Appendix S2. Tutorial to estimate abundance from orthomosaic counts

Ismael V. Brack Denis Valle

October 15, 2024

Table of contents

In this tutorial, we show the data simulation process and the model fitting process for mark-resight data and population counts obtained from orthomosaics of drone-based surveys. We do this in two steps: 1) we use the mark-resight data to estimate nesting, availability, double count, and mark identification probabilities; and 2) we use these probability estimates to model the population counts to estimate the total population size and the entry process.

We provide data simulation so users can explore model identifiability and model performance under different scenarios, but if you are interested only in analyzing data, you can jump directly to 2. Data analysis.

Required Packages

We will need the following packages:

```r
library(extraDistr)
library(ggplot2)
library(nimble)
library(MCMCvis)
```

1. Data simulation

Simulation settings

Let's first define the true values for the parameters and define the simulation settings :

```r
# Total population size
Ntot <- 40000
# Number of occasions
J <- 12
# Entry probabilities for Ntot
b <- rdirichlet(1, rep(1,J))
# Probability of identifying the mark of a marked individual
delta=3/4
# Nesting probability
theta=0.4
# Availability probabilty
phi=0.3
# Probability of a walking individual to be a double count (proportion of doubles)
omega=0.2
# Number of latent states
n.states=3
# Number of marked individuals per occasion
marked <- rep(100,J)
```

Define state-transition and detection matrices for mark-resight data

Now, we have to define the transition probabilities from a given state in time *t* to time *t+1*, and the detection probabilities for each latent state. The rows in the state-transition matrix `z.trans` correspond to the possible transitions from each one of the three state. The rows in the detection matrix `z.obs` correspond to the possible detection states for each one of the true latent states.

```r
# Get the first occasion for each individual
mark.occ <- rep(1:J, marked)

# State transition probabilities
z.trans <- matrix(c(
                  1-theta, theta, 0,
                  0,           0, 1,
                  0,           0, 1),
                  nrow=n.states,byrow=T)
```

```r
# State detection probabilities
z.obs <- matrix(c(
  phi,      0,          (1-phi),
  0,        phi,        (1-phi),
  0,        0,             1
),nrow=n.states,byrow=T)
```

Simulate mark-resight data

Next, we can simulate the latent state of each individual at each occasion, and the observation process over these states, using the matrices defined above. This simulation is done starting from the first occasion after marking (we are not simulating entry processes for marked individuals).

```r
# State history matrix
z <- matrix(NA, ncol=J, nrow=sum(marked))
# Observation history matrix
y <- matrix(NA, ncol=J, nrow=sum(marked))

for(i in 1:sum(marked)){
  # first occasion after marking
  # define true state
  z[i,mark.occ[i]] <- rcat(1,z.trans[1,])

  # define observation
  det.prob <- z.obs[z[i,mark.occ[i]],]
  y[i,mark.occ[i]] <- rcat(1,det.prob)

  # if it was marked in the last occasion
  if(mark.occ[i]==J) next;

  # for subsequent occasions
  for(t in (mark.occ[i]+1):J){
    # define true state
    probs <- z.trans[z[i,t-1],]
    z[i,t] <- rcat(1,probs)
    # define observation
    det.prob <- z.obs[z[i,t],]
    y[i,t] <- rcat(1,det.prob)
  } #t
} #i
```

However, note that some marked individuals that are detected (states 1 and 2) can have their marks unidentifiable. Thus, we randomly exclude a proportion of these detections per occasion, based on the identification probability `delta`:

```r
#* excluding some detections given by unidentified markers
y2 <- y
m.ids <- matrix(NA, nrow=J, ncol=2, dimnames=list(1:J,c("identified","unidentified")))
```

```
for(j in 1:J){
  # which individuals were detected
  dets <- which(y[,j]==1 | y[,j]==2)
  # sample some to be unidentified
  unids <- rbinom(1,length(dets),1-delta)

  m.ids[j,"unidentified"] <- unids # number of unidentified marked individuals
  m.ids[j,"identified"] <- length(dets) - unids # number of identified marked inds.

  # sample unidentified individuals and replace by a non-detection
  y2[sample(dets,unids),j] <- 3
}

# See number of individuals in each state per occasion
apply(y, 2, table)
```

```
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
1   25   20   28   29   26   56   44   57   41    47    47    48
2    6   23   20   28   29   27   30   23   27    28    25    27
3   69  157  252  343  445  517  626  720  832   925  1028  1125
```

