Supplementary information: Among-species heterogeneity in sample size does not bias the estimation of synchrony

Timothée Bonnet

November 23, 2017

Contents

1 Goal and rationale

The primary goal of this appendix is to test the idea that among-species heterogeneity in sample size might bias the estimation of synchrony among species. The verbal argument behind this concern is that the time-dynamic of the most common species might "spill over" the estimation of the time-dynamic common to all species. A suggested solution to this problem has been not to model synchrony explicitly, so that the model does not contain an across-species time-dynamic, but instead to estimate species-specific time dynamics independently, and reconstruct the synchrony a posteriori. Technically, this would be done by fitting a species-by-time random interaction without fitting the main random effect of time (but fitting the main random effect of species).

However, theory suggest that mixed models are generally able to handle unbalanced data, and that the estimation of synchrony should not be biased by unequal sample sizes. Moreover, theory also suggests that the "solution" of not modeling synchrony explicitly and reconstructed it a posteriori from model prediction is flawed, and will generally lead to under-estimation.

Here, we test that these two expectations hold for the special case of mixed mark-recapture models used in the main text, and that the results presented in the main text do not suffer from a bias due to unequal sample size among species. To this end, we first multi-species mark-recapture data with synchrony among species or with no synchrony among species. We use the same sample size per year and per species as in the real data, but halved, in order to speed up modeling. We then analyse the simulated data either with a model that explicitly contain synchrony, or with a model that does not contain explicit synchrony but only allow for its reconstruction from species-specific estimates.

We first show that when sychrony is present in simulated data, a model explicitly modeling synchrony can recover it. We then show that on the same data, a model that does not explicitly model synchrony, but instead makes it estimation possible a posteriori, fails to estimate synchrony. Finally, we show that the first model (explicitly modeling synchrony) does not detect synchrony if none is simulated in the data.

2 Load packages and data

We need `R2jags` to communicate with `JAGS`. We use `reshape` to convert a matrix into a data frame. We use `lme4` for variance partitioning. We also load the real sample sizes per species per year.

```
setwd(dir = "~/Documents/GitHub/penguinteam/SimulsForPubli/")
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

library(reshape)
library(lme4)

## Loading required package:  Matrix
##
## Attaching package:  'Matrix'
## The following object is masked from 'package:reshape':
##
##    expand

SpNumbers <- read.table(file = "SpNumbers.txt", header=TRUE)
SpNumbers <- SpNumbers[,-1]
```

3 Strong synchrony

3.1 Data simulation

We use an additive model to simulate data, so that the effect of time is the same for all species, leading to strong (virtually perfect) synchrony.

```
n.occasions <- ncol(SpNumbers)  # Number of capture occasions
meanphi <- 0 #logit intercept survival (i.e 0.5 on survival scale)
effect_Occasions<-c(0,rnorm(n = n.occasions-2,mean = 0,sd = 0.5))#time effect

n.sp <- nrow(SpNumbers)
effect_Sp<-c(0,rnorm(n = n.sp-1,mean=0,sd=1))


#marked <- round(100*(1:n.sp)^-2)   # Annual number of newly marked individuals
p <- rep(0.4, n.occasions-1)#constant recapture probability

# Define matrices with survival and recapture probabilities
PHI <- matrix( data = NA, ncol = n.occasions-1, nrow = sum( SpNumbers) )
SpTots <- rowSums(SpNumbers)
CumulTots <- c(0, cumsum(SpTots))
sp_ind <- vector(length = sum(SpTots))
for (sp in 1:n.sp)
{
  spe <-  effect_Sp[sp]
  sp_ind[(CumulTots[sp]+1):CumulTots[sp+1]] <- sp
  #nind <- sum(SpNumbers[sp,])
  for (occ in 1:(n.occasions-1))
  {
    oce <- effect_Occasions[occ]
    PHI[(CumulTots[sp]+1):CumulTots[sp+1],occ] <- 1/(1+exp(-( meanphi + oce + spe) ) )
  }

}

P <- matrix(p, ncol = n.occasions-1, nrow = sum(SpTots))
```

```
# Define function to simulate a capture-history (CH) matrix
simul.cjs <- function(PHI, P, SpTots){
  n.occasions <- dim(PHI)[2] + 1
  CH <- matrix(0, ncol = n.occasions, nrow = sum(SpTots))
  mark.occ <- rep(1:length(SpTots), SpTots[1:length(SpTots)])  # Define a vector with the occasion o

  for (i in 1:sum(SpTots)){# Fill the CH matrix
    CH[i, mark.occ[i]] <- 1       # Write an 1 at the release occasion
    if (mark.occ[i]==n.occasions) next
    for (t in (mark.occ[i]+1):n.occasions){
      # Bernoulli trial: does individual survive occasion?
      sur <- rbinom(1, 1, PHI[i,t-1])
      if (sur==0) break          # If dead, move to next individual

      rp <- rbinom(1, 1, P[i,t-1])# Bernoulli trial: is individual recaptured?
      if (rp==1) CH[i,t] <- 1
    } #t
  } #i
  return(CH)
}
```

```
# Execute function
CH <- simul.cjs(PHI, P, SpTots)

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)
```

