Local minima in multistate capture-recapture models

## Model

We consider a state-space formulation of multistate capture-recapture model. We refer to [Gimenez et al. (2007). State-space modelling of data on marked individuals. Ecological Modelling 206: 431-438](https://dl.dropboxusercontent.com/u/23160641/my-pubs/Servantyetal2010Ecology.pdf) and [Servanty et al. (2010). Assessing whether mortality is additive using marked animals: a Bayesian state-space modeling approach. Ecology 91: 1916–1923](https://dl.dropboxusercontent.com/u/23160641/my-pubs/Gimenezetal2007EcologicalModelling.pdf) for the theory.

## Read in data

I use female roe deer data monitored over 20 years and provided by J.-M. Gaillard. Population in Trois Fontaine managed by ONCFS. Two states, with and without fawn.

his <- matrix(c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,1,  
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,1,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,2,1,  
0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,2,2,0,1,1,1,  
0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,2,1,2,0,1,  
0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,1,1,1,1,  
0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,2,1,2,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,1,0,0,2,0,1,2,2,0,1,  
0,0,0,0,0,0,0,0,0,0,0,1,2,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,1,0,1,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,2,2,1,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,2,1,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,2,1,2,0,1,1,1,0,0,1,  
0,0,0,0,0,0,0,0,0,0,0,2,2,0,2,2,2,2,1,0,1,  
0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,0,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,2,1,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,2,0,0,0,2,2,0,1,1,0,1,  
0,0,0,0,0,0,0,0,0,0,2,1,2,1,1,2,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,0,2,2,2,1,0,2,2,2,0,0,1,  
0,0,0,0,0,0,0,0,0,1,2,2,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,0,2,0,2,2,0,0,0,0,0,1,0,0,1,  
0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,1,0,0,2,2,0,2,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,2,0,0,2,1,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,0,2,1,1,0,2,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,1,0,0,0,0,0,2,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,2,2,0,2,2,0,0,2,0,2,0,0,0,0,1,  
0,0,0,0,0,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,0,2,2,0,0,0,1,0,2,0,0,0,0,0,0,1,  
0,0,0,0,0,0,2,2,0,0,0,2,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,1,1,1,0,0,1,2,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,1,1,2,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,1,1,2,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,1,2,2,0,1,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,2,1,2,0,0,2,2,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,2,1,2,2,2,2,2,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,0,2,2,2,2,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,2,0,2,0,2,1,0,0,0,2,0,0,0,0,0,0,1,  
0,0,0,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,2,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,2,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,2,0,2,0,0,2,2,0,0,0,0,0,0,0,0,0,1,  
0,0,0,0,2,0,2,2,2,2,2,2,0,0,0,0,0,0,0,0,1,  
0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,2,2,0,2,2,0,2,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,2,2,2,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,0,2,2,2,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,2,2,2,1,2,0,0,0,2,2,0,1,0,0,0,0,0,0,1,  
0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,2,0,0,0,2,0,0,0,0,1,0,0,0,0,0,0,0,0,1,  
0,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,2,2,0,0,2,2,1,0,0,0,0,0,0,0,0,0,0,0,1,  
0,0,2,2,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,2,2,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,  
0,2,2,2,2,2,2,2,0,0,2,2,0,1,0,0,0,0,0,0,1,  
2,1,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1),byrow=T,ncol=21)

## Frequentist Analysis with Mark

Let's format the data to be analysed.

his = his[,-ncol(his)] # remove last column, these are the counts  
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)

Load RMark package

library(RMark)

## This is RMark 2.2.0

Process data and create default design matrix

mstrata.processed=process.data(capt.hist,model="Multistrata")   
mstrata.ddl=make.design.data(mstrata.processed)

Define structure on parameters, all state dependent

# survival probability   
S.site = list(formula=~stratum)   
# detection probability   
p.site = list(formula=~stratum)   
# transition probs   
Psi.site = list(formula=~-1+stratum:tostratum)

Fit model

mstrata.mod = mark(mstrata.processed,mstrata.ddl,model.parameters=list(S=S.site,p= p.site,Psi=Psi.site),output=FALSE,silent=T,delete=T)

Let's have a look to the parameter estimates

mstrata.mod$results$real[c(1:5,195),]

## estimate se lcl ucl  
## S s1 g1 c1 a0 o1 t1 0.7894772 0.0407035000 6.988374e-01 0.8583662  
## S s2 g1 c1 a0 o1 t1 0.8624650 0.0484990000 7.377958e-01 0.9332232  
## p s1 g1 c1 a1 o1 t2 0.2503825 0.0381443000 1.831870e-01 0.3322017  
## p s2 g1 c1 a1 o1 t2 0.9999999 0.0001230681 1.699050e-297 1.0000000  
## Psi s1 to2 g1 c1 a0 o1 t1 0.3185984 0.0425212000 2.415652e-01 0.4070140  
## Psi s2 to1 g1 c1 a0 o1 t1 0.4814442 0.0457104000 3.933814e-01 0.5706752  
## fixed note  
## S s1 g1 c1 a0 o1 t1   
## S s2 g1 c1 a0 o1 t1   
## p s1 g1 c1 a1 o1 t2   
## p s2 g1 c1 a1 o1 t2   
## Psi s1 to2 g1 c1 a0 o1 t1   
## Psi s2 to1 g1 c1 a0 o1 t1

Now let's re-run the same model but changing the initial values

nb\_inits = 5 # five runs with diff inits  
set.seed(3) # to reproduce the results  
res\_dev = rep(NA,nb\_inits)  
res = list()  
for (i in 1:nb\_inits){  
 inits = rnorm(6,0,1)  
 mstrata.sim = mark(mstrata.processed,mstrata.ddl,model.parameters=  
 list(S=S.site,p= p.site,Psi=Psi.site),output=FALSE,silent=TRUE,delete=T,initial = inits)  
 res\_dev[i] = mstrata.sim$results$deviance  
 res[[i]] = mstrata.sim$results  
}