Moreover, the marked individuals also provide information about the double counting. Then, we simulate the number of times a marked walking individual appears in the orthomosaic in each occasion. Note that, because we defined the double count parameter as being a probability `omega`, we have to calculate the expected number of double counts to simulate it.

```
#** Double counts of walking individuals
# Number of true walking marked individuals
m.truewalk <- apply(y2==1,2,sum,na.rm=T)

# number of double detections of marked individuals walking
# m.truewalk =  m.walk*(1-omega)
# m.double   =  m.walk*omega
# Thus: m.double*m.walk*(1-omega) = m.truewalk*m.walk*omega
#  E(m.double) = m.truewalk*omega / (1-omega)
m.double <- rpois(J,omega*m.truewalk/(1-omega))

# total number of detections of marked individuals walking
m.walk <- m.truewalk + m.double

cbind(m.walk, m.truewalk, m.double)
```

```
     m.walk m.truewalk m.double
 [1,]     21         19        2
 [2,]     18         15        3
 [3,]     28         20        8
 [4,]     30         23        7
 [5,]     26         23        3
```

```
 [6,]     53         44        9
 [7,]     46         32       14
 [8,]     42         32       10
 [9,]     29         25        4
[10,]     46         38        8
[11,]     40         34        6
[12,]     40         35        5
```

Simulate overall population counts

We first simulate the number of entrant individuals (`B`) per occasion based on the total population size and the entry probabilities:

```r
# number of entrant individuals per t
B <- rmultinom(1,Ntot,b)
```

Then, we simulate the dynamics in the population throughout the occasions and the detection, availability, and double counting processes:

```r
# Create the empty vectors to receive simulated data
Nt <- # population size at each occasion
Nt.nest <- # Number of nesting individuals
Nt.walk <- # Number of walkig individuals
Ct.nest <- # Number of available and detected nesting individuals
Ct.truewalk <- # True number of unique walking individuals available
Ct.double <- # Number of double counts
Ct.walk <- # Number of detected walking individuals
as.numeric(J)

# We define the first occasion and then simulate the dynamics
for(t in 1:J){
  # First occasion
  if(t==1){Nt[1] <- B[1]}
  # Subsequent occasions
  if(t>1){Nt[t] <- Nt.walk[t-1] + B[t]}

  # True number of nesting and walking individuals
  Nt.nest[t] <- rbinom(1,Nt[t],theta)
  Nt.walk[t] <- Nt[t] - Nt.nest[t]

  # Number of nesting and walking individuals available at the beach
  Ct.nest[t] <- rbinom(1,Nt.nest[t],phi)
  Ct.truewalk[t] <- rbinom(1,Nt.walk[t],phi)

  # Number of double counts
  Ct.double[t] <- rpois(1,omega*Ct.truewalk[t]/(1-omega))

  # Total observerd counts of walking individuals
  Ct.walk[t] <- Ct.truewalk[t] + Ct.double[t]
```

```
} #t

cbind(N=Nt,N.nest=Nt.nest,N.walk=Nt.walk,C.nest=Ct.nest,C.walk=Ct.walk)
```

```
          N N.nest N.walk C.nest C.walk
 [1,]  9917   3929   5988   1203   2250
 [2,] 10361   4127   6234   1223   2344
 [3,]  8236   3199   5037    926   1913
 [4,]  5814   2329   3485    691   1275
 [5,]  3628   1440   2188    415    811
 [6,]  4096   1679   2417    531    919
 [7,]  5481   2194   3287    643   1209
 [8,]  5379   2112   3267    615   1199
 [9,]  4496   1829   2667    555   1069
[10,]  2898   1175   1723    376    648
[11,] 10491   4172   6319   1242   2405
[12,] 11815   4704   7111   1368   2675
```

