Supplementary Materials:

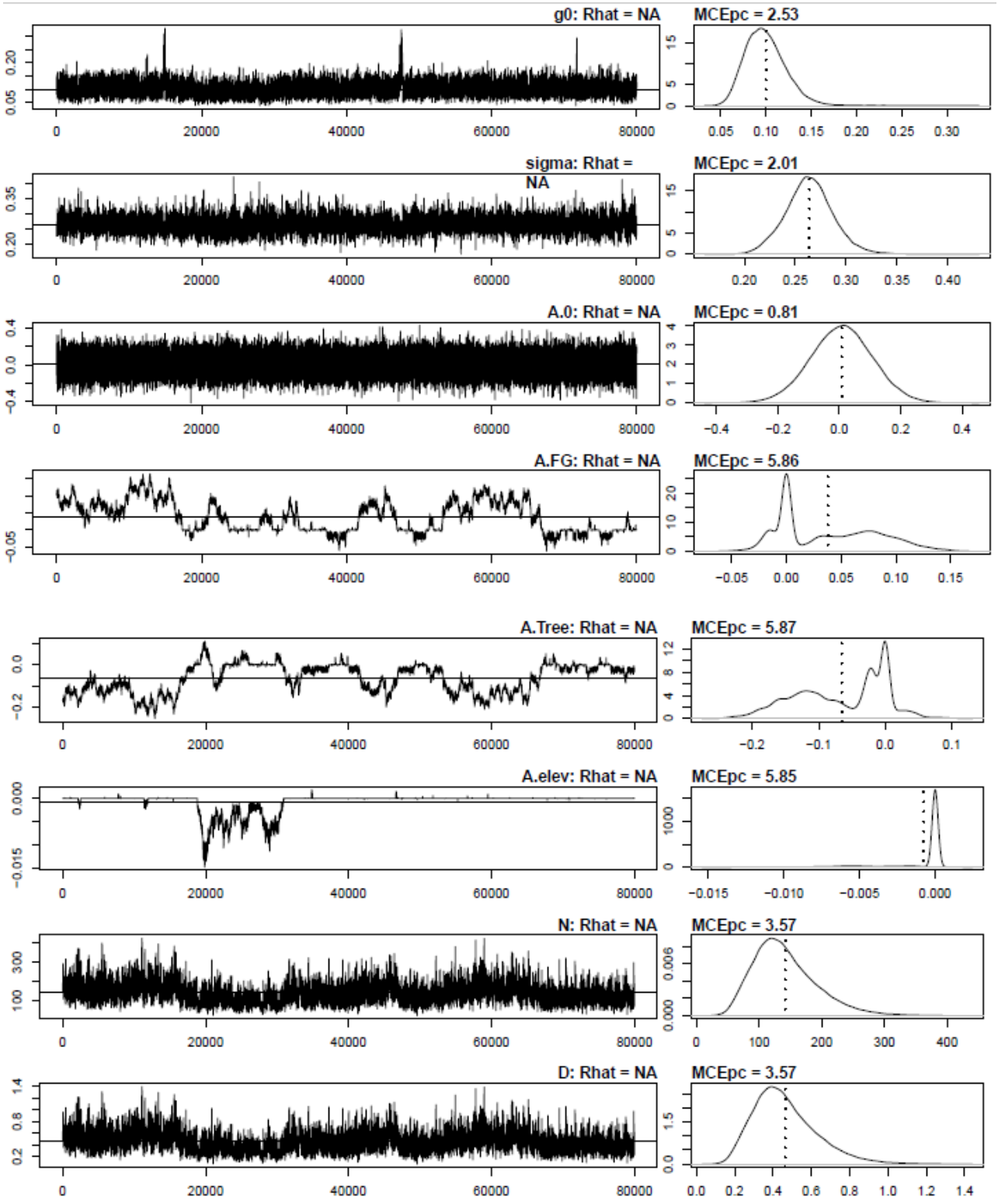


Figure S1. Trace plots and Monte Carlo standard error as a percentage of the posterior SD (MCEpc) values for variables in models of counts of elk independent detections. The trace plots and MCEpc values indicate adequate mixing of most variables but problematic mixing of the environmental covariate variables (chains not clearly converged and MCEpc values over 5).