Display the deviance values

res\_dev

## [1] 674.9109 669.3424 669.3424 669.3424 674.9109

There seems to be a local minimum. Let's have a look to the estimates corresponding to the global minimum

res[[2]]$real[c(1:5,195),1:4]

## estimate se lcl ucl  
## S s1 g1 c1 a0 o1 t1 0.7894771 0.0407036000 6.988372e-01 0.8583661  
## S s2 g1 c1 a0 o1 t1 0.8624651 0.0484991000 7.377956e-01 0.9332233  
## p s1 g1 c1 a1 o1 t2 0.2503827 0.0381444000 1.831871e-01 0.3322019  
## p s2 g1 c1 a1 o1 t2 1.0000000 0.0001004167 2.139179e-297 1.0000000  
## Psi s1 to2 g1 c1 a0 o1 t1 0.3185986 0.0425212000 2.415654e-01 0.4070142  
## Psi s2 to1 g1 c1 a0 o1 t1 0.4814443 0.0457104000 3.933814e-01 0.5706752

What about the estimates with a local minimum?

res[[1]]$real[c(1:5,195),1:4]

## estimate se lcl ucl  
## S s1 g1 c1 a0 o1 t1 0.6927312 0.0688786000 5.445499e-01 0.8095616  
## S s2 g1 c1 a0 o1 t1 0.8360109 0.0221532000 7.878692e-01 0.8749611  
## p s1 g1 c1 a1 o1 t2 0.9999990 0.0001886039 2.353401e-157 1.0000000  
## p s2 g1 c1 a1 o1 t2 0.5358716 0.0358389000 4.653811e-01 0.6049573  
## Psi s1 to2 g1 c1 a0 o1 t1 0.7112855 0.0693976000 5.595374e-01 0.8269246  
## Psi s2 to1 g1 c1 a0 o1 t1 0.1324593 0.0212327000 9.608190e-02 0.1798688

## Graphical examination

We use a graph of profile deviance to examine local minima. Let's do it for survival first:

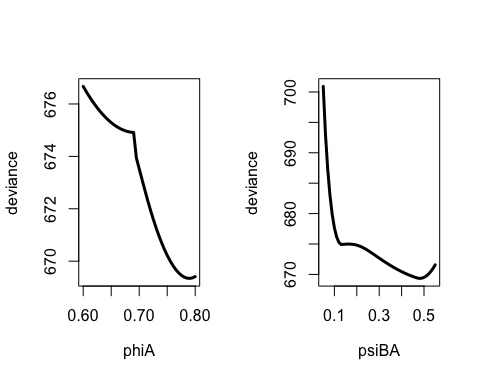
fr = mstrata.ddl$S$stratum  
ii = (fr==1)  
phi.fix = as.numeric(row.names(mstrata.ddl$S[ii,]))  
  
grid = seq(.6,0.8,0.01/2)  
#length(grid)  
dev = rep(NA,length(grid))  
ind = 1  
for (i in grid){  
 S.fix = list(formula=~stratum,fixed=list(index=phi.fix,value=i)) # survival depends on states   
 res.fix = mark(mstrata.processed,mstrata.ddl,model.parameters=list(S=S.fix,p=p.site,Psi=Psi.site),output = FALSE,delete=T,silent=T)  
 dev[ind] = res.fix$results$deviance  
 ind = ind + 1  
}  
dev\_phi = dev  
grid\_phi = grid

Then for a transition

fr = mstrata.ddl$Psi$stratum  
to = mstrata.ddl$Psi$tostratum  
ii = (fr==2 & to==1)  
psi.fix = as.numeric(row.names(mstrata.ddl$Psi[ii,])) # careful not to define Psi.fix twice  
  
grid = seq(.05,0.55,0.01)  
#length(grid)  
dev = rep(NA,length(grid))  
ind = 1  
for (i in grid){  
 Psi.fix = list(formula=~-1+stratum:tostratum,fixed=list(index=psi.fix,value=i),link='logit') # needs logit here, mlogit is a mess  
 res.fix = mark(mstrata.processed,mstrata.ddl,model.parameters=list(S=S.site,p=p.site,Psi=Psi.fix),output = FALSE,delete=T,silent=T)  
 dev[ind] = res.fix$results$deviance  
 ind = ind + 1  
}  
dev\_psi = dev  
grid\_psi = grid

Let's visualize the results

par(mfrow=c(1,2))  
plot(grid\_phi,dev\_phi,type='l',xlab=expression(phiA),ylab='deviance',col='black',lwd=3)  
plot(grid\_psi,dev\_psi,type='l',xlab=expression(psiBA),ylab='deviance',col='black',lwd=3)



## Frequentist analysis with ADMB

Load R2admb package:

library(R2admb)