Functions to help fit MR models:

```
known.state.cjs <- function(ch){
  state <- ch
  for (i in 1:dim(ch)[1]){
    n1 <- min(which(ch[i,]==1))
    n2 <- max(which(ch[i,]==1))
    state[i,n1:n2] <- 1
    state[i,n1] <- NA
  }
  state[state==0] <- NA
  return(state)
}


cjs.init.z <- function(ch,f){
  for (i in 1:dim(ch)[1]){
    if (sum(ch[i,])==1) next
    n2 <- max(which(ch[i,]==1))
    ch[i,f[i]:n2] <- NA
  }
  for (i in 1:dim(ch)[1]){
    ch[i,1:f[i]] <- NA
  }
  return(ch)
}
```

3.2 Interaction model with explicit synchorny

We now fit a model that try to estimate the synchrony on the data containing strong synchrony.

```
sink("cjs-phi(spXt)p(.).jags")
cat([1204 chars quoted with '"'],fill = TRUE)
sink()
```

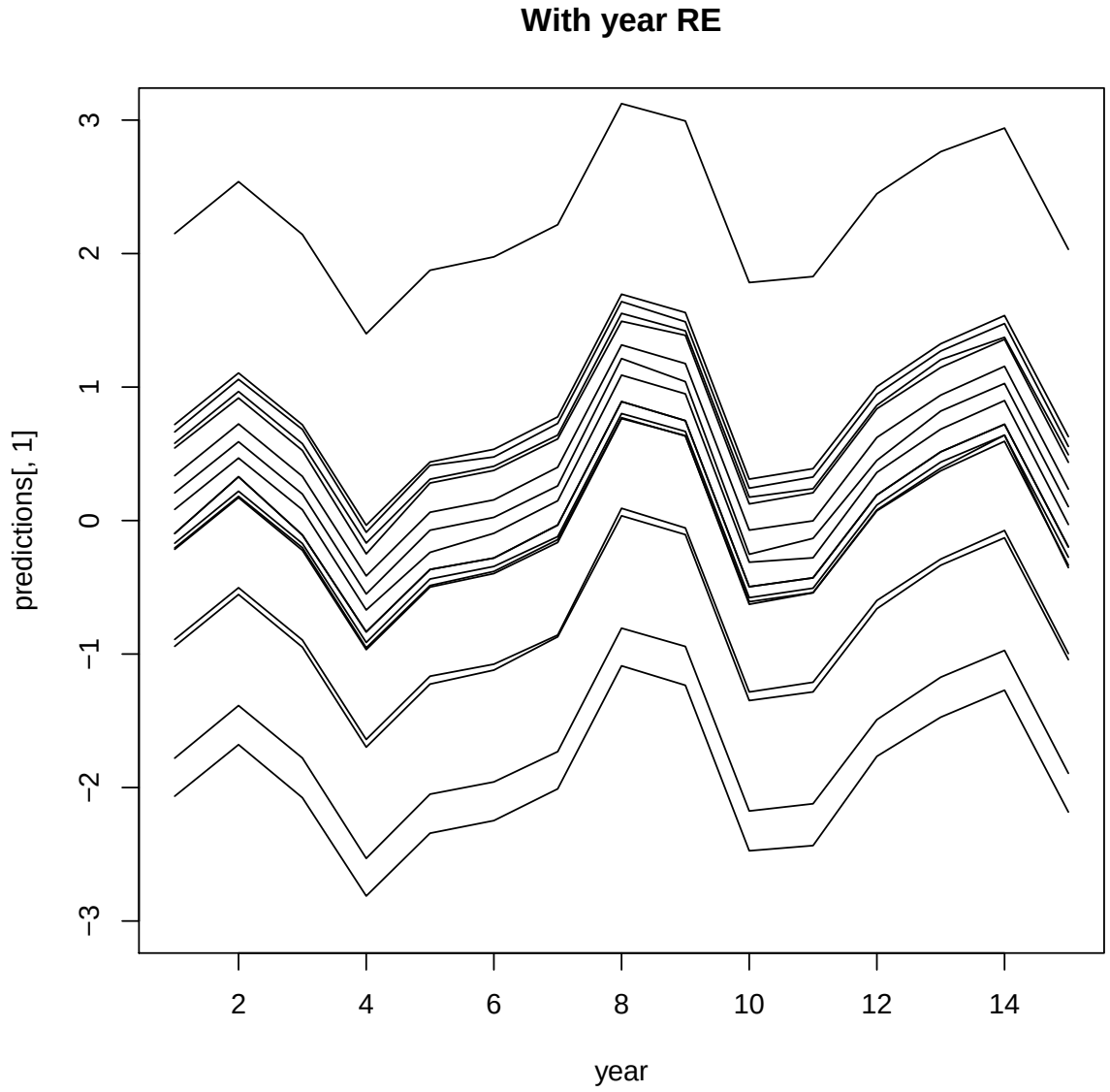
```
# Bundle data
jags.data <- list(y = CH, f = f, nind = dim(CH)[1],
                  n.occasions = dim(CH)[2],
                  z = known.state.cjs(CH),
                  sp_ind=sp_ind, n.sp=n.sp)

#### Initial values####
#interaction init

beta.tXsp_ini<-function()
{beta.tXsp<-matrix(NA,nrow = n.occasions-1,ncol = n.sp)
for (i in 1:(n.occasions-1))
  {
    for (j in 1:n.sp)
```

```
      {
        beta.tXsp[i,j]<-rnorm(n = 1,mean = 0,sd = 1)
      }
  }
return(beta.tXsp)
}

inits <- function(){list(z = cjs.init.z(CH, f),
                phi.mean = runif(1, 0, 1), p.mean = runif(1, 0, 1),
                beta.t=c(rnorm(n = n.occasions-1,mean = 0,sd=1 ) ),
                beta.sp=c(rnorm(n = n.sp,mean = 0,sd=1 ) ),
                beta.tXsp= beta.tXsp_ini() )}

# Parameters monitored
parameters <- c("phi.mean", "p.mean",
                "beta.t","beta.sp",
                "beta.tXsp", "sigma.sp",
                "sigma.t", "sigma.tXsp")

# MCMC settings
ni <- 11000
nt <- 10
nb <- 1000
nc <- 3

cjs.interactionS <- jags(jags.data, inits, parameters,
                         "cjs-phi(spXt)p(.).jags",
                        n.chains = nc, n.thin = nt,
                        n.iter = ni, n.burnin = nb,
                        working.directory = getwd())

print(cjs.interactionS, digits = 3)
```

```
load(file = "ForCluster/cjs.interactionS_dataSynchro")
```

