Simulations

1. Introduction

In this appendix, we carry out a small simulation study to assess the bias in the parameter estimates of a capture-recapture model to infer social networks.

2. Code writing for simulation and estimation

First, we load the `R2jags` package that will be used to fit models.

```
library(R2jags)
```

Then we write a function `sim_CRnetwork` to simulate data. Note that it is used in `Jags`, as explained here.

```
sim_CRnetwork <- function(J = 5, n = 105, ppA = 0.7, ppB = 0.7, psiAA = 0.3,
psiBB = 0.8, pi = 0.7){
library(runjags)
# code to simulate with jags, note the use of the data block
# parameters for simulations
# J = nb occasions
# n = nb of dyads (= N(N-1)/2 where N is the number of individuals); by
default, we consider N = 15 individuals, hence N(N-1)/2 = 105 possible dyads
# ppA = detection for associated dyads
# ppB = detection for non-associated dyads
# psiAA = pr of staying associated
# psiBB = pr of staying non-associated
# pi = initial state pr
txtstring <- '
data{

# A = associated
# B = non-associated

#Pr(dyads state)
px[1,1] <- psiAA # probability of staying associated
px[1,2] <- 1 - psiAA # probability of associated -> non-associated
px[2,1] <- 1 - psiBB # probability of non-associated -> associated
px[2,2] <- psiBB # probability of staying non-associated

# Pr(dyads obs given dyads state)
## ppA is the individual detection probability for associated dyads
## ppB is the individual detection probability for non-associated dyads
    po[1,1] <- (1-ppA) * (1-ppA)
    po[1,2] <- 2 * ppA * (1-ppA)
    po[1,3] <- ppA * ppA
```

```
    po[1,4] <- 0
    po[2,1] <- (1-ppB) * (1-ppB)
    po[2,2] <- 2 * ppB * (1-ppB)
    po[2,3] <- 0
    po[2,4] <- ppB * ppB

# Pr(initial states)
px0[1] <- pi # prob. of being in initial state A
px0[2] <- 1-pi # prob. of being in initial state B

# Model likelihood
    for (i in 1:n){

        # record states for every sampling occasion
        x1[i] <- x[i,1]
        x2[i] <- x[i,2]
        x3[i] <- x[i,3]
        x4[i] <- x[i,4]
        x5[i] <- x[i,5]

        # for t = 1
        x[i,1] ~ dcat(px0[1:2])
        obs[i,1] ~ dcat(po[x[i,1],1:4])

        # for t > 1
        for (t in 2:J){

            #-- state equation
            # 1 = associated
            # 2 = non-associated
            x[i,t] ~ dcat(px[x[i,t-1],1:2])

            #-- observation equation
            # 1 = dyad non-observed,
            # 2 = one of the two individuals non-observed,
            # 3 = dyad seen and associated
            # 4 = dyad seen and non-associated
            obs[i,t] ~ dcat(po[x[i,t],1:4])
                        }
                    }
}
model{
fake <- 0
}
'

# parameters are treated as data for the simulation step
data<-list(n=n, J=J, ppA=ppA, ppB=ppB, psiAA=psiAA, psiBB=psiBB, pi=pi)
```

```
# run jags
out <- run.jags(txtstring, data = data, monitor=c("obs","x"), sample=1,
n.chains=1, summarise=FALSE)

# reformat the outputs
Simulated <- coda::as.mcmc(out)
#Simulated
#dim(Simulated)
dat <- matrix(Simulated[1:(n*J)],ncol=J)
#dat
states <- matrix(Simulated[-(1:(n*J))],ncol=J)
#states
list(dat=dat,states=states) # outputs: dat = detections/non-detections;
states = underlying states
}
```