Define model in ADMB language

model <-   
paste("  
DATA\_SECTION  
 init\_int k // Number of capture occations  
 init\_int nh // Number of encouter histories  
 init\_ivector e(1,nh) // Date of first capture  
 init\_imatrix data(1,nh,1,k) // Data matrix  
 init\_ivector eff(1,nh) // NUmber of individuals per capture history  
 init\_ivector garb(1,nh) // State of first capture  
PARAMETER\_SECTION  
 init\_bounded\_number logit\_pi(-20.0,20.0,1) // Prop  
 init\_bounded\_number logit\_phi1(-20.0,20.0,1) // Survival  
 init\_bounded\_number logit\_phi2(-20.0,20.0,1) // Survival  
 init\_bounded\_number logit\_det1(-20.0,20.0,1) // Capture probability  
 init\_bounded\_number logit\_det2(-20.0,20.0,1) // Capture probability  
 init\_bounded\_number logit\_psi1(-20.0,20.0,1) // Survival  
 init\_bounded\_number logit\_psi2(-20.0,20.0,1) // Survival  
 objective\_function\_value g  
 number pi  
 number phi1  
 number phi2  
 number det1  
 number det2  
 number psi1  
 number psi2  
PROCEDURE\_SECTION  
 pi = mfexp(logit\_pi);  
 pi = pi/(1+pi);   
 phi1 = mfexp(logit\_phi1);  
 phi1 = phi1/(1+phi1);   
 phi2 = mfexp(logit\_phi2);  
 phi2 = phi2/(1+phi2);   
 det1 = mfexp(logit\_det1);  
 det1 = det1/(1+det1);   
 det2 = mfexp(logit\_det2);  
 det2 = det2/(1+det2);   
 psi1 = mfexp(logit\_psi1);  
 psi1 = psi1/(1+psi1);   
 psi2 = mfexp(logit\_psi2);  
 psi2 = psi2/(1+psi2);   
 dvar\_vector prop(1,3);  
 prop(1) = pi; prop(2) = 1.0 - pi; prop(3) = 0.0;  
 // B and BE are transposed:  
 // we need to have obs in rows and states in columns  
 // so that we can access rows of BE and B in the ADMB  
 // way, e.g. BE(oe)  
 dvar\_matrix BE(1,3,1,3);  
 BE(1,1) = 1.0;  
 BE(1,2) = 0.0;  
 BE(1,3) = 0.0;  
 BE(2,1) = 0.0;  
 BE(2,2) = 1.0;  
 BE(2,3) = 0.0;  
 BE(3,1) = 0.0;  
 BE(3,2) = 0.0;  
 BE(3,3) = 1.0;  
 dvar\_matrix B(1,3,1,3);  
 B(1,1) = det1;  
 B(1,2) = 0.0;  
 B(1,3) = 0.0;  
 B(2,1) = 0.0;  
 B(2,2) = det2;  
 B(2,3) = 0.0;  
 B(3,1) = 1.0 - det1;  
 B(3,2) = 1.0 - det2;  
 B(3,3) = 1.0;  
 dvar\_matrix PHI(1,3,1,3);  
 PHI(1,1) = phi1 \* (1.0 - psi1);  
 PHI(1,2) = phi1 \* psi1;  
 PHI(1,3) = 1.0 - phi1;  
 PHI(2,1) = phi2 \* psi2;  
 PHI(2,2) = phi2 \* (1.0 - psi2);  
 PHI(2,3) = 1.0 - phi2;  
 PHI(3,1) = 0.0;  
 PHI(3,2) = 0.0;  
 PHI(3,3) = 1.0;  
 for(int i=1;i<=nh;i++){  
 int ei = e(i); // date of marking  
 int oe = garb(i); // initial obs  
 ivector evennt = data(i);  
 dvar\_vector ALPHA = elem\_prod(prop,BE(oe));  
 for(int j=ei+1;j<=k;j++){  
 ALPHA = elem\_prod(ALPHA\*PHI,B(evennt(j)));  
 g -= log(sum(ALPHA))\*eff(i);  
 }  
 }  
")  
writeLines(model,"model.tpl")

Specify where ADMB is

setup\_admb("/Applications/ADMBTerminal.app/admb")

## [1] "/Applications/ADMBTerminal.app/admb"

Prepare data

dat = his   
k = ncol(dat) # nb of capture occasions   
nh = nrow(dat) # nb of individuals   
eff = rep(1,nh)  
fc <- NULL  
init.state <- NULL  
for (i in 1:nh){  
 temp <- 1:k  
 fc <- c(fc,min(which(dat[i,]!=0)))  
 init.state <- c(init.state,dat[i,fc[i]])  
}  
dat[dat==0] <- 3 # replace 0's of non-detection by 3's  
df = list(k=k,nh=nh,e=fc,data=dat,eff=eff,garb=init.state)

Fit model

params <- list(logit\_pi = 0,logit\_phi1 = 0, logit\_phi2 = 0, logit\_det1 = 0, logit\_det2 = 0,logit\_psi1 = 0, logit\_psi2 = 0) ## starting parameters  
res <- do\_admb('model', data=df, params = params,verbose=TRUE)