The synchrony is well recovered graphically:

```
# Summarize posteriors

predictions <- cjs.interactionS$BUGSoutput$mean$beta.tXsp +
  matrix(cjs.interactionS$BUGSoutput$mean$beta.t,
         nrow = n.occasions-1, ncol = n.sp, byrow = FALSE) +
  matrix(cjs.interactionS$BUGSoutput$mean$beta.sp,
         nrow = n.occasions-1, ncol=n.sp, byrow = TRUE)


plot(predictions[,1], type="l", ylim=c(-3,3),
     xlab="year", main="With year RE")
for(i in 1:n.sp)
{
  lines(predictions[,i])
}
```



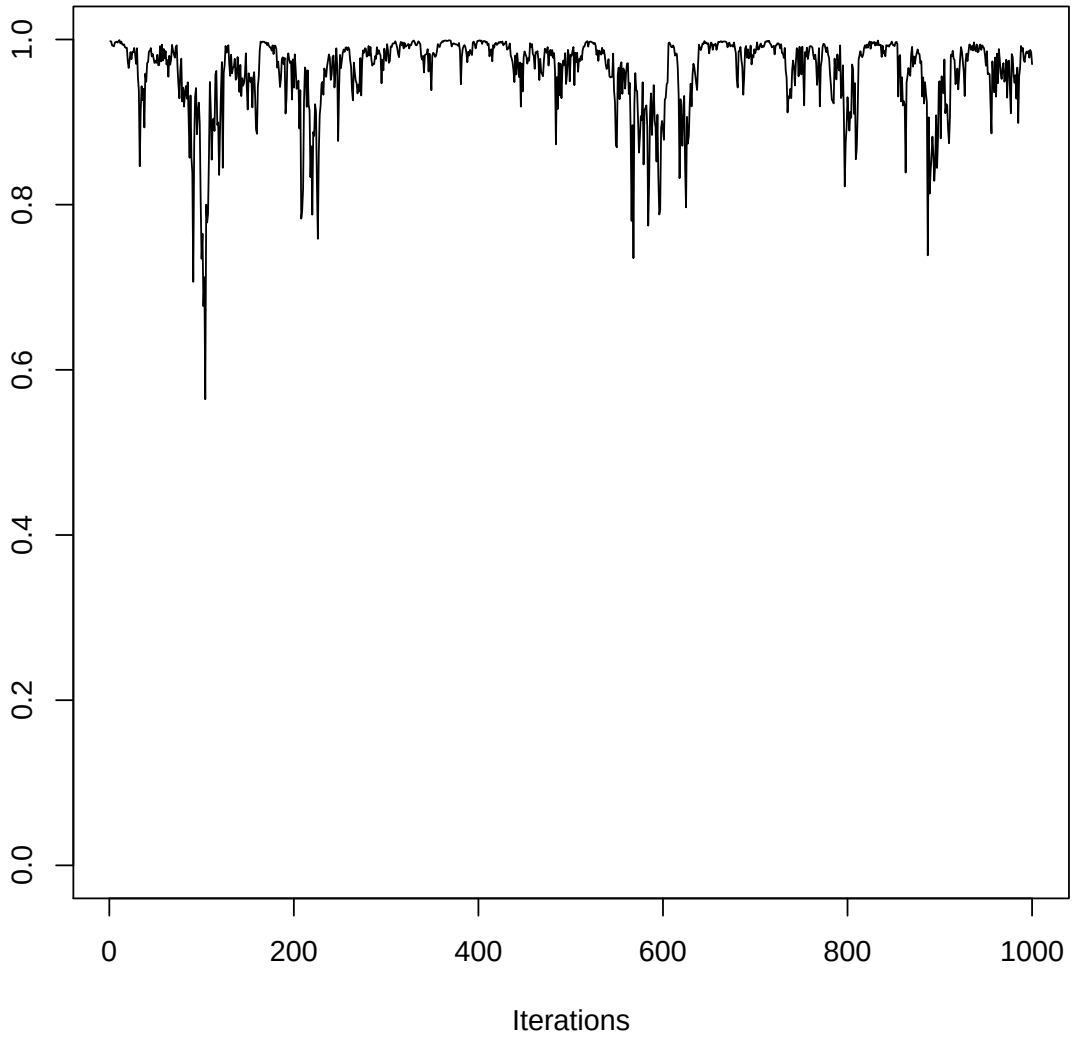
The intra-Class correlation is close to one, as revealed by the point estimate:

```
(cjs.interactionS$BUGSoutput$mean$sigma.t^2)/
  (cjs.interactionS$BUGSoutput$mean$sigma.tXsp^2+
     cjs.interactionS$BUGSoutput$mean$sigma.t^2)

## [1] 0.9717394
```

