R code for the statistical analysis of the paper: Bayesian modeling of spatial ordinal data from health surveys

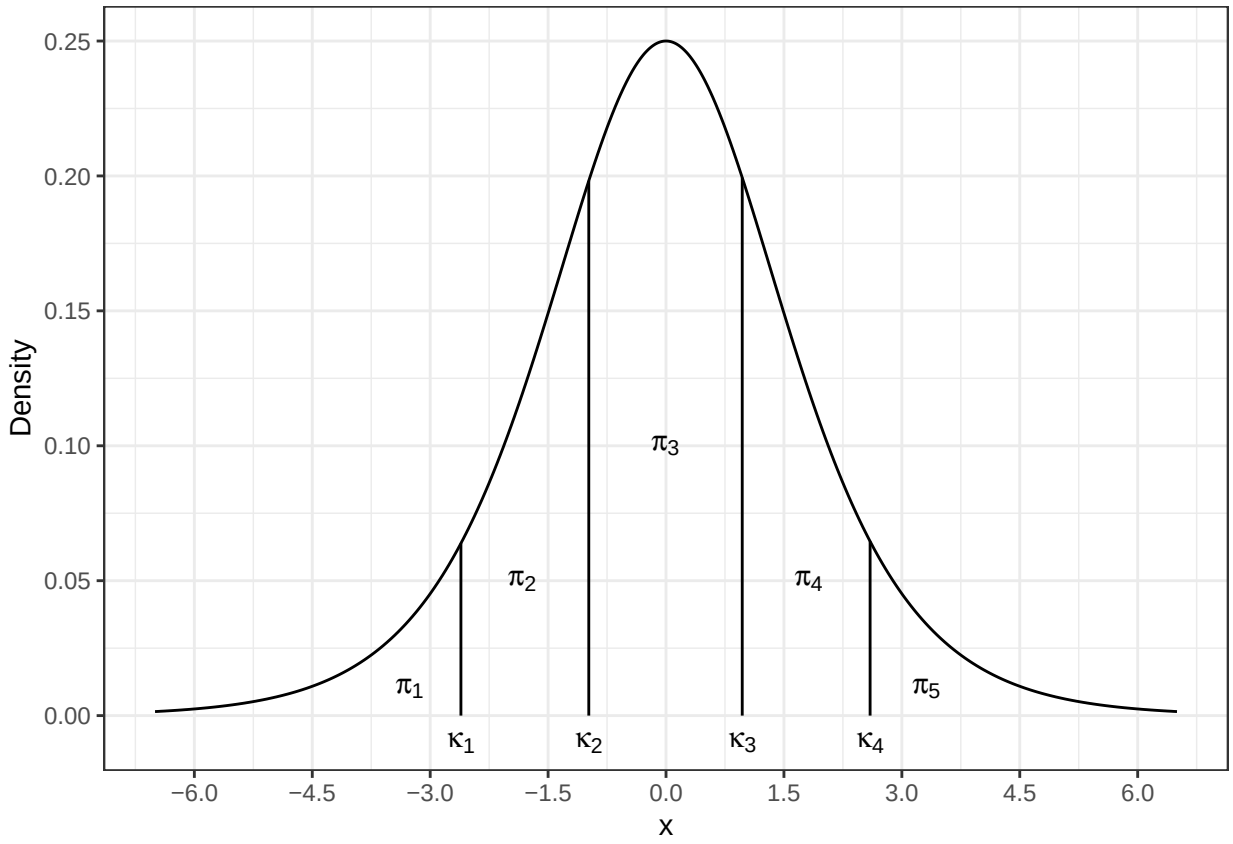
Beltrán-Sánchez, MA[1]; Martínez-Beneito, MA[2]; Corberán-Vallet, A[3]
[1]angel.beltran@uv.es; [2]miguel.a.martinez@uv.es; [3]ana.corberan@uv.es
Department of Statistics and Operations Research, University of Valencia, Spain

Required packages

```
## install.packages("pacman")
pacman::p_load(foreign, readxl, faraway, spdep, ggplot2, RColorBrewer, graphics,
               sp, ggpubr, leaflet, nimble, ggmcmc, extraDistr, parallel, MCMCvis,
               gridExtra, install = FALSE)
```

Figure 1: `kappa[1:(NCats-1)]` interpretation as cut points

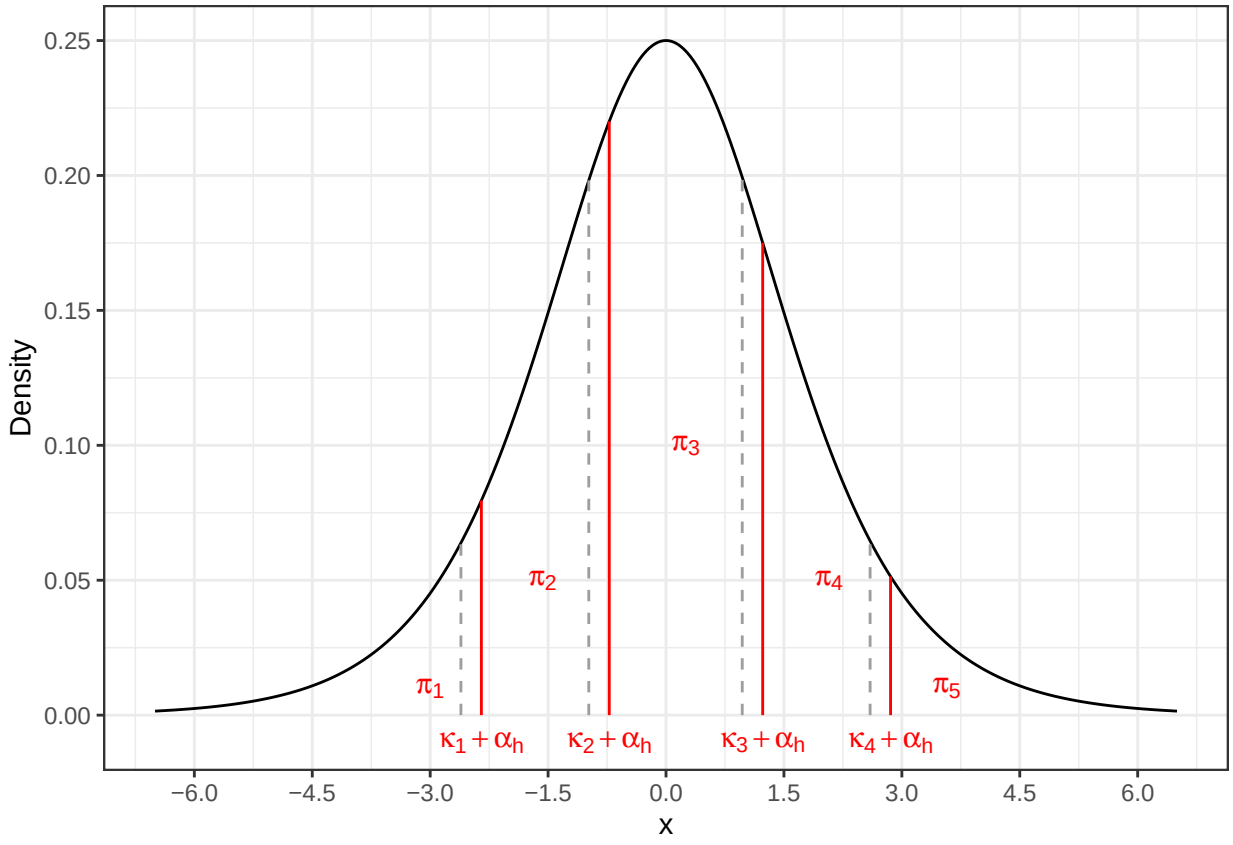
```
x <- seq(from = -6.5, to = 6.5, length.out = 1000)
y <- dlogis(x)

df <- data.frame("x" = x, "y" = y)
lines <- data.frame("init" = c(x[300], x[425], x[575], x[700]),
                    "end" = c(y[300], y[425], y[575], y[700]))
p <- ggplot() + geom_line(data = df, mapping = aes(x = x, y = y)) +
  geom_segment(data = lines, mapping = aes(x = init, y = 0, xend = init, yend = end)) +
  annotate("text", x = x[300], y = -0.01, label = expression(kappa[1]), size = 4) +
  annotate("text", x = x[425], y = -0.01, label = expression(kappa[2]), size = 4) +
  annotate("text", x = x[575], y = -0.01, label = expression(kappa[3]), size = 4) +
  annotate("text", x = x[700], y = -0.01, label = expression(kappa[4]), size = 4) +

  annotate("text", x = x[250], y = 0.01, label = expression(pi[1]), size = 4) +
  annotate("text", x = x[360], y = 0.05, label = expression(pi[2]), size = 4) +
  annotate("text", x = x[500], y = 0.10, label = expression(pi[3]), size = 4) +
  annotate("text", x = x[640], y = 0.05, label = expression(pi[4]), size = 4) +
  annotate("text", x = x[755], y = 0.01, label = expression(pi[5]), size = 4) +
  coord_cartesian(ylim = c(-0.0075, 0.25)) +
  theme_bw() +
  scale_x_continuous(breaks = seq(-7.5, 7.5, by = 1.5)) +
  labs(y = "Density")

# Warnings: do not pay attention
p
```



alpha[h] positive effect

```
df <- data.frame("x" = x, "y" = y)
effect <- 20
lines <- data.frame("initgrey" = c(x[300], x[425], x[575], x[700]),
                    "endgrey" = c(y[300], y[425], y[575], y[700]),
                    "initcol" = c(x[300 + effect], x[425 + effect], x[575 + effect], x[700 +
                    ↪  effect]),
                    "endcol" = c(y[300 + effect], y[425 + effect], y[575 + effect], y[700 +
                    ↪  effect]))
p <- ggplot() + geom_line(data = df, mapping = aes(x = x, y = y)) +
  geom_segment(data = lines, mapping = aes(x = initgrey, y = 0, xend = initgrey, yend = endgrey),
               col = "gray62", lty = 2, lwd = 0.50) +

  # Effect
  geom_segment(data = lines, mapping = aes(x = initcol, y = 0, xend = initcol, yend = endcol),
  ↪  col = "red") +
  annotate("text", x = x[300 + effect], y = -0.01, label = expression(kappa[1] + alpha[h]), size
  ↪  = 4, col = "red") +
  annotate("text", x = x[425 + effect], y = -0.01, label = expression(kappa[2] + alpha[h]), size
  ↪  = 4, col = "red") +
  annotate("text", x = x[575 + effect], y = -0.01, label = expression(kappa[3] + alpha[h]), size
  ↪  = 4, col = "red") +
  annotate("text", x = x[700 + effect], y = -0.01, label = expression(kappa[4] + alpha[h]), size
  ↪  = 4, col = "red") +

  annotate("text", x = x[250 + effect], y = 0.01, label = expression(pi[1]), size = 4, col =
  ↪  "red") +
```