## writing data and parameter files ...  
## compiling with args: ' -s ' ...  
## compile output:  
## \*\*\* Parse: model.tpl tpl2cpp model || tpl2rem model \*\*\* Compile: model.cpp c++ -c -std=c++14 -O3 -I. -I"/Applications/ADMBTerminal.app/admb/include" -I"/Applications/ADMBTerminal.app/admb/contrib/include" -omodel.obj model.cpp \*\*\* Linking: model.obj c++ -std=c++14 -O3 -omodel model.obj "/Applications/ADMBTerminal.app/admb/lib/libadmb-contrib.a" Successfully built executable.   
## compile log:  
##   
## running compiled executable with args: ' '...  
## Run output:  
##   
##   
##   
##   
##   
## Initial statistics: 7 variables; iteration 0; function evaluation 0; phase 1  
## Function value 5.7642880e+03; maximum gradient component mag -2.4747e+04  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 0.00000 8.24668e+03 | 2 0.00000 -8.15948e+03 | 3 0.00000 -2.47466e+04  
## 4 0.00000 1.73534e+03 | 5 0.00000 -1.58105e+04 | 6 0.00000 -2.78031e+03  
## 7 0.00000 1.47655e+04 |  
##   
## Intermediate statistics: 7 variables; iteration 10; function evaluation 23; phase 1  
## Function value 4.3835920e+03; maximum gradient component mag -1.7215e+03  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.03892 1.08638e+02 | 2 0.00655 -1.72147e+03 | 3 0.04980 -1.54759e+03  
## 4 0.11943 -4.76513e+01 | 5 0.00937 -1.08907e+03 | 6 0.03209 -1.16828e+01  
## 7 -0.07117 -1.03375e+03 |  
## 7 variables; iteration 20; function evaluation 34; phase 1  
## Function value 4.3635513e+03; maximum gradient component mag 3.2371e+01  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.03952 9.82803e+00 | 2 0.02340 -2.38586e+01 | 3 0.05344 2.27249e+01  
## 4 0.10758 -4.05730e+00 | 5 0.01088 3.23706e+01 | 6 0.03328 -1.76291e+00  
## 7 -0.06621 1.24650e+01 |  
## 7 variables; iteration 30; function evaluation 44; phase 1  
## Function value 4.3633833e+03; maximum gradient component mag -2.3618e+00  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.03958 -5.00217e-02 | 2 0.02348 8.68237e-02 | 3 0.05325 -7.43738e-01  
## 4 0.24968 -7.12736e-02 | 5 0.01049 -2.36178e+00 | 6 0.03449 4.82657e-01  
## 7 -0.06737 -2.34716e+00 |  
## 7 variables; iteration 40; function evaluation 54; phase 1  
## Function value 4.3633809e+03; maximum gradient component mag 3.5047e-01  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.03958 2.24958e-02 | 2 0.02348 -1.72662e-01 | 3 0.05325 7.29666e-02  
## 4 0.47336 -1.53086e-04 | 5 0.01049 3.50469e-01 | 6 0.03450 3.39353e-02  
## 7 -0.06737 4.83928e-02 |  
## 7 variables; iteration 50; function evaluation 64; phase 1  
## Function value 4.3633809e+03; maximum gradient component mag 5.6271e-03  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.03958 3.86775e-04 | 2 0.02348 -3.16799e-03 | 3 0.05325 1.02598e-03  
## 4 0.78036 -1.38824e-07 | 5 0.01049 5.62708e-03 | 6 0.03450 8.44139e-04  
## 7 -0.06737 -7.83391e-05 |  
##   
## - final statistics:  
## 7 variables; iteration 51; function evaluation 65  
## Function value 4.3634e+03; maximum gradient component mag 4.8016e-05  
## Exit code = 1; converg criter 1.0000e-04  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.03958 3.33208e-06 | 2 0.02348 -2.85145e-05 | 3 0.05325 8.04471e-06  
## 4 0.79797 -3.63062e-08 | 5 0.01049 4.80157e-05 | 6 0.03450 8.49209e-06  
## 7 -0.06737 -4.25146e-06 |  
## Estimating row 1 out of 7 for hessian  
## Estimating row 2 out of 7 for hessian  
## Estimating row 3 out of 7 for hessian  
## Estimating row 4 out of 7 for hessian  
## Estimating row 5 out of 7 for hessian  
## Estimating row 6 out of 7 for hessian  
## Estimating row 7 out of 7 for hessian  
##   
##   
## reading output ...

Examine results

summary(res)

## Model file: model   
## Negative log-likelihood: 4363.4 AIC: 8740.8   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## logit\_pi -1.24265 0.07778 -15.976 < 2e-16 \*\*\*  
## logit\_phi1 0.73740 0.11008 6.699 2.11e-11 \*\*\*  
## logit\_phi2 1.67095 0.05572 29.991 < 2e-16 \*\*\*  
## logit\_det1 19.00137 4265.72729 0.004 0.996   
## logit\_det2 0.32960 0.04779 6.897 5.32e-12 \*\*\*  
## logit\_psi1 1.08333 0.12398 8.738 < 2e-16 \*\*\*  
## logit\_psi2 -2.11260 0.06470 -32.652 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

res$coefficients

## logit\_pi logit\_phi1 logit\_phi2 logit\_det1 logit\_det2 logit\_psi1   
## -1.2426517 0.7374045 1.6709454 19.0013670 0.3295998 1.0833302   
## logit\_psi2   
## -2.1126039

1/(1+exp(-res$coefficients))

## logit\_pi logit\_phi1 logit\_phi2 logit\_det1 logit\_det2 logit\_psi1   
## 0.2239748 0.6764280 0.8417018 1.0000000 0.5816620 0.7471237   
## logit\_psi2   
## 0.1078778

res$se

## logit\_pi logit\_phi1 logit\_phi2 logit\_det1 logit\_det2 logit\_psi1   
## 7.7781e-02 1.1008e-01 5.5716e-02 4.2657e+03 4.7791e-02 1.2398e-01   
## logit\_psi2   
## 6.4701e-02

Fit again model, changing initial values

params <- list(logit\_pi = -0.5,logit\_phi1 = -0.5, logit\_phi2 = -0.5, logit\_det1 = -0.5, logit\_det2 = -0.5,logit\_psi1 = -0.5, logit\_psi2 = -0.5) ## starting parameters  
res <- do\_admb('model', data=df, params = params,verbose=TRUE)

