Supporting information for "A multivariate framework for analyzing spatial ordinal survey-based data / A multivariate spatial model for ordinal survey-based data"

Beltrán-Sánchez, MA[1]; Martínez-Beneito, MA[2]; Corberán-Vallet, A[3]
[1]angel.beltran@uv.es; [2]miguel.a.martinez@uv.es; [3]ana.corberan@uv.es
Department of Statistics and Operations Research, University of Valencia, Spain

Required packages

```
pacman::p_load(foreign, readxl, faraway, spdep, sp, ggplot2,
               RColorBrewer, graphics, ggpubr, leaflet, nimble,
               ggmcmc, extraDistr, parallel, MCMCvis, gridExtra,
               corrplot, ggcorrplot, readr, lattice, install = FALSE)
```

Data loading: GHQ-12 items

```
rm(list = ls())
HSRV2022 <- read.spss(file.path("..", "data", "ESCV2022_UV_Matem.sav"),
                      use.value.labels = TRUE, to.data.frame = TRUE)
# Sample size
NResp <- nrow(HSRV2022)

# P8_1 Concentrate
HSRV2022$P8_1[HSRV2022$P8_1 == "NS/NC"] <- NA
HSRV2022$P8_1 <- factor(HSRV2022$P8_1, levels = c("Más que lo habitual", "Igual que lo habitual",
↪
                                                  "Menos que lo habitual", "Mucho menos que lo
                                                  ↪  habitual"))
levels(HSRV2022$P8_1) <- c("More", "Same", "Less", "Much less")
table(HSRV2022$P8_1)
```

```
##
##      More      Same      Less Much less
##       470      8195       919       194
```

```
P8_1 <- as.numeric(HSRV2022$P8_1)

# P8_2 Lose sleep over worries
HSRV2022$P8_2[HSRV2022$P8_2 == "NS/NC"] <- NA
```

```r
HSRV2022$P8_2 <- factor(HSRV2022$P8_2, levels = c("Más que lo habitual", "Igual que lo habitual",
↪
                                                  "Menos que lo habitual", "Mucho menos que lo
                                                  ↪  habitual"))
levels(HSRV2022$P8_2) <- c("Not at all", "No more", "Rather more", "Much more than usual")
table(HSRV2022$P8_2)
```

```
##
##           Not at all              No more          Rather more
##                 1706                 6490                 1094
## Much more than usual
##                  488
```

```r
P8_2 <- as.numeric(HSRV2022$P8_2)

# P8_3 Play a useful role
HSRV2022$P8_3[HSRV2022$P8_3 == "NS/NC"] <- NA
HSRV2022$P8_3 <- factor(HSRV2022$P8_3, levels = c("Más que lo habitual", "Igual que lo habitual",
↪
                                                  "Menos que lo habitual", "Mucho menos que lo
                                                  ↪  habitual"))
levels(HSRV2022$P8_3) <- c("More", "Same", "Less", "Much less")
table(HSRV2022$P8_3)
```

```
##
##      More      Same      Less Much less
##       823      8329       483       137
```

```r
P8_3 <- as.numeric(HSRV2022$P8_3)

# P8_4 Make decisions
HSRV2022$P8_4[HSRV2022$P8_4 == "NS/NC"] <- NA
HSRV2022$P8_4 <- factor(HSRV2022$P8_4, levels = c("Más que lo habitual", "Igual que lo habitual",
↪
                                                  "Menos que lo habitual", "Mucho menos que lo
                                                  ↪  habitual"))
levels(HSRV2022$P8_4) <- c("More", "Same", "Less", "Much less")
table(HSRV2022$P8_4)
```

```
##
##      More      Same      Less Much less
##       619      8572       446       140
```

```r
P8_4 <- as.numeric(HSRV2022$P8_4)

# P8_5 Constantly under strain
HSRV2022$P8_5[HSRV2022$P8_5 == "NS/NC"] <- NA
HSRV2022$P8_5 <- factor(HSRV2022$P8_5, levels = c("Más que lo habitual", "Igual que lo habitual",
↪
                                                  "Menos que lo habitual", "Mucho menos que lo
                                                  ↪  habitual"))
levels(HSRV2022$P8_5) <- c("Not at all", "No more", "Rather more", "Much more than usual")
table(HSRV2022$P8_5)
```

```
##
##           Not at all              No more          Rather more
##                 1621                 5385                 2175
## Much more than usual
##                  593
```

```
P8_5 <- as.numeric(HSRV2022$P8_5)

# P8_6 Unable to overcome difficulties
HSRV2022$P8_6[HSRV2022$P8_6 == "NS/NC"] <- NA
HSRV2022$P8_6 <- factor(HSRV2022$P8_6, levels = c("Más que lo habitual", "Igual que lo habitual",
↪
                                                  "Menos que lo habitual", "Mucho menos que lo
                                                  ↪  habitual"))
levels(HSRV2022$P8_6) <- c("Not at all", "No more", "Rather more", "Much more than usual")
table(HSRV2022$P8_6)
```

```
##
##           Not at all              No more          Rather more
##                 1173                 5748                 2258
## Much more than usual
##                  593
```

```
P8_6 <- as.numeric(HSRV2022$P8_6)

# P8_7 Enjoy activities
HSRV2022$P8_7[HSRV2022$P8_7 == "NS/NC"] <- NA
HSRV2022$P8_7 <- factor(HSRV2022$P8_7, levels = c("Más que lo habitual", "Igual que lo habitual",
↪
                                                  "Menos que lo habitual", "Mucho menos que lo
                                                  ↪  habitual"))
levels(HSRV2022$P8_7) <- c("More", "Same", "Less", "Much less")
table(HSRV2022$P8_7)
```

```
##
##      More      Same      Less Much less
##       624      8176       798       175
```

```
P8_7 <- as.numeric(HSRV2022$P8_7)

# P8_8 Face up to problems
HSRV2022$P8_8[HSRV2022$P8_8 == "NS/NC"] <- NA
HSRV2022$P8_8 <- factor(HSRV2022$P8_8, levels = c("Más que lo habitual", "Igual que lo habitual",
↪
                                                  "Menos que lo habitual", "Mucho menos que lo
                                                  ↪  habitual"))
levels(HSRV2022$P8_8) <- c("More", "Same", "Less", "Much less")
table(HSRV2022$P8_8)
```

```
##
##      More      Same      Less Much less
##       508      8459       624       185
```

```r
P8_8 <- as.numeric(HSRV2022$P8_8)

# P8_9 Feel depressed
HSRV2022$P8_9[HSRV2022$P8_9 == "NS/NC"] <- NA
HSRV2022$P8_9 <- factor(HSRV2022$P8_9, levels = c("Más que lo habitual", "Igual que lo habitual",
↪
                                                  "Menos que lo habitual", "Mucho menos que lo
                                                  ↪  habitual"))
levels(HSRV2022$P8_9) <- c("Not at all", "No more", "Rather more", "Much more than usual")
table(HSRV2022$P8_9)
```

```
##
##           Not at all              No more          Rather more
##                 1535                 4863                 2271
## Much more than usual
##                 1102
```

```r
P8_9 <- as.numeric(HSRV2022$P8_9)

# P8_10 Lose confidence
HSRV2022$P8_10[HSRV2022$P8_10 == "NS/NC"] <- NA
HSRV2022$P8_10 <- factor(HSRV2022$P8_10, levels = c("Más que lo habitual", "Igual que lo
↪  habitual",
                                                    "Menos que lo habitual", "Mucho menos que lo
                                                    ↪  habitual"))
levels(HSRV2022$P8_10) <- c("Not at all", "No more", "Rather more", "Much more than usual")
table(HSRV2022$P8_10)
```

```
##
##           Not at all              No more          Rather more
##                 1355                 4984                 2287
## Much more than usual
##                 1145
```

```r
P8_10 <- as.numeric(HSRV2022$P8_10)

# P8_11 Feel worthless
HSRV2022$P8_11[HSRV2022$P8_11 == "NS/NC"] <- NA
HSRV2022$P8_11 <- factor(HSRV2022$P8_11, levels = c("Más que lo habitual", "Igual que lo
↪  habitual",
                                                    "Menos que lo habitual", "Mucho menos que lo
                                                    ↪  habitual"))
levels(HSRV2022$P8_11) <- c("Not at all", "No more", "Rather more", "Much more than usual")
table(HSRV2022$P8_11)
```

```
##
##           Not at all              No more          Rather more
##                 1151                 5865                 1513
## Much more than usual
##                 1238
```

```r
P8_11 <- as.numeric(HSRV2022$P8_11)

# P8_12 Feel reasonably happy
HSRV2022$P8_12[HSRV2022$P8_12 == "NS/NC"] <- NA
HSRV2022$P8_12 <- factor(HSRV2022$P8_12, levels = c("Más que lo habitual", "Igual que lo
↪  habitual",
                                                    "Menos que lo habitual", "Mucho menos que lo
                                                    ↪  habitual"))
levels(HSRV2022$P8_12) <- c("More", "Same", "Less", "Much less")
table(HSRV2022$P8_12)
```

```
##
##       More       Same       Less Much less
##        754       8347        566        101
```

```r
P8_12 <- as.numeric(HSRV2022$P8_12)

y <- data.frame(P8_1, P8_2, P8_3, P8_4, P8_5, P8_6,
                P8_7, P8_8, P8_9, P8_10, P8_11, P8_12)

# Number of response variables
NVars <- ncol(y)
# Number of levels
NCats <- unique(apply(y, 2, function(x) {length(table(x))}))

ones <- rep(1, NCats)

rm(list = c("P8_1", "P8_2", "P8_3", "P8_4", "P8_5", "P8_6",
            "P8_7", "P8_8", "P8_9", "P8_10", "P8_11", "P8_12"))

# Covariate sex: 1 = Male; 2 = Female
sexC <- HSRV2022$sexo
levels(sexC) <- c("Male", "Female")
sex <- as.numeric(sexC)

# Covariate age group: 1 = [15,25); 2 = [25,35); 3 = [35,45); 4 = [45,55); 5 = [55,65);
# 6 = [65,70); 7 = [70,75); 8 = [75,...)
ageC <- cut(HSRV2022$Edad, breaks = c(15, 25, 35, 45, 55, 65, 70, 75, 103),
            include.lowest = TRUE, right = FALSE)
levels(ageC)[length(table(ageC))] <- "[75,...)"
age <- as.numeric(ageC)

# Number of respondents by sex and age group
table(sexC, ageC)
```

```
##         ageC
## sexC     [15,25) [25,35) [35,45) [45,55) [55,65) [65,70) [70,75) [75,...)
##   Male       467     504     671     815     655     507     477      830
##   Female     458     524     683     771     702     437     414      882
```

```r
# Number of levels of each (categorical) covariate
NSex <- length(table(sex))
NAges <- length(table(age))
```

```r
# Cartography of the Region of Valencia
load(file.path("..", "data", "CartoCV.Rdata"))
# Neighborhood structure by contiguity
cv.nb <- poly2nb(carto_muni)

# Some extra neighborhoods are added for Rincón de Ademuz comarca
cv.nb[[277]] <- as.integer(sort(c(cv.nb[[277]], 312, 317, 517, 523, 508)))
cv.nb[[363]] <- as.integer(sort(c(cv.nb[[363]], 312, 317, 517, 523, 508)))
cv.nb[[364]] <- as.integer(sort(c(cv.nb[[364]], 312, 317, 517, 523, 508)))
cv.nb[[477]] <- as.integer(sort(c(cv.nb[[477]], 312, 317, 517, 523, 508)))

cv.nb[[312]] <- as.integer(sort(c(cv.nb[[312]], 277, 363, 364, 477)))
cv.nb[[317]] <- as.integer(sort(c(cv.nb[[317]], 277, 363, 364, 477)))
cv.nb[[517]] <- as.integer(sort(c(cv.nb[[517]], 277, 363, 364, 477)))
cv.nb[[523]] <- as.integer(sort(c(cv.nb[[523]], 277, 363, 364, 477)))
cv.nb[[508]] <- as.integer(sort(c(cv.nb[[508]], 277, 363, 364, 477)))

# Municipality codes
INE_MUN <- as.numeric(as.character(carto_muni@data$INE_MUN))

# Municipality of each respondent
muni <- as.numeric(HSRV2022$LOCALIDAD)
# Number of (distinct) municipalities (542)
NMuni <- length(INE_MUN); rm(INE_MUN)

# Number of neighbors of each municipality
nadj <- card(cv.nb)
# Neighbors of each municipality
map <- unlist(cv.nb)
# Sum of all the neighbor numbers of all municipalities
nadj.tot <- length(map)
# Cumulative sums of the number of neighbors of each municipality
index <- c(0, cumsum(nadj))

# Diagonal matrix with the number of neighbors of each area
D <- diag(nadj)
# Adjacency matrix
W <- nb2mat(cv.nb, style = "B", zero.policy = TRUE)
# Eigenvalues of D-W
Lambda <- eigen(D - W)$values
# Identity matrix
I <- diag(rep(1, NMuni))

# All the neighborhoods j ~ i where i < j
from.to <- cbind(rep(1:NMuni, times = nadj), map); colnames(from.to) <- c("from", "to")
from.to <- from.to[which(from.to[, 1] < from.to[, 2]), ]
NDist <- nrow(from.to)

dcar_leroux <- nimbleFunction(
  name = 'dcar_leroux',
  run = function(x = double(1),         # Spatial random effect (vector)
                 rho = double(0),       # Amount of spatial dependence (scalar)
                 sd.theta = double(0), # Standard deviation (scalar)
                 Lambda = double(1),    # Eigenvalues of matrix D - W
                 from.to = double(2),  # Matrix of distinct pairs of neighbors from.to[, 1] <
                 ↪  from.to[, 2]
                 log = integer(0, default = 0)) {
    returnType(double(0))
```

```
    # Number of small areas
    NMuni <- dim(x)[1]
    # Number of distinct pairs of neighbors
    NDist <- dim(from.to)[1]
    # Required vectors
    x.from <- nimNumeric(NDist)
    x.to <- nimNumeric(NDist)
    for (Dist in 1:NDist) {
      x.from[Dist] <- x[from.to[Dist, 1]]
      x.to[Dist] <- x[from.to[Dist, 2]]
    }

    # Log-density
    logDens <- sum(dnorm(x[1:NMuni], mean = 0, sd = sd.theta * pow(1 - rho, -1/2), log = TRUE)) -
      NMuni/2 * log(1 - rho) + 1/2 * sum(log(rho * (Lambda[1:NMuni] - 1) + 1)) -
      1/2 * pow(sd.theta, -2) * rho * sum(pow(x.from[1:NDist] - x.to[1:NDist], 2))
    if(log) return(logDens)
    else return(exp(logDens))
  }
)
```

Proposed methods

```
# n.chains <- 5
# this_cluster <- makeCluster(n.chains)
```

Model-Indep

```
# Model code
modelCode <- nimbleCode(
  {
    for(Var in 1:NVars) {
      # Likelihood
      for (Resp in 1:NResp) {
        y[Resp, Var] ~ dcat(prlevels[Resp, Var, 1:NCats])

        # Definition of the probabilities of each category as a function of the
        # cumulative probabilities
        prlevels[Resp, Var, 1] <- p.gamma[Resp, Var, 1]
        for (Cat in 2:(NCats-1)) {
          prlevels[Resp, Var, Cat] <- p.gamma[Resp, Var, Cat] - p.gamma[Resp, Var, Cat-1]
        }
        prlevels[Resp, Var, NCats] <- 1 - p.gamma[Resp, Var, NCats-1]

        # Linear predictor
        for (Cat in 1:(NCats-1)) {
          logit(p.gamma[Resp, Var, Cat]) <- kappa[sex[Resp], age[Resp], Cat, Var] +
            sd.theta[Var] * theta[muni[Resp], Var]
        }
      }
    }
```

```
    # Prior distributions

    # kappa[1:NSex, 1:NAges, 1:(NCats-1), 1:NVars] cut points
    # Monotonic transformation
    for (Var in 1:NVars) {
      for (SexGroup in 1:NSex) {
        for (AgeGroup in 1:NAges) {
          for (Cat in 1:(NCats-1)) {
            kappa[SexGroup, AgeGroup, Cat, Var] <- logit(sum(delta[SexGroup, AgeGroup, Var,
↪   1:Cat]))
          }
          # delta[1:NSex, 1:NAges, 1:NVars, 1:NCats] Dirichlet prior
          delta[SexGroup, AgeGroup, Var, 1:NCats] ~ ddirch(ones[1:NCats])
        }
      }
    }

    # theta[1:NMuni, 1:NVars] spatial random effects
    for (Var in 1:NVars) {
      # theta[1:NMuni, 1:NVars] spatial random effects
      # LCAR distribution
      theta[1:NMuni, Var] ~ dcar_leroux(rho = rho[Var],
                                        sd.theta = 1,
                                        Lambda = Lambda[1:NMuni],
                                        from.to = from.to[1:NDist, 1:2])
    }

    # Hyperparameters of the spatial random effects
    for (Var in 1:NVars) {
      rho[Var] ~ dunif(0, 1)
      sd.theta[Var] ~ dhalfflat()
    }

    # Stochastic restrictions
    # Required vectors
    for (Var in 1:NVars) {
      for (Resp in 1:NResp) {
        theta.Resp[Resp, Var] <- theta[muni[Resp], Var]
      }

      # Zero-mean constraint for theta.Resp
      zero.theta.resp[Var] ~ dnorm(mean.thetas.resp[Var], 10000)
      mean.thetas.resp[Var] <- mean(theta.Resp[1:NResp, Var])
    }

  }
)

# Data to be loaded

modelData <- list(y = as.matrix(y), zero.theta.resp = rep(0, NVars))

modelConstants <- list(NResp = NResp, NCats = NCats, NVars = NVars, sex = sex,
                       age = age, muni = muni, NSex = NSex, NAges = NAges,
                       NMuni = NMuni, ones = ones, NDist = NDist, Lambda = Lambda,
                       from.to = from.to)

# Parameters to be saved
```

```
modelParameters <- c("kappa", "theta", "sd.theta", "rho",
                     "delta")

# Create a function with all the needed code
run_MCMC_allcode <- function(X, code, constants, data, monitors) {

  pacman::p_load(nimble, extraDistr, install = FALSE)

  dcar_leroux <- nimbleFunction(
    name = 'dcar_leroux',
    run = function(x = double(1),        # Spatial random effect (vector)
                   rho = double(0),      # Amount of spatial dependence (scalar)
                   sd.theta = double(0), # Standard deviation (scalar)
                   Lambda = double(1),   # Eigenvalues of matrix D - W
                   from.to = double(2),  # Matrix of distinct pairs of neighbors from.to[, 1] <
                   ↪  from.to[, 2]
                   log = integer(0, default = 0)) {
      returnType(double(0))

      # Number of small areas
      NMuni <- dim(x)[1]
      # Number of distinct pairs of neighbors
      NDist <- dim(from.to)[1]
      # Required vectors
      x.from <- nimNumeric(NDist)
      x.to <- nimNumeric(NDist)
      for (Dist in 1:NDist) {
        x.from[Dist] <- x[from.to[Dist, 1]]
        x.to[Dist] <- x[from.to[Dist, 2]]
      }

      logDens <- sum(dnorm(x[1:NMuni], mean = 0, sd = sd.theta * pow(1 - rho, -1/2), log = TRUE))
↪   -
        NMuni/2 * log(1 - rho) + 1/2 * sum(log(rho * (Lambda[1:NMuni] - 1) + 1)) -
        1/2 * pow(sd.theta, -2) * rho * sum(pow(x.from[1:NDist] - x.to[1:NDist], 2))
      if(log) return(logDens)
      else return(exp(logDens))
    }
  )

  rcar_leroux <- nimbleFunction(
    name = 'rcar_leroux',
    run = function(n = integer(0),
                   rho = double(0),
                   sd.theta = double(0),
                   Lambda = double(1),
                   from.to = double(2)) {
      returnType(double(1))

      nimStop("user-defined distribution dcar_leroux provided without random generation
      ↪  function.")
      x <- nimNumeric(542)
      return(x)
    }
  )

  assign('dcar_leroux', dcar_leroux, envir = .GlobalEnv)
  assign('rcar_leroux', rcar_leroux, envir = .GlobalEnv)
```

```
NSex <- constants$NSex
NAges <- constants$NAges
NCats <- constants$NCats
ones <- constants$ones
NResp <- constants$NResp
NMuni <- constants$NMuni
NVars <- constants$NVars

# Let's create the Nimble model, creates the nodes (inits should be passed now)
model <- nimbleModel(code = code,
                     constants = constants,
                     data = data,
                     inits = list(delta = array(rdirichlet(NSex * NAges * NVars, ones),
                                                dim = c(NSex, NAges, NVars, NCats)),
                                  rho = runif(NVars),
                                  theta = matrix(rnorm(NMuni * NVars, sd = 0.1), nrow = NMuni,
                                  ↪  ncol = NVars),
                                  sd.theta = runif(NVars)),
                     calculate = FALSE)

# Compile the model, which means generating C++ code, compiling that code, and loading it back
↪  into R
Cmodel <- compileNimble(model)

# model$getParents(model$getNodeNames(dataOnly = TRUE), stochOnly = TRUE)

# Configuration
modelMCMCconfiguration <- configureMCMC(model, useConjugacy = FALSE,
                                        enableWAIC = TRUE)

# Remove desire samplers
modelMCMCconfiguration$removeSamplers(c("theta", "rho", "sd.theta"))

# Add slice/RW-MH theta[1:NMuni, 1:NVars] samplers
thetas <- matrix(nrow = NMuni, ncol = NVars)
for (Var in 1:NVars) {
  for (Muni in 1:NMuni) {
    thetas[Muni, Var] <- paste0("theta[",Muni,",",Var,"]")
  }
}

smuni <- sort(unique(constants$muni))
for (Var in 1:NVars) {
  for (Muni in 1:NMuni) {
    ifelse(Muni %in% smuni,
           modelMCMCconfiguration$addSampler(target = thetas[Muni, Var], type = "RW"),
           modelMCMCconfiguration$addSampler(target = thetas[Muni, Var], type = "RW"))
  }
}

# Add slice rho sampler
rhos <- character(NVars)
for (Var in 1:NVars) {
  rhos[Var] <- paste0("rho[",Var,"]")
}

for (Var in 1:NVars) {
  modelMCMCconfiguration$addSampler(target = rhos[Var], type = "slice")
```

```
  }

  # Add slice sd.M.Muni sampler
  sd.thetas <- character(NVars)
  for (Var in 1:NVars) {
    sd.thetas[Var] <- paste0("sd.theta[",Var,"]")
  }

  for (Var in 1:NVars) {
    modelMCMCconfiguration$addSampler(target = sd.thetas[Var], type = "slice")
  }

  # Add new monitors
  modelMCMCconfiguration$monitors <- c()
  modelMCMCconfiguration$addMonitors(monitors)
  # Build MCMC object
  modelMCMC <- buildMCMC(modelMCMCconfiguration)
  # Need to reset the nimbleFunctions in order to add the new MCMC
  CmodelMCMC <- compileNimble(modelMCMC, project = model,
                              resetFunctions = TRUE)
  # Results
  results <- runMCMC(CmodelMCMC, niter = 8000, nburnin = 2000, thin = 30, setSeed = X)

  return(results)
}

# system.time(salnimble <- parLapply(cl = this_cluster, X = 1:n.chains,
#                                    fun = run_MCMC_allcode,
#                                    code = modelCode,
#                                    constants = modelConstants,
#                                    data = modelData,
#                                    monitors = modelParameters))
#
# # It's good practice to close the cluster when you're done with it.
# stopCluster(this_cluster)

# 1.84h on a server with: niter = 8000, nburnin = 2000, thin = 30
# saveRDS(salnimble, file = file.path("results", "multi-2022-nimble-MH-indep-8k-2k-30-WAIC.rds"))
salnimble1 <- readRDS(file = file.path("..", "results",
↪  "multi-2022-nimble-MH-indep-8k-2k-30-WAIC.rds"))

# WAIC for Model-Indep
# Let's create the Nimble model, creates the nodes
modelWAIC <- nimbleModel(code = modelCode,
                         constants = modelConstants,
                         data = modelData,
                         inits = list(delta = array(rdirichlet(NSex * NAges * NVars, ones),
                                                    dim = c(NSex, NAges, NVars, NCats)),
                                      rho = runif(NVars),
                                      theta = matrix(rnorm(NMuni * NVars, sd = 0.1), nrow =
                                      ↪  NMuni, ncol = NVars),
                                      sd.theta = runif(NVars)),
                         calculate = FALSE)
```

```
## Defining model

##   [Note] Registering 'dcar_leroux' as a distribution based on its use in BUGS code. If you make chan

##   [Warning] Random generation function for dcar_leroux is not available. NIMBLE is generating a plac
```

```
## Building model

## Setting data and initial values

## Checking model sizes and dimensions

##   [Warning] Possible size/dimension mismatch amongst vectors and matrices in BUGS expression: 'theta

##   [Note] This model is not fully initialized. This is not an error.
##          To see which variables are not initialized, use model$initializeInfo().
##          For more information on model initialization, see help(modelInitialization).
```

```
CmodelWAIC <- compileNimble(modelWAIC)         # calculateWAIC needs compiled model to exist
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
samples <- do.call(rbind, salnimble1)           # single matrix of samples
(waic <- calculateWAIC(samples, modelWAIC))
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.

## Calculating WAIC.

## nimbleList object of type waicNimbleList
## Field "WAIC":
## [1] 192525.4
## Field "lppd":
## [1] -94783.11
## Field "pWAIC":
## [1] 1479.595
```