Let's export the created objects containing only the observed data to be used in the model fitting:

```
# Observed encounter histories
write.csv(y2, "encounter_history.csv",row.names=F)
# First occasion after marking
write.csv(data.frame(id=1:sum(marked), fo=mark.occ),
          "marking_occasion.csv", row.names=F)
# Counts of marked individuals walking (double counts and uniques)
write.csv(data.frame(m.walk=m.walk, m.detwalk=(m.walk+m.double)),
          "double_counts.csv", row.names=F)
# Counts of marked individuals with marks identified and unidentified
write.csv(m.ids,"mark_identification.csv", row.names=F)

# Overall population counts
write.csv(cbind(Ct.walk, Ct.nest), "population_counts.csv", row.names=F)
```

2. Data analysis

Import and Arrange data

We will need 4 objects to fit the mark-resight model and one object to fit the population counts model, besides the estimates from the mark-resight step.

```
mark.occ <- read.csv("marking_occasion.csv") # first/marking occasion
Y <- read.csv("marking_occasion.csv") # encounter history
m.walks <- read.csv("double_counts.csv") # marked inds. walking and duoble counts
m.ids <- read.csv("double_counts.csv") # marks identified or unidentified
counts <- read.csv("population_counts.csv") # overall population counts
```

For the mark-resight model, we will need:

- `mark.occ`: The occasion in which each individual was marked (we show in the table a random sample of 10 individuals with their corresponding first occasion):

id	fo
427	5
1038	11
552	6
544	6
762	8
591	6
400	4
517	6
825	9
82	1

- `Y`: The encounter history of each detected individual, considering the 3 states: state 1 = detected walking; state 2 = detected nesting; and state 3 = not detected. In the table below, we show the encounter history for the same 10 randomly selected individuals throughout the 12 occasions:

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
NA	NA	NA	NA	3	3	3	3	3	3	3	3
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	1	3
NA	NA	NA	NA	NA	3	3	3	3	3	3	3
NA	NA	NA	NA	NA	3	3	2	3	3	3	3
NA	NA	NA	NA	NA	NA	NA	3	3	3	3	3
NA	NA	NA	NA	NA	2	3	3	3	3	3	3
NA	NA	NA	3	3	3	3	3	3	3	3	3
NA	NA	NA	NA	NA	1	3	3	2	3	3	3
NA	NA	NA	NA	NA	NA	NA	NA	1	3	3	3
1	3	3	3	3	3	3	3	3	3	3	3

  Note that, since we do not use any information previous marking, encounter histories are filled with `NA` before the first occasion of each individual.

- `m.walks`: The compiled number of unique marked individuals that were detected as walking in each occasion and the number of appearances in the mosaic for these individuals. Note that the number of double counts will be the number of appearances subtracted by the number of unique individuals detected `m.double = m.detwalk - m.walk`. See the numbers for the first 5 occasions:

m.walk	m.detwalk	m.double
17	20	3
33	40	7
40	47	7
36	41	5
39	44	5
35	42	7

m.walk	m.detwalk	m.double

- `m.ids` : The number of marked individuals detected in each occasion that have their marks identified or unidentified. See the numbers for the first 5 occasions:

identified	unidentified
26	3
37	4
50	10
49	19
62	18
55	13

  It is important to check if the number of individuals with the marks identified at a given occasion is equal to the number of individuals detected in states 1 or 2 in the encounter history matrix.