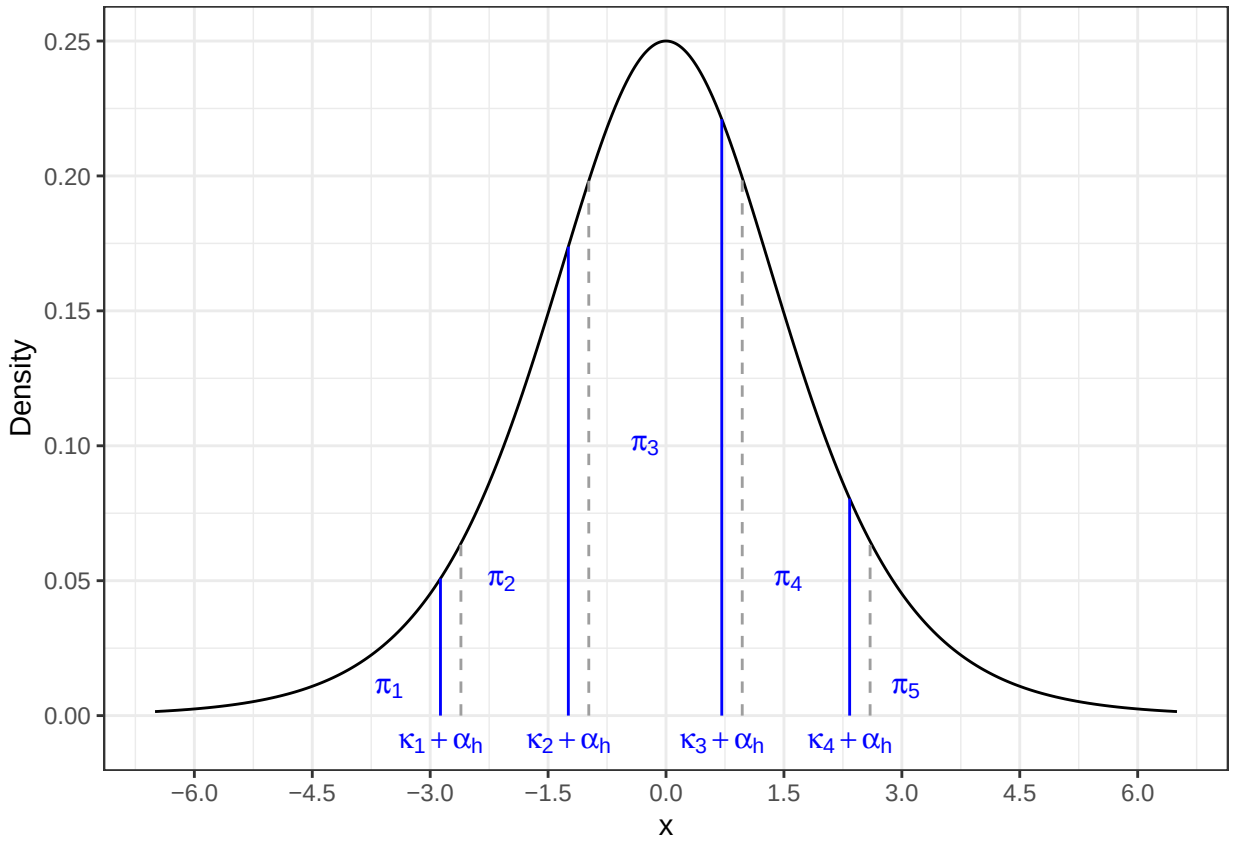
```
  annotate("text", x = x[360 + effect], y = 0.05, label = expression(pi[2]), size = 4, col =
  ↪  "red") +
  annotate("text", x = x[500 + effect], y = 0.10, label = expression(pi[3]), size = 4, col =
  ↪  "red") +
  annotate("text", x = x[640 + effect], y = 0.05, label = expression(pi[4]), size = 4, col =
  ↪  "red") +
  annotate("text", x = x[755 + effect], y = 0.01, label = expression(pi[5]), size = 4, col =
  ↪  "red") +
  coord_cartesian(ylim = c(-0.0075, 0.25)) +
  theme_bw() +
  scale_x_continuous(breaks = seq(-7.5, 7.5, by = 1.5)) +
  labs(y = "Density")

# Warnings: do not pay attention
p
```



alpha[h] negative effect

```
df <- data.frame("x" = x, "y" = y)
effect <- -20
lines <- data.frame("initgrey" = c(x[300], x[425], x[575], x[700]),
                    "endgrey" = c(y[300], y[425], y[575], y[700]),
                    "initcol" = c(x[300 + effect], x[425 + effect], x[575 + effect], x[700 +
                    ↪  effect]),
                    "endcol" = c(y[300 + effect], y[425 + effect], y[575 + effect], y[700 +
                    ↪  effect]))
```

```
p <- ggplot() + geom_line(data = df, mapping = aes(x = x, y = y)) +
  geom_segment(data = lines, mapping = aes(x = initgrey, y = 0, xend = initgrey, yend = endgrey),
               col = "gray62", lty = 2, lwd = 0.50) +

  # Effect
  geom_segment(data = lines, mapping = aes(x = initcol, y = 0, xend = initcol, yend = endcol),
  ↪  col = "blue") +
  annotate("text", x = x[300 + effect], y = -0.01, label = expression(kappa[1] + alpha[h]), size
  ↪  = 4, col = "blue") +
  annotate("text", x = x[425 + effect], y = -0.01, label = expression(kappa[2] + alpha[h]), size
  ↪  = 4, col = "blue") +
  annotate("text", x = x[575 + effect], y = -0.01, label = expression(kappa[3] + alpha[h]), size
  ↪  = 4, col = "blue") +
  annotate("text", x = x[700 + effect], y = -0.01, label = expression(kappa[4] + alpha[h]), size
  ↪  = 4, col = "blue") +

  annotate("text", x = x[250 + effect], y = 0.01, label = expression(pi[1]), size = 4, col =
  ↪  "blue") +
  annotate("text", x = x[360 + effect], y = 0.05, label = expression(pi[2]), size = 4, col =
  ↪  "blue") +
  annotate("text", x = x[500 + effect], y = 0.10, label = expression(pi[3]), size = 4, col =
  ↪  "blue") +
  annotate("text", x = x[640 + effect], y = 0.05, label = expression(pi[4]), size = 4, col =
  ↪  "blue") +
  annotate("text", x = x[755 + effect], y = 0.01, label = expression(pi[5]), size = 4, col =
  ↪  "blue") +
  coord_cartesian(ylim = c(-0.0075, 0.25)) +
  theme_bw() +
  scale_x_continuous(breaks = seq(-7.5, 7.5, by = 1.5)) +
  labs(y = "Density")

# Warnings: do not pay attention
p
```



Data loading

```
rm(list = ls())
# Health Survey of the Region of Valencia (Spain) for the year 2016 (HSRV2016)
HSRV2016 <- read.spss(file.path("..", "data", "ESCV2016_Adultos.sav"),
                      use.value.labels = TRUE, to.data.frame = TRUE)
```

```
## re-encoding from CP1252
```

```
# Response variable: self-perceived health
levels(HSRV2016$P1) <- c("Very good", "Good", "Regular", "Bad", "Very bad")
y <- as.numeric(HSRV2016$P1)
# Number of categories
NCats <- length(table(y))
# Number of respondents
NResp <- length(y)

# Covariate sex: 1 = Male; 2 = Female
sexC <- HSRV2016$gSexo
levels(sexC) <- c("Male", "Female")
sex <- as.numeric(sexC)

# Covariate age group: 1 = [15,45); 2 = [45,65); 3 = [65,75); 4 = [75,85); 5 = [85,...)
ageC <- cut(HSRV2016$edad, breaks = c(15, 45, 65, 75, 85, 107),
```

```
            include.lowest = TRUE, right = FALSE)
levels(ageC)[length(table(ageC))] <- "[85,...)"
age <- as.numeric(ageC)

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

```
##         ageC
## sexC     [15,45) [45,65) [65,75) [75,85) [85,...)
##   Male        523     517     204     714      274
##   Female      674     572     394    1099      514
```

```
# Covariate dwelling stratum:
# 1 = Dwellings with 1 minor (without residents over 74)
# 2 = Dwellings with 2 or more minors (without residents over 74)
# 3 = Dwellings with residents over 74
# 4 = Other dwellings
dwell <- as.numeric(HSRV2016$estrato)

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

# Loading the population size of each municipality by sex and age group
# Source: https://www.ine.es/dynt3/inebase/index.htm?padre=6225
population <- readRDS(file = file.path("..", "data", "population.rds"))
```

R objects to be used in the NIMBLE model

```
# Vector of 1s
ones <- rep(1, NCats)

# index.groupSA contains the number of respondents sorted by sex and age group
index.groupSA <- c(table(sex, age)[1, ], table(sex, age)[2, ])
index.groupSA <- as.vector(c(0, cumsum(index.groupSA)))

# groupSA contains the respondents sorted by sex and age group
groupSA <- c()
for (SexGroup in 1:NSex) {
  for (AgeGroup in 1:NAges) {
    groupSA <- c(groupSA, which(sex == SexGroup & age == AgeGroup))
  }
}
# Number of all sex-age group combinations
NGroupSA <- prod(dim(table(sex, age)))

# index.groupD contains the number of respondents sorted by dwelling strata
index.groupD <- as.numeric(table(dwell))
index.groupD <- as.vector(c(0, cumsum(index.groupD)))

# groupD contains the respondents sorted by dwelling strata
groupD <- c()
```

```r
for (Dwell in 1:NDwells) {
  groupD <- c(groupD, which(dwell == Dwell))
}
```

Preparation of maps: Option 1

```r
load(file.path("..", "data", "cartography.RData"))

plot_map_neig <- function(neig) {
  plot(carto_muni)
  plot(carto_muni[neig, ], border = "black", col = "red", add = TRUE)
  plot(carto_muni[cv.nb[[neig]], ], border = "black", col = "pink",
       add = TRUE)
}

# # Valencia neighboring municipalities
# par(mar=c(c(2, 0, 2, 0) + 0.1))
# plot_map_neig(526)
# # Alicante neighboring municipalities
# plot_map_neig(14)
# # Elche neighboring municipalities
# plot_map_neig(65)
# # Castellón de la Plana neighboring municipalities
# plot_map_neig(177)
# # Ademuz neighboring municipalities
# plot_map_neig(277)
```

Preparation of maps: Option 2 (do not run if you have run Option 1)

```r
# # Cartography of the Region of Valencia
# load(file.path("..", "data", "CartoCV.Rdata"))
# # Cartography is sorted by municipality code
# order(carto_muni$INE_MUN)-1:542
# # Neighborhood structure by contiguity
# cv.nb <- poly2nb(carto_muni)
#
# # Some extra neighborhoods are added for Rincón de Ademuz comarca
# cv.nb[[277]] <- as.integer(sort(c(cv.nb[[277]], 312, 317, 517, 523, 508)))
# cv.nb[[363]] <- as.integer(sort(c(cv.nb[[363]], 312, 317, 517, 523, 508)))
# cv.nb[[364]] <- as.integer(sort(c(cv.nb[[364]], 312, 317, 517, 523, 508)))
# cv.nb[[477]] <- as.integer(sort(c(cv.nb[[477]], 312, 317, 517, 523, 508)))
#
# cv.nb[[312]] <- as.integer(sort(c(cv.nb[[312]], 277, 363, 364, 477)))
# cv.nb[[317]] <- as.integer(sort(c(cv.nb[[317]], 277, 363, 364, 477)))
# cv.nb[[517]] <- as.integer(sort(c(cv.nb[[517]], 277, 363, 364, 477)))
# cv.nb[[523]] <- as.integer(sort(c(cv.nb[[523]], 277, 363, 364, 477)))
# cv.nb[[508]] <- as.integer(sort(c(cv.nb[[508]], 277, 363, 364, 477)))
#
# # Municipality codes
# INE_MUN <- as.numeric(as.character(carto_muni@data$INE_MUN))
# # Municipality of each respondent
# muni <- HSRV2016$localidad
```

```
# muni <- match(muni, INE_MUN)
# # 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))
```

Coding the Leroux CAR distribution in NIMBLE

```
# 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 ~ k where k < 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))
  }
)
```

Statistical analysis

```
# Code to be able to reproduce the results shown in the paper.