Model-Corr

```
# Model code

modelCode <- nimbleCode(
  {
    for(Var in 1:NVars) {
      # Likelihood
      for (Resp in 1:NResp) {
        y[Resp, Var] ~ dcat(prlevels[Resp, Var, 1:NCats])

        # Definition of the probabilities of each category as a function of the
        # cumulative probabilities
```

```
      prlevels[Resp, Var, 1] <- p.gamma[Resp, Var, 1]
      for (Cat in 2:(NCats-1)) {
        prlevels[Resp, Var, Cat] <- p.gamma[Resp, Var, Cat] - p.gamma[Resp, Var, Cat-1]
      }
      prlevels[Resp, Var, NCats] <- 1 - p.gamma[Resp, Var, NCats-1]

      # Linear predictor
      for (Cat in 1:(NCats-1)) {
        logit(p.gamma[Resp, Var, Cat]) <- kappa[sex[Resp], age[Resp], Cat, Var] +
          theta[muni[Resp], Var]
      }
    }
  }

  # Prior distributions

  # kappa[1:NSex, 1:NAges, 1:(NCats-1), 1:NVars] cut points
  # Monotonic transformation
  for (Var in 1:NVars) {
    for (SexGroup in 1:NSex) {
      for (AgeGroup in 1:NAges) {
        for (Cat in 1:(NCats-1)) {
          kappa[SexGroup, AgeGroup, Cat, Var] <- logit(sum(delta[SexGroup, AgeGroup, Var,
↪ 1:Cat]))
        }
        # delta[1:NSex, 1:NAges, 1:NVars, 1:NCats] Dirichlet prior
        delta[SexGroup, AgeGroup, Var, 1:NCats] ~ ddirch(ones[1:NCats])
      }
    }
  }

  # theta[1:NMuni, 1:NVars] spatial random effects
  for (Var in 1:NVars) {
    for (Muni in 1:NMuni) {
      theta[Muni, Var] <- inprod(sub.Muni[Muni, ], M.Muni[, Var])
    }
    # sub.Muni[1:NMuni, 1:NVars] underlying spatial REs
    # LCAR distribution
    sub.Muni[1:NMuni, Var] ~ dcar_leroux(rho = rho[Var],
                                         sd.theta = 1,
                                         Lambda = Lambda[1:NMuni],
                                         from.to = from.to[1:NDist, 1:2])
  }

  # Hyperparameter of the spatial random effects
  for (Var in 1:NVars) {
    rho[Var] ~ dunif(0, 1)
  }

  # M.Resp[1:NVars, 1:NVars] and M.Muni[1:NVars, 1:NVars] M-matrices
  for (Var1 in 1:NVars) {
    for (Var2 in 1:NVars) {
      M.Muni[Var1, Var2] ~ dnorm(0, tau.M.Muni)
    }
  }

  # Prior for precisions of M.Muni
  tau.M.Muni <- pow(sd.M.Muni, -2)
  sd.M.Muni ~ dhalfflat()
```

```
    # Stochastic restrictions
    # Required vectors
    for (Var in 1:NVars) {
      for (Resp in 1:NResp) {
        sub.Muni.Resp[Resp, Var] <- sub.Muni[muni[Resp], Var]
      }

      # Zero-mean constraint for sub.Muni.Resp
      zero.sub.Muni.resp[Var] ~ dnorm(mean.sub.Munis.resp[Var], 10000)
      mean.sub.Munis.resp[Var] <- mean(sub.Muni.Resp[1:NResp, Var])
    }

  }
)

# Data to be loaded

modelData <- list(y = as.matrix(y), zero.sub.Muni.resp = rep(0, NVars))

modelConstants <- list(NResp = NResp, NCats = NCats, NVars = NVars, sex = sex,
                       age = age, muni = muni, NSex = NSex, NAges = NAges,
                       NMuni = NMuni, ones = ones, NDist = NDist, Lambda = Lambda,
                       from.to = from.to)

# Parameters to be saved

modelParameters <- c("kappa", "theta", "M.Muni", "sd.M.Muni", "rho",
                     "delta", "sub.Muni")

# Create a function with all the needed code
run_MCMC_allcode <- function(X, code, constants, data, monitors) {

  pacman::p_load(nimble, extraDistr, install = FALSE)

  dcar_leroux <- nimbleFunction(
    name = 'dcar_leroux',
    run = function(x = double(1),        # Spatial random effect (vector)
                   rho = double(0),      # Amount of spatial dependence (scalar)
                   sd.theta = double(0), # Standard deviation (scalar)
                   Lambda = double(1),   # Eigenvalues of matrix D - W
                   from.to = double(2),  # Matrix of distinct pairs of neighbors from.to[, 1] <
                   ↪  from.to[, 2]
                   log = integer(0, default = 0)) {
      returnType(double(0))

      # Number of small areas
      NMuni <- dim(x)[1]
      # Number of distinct pairs of neighbors
      NDist <- dim(from.to)[1]
      # Required vectors
      x.from <- nimNumeric(NDist)
      x.to <- nimNumeric(NDist)
      for (Dist in 1:NDist) {
        x.from[Dist] <- x[from.to[Dist, 1]]
        x.to[Dist] <- x[from.to[Dist, 2]]
      }

      logDens <- sum(dnorm(x[1:NMuni], mean = 0, sd = sd.theta * pow(1 - rho, -1/2), log = TRUE))
↪   -
```

```
      NMuni/2 * log(1 - rho) + 1/2 * sum(log(rho * (Lambda[1:NMuni] - 1) + 1)) -
      1/2 * pow(sd.theta, -2) * rho * sum(pow(x.from[1:NDist] - x.to[1:NDist], 2))
    if(log) return(logDens)
    else return(exp(logDens))
  }
)

rcar_leroux <- nimbleFunction(
  name = 'rcar_leroux',
  run = function(n = integer(0),
                 rho = double(0),
                 sd.theta = double(0),
                 Lambda = double(1),
                 from.to = double(2)) {
    returnType(double(1))

    nimStop("user-defined distribution dcar_leroux provided without random generation
    ↪  function.")
    x <- nimNumeric(542)
    return(x)
  }
)

assign('dcar_leroux', dcar_leroux, envir = .GlobalEnv)
assign('rcar_leroux', rcar_leroux, envir = .GlobalEnv)

NSex <- constants$NSex
NAges <- constants$NAges
NCats <- constants$NCats
ones <- constants$ones
NResp <- constants$NResp
NMuni <- constants$NMuni
NVars <- constants$NVars

# Let's create the Nimble model, creates the nodes (inits should be passed now)
model <- nimbleModel(code = code,
                     constants = constants,
                     data = data,
                     inits = list(delta = array(rdirichlet(NSex * NAges * NVars, ones),
                                                dim = c(NSex, NAges, NVars, NCats)),
                                  rho = runif(NVars),
                                  sub.Muni = matrix(rnorm(NMuni * NVars, sd = 0.01), nrow =
                                  ↪  NMuni, ncol = NVars),
                                  M.Muni = matrix(rnorm(NVars * NVars, sd = 0.5), ncol = NVars,
                                  ↪  nrow = NVars),
                                  sd.M.Muni = runif(1, min = 0.2, max = 0.8)),
                     calculate = FALSE)

# Compile the model, which means generating C++ code, compiling that code, and loading it back
↪  into R
Cmodel <- compileNimble(model)

# model$getParents(model$getNodeNames(dataOnly = TRUE), stochOnly = TRUE)

# Configuration
modelMCMCconfiguration <- configureMCMC(model, useConjugacy = FALSE,
                                        enableWAIC = TRUE)

# Remove desire samplers
```

```
  modelMCMCconfiguration$removeSamplers(c("sub.Muni", "rho", "sd.M.Muni"))

  # Add slice/RW-MH sub.Muni[1:NMuni, 1:NVars] samplers
  sub.Munis <- matrix(nrow = NMuni, ncol = NVars)
  for (Var in 1:NVars) {
    for (Muni in 1:NMuni) {
      sub.Munis[Muni, Var] <- paste0("sub.Muni[",Muni,",",Var,"]")
    }
  }

  smuni <- sort(unique(constants$muni))
  for (Var in 1:NVars) {
    for (Muni in 1:NMuni) {
      ifelse(Muni %in% smuni,
             modelMCMCconfiguration$addSampler(target = sub.Munis[Muni, Var], type = "RW"),
             modelMCMCconfiguration$addSampler(target = sub.Munis[Muni, Var], type = "RW"))
    }
  }

  # Add slice rho sampler
  rhos <- character(NVars)
  for (Var in 1:NVars) {
    rhos[Var] <- paste0("rho[",Var,"]")
  }

  for (Var in 1:NVars) {
    modelMCMCconfiguration$addSampler(target = rhos[Var], type = "slice")
  }

  # Add slice sd.M.Muni sampler
  modelMCMCconfiguration$addSampler(target = "sd.M.Muni", type = "slice")

  # Add new monitors
  modelMCMCconfiguration$monitors <- c()
  modelMCMCconfiguration$addMonitors(monitors)
  # Build MCMC object
  modelMCMC <- buildMCMC(modelMCMCconfiguration)
  # Need to reset the nimbleFunctions in order to add the new MCMC
  CmodelMCMC <- compileNimble(modelMCMC, project = model,
                              resetFunctions = TRUE)
  # Results
  results <- runMCMC(CmodelMCMC, niter = 8000, nburnin = 2000, thin = 30, setSeed = X)

  return(results)
}

# system.time(salnimble <- parLapply(cl = this_cluster, X = 1:n.chains,
#                                    fun = run_MCMC_allcode,
#                                    code = modelCode,
#                                    constants = modelConstants,
#                                    data = modelData,
#                                    monitors = modelParameters))
#
# # It's good practice to close the cluster when you're done with it.
# stopCluster(this_cluster)

# 6.18h on a server with: niter = 8000, nburnin = 2000, thin = 30
# saveRDS(salnimble, file = file.path("results", "multi-2022-nimble-MH-corr-8k-2k-30-WAIC.rds"))
salnimble2 <- readRDS(file = file.path("..", "results",
↪  "multi-2022-nimble-MH-corr-8k-2k-30-WAIC.rds"))
```

```
# WAIC for Model-Corr
# Let's create the Nimble model, creates the nodes
modelWAIC <- nimbleModel(code = modelCode,
                         constants = modelConstants,
                         data = modelData,
                         inits = list(delta = array(rdirichlet(NSex * NAges * NVars, ones),
                                                    dim = c(NSex, NAges, NVars, NCats)),
                                      rho = runif(NVars),
                                      sub.Muni = matrix(rnorm(NMuni * NVars, sd = 0.01), nrow =
                                      ↪  NMuni, ncol = NVars),
                                      M.Muni = matrix(rnorm(NVars * NVars, sd = 0.5), ncol =
                                      ↪  NVars, nrow = NVars),
                                      sd.M.Muni = runif(1, min = 0.2, max = 0.8)),
                         calculate = FALSE)
```

```
## Defining model

## Building model

## Setting data and initial values

## Checking model sizes and dimensions

##   [Warning] Possible size/dimension mismatch amongst vectors and matrices in BUGS expression: 'sub.Mu

##   [Note] This model is not fully initialized. This is not an error.
##          To see which variables are not initialized, use model$initializeInfo().
##          For more information on model initialization, see help(modelInitialization).
```

```
CmodelWAIC <- compileNimble(modelWAIC)         # calculateWAIC needs compiled model to exist
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
samples <- do.call(rbind, salnimble2)           # single matrix of samples
(waic <- calculateWAIC(samples, modelWAIC))
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.

## Calculating WAIC.

## nimbleList object of type waicNimbleList
## Field "WAIC":
## [1] 190447.6
## Field "lppd":
## [1] -93940.28
## Field "pWAIC":
## [1] 1283.548
```

Model-Corr&IRE

```
# Model code

modelCode <- nimbleCode(
  {
    for(Var in 1:NVars) {
      # Likelihood
      for (Resp in 1:NResp) {
        y[Resp, Var] ~ dcat(prlevels[Resp, Var, 1:NCats])

        # Definition of the probabilities of each category as a function of the
        # cumulative probabilities
        prlevels[Resp, Var, 1] <- p.gamma[Resp, Var, 1]
        for (Cat in 2:(NCats-1)) {
          prlevels[Resp, Var, Cat] <- p.gamma[Resp, Var, Cat] - p.gamma[Resp, Var, Cat-1]
        }
        prlevels[Resp, Var, NCats] <- 1 - p.gamma[Resp, Var, NCats-1]

        # Linear predictor
        for (Cat in 1:(NCats-1)) {
          logit(p.gamma[Resp, Var, Cat]) <- kappa[sex[Resp], age[Resp], Cat, Var] +
            theta[muni[Resp], Var] + psi[Resp, Var]
        }
      }
    }

    # Prior distributions

    # kappa[1:NSex, 1:NAges, 1:(NCats-1), 1:NVars] cut points
    # Monotonic transformation
    for (Var in 1:NVars) {
      for (SexGroup in 1:NSex) {
        for (AgeGroup in 1:NAges) {
          for (Cat in 1:(NCats-1)) {
            kappa[SexGroup, AgeGroup, Cat, Var] <- logit(sum(delta[SexGroup, AgeGroup, Var,
↪  1:Cat]))
          }
          # delta[1:NSex, 1:NAges, 1:NVars, 1:NCats] Dirichlet prior
          delta[SexGroup, AgeGroup, Var, 1:NCats] ~ ddirch(ones[1:NCats])
        }
      }
    }

    # psi[1:NResp, 1:NVars] individual random effects
    for(Var in 1:NVars) {
      for (Resp in 1:NResp) {
        psi[Resp, Var] <- inprod(sub.Resp[Resp, ], M.Resp[, Var])
        # sub.Resp[1:NResp, 1:NVars] underlying individual REs
        sub.Resp[Resp, Var] ~ dnorm(0, 1)
      }
    }

    # theta[1:NMuni, 1:NVars] spatial random effects
    for (Var in 1:NVars) {
      for (Muni in 1:NMuni) {
        theta[Muni, Var] <- inprod(sub.Muni[Muni, ], M.Muni[, Var])
      }
```

```
      # sub.Muni[1:NMuni, 1:NVars] underlying spatial REs
      # LCAR distribution
      sub.Muni[1:NMuni, Var] ~ dcar_leroux(rho = rho[Var],
                                           sd.theta = 1,
                                           Lambda = Lambda[1:NMuni],
                                           from.to = from.to[1:NDist, 1:2])
    }

    # Hyperparameter of the spatial random effects
    for (Var in 1:NVars) {
      rho[Var] ~ dunif(0, 1)
    }

    # M.Resp[1:NVars, 1:NVars] and M.Muni[1:NVars, 1:NVars] M-matrices
    for (Var1 in 1:NVars) {
      for (Var2 in 1:NVars) {
        M.Resp[Var1, Var2] ~ dnorm(0, tau.M.Resp)
        M.Muni[Var1, Var2] ~ dnorm(0, tau.M.Muni)
      }
    }

    # Prior for precisions of M.Resp and M.Muni
    tau.M.Resp <- pow(sd.M.Resp, -2)
    sd.M.Resp ~ dhalfflat()

    tau.M.Muni <- pow(sd.M.Muni, -2)
    sd.M.Muni ~ dhalfflat()

    # Stochastic restrictions
    # Required vectors
    for (Var in 1:NVars) {
      for (Resp in 1:NResp) {
        sub.Muni.Resp[Resp, Var] <- sub.Muni[muni[Resp], Var]
      }

      # Zero-mean constraint for sub.Muni.Resp
      zero.sub.Muni.resp[Var] ~ dnorm(mean.sub.Munis.resp[Var], 10000)
      mean.sub.Munis.resp[Var] <- mean(sub.Muni.Resp[1:NResp, Var])
    }

  }
)

# Data to be loaded

modelData <- list(y = as.matrix(y), zero.sub.Muni.resp = rep(0, NVars))

modelConstants <- list(NResp = NResp, NCats = NCats, NVars = NVars, sex = sex,
                       age = age, muni = muni, NSex = NSex, NAges = NAges,
                       NMuni = NMuni, ones = ones, NDist = NDist, Lambda = Lambda,
                       from.to = from.to)

# Parameters to be saved

modelParameters <- c("kappa", "theta", "M.Muni", "rho",
                     "sd.M.Muni", "psi", "M.Resp", "sd.M.Resp",
                     "delta", "sub.Muni", "sub.Resp")

# Create a function with all the needed code
```

```
run_MCMC_allcode <- function(X, code, constants, data, monitors) {

  pacman::p_load(nimble, extraDistr, install = FALSE)

  dcar_leroux <- nimbleFunction(
    name = 'dcar_leroux',
    run = function(x = double(1),        # Spatial random effect (vector)
                   rho = double(0),      # Amount of spatial dependence (scalar)
                   sd.theta = double(0), # Standard deviation (scalar)
                   Lambda = double(1),   # Eigenvalues of matrix D - W
                   from.to = double(2),  # Matrix of distinct pairs of neighbors from.to[, 1] <
                   ↪  from.to[, 2]
                   log = integer(0, default = 0)) {
      returnType(double(0))

      # Number of small areas
      NMuni <- dim(x)[1]
      # Number of distinct pairs of neighbors
      NDist <- dim(from.to)[1]
      # Required vectors
      x.from <- nimNumeric(NDist)
      x.to <- nimNumeric(NDist)
      for (Dist in 1:NDist) {
        x.from[Dist] <- x[from.to[Dist, 1]]
        x.to[Dist] <- x[from.to[Dist, 2]]
      }

      logDens <- sum(dnorm(x[1:NMuni], mean = 0, sd = sd.theta * pow(1 - rho, -1/2), log = TRUE))
↪   -
        NMuni/2 * log(1 - rho) + 1/2 * sum(log(rho * (Lambda[1:NMuni] - 1) + 1)) -
        1/2 * pow(sd.theta, -2) * rho * sum(pow(x.from[1:NDist] - x.to[1:NDist], 2))
      if(log) return(logDens)
      else return(exp(logDens))
    }
  )

  rcar_leroux <- nimbleFunction(
    name = 'rcar_leroux',
    run = function(n = integer(0),
                   rho = double(0),
                   sd.theta = double(0),
                   Lambda = double(1),
                   from.to = double(2)) {
      returnType(double(1))

      nimStop("user-defined distribution dcar_leroux provided without random generation
      ↪  function.")
      x <- nimNumeric(542)
      return(x)
    }
  )

  assign('dcar_leroux', dcar_leroux, envir = .GlobalEnv)
  assign('rcar_leroux', rcar_leroux, envir = .GlobalEnv)

  NSex <- constants$NSex
  NAges <- constants$NAges
  NCats <- constants$NCats
  ones <- constants$ones
```

```r
NResp <- constants$NResp
NMuni <- constants$NMuni
NVars <- constants$NVars

# Let's create the Nimble model, creates the nodes (inits should be passed now)
model <- nimbleModel(code = code,
                     constants = constants,
                     data = data,
                     inits = list(delta = array(rdirichlet(NSex * NAges * NVars, ones),
                                                dim = c(NSex, NAges, NVars, NCats)),
                                  rho = runif(NVars),
                                  sub.Resp = matrix(rnorm(NResp * NVars, sd = 0.01), nrow =
                                  ↪  NResp, ncol = NVars),
                                  M.Resp = matrix(rnorm(NVars * NVars, sd = 0.5), ncol = NVars,
                                  ↪  nrow = NVars),
                                  sd.M.Resp = runif(1, min = 0.2, max = 0.8),
                                  sub.Muni = matrix(rnorm(NMuni * NVars, sd = 0.01), nrow =
                                  ↪  NMuni, ncol = NVars),
                                  M.Muni = matrix(rnorm(NVars * NVars, sd = 0.5), ncol = NVars,
                                  ↪  nrow = NVars),
                                  sd.M.Muni = runif(1, min = 0.2, max = 0.8)),
                     calculate = FALSE)

# Compile the model, which means generating C++ code, compiling that code, and loading it back
↪  into R
Cmodel <- compileNimble(model)

# model$getParents(model$getNodeNames(dataOnly = TRUE), stochOnly = TRUE)

# Configuration
modelMCMCconfiguration <- configureMCMC(model, useConjugacy = FALSE,
                                        enableWAIC = TRUE)

# Remove desire samplers
modelMCMCconfiguration$removeSamplers(c("sub.Muni", "rho", "sd.M.Muni",
                                        "sd.M.Resp"))

# Add slice/RW-MH sub.Muni[1:NMuni, 1:NVars] samplers
sub.Munis <- matrix(nrow = NMuni, ncol = NVars)
for (Var in 1:NVars) {
  for (Muni in 1:NMuni) {
    sub.Munis[Muni, Var] <- paste0("sub.Muni[",Muni,",",Var,"]")
  }
}

smuni <- sort(unique(constants$muni))
for (Var in 1:NVars) {
  for (Muni in 1:NMuni) {
    ifelse(Muni %in% smuni,
           modelMCMCconfiguration$addSampler(target = sub.Munis[Muni, Var], type = "RW"),
           modelMCMCconfiguration$addSampler(target = sub.Munis[Muni, Var], type = "RW"))
  }
}

# Add slice rho sampler
rhos <- character(NVars)
for (Var in 1:NVars) {
  rhos[Var] <- paste0("rho[",Var,"]")
}
```

```
  for (Var in 1:NVars) {
    modelMCMCconfiguration$addSampler(target = rhos[Var], type = "slice")
  }

  # Add slice sd.M.Muni and sd.M.Resp samplers
  modelMCMCconfiguration$addSampler(target = "sd.M.Resp", type = "slice")
  modelMCMCconfiguration$addSampler(target = "sd.M.Muni", type = "slice")

  # Add new monitors
  modelMCMCconfiguration$monitors <- c()
  modelMCMCconfiguration$addMonitors(monitors)
  # Build MCMC object
  modelMCMC <- buildMCMC(modelMCMCconfiguration)
  # Need to reset the nimbleFunctions in order to add the new MCMC
  CmodelMCMC <- compileNimble(modelMCMC, project = model,
                              resetFunctions = TRUE)
  # Results
  results <- runMCMC(CmodelMCMC, niter = 8000, nburnin = 2000, thin = 30, setSeed = X)

  return(results)
}

# system.time(salnimble <- parLapply(cl = this_cluster, X = 1:n.chains,
#                                    fun = run_MCMC_allcode,
#                                    code = modelCode,
#                                    constants = modelConstants,
#                                    data = modelData,
#                                    monitors = modelParameters))
#
# # It's good practice to close the cluster when you're done with it.
# stopCluster(this_cluster)

# 14.42h on a server with: niter = 8000, nburnin = 2000, thin = 30
# saveRDS(salnimble, file = file.path("results",
↪  "multi-2022-nimble-MH-corr-ire-8k-2k-30-WAIC.rds"))
salnimble3 <- readRDS(file = file.path("..", "results",
↪  "multi-2022-nimble-MH-corr-ire-8k-2k-30-WAIC.rds"))

# WAIC for Model-Corr&IRE
# Let's create the Nimble model, creates the nodes
modelWAIC <- nimbleModel(code = modelCode,
                         constants = modelConstants,
                         data = modelData,
                         inits = list(delta = array(rdirichlet(NSex * NAges * NVars, ones),
                                                    dim = c(NSex, NAges, NVars, NCats)),
                                      rho = runif(NVars),
                                      sub.Resp = matrix(rnorm(NResp * NVars, sd = 0.01), nrow =
                                      ↪  NResp, ncol = NVars),
                                      M.Resp = matrix(rnorm(NVars * NVars, sd = 0.5), ncol =
                                      ↪  NVars, nrow = NVars),
                                      sd.M.Resp = runif(1, min = 0.2, max = 0.8),
                                      sub.Muni = matrix(rnorm(NMuni * NVars, sd = 0.01), nrow =
                                      ↪  NMuni, ncol = NVars),
                                      M.Muni = matrix(rnorm(NVars * NVars, sd = 0.5), ncol =
                                      ↪  NVars, nrow = NVars),
                                      sd.M.Muni = runif(1, min = 0.2, max = 0.8)),
                         calculate = FALSE)
```

```
## Defining model

## Building model

## Setting data and initial values

## Checking model sizes and dimensions

##   [Warning] Possible size/dimension mismatch amongst vectors and matrices in BUGS expression: 'sub.Mu

##   [Note] This model is not fully initialized. This is not an error.
##          To see which variables are not initialized, use model$initializeInfo().
##          For more information on model initialization, see help(modelInitialization).
```

```
CmodelWAIC <- compileNimble(modelWAIC)         # calculateWAIC needs compiled model to exist
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
samples <- do.call(rbind, salnimble3)           # single matrix of samples
(waic <- calculateWAIC(samples, modelWAIC))
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.

## Calculating WAIC.

## nimbleList object of type waicNimbleList
## Field "WAIC":
## [1] 103170.1
## Field "lppd":
## [1] -36079.51
## Field "pWAIC":
## [1] 15505.53
```