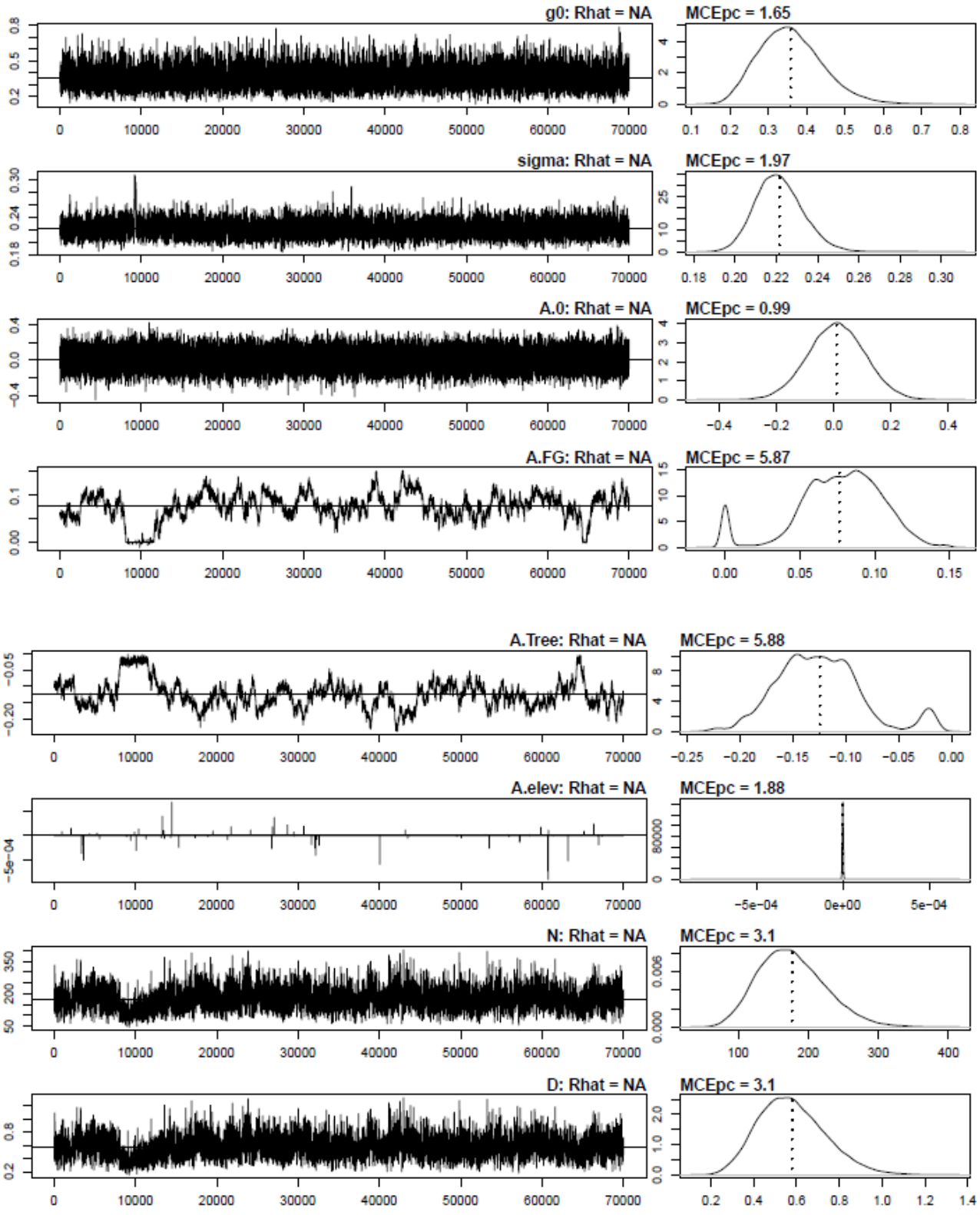


Figure S2. Trace plots for variables in models of total counts of elk individuals. These trace plots and MCEpc values indicate adequate mixing of all variables, though the MCEpc values for the A.FG and A.Tree environmental covariate coefficients are high.

