SUPPLEMENT 1

SSSM WinBUGS 1.4.1 code

```
model;
{
# Switching State-Space Model (SSSM) code
# First difference CRW with stochastic switching between 2 behavioural modes
# Ian D. Jonsen, Ransom A. Myers &  Michael C. James
# IDJ 01/04/05, rev - IDJ 29/05/06

pi <- 3.1416

Omega[1,1] <- 1
Omega[1,2] <- 0
Omega[2,1] <- 0
Omega[2,2] <- 1

# Priors on process uncertainty
iSigma[1:2,1:2] ~ dwish(Omega[,],2)
Sigma[1:2,1:2] <- inverse(iSigma[,])

# Priors for theta's - mean turn angles
tmp[1] ~ dbeta(20, 20)
tmp[2] ~ dbeta(10, 10)

# assume theta when transitting is centered on 0 rad
theta[1] <- (2 * tmp[1] - 1) * pi

# assume theta when foraging is not centered on 0 rad
theta[2] <- tmp[2] * pi * 2

# prior for gamma
# assume high autocorrelation when transitting
gamma[1] ~ dbeta(48, 16)

# assume low autocorrelation when foraging
gamma[2] ~ dbeta(10, 10)


alpha[1] ~ dbeta(1, 1)         # prob of transitting at t, given transitting at t-1
alpha[2] ~ dbeta(1, 1)         # prob of transitting at t, given foraging at t-1

lambda[1] ~ dunif(0, 1)
lambda[2] <- 1 - lambda[1]
b[1] ~ dcat(lambda[])          # assign b for t=1

# scaling parameter for estimation error, accounts for diffs. among tags
logpsi ~ dunif(-10, 10)
```

```
psi <- exp(logpsi)

# Priors for first state estimate
for(k in 1:2){
  x[1,k] ~ dt(y[1,k], itau2[1,k], nu[1,k])
  }

# Assume simple random walk to estimate 2nd state estimate
x[2,1:2] ~ dmnorm(x[1,], iSigma[,])

# Process model
for(t in 2:(RegN-1)){

  phi[t,1] <- alpha[b[t-1]]
  phi[t,2] <- 1 - alpha[b[t-1]]
  b[t] ~ dcat(phi[t,])

  Tdx[t,1] <- cos(theta[b[t]]) * (x[t,1] - x[t-1,1]) +
               sin(theta[b[t]]) * (x[t,2] - x[t-1,2])
  x.mn[t,1] <- x[t,1] +  Tdx[t,1] * gamma[b[t]]

  Tdx[t,2] <- -sin(theta[b[t]]) * (x[t,1] - x[t-1,1]) +
                cos(theta[b[t]]) * (x[t,2] - x[t-1,2])
  x.mn[t,2] <- x[t,2] +  Tdx[t,2] * gamma[b[t]]

  # predict next state estimate w process error
  x[t+1,1:2] ~ dmnorm(x.mn[t,], iSigma[,])
  }

# Observation model
for(t in 2:RegN){                    # loop over regular time intervals (t)
  for(i in idx[t-1]:(idx[t]-1)){     # loop over observed locations within interval t
    for(k in 1:2){                   # loop over lon & lat
      # re-scale obs. error
      itau2.psi[i,k] <- itau2[i,k] * psi

      # reconcile irregular obs. with states
      zhat[i,k] <- (1-j[i]) * x[t-1,k] + j[i] * x[t,k]

      # update state predictions with observations
      y[i,k] ~ dt(zhat[i,k], itau2.psi[i,k], nu[i,k])
      }
    }
  }
}

```