Mapping patterns of occupancy and richness in a community characterized by strong philopatry

Appendix S1: Model Code

The model was run indivudally for each species, using data extracted from the main Antarctic Site Inventory database.

Code from one species (Gentoo Penguin) is displayed here.

Data Preparation:

```
jags.path="/usr/local/bin/jags/"
library('R2jags')
library('coda')
library('gtools')
library('boot')
library('reshape')
library('reshape2') # used for acast function
library('snowfall') # used to run in parallel
library('parallel') # used to run in parallel
library('SpatialEpi') # used to run in parallel

seaice.data <- read.csv("SeaIceNov.csv", header = TRUE)
visit.data <- read.csv("GEPE.visit.csv", header = TRUE)
temp.data <- merge(visit.data, seaice.data, by="Site", all.x=TRUE)

# Taking out any sites for which no sea-ice data exist:
temp.data <- subset(temp.data, is.na(Mean.Nov.Ice)==FALSE)
temp.data <- temp.data[order(temp.data$Site, temp.data$Season),]

# Creating simpler data columns:
temp.data$seaice <- as.numeric(temp.data$Mean.Nov.Ice)
temp.data$state <- as.numeric(temp.data$State+1)

# Labels the number of repeat visits in a year
temp.data$visit <- 1
for (i in 2:dim(temp.data)[1]){
  if (temp.data$Site[i] == temp.data$Site[i-1] &
      temp.data$Season[i] == temp.data$Season[i-1])
    temp.data$visit[i] <- temp.data$visit[i-1]+1
}

# turning variables into numeric for ease in modeling
temp.data$site <- as.numeric(as.factor(temp.data$Site))
temp.data$season <- as.numeric(as.factor(temp.data$Season))
final.data.matrix <- subset(temp.data, select=c(site,season,visit,state))

# Creating sea-ice object
```

```
unique.sea.ice <- unique(subset(temp.data, select=c(site,seaice)))
seaice <- as.numeric(scale(unique.sea.ice$seaice))

V <- max(final.data.matrix$visit) # Highest number of visits
S <- length(unique(final.data.matrix$site)) # Number of sites
Y <- length(unique(final.data.matrix$season)) # Number of Years
y <- acast(final.data.matrix, site ~ season ~ visit) # Data array for model


# For keeping site site numbers clear later:
SiteCodes <- data.frame(Num = c(1:S),
                        OldNum = unique(final.data.matrix$site),
                        Code = temp.data$Site[match(unique(final.data.matrix$site),
                                                    temp.data$site)]
                        )
```