Model results

```
n.chains <- 5
labels <- c("Q1", "Q2", "Q3", "Q4", "Q5", "Q6", "Q7", "Q8", "Q9", "Q10", "Q11", "Q12")
```

Convergence assessment for Model-Indep

```r
# Function: salnimble to salwinbugs for Model-Indep

NimToWin <- function(salnimble) {

  n.chains <- length(salnimble)
  n.sims <- n.chains * nrow(salnimble[[1]])

  kappa <- array(dim = c(n.sims, NSex, NAges, NCats, NVars))
  theta <- array(dim = c(n.sims, NMuni, NVars))
  rho <- matrix(nrow = n.sims, ncol = NVars)
  sd.theta <- matrix(nrow = n.sims, ncol = NVars)

  for (Var in 1:NVars) {
    for (Cat in 1:(NCats - 1)) {
      for (Sex in 1:NSex) {
        for (Age in 1:NAges) {
          kappa[, Sex, Age, Cat, Var] <- c(salnimble[[1]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")],
                                           salnimble[[2]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")],
                                           salnimble[[3]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")],
                                           salnimble[[4]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")],
                                           salnimble[[5]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")])
        }
      }
    }
  }

  for (Var in 1:NVars) {
    for (Muni in 1:NMuni) {
      theta[, Muni, Var] <- c(salnimble[[1]][, paste0("theta[", Muni, ", ", Var, "]")],
                              salnimble[[2]][, paste0("theta[", Muni, ", ", Var, "]")],
                              salnimble[[3]][, paste0("theta[", Muni, ", ", Var, "]")],
                              salnimble[[4]][, paste0("theta[", Muni, ", ", Var, "]")],
                              salnimble[[5]][, paste0("theta[", Muni, ", ", Var, "]")])
    }
  }

  for (Var in 1:NVars) {
    rho[, Var] <- c(salnimble[[1]][, paste0("rho[", Var, "]")],
                    salnimble[[2]][, paste0("rho[", Var, "]")],
                    salnimble[[3]][, paste0("rho[", Var, "]")],
                    salnimble[[4]][, paste0("rho[", Var, "]")],
                    salnimble[[5]][, paste0("rho[", Var, "]")])
  }

  for (Var in 1:NVars) {
    sd.theta[, Var] <- c(salnimble[[1]][, paste0("sd.theta[", Var, "]")],
                         salnimble[[2]][, paste0("sd.theta[", Var, "]")],
                         salnimble[[3]][, paste0("sd.theta[", Var, "]")],
                         salnimble[[4]][, paste0("sd.theta[", Var, "]")],
                         salnimble[[5]][, paste0("sd.theta[", Var, "]")])
  }

  summary <- MCMCsummary(object = salnimble, round = 4)
```

```
  # summary <- "not available"
  sims.list <- list("kappa" = kappa, "theta" = theta,
                    "sd.theta" = sd.theta, "rho" = rho)

  salwinbugs <- list("summary" = summary, "sims.list" = sims.list,
                     "n.chains" = n.chains, "n.sims" = n.sims)

  return(salwinbugs)
}

# salwinbugs1 <- NimToWin(salnimble = salnimble1)
# saveRDS(salwinbugs1, file = file.path("results", "salwinbugs1.rds"))
salwinbugs1 <- readRDS(file = file.path("..", "results", "salwinbugs1.rds"))

which((salwinbugs1$summary[startsWith(labels(salwinbugs1$summary)[[1]], "kappa"), 6] > 1.1) |
↪  (salwinbugs1$summary[startsWith(labels(salwinbugs1$summary)[[1]], "kappa"), 7] < 100))
```

```
## integer(0)
```

```
which((salwinbugs1$summary[startsWith(labels(salwinbugs1$summary)[[1]], "theta"), 6] > 1.1) |
↪  (salwinbugs1$summary[startsWith(labels(salwinbugs1$summary)[[1]], "theta"), 7] < 100))
```

```
## integer(0)
```

```
which((salwinbugs1$summary[startsWith(labels(salwinbugs1$summary)[[1]], "sd.theta"), 6] > 1.1) |
↪  (salwinbugs1$summary[startsWith(labels(salwinbugs1$summary)[[1]], "sd.theta"), 7] < 100))
```

```
## integer(0)
```

```
which((salwinbugs1$summary[startsWith(labels(salwinbugs1$summary)[[1]], "rho"), 6] > 1.1) |
↪  (salwinbugs1$summary[startsWith(labels(salwinbugs1$summary)[[1]], "rho"), 7] < 100))
```

```
## [1]  1 12
```

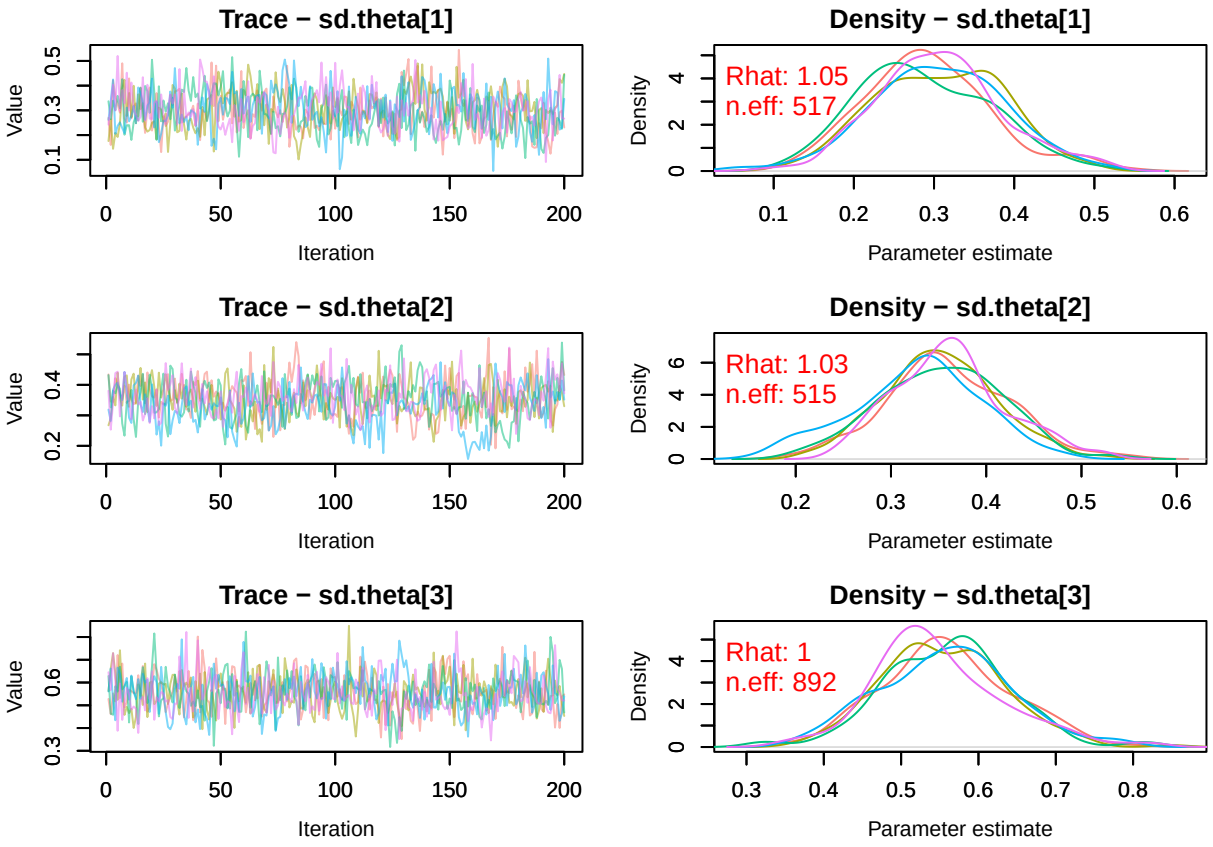
```
MCMCsummary(object = salnimble1, params = "kappa",
            # exact = TRUE,
            # ISB = FALSE,
            round = 4)[1:10, ]
```

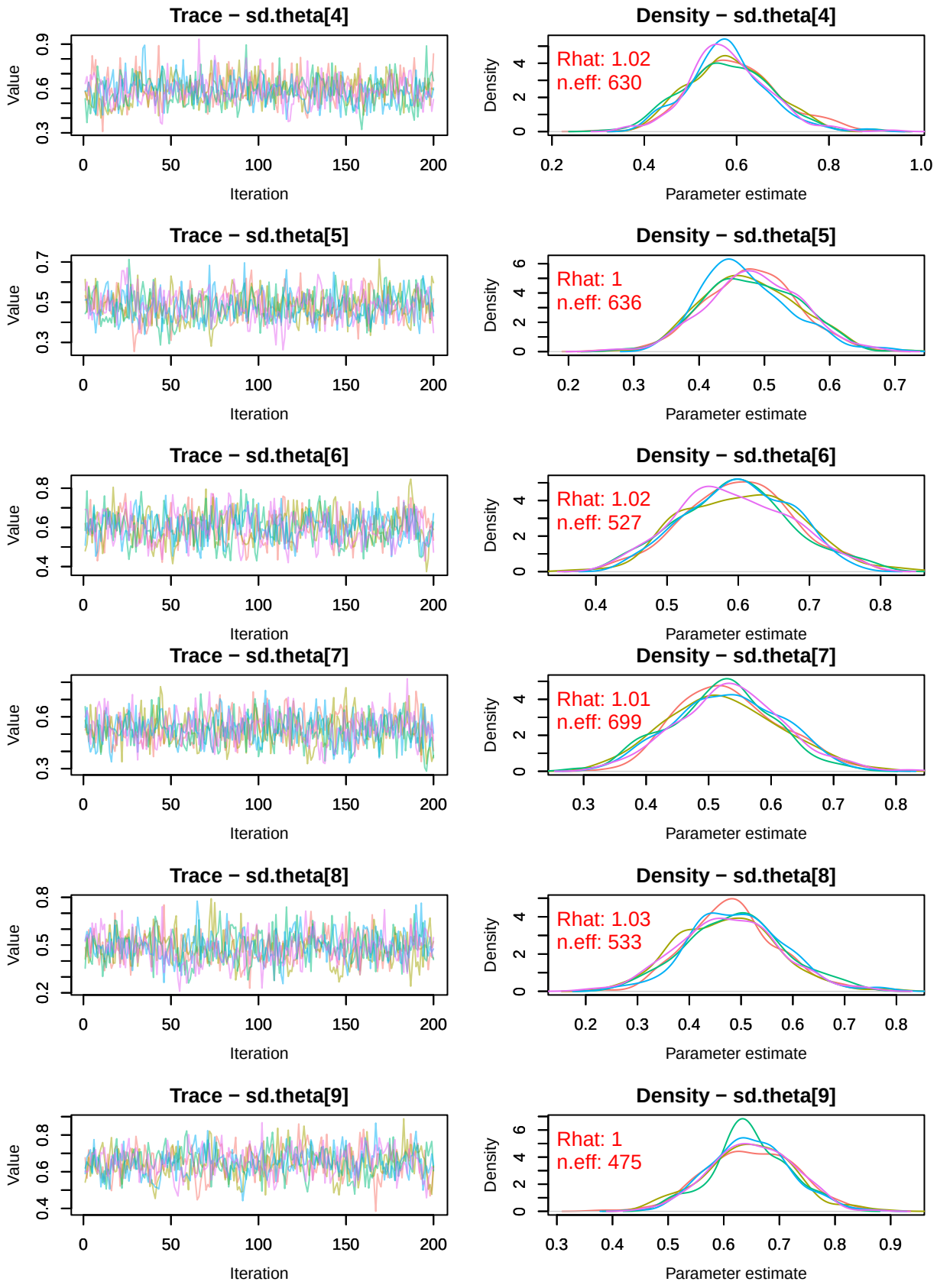
```
##                       mean     sd    2.5%     50%   97.5% Rhat n.eff
## kappa[1, 1, 1, 1] -2.5754 0.1739 -2.9327 -2.5699 -2.2494 1.00  1174
## kappa[2, 1, 1, 1] -2.3844 0.1686 -2.7289 -2.3794 -2.0781 1.00  1208
## kappa[1, 2, 1, 1] -2.4628 0.1665 -2.7966 -2.4550 -2.1498 1.00   989
## kappa[2, 2, 1, 1] -2.5266 0.1589 -2.8632 -2.5188 -2.2301 1.00  1000
## kappa[1, 3, 1, 1] -2.7292 0.1533 -3.0197 -2.7199 -2.4244 1.01  1000
## kappa[2, 3, 1, 1] -2.4652 0.1396 -2.7382 -2.4634 -2.1960 1.01   960
## kappa[1, 4, 1, 1] -3.2979 0.1852 -3.6819 -3.2894 -2.9725 1.00  1000
## kappa[2, 4, 1, 1] -2.8549 0.1541 -3.1922 -2.8493 -2.5666 1.00   780
## kappa[1, 5, 1, 1] -3.1442 0.1983 -3.5294 -3.1397 -2.7843 1.01   892
## kappa[2, 5, 1, 1] -3.1827 0.1937 -3.5774 -3.1806 -2.8277 1.01  2756
```

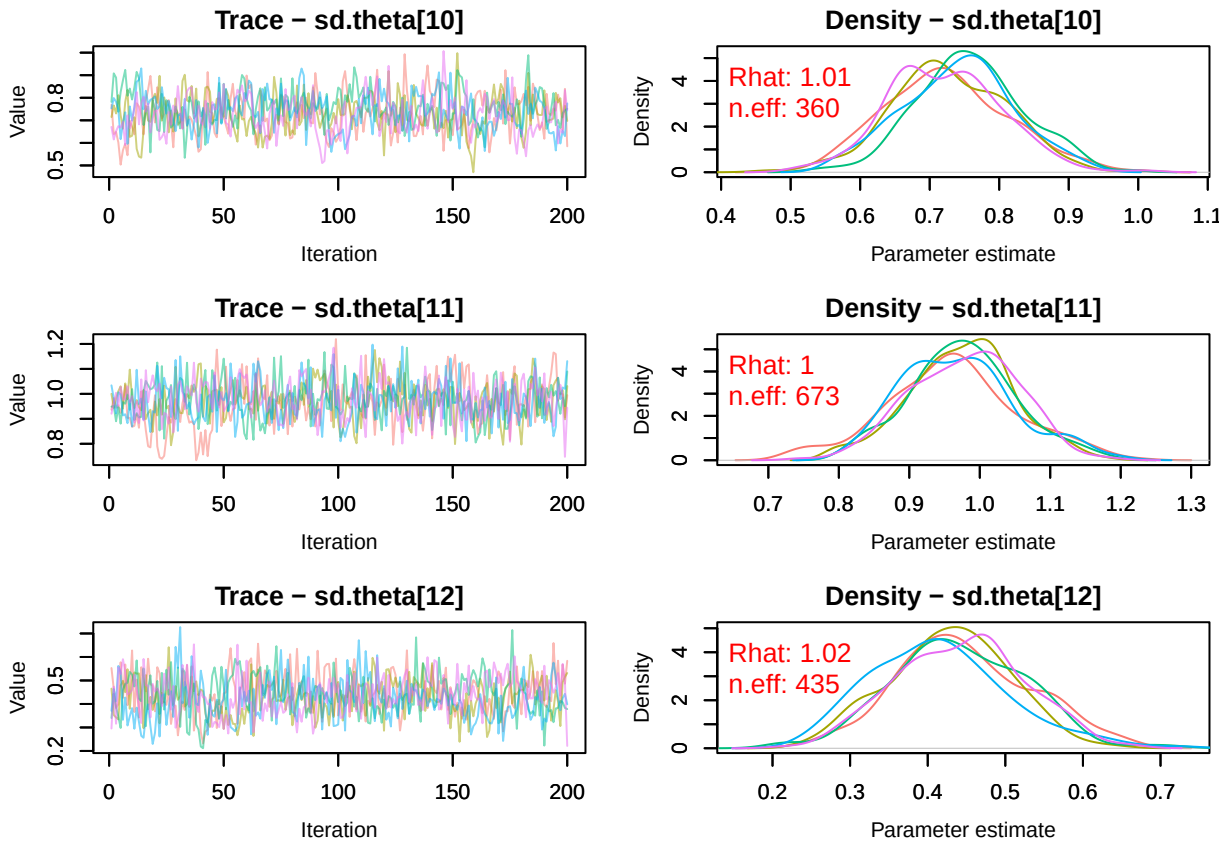
```
MCMCsummary(object = salnimble1, params = "theta",
            # exact = TRUE,
            # ISB = FALSE,
            round = 4)[1:10, ]
```

```
##                  mean     sd    2.5%     50%  97.5% Rhat n.eff
## theta[1, 1]   -0.0439 0.6895 -1.3864 -0.0467 1.3720 1.01  1075
## theta[2, 1]    0.0441 0.6638 -1.2400  0.0196 1.4005 1.00  1138
## theta[3, 1]   -0.1012 0.6461 -1.2719 -0.1251 1.1998 1.01  1003
## theta[4, 1]    0.3868 0.6811 -1.0585  0.4176 1.5716 1.01   953
## theta[5, 1]   -0.2265 0.5726 -1.3945 -0.2280 0.9430 1.02  1077
## theta[6, 1]    0.0715 0.5737 -1.0723  0.0555 1.1740 1.00  1000
## theta[7, 1]   -0.0266 0.7226 -1.4569 -0.0339 1.3666 1.00   953
## theta[8, 1]    0.0445 0.7412 -1.4635  0.0331 1.5323 1.02  1055
## theta[9, 1]    0.3294 0.4231 -0.4975  0.3266 1.1545 1.02  1012
## theta[10, 1] -0.1043 0.7396 -1.5696 -0.1314 1.3917 1.00   956
```

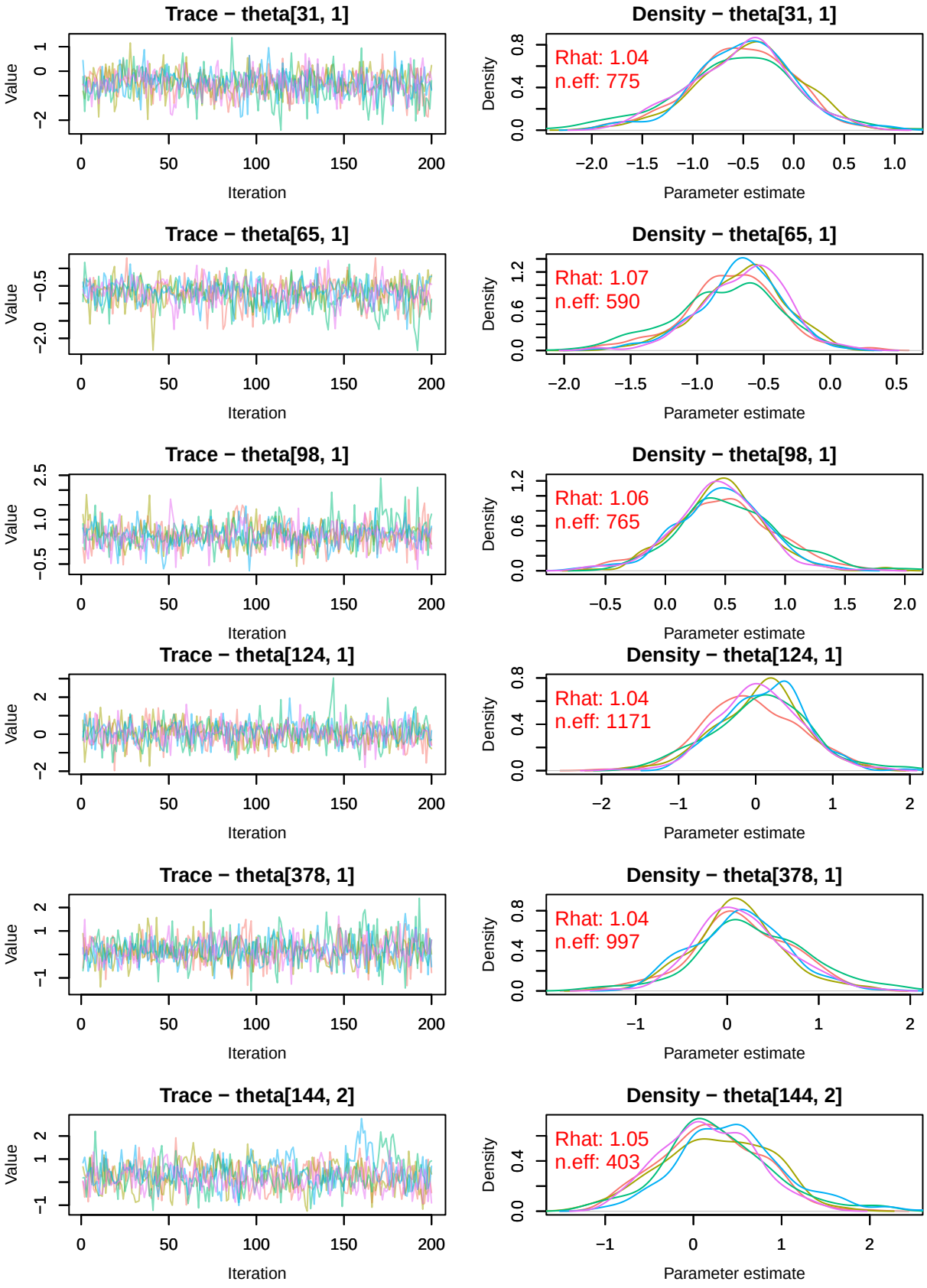
```
MCMCtrace(object = salnimble1,
          pdf = FALSE, # no export to PDF
          ind = TRUE, # separate density lines per chain
          Rhat = TRUE,
          n.eff = TRUE,
          params = "sd.theta")
```