In another step, we specify the model that will be used to estimate network parameters:

```
sink("sim_network_hom.txt")
cat("
model{

# Pr(dyads state)
px[1,1] <- psiAA            # probability of staying associated
px[1,2] <- 1 - psiAA        # probability of associated -> non-associated
px[2,1] <- 1 - psiBB        # probability of non-associated -> associated
px[2,2] <- psiBB            # probability of staying non-associated

# Pr(dyads obs given dyads state)
## pp is the individual detection probability
po[1,1] <- (1-pp) * (1-pp)
po[1,2] <- 2 * pp * (1-pp)
po[1,3] <- pp * pp
po[1,4] <- 0
po[2,1] <- (1-pp) * (1-pp)
po[2,2] <- 2 * pp * (1-pp)
po[2,3] <- 0
po[2,4] <- pp * pp

# Pr(initial states)
px0[1] <- pi                # prob. of being in initial state A
px0[2] <- 1-pi              # prob. of being in initial state B

# Model likelihood
for (i in 1:n){

    # record states for every sampling occasion
    x1[i] <- x[i,1]
    x2[i] <- x[i,2]
    x3[i] <- x[i,3]
```

```
    x4[i] <- x[i,4]
    x5[i] <- x[i,5]

    # for t = 1
    x[i,1] ~ dcat(px0[1:2])
    obs[i,1] ~ dcat(po[x[i,1],1:4])

    # for t > 1
    for (t in 2:J){
        #-- state equation # 1 = associated # 2 = non-associated
        x[i,t] ~ dcat(px[x[i,t-1],1:2])
        #-- observation equation (1,2,3 ó 4)
        obs[i,t] ~ dcat(po[x[i,t],1:4])
            }
        }

# Priors
pp ~ dunif(0,1) # detection pr
psiAA ~ dunif(0,1) # pr of staying associated
psiBB ~ dunif(0,1) # pr of staying non-associated
pi ~ dunif(0,1) # initial state pr
}
",fill=TRUE)

##
## model{
##
## # Pr(dyads state)
## px[1,1] <- psiAA             # probability of staying associated
## px[1,2] <- 1 - psiAA         # probability of associated -> non-associated
## px[2,1] <- 1 - psiBB         # probability of non-associated -> associated
## px[2,2] <- psiBB             # probability of staying non-associated
##
## # Pr(dyads obs given dyads state)
## ## pp is the individual detection probability
## po[1,1] <- (1-pp) * (1-pp)
## po[1,2] <- 2 * pp * (1-pp)
## po[1,3] <- pp * pp
## po[1,4] <- 0
## po[2,1] <- (1-pp) * (1-pp)
## po[2,2] <- 2 * pp * (1-pp)
## po[2,3] <- 0
## po[2,4] <- pp * pp
##
## # Pr(initial states)
## px0[1] <- pi                 # prob. of being in initial state A
## px0[2] <- 1-pi               # prob. of being in initial state B
##
## # Model likelihood
## for (i in 1:n){
```

```
##
##  # record states for every sampling occasion
##  x1[i] <- x[i,1]
##  x2[i] <- x[i,2]
##  x3[i] <- x[i,3]
##  x4[i] <- x[i,4]
##  x5[i] <- x[i,5]
##
##  # for t = 1
##  x[i,1] ~ dcat(px0[1:2])
##  obs[i,1] ~ dcat(po[x[i,1],1:4])
##
##  # for t > 1
##  for (t in 2:J){
##      #-- state equation # 1 = associated # 2 = non-associated
##      x[i,t] ~ dcat(px[x[i,t-1],1:2])
##      #-- observation equation (1,2,3 ó 4)
##      obs[i,t] ~ dcat(po[x[i,t],1:4])
##          }
##      }
##
## # Priors
## pp ~ dunif(0,1) # detection pr
## psiAA ~ dunif(0,1) # pr of staying associated
## psiBB ~ dunif(0,1) # pr of staying non-associated
## pi ~ dunif(0,1) # initial state pr
## }

sink()
```