## writing data and parameter files ...  
## compiling with args: ' -s ' ...  
## compile output:  
## \*\*\* Parse: model.tpl tpl2cpp model || tpl2rem model \*\*\* Compile: model.cpp c++ -c -std=c++14 -O3 -I. -I"/Applications/ADMBTerminal.app/admb/include" -I"/Applications/ADMBTerminal.app/admb/contrib/include" -omodel.obj model.cpp \*\*\* Linking: model.obj c++ -std=c++14 -O3 -omodel model.obj "/Applications/ADMBTerminal.app/admb/lib/libadmb-contrib.a" Successfully built executable.   
## compile log:  
##   
## running compiled executable with args: ' '...  
## Run output:  
##   
##   
##   
##   
##   
## Initial statistics: 7 variables; iteration 0; function evaluation 0; phase 1  
## Function value 6.4529233e+03; maximum gradient component mag -3.5199e+04  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.01592 4.58658e+03 | 2 -0.01592 -1.05579e+04 | 3 -0.01592 -3.51995e+04  
## 4 -0.01592 -2.53878e+03 | 5 -0.01592 -2.26527e+04 | 6 -0.01592 -3.67465e+03  
## 7 -0.01592 1.06732e+04 |  
##   
## Intermediate statistics: 7 variables; iteration 10; function evaluation 22; phase 1  
## Function value 4.3646460e+03; maximum gradient component mag -6.2071e+02  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.03941 2.75700e+01 | 2 0.02437 -4.34053e+01 | 3 0.05167 -6.20709e+02  
## 4 0.06121 -5.98626e+00 | 5 0.01244 1.16247e+02 | 6 0.03010 1.98334e+01  
## 7 -0.06235 1.00584e+02 |  
## 7 variables; iteration 20; function evaluation 32; phase 1  
## Function value 4.3636796e+03; maximum gradient component mag -1.0840e+02  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.03959 -9.07306e-01 | 2 0.02425 2.35537e+01 | 3 0.05348 -2.70984e+00  
## 4 0.09191 -1.45604e+01 | 5 0.01075 -8.13619e+01 | 6 0.03288 1.17473e+01  
## 7 -0.06611 -1.08402e+02 |  
## 7 variables; iteration 30; function evaluation 42; phase 1  
## Function value 4.3633826e+03; maximum gradient component mag -2.2268e+00  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.03958 -1.71801e-01 | 2 0.02347 -1.27635e+00 | 3 0.05324 -2.22680e+00  
## 4 0.26293 -4.81314e-02 | 5 0.01050 1.03265e-01 | 6 0.03451 1.24823e+00  
## 7 -0.06737 -1.58171e+00 |  
## 7 variables; iteration 40; function evaluation 52; phase 1  
## Function value 4.3633809e+03; maximum gradient component mag 7.6698e-03  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.03958 1.04291e-03 | 2 0.02348 4.27098e-03 | 3 0.05325 7.66977e-03  
## 4 0.51672 -5.42116e-05 | 5 0.01049 -6.86554e-03 | 6 0.03450 -2.36502e-03  
## 7 -0.06737 -6.07007e-03 |  
## 7 variables; iteration 50; function evaluation 62; phase 1  
## Function value 4.3633809e+03; maximum gradient component mag 1.2138e-04  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.03958 1.45630e-05 | 2 0.02348 6.74379e-05 | 3 0.05325 1.21379e-04  
## 4 0.80366 -7.08182e-09 | 5 0.01049 -7.55952e-05 | 6 0.03450 -4.68928e-05  
## 7 -0.06737 -3.79112e-05 |  
##   
## - final statistics:  
## 7 variables; iteration 51; function evaluation 63  
## Function value 4.3634e+03; maximum gradient component mag 3.2153e-05  
## Exit code = 1; converg criter 1.0000e-04  
## Var Value Gradient |Var Value Gradient |Var Value Gradient  
## 1 -0.03958 3.83149e-06 | 2 0.02348 1.78164e-05 | 3 0.05325 3.21529e-05  
## 4 0.80502 -2.97830e-10 | 5 0.01049 -1.95970e-05 | 6 0.03450 -1.25231e-05  
## 7 -0.06737 -9.22151e-06 |  
## Estimating row 1 out of 7 for hessian  
## Estimating row 2 out of 7 for hessian  
## Estimating row 3 out of 7 for hessian  
## Estimating row 4 out of 7 for hessian  
## Estimating row 5 out of 7 for hessian  
## Estimating row 6 out of 7 for hessian  
## Estimating row 7 out of 7 for hessian  
##   
##   
## reading output ...

Examine results

summary(res)

## Model file: model   
## Negative log-likelihood: 4363.4 AIC: 8740.8   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## logit\_pi -1.24265 0.07778 -15.976 < 2e-16 \*\*\*  
## logit\_phi1 0.73740 0.11008 6.699 2.11e-11 \*\*\*  
## logit\_phi2 1.67095 0.05572 29.991 < 2e-16 \*\*\*  
## logit\_det1 19.06927 4257.85794 0.004 0.996   
## logit\_det2 0.32960 0.04779 6.897 5.32e-12 \*\*\*  
## logit\_psi1 1.08333 0.12398 8.738 < 2e-16 \*\*\*  
## logit\_psi2 -2.11260 0.06470 -32.652 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

res$coefficients

## logit\_pi logit\_phi1 logit\_phi2 logit\_det1 logit\_det2 logit\_psi1   
## -1.2426517 0.7374045 1.6709454 19.0692710 0.3295998 1.0833302   
## logit\_psi2   
## -2.1126039

1/(1+exp(-res$coefficients))

## logit\_pi logit\_phi1 logit\_phi2 logit\_det1 logit\_det2 logit\_psi1   
## 0.2239748 0.6764280 0.8417018 1.0000000 0.5816620 0.7471237   
## logit\_psi2   
## 0.1078778

res$se

## logit\_pi logit\_phi1 logit\_phi2 logit\_det1 logit\_det2 logit\_psi1   
## 7.7781e-02 1.1008e-01 5.5716e-02 4.2579e+03 4.7791e-02 1.2398e-01   
## logit\_psi2   
## 6.4701e-02

## Bayesian analysis with Jags

Let us get the occasion of first capture for each individual:

get.first <- function(x) min(which(x!=0))  
f <- apply(his, 1, get.first)