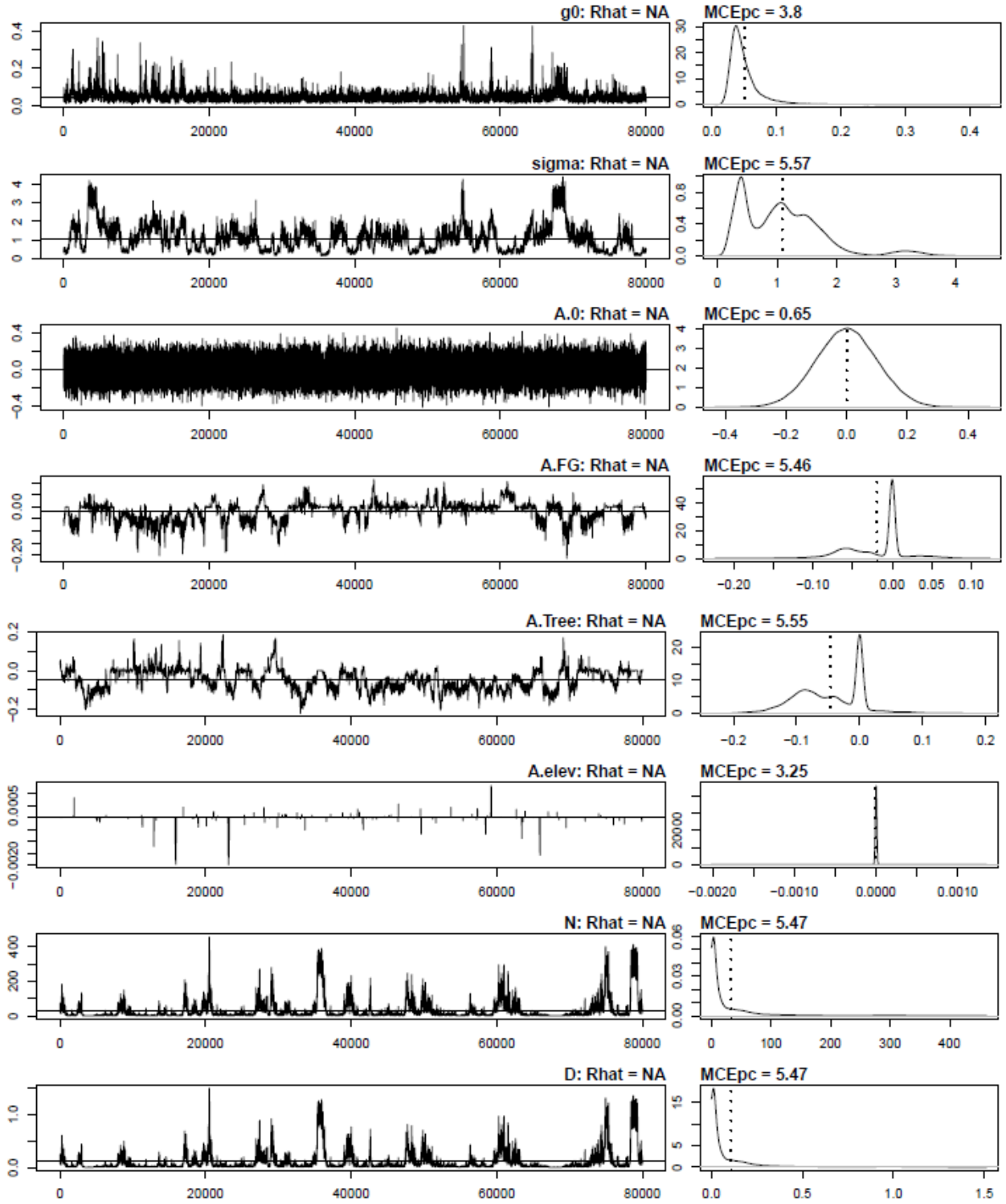


Figure S3. Trace plots for variables in models of presence-absence of elk. Trace plots for variables in models of total counts of elk individuals. These trace plots and MCEpc values indicate inadequate mixing of all variables, except g0.