# Number of chains to run in parallel
n.chains <- 5
this_cluster <- makeCluster(n.chains)
```

Model code

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

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

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

    # Prior distributions

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

```
    # alpha[1:NDwells] dwelling fixed effects (zero-sum constraint)
    alpha[1] <- -sum(alpha[2:NDwells])
    for (Dwell in 2:NDwells) {
      alpha[Dwell] ~ dflat()
    }

    # theta[1:NMuni] spatial random effect
    # LCAR distribution
    theta[1:NMuni] ~ dcar_leroux(rho = rho,
                                 sd.theta = 1,
                                 Lambda = Lambda[1:NMuni],
                                 from.to = from.to[1:NDist, 1:2])

    # Hyperparameters of the spatial random effect
    rho ~ dunif(0, 1)
    sd.theta ~ dhalfflat()

    # Zero-mean constraint for theta[1:NMuni]
    zero.theta ~ dnorm(mean.thetas, 10000)
    mean.thetas <- mean(theta[1:NMuni])

    # Stochastic restrictions in order to avoid confounding problems
    # Required vectors
    for (Resp in 1:NResp) {
      theta.Resp[Resp] <- theta[muni[Resp]]
      theta.Resp.groupSA[Resp] <- theta.Resp[groupSA[Resp]]
      theta.Resp.groupD[Resp] <- theta.Resp[groupD[Resp]]
    }
    # Constraint for theta[1:NMuni] - (sex, age group) covariates
    for (GroupSA in 1:NGroupSA) {
      zero.theta.groupSA[GroupSA] ~ dnorm(mean.thetas.groupSA[GroupSA], 10000)
      mean.thetas.groupSA[GroupSA] <- mean(theta.Resp.groupSA[(index.groupSA[GroupSA] +
↪   1):index.groupSA[GroupSA + 1]])
    }
    # Constraint for theta[1:NMuni] - dwelling covariate
    for (Dwell in 1:NDwells) {
      zero.theta.groupD[Dwell] ~ dnorm(mean.thetas.groupD[Dwell], 10000)
      mean.thetas.groupD[Dwell] <- mean(theta.Resp.groupD[(index.groupD[Dwell] +
↪   1):index.groupD[Dwell + 1]])
    }

  }
)
```

Data to be loaded

```
modelData <- list(y = y,
                  zero.theta = 0,
                  zero.theta.groupSA = rep(0, NGroupSA),
                  zero.theta.groupD = rep(0, NDwells)
)

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

```
                        NGroupSA = NGroupSA, groupSA = groupSA, groupD = groupD,
                        index.groupSA = index.groupSA, index.groupD = index.groupD
)
```

Parameters to be saved

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

Parallelization

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

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

  # Function that returns the value of the joint probability density function of the CAR Leroux
  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))
    }
  )

  # Another function of type rcar_leroux must be defined, otherwise it will cause an error
  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
NDwells <- constants$NDwells
NResp <- constants$NResp
NMuni <- constants$NMuni

# Let's create the NIMBLE model, creates the nodes (inits should be passed now)
# calculate = FALSE helps speed
model <- nimbleModel(code = code,
                     constants = constants,
                     data = data,
                     inits = list(delta = array(rdirichlet(NSex * NAges, ones),
                                                dim = c(NSex, NAges, NCats)),
                                  alpha = c(NA, runif(NDwells - 1)),
                                  rho = runif(1), sd.theta = runif(1),
                                  theta = rnorm(NMuni, sd = 0.1))
                     , calculate = FALSE)

# Compile the model, which means generating C++ code, compiling that code, and loading it back
↪  into R
Cmodel <- compileNimble(model)
# Let's change the default configuration
# The conjugacy checking is a slow part of MCMC configuration, skipping it
# (useConjugacy = FALSE) helps speed
modelMCMCconfiguration <- configureMCMC(model, useConjugacy = FALSE)
# Remove desire samplers
modelMCMCconfiguration$removeSamplers(c("alpha", "sd.theta", "theta", "rho"))

# Add slice alpha[2:NDwells] samplers
alphas <- numeric()
for (Dwell in 2:NDwells) {
  alphas[Dwell] <- paste("alpha[",Dwell,"]")
}
alphas <- gsub(" ", "", alphas)

for (Dwell in 2:NDwells) {
  modelMCMCconfiguration$addSampler(target = alphas[Dwell], type = "slice")
}

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

```
  # Add slice/RW-MH theta[1:NMuni] samplers
  thetas <- character(NMuni)
  for (Muni in 1:NMuni) {
    thetas[Muni] <- paste("theta[",Muni,"]")
  }
  thetas <- gsub(" ", "", thetas)

  smuni <- sort(unique(constants$muni))
  for (Muni in 1:NMuni) {
    ifelse(Muni %in% smuni,
           # Alternative: type = "slice", better mixing but slower than RW
           modelMCMCconfiguration$addSampler(target = thetas[Muni], type = "RW"),
           # This sampler should be RW
           modelMCMCconfiguration$addSampler(target = thetas[Muni], type = "RW"))
  }

  # Add slice rho sampler
  modelMCMCconfiguration$addSampler(target = "rho", 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 = 8500, nburnin = 1000, thin = 15,
                     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))
```

```
##    user  system elapsed
##   0.052   0.035 529.176
```

```
# It's good practice to close the cluster when you're done with it.
stopCluster(this_cluster)
```

Model results

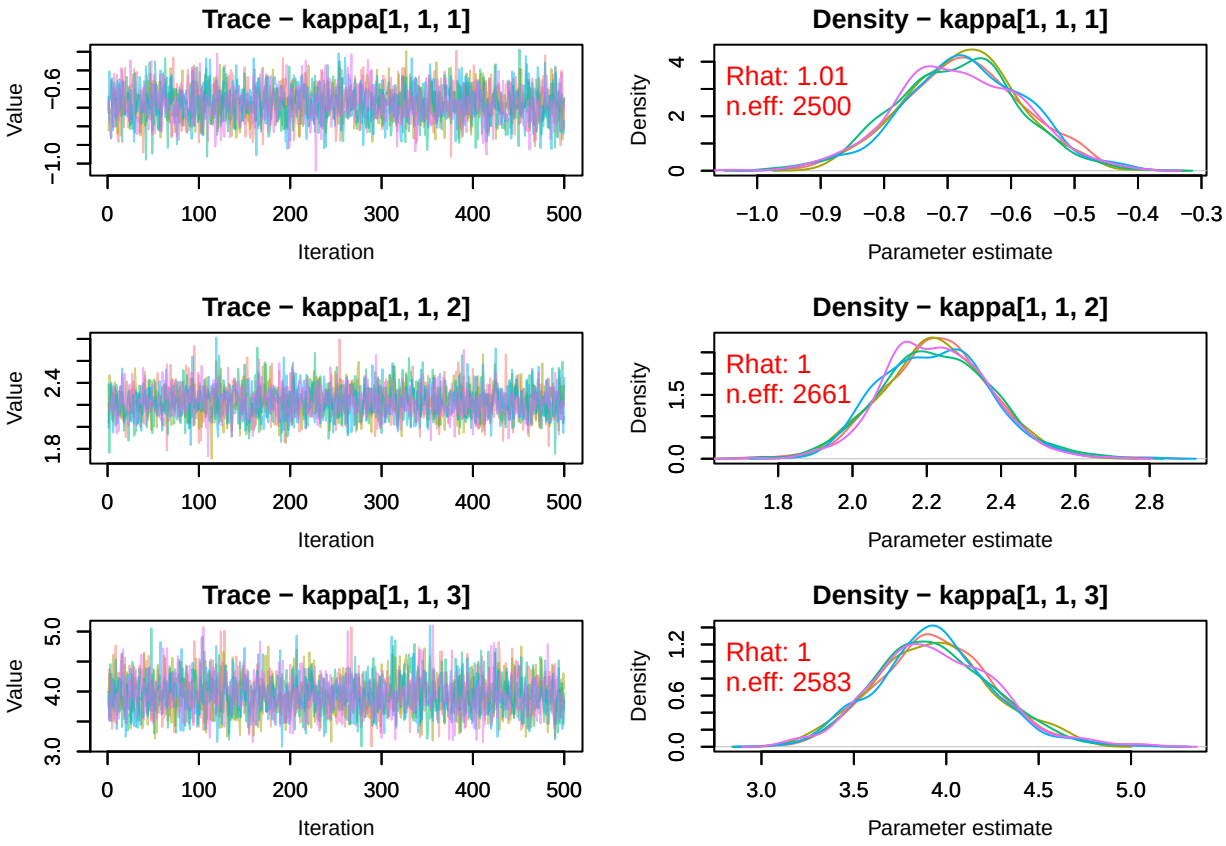
Convergence assessment

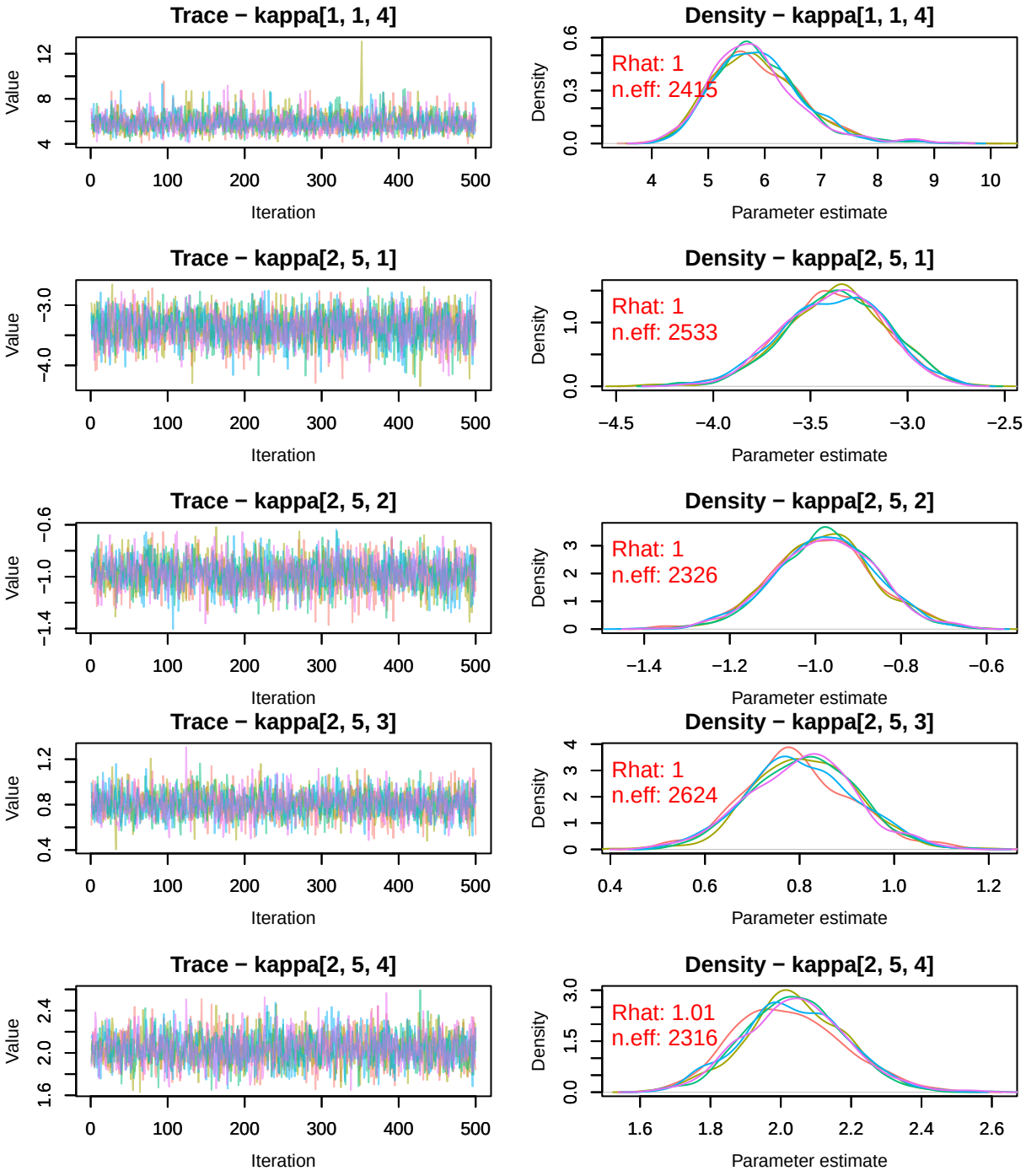
```
MCMCsummary(object = salnimble, params = "kappa",
            # exact = TRUE,
            # ISB = FALSE,
            round = 4)
```

```
##                    mean     sd    2.5%     50%   97.5% Rhat n.eff
## kappa[1, 1, 1] -0.6780 0.0955 -0.8674 -0.6768 -0.4885 1.01  2500
## kappa[2, 1, 1] -0.6636 0.0833 -0.8296 -0.6638 -0.5030 1.00  2741
## kappa[1, 2, 1] -1.6325 0.1184 -1.8666 -1.6349 -1.3948 1.00  2531
## kappa[2, 2, 1] -1.7883 0.1225 -2.0446 -1.7827 -1.5559 1.00  2678
## kappa[1, 3, 1] -2.3433 0.2510 -2.8463 -2.3359 -1.8802 1.00  2331
## kappa[2, 3, 1] -2.2809 0.1798 -2.6407 -2.2779 -1.9361 1.00  2416
## kappa[1, 4, 1] -2.7980 0.1701 -3.1536 -2.7958 -2.4890 1.00  2405
## kappa[2, 4, 1] -2.8005 0.1465 -3.0806 -2.7975 -2.5267 1.00  2447
## kappa[1, 5, 1] -2.7542 0.2630 -3.2934 -2.7482 -2.2471 1.00  2645
## kappa[2, 5, 1] -3.3743 0.2584 -3.8882 -3.3652 -2.8979 1.00  2533
## kappa[1, 1, 2]  2.2273 0.1440  1.9598  2.2255  2.5235 1.00  2661
## kappa[2, 1, 2]  1.7536 0.1107  1.5376  1.7531  1.9781 1.00  2778
## kappa[1, 2, 2]  0.9299 0.0991  0.7368  0.9276  1.1162 1.00  2500
## kappa[2, 2, 2]  0.8447 0.0933  0.6664  0.8454  1.0316 1.00  2508
## kappa[1, 3, 2]  0.2097 0.1414 -0.0714  0.2086  0.4851 1.00  2864
## kappa[2, 3, 2] -0.0748 0.1110 -0.2905 -0.0759  0.1491 1.01  2487
## kappa[1, 4, 2] -0.2331 0.0971 -0.4238 -0.2346 -0.0498 1.00  2261
## kappa[2, 4, 2] -0.6543 0.0886 -0.8264 -0.6531 -0.4883 1.00  1947
## kappa[1, 5, 2] -0.3911 0.1379 -0.6632 -0.3903 -0.1223 1.00  2302
## kappa[2, 5, 2] -0.9782 0.1171 -1.2136 -0.9774 -0.7493 1.00  2326
## kappa[1, 1, 3]  3.9410 0.3170  3.3636  3.9284  4.6138 1.00  2583
## kappa[2, 1, 3]  3.6626 0.2426  3.2130  3.6537  4.1560 1.00  2555
## kappa[1, 2, 3]  2.5437 0.1635  2.2340  2.5396  2.8778 1.00  2436
## kappa[2, 2, 3]  2.5069 0.1551  2.2073  2.4998  2.8222 1.01  2612
## kappa[1, 3, 3]  1.9531 0.2102  1.5562  1.9460  2.3872 1.01  2598
## kappa[2, 3, 3]  1.9298 0.1525  1.6339  1.9319  2.2342 1.00  2358
## kappa[1, 4, 3]  1.7651 0.1184  1.5302  1.7633  2.0073 1.00  2329
## kappa[2, 4, 3]  1.1320 0.0908  0.9516  1.1330  1.3078 1.00  2052
## kappa[1, 5, 3]  1.3499 0.1585  1.0350  1.3517  1.6514 1.01  2335
## kappa[2, 5, 3]  0.8057 0.1121  0.5921  0.8030  1.0299 1.00  2624
## kappa[1, 1, 4]  5.8775 0.8086  4.5993  5.7866  7.7146 1.00  2415
## kappa[2, 1, 4]  5.6456 0.6373  4.5491  5.5812  7.0753 1.00  2723
## kappa[1, 2, 4]  3.7525 0.2871  3.2184  3.7391  4.3481 1.00  2444
## kappa[2, 2, 4]  3.6415 0.2621  3.1739  3.6181  4.1832 1.00  2388
## kappa[1, 3, 4]  4.0891 0.5376  3.1601  4.0467  5.2261 1.00  2700
## kappa[2, 3, 4]  3.2896 0.2638  2.8092  3.2836  3.8440 1.00  2631
## kappa[1, 4, 4]  3.1266 0.1874  2.7739  3.1214  3.4954 1.00  2433
## kappa[2, 4, 4]  2.7409 0.1382  2.4766  2.7403  3.0199 1.01  2260
## kappa[1, 5, 4]  2.8603 0.2614  2.3572  2.8533  3.3960 1.00  2612
## kappa[2, 5, 4]  2.0369 0.1452  1.7622  2.0327  2.3270 1.01  2316
```