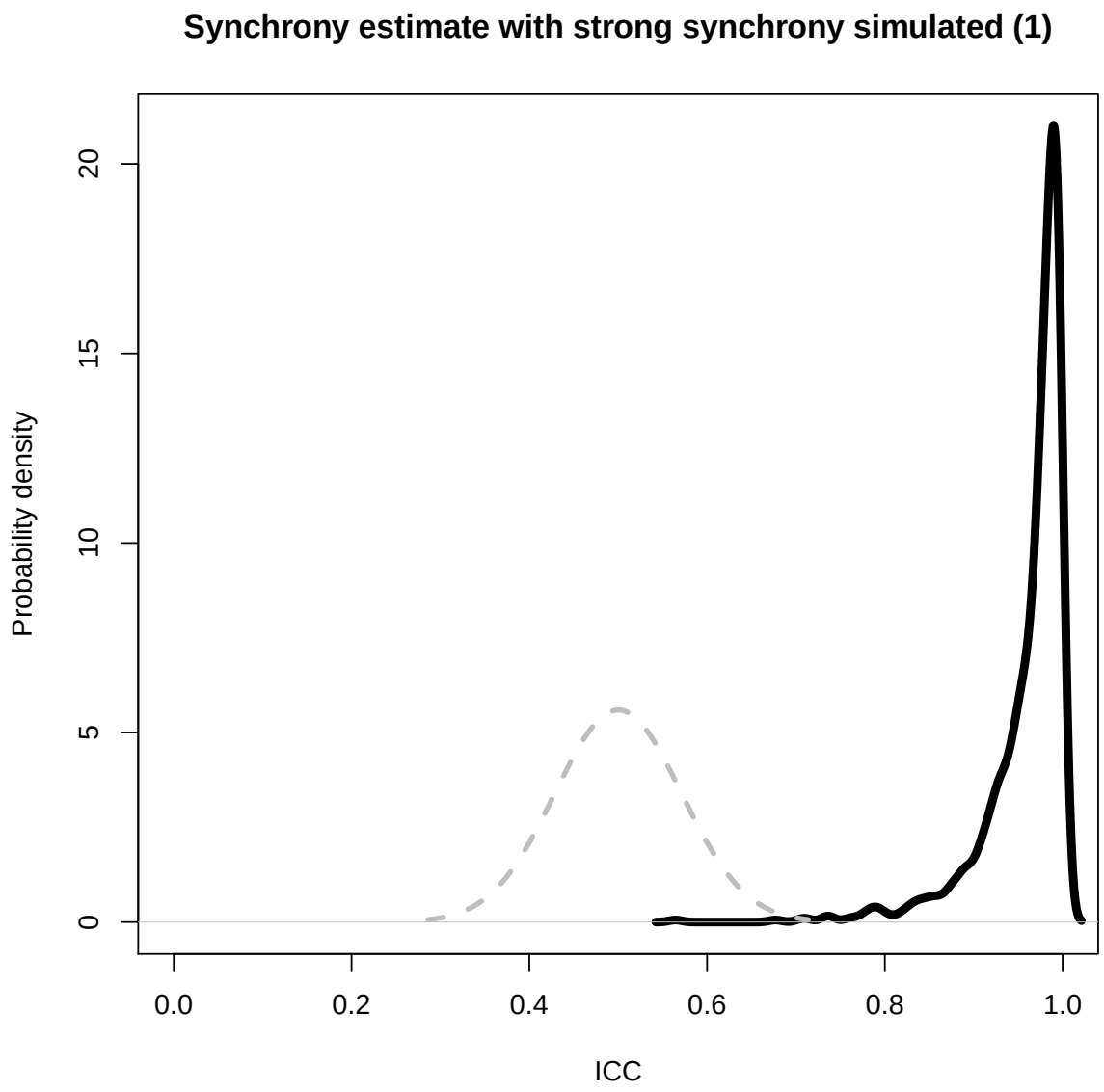
As well as the posterior distribution:

```
iccpost <- vector(length=1000)
for (ch in 1:1)
{
  for (itt in 1:1000)
  {
    iccpost[itt + 1000*(ch-1)] <-
      cjs.interactionS$BUGSoutput$sims.array[itt, ch,"sigma.t"]^2 /
      ( cjs.interactionS$BUGSoutput$sims.array[itt, ch,"sigma.t"]^2 +
          cjs.interactionS$BUGSoutput$sims.array[itt, ch,"sigma.tXsp"]^2)
  }
}
```

```r
sdprior <- runif(n = 10000, min = 0, max = 10)
sdprior2 <- sdprior

iccprior <- (sdprior^2)/(sdprior^2+sdprior2^2)

traceplot(as.mcmc(iccpost), ylim=c(0,1))
```




```r
plot(density(iccpost), xlim = c(0,1), lwd=5,
     main="Synchrony estimate with strong synchrony simulated (1)",
     xlab="ICC", ylab="Probability density")

lines(density(iccprior), lwd=3, lty=2, col="gray")
```



3.3 Interactive model without explicit synchrony

We analyse the same data, but this time using a model that does not explicitly contain synchrony.

```
sink("cjs-phi(spXt_no_t)p(.).jags")
cat([1060 chars quoted with '"'],fill = TRUE)
sink()
```