Recode the data such that 1 is seen alive in A (with fawn), 2 is seen alive in B (without fawn), 3 = not seen:

his\_recoded <- his  
his\_recoded[his\_recoded==0] <- 3

Then we fit a multistate model:

{  
sink("state\_on\_survival.jags")  
cat("  
model {  
  
# -------------------------------------------------  
# Parameters:  
# phiA: survival probability site A  
# phiB: survival probability site B  
# psiAB: movement probability from site A to site B  
# psiBA: movement probability from site B to site A  
# pA: recapture probability site A  
# pB: recapture probability site B  
# -------------------------------------------------  
# States (S):  
# 1 alive at A  
# 2 alive at B  
# 3 dead  
# Observations (O):   
# 1 seen at A   
# 2 seen at B  
# 3 not seen  
# -------------------------------------------------  
  
# Priors  
 phiA ~ dunif(0, 1)  
 phiB ~ dunif(0, 1)  
 psiAB ~ dunif(0, 1)  
 psiBA ~ dunif(0, 1)  
 pA ~ dunif(0, 1)  
 pB ~ dunif(0, 1)  
  
# Define state-transition and observation matrices  
for (i in 1:nind){   
 # Define probabilities of state S(t+1) given S(t)  
 for (t in f[i]:(n.occasions-1)){  
 ps[1,i,t,1] <- phiA \* (1-psiAB)  
 ps[1,i,t,2] <- phiA \* psiAB  
 ps[1,i,t,3] <- 1-phiA  
 ps[2,i,t,1] <- phiB \* psiBA  
 ps[2,i,t,2] <- phiB \* (1-psiBA)  
 ps[2,i,t,3] <- 1-phiB  
 ps[3,i,t,1] <- 0  
 ps[3,i,t,2] <- 0  
 ps[3,i,t,3] <- 1  
   
 # Define probabilities of O(t) given S(t)  
 po[1,i,t,1] <- pA  
 po[1,i,t,2] <- 0  
 po[1,i,t,3] <- 1-pA  
 po[2,i,t,1] <- 0  
 po[2,i,t,2] <- pB  
 po[2,i,t,3] <- 1-pB  
 po[3,i,t,1] <- 0  
 po[3,i,t,2] <- 0  
 po[3,i,t,3] <- 1  
 } #t  
 } #i  
  
# Likelihood   
for (i in 1:nind){  
 # Define latent state at first capture  
 z[i,f[i]] <- y[i,f[i]]  
 for (t in (f[i]+1):n.occasions){  
 # State process: draw S(t) given S(t-1)  
 z[i,t] ~ dcat(ps[z[i,t-1], i, t-1,])  
 # Observation process: draw O(t) given S(t)  
 y[i,t] ~ dcat(po[z[i,t], i, t-1,])  
 } #t  
 } #i  
}  
",fill = TRUE)  
sink()  
}

Function to create known latent states z

known.state.ms <- function(ms, notseen){  
 # notseen: label for ënot seení  
 state <- ms  
 state[state==notseen] <- NA  
 for (i in 1:dim(ms)[1]){  
 m <- min(which(!is.na(state[i,])))  
 state[i,m] <- NA  
 }  
 return(state)  
 }

Function to create initial values for unknown z

ms.init.z <- function(ch, f){  
 for (i in 1:dim(ch)[1]){ch[i,1:f[i]] <- NA}  
 states <- max(ch, na.rm = TRUE)  
 known.states <- 1:(states-1)  
 v <- which(ch==states)  
 ch[-v] <- NA  
 ch[v] <- sample(known.states, length(v), replace = TRUE)  
 return(ch)  
 }

Create a list of data

jags.data <- list(y = his\_recoded, f = f, n.occasions = dim(his\_recoded)[2], nind = dim(his\_recoded)[1], z = known.state.ms(his\_recoded, 3))

Initial values

inits <- function(){list(phiA = runif(1, 0, 1), psiAB = runif(1, 0, 1), pA = runif(1, 0, 1), z = ms.init.z(his\_recoded, f))}

Parameters to be monitored

parameters <- c("phiA","phiB", "psiAB", "psiBA", "pA", "pB")

MCMC settings

ni <- 5000 # nb iter  
nt <- 6 # thinning  
nb <- 1000 # burn-in  
nc <- 2 # nb chains

Call JAGS from R

library(R2jags)

## Loading required package: rjags

## Loading required package: coda

## Linked to JAGS 4.2.0

## Loaded modules: basemod,bugs

##   
## Attaching package: 'R2jags'

## The following object is masked from 'package:coda':  
##   
## traceplot

ms <- jags(jags.data, inits, parameters, "state\_on\_survival.jags", n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb)

## module glm loaded

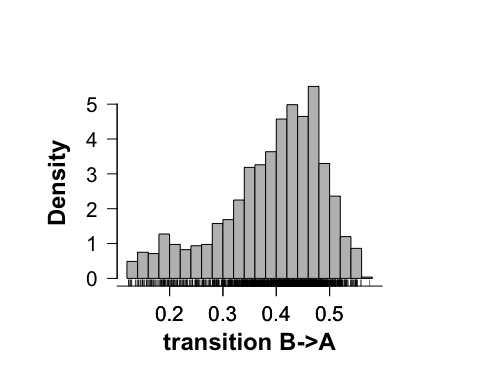
## Compiling model graph  
## Resolving undeclared variables  
## Allocating nodes  
## Graph information:  
## Observed stochastic nodes: 1139  
## Unobserved stochastic nodes: 769  
## Total graph size: 27933  
##   
## Initializing model

print(ms, digits = 3)