We also consider the same model as above with heterogeneous detection probabilities:

```
sink("sim_network_het.txt")
cat("
model{

# Pr(dyads state)
px[1,1] <- psiAA            # probability of staying associated
px[1,2] <- 1 - psiAA        # probability of associated -> non-associated
px[2,1] <- 1 - psiBB        # probability of non-associated -> associated
px[2,2] <- psiBB            # probability of staying non-associated

# Pr(dyads obs given dyads state)
## ppA is the individual detection probability for associated dyads
## ppB is the individual detection probability for non-associated dyads
po[1,1] <- (1-ppA) * (1-ppA)
po[1,2] <- 2 * ppA * (1-ppA)
po[1,3] <- ppA * ppA
po[1,4] <- 0
po[2,1] <- (1-ppB) * (1-ppB)
```

```
po[2,2] <- 2 * ppB * (1-ppB)
po[2,3] <- 0
po[2,4] <- ppB * ppB

# Pr(initial states)
px0[1] <- pi                # prob. of being in initial state A
px0[2] <- 1-pi              # prob. of being in initial state B

# Model likelihood
for (i in 1:n){

    # record states for every sampling occasion
    x1[i] <- x[i,1]
    x2[i] <- x[i,2]
    x3[i] <- x[i,3]
    x4[i] <- x[i,4]
    x5[i] <- x[i,5]

    # for t = 1
    x[i,1] ~ dcat(px0[1:2])
    obs[i,1] ~ dcat(po[x[i,1],1:4])

    # for t > 1
    for (t in 2:J){
        #-- state equation # 1 = associated # 2 = non-associated
        x[i,t] ~ dcat(px[x[i,t-1],1:2])
        #-- observation equation (1,2,3 ó 4)
        obs[i,t] ~ dcat(po[x[i,t],1:4])
            }
        }

# Priors
ppA ~ dunif(0,1) # detection pr
ppB ~ dunif(0,1) # detection pr
psiAA ~ dunif(0,1) # pr of staying associated
psiBB ~ dunif(0,1) # pr of staying non-associated
pi ~ dunif(0,1) # initial state pr
}
",fill=TRUE)

##
## model{
##
## # Pr(dyads state)
## px[1,1] <- psiAA             # probability of staying associated
## px[1,2] <- 1 - psiAA         # probability of associated -> non-associated
## px[2,1] <- 1 - psiBB         # probability of non-associated -> associated
## px[2,2] <- psiBB             # probability of staying non-associated
##
## # Pr(dyads obs given dyads state)
```

```
## ## ppA is the individual detection probability for associated dyads
## ## ppB is the individual detection probability for non-associated dyads
## po[1,1] <- (1-ppA) * (1-ppA)
## po[1,2] <- 2 * ppA * (1-ppA)
## po[1,3] <- ppA * ppA
## po[1,4] <- 0
## po[2,1] <- (1-ppB) * (1-ppB)
## po[2,2] <- 2 * ppB * (1-ppB)
## po[2,3] <- 0
## po[2,4] <- ppB * ppB
##
## # Pr(initial states)
## px0[1] <- pi                 # prob. of being in initial state A
## px0[2] <- 1-pi               # prob. of being in initial state B
##
## # Model likelihood
## for (i in 1:n){
##
##  # record states for every sampling occasion
##  x1[i] <- x[i,1]
##  x2[i] <- x[i,2]
##  x3[i] <- x[i,3]
##  x4[i] <- x[i,4]
##  x5[i] <- x[i,5]
##
##  # for t = 1
##  x[i,1] ~ dcat(px0[1:2])
##  obs[i,1] ~ dcat(po[x[i,1],1:4])
##
##  # for t > 1
##  for (t in 2:J){
##      #-- state equation # 1 = associated # 2 = non-associated
##      x[i,t] ~ dcat(px[x[i,t-1],1:2])
##      #-- observation equation (1,2,3 ó 4)
##      obs[i,t] ~ dcat(po[x[i,t],1:4])
##          }
##      }
##
## # Priors
## ppA ~ dunif(0,1) # detection pr
## ppB ~ dunif(0,1) # detection pr
## psiAA ~ dunif(0,1) # pr of staying associated
## psiBB ~ dunif(0,1) # pr of staying non-associated
## pi ~ dunif(0,1) # initial state pr
## }

sink()
```