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

```
          exact = TRUE,
          ISB = FALSE,
          params = c("kappa[1, 1, 1]", "kappa[1, 1, 2]",
                     "kappa[1, 1, 3]", "kappa[1, 1, 4]",
                     "kappa[2, 5, 1]", "kappa[2, 5, 2]",
                     "kappa[2, 5, 3]", "kappa[2, 5, 4]"))
```



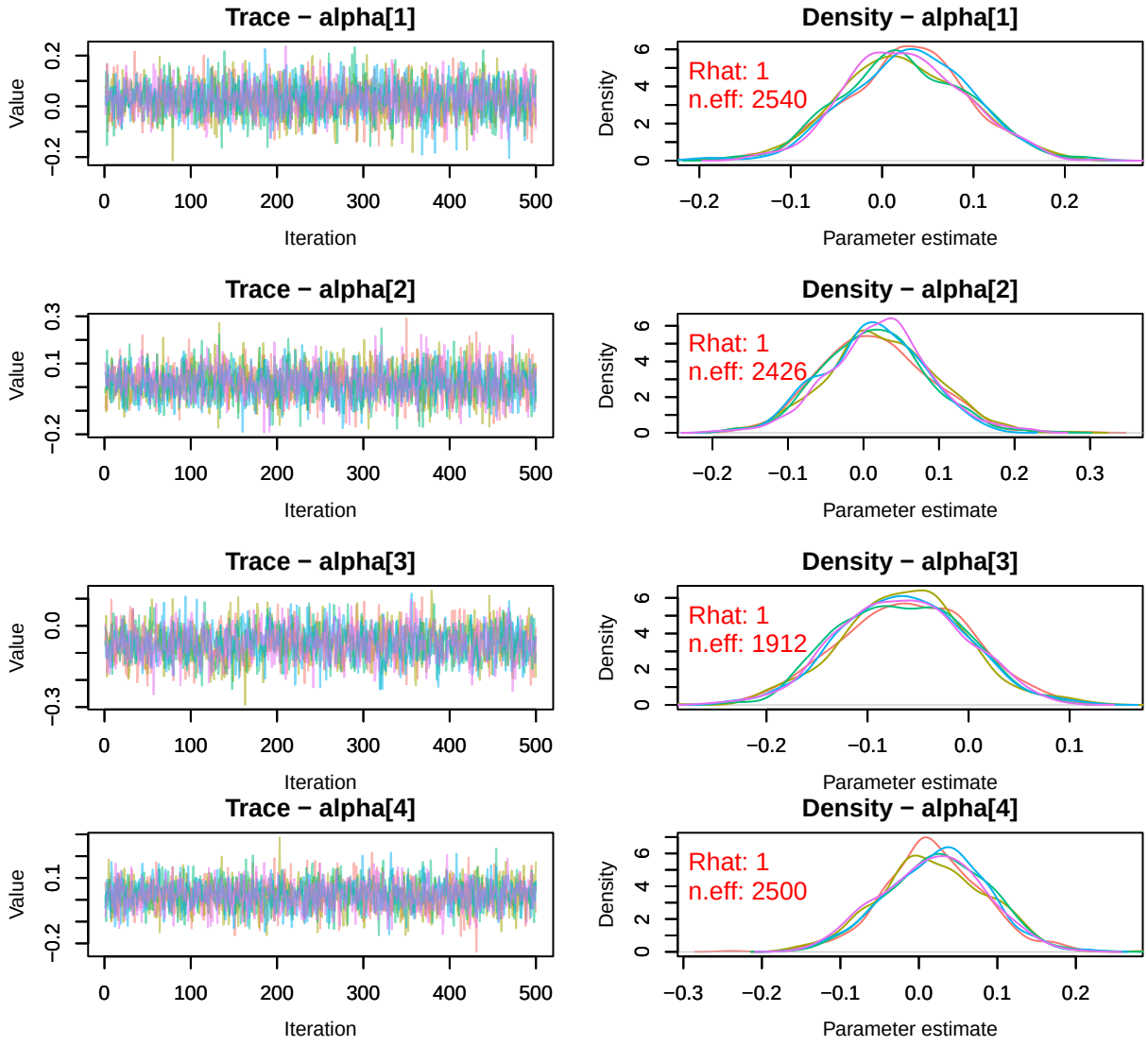



```
MCMCsummary(object = salnimble, params = "alpha",
            # exact = TRUE,
            # ISB = FALSE,
            round = 4)
```

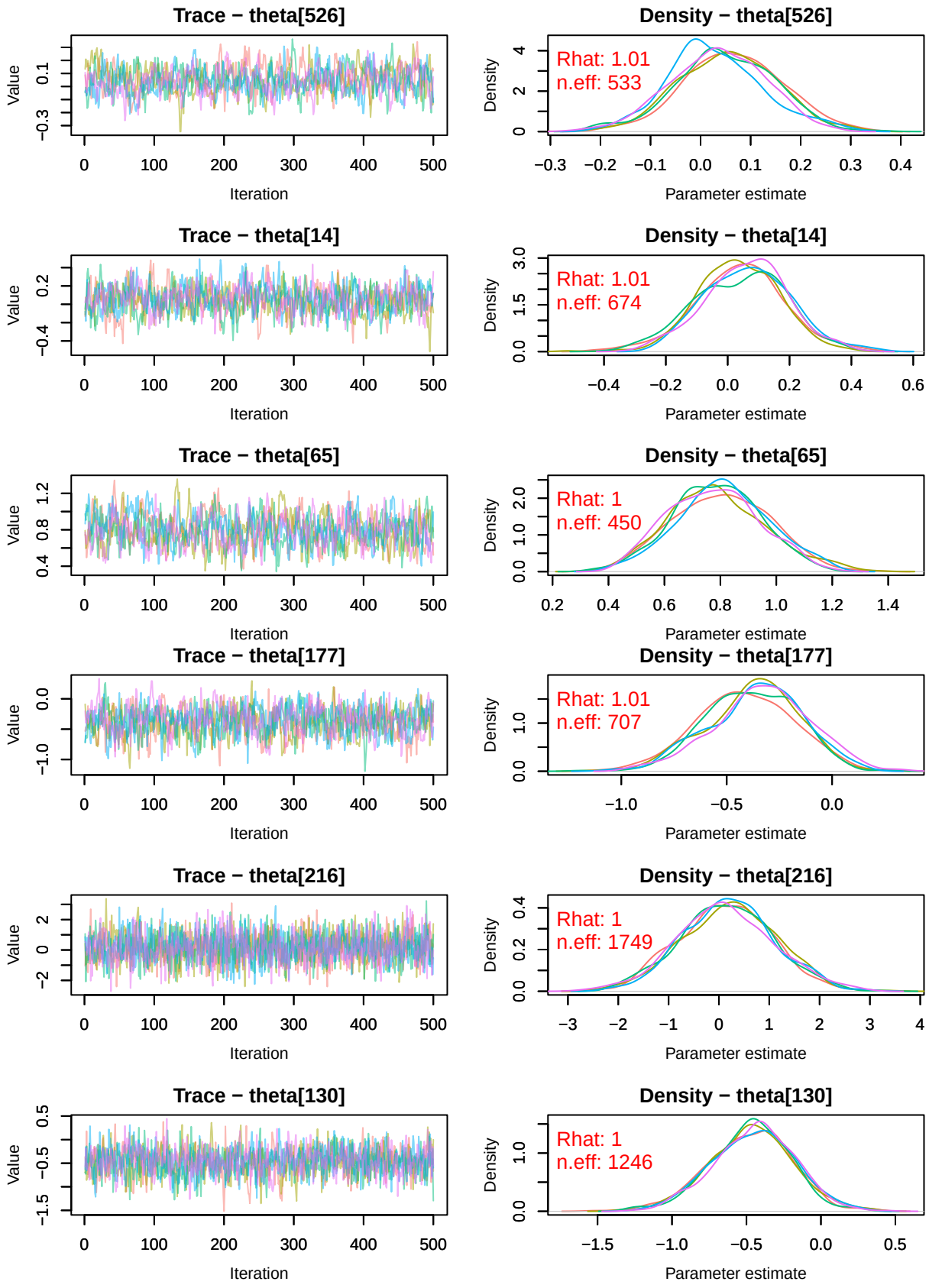
```
##              mean     sd    2.5%     50%  97.5% Rhat n.eff
## alpha[1]   0.0267 0.0667 -0.0979  0.0255 0.1580    1  2540
## alpha[2]   0.0173 0.0685 -0.1114  0.0167 0.1510    1  2426
## alpha[3] -0.0647 0.0619 -0.1823 -0.0641 0.0564    1  1912
```

```
## alpha[4]  0.0207 0.0650 -0.1063  0.0211 0.1459    1  2500
```