## Inference for Bugs model at "state\_on\_survival.jags", fit using jags,  
## 2 chains, each with 5000 iterations (first 1000 discarded), n.thin = 6  
## n.sims = 1334 iterations saved  
## mu.vect sd.vect 2.5% 25% 50% 75% 97.5% Rhat  
## pA 0.335 0.139 0.202 0.256 0.293 0.353 0.779 1.019  
## pB 0.842 0.126 0.545 0.765 0.872 0.944 0.995 1.026  
## phiA 0.769 0.051 0.657 0.741 0.773 0.805 0.853 1.002  
## phiB 0.856 0.040 0.783 0.829 0.854 0.882 0.939 1.005  
## psiAB 0.389 0.099 0.260 0.327 0.364 0.425 0.671 1.017  
## psiBA 0.391 0.096 0.161 0.340 0.412 0.462 0.529 1.025  
## deviance 649.793 43.561 567.810 616.549 650.741 684.022 724.738 1.028  
## n.eff  
## pA 82  
## pB 62  
## phiA 1300  
## phiB 390  
## psiAB 96  
## psiBA 64  
## deviance 60  
##   
## For each parameter, n.eff is a crude measure of effective sample size,  
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).  
##   
## DIC info (using the rule, pD = var(deviance)/2)  
## pD = 933.6 and DIC = 1583.4  
## DIC is an estimate of expected predictive error (lower deviance is better).

Plot of the transition to illustrate local minimum

par(cex.main = 1.5, mar = c(5, 6, 4, 5) + 0.1, mgp = c(3.5, 1, 0), cex.lab = 1.5 , font.lab = 2, cex.axis = 1.3, bty = "n", las = 1)  
h <- hist(ms$BUGSoutput$sims.matrix[,7],breaks=20,freq=F,main="",xlab="",ylab="",col="gray")  
axis(1, seq(0, .60, by = .1))  
axis(2, labels = FALSE, lwd.ticks = 0)  
rug(jitter(ms$BUGSoutput$sims.matrix[,7]))  
mtext("transition B->A", side = 1, line = 2.5, cex = 1.5, font = 2)  
mtext("Density", side = 2, line = 2.5, cex = 1.5, font = 2, las = 0)



## Bayesian analysis with Stan

# https://github.com/stan-dev/example-models/tree/master/BPA  
library(rstan)

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.2

## Loading required package: StanHeaders

## Warning: package 'StanHeaders' was built under R version 3.3.2

## rstan (Version 2.12.1, packaged: 2016-09-11 13:07:50 UTC, GitRev: 85f7a56811da)

## For execution on a local, multicore CPU with excess RAM we recommend calling  
## rstan\_options(auto\_write = TRUE)  
## options(mc.cores = parallel::detectCores())

##   
## Attaching package: 'rstan'

## The following object is masked from 'package:R2jags':  
##   
## traceplot

## The following object is masked from 'package:coda':  
##   
## traceplot

rstan\_options(auto\_write = TRUE)  
options(mc.cores = parallel::detectCores())  
set.seed(123)

Data

dat = list(y=his\_recoded,n\_occasions=k,nind=n)

Model

{  
sink("mstate.stan")  
cat("  
// -------------------------------------------------  
// States (S):  
// 1 alive at A  
// 2 alive at B  
// 3 dead  
// Observations (O):  
// 1 seen at A  
// 2 seen at B  
// 3 not seen  
// -------------------------------------------------  
  
functions {  
 /\*\*  
 \* Return an integer value denoting occasion of first capture.  
 \* This function is derived from Stan Modeling Language  
 \* User's Guide and Reference Manual.  
 \*  
 \* @param y Observed values  
 \* @return Occasion of first capture  
 \*/  
 int first\_capture(int[] y\_i) {  
 for (k in 1:size(y\_i))  
 if (y\_i[k] != 3)  
 return k;  
 return 0;  
 }  
}  
  
data {  
 int<lower=0> nind;  
 int<lower=0> n\_occasions;  
 int<lower=1,upper=3> y[nind, n\_occasions];  
}  
  
transformed data {  
 int<lower=0> n\_occ\_minus\_1;  
 int<lower=0,upper=n\_occasions> first[nind];  
  
 n\_occ\_minus\_1 = n\_occasions - 1;  
 for (i in 1:nind)  
 first[i] = first\_capture(y[i]);  
}  
  
parameters {  
 real<lower=0,upper=1> mean\_phi[2]; // Mean state-spec. survival  
 real<lower=0,upper=1> mean\_psi[2]; // Mean transitions  
 real<lower=0,upper=1> mean\_p[2]; // Mean state-spec. recapture  
}  
  
transformed parameters {  
 vector<lower=0,upper=1>[n\_occ\_minus\_1] phiA; // Survival prob. at site A  
 vector<lower=0,upper=1>[n\_occ\_minus\_1] phiB; // Survival prob. at site B  
 vector<lower=0,upper=1>[n\_occ\_minus\_1] psiAB; // Movement prob. from site A to site B  
 vector<lower=0,upper=1>[n\_occ\_minus\_1] psiBA; // Movement prob. from site B to site A  
 vector<lower=0,upper=1>[n\_occ\_minus\_1] pA; // Recapture prob. at site A  
 vector<lower=0,upper=1>[n\_occ\_minus\_1] pB; // Recapture prob. at site B  
 simplex[3] ps[3, nind, n\_occ\_minus\_1];  
 simplex[3] po[3, nind, n\_occ\_minus\_1];  
  
 // Constraints  
 for (t in 1:n\_occ\_minus\_1) {  
 phiA[t] = mean\_phi[1];  
 phiB[t] = mean\_phi[2];  
 psiAB[t] = mean\_psi[1];  
 psiBA[t] = mean\_psi[2];  
 pA[t] = mean\_p[1];  
 pB[t] = mean\_p[2];  
 }  
  