3. Simulations: Scenarios with homogeneous detection probabilities

Now we proceed with the simulations. First, we consider the situation where detection probabilities are homogeneous irrespective of the status of the dyads. We define the scenarios we would like to investigate:

- scenarios on the detection probability: 0.3, 0.8;
- scenarios on π: 0.2, 0.7;
- scenarios on ψ_{AA}: 0.1, 0.4, 0.9;
- scenarios on ψ_{BB}: 0.1, 0.4, 0.9.

Therefore, in total we have 36 scenarios.

```
grid <-
expand.grid(pp=c(0.3,0.8),pi=c(0.2,0.7),psiAA=c(0.1,0.4,0.9),psiBB=c(0.1,0.4,
0.9))
grid
```

```
##     pp  pi psiAA psiBB
## 1  0.3 0.2   0.1   0.1
## 2  0.8 0.2   0.1   0.1
## 3  0.3 0.7   0.1   0.1
## 4  0.8 0.7   0.1   0.1
## 5  0.3 0.2   0.4   0.1
## 6  0.8 0.2   0.4   0.1
## 7  0.3 0.7   0.4   0.1
## 8  0.8 0.7   0.4   0.1
## 9  0.3 0.2   0.9   0.1
## 10 0.8 0.2   0.9   0.1
## 11 0.3 0.7   0.9   0.1
## 12 0.8 0.7   0.9   0.1
## 13 0.3 0.2   0.1   0.4
## 14 0.8 0.2   0.1   0.4
## 15 0.3 0.7   0.1   0.4
## 16 0.8 0.7   0.1   0.4
## 17 0.3 0.2   0.4   0.4
## 18 0.8 0.2   0.4   0.4
## 19 0.3 0.7   0.4   0.4
## 20 0.8 0.7   0.4   0.4
## 21 0.3 0.2   0.9   0.4
## 22 0.8 0.2   0.9   0.4
## 23 0.3 0.7   0.9   0.4
## 24 0.8 0.7   0.9   0.4
## 25 0.3 0.2   0.1   0.9
## 26 0.8 0.2   0.1   0.9
## 27 0.3 0.7   0.1   0.9
## 28 0.8 0.7   0.1   0.9
## 29 0.3 0.2   0.4   0.9
```

```
## 30 0.8 0.2   0.4   0.9
## 31 0.3 0.7   0.4   0.9
## 32 0.8 0.7   0.4   0.9
## 33 0.3 0.2   0.9   0.9
## 34 0.8 0.2   0.9   0.9
## 35 0.3 0.7   0.9   0.9
## 36 0.8 0.7   0.9   0.9
```

Let us run the simulations, with 100 Monte Carlo iterations. These simulations take ages, we do not recommend to run them. For convenience, all the results are stored in the object `simul_network_index36_sim100homogeneous.RData` that we provide.

```
# nb of monte carlo iterations
nb_simulations <- 100

# matrix to store results with estimated values for pp, psiAA, psiBB, pi
res <- array(NA,dim=c(nrow(grid),nb_simulations,4))

# run simulation
for (index in 1:nrow(grid)){ # go through grid of scenarios
        for (i in 1:nb_simulations){

# 1. simulate

pp <- grid[index,1]
pi <- grid[index,2]
psiAA <- grid[index,3]
psiBB <- grid[index,4]

sim_data <- sim_CRnetwork(J=5,n=105, ppA = pp, ppB = pp, psiAA = psiAA, psiBB
= psiBB, pi = pi)
dat <- sim_data[[1]]
states <- sim_data[[2]]

# 2. estimation

# initial values
init1 <- list(psiAA=grid[index,3],pp=grid[index,1],x=states)
inits <- list(init1)

# data
jags.data <- list(obs = dat, n = nrow(dat), J = ncol(dat))

# nb iterations
ni <- 2000
# nb burn-in
nb <- 1000
# nb thin
nt <- 1
```

```
# nb chains
nc <- 1

# parameters to be monitored
parameters_sim <- c("psiAA","psiBB","pi","pp","x1","x2","x3","x4","x5")

# call JAGS from R
mod <- jags(jags.data, inits, parameters_sim, 'sim_network.txt', n.chains =
nc, n.thin = nt,
n.iter = ni, n.burnin = nb, working.directory = getwd())

res[index,i,1] <- mean(mod$BUGSoutput$sims.matrix[,'pp']) # detection
res[index,i,2] <- mean(mod$BUGSoutput$sims.matrix[,'psiAA']) # associated
res[index,i,3] <- mean(mod$BUGSoutput$sims.matrix[,'psiBB']) # non-associated
res[index,i,4] <- mean(mod$BUGSoutput$sims.matrix[,'pi']) # prop of
associated

}

}
save(res,file='simul_network_index36_sim100homogeneous.RData')
```

