from first class  
lines(1:nyear,rep(phi2,nyear),lwd=2,col='green') # add survival from second class



Not too bad. Obviously, for frail individuals, we miss the age effect to be able to detect senescence. Now let's add age to this model:

# age-dependent heterogenous survival  
phi.mix.age = list(formula=~mixture\*cov)

Let's fit that model:

model.het.age = mark(dat.proc2,dat.ddl2,model.parameters=list(Phi=phi.mix.age,p=p.ct,pi=pi.dot),output = FALSE,delete=T)

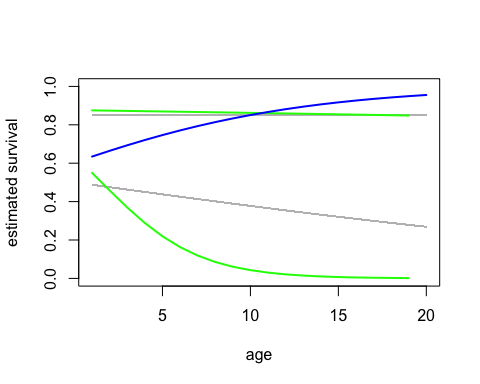
Let's have a look to the parameter estimates of the model with heterogeneity:

model.het.age$results$real[,1:4]

## estimate se lcl ucl  
## pi g1 a0 t1 m1 0.7057418 0.0787821 5.327638e-01 0.8345661  
## Phi g1 c1 a0 t1 m1 0.5494191 0.0316805 4.868776e-01 0.6104370  
## Phi g1 c1 a1 t2 m1 0.4579715 0.0515541 3.599520e-01 0.5593560  
## Phi g1 c1 a2 t3 m1 0.3692706 0.0908708 2.141565e-01 0.5570889  
## Phi g1 c1 a3 t4 m1 0.2886021 0.1203551 1.139360e-01 0.5613854  
## Phi g1 c1 a4 t5 m1 0.2194253 0.1345805 5.683570e-02 0.5673480  
## Phi g1 c1 a5 t6 m1 0.1630300 0.1346296 2.739310e-02 0.5739486  
## Phi g1 c1 a6 t7 m1 0.1189208 0.1245013 1.297560e-02 0.5808454  
## Phi g1 c1 a7 t8 m1 0.0855262 0.1087293 6.094200e-03 0.5878921  
## Phi g1 c1 a8 t9 m1 0.0608616 0.0910173 2.850400e-03 0.5950148  
## Phi g1 c1 a9 t10 m1 0.0429756 0.0738158 1.330400e-03 0.6021712  
## Phi g1 c1 a10 t11 m1 0.0301770 0.0584514 6.203643e-04 0.6093351  
## Phi g1 c1 a11 t12 m1 0.0211060 0.0454482 2.891120e-04 0.6164886  
## Phi g1 c1 a12 t13 m1 0.0147202 0.0348428 1.346972e-04 0.6236193  
## Phi g1 c1 a13 t14 m1 0.0102463 0.0264183 6.274463e-05 0.6307174  
## Phi g1 c1 a14 t15 m1 0.0071223 0.0198549 2.922455e-05 0.6377754  
## Phi g1 c1 a15 t16 m1 0.0049460 0.0148160 1.361091e-05 0.6447873  
## Phi g1 c1 a16 t17 m1 0.0034324 0.0109911 6.338747e-06 0.6517476  
## Phi g1 c1 a17 t18 m1 0.0023809 0.0081137 2.951902e-06 0.6586522  
## Phi g1 c1 a18 t19 m1 0.0016510 0.0059646 1.374632e-06 0.6654969  
## Phi g1 c1 a0 t1 m2 0.8750541 0.0438460 7.614041e-01 0.9389127  
## Phi g1 c1 a1 t2 m2 0.8736803 0.0402685 7.718430e-01 0.9339525  
## Phi g1 c1 a2 t3 m2 0.8722936 0.0366601 7.818314e-01 0.9286685  
## Phi g1 c1 a3 t4 m2 0.8708939 0.0330365 7.913361e-01 0.9230675  
## Phi g1 c1 a4 t5 m2 0.8694812 0.0294223 8.003080e-01 0.9171727  
## Phi g1 c1 a5 t6 m2 0.8680553 0.0258569 8.086703e-01 0.9110361  
## Phi g1 c1 a6 t7 m2 0.8666163 0.0224057 8.162962e-01 0.9047609  
## Phi g1 c1 a7 t8 m2 0.8651640 0.0191815 8.229668e-01 0.8985431  
## Phi g1 c1 a8 t9 m2 0.8636983 0.0163808 8.283000e-01 0.8927438  
## Phi g1 c1 a9 t10 m2 0.8622193 0.0143275 8.316632e-01 0.8879760  
## Phi g1 c1 a10 t11 m2 0.8607268 0.0134500 8.322180e-01 0.8850606  
## Phi g1 c1 a11 t12 m2 0.8592208 0.0140497 8.293711e-01 0.8845754  
## Phi g1 c1 a12 t13 m2 0.8577012 0.0160312 8.233020e-01 0.8863282  
## Phi g1 c1 a13 t14 m2 0.8561679 0.0190272 8.147203e-01 0.8896000  
## Phi g1 c1 a14 t15 m2 0.8546210 0.0226907 8.042951e-01 0.8937156  
## Phi g1 c1 a15 t16 m2 0.8530602 0.0267944 7.924634e-01 0.8982359  
## Phi g1 c1 a16 t17 m2 0.8514856 0.0312051 7.794813e-01 0.9029074  
## Phi g1 c1 a17 t18 m2 0.8498971 0.0358451 7.655001e-01 0.9075863  
## Phi g1 c1 a18 t19 m2 0.8482946 0.0406685 7.506149e-01 0.9121906  
## p g1 c1 a1 t2 m1 0.5024941 0.0128030 4.774151e-01 0.5275607

Let's have a look graphically:

plot(age[1,],phi[1,],type='l',col='grey',ylim=c(0,1),xlab='age',ylab='estimated survival')  
for (i in 2:nind){  
 lines(age[i,],phi[i,],type='l',col='grey')  
}  
phi1 = model.het.age$results$real[2:20,1]  
phi2 = model.het.age$results$real[21:39,1]  
lines(1:(nyear-1),phi1,lwd=2,col='green') # add survival from first class  
lines(1:(nyear-1),phi2,lwd=2,col='green') # add survival from second class  
lines(1:nyear,expit(model.hom.phi.age$results$beta[1,1]+model.hom.phi.age$results$beta[2,1]\*(1:nyear)),lwd=2,col='blue') # add survival from age model



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 compare the AIC values:

summary(model.het.age)$AICc # heterogeneity and age

## [1] 5584.445

summary(model.hom.phi.age)$AICc # age

## [1] 5602.422

summary(model.het)$AICc # heterogeneity

## [1] 5586.721

summary(model.hom.phi)$AICc # null

## [1] 5714.987

Sounds like there is some heterogeneity and an age effect.