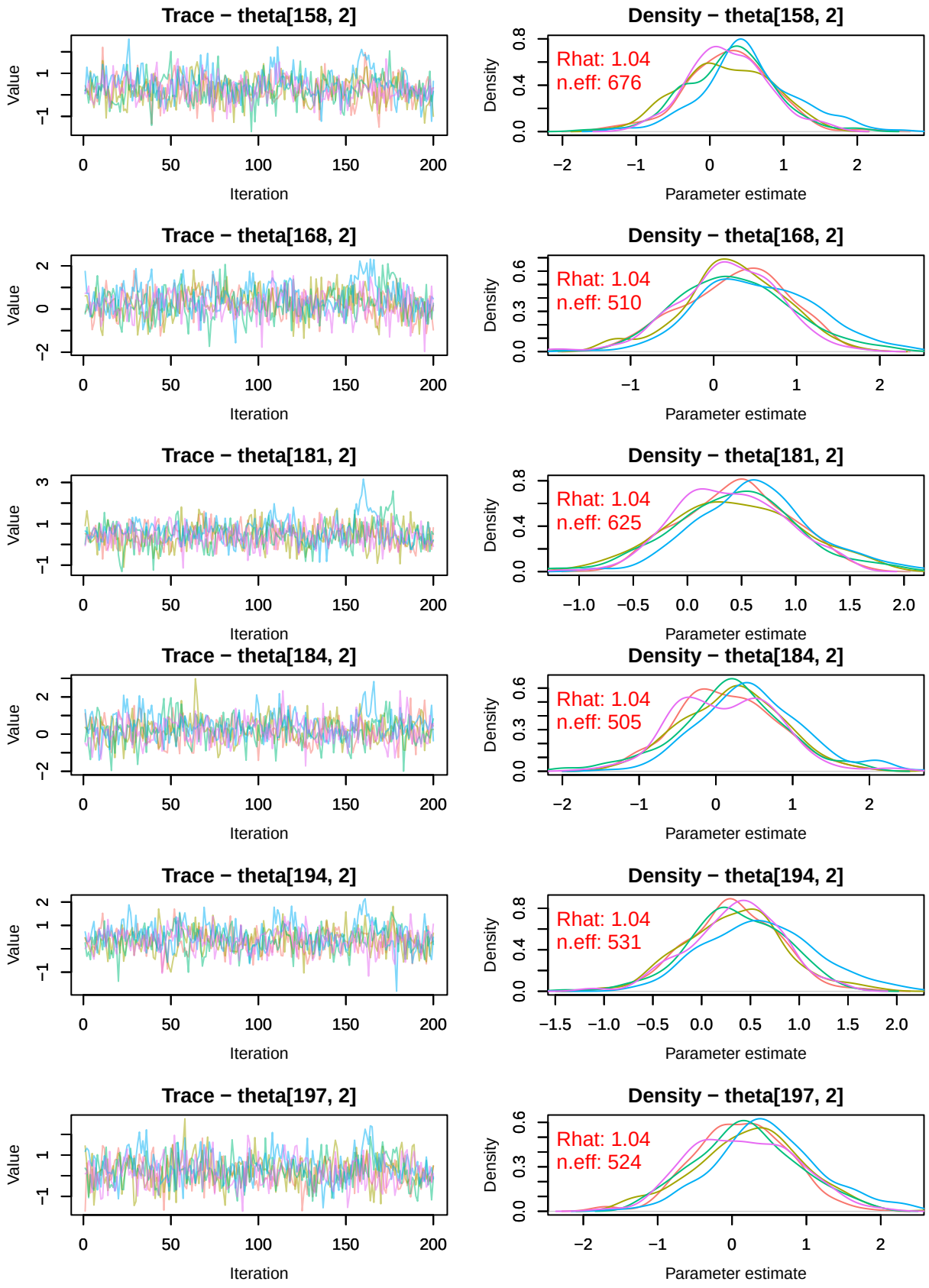


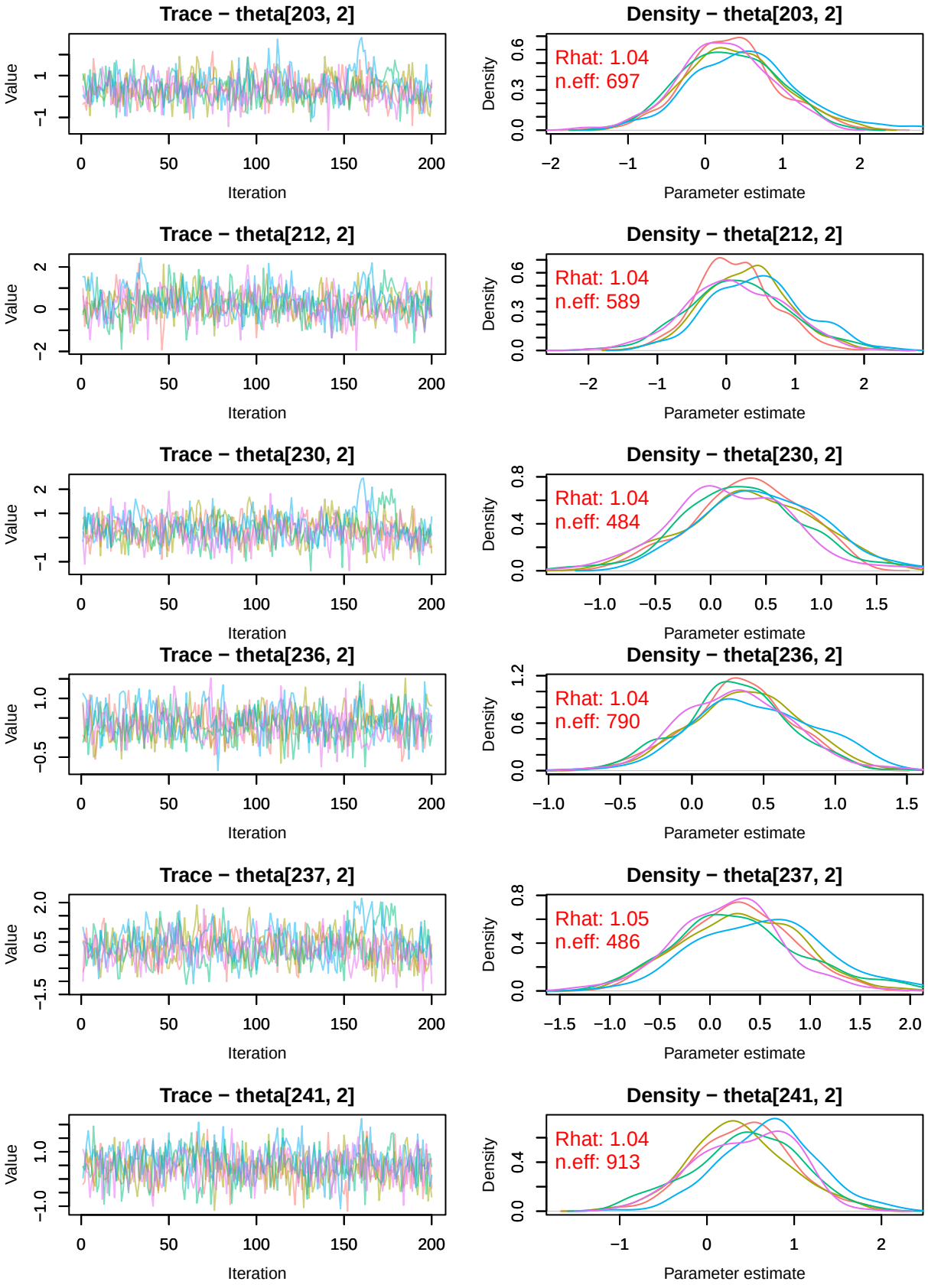

Trace − sd.theta[4]
Value
Iteration
Density − sd.theta[4]
Rhat: 1.02
n.eff: 630
Density
Parameter estimate
Trace − sd.theta[5]
Value
Iteration
Density − sd.theta[5]
Rhat: 1
n.eff: 636
Density
Parameter estimate
Trace − sd.theta[6]
Value
Iteration
Density − sd.theta[6]
Rhat: 1.02
n.eff: 527
Density
Parameter estimate
Trace − sd.theta[7]
Value
Iteration
Density − sd.theta[7]
Rhat: 1.01
n.eff: 699
Density
Parameter estimate
Trace − sd.theta[8]
Value
Iteration
Density − sd.theta[8]
Rhat: 1.03
n.eff: 533
Density
Parameter estimate
Trace − sd.theta[9]
Value
Iteration
Density − sd.theta[9]
Rhat: 1
n.eff: 475
Density
Parameter estimate

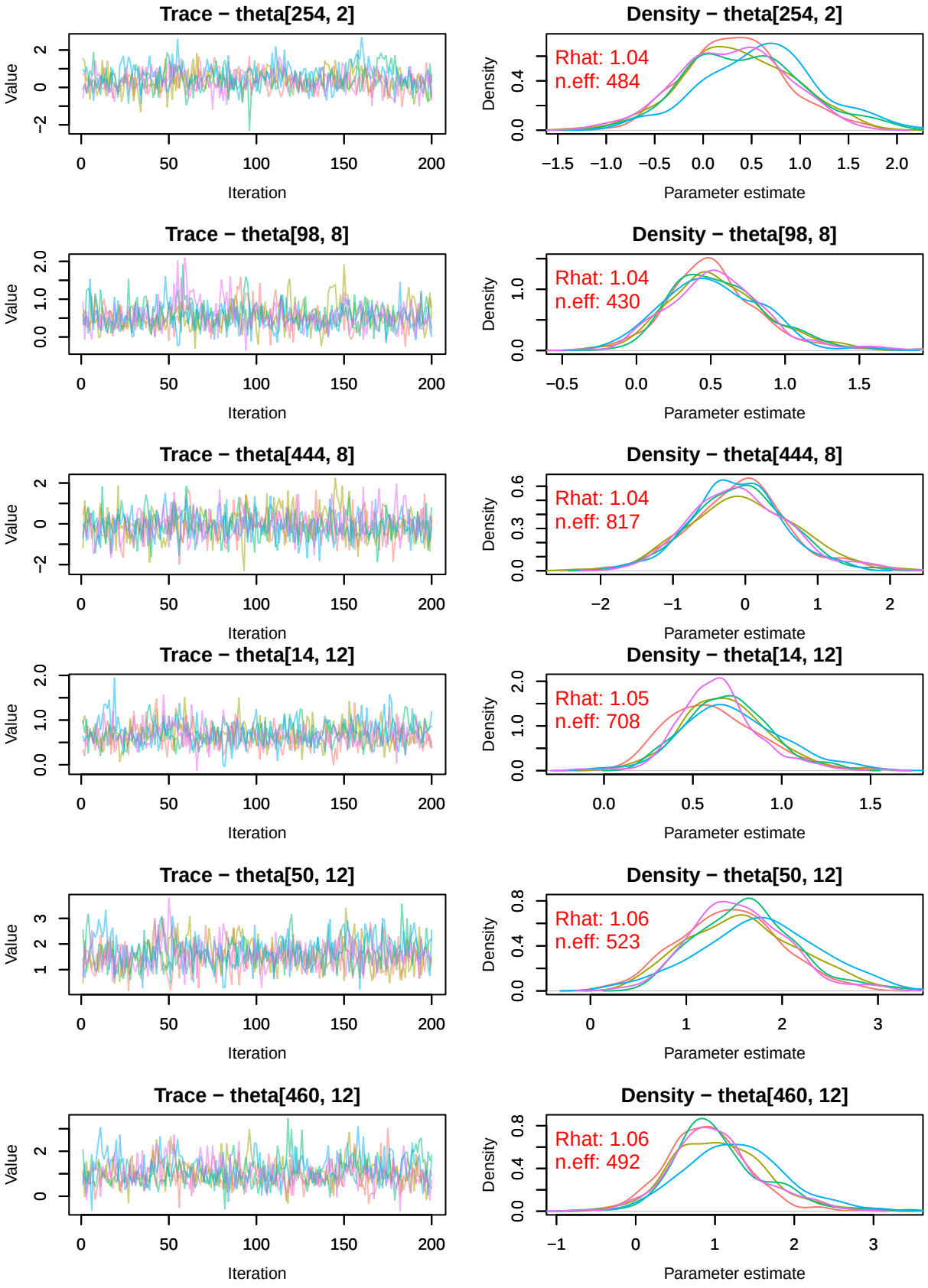


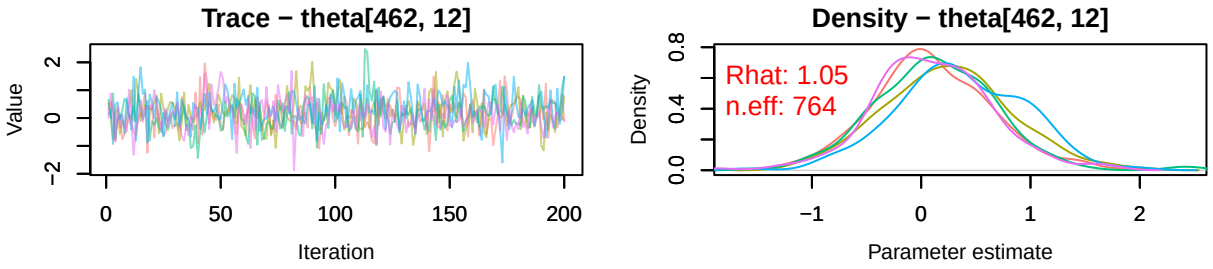

```
test <- "theta"

MCMCtrace(object = salnimble1,
          pdf = FALSE, # no export to PDF
          ind = TRUE, # separate density lines per chain
          Rhat = TRUE,
          n.eff = TRUE,
          exact = TRUE,
          ISB = FALSE,
          params = rownames(salwinbugs1$summary[startsWith(labels(salwinbugs1$summary)[[1]],
          ↪  test), ])[(salwinbugs1$summary[startsWith(labels(salwinbugs1$summary)[[1]], test),
          ↪  6] > 1.03) | (salwinbugs1$summary[startsWith(labels(salwinbugs1$summary)[[1]],
          ↪  test), 7] < 200)])
```


Trace − theta[31, 1]
Density − theta[31, 1]
Rhat: 1.04
n.eff: 775
Trace − theta[65, 1]
Density − theta[65, 1]
Rhat: 1.07
n.eff: 590
Trace − theta[98, 1]
Density − theta[98, 1]
Rhat: 1.06
n.eff: 765
Trace − theta[124, 1]
Density − theta[124, 1]
Rhat: 1.04
n.eff: 1171
Trace − theta[378, 1]
Density − theta[378, 1]
Rhat: 1.04
n.eff: 997
Trace − theta[144, 2]
Density − theta[144, 2]
Rhat: 1.05
n.eff: 403
Value
Iteration
Density
Parameter estimate


Trace − theta[158, 2]
Density − theta[158, 2]
Rhat: 1.04
n.eff: 676
Trace − theta[168, 2]
Density − theta[168, 2]
Rhat: 1.04
n.eff: 510
Trace − theta[181, 2]
Density − theta[181, 2]
Rhat: 1.04
n.eff: 625
Trace − theta[184, 2]
Density − theta[184, 2]
Rhat: 1.04
n.eff: 505
Trace − theta[194, 2]
Density − theta[194, 2]
Rhat: 1.04
n.eff: 531
Trace − theta[197, 2]
Density − theta[197, 2]
Rhat: 1.04
n.eff: 524
Value
Iteration
Density
Parameter estimate


Trace − theta[203, 2]
Density − theta[203, 2]
Rhat: 1.04
n.eff: 697
Trace − theta[212, 2]
Density − theta[212, 2]
Rhat: 1.04
n.eff: 589
Trace − theta[230, 2]
Density − theta[230, 2]
Rhat: 1.04
n.eff: 484
Trace − theta[236, 2]
Density − theta[236, 2]
Rhat: 1.04
n.eff: 790
Trace − theta[237, 2]
Density − theta[237, 2]
Rhat: 1.05
n.eff: 486
Trace − theta[241, 2]
Density − theta[241, 2]
Rhat: 1.04
n.eff: 913
Value
Density
Iteration
Parameter estimate


Trace − theta[254, 2]
Density − theta[254, 2]
Rhat: 1.04
n.eff: 484
Trace − theta[98, 8]
Density − theta[98, 8]
Rhat: 1.04
n.eff: 430
Trace − theta[444, 8]
Density − theta[444, 8]
Rhat: 1.04
n.eff: 817
Trace − theta[14, 12]
Density − theta[14, 12]
Rhat: 1.05
n.eff: 708
Trace − theta[50, 12]
Density − theta[50, 12]
Rhat: 1.06
n.eff: 523
Trace − theta[460, 12]
Density − theta[460, 12]
Rhat: 1.06
n.eff: 492
Value
Density
Iteration
Parameter estimate



Convergence assessment for Model-Corr

```
# Function: salnimble to salwinbugs for Model-Corr

NimToWin <- function(salnimble) {

  n.chains <- length(salnimble)
  n.sims <- n.chains * nrow(salnimble[[1]])

  kappa <- array(dim = c(n.sims, NSex, NAges, NCats, NVars))
  theta <- array(dim = c(n.sims, NMuni, NVars))
  sd.M.Muni <- numeric(length = n.sims)
  M.Muni <- array(dim = c(n.sims, NVars, NVars))
  rho <- matrix(nrow = n.sims, ncol = NVars)

  for (Var in 1:NVars) {
    for (Cat in 1:(NCats - 1)) {
      for (Sex in 1:NSex) {
        for (Age in 1:NAges) {
          kappa[, Sex, Age, Cat, Var] <- c(salnimble[[1]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")],
                                           salnimble[[2]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")],
                                           salnimble[[3]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")],
                                           salnimble[[4]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")],
                                           salnimble[[5]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")])
        }
      }
    }
  }

  for (Var in 1:NVars) {
    for (Muni in 1:NMuni) {
      theta[, Muni, Var] <- c(salnimble[[1]][, paste0("theta[", Muni, ", ", Var, "]")],
                              salnimble[[2]][, paste0("theta[", Muni, ", ", Var, "]")],
                              salnimble[[3]][, paste0("theta[", Muni, ", ", Var, "]")],
                              salnimble[[4]][, paste0("theta[", Muni, ", ", Var, "]")],
                              salnimble[[5]][, paste0("theta[", Muni, ", ", Var, "]")])
    }
  }

  sd.M.Muni <- c(salnimble[[1]][, "sd.M.Muni"], salnimble[[2]][, "sd.M.Muni"],
                 salnimble[[3]][, "sd.M.Muni"], salnimble[[4]][, "sd.M.Muni"],
                 salnimble[[5]][, "sd.M.Muni"])
```

```
  for (Var1 in 1:NVars) {
    for (Var2 in 1:NVars) {
      M.Muni[, Var1, Var2] <- c(salnimble[[1]][, paste0("M.Muni[", Var1, ", ", Var2, "]")],
                                salnimble[[2]][, paste0("M.Muni[", Var1, ", ", Var2, "]")],
                                salnimble[[3]][, paste0("M.Muni[", Var1, ", ", Var2, "]")],
                                salnimble[[4]][, paste0("M.Muni[", Var1, ", ", Var2, "]")],
                                salnimble[[5]][, paste0("M.Muni[", Var1, ", ", Var2, "]")])
    }
  }

  for (Var in 1:NVars) {
    rho[, Var] <- c(salnimble[[1]][, paste0("rho[", Var, "]")],
                    salnimble[[2]][, paste0("rho[", Var, "]")],
                    salnimble[[3]][, paste0("rho[", Var, "]")],
                    salnimble[[4]][, paste0("rho[", Var, "]")],
                    salnimble[[5]][, paste0("rho[", Var, "]")])
  }

  summary <- MCMCsummary(object = salnimble, round = 4)
  # summary <- "not available"
  sims.list <- list("kappa" = kappa, "theta" = theta, "sd.M.Muni" = sd.M.Muni,
                    "M.Muni" = M.Muni, "rho" = rho)

  salwinbugs <- list("summary" = summary, "sims.list" = sims.list,
                     "n.chains" = n.chains, "n.sims" = n.sims)

  return(salwinbugs)
}

# salwinbugs2 <- NimToWin(salnimble = salnimble2)
# saveRDS(salwinbugs2, file = file.path("results", "salwinbugs2.rds"))
salwinbugs2 <- readRDS(file = file.path("..", "results", "salwinbugs2.rds"))

which((salwinbugs2$summary[startsWith(labels(salwinbugs2$summary)[[1]], "kappa"), 6] > 1.1) |
↪  (salwinbugs2$summary[startsWith(labels(salwinbugs2$summary)[[1]], "kappa"), 7] < 100))
```

```
## integer(0)
```

```
which((salwinbugs2$summary[startsWith(labels(salwinbugs2$summary)[[1]], "sd.M.Muni"), 6] > 1.1)
↪  | (salwinbugs2$summary[startsWith(labels(salwinbugs2$summary)[[1]], "sd.M.Muni"), 7] < 100))
```

```
## integer(0)
```

```
which((salwinbugs2$summary[startsWith(labels(salwinbugs2$summary)[[1]], "theta"), 6] > 1.1) |
↪  (salwinbugs2$summary[startsWith(labels(salwinbugs2$summary)[[1]], "theta"), 7] < 100))
```

```
## integer(0)
```

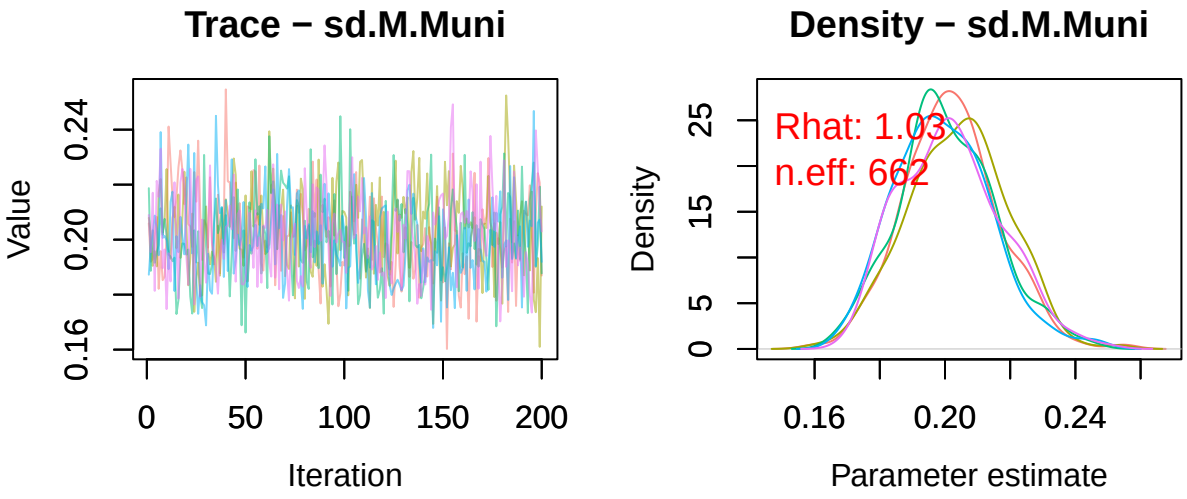
```
MCMCsummary(object = salnimble2, params = "kappa",
            # exact = TRUE,
            # ISB = FALSE,
            round = 4)[1:10, ]
```

```
##                       mean     sd    2.5%     50%   97.5% Rhat n.eff
## kappa[1, 1, 1, 1] -2.6112 0.1731 -2.9459 -2.6110 -2.2738 1.01  1058
## kappa[2, 1, 1, 1] -2.3826 0.1647 -2.7222 -2.3787 -2.0768 1.01   861
## kappa[1, 2, 1, 1] -2.4966 0.1690 -2.8469 -2.4918 -2.1651 1.00   982
## kappa[2, 2, 1, 1] -2.5389 0.1684 -2.8639 -2.5331 -2.2225 1.00  1046
## kappa[1, 3, 1, 1] -2.7670 0.1606 -3.0860 -2.7713 -2.4757 1.01  1020
## kappa[2, 3, 1, 1] -2.4918 0.1441 -2.7835 -2.4894 -2.2361 1.00  1000
## kappa[1, 4, 1, 1] -3.3326 0.1989 -3.7429 -3.3224 -2.9798 1.01   960
## kappa[2, 4, 1, 1] -2.9040 0.1578 -3.2336 -2.9044 -2.6051 1.01  1000
## kappa[1, 5, 1, 1] -3.1721 0.1934 -3.5596 -3.1598 -2.8165 1.01   926
## kappa[2, 5, 1, 1] -3.2198 0.1860 -3.5956 -3.2112 -2.8751 1.01  1046
```