Let us post-process the results by computing relative bias (in percent) for all parameters:

```
load("simul_network_index36_sim100homogeneous.RData")
bias_param <- matrix(NA,nrow(grid),4)
for(i in 1:nrow(grid)){
    for (j in 1:4){
        bias_param[i,j] <- (mean(res[i,,c(1,4,2,3)[j]]) -
grid[i,j])/grid[i,c(1,3,4,2)[j]]*100
    }
}
res_bias <- round(cbind(1:nrow(bias_param),grid,bias_param),2)
colnames(res_bias) <-
c('scenario',names(grid),'bias_pp','bias_pi','bias_psiAA','bias_psiBB')
```

The results are given in the following table, with in the first column the scenarios labels, in columns 2-5 the simulation parameters and in columns 6-9 the relative bias:

```
knitr::kable(res_bias)
```

scenario	pp	pi	psiAA	psiBB	bias_pp	bias_pi	bias_psiAA	bias_psiBB
1	0.3	0.2	0.1	0.1	0.50	26.49	120.98	58.77
2	0.8	0.2	0.1	0.1	0.08	9.46	4.37	1.36
3	0.3	0.7	0.1	0.1	-0.33	-1.23	142.04	22.83
4	0.8	0.7	0.1	0.1	-0.21	-3.23	8.91	-0.04
5	0.3	0.2	0.4	0.1	0.07	14.73	27.30	53.03
6	0.8	0.2	0.4	0.1	-0.04	1.02	-1.65	4.74

7	0.3	0.7	0.4	0.1	0.60	-10.96	65.19	26.16
8	0.8	0.7	0.4	0.1	-0.04	-0.37	-8.88	0.79
9	0.3	0.2	0.9	0.1	0.29	4.46	-23.10	37.30
10	0.8	0.2	0.9	0.1	0.11	2.29	-5.57	7.26
11	0.3	0.7	0.9	0.1	-0.25	0.30	-14.44	28.57
12	0.8	0.7	0.9	0.1	0.07	-0.55	-7.99	3.91
13	0.3	0.2	0.1	0.4	-0.74	54.58	45.20	24.95
14	0.8	0.2	0.1	0.4	-0.08	6.23	2.19	4.71
15	0.3	0.7	0.1	0.4	0.27	-25.83	29.36	7.66
16	0.8	0.7	0.1	0.4	-0.11	-11.71	3.05	1.72
17	0.3	0.2	0.4	0.4	0.45	14.96	10.59	21.80
18	0.8	0.2	0.4	0.4	-0.09	3.22	-1.45	-0.27
19	0.3	0.7	0.4	0.4	0.64	-13.37	5.67	5.96
20	0.8	0.7	0.4	0.4	0.02	0.24	-1.44	-0.71
21	0.3	0.2	0.9	0.4	-0.26	8.35	-17.84	-28.74
22	0.8	0.2	0.9	0.4	0.01	1.28	-1.62	-1.72
23	0.3	0.7	0.9	0.4	0.45	-1.59	-10.12	-5.75
24	0.8	0.7	0.9	0.4	-0.08	-0.52	-2.47	-0.54
25	0.3	0.2	0.1	0.9	0.94	38.86	21.21	-1.08
26	0.8	0.2	0.1	0.9	0.08	8.48	2.90	0.87
27	0.3	0.7	0.1	0.9	0.11	-47.67	10.35	-2.45
28	0.8	0.7	0.1	0.9	-0.34	2.48	1.29	-0.87
29	0.3	0.2	0.4	0.9	-0.46	11.66	-4.68	-16.83
30	0.8	0.2	0.4	0.9	-0.22	2.55	-0.36	-1.68
31	0.3	0.7	0.4	0.9	-0.27	-6.82	-7.96	-7.75
32	0.8	0.7	0.4	0.9	0.04	-1.00	-0.86	-1.29
33	0.3	0.2	0.9	0.9	1.18	3.33	-30.90	-55.94
34	0.8	0.2	0.9	0.9	0.12	0.47	-1.45	-2.53
35	0.3	0.7	0.9	0.9	-0.74	-3.30	-20.09	-38.39
36	0.8	0.7	0.9	0.9	-0.16	-1.19	-0.85	-1.26