 // Define state-transition and observation matrices  
 for (i in 1:nind) {  
 // Define probabilities of state S(t+1) given S(t)  
 for (t in 1:(n\_occasions - 1)) {  
 ps[1, i, t, 1] = phiA[t] \* (1.0 - psiAB[t]);  
 ps[1, i, t, 2] = phiA[t] \* psiAB[t];  
 ps[1, i, t, 3] = 1.0 - phiA[t];  
 ps[2, i, t, 1] = phiB[t] \* psiBA[t];  
 ps[2, i, t, 2] = phiB[t] \* (1 - psiBA[t]);  
 ps[2, i, t, 3] = 1.0 - phiB[t];  
 ps[3, i, t, 1] = 0.0;  
 ps[3, i, t, 2] = 0.0;  
 ps[3, i, t, 3] = 1.0;  
  
 // Define probabilities of O(t) given S(t)  
 po[1, i, t, 1] = pA[t];  
 po[1, i, t, 2] = 0.0;  
 po[1, i, t, 3] = 1.0 - pA[t];  
 po[2, i, t, 1] = 0.0;  
 po[2, i, t, 2] = pB[t];  
 po[2, i, t, 3] = 1.0 - pB[t];  
 po[3, i, t, 1] = 0.0;  
 po[3, i, t, 2] = 0.0;  
 po[3, i, t, 3] = 1.0;  
 }  
 }  
}  
  
model {  
 real acc[3];  
 vector[3] gamma[n\_occasions];  
  
 // Priors  
 // Uniform priors are implicitly defined.  
 // mean\_phi ~ uniform(0, 1);  
 // mean\_psi ~ uniform(0, 1);  
 // mean\_p ~ uniform(0, 1);  
  
 // Likelihood  
 // Forward algorithm derived from Stan Modeling Language  
 // User's Guide and Reference Manual  
 for (i in 1:nind) {  
 if (first[i] > 0) {  
 for (k in 1:3)  
 gamma[first[i], k] = (k == y[i, first[i]]);  
  
 for (t in (first[i] + 1):n\_occasions) {  
 for (k in 1:3) {  
 for (j in 1:3)  
 acc[j] = gamma[t - 1, j] \* ps[j, i, t - 1, k]  
 \* po[k, i, t - 1, y[i, t]];  
 gamma[t, k] = sum(acc);  
 }  
 }  
 target += log(sum(gamma[n\_occasions]));  
 }  
 }  
}  
",fill = TRUE)  
sink()  
}

Initial values

inits <- function() list(mean\_phi = runif(2, 0, 1),  
 mean\_psi = runif(2, 0, 1),  
 mean\_p = runif(2, 0, 1))

Parameters monitored

params <- c("mean\_phi", "mean\_psi", "mean\_p")

MCMC settings

ni <- 5000  
nt <- 6  
nb <- 1000  
nc <- 2

Call Stan from R

ms <- stan("mstate.stan",  
 data = dat, init = inits, pars = params,  
 chains = nc, iter = ni, warmup = nb, thin = nt,  
 seed = 1,  
 open\_progress = FALSE)

## Warning: There were 31 divergent transitions after warmup. Increasing adapt\_delta above 0.8 may help. See  
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

## Warning: Examine the pairs() plot to diagnose sampling problems

print(ms, digits = 3)

## Inference for Stan model: mstate.  
## 2 chains, each with iter=5000; warmup=1000; thin=6;   
## post-warmup draws per chain=667, total post-warmup draws=1334.  
##   
## mean se\_mean sd 2.5% 25% 50% 75%  
## mean\_phi[1] 0.756 0.016 0.059 0.618 0.723 0.763 0.798  
## mean\_phi[2] 0.852 0.002 0.037 0.788 0.828 0.850 0.873  
## mean\_psi[1] 0.437 0.045 0.129 0.270 0.343 0.398 0.509  
## mean\_psi[2] 0.341 0.039 0.114 0.140 0.252 0.357 0.431  
## mean\_p[1] 0.415 0.083 0.204 0.211 0.278 0.331 0.476  
## mean\_p[2] 0.777 0.046 0.144 0.523 0.656 0.794 0.898  
## lp\_\_ -438.631 0.332 1.920 -443.493 -439.662 -438.203 -437.213  
## 97.5% n\_eff Rhat  
## mean\_phi[1] 0.849 14 1.091  
## mean\_phi[2] 0.935 538 1.018  
## mean\_psi[1] 0.739 8 1.277  
## mean\_psi[2] 0.529 9 1.206  
## mean\_p[1] 0.980 6 1.386  
## mean\_p[2] 0.991 10 1.186  
## lp\_\_ -435.983 33 1.048  
##   
## Samples were drawn using NUTS(diag\_e) at Mon May 8 15:42:51 2017.  
## For each parameter, n\_eff is a crude measure of effective sample size,  
## and Rhat is the potential scale reduction factor on split chains (at   
## convergence, Rhat=1).

Plot of the transition to illustrate local minimum

par(cex.main = 1.5, mar = c(5, 6, 4, 5) + 0.1, mgp = c(3.5, 1, 0), cex.lab = 1.5 , font.lab = 2, cex.axis = 1.3, bty = "n", las = 1)  
h <- hist(ms@sim$samples[[1]]$`mean\_psi[2]`,breaks=20,freq=F,main="",xlab="",ylab="",col="gray")  
axis(1, seq(0, .60, by = .1))  
axis(2, labels = FALSE, lwd.ticks = 0)  
rug(jitter(ms@sim$samples[[1]]$`mean\_psi[2]`))  
mtext("transition B->A", side = 1, line = 2.5, cex = 1.5, font = 2)  
mtext("Density", side = 2, line = 2.5, cex = 1.5, font = 2, las = 0)