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



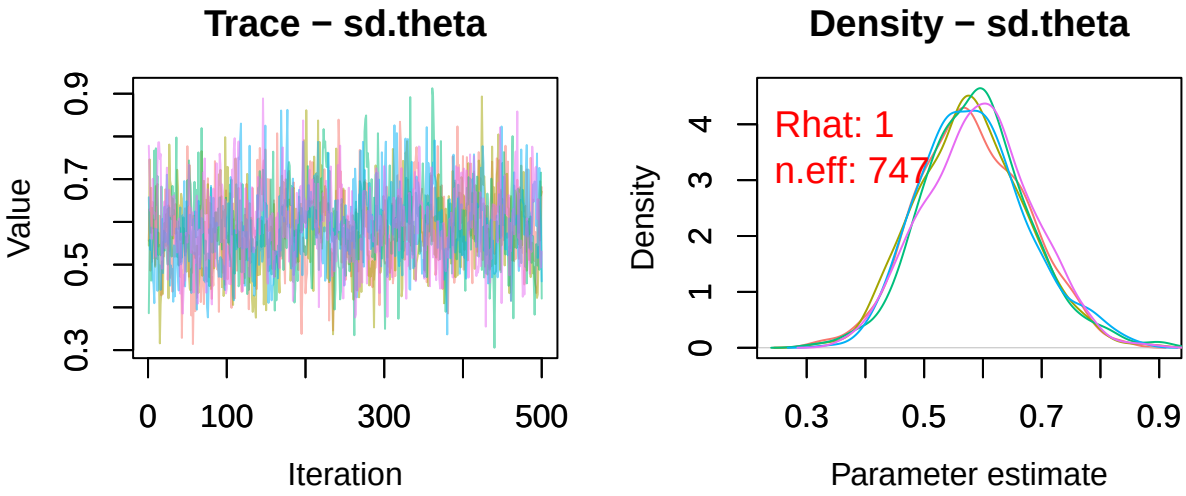

```
MCMCtrace(object = salnimble,
          pdf = FALSE, # no export to PDF
          ind = TRUE,  # separate density lines per chain
          Rhat = TRUE,
          n.eff = TRUE,
          exact = TRUE,
          ISB = FALSE,
          params = c("theta[526]", "theta[14]", "theta[65]", "theta[177]",
                     "theta[216]", "theta[130]"))
```


Trace − theta[526]
Value
Iteration
Density − theta[526]
Rhat: 1.01
n.eff: 533
Density
Parameter estimate
Trace − theta[14]
Value
Iteration
Density − theta[14]
Rhat: 1.01
n.eff: 674
Density
Parameter estimate
Trace − theta[65]
Value
Iteration
Density − theta[65]
Rhat: 1
n.eff: 450
Density
Parameter estimate
Trace − theta[177]
Value
Iteration
Density − theta[177]
Rhat: 1.01
n.eff: 707
Density
Parameter estimate
Trace − theta[216]
Value
Iteration
Density − theta[216]
Rhat: 1
n.eff: 1749
Density
Parameter estimate
Trace − theta[130]
Value
Iteration
Density − theta[130]
Rhat: 1
n.eff: 1246
Density
Parameter estimate

```
MCMCsummary(object = salnimble, params = "sd.theta",
            # exact = TRUE,
            # ISB = FALSE,
            round = 4)
```

```
##             mean     sd 2.5%    50% 97.5% Rhat n.eff
## sd.theta 0.5862 0.0913 0.42 0.5831 0.776    1   747
```

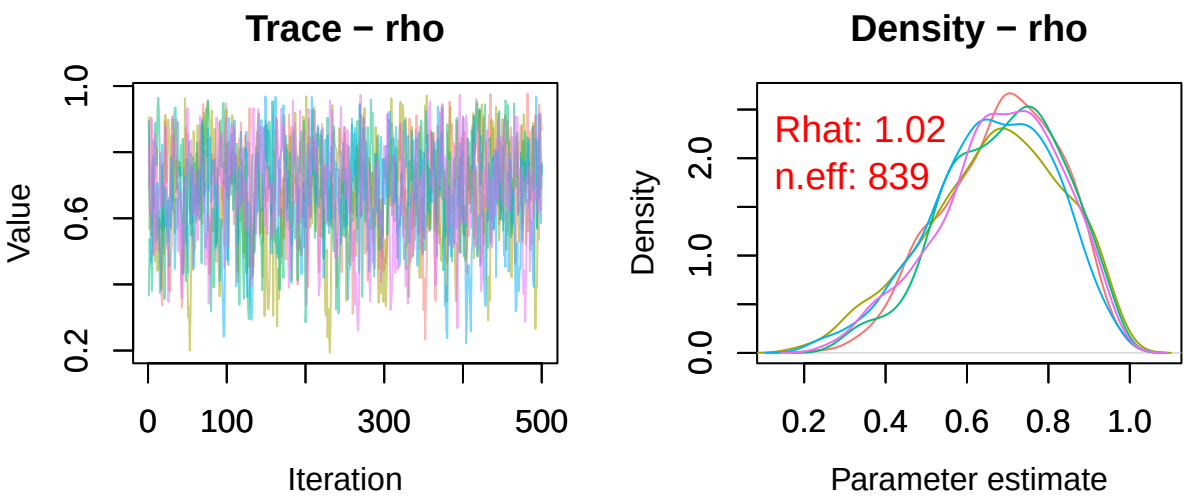
```
MCMCtrace(object = salnimble,
          pdf = FALSE, # no export to PDF
          ind = TRUE,  # separate density lines per chain
          Rhat = TRUE,
          n.eff = TRUE,
          params = "sd.theta")
```




```
MCMCsummary(object = salnimble, params = "rho",
            # exact = TRUE,
            # ISB = FALSE,
            round = 4)
```

```
##       mean    sd   2.5%    50%  97.5% Rhat n.eff
## rho 0.6782 0.151 0.3481 0.6899 0.9306 1.02   839
```

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




```
which((MCMCsummary(object = salnimble, params = "kappa", round = 4)[, 6]) > 1.02 |
↪   (MCMCsummary(object = salnimble, params = "kappa", round = 4)[, 7] < 400))
```

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

```
which((MCMCsummary(object = salnimble, params = "alpha", round = 4)[, 6]) > 1.02 |
↪   (MCMCsummary(object = salnimble, params = "alpha", round = 4)[, 7] < 400))
```

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

```
which((MCMCsummary(object = salnimble, params = "theta", round = 4)[, 6]) > 1.02 |
↪   (MCMCsummary(object = salnimble, params = "theta", round = 4)[, 7] < 400))
```

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

```
which((MCMCsummary(object = salnimble, params = "sd.theta", round = 4)[, 6]) > 1.02 |
↪   (MCMCsummary(object = salnimble, params = "sd.theta", round = 4)[, 7] < 400))
```

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

```
which((MCMCsummary(object = salnimble, params = "rho", round = 4)[, 6]) > 1.02 |
↪   (MCMCsummary(object = salnimble, params = "rho", round = 4)[, 7] < 400))
```

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

Function NimtoWin: NIMBLE's output to WinBUGS's sims.list output format

```r
# NimToWin:
# - transforms NIMBLE output to WinBUGS sims.list output format

NimToWin <- function(salnimble) {

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

  kappa <- array(dim = c(n.sims, NSex, NAges, NCats - 1))
  alpha <- matrix(nrow = n.sims, ncol = NDwells)
  theta <- matrix(nrow = n.sims, ncol = NMuni)
  sd.theta <- numeric(length = n.sims)
  rho <- numeric(length = n.sims)

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

  for (Dwell in 1:NDwells) {
    alpha[, Dwell] <- c(salnimble[[1]][, gsub(" ", "", paste("alpha[", Dwell, "]"))],
                        salnimble[[2]][, gsub(" ", "", paste("alpha[", Dwell, "]"))],
                        salnimble[[3]][, gsub(" ", "", paste("alpha[", Dwell, "]"))],
                        salnimble[[4]][, gsub(" ", "", paste("alpha[", Dwell, "]"))],
                        salnimble[[5]][, gsub(" ", "", paste("alpha[", Dwell, "]"))])
  }

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

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

  rho <- c(salnimble[[1]][, "rho"], salnimble[[2]][, "rho"],
           salnimble[[3]][, "rho"], salnimble[[4]][, "rho"],
           salnimble[[5]][, "rho"])

  summary <- MCMCsummary(object = salnimble, round = 4)
  sims.list <- list("kappa" = kappa, "alpha" = alpha, "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)
}

salwinbugs <- NimToWin(salnimble = salnimble)
```

Supplementary Material: Dwelling stratum effect is not relevant

```
subtitles <- c("Dwellings with 1 minor (without residents over 74)",
               "Dwellings with 2+ (without residents over 74)",
               "Dwellings with residents over 74",
               "Other dwellings")

CIalpha <- apply(salwinbugs$sims.list$alpha, 2, quantile, probs = c(0.025, 0.975))

p <- list()
for (Dwell in 1:NDwells) {
  df <- data.frame("x" = density(salwinbugs$sims.list$alpha[, Dwell])$x,
                   "y" = density(salwinbugs$sims.list$alpha[, Dwell])$y)
  p[[Dwell]] <- ggplot(df, aes(x = x, y = y)) +
    geom_line() +
    geom_vline(xintercept = 0, linetype = "dashed", color = "black") +
    geom_vline(xintercept = CIalpha[, Dwell], linetype = "twodash", color = "red") +
    labs(title = subtitles[Dwell], x = "x", y = "Density") +
    theme_bw()
}

supplementary <- ggarrange(p[[1]], p[[2]], p[[3]], p[[4]], nrow = 2, ncol = 2)
supplementary
```