4. Simulations: Scenarios with heterogeneous detection probabilities

Second, we consider the situation where detection probabilities are heterogeneous depending on the status of the dyads. We define the scenarios we would like to investigate:

- scenarios on the detection probability: $p_A = 0.3$ and $p_B = 0.8$ vs. $p_A = 0.8$ and $p_B = 0.3$
- scenarios on π: 0.2, 0.7;

- scenarios on ψ_{AA}: 0.1, 0.4, 0.9;
- scenarios on ψ_{BB}: 0.1, 0.4, 0.9.

Therefore, in total we have 36 scenarios.

```
grid3 <- expand.grid(pi=c(0.2,0.7),psiAA=c(0.1,0.4,0.9),psiBB=c(0.1,0.4,0.9))
grid2 <- cbind(pA = 0.3, pB = 0.8, grid3)
grid1 <- cbind(pA = 0.8, pB = 0.3, grid3)
grid <- rbind(grid2,grid1)
grid
```

```
##     pA  pB  pi psiAA psiBB
## 1  0.3 0.8 0.2   0.1   0.1
## 2  0.3 0.8 0.7   0.1   0.1
## 3  0.3 0.8 0.2   0.4   0.1
## 4  0.3 0.8 0.7   0.4   0.1
## 5  0.3 0.8 0.2   0.9   0.1
## 6  0.3 0.8 0.7   0.9   0.1
## 7  0.3 0.8 0.2   0.1   0.4
## 8  0.3 0.8 0.7   0.1   0.4
## 9  0.3 0.8 0.2   0.4   0.4
## 10 0.3 0.8 0.7   0.4   0.4
## 11 0.3 0.8 0.2   0.9   0.4
## 12 0.3 0.8 0.7   0.9   0.4
## 13 0.3 0.8 0.2   0.1   0.9
## 14 0.3 0.8 0.7   0.1   0.9
## 15 0.3 0.8 0.2   0.4   0.9
## 16 0.3 0.8 0.7   0.4   0.9
## 17 0.3 0.8 0.2   0.9   0.9
## 18 0.3 0.8 0.7   0.9   0.9
## 19 0.8 0.3 0.2   0.1   0.1
## 20 0.8 0.3 0.7   0.1   0.1
## 21 0.8 0.3 0.2   0.4   0.1
## 22 0.8 0.3 0.7   0.4   0.1
## 23 0.8 0.3 0.2   0.9   0.1
## 24 0.8 0.3 0.7   0.9   0.1
## 25 0.8 0.3 0.2   0.1   0.4
## 26 0.8 0.3 0.7   0.1   0.4
## 27 0.8 0.3 0.2   0.4   0.4
## 28 0.8 0.3 0.7   0.4   0.4
## 29 0.8 0.3 0.2   0.9   0.4
## 30 0.8 0.3 0.7   0.9   0.4
## 31 0.8 0.3 0.2   0.1   0.9
## 32 0.8 0.3 0.7   0.1   0.9
## 33 0.8 0.3 0.2   0.4   0.9
## 34 0.8 0.3 0.7   0.4   0.9
## 35 0.8 0.3 0.2   0.9   0.9
## 36 0.8 0.3 0.7   0.9   0.9
```