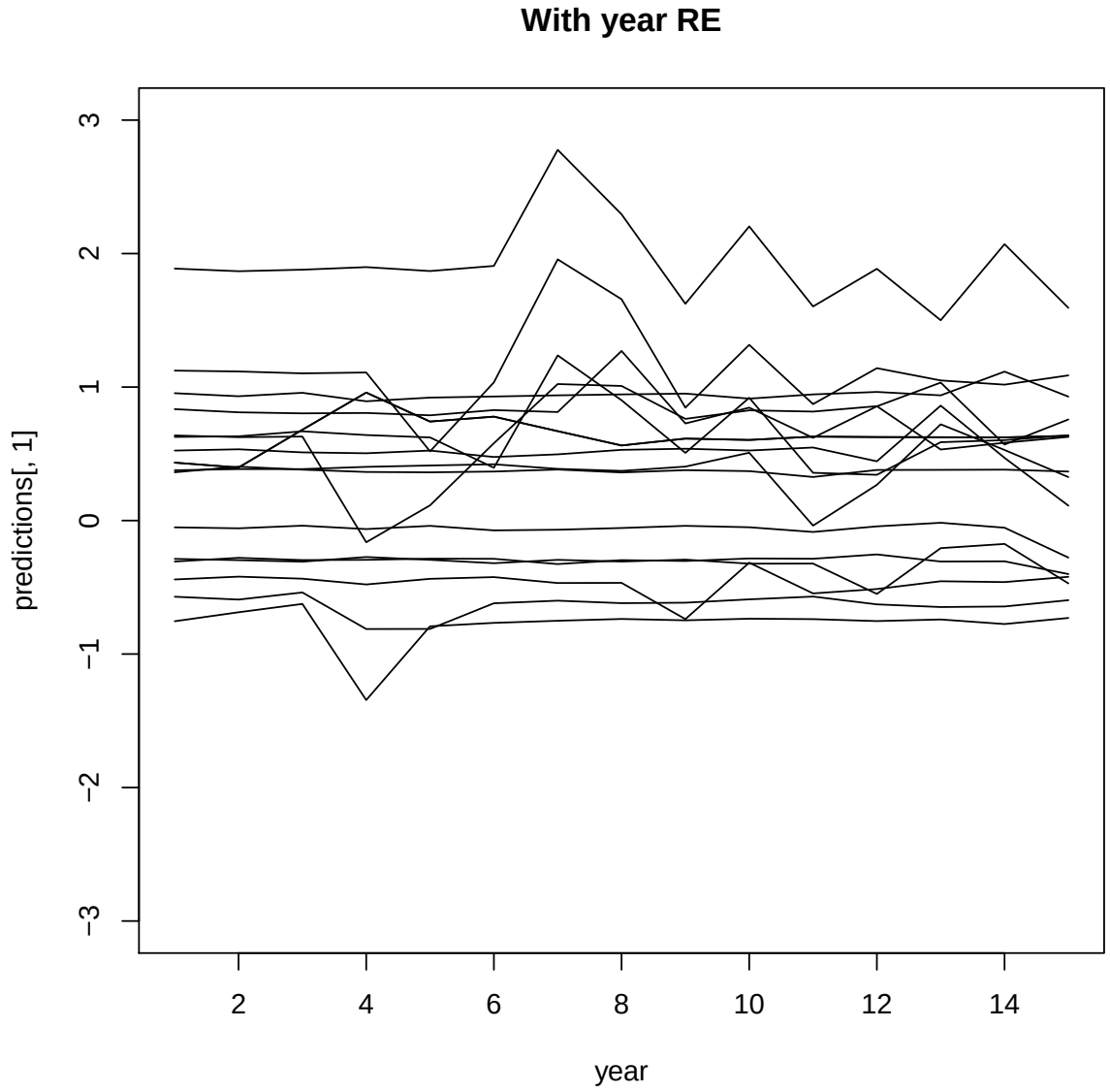
```
# Bundle data
jags.data <- list(y = CH, f = f, nind = dim(CH)[1],
                  n.occasions = dim(CH)[2],
                  z = known.state.cjs(CH),
                  sp_ind=sp_ind, n.sp=n.sp)

#### Initial values####
#interaction init

beta.tXsp_ini<-function()
{beta.tXsp<-matrix(NA,nrow = n.occasions-1,ncol = n.sp)
```

```
for (i in 1:(n.occasions-1))
  {
    for (j in 1:n.sp)
      {
        beta.tXsp[i,j]<-rnorm(n = 1,mean = 0,sd = 1)
      }
  }
return(beta.tXsp)
}

inits <- function(){list(z = cjs.init.z(CH, f),
                  phi.mean = runif(1, 0, 1), p.mean = runif(1, 0, 1),
                  beta.sp=c(rnorm(n = n.sp,mean = 0,sd=1 ) ),
                  beta.tXsp= beta.tXsp_ini() )}

# Parameters monitored
parameters <- c("phi.mean", "p.mean",
                "beta.sp","beta.tXsp",
                "sigma.sp", "sigma.tXsp")

# MCMC settings
ni <- 11000
nt <- 10
nb <- 1000
nc <- 3

cjs.interactionNot <- jags(jags.data, inits,
                           parameters, "cjs-phi(spXt_no_t)p(.).jags",
                        n.chains = nc, n.thin = nt,
                        n.iter = ni, n.burnin = nb,
                        working.directory = getwd())

print(cjs.interactionNot, digits = 3)
```

```
load(file = "ForCluster/cjs.interactionSNot_dataSynchro")
```