The JAGS model:

```
sink("Occupancy.jags")
cat("

    model {

    #### Priors ####

    beta.psi ~ dnorm(0,0.386)
    beta.r ~ dnorm(0,0.386)
    alpha.psi ~ dnorm(0,0.386)
    alpha.r ~ dnorm(0,0.386)
    gamma.psi ~ dnorm(0,0.386)
    gamma.r ~ dnorm(0,0.386)

    p2 ~ dunif(0,1)
    p3[1:3] ~ ddirch(alpha.p3[1:3])
    for (i in 1:3){
      alpha.p3[i] <- 1
    }

    #### Likelihood ####

    ## states-space model ##

    # First year:
    for (i in 1:S){
      logit(mu.psi[i,1]) <- alpha.psi + beta.psi*seaice[i]
      logit(mu.r[i,1]) <- alpha.r + beta.r*seaice[i]
      z1[i,1] ~ dbin(mu.psi[i,1], 1)
      z2[i,1] ~ dbin(mu.r[i,1]*z1[i,1], 1)
      z[i,1] <- c(1-max(z1[i,1], z2[i,1]),
                  z1[i,1]-z2[i,1],
                  z2[i,1]
                  ) %*% c(1,2,3)
```

```
}

# Subsequent years:
for (i in 1:S){
  for (j in 2:Y){
   logit(mu.psi[i,j]) <- alpha.psi +
                         beta.psi*seaice[i] +
                         gamma.psi*z1[i,j-1]
   logit(mu.r[i,j]) <- alpha.r +
                       beta.r*seaice[i] +
                       gamma.r*z2[i,j-1]
    z1[i,j] ~ dbin(mu.psi[i,j], 1)
    z2[i,j] ~ dbin(mu.r[i,j]*z1[i,j], 1)
    z[i,j] <- c(1-max(z1[i,j], z2[i,j]),
                z1[i,j]-z2[i,j],
                z2[i,j]
                ) %*% c(1,2,3)
  }
}

## observation model ##

# Define observation matrix

p[1,1] <- 1
p[1,2] <- 0
p[1,3] <- 0
p[2,1] <- 1-p2
p[2,2] <- p2
p[2,3] <- 0
p[3,1] <- p3[1]
p[3,2] <- p3[2]
p[3,3] <- p3[3]

for (i in 1:S){
  for (j in 1:Y){
    for (k in 1:V){
      y[i,j,k] ~ dcat(p[z[i,j],])
      y.new[i,j,k] ~ dcat(p[z[i,j],]) # for posterior predictive check
    }
  }
}

#### Derived values ####

## phi = the probability of being in each state
for (i in 1:S) {
  for (j in 1:Y) {
    phi[i,j,1] <- 1 - mu.psi[i,j]
    phi[i,j,2] <- mu.psi[i,j] * (1 - mu.r[i,j])
    phi[i,j,3] <- mu.psi[i,j] * mu.r[i,j]
  }
}
```

```
    ## eval = the probability of recording each state
    for (i in 1:S) {
      for (j in 1:Y) {
        eval[i,j,1] <- (phi[i,j,1] * 1) +
                       (phi[i,j,2] * (1 - p2)) +
                       (phi[i,j,3] * p3[1])
        eval[i,j,2] <- (phi[i,j,1] * 0) +
                       (phi[i,j,2] * p2) +
                       (phi[i,j,3] * p3[2])
        eval[i,j,3] <- (phi[i,j,1] * 0) +
                       (phi[i,j,2] * 0) +
                       (phi[i,j,3] * p3[3])
      }
    }

    }",fill = TRUE)
sink()
```

Running JAGS

```
#####  Bundle the data for JAGS #####

Dat <- list(y = y, S = dim(y)[1], Y = dim(y)[2], V = dim(y)[3], seaice = seaice)



##### Select Initial Values #####

InitStage <- function()
{
  list(
    z1 = matrix(1, nrow = dim(y)[1], ncol = dim(y)[2]),
    z2 = matrix(1, nrow = dim(y)[1], ncol = dim(y)[2]),
    beta.psi = runif(1,0.2,0.2),
    beta.r = runif(1,0.2,0.2),
    alpha.psi = runif(1, 0.2, 0.3),
    alpha.r = runif(1, 0.2, 0.3),
    gamma.psi = 0,
    gamma.r = 0,
    p3 = as.vector(rdirichlet(1,c(1,1,1))),
    p2 = runif(1,.1,.9)
    )
}



##### Posteriors to return #####

# Note: it may be necessary to limit this list to limit object size
# (recommend only returning "y.new" OR "z")
```

```
ParsStage <- c("beta.psi","beta.r",
               "alpha.psi","alpha.r",
               "gamma.psi","gamma.r",
               "p2","p3",
               "mu.psi","mu.r",
               "eval","y.new",
               "z")



##### MCMC settings #####

ni <- 100000 # final iterations
nt <- 50 #thinning rate
nb <- 250000 # burn-in
nc <- 3 # number of chains
n.adapt <- 10000



##### Parallel Execution #####

cl <- makeCluster(3)

clusterExport(cl, c("Dat", "y", "InitStage", "ParsStage", "n.adapt", "nb", "ni", "nt"))

system.time({
  out1 <- clusterEvalQ(cl, {
    library(rjags); library(SpatialEpi); library('gtools')
    jm <- jags.model("Occupancy.jags",
                     data = Dat, InitStage,
                     n.chains=1, n.adapt=n.adapt)
    update(jm, n.iter=nb, thin=nt)
    samples = coda.samples(jm, n.iter=ni, variable.names=ParsStage, thin=nt)
    return(as.mcmc(samples))
  })
})

stopCluster(cl)

zm.GEPE <- mcmc.list(out1)

save(zm.GEPE,file="GEPE.results.Rdata")
```