Let us run the simulations, with 100 Monte Carlo iterations. These simulations take ages, we do not recommend to run them. For convenience, all the results are stored in the object `simul_network_index36_sim100heterogeneous.RData` that we provide.

```
# nb of monte carlo iterations
nb_simulations <- 100

# matrix to store results with estimated values for ppA, ppB, psiAA, psiBB,
pi
res <- array(NA,dim=c(nrow(grid),nb_simulations,5))

# run simulation
for (index in 1:nrow(grid)){ # go through grid of scenarios
        for (i in 1:nb_simulations){

# 1. simulate

ppA <- grid[index,1]
ppB <- grid[index,2]
pi <- grid[index,3]
psiAA <- grid[index,4]
psiBB <- grid[index,5]

sim_data <- sim_CRnetwork(J=5,n=105, ppA = ppA, ppB = ppB, psiAA = psiAA,
psiBB = psiBB, pi = pi)
dat <- sim_data[[1]]
states <- sim_data[[2]]

# 2. estimation

# initial values
init1 <-
list(psiAA=grid[index,3],ppA=grid[index,1],ppB=grid[index,2],x=states)
inits <- list(init1)

# data
jags.data <- list(obs = dat, n = nrow(dat), J = ncol(dat))

# nb iterations
ni <- 2000
# nb burn-in
nb <- 1000
# nb thin
nt <- 1
# nb chains
nc <- 1

# parameters to be monitored
parameters_sim <-
```

```
c("psiAA","psiBB","pi","ppA","ppB","x1","x2","x3","x4","x5")

# call JAGS from R
mod <- jags(jags.data, inits, parameters_sim, 'sim_network_het.txt', n.chains
= nc, n.thin = nt,
n.iter = ni, n.burnin = nb, working.directory = getwd())

res[index,i,1] <- mean(mod$BUGSoutput$sims.matrix[,'ppA']) # detection
res[index,i,2] <- mean(mod$BUGSoutput$sims.matrix[,'ppB']) # detection
res[index,i,3] <- mean(mod$BUGSoutput$sims.matrix[,'psiAA']) # associated
res[index,i,4] <- mean(mod$BUGSoutput$sims.matrix[,'psiBB']) # non-associated
res[index,i,5] <- mean(mod$BUGSoutput$sims.matrix[,'pi']) # prop of
associated

}

}
save(res,file='simul_network_index36_sim100heterogeneous.RData')
```

Let us post-process the results by computing relative bias (in percent) for all parameters:

```
load("simul_network_index36_sim100heterogeneous.RData")
bias_param <- matrix(NA,nrow(grid),5)
for(i in 1:nrow(grid)){
    for (j in 1:5){
        bias_param[i,j] <- (mean(res[i,,c(1,2,5,3,4)[j]]) -
grid[i,j])/grid[i,c(1,2,5,3,4)[j]]*100
    }
}
res_bias <- round(cbind(1:nrow(bias_param),grid,bias_param),2)
colnames(res_bias) <-
c('scenario',names(grid),'bias_ppA','bias_ppB','bias_pi','bias_psiAA','bias_p
siBB')
```