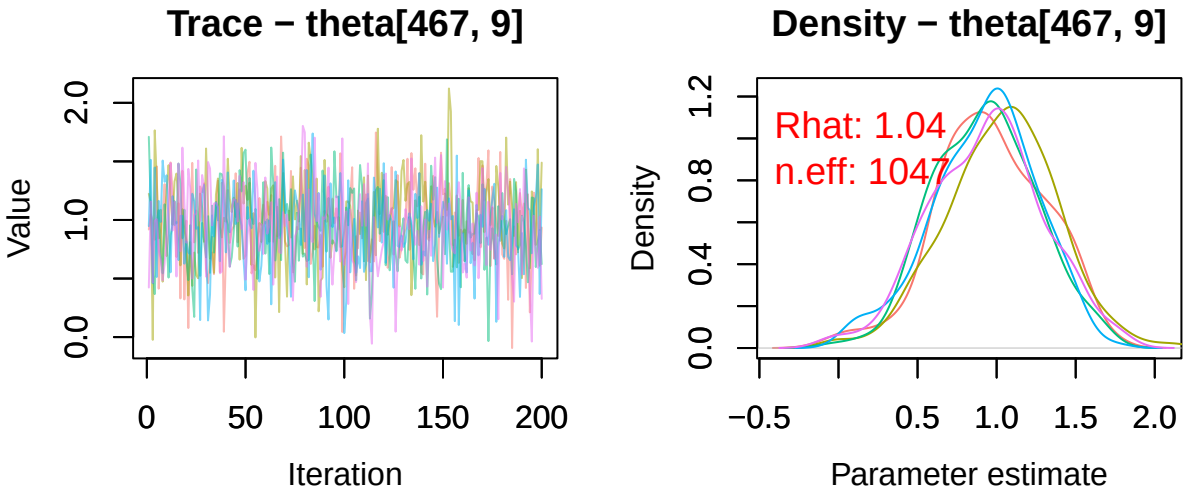
```r
MCMCsummary(object = salnimble2, params = "theta",
            # exact = TRUE,
            # ISB = FALSE,
            round = 4)[1:10, ]
```

```
##                   mean     sd    2.5%     50%  97.5% Rhat n.eff
## theta[1, 1]    0.0150 0.4553 -0.8817  0.0201 0.9334 1.01  1000
## theta[2, 1]   -0.3297 0.2982 -0.9045 -0.3354 0.2538 1.01   945
## theta[3, 1]   -0.0762 0.4359 -0.8942 -0.0900 0.7585 1.00   953
## theta[4, 1]   -0.1383 0.3822 -0.9428 -0.1223 0.5782 1.00  1000
## theta[5, 1]   -0.1882 0.1914 -0.5586 -0.1828 0.1905 1.00  1088
## theta[6, 1]    0.0630 0.3232 -0.5488  0.0655 0.6981 1.00  1011
## theta[7, 1]    0.0570 0.4576 -0.8418  0.0560 0.9570 1.00  1138
## theta[8, 1]   -0.0498 0.4168 -0.9187 -0.0486 0.7417 1.00   955
## theta[9, 1]    0.0888 0.1247 -0.1581  0.0906 0.3315 1.01  1123
## theta[10, 1] -0.0414 0.4793 -0.9397 -0.0442 0.9253 1.00   962
```

```r
MCMCtrace(object = salnimble2,
          pdf = FALSE, # no export to PDF
          ind = TRUE, # separate density lines per chain
          Rhat = TRUE,
          n.eff = TRUE,
          params = "sd.M.Muni")
```



```r
test <- "theta"

MCMCtrace(object = salnimble2,
          pdf = FALSE, # no export to PDF
          ind = TRUE, # separate density lines per chain
          Rhat = TRUE,
          n.eff = TRUE,
          exact = TRUE,
          ISB = FALSE,
          params = rownames(salwinbugs2$summary[startsWith(labels(salwinbugs2$summary)[[1]],
          ↪  test), ])[(salwinbugs2$summary[startsWith(labels(salwinbugs2$summary)[[1]], test),
          ↪  6] > 1.03) | (salwinbugs2$summary[startsWith(labels(salwinbugs2$summary)[[1]],
          ↪  test), 7] < 200)])
```



Convergence assessment for Model-Corr&IRE

```r
# Function: salnimble to salwinbugs for Model-Corr&IRE

NimToWin <- function(salnimble) {

  n.chains <- length(salnimble)
  n.sims <- n.chains * nrow(salnimble[[1]])

  kappa <- array(dim = c(n.sims, NSex, NAges, NCats, NVars))
  psi <- array(dim = c(n.sims, NResp, NVars))
  sd.M.Resp <- numeric(length = n.sims)
  M.Resp <- array(dim = c(n.sims, NVars, NVars))
  theta <- array(dim = c(n.sims, NMuni, NVars))
  sd.M.Muni <- numeric(length = n.sims)
  M.Muni <- array(dim = c(n.sims, NVars, NVars))
  rho <- matrix(nrow = n.sims, ncol = NVars)

  for (Var in 1:NVars) {
    for (Cat in 1:(NCats - 1)) {
      for (Sex in 1:NSex) {
        for (Age in 1:NAges) {
          kappa[, Sex, Age, Cat, Var] <- c(salnimble[[1]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")],
```

```
                                          salnimble[[2]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")],
                                          salnimble[[3]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")],
                                          salnimble[[4]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")],
                                          salnimble[[5]][,  paste0("kappa[", Sex, ", ", Age, ",
↪  ", Cat, ", ", Var, "]")])
       }
     }
   }
 }

 for (Var in 1:NVars) {
   for (Resp in 1:NResp) {
     psi[, Resp, Var] <- c(salnimble[[1]][, paste0("psi[", Resp, ", ", Var, "]")],
                           salnimble[[2]][, paste0("psi[", Resp, ", ", Var, "]")],
                           salnimble[[3]][, paste0("psi[", Resp, ", ", Var, "]")],
                           salnimble[[4]][, paste0("psi[", Resp, ", ", Var, "]")],
                           salnimble[[5]][, paste0("psi[", Resp, ", ", Var, "]")])
   }
 }

 sd.M.Resp <- c(salnimble[[1]][, "sd.M.Resp"], salnimble[[2]][, "sd.M.Resp"],
                salnimble[[3]][, "sd.M.Resp"], salnimble[[4]][, "sd.M.Resp"],
                salnimble[[5]][, "sd.M.Resp"])

 for (Var1 in 1:NVars) {
   for (Var2 in 1:NVars) {
     M.Resp[, Var1, Var2] <- c(salnimble[[1]][, paste0("M.Resp[", Var1, ", ", Var2, "]")],
                               salnimble[[2]][, paste0("M.Resp[", Var1, ", ", Var2, "]")],
                               salnimble[[3]][, paste0("M.Resp[", Var1, ", ", Var2, "]")],
                               salnimble[[4]][, paste0("M.Resp[", Var1, ", ", Var2, "]")],
                               salnimble[[5]][, paste0("M.Resp[", Var1, ", ", Var2, "]")])
   }
 }

 for (Var in 1:NVars) {
   for (Muni in 1:NMuni) {
     theta[, Muni, Var] <- c(salnimble[[1]][, paste0("theta[", Muni, ", ", Var, "]")],
                             salnimble[[2]][, paste0("theta[", Muni, ", ", Var, "]")],
                             salnimble[[3]][, paste0("theta[", Muni, ", ", Var, "]")],
                             salnimble[[4]][, paste0("theta[", Muni, ", ", Var, "]")],
                             salnimble[[5]][, paste0("theta[", Muni, ", ", Var, "]")])
   }
 }

 sd.M.Muni <- c(salnimble[[1]][, "sd.M.Muni"], salnimble[[2]][, "sd.M.Muni"],
                salnimble[[3]][, "sd.M.Muni"], salnimble[[4]][, "sd.M.Muni"],
                salnimble[[5]][, "sd.M.Muni"])

 for (Var1 in 1:NVars) {
   for (Var2 in 1:NVars) {
     M.Muni[, Var1, Var2] <- c(salnimble[[1]][, paste0("M.Muni[", Var1, ", ", Var2, "]")],
                               salnimble[[2]][, paste0("M.Muni[", Var1, ", ", Var2, "]")],
                               salnimble[[3]][, paste0("M.Muni[", Var1, ", ", Var2, "]")],
                               salnimble[[4]][, paste0("M.Muni[", Var1, ", ", Var2, "]")],
                               salnimble[[5]][, paste0("M.Muni[", Var1, ", ", Var2, "]")])
   }
```

```
  }

  for (Var in 1:NVars) {
    rho[, Var] <- c(salnimble[[1]][, paste0("rho[", Var, "]")],
                    salnimble[[2]][, paste0("rho[", Var, "]")],
                    salnimble[[3]][, paste0("rho[", Var, "]")],
                    salnimble[[4]][, paste0("rho[", Var, "]")],
                    salnimble[[5]][, paste0("rho[", Var, "]")])
  }

  summary <- MCMCsummary(object = salnimble, round = 4)
  # summary <- "not available"
  sims.list <- list("kappa" = kappa, "theta" = theta, "M.Muni" = M.Muni,
                    "rho" = rho, "sd.M.Muni" = sd.M.Muni, "psi" = psi,
                    "sd.M.Resp" = sd.M.Resp, "M.Resp" = M.Resp)

  salwinbugs <- list("summary" = summary, "sims.list" = sims.list,
                     "n.chains" = n.chains, "n.sims" = n.sims)

  return(salwinbugs)
}

# salwinbugs3 <- NimToWin(salnimble = salnimble3)
# saveRDS(salwinbugs3, file = file.path("results", "salwinbugs3.rds"))
salwinbugs3 <- readRDS(file = file.path("..", "results", "salwinbugs3.rds"))

which((salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]], "kappa"), 6] > 1.1) |
↪  (salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]], "kappa"), 7] < 100))
```

```
## integer(0)
```

```
which((salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]], "sd.M.Muni"), 6] > 1.1)
↪  | (salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]], "sd.M.Muni"), 7] < 100))
```

```
## integer(0)
```

```
which((salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]], "theta"), 6] > 1.1) |
↪  (salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]], "theta"), 7] < 100))
```

```
## integer(0)
```

```
which((salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]], "sd.M.Resp"), 6] > 1.1)
↪  | (salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]], "sd.M.Resp"), 7] < 100))
```

```
## integer(0)
```

```
which((salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]], "psi"), 6] > 1.1) |
↪  (salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]], "psi"), 7] < 100))
```

```
## integer(0)
```

```
MCMCsummary(object = salnimble3, params = "kappa",
            # exact = TRUE,
            # ISB = FALSE,
            round = 4)[1:10, ]
```

```
##                       mean     sd    2.5%     50%   97.5% Rhat n.eff
## kappa[1, 1, 1, 1] -4.0030 0.2768 -4.5731 -4.0036 -3.4708 1.00  1003
## kappa[2, 1, 1, 1] -3.7967 0.2473 -4.2924 -3.7924 -3.3261 1.00   970
## kappa[1, 2, 1, 1] -3.9218 0.2430 -4.4046 -3.9187 -3.4581 1.01  1445
## kappa[2, 2, 1, 1] -3.8904 0.2363 -4.3534 -3.8956 -3.4216 1.00   988
## kappa[1, 3, 1, 1] -4.2577 0.2305 -4.6876 -4.2617 -3.8036 1.01  1111
## kappa[2, 3, 1, 1] -3.8801 0.2070 -4.2995 -3.8756 -3.4905 1.00   965
## kappa[1, 4, 1, 1] -4.9890 0.2604 -5.5495 -4.9876 -4.5058 1.00   944
## kappa[2, 4, 1, 1] -4.2869 0.2238 -4.7612 -4.2826 -3.8899 1.00   934
## kappa[1, 5, 1, 1] -4.7905 0.2562 -5.2968 -4.7829 -4.3253 1.00  1097
## kappa[2, 5, 1, 1] -4.7581 0.2590 -5.2821 -4.7522 -4.2430 1.00  1000
```

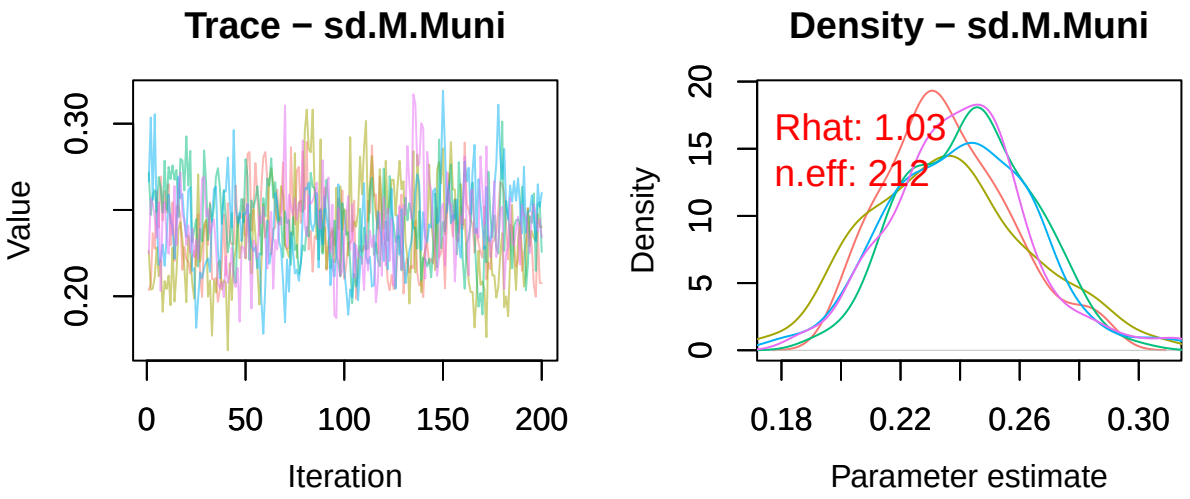
```
MCMCsummary(object = salnimble3, params = "theta",
            # exact = TRUE,
            # ISB = FALSE,
            round = 4)[1:10, ]
```

```
##                  mean     sd    2.5%     50%  97.5% Rhat n.eff
## theta[1, 1]  -0.0448 0.2911 -0.6366 -0.0442 0.5428 1.00  1084
## theta[2, 1]  -0.0203 0.2791 -0.5612 -0.0281 0.5374 1.00  1443
## theta[3, 1]  -0.0414 0.2641 -0.5677 -0.0434 0.4668 1.00   900
## theta[4, 1]   0.2203 0.2884 -0.3022  0.2122 0.8244 1.01   934
## theta[5, 1]  -0.0733 0.2362 -0.5524 -0.0800 0.3920 1.00   936
## theta[6, 1]   0.0367 0.2505 -0.4401  0.0394 0.5253 1.01   921
## theta[7, 1]  -0.0124 0.3143 -0.6595 -0.0081 0.6258 1.00  1006
## theta[8, 1]   0.0285 0.3105 -0.5866  0.0309 0.6456 1.00  1047
## theta[9, 1]   0.0687 0.1959 -0.3015  0.0717 0.4520 1.00   941
## theta[10, 1] -0.0266 0.3279 -0.6564 -0.0208 0.5887 1.01  1084
```

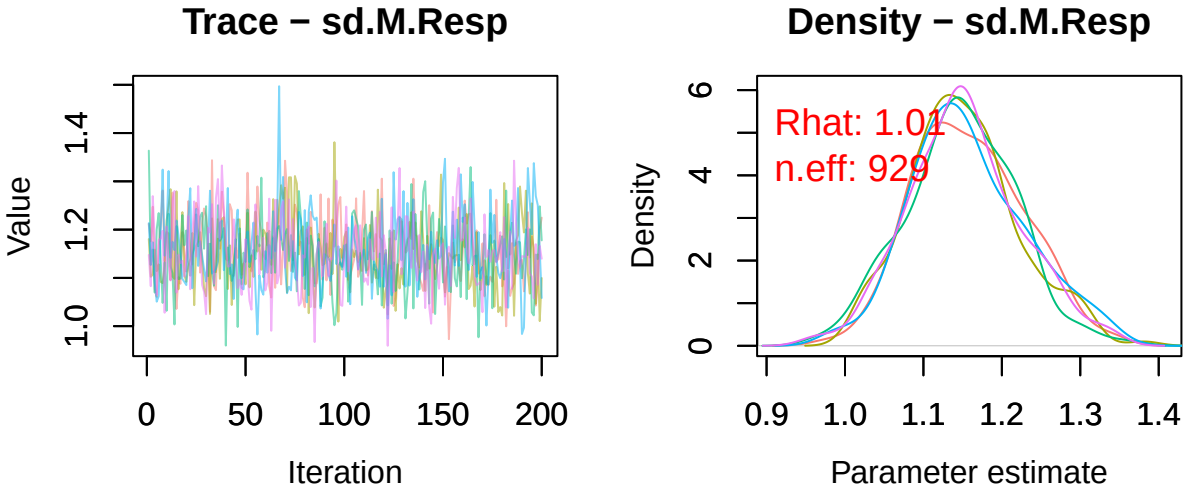
```
MCMCsummary(object = salnimble3, params = "psi",
            # exact = TRUE,
            # ISB = FALSE,
            round = 4)[1:10, ]
```

```
##                mean     sd    2.5%     50%   97.5% Rhat n.eff
## psi[1, 1]   4.2731 0.7922  2.8521  4.2324  5.9855 1.00  1000
## psi[2, 1]  -2.9513 0.7271 -4.1727 -2.9746 -1.4593 1.00  1121
## psi[3, 1]   0.2615 1.2542 -2.2734  0.2722  2.6930 1.00   878
## psi[4, 1]  -4.8959 0.6597 -6.1497 -4.9064 -3.4985 1.01   967
## psi[5, 1]   3.1538 0.8661  1.3123  3.1739  4.7894 1.01  1214
## psi[6, 1]  -1.7509 0.6943 -3.0997 -1.7370 -0.3800 1.00  1000
## psi[7, 1]  -0.1897 1.1725 -2.3539 -0.1894  2.0362 1.00  1046
## psi[8, 1]  -0.0807 1.2117 -2.4073 -0.0721  2.1425 1.00  1702
## psi[9, 1]   0.3816 1.1431 -1.8230  0.3410  2.6404 1.01  1253
## psi[10, 1]  1.2283 1.0045 -0.8305  1.2454  3.0538 1.00   956
```

```
MCMCtrace(object = salnimble3,
          pdf = FALSE, # no export to PDF
          ind = TRUE, # separate density lines per chain
          Rhat = TRUE,
          n.eff = TRUE,
          params = "sd.M.Muni")
```

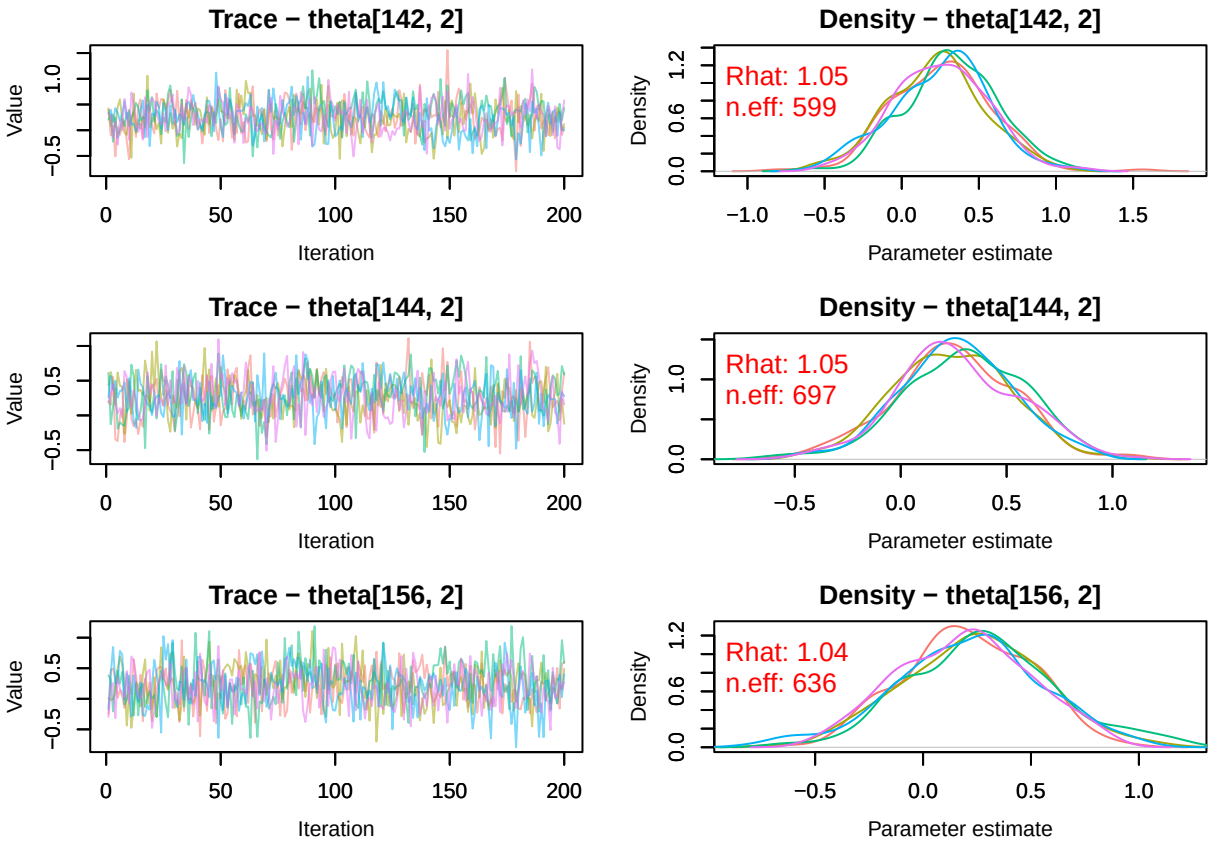


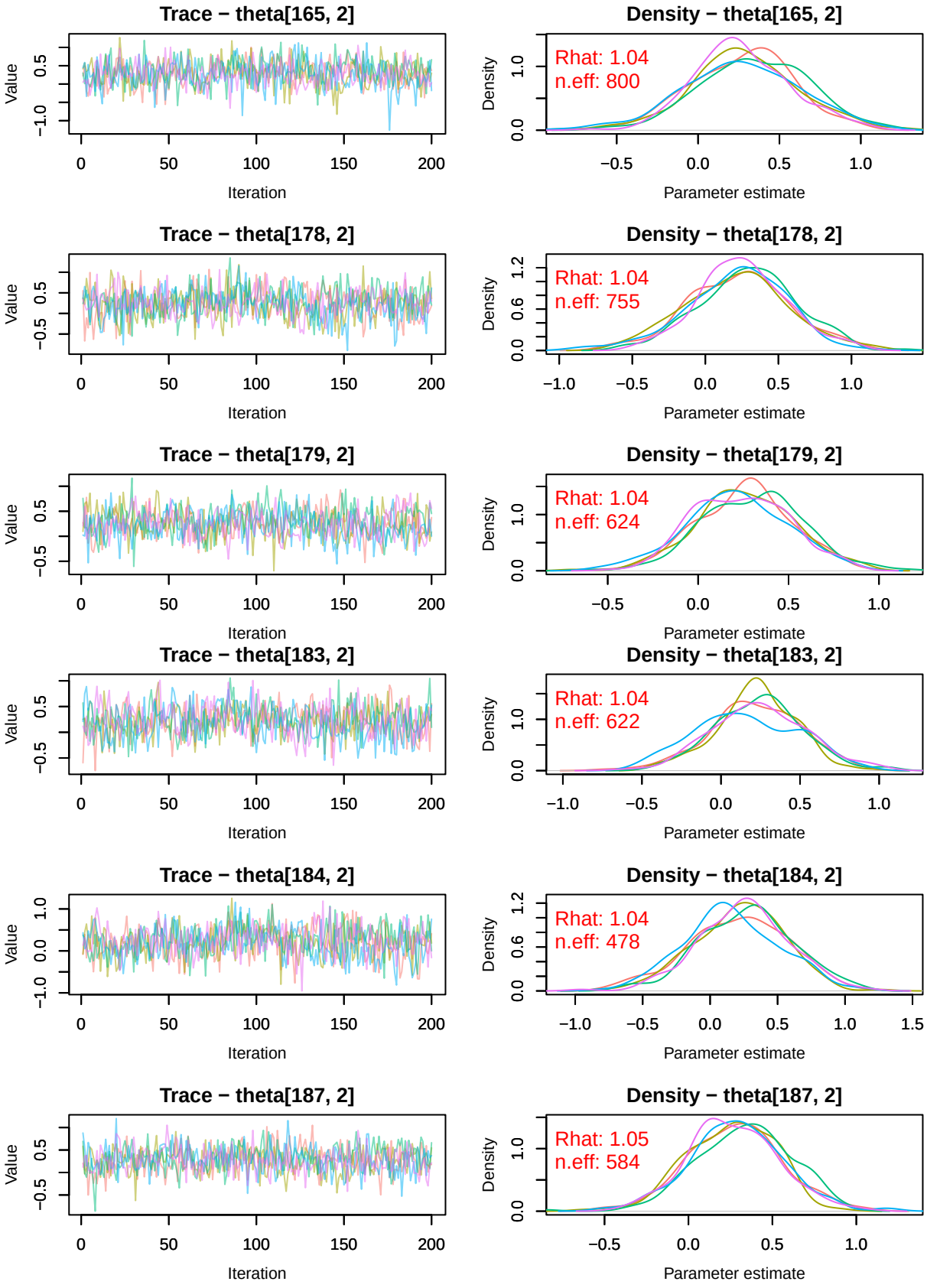

```
MCMCtrace(object = salnimble3,
          pdf = FALSE, # no export to PDF
          ind = TRUE, # separate density lines per chain
          Rhat = TRUE,
          n.eff = TRUE,
          params = "sd.M.Resp")
```