Graphically, much less synchrony is apparent:

```
predictions <-  matrix(cjs.interactionNot$BUGSoutput$mean$phi.mean,
                       nrow = n.occasions-1, ncol=n.sp, byrow = TRUE) +
      cjs.interactionNot$BUGSoutput$mean$beta.tXsp +
  matrix(cjs.interactionNot$BUGSoutput$mean$beta.sp,
         nrow = n.occasions-1, ncol=n.sp, byrow = TRUE)


plot(predictions[,1], type="l", ylim=c(-3,3), xlab="year", main="With year RE")
for(i in 1:n.sp)
{
  lines(predictions[,i])
}
```



The intra class correlation (ratio of temporal variance common to all species over the total temporal variance) is weak and close to zero.

```
meltedpredictions <- melt(predictions)
summary(aov(formula = value ~ X2 + X1, data=meltedpredictions))

##              Df Sum Sq Mean Sq F value Pr(>F)
## X2            1   0.13  0.1290   0.244  0.622
## X1            1   0.00  0.0035   0.007  0.935
## Residuals   237 125.32  0.5288

summary(lmer(value ~ 1 + (1|X1)+ (1|X2), data=meltedpredictions))

## Linear mixed model fit by REML ['lmerMod']
## Formula: value ~ 1 + (1 | X1) + (1 | X2)
##    Data: meltedpredictions
##
## REML criterion at convergence: -70.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -4.4036 -0.3071  0.0389  0.3131  4.4310
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  X2       (Intercept) 0.523956 0.7238
##  X1       (Intercept) 0.004045 0.0636
##  Residual             0.027855 0.1669
## Number of obs: 240, groups:  X2, 16; X1, 15
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   0.3405     0.1820   1.871
```

4 No synchrony

We now simulate data without any synchrony.

```
effect_OccasionsSP<-matrix(c(rnorm(n = (n.occasions-1)*n.sp,
                                   mean = 0,sd = 0.5)),
                           nrow=n.sp,
                           ncol=(n.occasions-1))#time effect

effect_Sp<-c(0,rnorm(n = n.sp-1,mean=0,sd=1))


# Define matrices with survival and recapture probabilities
PHI <- matrix( data = NA, ncol = n.occasions-1, nrow = sum( SpNumbers) )
SpTots <- rowSums(SpNumbers)
CumulTots <- c(0, cumsum(SpTots))
sp_ind <- vector(length = sum(SpTots))
for (sp in 1:n.sp)
{
  spe <-  effect_Sp[sp]
  sp_ind[(CumulTots[sp]+1):CumulTots[sp+1]] <- sp
  #nind <- sum(SpNumbers[sp,])
  for (occ in 1:(n.occasions-1))
  {
    oce <- effect_Occasions[occ]
    PHI[(CumulTots[sp]+1):CumulTots[sp+1],occ] <-
      1/(1+exp(-( meanphi + effect_OccasionsSP[sp,occ] + spe) ) )
  }
}
```

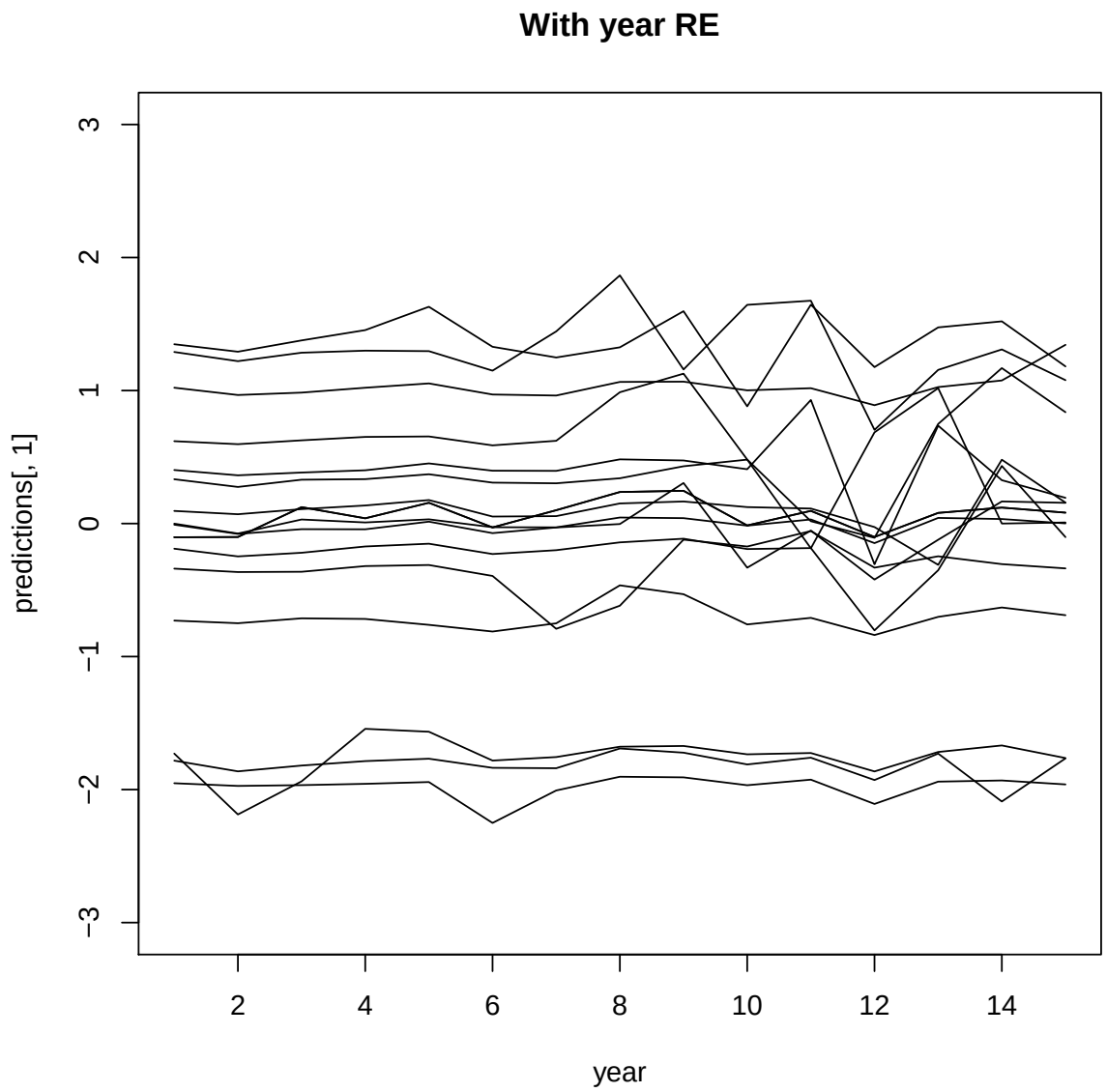
```
# Execute function
CH <- simul.cjs(PHI, P, SpTots)

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)
```