The results are given in the following table, with in the first column the scenarios labels, in columns 2-6 the simulation parameters and in columns 7-11 the relative bias:

```
knitr::kable(res_bias)
```

scenario	pA	pB	pi	psiAA	psiBB	bias_ppA	bias_ppB	bias_pi	bias_psiAA	bias_psiBB
1	0.3	0.8	0.2	0.1	0.1	0.59	-0.24	4.67	5.86	5.61
2	0.3	0.8	0.7	0.1	0.1	0.18	-0.39	-6.01	1.17	12.27
3	0.3	0.8	0.2	0.4	0.1	-0.16	-0.65	-2.13	-6.94	2.94
4	0.3	0.8	0.7	0.4	0.1	0.84	-0.49	-25.65	-2.16	4.77
5	0.3	0.8	0.2	0.9	0.1	-0.02	-2.53	-15.23	-6.76	1.87
6	0.3	0.8	0.7	0.9	0.1	0.67	-14.33	-131.51	-8.99	8.40
7	0.3	0.8	0.2	0.1	0.4	-0.11	0.34	2.82	9.96	6.86
8	0.3	0.8	0.7	0.1	0.4	1.55	0.30	-0.32	2.34	0.53

9	0.3	0.8	0.2	0.4	0.4	-0.65	-0.69	0.88	-2.29	2.30
10	0.3	0.8	0.7	0.4	0.4	1.41	-1.75	-3.50	-2.06	7.67
11	0.3	0.8	0.2	0.9	0.4	0.49	-0.06	2.86	-3.88	-0.25
12	0.3	0.8	0.7	0.9	0.4	0.81	-4.39	-11.31	-1.57	2.71
13	0.3	0.8	0.2	0.1	0.9	4.24	-0.21	0.26	27.14	-1.08
14	0.3	0.8	0.7	0.1	0.9	1.63	-0.22	-0.49	1.94	-2.15
15	0.3	0.8	0.2	0.4	0.9	7.33	-0.96	-1.02	-1.77	0.34
16	0.3	0.8	0.7	0.4	0.9	-0.05	-0.38	-1.43	-1.41	-0.78
17	0.3	0.8	0.2	0.9	0.9	-1.53	-0.24	0.43	-8.22	-0.36
18	0.3	0.8	0.7	0.9	0.9	0.84	-0.48	0.53	-1.08	-1.45
19	0.8	0.3	0.2	0.1	0.1	-0.39	-0.01	11.75	4.63	3.82
20	0.8	0.3	0.7	0.1	0.1	0.03	-0.63	4.72	1.48	7.67
21	0.8	0.3	0.2	0.4	0.1	0.47	0.73	16.29	3.33	3.73
22	0.8	0.3	0.7	0.4	0.1	0.52	-1.16	-0.14	1.50	5.51
23	0.8	0.3	0.2	0.9	0.1	0.00	0.04	17.21	-2.35	2.46
24	0.8	0.3	0.7	0.9	0.1	-0.13	1.98	-8.68	0.17	4.71
25	0.8	0.3	0.2	0.1	0.4	-0.94	0.38	2.54	11.96	-26.02
26	0.8	0.3	0.7	0.1	0.4	-0.79	-0.64	-0.89	1.91	-11.83
27	0.8	0.3	0.2	0.4	0.4	-0.91	-0.14	5.53	8.95	-3.96
28	0.8	0.3	0.7	0.4	0.4	-0.52	0.57	-0.87	1.32	-3.13
29	0.8	0.3	0.2	0.9	0.4	-0.48	1.01	2.13	-3.45	0.13
30	0.8	0.3	0.7	0.9	0.4	-0.48	0.58	-1.17	0.17	-1.46
31	0.8	0.3	0.2	0.1	0.9	-30.02	21.00	27.86	112.23	-196.71
32	0.8	0.3	0.7	0.1	0.9	-1.90	0.23	2.38	1.93	-8.34
33	0.8	0.3	0.2	0.4	0.9	-9.87	2.22	7.98	27.76	-6.73
34	0.8	0.3	0.7	0.4	0.9	-1.40	0.28	0.70	1.65	-1.85
35	0.8	0.3	0.2	0.9	0.9	0.14	-0.29	0.86	-8.69	-1.06
36	0.8	0.3	0.7	0.9	0.9	0.22	0.16	-0.41	-0.32	-1.45