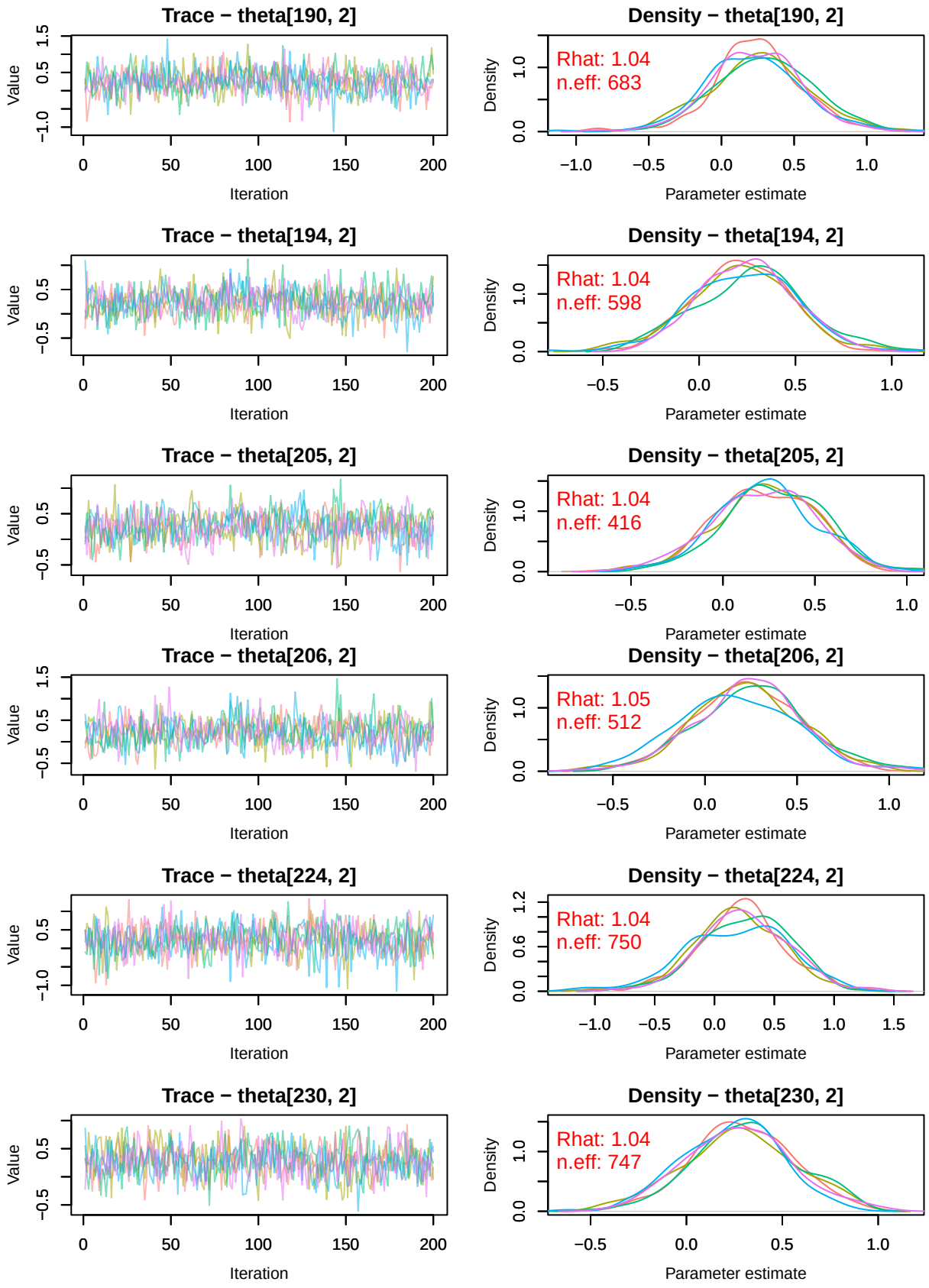


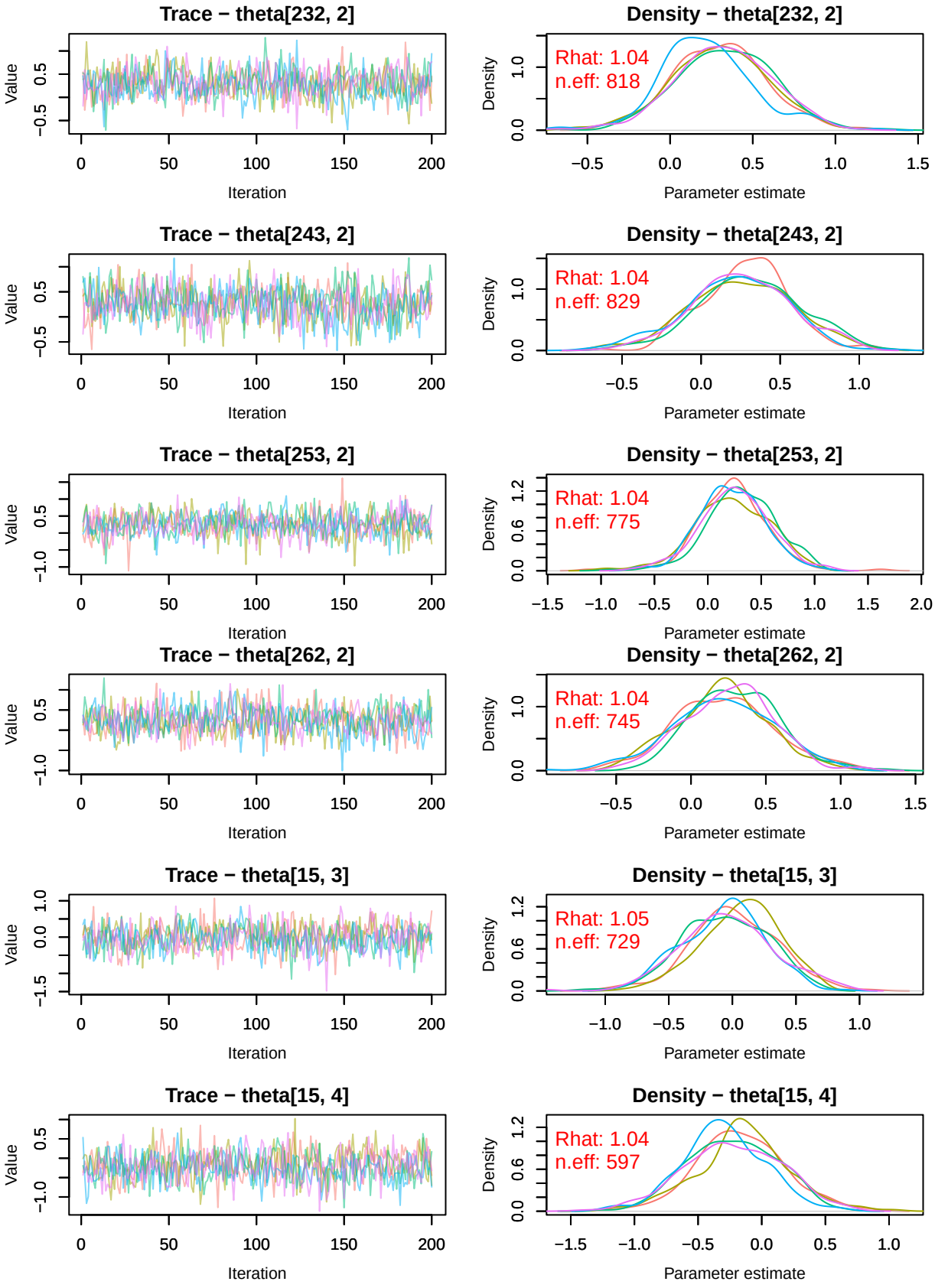

```
test <- "theta"

MCMCtrace(object = salnimble3,
          pdf = FALSE, # no export to PDF
          ind = TRUE, # separate density lines per chain
          Rhat = TRUE,
```

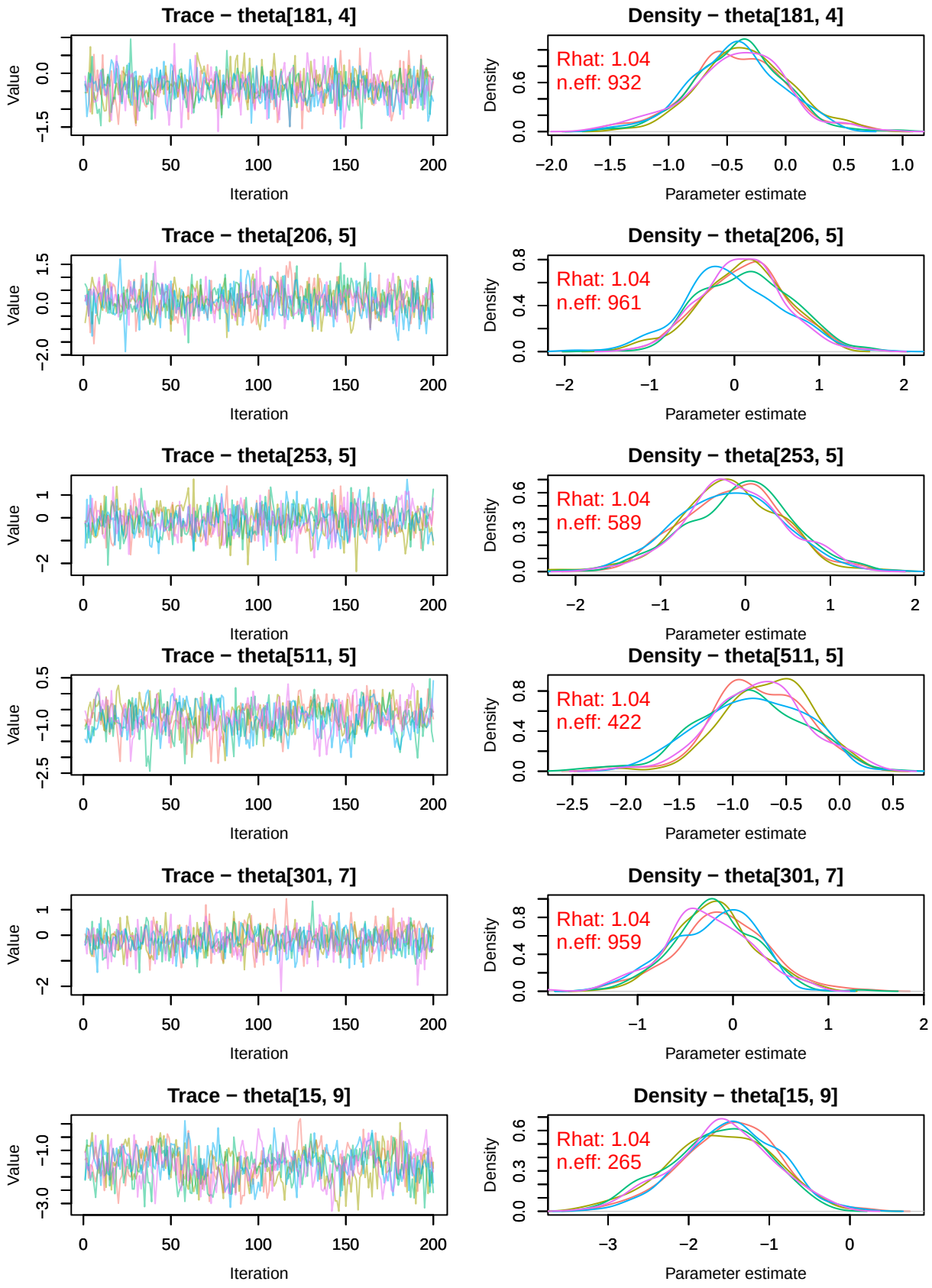
```
        n.eff = TRUE,
        exact = TRUE,
        ISB = FALSE,
        params = rownames(salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]],
        ↪  test), ])[(salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]], test),
        ↪  6] > 1.03) | (salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]],
        ↪  test), 7] < 200)])
```

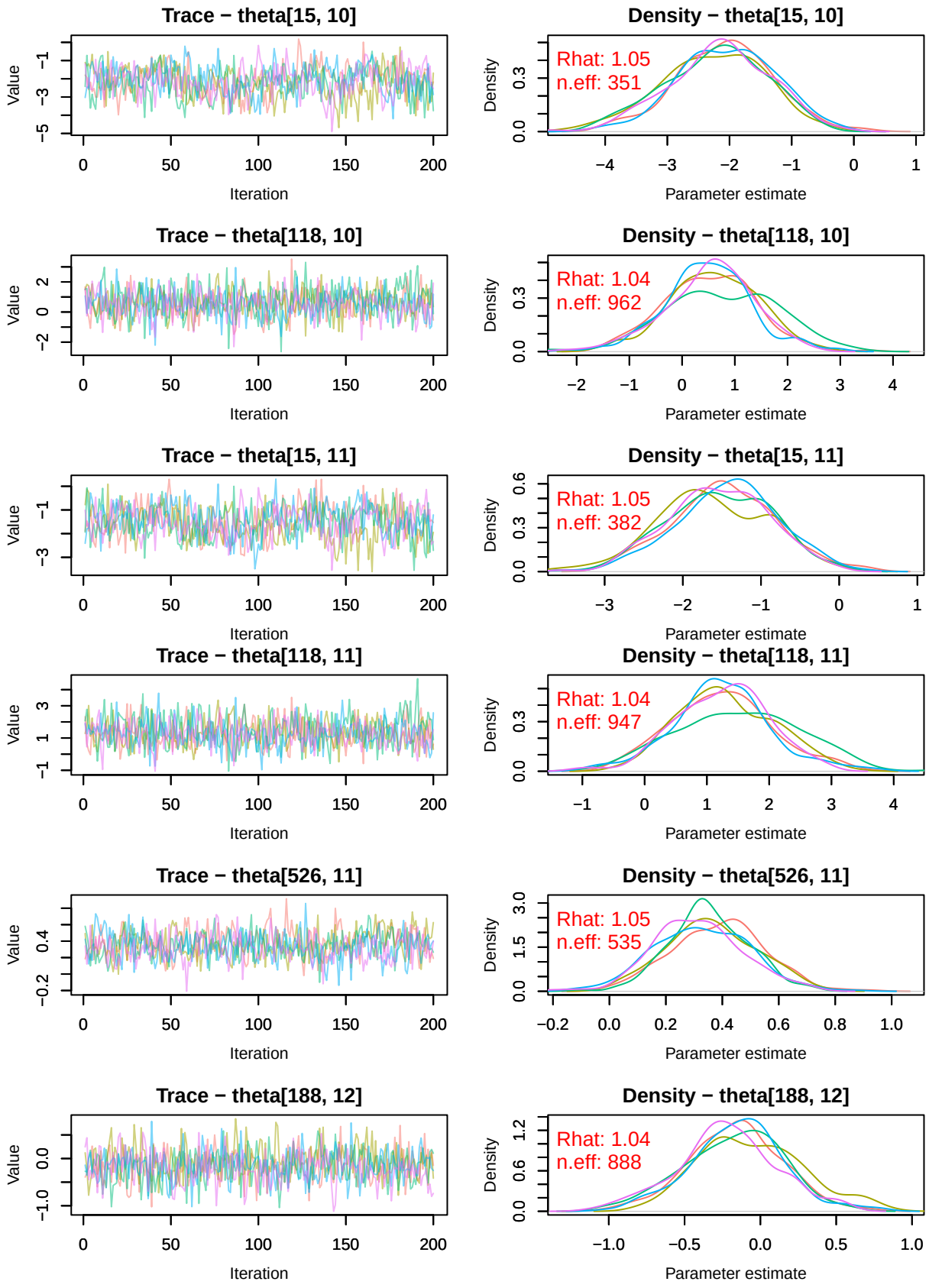



Trace − theta[165, 2]
Density − theta[165, 2]
Rhat: 1.04
n.eff: 800
Trace − theta[178, 2]
Density − theta[178, 2]
Rhat: 1.04
n.eff: 755
Trace − theta[179, 2]
Density − theta[179, 2]
Rhat: 1.04
n.eff: 624
Trace − theta[183, 2]
Density − theta[183, 2]
Rhat: 1.04
n.eff: 622
Trace − theta[184, 2]
Density − theta[184, 2]
Rhat: 1.04
n.eff: 478
Trace − theta[187, 2]
Density − theta[187, 2]
Rhat: 1.05
n.eff: 584
Value
Density
Iteration
Parameter estimate

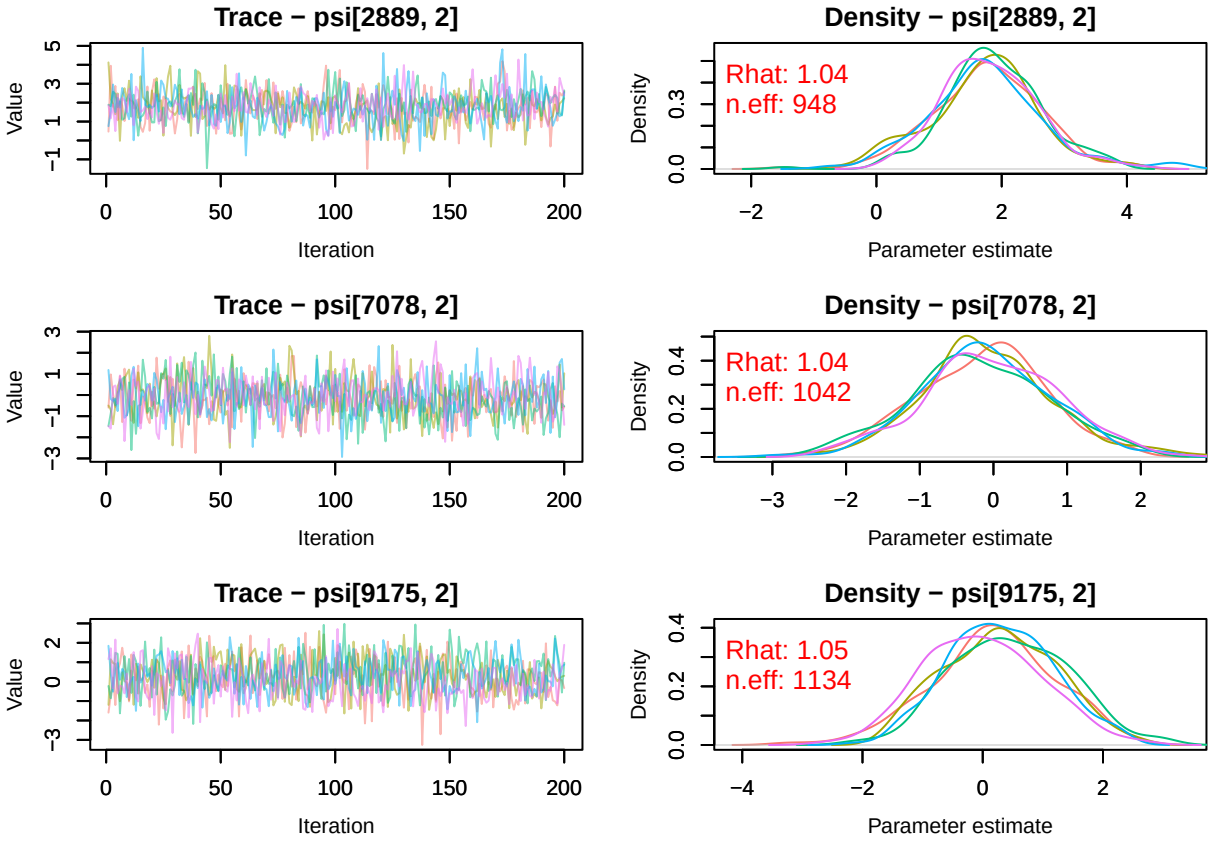

Trace − theta[190, 2]
Density − theta[190, 2]
Rhat: 1.04
n.eff: 683
Trace − theta[194, 2]
Density − theta[194, 2]
Rhat: 1.04
n.eff: 598
Trace − theta[205, 2]
Density − theta[205, 2]
Rhat: 1.04
n.eff: 416
Trace − theta[206, 2]
Density − theta[206, 2]
Rhat: 1.05
n.eff: 512
Trace − theta[224, 2]
Density − theta[224, 2]
Rhat: 1.04
n.eff: 750
Trace − theta[230, 2]
Density − theta[230, 2]
Rhat: 1.04
n.eff: 747
Value
Density
Iteration
Parameter estimate

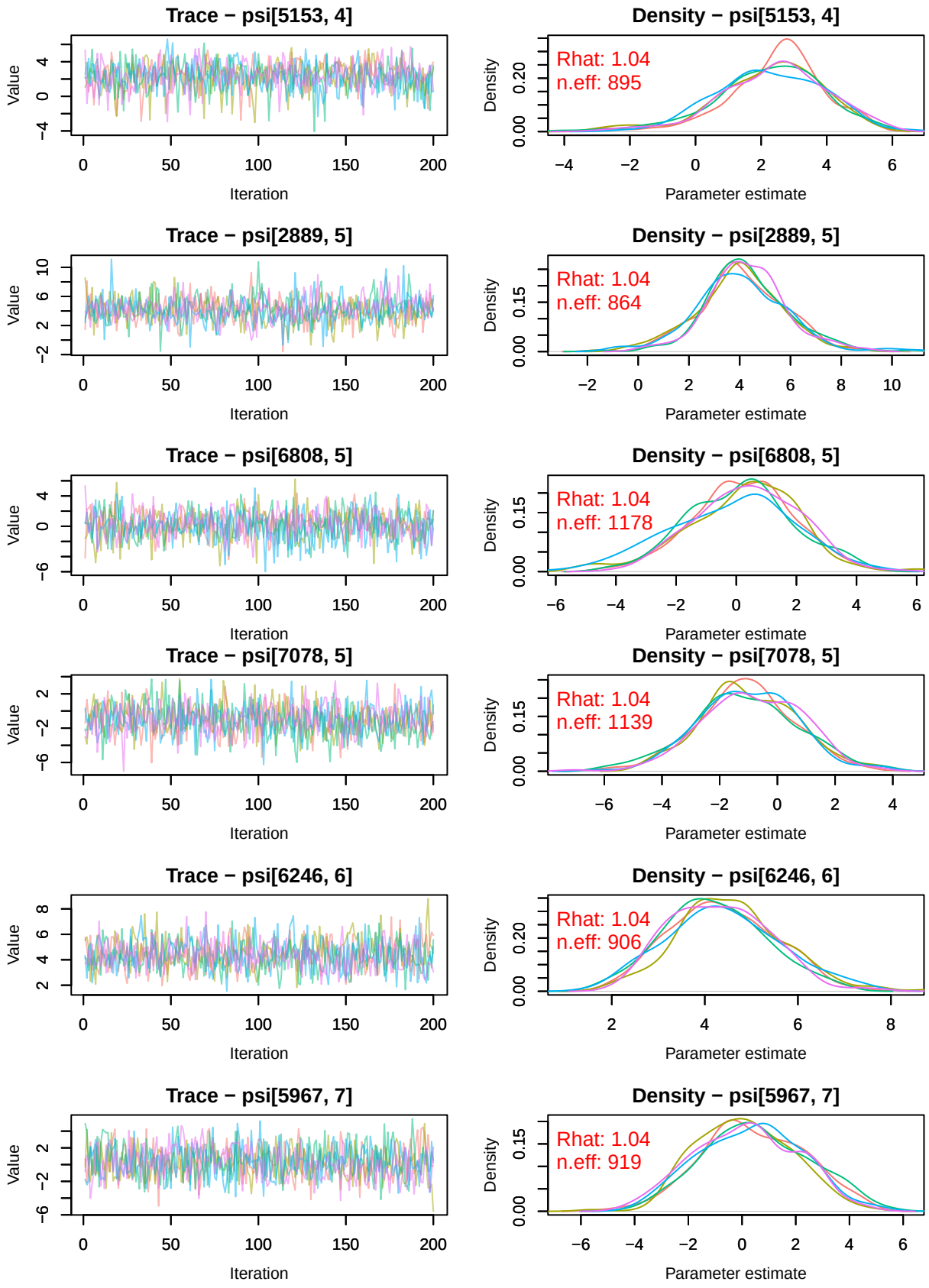

Trace − theta[232, 2]
Density − theta[232, 2]
Rhat: 1.04
n.eff: 818
Trace − theta[243, 2]
Density − theta[243, 2]
Rhat: 1.04
n.eff: 829
Trace − theta[253, 2]
Density − theta[253, 2]
Rhat: 1.04
n.eff: 775
Trace − theta[262, 2]
Density − theta[262, 2]
Rhat: 1.04
n.eff: 745
Trace − theta[15, 3]
Density − theta[15, 3]
Rhat: 1.05
n.eff: 729
Trace − theta[15, 4]
Density − theta[15, 4]
Rhat: 1.04
n.eff: 597
Value
Density
Iteration
Parameter estimate