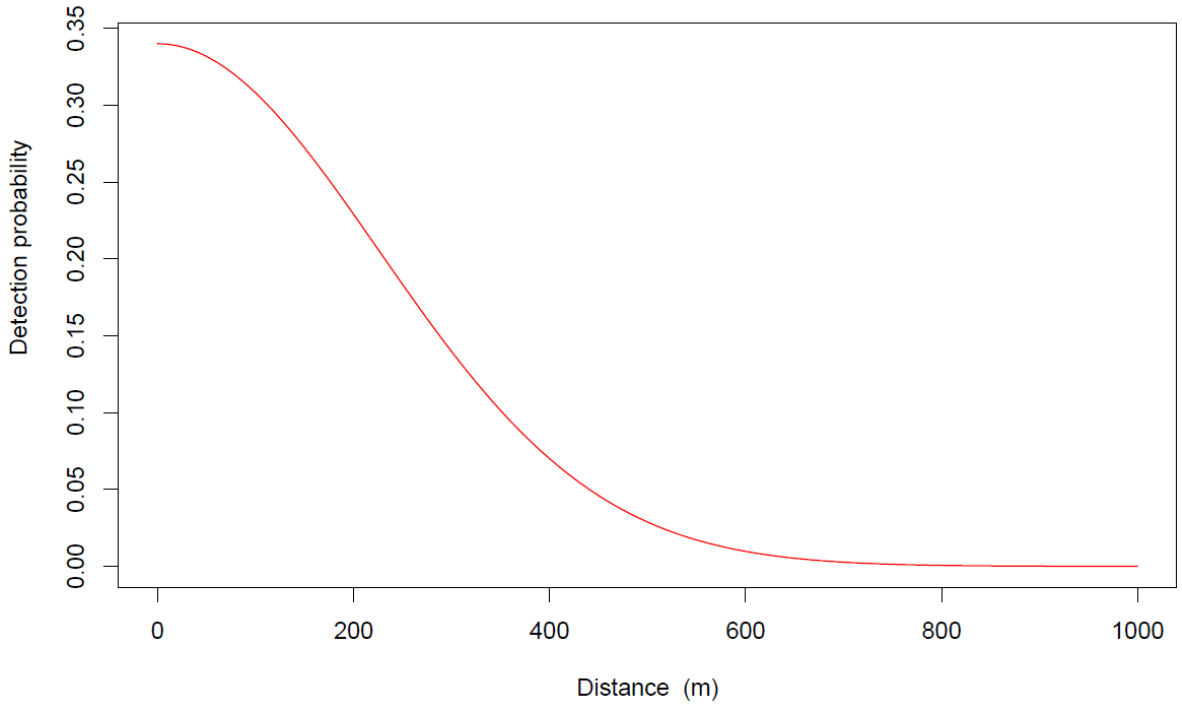


Figure S4. The effects of increasing distance from an individual's activity center on the probability of detecting that individual, parameterized by sigma. This plot represents the relationship estimated by the model fit to the 'total count' elk dataset.