For the population counts model, we will use the overall counts `counts`, separated in walking and nesting individuals. See the overall counts for the first 5 occasions:

Ct.walk	Ct.nest
1276	701
1053	588
751	452
846	465
1096	616
1332	702

Step 1: Mark-resight model

Specify the model for Nimble

This model is a multi-state open-population capture-recapture model that was adapted to include the identification probability and the double counting process. Note that, for the transitions from state 2, we have to define an estimable parameter `pepa` with a strong prior towards one (`dbeta(20,1)`) to be able to fit the model using `nimble`.

```
modMR <- nimbleCode({
  # Priors ------------------------------------------------------------------
  phi ~ dunif(0,1) # availability prob.
  omega ~ dunif(0,1) # prop. of repeated counts
  delta ~ dunif(0,1) # prop. of unidentified marked individuals
  theta ~ dunif(0,1)# Nesting prob.
  pepa ~ dbeta(20,1) # additional parameter to be able to estimate the model

  # z transition matrix -----------------------------------------------------
  # z[t-1]=1
```

```
psi[1,1] <- 1-theta
psi[1,2] <- theta
psi[1,3] <- 0
# z[t-1]=2
psi[2,1] <- 0
psi[2,2] <- 1-pepa
psi[2,3] <- pepa
# z[t-1]=3
psi[3,1] <- 0
psi[3,2] <- 0
psi[3,3] <- 1

# y observation matrix -------------------------------------------------------
p[1,1] <- phi*delta
p[1,2] <- 0
p[1,3] <- (1-phi) + phi*(1-delta)
# z=2
p[2,1] <- 0
p[2,2] <- phi*delta
p[2,3] <- (1-phi) + phi*(1-delta)
# z=3
p[3,1] <- 0
p[3,2] <- 0
p[3,3] <- 1

# Mark-resight likelihood ---------------------------------------------------

# Multi-state CJS for theta and phi
for(i in 1:M){
  #* First occasion (fo) after marking
  #z is the latent state
  z[i,fo[i]] ~ dcat(psi[1,1:3])
  #y is the observation
  y[i,fo[i]] ~ dcat(p[z[i,fo[i]] ,1:3])

  #Subsequent occasions
  for(t in (fo[i]+1):J){
    # latent state transition
    z[i,t] ~ dcat(psi[z[i,t-1], 1:3])
    # observation
    y[i,t] ~ dcat(p[z[i,t], 1:3])

  } # t
} # i
# Compiled count data
for(t in 1:J){
  # Number of repeated detections
  m.double[t] ~ dbin(omega, m.walk[t])
```

```
    # Number of unidentified individuals
    m.unids[t] ~ dbin((1-delta), m.detect[t])
  } # t
}) # model
```

Organize mark-resight data for Nimble

In the code below, we bundle all the data required by `nimble`:

```
dat1 <- list(
  y=Y,
  m.unids=m.ids[,"unidentified"],
  m.detect=(m.ids[,"identified"] + m.ids[,"unidentified"]),
  m.walk=m.walks[,"m.walk"],
  m.double=m.walks[,"m.detwalk"] - m.walks[,"m.walk"],
  J=nrow(m.ids),
  M=nrow(mark.occ),
  fo=mark.occ[,"fo"]
)
```

Note that, since the identification process is estimated as a proportion of identified marks out of all the marked individuals detected, we have to specify `m.detect` as the sum of marks identified and unidentified.

Initial values for the MCMC algorithm:

It can be tricky to provide initial values for latent states. Here, we define all the non-detections that occurr between detections as latent state 1. If the individual was not detected nesting, we also include this latent state as initial value.

```
z.in <- matrix(NA,nrow=nrow(Y),ncol=J) # empty array for true states initial values
fo=mark.occ[,"fo"] # first occasion (fo) after marking

for(i in 1:nrow(z.in)){
  foi <- fo[i]
  noi <- length(fo[i]:J)

  # If all detections are 3, put a 2 in the first occasion
  if(sum(dat1$y[i,]==3,na.rm=T)==noi){
    z.in[i,foi:J] <- c(2,rep(3,noi-1))
  }
  # If there is a detection in state 2
  if(any(dat1$y[i,]==2,na.rm=T)){
    t2 <- which(dat1$y[i,]==2) # get position of the 2
    if(t2>foi){z.in[i,foi:(t2-1)] <- 1} # fill with 1 until the 2
    z.in[i,t2] <- 2 # 2
    if(t2<J){z.in[i,(t2+1):J] <- 3} # fill with 3 after the 2
  }
  # If there are detections in state 1 but not in 2
```

```r
  if(any(dat1$y[i,]==1,na.rm=T) & !any(dat1$y[i,]==2,na.rm=T)){
    t1 <- max(which(dat1$y[i,]==1)) # get position of the last 1
    z.in[i,foi:t1] <- 1
    if(t1<J){z.in[i,t1+1] <- 2} # fill with 2 if there is a 3 after the last 1
    if(t1<(J-1)){z.in[i,(t1+2):J] <- 3} # fill with 3 if we filled with a 2
  }

}
inits1 <- function() list(
  phi=runif(1),
  theta=runif(1),
  pepa=runif(1,.9,1),
  delta=runif(1),
  omega=runif(1),
  z=z.in
)
```

Next, we define the parameters that we want to be monitored in MCMC and define the MCMC settings.

```r
# Parameters monitored
params <- c("phi","theta","pepa","omega","delta")

# MCMC settings
ni <- 60000 # number of iterations
nt <- 1 # thinning rate
nb <- 20000 # burn-in
nc <- 3 # number of chains
```

Fit mark-resight model

Finally, we run the mark-resight model! (Note that this can take a few hours to run)

```r
# Run Nimble
out1 <- nimbleMCMC(
  code=modMR,
  constants=dat1,
  inits=inits1,
  monitors=params,
  niter = ni,
  nburnin = nb,
  nchains = nc,
  summary=F
)
```