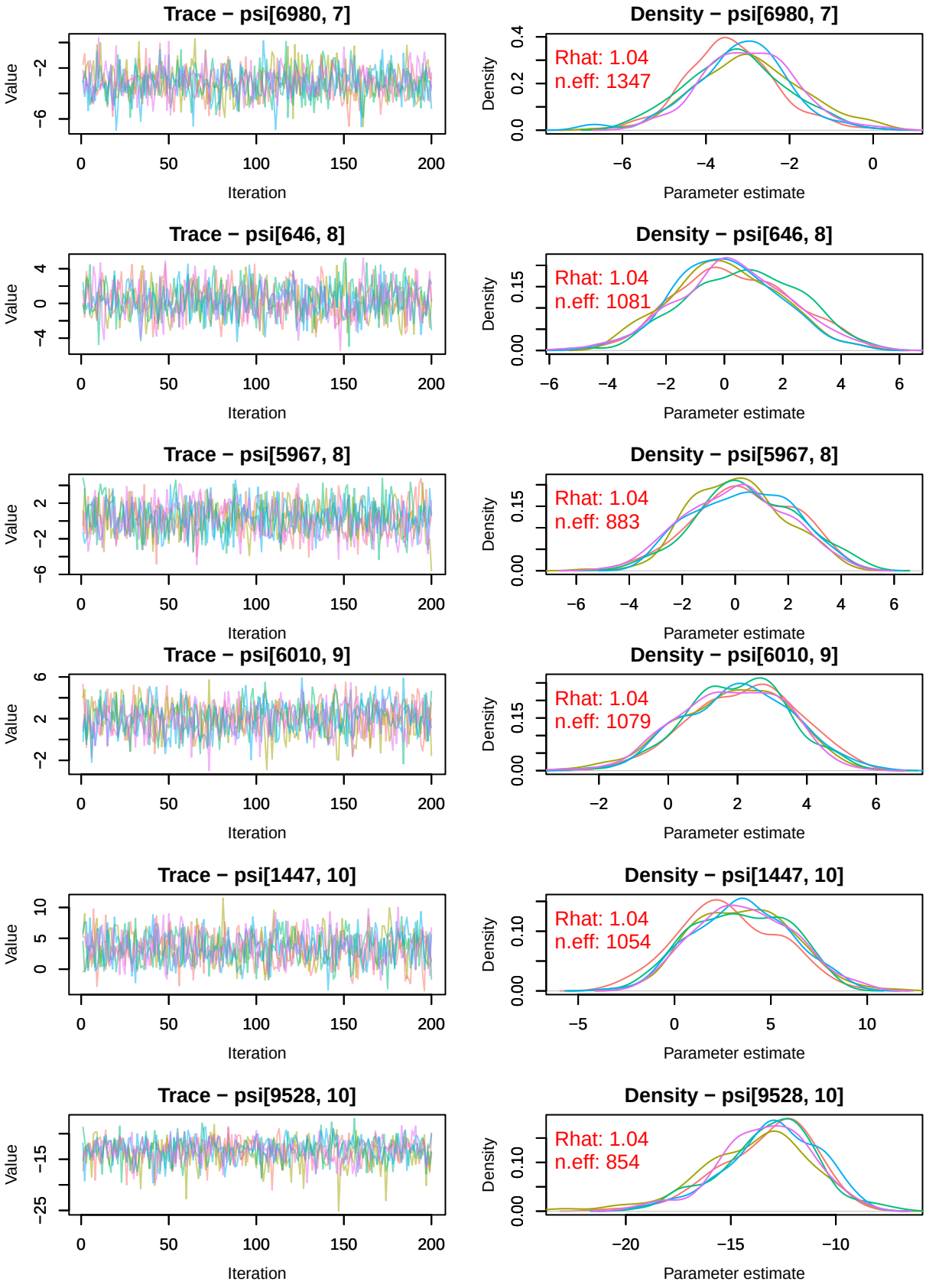

Trace − theta[181, 4]
Density − theta[181, 4]
Rhat: 1.04
n.eff: 932
Trace − theta[206, 5]
Density − theta[206, 5]
Rhat: 1.04
n.eff: 961
Trace − theta[253, 5]
Density − theta[253, 5]
Rhat: 1.04
n.eff: 589
Trace − theta[511, 5]
Density − theta[511, 5]
Rhat: 1.04
n.eff: 422
Trace − theta[301, 7]
Density − theta[301, 7]
Rhat: 1.04
n.eff: 959
Trace − theta[15, 9]
Density − theta[15, 9]
Rhat: 1.04
n.eff: 265
Value
Iteration
Density
Parameter estimate


Trace − theta[15, 10]
Density − theta[15, 10]
Rhat: 1.05
n.eff: 351
Trace − theta[118, 10]
Density − theta[118, 10]
Rhat: 1.04
n.eff: 962
Trace − theta[15, 11]
Density − theta[15, 11]
Rhat: 1.05
n.eff: 382
Trace − theta[118, 11]
Density − theta[118, 11]
Rhat: 1.04
n.eff: 947
Trace − theta[526, 11]
Density − theta[526, 11]
Rhat: 1.05
n.eff: 535
Trace − theta[188, 12]
Density − theta[188, 12]
Rhat: 1.04
n.eff: 888
Value
Density
Iteration
Parameter estimate

```
test <- "psi"

MCMCtrace(object = salnimble3,
          pdf = FALSE, # no export to PDF
          ind = TRUE, # separate density lines per chain
          Rhat = TRUE,
          n.eff = TRUE,
          exact = TRUE,
          ISB = FALSE,
          params = rownames(salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]],
          ↪  test), ])[(salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]], test),
          ↪  6] > 1.03) | (salwinbugs3$summary[startsWith(labels(salwinbugs3$summary)[[1]],
          ↪  test), 7] < 200)])
```




Trace − psi[5153, 4]
Density − psi[5153, 4]
Rhat: 1.04
n.eff: 895
Trace − psi[2889, 5]
Density − psi[2889, 5]
Rhat: 1.04
n.eff: 864
Trace − psi[6808, 5]
Density − psi[6808, 5]
Rhat: 1.04
n.eff: 1178
Trace − psi[7078, 5]
Density − psi[7078, 5]
Rhat: 1.04
n.eff: 1139
Trace − psi[6246, 6]
Density − psi[6246, 6]
Rhat: 1.04
n.eff: 906
Trace − psi[5967, 7]
Density − psi[5967, 7]
Rhat: 1.04
n.eff: 919
Value
Density
Iteration
Parameter estimate


Trace − psi[6980, 7]
Density − psi[6980, 7]
Rhat: 1.04
n.eff: 1347
Trace − psi[646, 8]
Density − psi[646, 8]
Rhat: 1.04
n.eff: 1081
Trace − psi[5967, 8]
Density − psi[5967, 8]
Rhat: 1.04
n.eff: 883
Trace − psi[6010, 9]
Density − psi[6010, 9]
Rhat: 1.04
n.eff: 1079
Trace − psi[1447, 10]
Density − psi[1447, 10]
Rhat: 1.04
n.eff: 1054
Trace − psi[9528, 10]
Density − psi[9528, 10]
Rhat: 1.04
n.eff: 854
Value
Density
Iteration
Parameter estimate

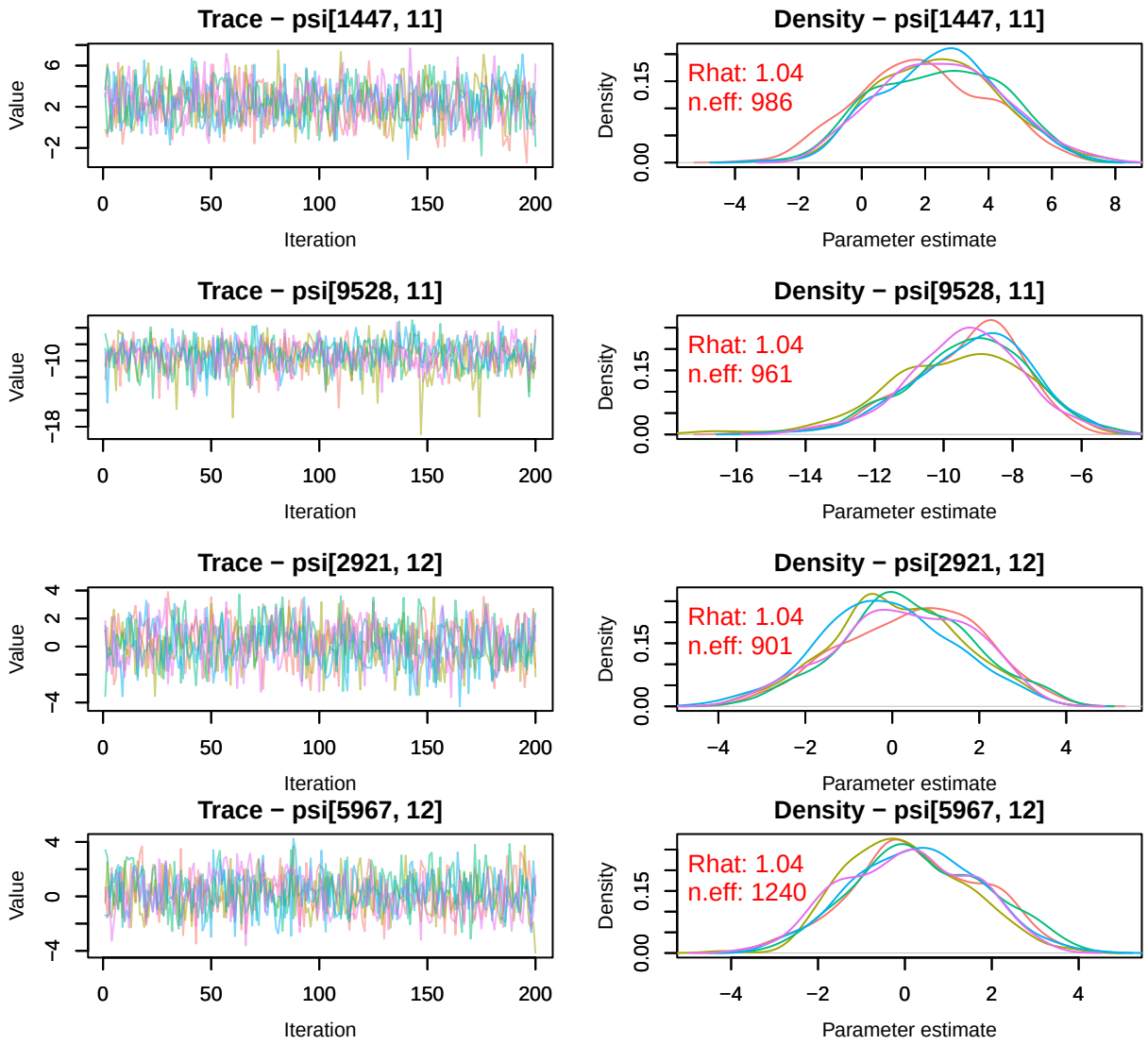


Figure 1: Maps of the RV - theta

```
# Cartography of the Region of Valencia
load(file.path("..", "data", "CartoCV.Rdata"))

n.sims <- salwinbugs3$n.sims
NMods <- 3
thetasim <- array(dim = c(n.sims, NMuni, NVars, NMods))
thetasim[, , , 1] <- salwinbugs1$sims.list$theta
thetasim[, , , 2] <- salwinbugs2$sims.list$theta
thetasim[, , , 3] <- salwinbugs3$sims.list$theta

selection <- 1:4
NSel <- length(selection)
thetasim <- thetasim[, , selection, ]

# Fiveteen equal-probability intervals
breaks <- c(min(apply(thetasim, 2:4, mean)) - 0.001, quantile(apply(thetasim, 2:4, mean), probs =
↪  seq(1/15, 14/15, length.out = 14)), max(apply(thetasim, 2:4, mean)))
```

```r
breaks <- c(-2.20, -0.5, -0.35, -0.25, -0.20, -0.15, -0.10, -0.05,
            0.05, 0.10, 0.15, 0.20, 0.25, 0.35, 0.5, 2.20)

mod_labels <- c("Indep", "Corr", "CorrIRE")
for (Mod in 1:NMods) {
  for (Sel in 1:NSel) {
    carto_muni@data[[paste0(colnames(y)[selection[Sel]], "thetamean", mod_labels[Mod])]] <-
↪   cut(apply(thetasim[, , Sel, Mod], 2, mean), breaks = breaks, include.lowest = FALSE, right =
↪   TRUE)

    levels(carto_muni@data[[paste0(colnames(y)[selection[Sel]], "thetamean", mod_labels[Mod])]])
    ↪  <- gsub("<=", "\u2264", c("<= -0.50", "(-0.50, -0.35]", "(-0.35, -0.25]", "(-0.25,
    ↪  -0.20]",
```

```
  }
}

mod_labels <- c("CorrIRE", "Corr", "Indep")
all_labels <- matrix(nrow = NSel, ncol = NMods)
for (Sel in 1:NSel) {
  for (Mod in 1:NMods) {
    all_labels[Sel, Mod] <- paste0(labels[selection[Sel]], ".", mod_labels[Mod])
  }
}
all_labels <- as.character(all_labels)
```

```
spplot(carto_muni,
       c(colnames(carto_muni@data[, endsWith(colnames(carto_muni@data), "CorrIRE")]),
         colnames(carto_muni@data[, endsWith(colnames(carto_muni@data), "Corr")]),
         colnames(carto_muni@data[, endsWith(colnames(carto_muni@data), "Indep")])),
       names.attr = all_labels,
       col.regions = colorRampPalette(brewer.pal(7,'BrBG'))(15),
       cuts = 14,
       par.settings = list(layout.widths = list(left.padding = 0, right.padding = 0),
                           layout.heights = list(top.padding = 0, bottom.padding = 0),
                           axis.line = list(col = 'transparent')),
       strip = strip.custom(par.strip.text = list(cex = 0.95)),
       col = "black",
       lwd = 0.025,
       layout = c(NSel, NMods))
```

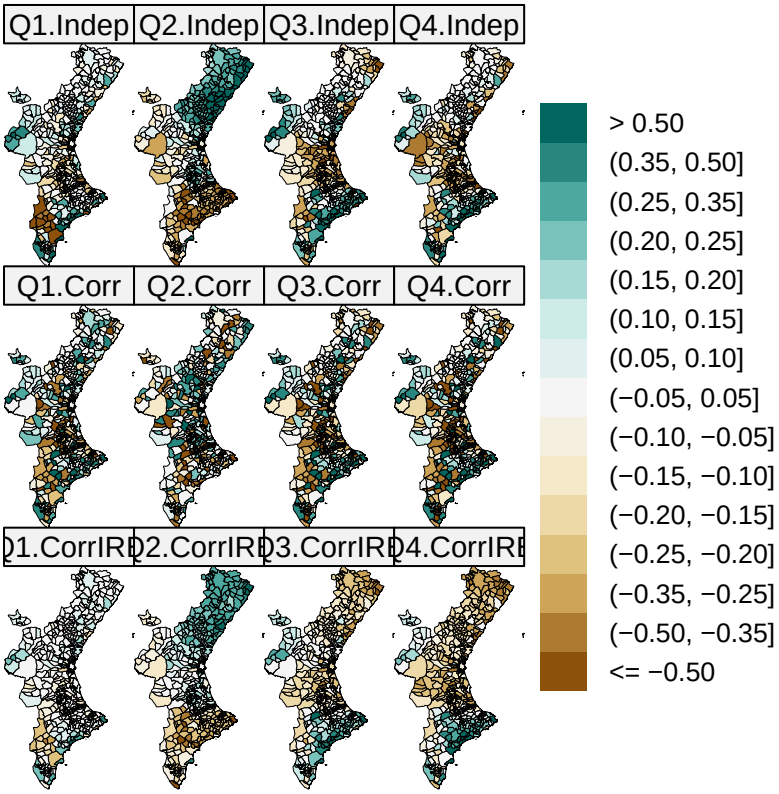


Figure 2: Maps of the RV - P(theta < 0 | y)

```
# Cartography of the Region of Valencia
load(file.path("..", "data", "CartoCV.Rdata"))

NMods <- 3
mod_labels <- c("Indep", "Corr", "CorrIRE")
thetasim <- array(dim = c(n.sims, NMuni, NVars, NMods))
thetasim[, , , 1] <- salwinbugs1$sims.list$theta
thetasim[, , , 2] <- salwinbugs2$sims.list$theta
thetasim[, , , 3] <- salwinbugs3$sims.list$theta

# Checking when theta is LESS than zero
stepsim <- array(dim = c(n.sims, NMuni, NVars, NMods))
for (Mod in 1:NMods) {
  for (Var in 1:NVars) {
    for (sim in 1:n.sims) {
      for (Muni in 1:NMuni) {
        stepsim[sim, Muni, Var, Mod] <- ifelse(thetasim[sim, Muni, Var, Mod] < 0, 1, 0)
      }
    }
  }
}

selection <- 1:4
NSel <- length(selection)
stepsim <- stepsim[, , selection, ]

mod_labels <- c("Indep", "Corr", "CorrIRE")
for (Mod in 1:NMods) {
  for (Sel in 1:NSel) {
    carto_muni@data[[paste0(colnames(y)[selection[Sel]], "probmean", mod_labels[Mod])]] <-
↪   apply(stepsim[, , Sel, Mod], 2, mean)
  }
}

kk <- unlist(carto_muni@data[, startsWith(colnames(carto_muni@data), "P8_")])

limit <- max(kk) + 0.01

mod_labels <- c("CorrIRE", "Corr", "Indep")
all_labels <- matrix(nrow = NSel, ncol = NMods)
for (Sel in 1:NSel) {
  for (Mod in 1:NMods) {
    all_labels[Sel, Mod] <- paste0(labels[selection[Sel]], ".", mod_labels[Mod])
  }
}
all_labels <- as.character(all_labels)
```

```
spplot(carto_muni,
       c(colnames(carto_muni@data[, endsWith(colnames(carto_muni@data), "CorrIRE")]),
         colnames(carto_muni@data[, endsWith(colnames(carto_muni@data), "Corr")]),
         colnames(carto_muni@data[, endsWith(colnames(carto_muni@data), "Indep")])),
       names.attr = all_labels,
       col.regions = colorRampPalette(brewer.pal(7,'RdYlGn'))(15)[15:1],
       par.settings = list(axis.line = list(col = 'transparent')),
       strip = strip.custom(par.strip.text = list(cex = 0.95)),
```

```
        col = "black",
        at = seq(0, limit, length.out = 16),
        lwd = 0.025,
        layout = c(NSel, NMods))
```

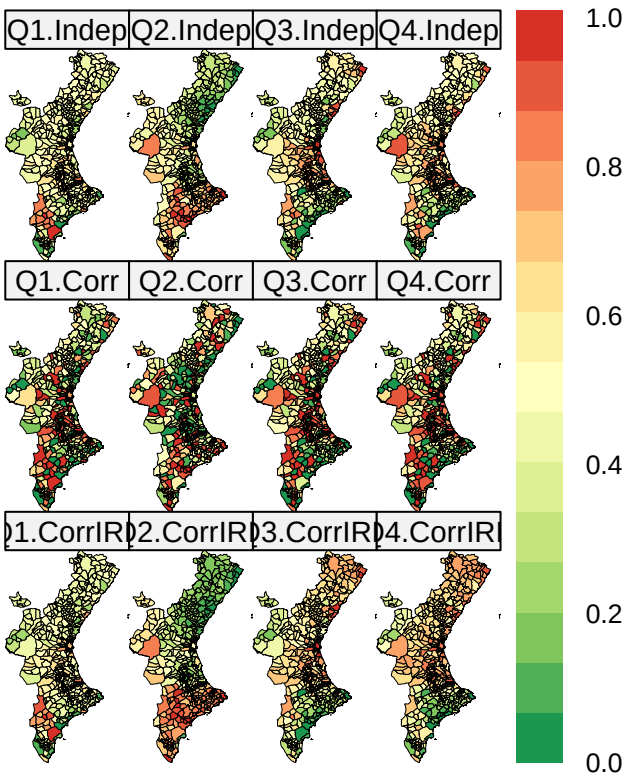


Figure 3: Individual correlation matrix for Model-Corr&IRE

```
SurveyMapping.Sigma.Resp <- function(salwinbugs) {

  n.sims <- salwinbugs$n.sims
  NVars <- dim(salwinbugs$sims.list$theta)[3]
  M.Resp <- salwinbugs$sims.list$M.Resp

  Sigma.Resp <- array(dim = c(n.sims, NVars, NVars))

  for (sim in 1:n.sims) {
    Sigma.Resp[sim, , ] <- t(M.Resp[sim, , ]) %*% M.Resp[sim, , ]
  }
  return(Sigma.Resp)
}

salwinbugs <- salwinbugs3

n.sims <- salwinbugs$n.sims
Sigma.Respsim <- SurveyMapping.Sigma.Resp(salwinbugs = salwinbugs)
Corr <- array(dim = c(n.sims, NVars, NVars))

# Set of the n.sims correlation matrices
```

```r
for (sim in 1:n.sims) {
  Corr[sim, , ] <- diag(diag(Sigma.Respsim[sim, , ])^(-1/2)) %*% Sigma.Respsim[sim, , ] %*%
↪  diag(diag(Sigma.Respsim[sim, , ])^(-1/2))
}

Corr.mean <- matrix(ncol = NVars, nrow = NVars)
Corr.quantileL <- matrix(ncol = NVars, nrow = NVars)
Corr.quantileU <- matrix(ncol = NVars, nrow = NVars)
Sigma.Respmean <- matrix(ncol = NVars, nrow = NVars)
for (Var1 in 1:NVars) {
  for (Var2 in 1:NVars) {
    Corr.mean[Var1, Var2] <- mean(Corr[, Var1, Var2])
    Corr.quantileL[Var1, Var2] <- quantile(Corr[, Var1, Var2], probs = 0.025)
    Corr.quantileU[Var1, Var2] <- quantile(Corr[, Var1, Var2], probs = 0.975)
    Sigma.Respmean[Var1, Var2] <- mean(Sigma.Respsim[, Var1, Var2])
  }
}

rm(Corr); rm(Sigma.Respsim)

Corr.mean <- data.frame(Corr.mean); rownames(Corr.mean) <- labels; colnames(Corr.mean) <- labels

orden <- corrMatOrder(as.matrix(Corr.mean), order = "hclust", hclust.method = "ward.D2")

Corr.mean.orden <- as.matrix(Corr.mean)
Corr.mean.orden <- Corr.mean.orden[orden, orden]

Corr.quantileL <- data.frame(Corr.quantileL); rownames(Corr.quantileL) <- labels;
colnames(Corr.quantileL) <- labels

Corr.quantileL.orden <- as.matrix(Corr.quantileL)
Corr.quantileL.orden <- Corr.quantileL.orden[orden, orden]

Corr.quantileU <- data.frame(Corr.quantileU); rownames(Corr.quantileU) <- labels;
colnames(Corr.quantileU) <- labels

Corr.quantileU.orden <- as.matrix(Corr.quantileU)
Corr.quantileU.orden <- Corr.quantileU.orden[orden, orden]

Relevance <- matrix(as.numeric(Corr.quantileL.orden > 0 | Corr.quantileU.orden < 0), ncol =
↪  NVars, nrow = NVars, byrow = FALSE)
colnames(Relevance) <- rownames(Relevance) <-  colnames(Corr.mean.orden)
Relevance <- (Relevance - 1) * (-1)
for (Var in 1:NVars) { Relevance[Var, Var] <- 1 }
```

```r
# First: ellipses in lower triangular
corrplot(as.matrix(Corr.mean.orden),
         type = "lower", method = "ellipse",
         p.mat = Relevance, sig.level = 0.05, insig = "label_sig",
         pch.cex = 1.5, pch.col = "grey20",
         addCoef.col = "black", number.cex = 0.8,
         tl.pos = "d", tl.cex = 0.9, cl.pos = "r")

# Second: CI in upper triangular
corrplot(as.matrix(Corr.mean.orden),
         type = "upper", method = "square",
         diag = FALSE, add = TRUE, cl.pos = "n",
         plotCI = "rect", lowCI = as.matrix(Corr.quantileL.orden),
```

```
        uppCI = as.matrix(Corr.quantileU.orden), rect.col = "navy", tl.pos = "n")
```