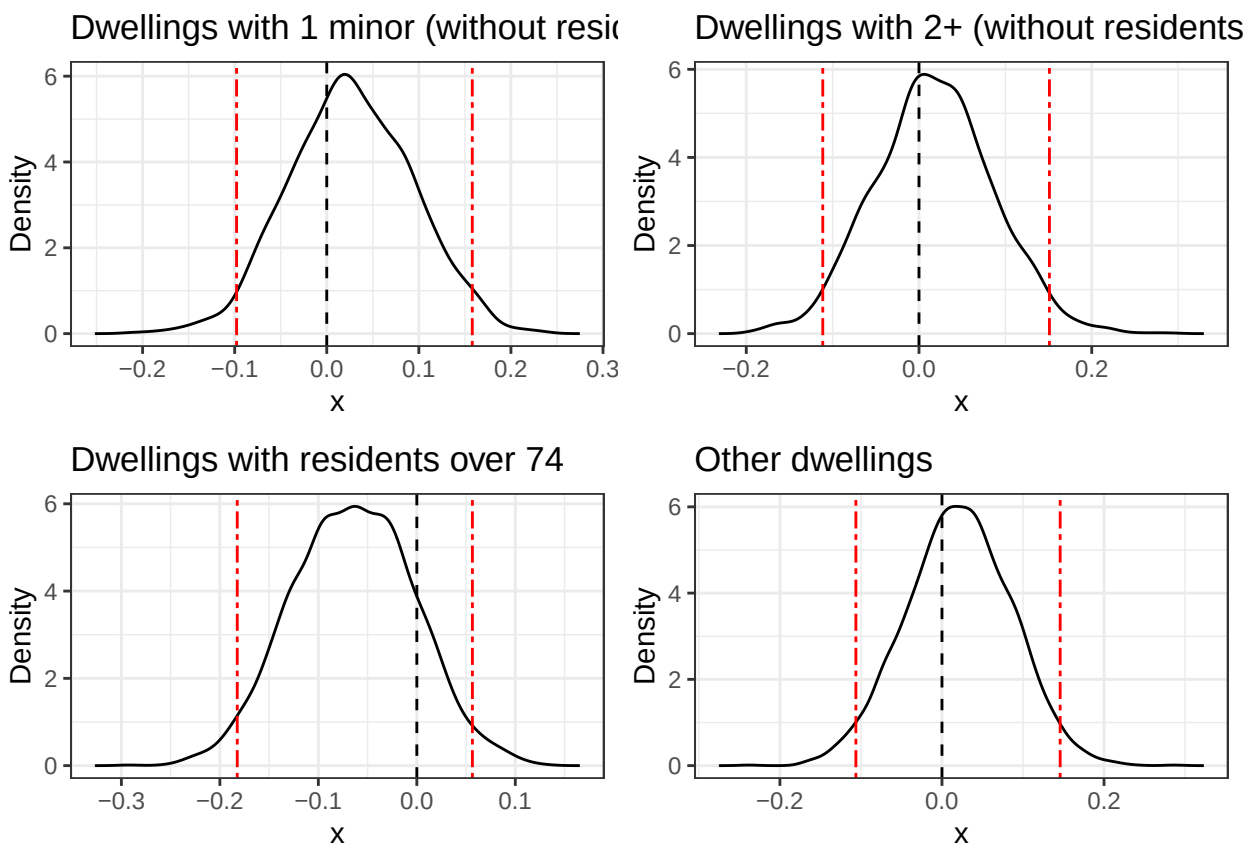
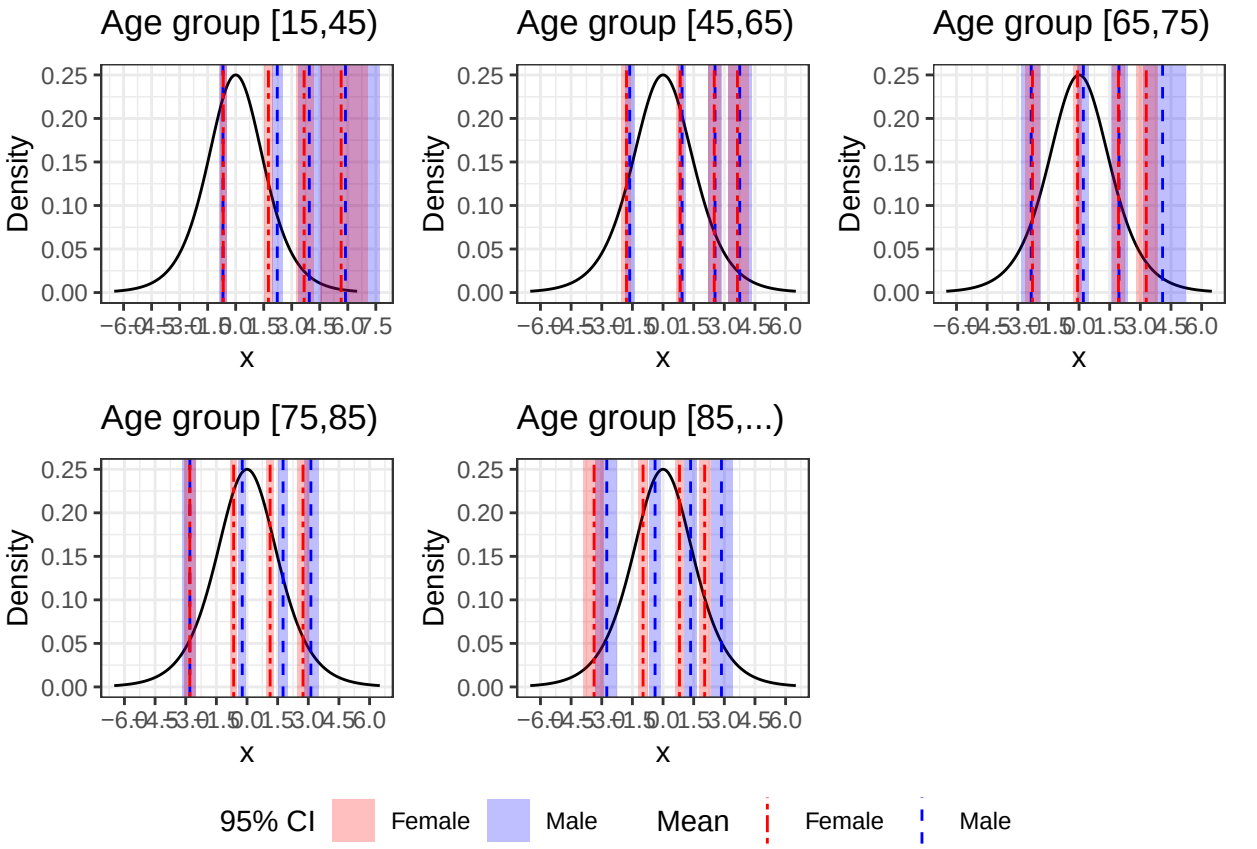


Figure 2: `kappa[1:NSex, 1:NAges, 1:(NCats-1)]`

```
kappamean <- apply(salwinbugs$sims.list$kappa, 2:4, mean)
kappaIC <- apply(salwinbugs$sims.list$kappa, 2:4, quantile, probs = c(0.025, 0.975))
x <- seq(from = -6.5, to = 6.5, length.out = 1000)

p <- list()
for (AgeGroup in 1:NAges) {
  AgeLevel <- levels(ageC)[AgeGroup]
  df <- data.frame("x" = x,
                   "y" = dlogis(x))
  lines <- data.frame("intercepts" = as.numeric(kappamean[, AgeGroup, ]),
                      "Sex group" = rep(c("Male", "Female"), NCats - 1))
  ic <- data.frame("lower" = as.numeric(kappaIC[1, , AgeGroup, ]),
                   "upper" = as.numeric(kappaIC[2, , AgeGroup, ]),
                   "Sex group" = rep(c("Male", "Female"), NCats - 1))
  p[[AgeGroup]] <- ggplot() +
    geom_line(data = df, mapping = aes(x = x, y = y)) +
    geom_rect(data = ic, mapping = aes(xmin = lower, xmax = upper, ymin = -0.1, ymax = 0.30,
                                       fill = Sex.group),
              alpha = 0.25) +
    geom_vline(data = lines,
               mapping = aes(xintercept = intercepts,
                             linetype = Sex.group,
                             color = Sex.group)) +
    scale_fill_manual(name = "95% CI", values = c("Male" = "blue", "Female" = "red")) +
    scale_color_manual(name = "Mean", values = c("Male" = "blue", "Female" = "red")) +
```

```
    scale_linetype_manual(name = "Mean", values = c("Male" = "dashed", "Female" = "twodash")) +
    labs(title = substitute(paste("Age group ", a), list(a = AgeLevel)), x = "x", y = "Density")
    ↪  +
    coord_cartesian(ylim = c(0, 0.25)) +
    #coord_cartesian(xlim = c(1.5, 5)) +
    theme_bw() +
    scale_x_continuous(breaks = seq(-7.5, 7.5, by = 1.5))
}

kappa <- ggarrange(p[[1]], p[[2]], p[[3]], p[[4]], p[[5]], nrow = 2, ncol = 3,
                   common.legend = TRUE, legend = "bottom")
kappa
```




```
table(y, sexC, ageC)
```

```
## , , ageC = [15,45)
##
##    sexC
## y   Male Female
##   1  180    234
##   2  293    341
##   3   41     83
##   4    8     14
##   5    1      2
##
```

```
## , , ageC = [45,65)
##
##    sexC
## y   Male Female
##   1   86     84
##   2  283    314
##   3  110    130
##   4   26     29
##   5   12     15
##
## , , ageC = [65,75)
##
##    sexC
## y   Male Female
##   1   18     36
##   2   94    150
##   3   66    156
##   4   22     37
##   5    3     15
##
## , , ageC = [75,85)
##
##    sexC
## y   Male Female
##   1   40     61
##   2  266    302
##   3  296    451
##   4   79    212
##   5   33     73
##
## , , ageC = [85,...)
##
##    sexC
## y   Male Female
##   1   16     16
##   2   91    119
##   3  107    212
##   4   44    103
##   5   16     64
```

Figure 3 (a): Map of the RV - `theta[1:NMuni]`

```
# Total saved simulations
n.sims <- salwinbugs$n.sims
# Saved theta simulations
thetasim <- salwinbugs$sims.list$sd.theta * salwinbugs$sims.list$theta

# Discretization by nine equal-probability intervals
breaks <- c(min(apply(thetasim, 2, mean)) - 0.001, quantile(apply(thetasim, 2, mean), probs =
↪  seq(1/9, 8/9, length.out = 8)), max(apply(thetasim, 2, mean)))
carto_muni@data$thetamean <- cut(apply(thetasim, 2, mean), breaks = breaks, include.lowest =
↪  FALSE, right = TRUE)
```

```
# We order the factor levels to match the colors appropriately
carto_muni@data$thetamean <- factor(carto_muni@data$thetamean, levels =
                                      rev(levels(carto_muni@data$thetamean)))
levels(carto_muni@data$thetamean) <- c("Better", " ", "  ", "   ", "    ",
                                       "     ", "      ", "       ", "Worse")

spplot(carto_muni,
       c("thetamean"),
       col.regions = colorRampPalette(brewer.pal(7,'BrBG'))(9)[9:1],
       cuts = 8,
       colorkey = list(key = list(labels = c("Better", " ", "  ", "   ", "    ",
                                             "     ", "      ", "       ", "Worse")),
                       width = 1.5, cex = 1.5, height = 0.75),
       par.settings = list(axis.line = list(col = 'transparent')),
       col = "black",
       lwd = 0.10)
```

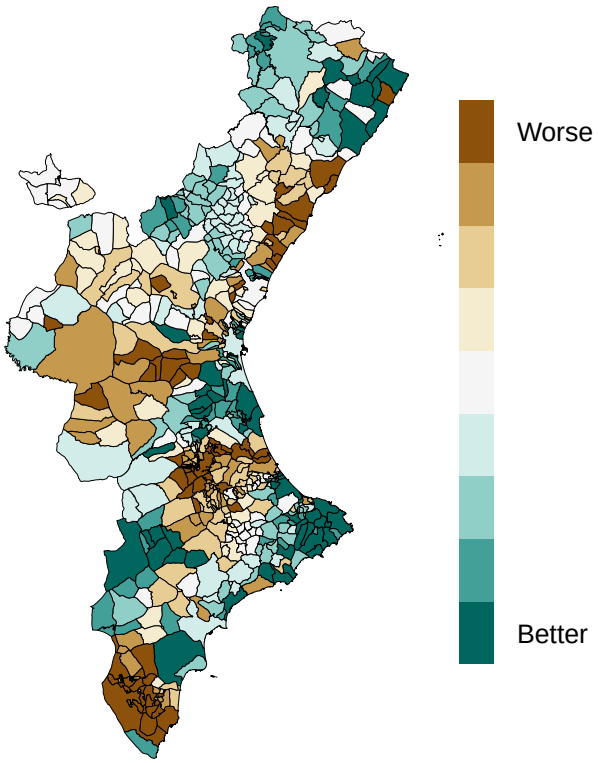


Figure 3 (b): Map of the RV - P(theta < 0 | y)

```
# Checking when theta is LESS than zero (worse self-perceived health)
stepsim <- matrix(nrow = n.sims, ncol = NMuni)
for (sim in 1:n.sims) {
  for (Muni in 1:NMuni) {
    stepsim[sim, Muni] <- ifelse(thetasim[sim, Muni] < 0, 1, 0)
  }
}

breaks <- c(-0.01, 0.05, 0.10, 0.20, 0.80, 0.90, 0.95, 1)
```

```
carto_muni@data$probmean <- cut(apply(stepsim, 2, mean), breaks = breaks, include.lowest = FALSE,
↪  right = TRUE)
levels(carto_muni@data$probmean) <- c("[0,0.05]",
                                      levels(carto_muni@data$probmean)[2:6],
                                      "(0.95,1]")

spplot(carto_muni,
       c("probmean"),
       col.regions = colorRampPalette(brewer.pal(7,'RdYlGn'))(9)[c(9:7, 5, 3:1)],
       cuts = 8,
       colorkey = list(key = list(labels = levels(carto_muni@data$probmean)),
                       width = 1.5, cex = 1.5, height = 0.75),
       par.settings = list(axis.line = list(col = 'transparent')),
       col = "black",
       lwd = 0.10)
```