R code for running uSCR models:

```
## unmarked spatial capture recapture (counts, and presence-absence) through nimble

## helpful packages for analyses (but just nimble and mcmcOutput needed to run models and look at output)
library(camtrapR)
library(nimble)
library(sf)
library(mcmcOutput)
library(raster)
library(jagsUI)
library(ggplot2)
library(ggpubr)
library(stringr)
library(secr)



#-------------------------------------------------
#
#   Nimble functions first
#
#-------------------------------------------------

# Binomial distribution for row vectors
dbin_by_row <- nimbleFunction(
   run = function(x = double(1), P = double(1), K = double(1), log = integer(0, default = 0)) {
      J <- length(x)
      ans <- 0.0
      for(j in 1:J)
         ans <- ans + dbinom(x[j], K[j], P[j], 1)
      returnType(double())
      if(log) return(ans)
      else return(exp(ans))
   })

rbin_by_row    <- nimbleFunction(
   run = function(n = integer(), P = double(1), K = double(1)) {
      declare(ans, double(1))
      J <- length(P)
      setSize(ans, J)
      for(j in 1:J)
         ans[j] <- rbinom(1, K[j], P[j])
```

```
    returnType(double(1))
    return(ans)
  })

registerDistributions(list(
  dbin_by_row = list(
    BUGSdist = "dbin_by_row(P, K)",
    Rdist = "dbin_by_row(P, K)",
    range = c(0, Inf),
    types = c('value = double(1)', 'P = double(1)', 'K = double(1)'))
))
#------------------------------
# Poisson distribution

dpois_by_row <- nimbleFunction(
  run = function(x = double(1), bigLambda = double(1), K = double(1), log = integer(0, default = 0)) {
    J <- length(x)
    ans <- 0.0
    for(j in 1:J)
      ans <- ans + dpois(x[j], bigLambda[j]*K[j], 1)
    returnType(double())
    if(log) return(ans)
    else return(exp(ans))
  })

rpois_by_row   <- nimbleFunction(
  run = function(n = integer(), bigLambda = double(1), K = double(1)) {
    declare(ans, double(1))
    J<- length(bigLambda)
    setSize(ans, J)
    for(j in 1:J)
      ans[j] <- rpois(1, bigLambda[j]*K[j])
    returnType(double(1))
    return(ans)
  })

registerDistributions(list(
  dpois_by_row = list(
    BUGSdist = "dpois_by_row(bigLambda, K)",
    Rdist = "dpois_by_row(bigLambda, K)",
    range = c(0, Inf),
    types = c('value = double(1)', 'bigLambda = double(1)', 'K = double(1)'))
))
```

```
## Now uSCR code (count response model first)

code.pois <- nimbleCode({
   ## Priors
   alpha ~ dnorm(0, sd=1)
   A.0 ~ dnorm(0, sd=0.1)
   A.FG ~ dnorm(0, sd=0.1)
   A.elev ~ dnorm(0, sd=0.1)
   A.Tree ~ dnorm(0, sd=0.1)
   logit(g0)<- alpha
   sigma ~ dnorm(1.5,0.5)
   z.FG ~ dbern(psi.ind) ## Indicator variables per covariate
   z.elev ~ dbern(psi.ind)
   z.Tree ~ dbern(psi.ind)

   ## Loop through state space
   for(j in 1:npix) {
      mu[j] <- exp(A.0 + z.FG*A.FG*FG.pix[j] + z.Tree*A.Tree*Tree.pix[j] +
z.elev*A.elev*Elevation.pix[j])*pixArea
      pp[j]<- mu[j]/EN
   }
   EN<- sum(mu[1:npix])
   psi <- EN/M

   for(i in 1:M) {
      w[i] ~ dbern(psi)
      S[i] ~ dcat(pp[1:npix])
      gx[i]<- grid[S[i],1]
      gy[i]<- grid[S[i],2]

      d2[i,1:J]<- (gx[i]-X[1:J,1])^2+(gy[i]-X[1:J,2])^2
      prob[i,1:J]<- g0 * exp(-d2[i,1:J]/2/sigma^2) * w[i]

   }

   for(j in 1:J) {
      Ptrap[j]<- 1-prod(1-prob[1:M,j])
   }
   n[1:J] ~ dpois_by_row(Ptrap[1:J],K[1:J])
```

```
   N<- sum(w[1:M])
   D<- N/A
})
#---------------------------------------------------------------------

## Now presence-absence model

code.binom <- nimbleCode({
   ## Priors
   alpha ~ dnorm(0, sd=1)
   A.0 ~ dnorm(0, sd=0.1)
   A.FG ~ dnorm(0, sd=0.1)
   A.elev ~ dnorm(0, sd=0.1)
   A.Tree ~ dnorm(0, sd=0.1)
   logit(g0)<- alpha
   sigma ~ dnorm(1.5,0.5)
   z.FG ~ dbern(psi.ind) ## Indicator variables per covariate
   z.elev ~ dbern(psi.ind)
   z.Tree ~ dbern(psi.ind)

   ## Loop through state space
   for(j in 1:npix) {
      mu[j] <- exp(A.0 + z.FG*A.FG*FG.pix[j] + z.Tree*A.Tree*Tree.pix[j] +
z.elev*A.elev*Elevation.pix[j])*pixArea
      pp[j]<- mu[j]/EN
   }
   EN<- sum(mu[1:npix])
   psi <- EN/M

   for(i in 1:M) {
      w[i] ~ dbern(psi)
      S[i] ~ dcat(pp[1:npix])
      gx[i]<- grid[S[i],1]
      gy[i]<- grid[S[i],2]

      d2[i,1:J]<- (gx[i]-X[1:J,1])^2+(gy[i]-X[1:J,2])^2
      prob[i,1:J]<- g0 * exp(-d2[i,1:J]/2/sigma^2) * w[i]

   }

   for(j in 1:J) {
      Ptrap[j]<- 1-prod(1-prob[1:M,j])
   }
```

```
   n[1:J] ~ dbin_by_row(Ptrap[1:J],K[1:J])


   N<- sum(w[1:M])
   D<- N/A
})
#-----------------------------------------------------------------------
```