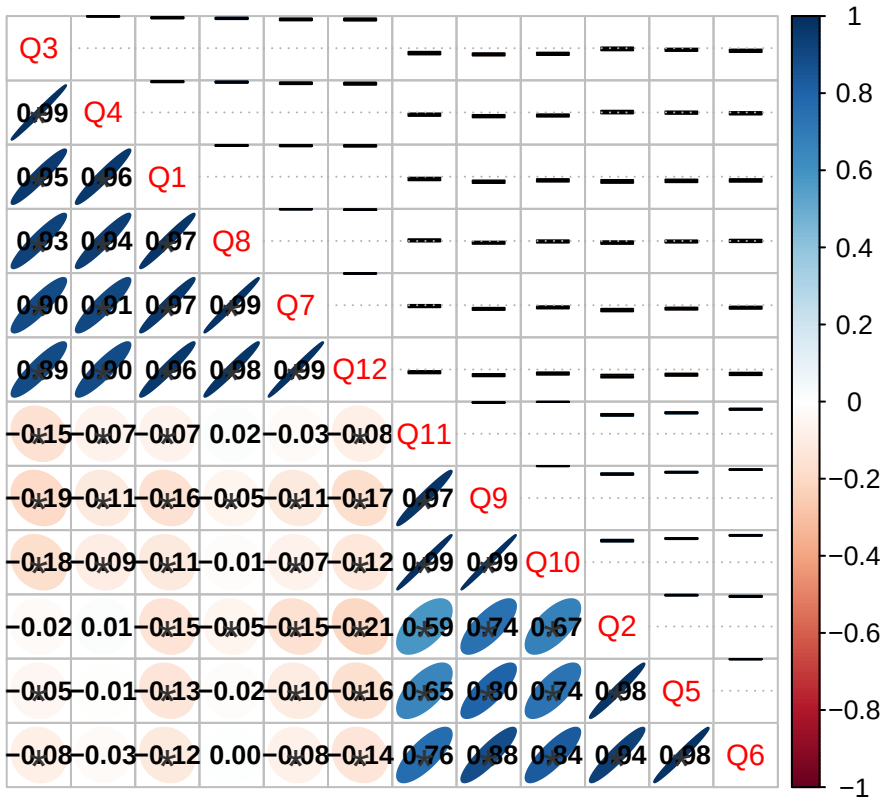


Figure 4: Spatial correlation matrix for Model-Corr&IRE

```
SurveyMapping.Sigma.Muni <- function(salwinbugs) {

  n.sims <- salwinbugs$n.sims
  NVars <- dim(salwinbugs$sims.list$theta)[3]
  M.Muni <- salwinbugs$sims.list$M.Muni

  Sigma.Muni <- array(dim = c(n.sims, NVars, NVars))

  for (sim in 1:n.sims) {
    Sigma.Muni[sim, , ] <- t(M.Muni[sim, , ]) %*% M.Muni[sim, , ]
  }
  return(Sigma.Muni)
}

salwinbugs <- salwinbugs3
n.sims <- salwinbugs$n.sims
Sigma.Munisim <- SurveyMapping.Sigma.Muni(salwinbugs = salwinbugs)
Corr <- array(dim = c(n.sims, NVars, NVars))

# Set of the n.sims correlation matrices
for (sim in 1:n.sims) {
  Corr[sim, , ] <- diag(diag(Sigma.Munisim[sim, , ])^(-1/2)) %*% Sigma.Munisim[sim, , ] %*%
↪  diag(diag(Sigma.Munisim[sim, , ])^(-1/2))
```

```
}

Corr.mean <- matrix(ncol = NVars, nrow = NVars)
Corr.quantileL <- matrix(ncol = NVars, nrow = NVars)
Corr.quantileU <- matrix(ncol = NVars, nrow = NVars)
Sigma.Munimean <- matrix(ncol = NVars, nrow = NVars)
for (Var1 in 1:NVars) {
  for (Var2 in 1:NVars) {
    Corr.mean[Var1, Var2] <- mean(Corr[, Var1, Var2])
    Corr.quantileL[Var1, Var2] <- quantile(Corr[, Var1, Var2], probs = 0.025)
    Corr.quantileU[Var1, Var2] <- quantile(Corr[, Var1, Var2], probs = 0.975)
    Sigma.Munimean[Var1, Var2] <- mean(Sigma.Munisim[, Var1, Var2])
  }
}

rm(Corr); rm(Sigma.Munisim)

Corr.mean <- data.frame(Corr.mean); rownames(Corr.mean) <- labels;
colnames(Corr.mean) <- labels

Corr.mean.orden <- as.matrix(Corr.mean)
Corr.mean.orden <- Corr.mean.orden[orden, orden]

Corr.quantileL <- data.frame(Corr.quantileL); rownames(Corr.quantileL) <- labels;
colnames(Corr.quantileL) <- labels

Corr.quantileL.orden <- as.matrix(Corr.quantileL)
Corr.quantileL.orden <- Corr.quantileL.orden[orden, orden]

Corr.quantileU <- data.frame(Corr.quantileU); rownames(Corr.quantileU) <- labels;
colnames(Corr.quantileU) <- labels

Corr.quantileU.orden <- as.matrix(Corr.quantileU)
Corr.quantileU.orden <- Corr.quantileU.orden[orden, orden]

Relevance <- matrix(as.numeric(Corr.quantileL.orden > 0 | Corr.quantileU.orden < 0), ncol =
↪  NVars, nrow = NVars, byrow = FALSE)
colnames(Relevance) <- rownames(Relevance) <-  colnames(Corr.mean.orden)
Relevance <- (Relevance - 1) * (-1)
for (Var in 1:NVars) { Relevance[Var, Var] <- 1 }
```

```
# First: ellipses in lower triangular
corrplot(as.matrix(Corr.mean.orden),
         type = "lower", method = "ellipse",
         p.mat = Relevance, sig.level = 0.05, insig = "label_sig",
         pch.cex = 1.5, pch.col = "grey20",
         addCoef.col = "black", number.cex = 0.8,
         tl.pos = "d", tl.cex = 0.9, cl.pos = "r")

# Second: CI in upper triangular
corrplot(as.matrix(Corr.mean.orden),
         type = "upper", method = "square",
         diag = FALSE, add = TRUE, cl.pos = "n",
         plotCI = "rect", lowCI = as.matrix(Corr.quantileL.orden),
         uppCI = as.matrix(Corr.quantileU.orden), rect.col = "navy", tl.pos = "n")
```

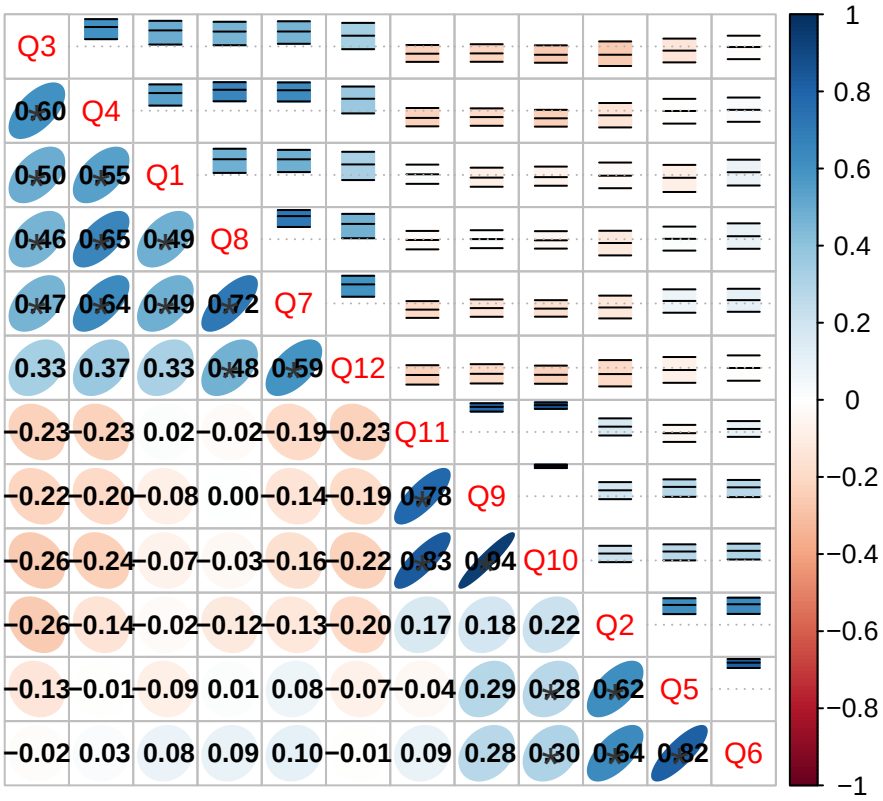


Figure 5: Maps of the RV - PCA

```
salwinbugs <- salwinbugs3
thetasim <- salwinbugs$sims.list$theta
thetamean <- apply(thetasim, 2:3, mean)
acp <- princomp(thetamean, cor = TRUE)
acp$loadings[, 1:3]
```

```
##            Comp.1     Comp.2      Comp.3
##  [1,]  0.1500401 0.46181484  0.15012780
##  [2,] -0.2984278 0.23852458 -0.28661029
##  [3,]  0.3397215 0.16086642  0.04076247
##  [4,]  0.3412119 0.21268818  0.04785622
##  [5,] -0.2052693 0.37894367 -0.45563317
##  [6,] -0.2249631 0.43823141 -0.35853430
##  [7,]  0.3246205 0.27712025  0.02077444
##  [8,]  0.2802306 0.35851188  0.22584355
##  [9,] -0.2963037 0.20552491  0.39755914
## [10,] -0.3065521 0.20412406  0.37866329
## [11,] -0.2994604 0.15756273  0.44765771
## [12,]  0.3293849 0.09494317 -0.04407364
```

```
thetaComp <- acp$scores[, 1:3]
cor(thetaComp)
```

```
##               Comp.1       Comp.2        Comp.3
## Comp.1  1.000000e+00 3.307950e-17 -4.689014e-16
## Comp.2  3.307950e-17 1.000000e+00  7.081673e-16
## Comp.3 -4.689014e-16 7.081673e-16  1.000000e+00
```

```
carto_muni@data$ACP1 <- thetaComp[, 1]

# Fiveteen equal-probability intervals
breaks <- c(min(carto_muni@data$ACP1) - 0.001, quantile(carto_muni@data$ACP1, probs = seq(1/15,
↪  14/15, length.out = 14)), max(carto_muni@data$ACP1))
breaks <- c(-17.50, -3.00, -2.50, -2.00, -1.50, -1.00, -0.50, -0.25,
            0.25, 0.50, 1.00, 1.50, 2.00, 2.50, 3.00, 17.50)

carto_muni@data$ACP1 <- cut(carto_muni@data$ACP1, breaks = breaks, include.lowest = FALSE, right
↪  = TRUE)
levels(carto_muni@data$ACP1) <- c("Worse S1", "Better S2, ", "  ", "   ", "    ",
                                  "     ", "      ", "       ", "        ",
                                  "         ", "          ", "           ",
                                  "               ", "Worse S2", "Better S1, ")

carto_muni@data$ACP2 <- thetaComp[, 2]

# Fiveteen equal-probability intervals
breaks <- c(min(carto_muni@data$ACP2) - 0.001, quantile(carto_muni@data$ACP2, probs = seq(1/15,
↪  14/15, length.out = 14)), max(carto_muni@data$ACP2))

carto_muni@data$ACP2 <- cut(carto_muni@data$ACP2, breaks = breaks, include.lowest = FALSE, right
↪  = TRUE)
levels(carto_muni@data$ACP2) <- c("Worse MH", " ", "  ", "   ", "    ",
                                  "     ", "      ", "       ", "        ",
                                  "         ", "          ", "           ",
                                  "               ", "                ", "Better MH")
rm(list = c("thetasim", "stepsim"))
```

```
grid.arrange(spplot(carto_muni,
                    c("ACP1"),
                    main = expression("(A)
                    ↪  "),
                    col.regions = colorRampPalette(brewer.pal(7,'RdYlBu'))(15),
                    cuts = 14,
                    par.settings = list(axis.line = list(col = 'transparent')),
                    strip = strip.custom(par.strip.text = list(cex = 0.95)),
                    col = "black",
                    lwd = 0.025),
             spplot(carto_muni,
                    c("ACP2"),
                    main = expression("(B)
                    ↪  "),
                    col.regions = colorRampPalette(brewer.pal(7,'Blues'))(15),
                    cuts = 14,
                    par.settings = list(axis.line = list(col = 'transparent')),
                    strip = strip.custom(par.strip.text = list(cex = 0.95)),
                    col = "black",
                    lwd = 0.025),
             ncol = 2)
```

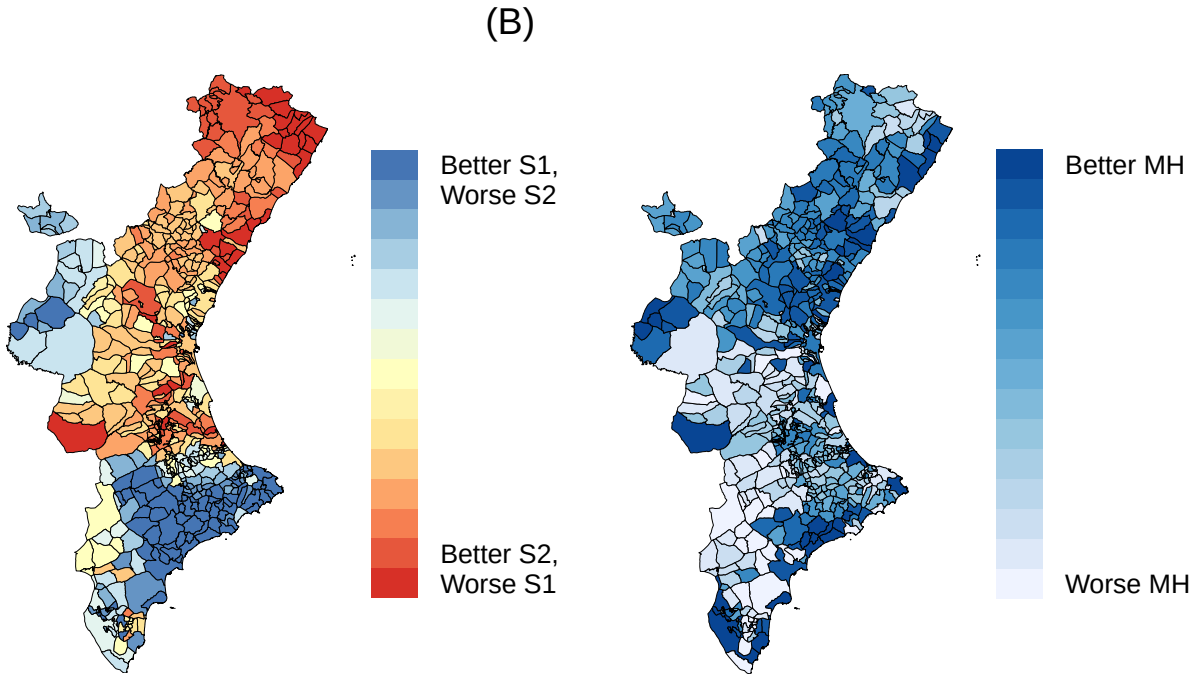


Table 2: Model assessment

```
# Model-Indep
# SurveyMapping.prlevels:
# - computes the n.sims simulated probabilities for each respondent

SurveyMapping.prlevels <- function(salwinbugs) {

  n.sims <- salwinbugs$n.sims
  n.chains <- salwinbugs$n.chains
  p.gamma <- array(dim = c(n.sims, NResp, NVars, NCats - 1))
  prlevels <- array(dim = c(n.sims, NResp, NVars, NCats))
  for (sim in 1:n.sims) {
    for (Resp in 1:NResp) {
      for (Var in 1:NVars) {
        for (Cat in 1:(NCats - 1)) {
          p.gamma[sim, Resp, Var, Cat] <-
            ilogit(salwinbugs$sims.list$kappa[sim, sex[Resp], age[Resp], Cat, Var] +
                     salwinbugs$sims.list$sd.theta[sim, Var] * salwinbugs$sims.list$theta[sim,
↪  muni[Resp], Var])
        }

        prlevels[sim, Resp, Var, 1] <- p.gamma[sim, Resp, Var, 1]
        prlevels[sim, Resp, Var, NCats] <- 1 - p.gamma[sim, Resp, Var, NCats - 1]

        for (Cat in 2:(NCats - 1)) {
          prlevels[sim, Resp, Var, Cat] <-
            p.gamma[sim, Resp, Var, Cat] - p.gamma[sim, Resp, Var, Cat-1]
        }
      }
    }

    if (sim %in% c(1, seq(n.sims/n.chains, n.sims, n.sims/n.chains))) {
```

```
      cat(sim, "of", n.sims, "simulations", "\n")
    } else {}
  }

  return(prlevels)
}

# # This may take a minute
# prlevels <- SurveyMapping.prlevels(salwinbugs = salwinbugs1)

# Model-Corr
# SurveyMapping.prlevels:
# - computes the n.sims simulated probabilities for each respondent

SurveyMapping.prlevels <- function(salwinbugs) {

  n.sims <- salwinbugs$n.sims
  n.chains <- salwinbugs$n.chains
  p.gamma <- array(dim = c(n.sims, NResp, NVars, NCats - 1))
  prlevels <- array(dim = c(n.sims, NResp, NVars, NCats))
  for (sim in 1:n.sims) {
    for (Resp in 1:NResp) {
      for (Var in 1:NVars) {
        for (Cat in 1:(NCats - 1)) {
          p.gamma[sim, Resp, Var, Cat] <-
            ilogit(salwinbugs$sims.list$kappa[sim, sex[Resp], age[Resp], Cat, Var] +
                     salwinbugs$sims.list$theta[sim, muni[Resp], Var])
        }

        prlevels[sim, Resp, Var, 1] <- p.gamma[sim, Resp, Var, 1]
        prlevels[sim, Resp, Var, NCats] <- 1 - p.gamma[sim, Resp, Var, NCats - 1]

        for (Cat in 2:(NCats - 1)) {
          prlevels[sim, Resp, Var, Cat] <-
            p.gamma[sim, Resp, Var, Cat] - p.gamma[sim, Resp, Var, Cat-1]
        }
      }
    }

    if (sim %in% c(1, seq(n.sims/n.chains, n.sims, n.sims/n.chains))) {
      cat(sim, "of", n.sims, "simulations", "\n")
    } else {}
  }

  return(prlevels)
}

# # This may take a minute
# prlevels <- SurveyMapping.prlevels(salwinbugs = salwinbugs2)

# Model-Corr&IRE
# SurveyMapping.prlevels:
# - computes the n.sims simulated probabilities for each respondent

SurveyMapping.prlevels <- function(salwinbugs) {

  n.sims <- salwinbugs$n.sims
  n.chains <- salwinbugs$n.chains
  p.gamma <- array(dim = c(n.sims, NResp, NVars, NCats - 1))
```

```r
  prlevels <- array(dim = c(n.sims, NResp, NVars, NCats))
  for (sim in 1:n.sims) {
    for (Resp in 1:NResp) {
      for (Var in 1:NVars) {
        for (Cat in 1:(NCats - 1)) {
          p.gamma[sim, Resp, Var, Cat] <-
            ilogit(salwinbugs$sims.list$kappa[sim, sex[Resp], age[Resp], Cat, Var] +
                     salwinbugs$sims.list$theta[sim, muni[Resp], Var] +
                     salwinbugs$sims.list$psi[sim, Resp, Var])
        }

        prlevels[sim, Resp, Var, 1] <- p.gamma[sim, Resp, Var, 1]
        prlevels[sim, Resp, Var, NCats] <- 1 - p.gamma[sim, Resp, Var, NCats - 1]

        for (Cat in 2:(NCats - 1)) {
          prlevels[sim, Resp, Var, Cat] <-
            p.gamma[sim, Resp, Var, Cat] - p.gamma[sim, Resp, Var, Cat-1]
        }
      }
    }

    if (sim %in% c(1, seq(n.sims/n.chains, n.sims, n.sims/n.chains))) {
      cat(sim, "of", n.sims, "simulations", "\n")
    } else {}
  }

  return(prlevels)
}

# # This may take a minute
# prlevels <- SurveyMapping.prlevels(salwinbugs = salwinbugs3)

# Validation

# Sample size of each municipality, sex and age group:
sample <- array(dim = c(NMuni, NSex, NAges))
for (Muni in 1:NMuni) {
  for (SexGroup in 1:NSex) {
    for (AgeGroup in 1:NAges) {
      sample[Muni, SexGroup, AgeGroup] <- sum(muni == Muni & sex == SexGroup & age == AgeGroup)
    }
  }
}

SurveyMapping.Validation <- function(prlevels, Muni) {

  NSamp <- sum(sample[Muni, , ])
  index <- which(muni == Muni)
  realvalue <- matrix(nrow = NVars, ncol = NCats)
  predictive <- array(dim = c(n.sims, NSamp, NVars))
  predictive.muni <- array(dim = c(n.sims, NVars, NCats))
  for (sim in 1:n.sims) {
    for (Var in 1:NVars) {
      for (Resp in 1:NSamp) {
        predictive[sim, Resp, Var] <- which(
          rmultinom(n = 1,
                    size = 1,
                    prob = prlevels[sim, index[Resp], Var, ]) == 1)
      }
```

```
      predictive.muni[sim, Var, ] <- table(factor(predictive[sim, , Var], levels =
↪  1:NCats))/NSamp * 100
    }

    if (sim %in% c(1, seq(n.sims/n.chains, n.sims, n.sims/n.chains))) {
      cat(sim, "of", n.sims, "simulations", "\n")
    } else {}
  }

  posteriormean <- round(apply(predictive.muni, 2:3, mean), 2)
  PInterval0.025 <- round(apply(predictive.muni, 2:3, quantile, prob = 0.025), 2)
  PInterval0.975 <- round(apply(predictive.muni, 2:3, quantile, prob = 0.975), 2)
  for (Var in 1:NVars) {
    realvalue[Var, ] <- round(table(factor(y[muni == Muni, Var], levels = 1:NCats))/NSamp * 100,
↪  2)
  }
  return(list("mean" = posteriormean,
              "PI" = list("lower" = PInterval0.025, "upper" = PInterval0.975),
              "real" = realvalue))
  return(predictive.muni)
}

# Four municipalities with the largest population in the RV
Munis <- order(apply(sample, 1, sum), decreasing = TRUE)[1:6]

# validation <- list()
# for (Muni in 1:length(Munis)) {
#   set.seed(9747783)
#   validation[[Muni]] <- SurveyMapping.Validation(prlevels = prlevels, Muni = Munis[Muni])
# }

# saveRDS(validation, file = file.path("results", "multi-2022-nimble-MH-indep-assessment.rds"))
# saveRDS(validation, file = file.path("results", "multi-2022-nimble-MH-corr-assessment.rds"))
# saveRDS(validation, file = file.path("results",
↪  "multi-2022-nimble-MH-corr-ire-assessment.rds"))

indep <- readRDS(file = file.path("..", "results", "multi-2022-nimble-MH-indep-assessment.rds"))
corr <- readRDS(file = file.path("..", "results", "multi-2022-nimble-MH-corr-assessment.rds"))
corrire <- readRDS(file = file.path("..", "results",
↪  "multi-2022-nimble-MH-corr-ire-assessment.rds"))

# Valencia results
# Model-Indep
indep[[1]]$mean[5, ]; indep[[1]]$PI$lower[5, ]; indep[[1]]$PI$upper[5, ]; indep[[1]]$real[5, ]
```

```
## [1] 17.81 56.02 20.73  5.43
```

```
## [1] 15.18 52.91 18.00  4.00
```

```
## [1] 20.55 59.27 23.73  7.00
```

```
## [1] 16.91 58.55 20.18  4.27
```

```
# Model-Corr
corr[[1]]$mean[5, ]; corr[[1]]$PI$lower[5, ]; corr[[1]]$PI$upper[5, ]; corr[[1]]$real[5, ]
```

```
## [1] 17.18 56.58 20.91  5.32
```

```
## [1] 14.45 53.45 18.09  4.00
```

```
## [1] 19.91 59.82 23.64  6.82
```

```
## [1] 16.91 58.55 20.18  4.27
```

```
# Model-Corr&IRE
corrire[[1]]$mean[5, ]; corrire[[1]]$PI$lower[5, ]; corrire[[1]]$PI$upper[5, ];
↪  corrire[[1]]$real[5, ]
```

```
## [1] 17.41 57.31 20.50  4.78
```

```
## [1] 15.73 54.82 18.45  3.91
```

```
## [1] 19.09 59.64 22.55  5.73
```

```
## [1] 16.91 58.55 20.18  4.27
```