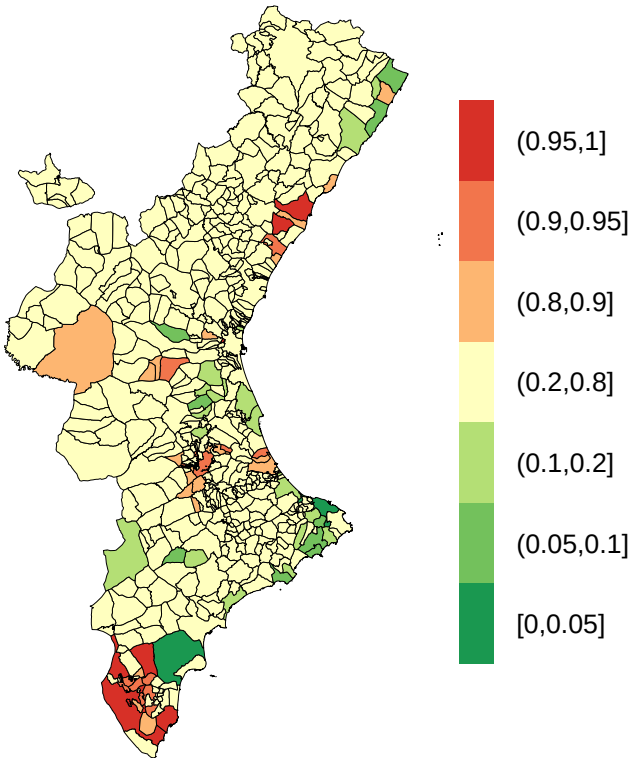


Figure 4: Maps of the RV - area population level percentages with post-stratification (MRP)

```
# function1:
# - (1) computes the n.sims simulated probabilities for each sex, age group and municipality
# - (2) post-stratifies these probabilities for each municipality and category

function1 <- function(salwinbugs) {

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

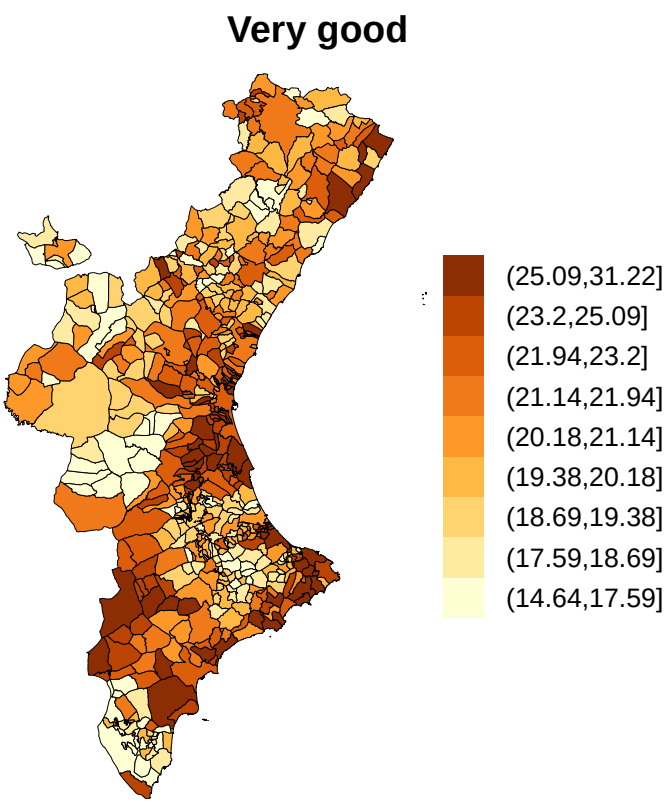
```
  prlevels <- array(dim = c(n.sims, NMuni, NSex, NAges, NCats))
  prlevels.post <- array(dim = c(n.sims, NMuni, NCats))

  # Probabilities for each sex, age group and municipality
  for (sim in 1:n.sims) {
    for (SexGroup in 1:NSex) {
      for (AgeGroup in 1:NAges) {
        for (Muni in 1:NMuni) {
          for (Cat in 1:(NCats - 1)) {
            p.gamma[sim, Muni, SexGroup, AgeGroup, Cat] <-
              ilogit(salwinbugs$sims.list$kappa[sim, SexGroup, AgeGroup, Cat] +
                       salwinbugs$sims.list$sd.theta[sim] * salwinbugs$sims.list$theta[sim,
↪  Muni])
          }

          prlevels[sim, Muni, SexGroup, AgeGroup, 1] <- p.gamma[sim, Muni, SexGroup, AgeGroup, 1]
          prlevels[sim, Muni, SexGroup, AgeGroup, NCats] <- 1 - p.gamma[sim, Muni, SexGroup,
↪  AgeGroup, NCats - 1]

          for (Cat in 2:(NCats - 1)) {
            prlevels[sim, Muni, SexGroup, AgeGroup, Cat] <-
              p.gamma[sim, Muni, SexGroup, AgeGroup, Cat] - p.gamma[sim, Muni, SexGroup,
↪  AgeGroup, Cat-1]
          }
        }
      }
    }

    # Post-stratification
    for (Muni in 1:NMuni) {
      for (Cat in 1:NCats) {
        prlevels.post[sim, Muni, Cat] <- sum(population[Muni, , ])^(-1) * sum(prlevels[sim, Muni,
↪  , , Cat] * population[Muni, , ])
      }
    }
    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(list("prlevels" = prlevels, "prlevels.post" = prlevels.post))
}

# This may take a minute
result1 <- function1(salwinbugs)
```

```
## 1 of 2500 simulations
## 500 of 2500 simulations
## 1000 of 2500 simulations
## 1500 of 2500 simulations
## 2000 of 2500 simulations
## 2500 of 2500 simulations
```

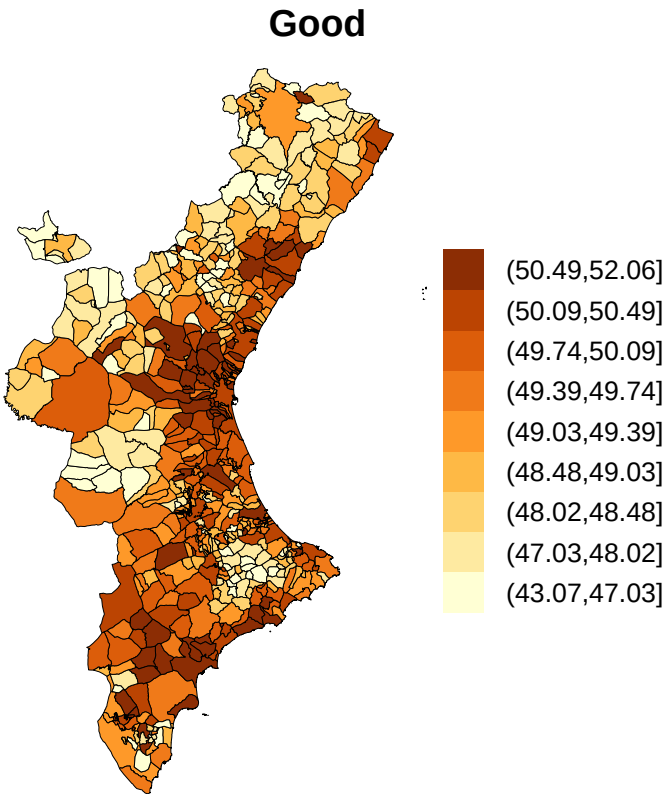
```
# Post-stratified posterior means of the percentages for each municipality and category
percentagesmean <- apply(result1$prlevels.post, c(2, 3), mean) * 100

category <- percentagesmean[, 1]
```

```r
breaks <- c(min(category) - 0.001, quantile(category, probs = seq(1/9, 8/9, length.out = 8)),
↪  max(category))
carto_muni@data$category1 <- cut(category, breaks = breaks, include.lowest = FALSE, right = TRUE,
↪  dig.lab = 4)

category <- percentagesmean[, 2]
breaks <- c(min(category) - 0.001, quantile(category, probs = seq(1/9, 8/9, length.out = 8)),
↪  max(category))
carto_muni@data$category2 <- cut(category, breaks = breaks, include.lowest = FALSE, right = TRUE,
↪  dig.lab = 4)

category <- percentagesmean[, 3]
breaks <- c(min(category) - 0.001, quantile(category, probs = seq(1/9, 8/9, length.out = 8)),
↪  max(category))
carto_muni@data$category3 <- cut(category, breaks = breaks, include.lowest = FALSE, right = TRUE,
↪  dig.lab = 4)

category <- percentagesmean[, 4]
breaks <- c(min(category) - 0.001, quantile(category, probs = seq(1/9, 8/9, length.out = 8)),
↪  max(category))
carto_muni@data$category4 <- cut(category, breaks = breaks, include.lowest = FALSE, right = TRUE,
↪  dig.lab = 4)

category <- percentagesmean[, 5]
breaks <- c(min(category) - 0.001, quantile(category, probs = seq(1/9, 8/9, length.out = 8)),
↪  max(category))
carto_muni@data$category5 <- cut(category, breaks = breaks, include.lowest = FALSE, right = TRUE,
↪  dig.lab = 4)

# Mean age of each municipality
# Loading the population size of each municipality by sex and age group
# Source: https://www.ine.es/dynt3/inebase/index.htm?padre=6225
valencia <- read_excel(file.path("..", "data", "todovalencia2016.xlsx"), col_names = TRUE)
castellon <- read_excel(file.path("..", "data", "todocastellon2016.xlsx"), col_names = TRUE)
alicante <- read_excel(file.path("..", "data", "todoalicante2016.xlsx"), col_names = TRUE)
region <- rbind(alicante, castellon, valencia)
rm(list = c("valencia", "alicante", "castellon"))

region <- region[order(region$Municipio), ]
region <- with(region, region[which(Sexo == "Ambos"), ])
region <- as.data.frame(region); region <- region[, -c(1, 2)]

# Average age five-year age group
age5 <- c(2, 7, 12, 17, 22, 27, 32, 37, 42, 47, 52, 57, 62, 67, 72, 77, 82, 87, 92, 97, 102) +
↪  0.5
region <- data.frame(apply(region, 1, function(x) {x/sum(x)}))
# Mean age
agemean <- apply(region * age5, 2, sum)

breaks <- c(min(agemean) - 0.001, quantile(agemean, probs = seq(1/9, 8/9, length.out = 8)),
↪  max(agemean))
# breaks
breaks <- c(36.38, round(breaks, 2)[2:9], 64.86)
carto_muni@data$agemean <- cut(agemean, breaks = breaks, include.lowest = FALSE, right = TRUE,
↪  dig.lab = 4)
levels(carto_muni@data$agemean) <- c(levels(carto_muni@data$agemean)[1:8],
                                     "(53.32,64.86)")