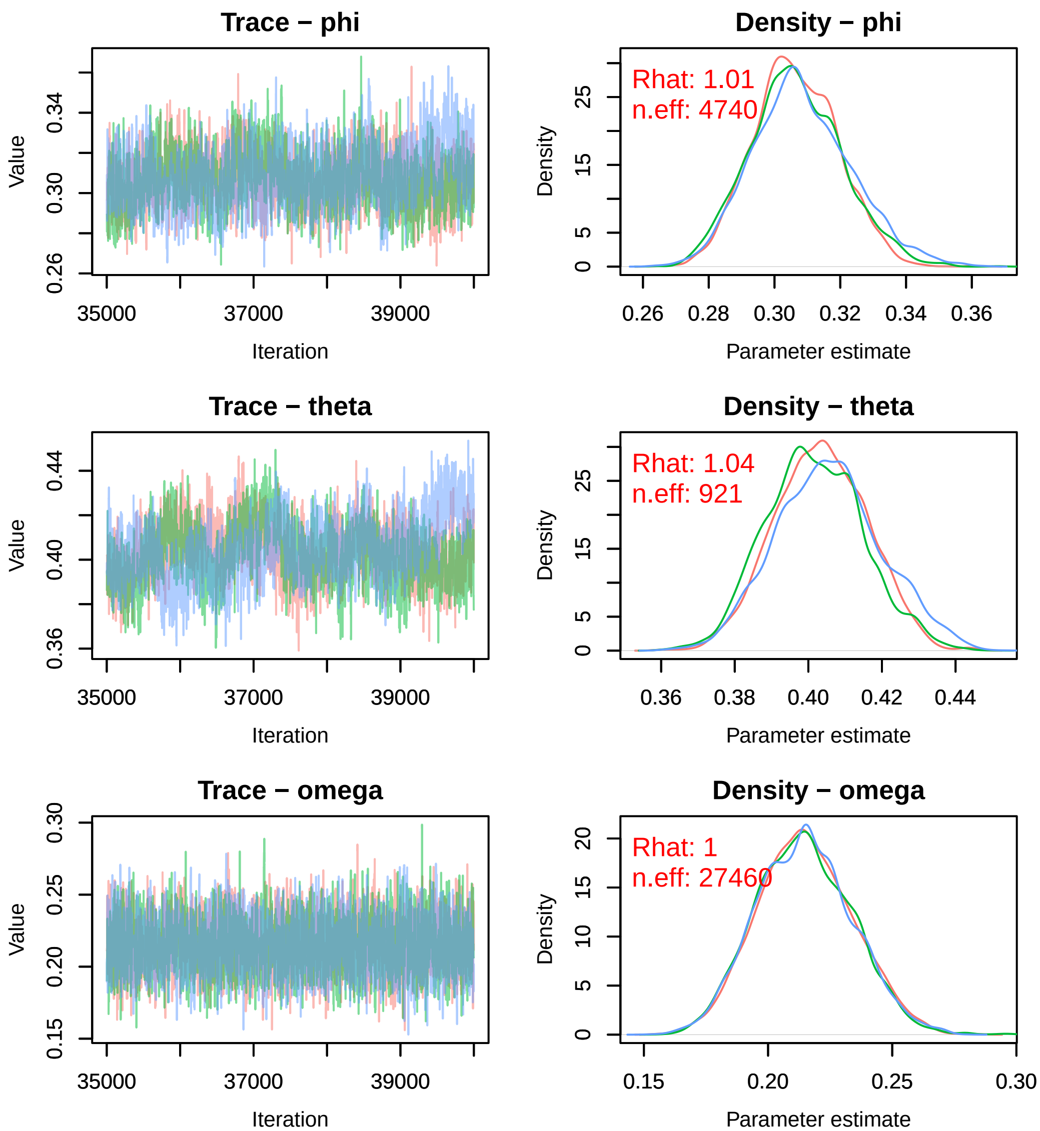
We summarize the posterior samples from the `nimble` output to examine the model results:

```r
resu1 <- MCMCsummary(out1)
print(resu1)
```

```
        mean         sd      2.5%       50%     97.5% Rhat n.eff
delta 0.7467320 0.01528961 0.7162787 0.7469690 0.7759447 1.00 16753
omega 0.2144841 0.01916410 0.1780352 0.2141102 0.2530085 1.00 27460
pepa  0.9861231 0.01301397 0.9521197 0.9899141 0.9996202 1.01  1443
phi   0.3079311 0.01381415 0.2816017 0.3075937 0.3356970 1.01  4740
theta 0.4048747 0.01331280 0.3790971 0.4048108 0.4310706 1.04   921
```

We can also see the traceplots and posterior distributions for the monitored parameters. Note that each chain is represented by one color.

```
# traceplots..
MCMCtrace(out1,ind=T,pdf=F,
          Rhat=T,n.eff=T,
          params=c("phi","theta","omega"))
```



Results look good! Estimated values were close to the true values used to simulate the data: `phi=0.3; theta=0.4; omega=0.2`. It is important to check MCMC convergence in the results (e.g., `Rhat<1.1`). Note that the posterior samples for `theta` are a bit auto-correlated (proportionally low number of effective sample size). Thus, longer chains may be necessary. For a better performance, one can try changing the model specification using a marginalization approach to avoid the estimation of latent states.

Step 2: Population counts model

Now, we will use the parameters estimated from the mark-resight data to estimate: the entries probabilities and the total population size. For simplicity here, we will use only the mean estimated values from the previous step. However, for a more robust approach, to fully incorporate the uncertainty from the mark-resight model, a better alternative is to use random samples from the posterior distribution, fit the model below for each set of samples, and then combine the results.