We fit the model with explicit synchrony.

```
load(file = "ForCluster/cjs.interactionS_dataNoSynchro")
```

There graphically:

```
# Summarize posteriors

predictions <- cjs.interactionSdataNoS$BUGSoutput$mean$beta.tXsp +
  matrix(cjs.interactionSdataNoS$BUGSoutput$mean$beta.t,
         nrow = n.occasions-1, ncol = n.sp, byrow = FALSE) +
  matrix(cjs.interactionSdataNoS$BUGSoutput$mean$beta.sp,
         nrow = n.occasions-1, ncol=n.sp, byrow = TRUE)


plot(predictions[,1], type="l", ylim=c(-3,3), xlab="year", main="With year RE")
for(i in 1:n.sp)
{
  lines(predictions[,i])
}
```

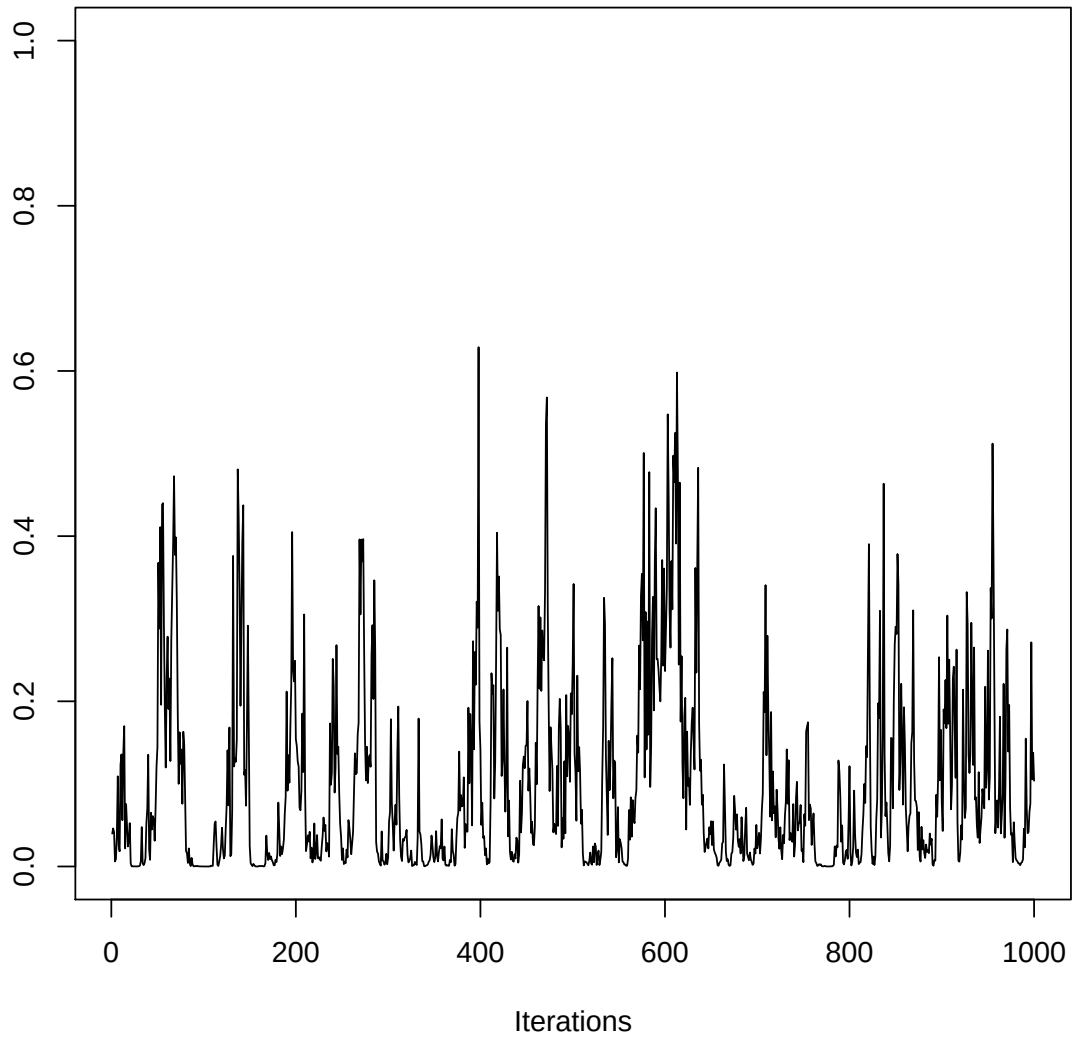


Intra-Class correlation:

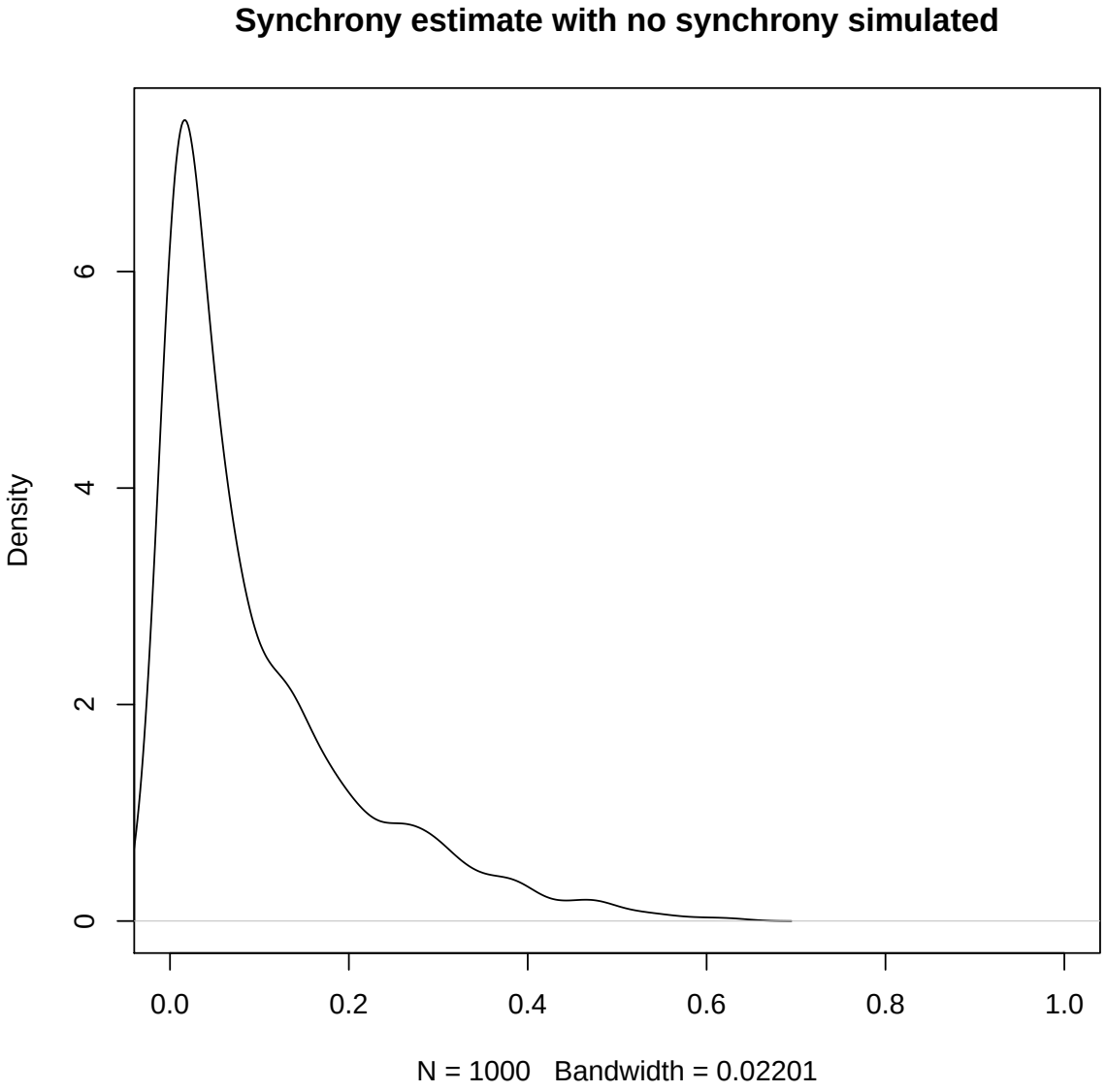
```
(cjs.interactionSdataNoS$BUGSoutput$mean$sigma.t^2)/
  (cjs.interactionSdataNoS$BUGSoutput$mean$sigma.tXsp^2+
     cjs.interactionSdataNoS$BUGSoutput$mean$sigma.t^2)
```

```
## [1] 0.07141729
```

```
iccpost <- vector(length=1000)
for (ch in 1:1)
{
  for (itt in 1:1000)
  {
    iccpost[itt + 1000*(ch-1)] <-
      cjs.interactionSdataNoS$BUGSoutput$sims.array[itt, ch,"sigma.t"]^2 /
      ( cjs.interactionSdataNoS$BUGSoutput$sims.array[itt, ch,"sigma.t"]^2 +
          cjs.interactionSdataNoS$BUGSoutput$sims.array[itt, ch,"sigma.tXsp"]^2)
  }
}
traceplot(as.mcmc(iccpost), ylim=c(0,1))
```




```
plot(density(iccpost), xlim = c(0,1),
     main="Synchrony estimate with no synchrony simulated")
```


Synchrony estimate with no synchrony simulated
Density
0
2
4
6
0.0
0.2
0.4
0.6
0.8
1.0
N = 1000 Bandwidth = 0.02201