spplot(carto_muni,
```

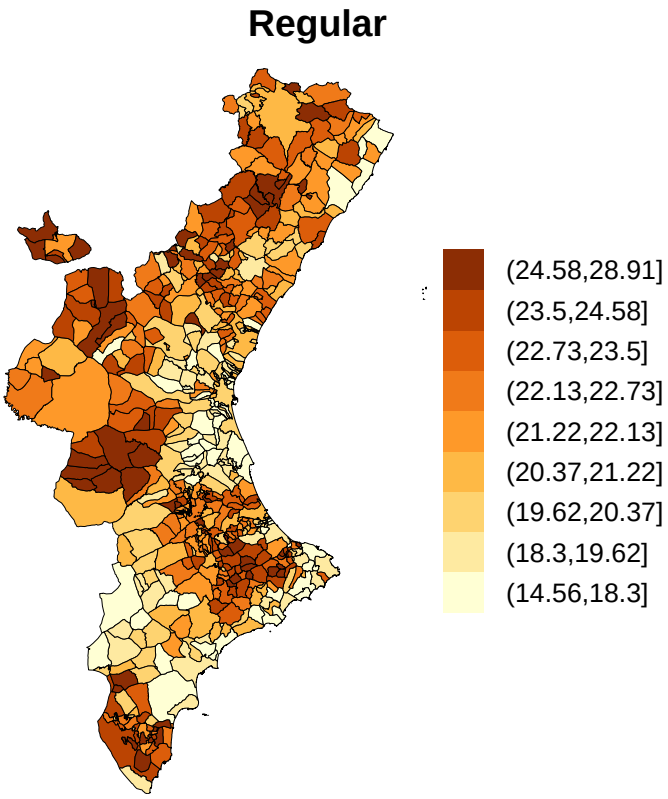
```
      c("category1"),
      main = "Very good",
      col.regions = colorRampPalette(brewer.pal(7,'YlOrBr'))(9),
      cuts = 8,
      colorkey = list(key = list(labels = levels(carto_muni@data$category1)),
                      width = 1.75, cex = 1.5, height = 0.5),
      par.settings = list(axis.line = list(col = 'transparent')),
      col = "black",
      lwd = 0.10)
```



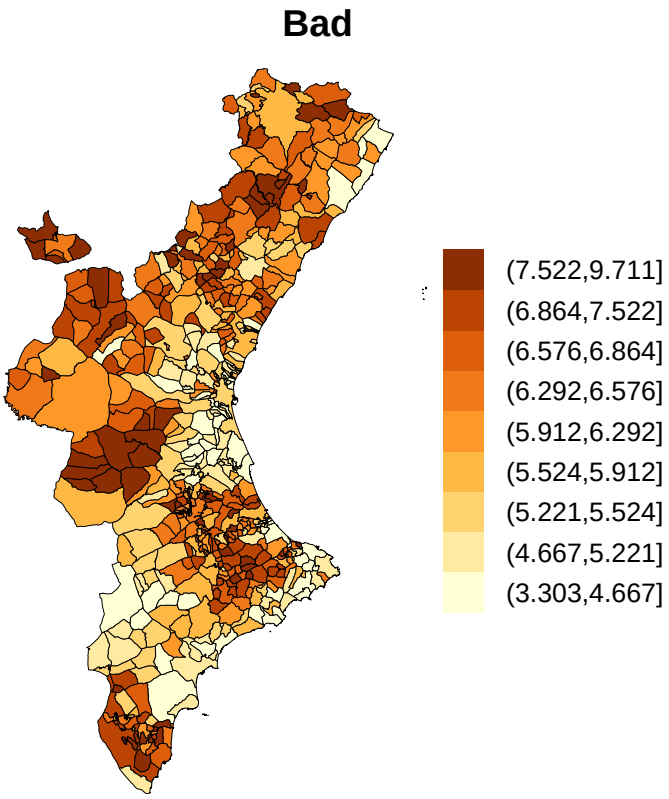

```
spplot(carto_muni,
      c("category2"),
      main = "Good",
      col.regions = colorRampPalette(brewer.pal(7,'YlOrBr'))(9),
      cuts = 8,
      colorkey = list(key = list(labels = levels(carto_muni@data$category2)),
                      width = 1.75, cex = 1.5, height = 0.5),
      par.settings = list(axis.line = list(col = 'transparent')),
      col = "black",
      lwd = 0.10)
```



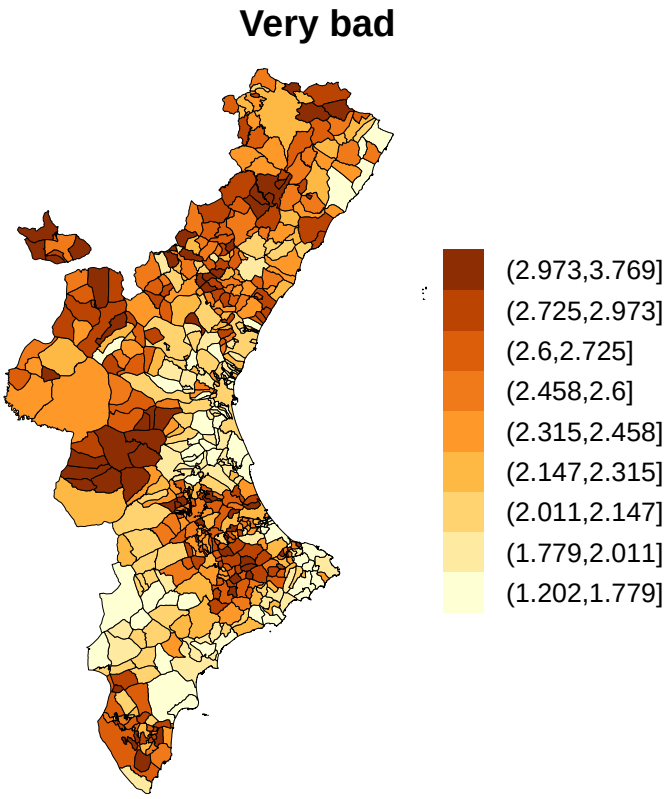

```
spplot(carto_muni,
       c("category3"),
       main = "Regular",
       col.regions = colorRampPalette(brewer.pal(7,'YlOrBr'))(9),
       cuts = 8,
       colorkey = list(key = list(labels = levels(carto_muni@data$category3)),
                       width = 1.75, cex = 1.5, height = 0.5),
       par.settings = list(axis.line = list(col = 'transparent')),
       col = "black",
       lwd = 0.10)
```




```
spplot(carto_muni,
       c("category4"),
       main = "Bad",
       col.regions = colorRampPalette(brewer.pal(7,'YlOrBr'))(9),
       cuts = 8,
       colorkey = list(key = list(labels = levels(carto_muni@data$category4)),
                       width = 1.75, cex = 1.5, height = 0.5),
       par.settings = list(axis.line = list(col = 'transparent')),
       col = "black",
       lwd = 0.10)
```




```
spplot(carto_muni,
       c("category5"),
       main = "Very bad",
       col.regions = colorRampPalette(brewer.pal(7,'YlOrBr'))(9),
       cuts = 8,
       colorkey = list(key = list(labels = levels(carto_muni@data$category5)),
                       width = 1.75, cex = 1.5, height = 0.5),
       par.settings = list(axis.line = list(col = 'transparent')),
       col = "black",
       lwd = 0.10)
```




```
spplot(carto_muni,
       c("agemean"),
       main = "Mean age",
       col.regions = colorRampPalette(brewer.pal(7,'GnBu'))(9),
       cuts = 8,
       colorkey = list(key = list(labels = levels(carto_muni@data$agemean)),
                       width = 1.75, cex = 1.5, height = 0.5),
       par.settings = list(axis.line = list(col = 'transparent')),
       col = "black",
       lwd = 0.10)
```

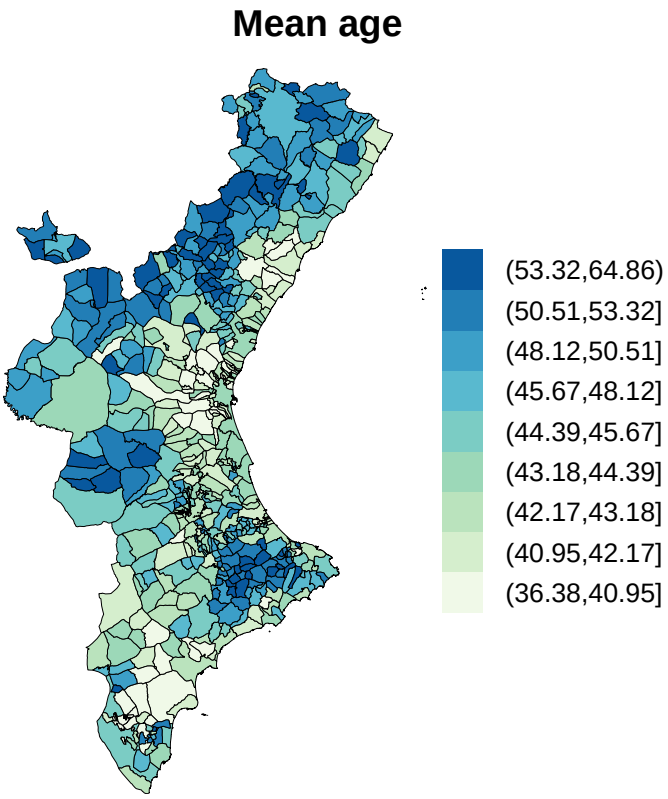


Table 1: Model assessment

```
# 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)
    }
  }
}

# function2:
# - (1) Input: Estimates of the probabilities for each sex, age group and municipality
# from function1; that is, result1$prlevels.
# - (2) We simulate values of the response variable for each sex-age group-municipality
# combination from the corresponding posterior predictive distributions.
# - (3) The estimated number of individuals in each category and municipality is the
# result of adding all the previous values.
# - (4) We compute posterior mean, 95% prediction intervals and observed value of the
# percentage of respondents in each category and municipality.

function2 <- function(prlevels, Muni) {

  n.sims <- salwinbugs$n.sims
  predictive <- array(dim = c(n.sims, NSex, NAges, NCats))
  for (sim in 1:n.sims) {
    for (SexGroup in 1:NSex) {
```

```
      for (AgeGroup in 1:NAges) {
        predictive[sim, SexGroup, AgeGroup, ] <- as.numeric(
          rmultinom(n = 1,
                    size = sum(sample[Muni, SexGroup, AgeGroup]) - sum(is.na(y[muni == Muni & sex
                    ↪  == SexGroup & age == AgeGroup])),
                    prob = prlevels[sim, Muni, SexGroup, AgeGroup, ]))
      }
    }
  }
  predictive.muni <- matrix(nrow = n.sims, ncol = NCats)
  predictive.muni <- apply(predictive, c(1, 4), sum)/(sum(sample[Muni, , ]) - sum(is.na(y[muni
↪  == Muni]))) * 100

  posteriormean <- round(apply(predictive.muni, 2, mean), 2)
  PInterval0.025 <- round(apply(predictive.muni, 2, quantile, prob = 0.025), 2)
  PInterval0.975 <- round(apply(predictive.muni, 2, quantile, prob = 0.975), 2)
  realvalue <- round(table(factor(y[muni == Muni], levels = 1:NCats))/(sum(sample[Muni, , ]) -
↪  sum(is.na(y[muni == Muni]))) * 100, 2)

  return(list("mean" = posteriormean,
              "PI" = list("lower" = PInterval0.025, "upper" = PInterval0.975),
              "real" = realvalue))
}

# Four municipalities with the largest population in the RV
Munis <- order(apply(population, 1, sum), decreasing = TRUE)[1:4]

result2 <- list()
for (Muni in 1:length(Munis)) {
  set.seed(3687241)
  result2[[Muni]] <- function2(prlevels = result1$prlevels, Muni = Munis[Muni])
}

result2
```

```
## [[1]]
## [[1]]$mean
## [1] 12.91 40.57 31.43 10.68  4.41
##
## [[1]]$PI
## [[1]]$PI$lower
## [1] 10.13 36.11 27.53  8.01  2.78
##
## [[1]]$PI$upper
## [1] 16.01 45.10 35.29 13.56  6.21
##
##
## [[1]]$real
##
##     1     2     3     4     5
## 12.09 42.97 32.19  7.35  5.39
##
##
## [[2]]
## [[2]]$mean
## [1] 14.04 41.23 30.38 10.17  4.18
```

```
##
## [[2]]$PI
## [[2]]$PI$lower
## [1]  9.76 35.64 25.15  6.80  2.07
##
## [[2]]$PI$upper
## [1] 18.05 47.04 35.50 13.91  6.51
##
##
## [[2]]$real
##
##     1     2     3     4     5
## 12.13 41.42 34.62  9.17  2.66
##
##
## [[3]]
## [[3]]$mean
## [1] 20.36 45.69 24.42  6.90  2.63
##
## [[3]]$PI
## [[3]]$PI$lower
## [1] 15.57 40.12 19.46  4.19  0.90
##
## [[3]]$PI$upper
## [1] 25.15 51.50 29.64  9.88  4.49
##
##
## [[3]]$real
##
##     1     2     3     4     5
## 29.34 33.83 27.84  5.99  2.99
##
##
## [[4]]
## [[4]]$mean
## [1] 12.30 40.53 30.92 11.31  4.93
##
## [[4]]$PI
## [[4]]$PI$lower
## [1]  7.57 33.51 24.32  6.49  2.16
##
## [[4]]$PI$upper
## [1] 17.84 48.11 37.84 16.76  8.65
##
##
## [[4]]$real
##
##     1     2     3     4     5
## 14.05 37.30 29.19 13.51  5.95
```