Write model for Nimble

We specify the Nimble model using the code below. Note that MCMC algorithms usually cannot handle a random variable (in our case `Ntot`) for the multinomial distribution. Thus, to represent the multinomial entries, we have to implement a stick-breaking approach using a series of binomials with conditional entry probabilities.

```
modCounts <- nimbleCode({
  # Priors ------------------------------------------------------------------
  # Entry probs.
  b[1:J] ~ ddirch(b.pri[1:J])

  # Total pop. size
  Ntot2 ~ dunif(0,20)
  Ntot <- round(Ntot2*10000)

  # Population counts -------------------------------------------------------
  # Stick-breaking approach to represent multinomial entries
  # First occasion
  B[1] ~ dbin(b[1],Ntot)
  for(t in 2:(J-1)){
    # remaining Ntot that have not entered the population yet
    r.Ntot[t] <- Ntot - sum(B[1:(t-1)])
    # conditional entry prob.
    b.cond[t] <- b[t] / (1 - sum(b[1:(t-1)]))

    B[t] ~ dbin(b.cond[t], r.Ntot[t])
  } #t
  # Last occasion
  B[J] <- Ntot - sum(B[1:(J-1)])

  # Latent population dynamics
  N[1] <- B[1]
  for(t in 2:J){
    N[t] <- N.walk[t-1] + B[t]
```

```
  } #t

  for(t in 1:J){
    # Nesting and Walking latent variables
    N.nest[t] ~ dbin(theta,N[t])
    N.walk[t] <- N[t] - N.nest[t]

    # Observation process for counts
    # Availability
    C.nest[t] ~ dbin(phi,N.nest[t])
    C.truewalk[t] ~ dbin(phi, N.walk[t])

    # Count errors
    C.double[t] ~ dbin(omega, C.walk[t])

    # Total counts
    C.tot[t] = C.nest[t] + C.truewalk[t] + C.double[t]

  } # t

}) # model
```

Note that in this model structure, by using a Dirichlet distribution for the entry probabilities (`ddirch(b.pri[1:J])`), we are considering this process to be time-independent (one parameter for each occasion). Alternatively, one could model the entry probabilities using a linear or quadratic time trend (e.g., under a multinomial logit link) or random effects.

Organize count data for Nimble

We start by bundling the data required by `nimble` using the code below. We get the mean estimated values for `phi, theta, omega` from the summarized result of the mark-resight model (step 1).

```
dat2 <- list(
  J=J, # number of occasions
  b.pri=rep(1,J), # prior for entry probs.
  # Parameters from MR
  phi=resu1["phi","mean"],
  theta=resu1["theta","mean"],
  omega=resu1["omega","mean"],
  # Count data
  C.tot=counts[,"Ct.nest"] + counts[,"Ct.walk"],
  C.nest=counts[,"Ct.nest"], C.walk=counts[,"Ct.walk"]
)
```

For the initial values for the population dynamics, we have to assure that the values of the latent counts are consistent among themselves and are consistent with `Ntot`.

```r
Cd.in <- rbinom(J,Ct.walk, 0.1) # double counts
Ctw.in <- Ct.walk - Cd.in # true walking inds.

Nn.in <- Ct.nest+500 # N.nest
Nw.in <- Ctw.in+500 # N.walk
N.in  <- Nn.in + Nw.in # Nt

B.in <- c(N.in[1],N.in[2:J] - Nw.in[1:(J-1)]) # B

inits2 <- function() list(
  C.truewalk=Ctw.in,
  C.double=Cd.in,
  N.nest=Nn.in,
  B=c(B.in[1:(J-1)], NA),
  Ntot2=sum(B.in)/10000

)
```

We now specify which parameters will be monitored and the MCMC settings:

```r
# Parameters to be monitored
params <- c("Ntot","b","N","N.nest","N.walk","B")

# MCMC settings
ni <- 40000; nt <- 2; nb <- 20000; nc <- 3
```

Fit count model

Now, we can fit the population counts model:

```r
# Run Nimble!
  out2 <- nimbleMCMC(
    code=modCounts,
    constants=dat2,
    inits=inits2,
    monitors=params,
    niter = ni,
    nburnin = nb,
    nchains = nc
  )
```

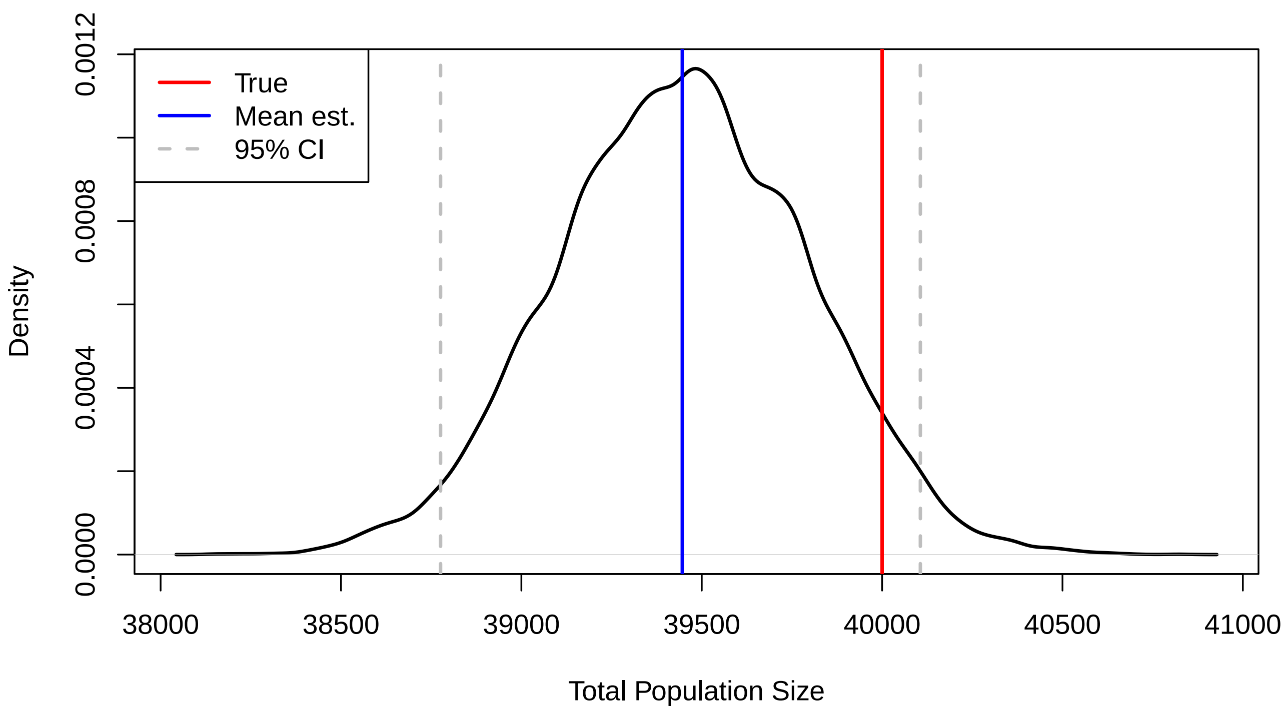
We summarize the posterior samples and see the model results for the total population size with the code below:

```r
resu2 <- MCMCsummary(out2)
print(resu2["Ntot",])
```

```
         mean       sd  2.5%   50% 97.5% Rhat n.eff
Ntot 39446.09 344.2798 38776 39447 40106 1.02   337
```

See the posterior distribution for the total population size and compare to the true value:

```
plot(density(do.call(rbind, out2)[,"Ntot"]),col="black",lwd=2,
     main="",xlab="Total Population Size")
abline(v=resu2["Ntot",c("mean","2.5%","97.5%")],
       col=c("blue","gray","gray"), lty=c(1,2,2), lwd=2)
abline(v=Ntot, col="red",lwd=2)
legend("topleft",c("True","Mean est.","95% CI"),
       col=c("red","blue","gray"),lwd=2,lty=c(1,1,2))
```



The figure above suggests that the model performs well. However, using only the mean estimates from the mark-resight model clearly does not provide a comprehensive estimation of the uncertainty for total population size. To fully accommodate uncertainty, one could run this analysis multiple times using random posterior samples from the mark-resight model. Other option is to run an integrated model, combining the two data sets. However, be aware that in this last case the population counts will also influence ("contaminate") the estimation of the parameters that were being estimated only with the mark-resight data.