BYM2 - Modelling

2023-09-06

Authors 2023

Diego Perez Ruiz has written this code in 2021/2022.

A few minor amendments by Wendy Olsen help us access data for different age groups, years, or sex.

Furthermore, the advisor on all the materials is Arkadiusz Wisniowski, and the research assistant in 2021/22 was Madhu Chauhan.

We thank the funder, University of Manchester - School of Social Sciences.

BYM2 - Modelling

The following R code is a series of commands and operations for spatial data analysis and Bayesian modeling. It involves loading data, setting working directories, preparing data, fitting Bayesian models, and visualizing results. Let's break it down step by step.

Clear the R Environment and Load Packages

In this section, the code begins by clearing the R environment to remove any existing variables. Then, it loads several required R packages:

- `tidyverse`: A collection of packages for data manipulation and visualization.
- `spdep`: A package for spatial dependence analysis.
- `INLA`: The Integrated Nested Laplace Approximations package for Bayesian inference.
- `rstan`: A package for Bayesian data analysis using the Stan modeling language.

```r
# Clear the R environment
rm(list = ls())

# Load necessary packages
library(tidyverse)
library(spdep)
library(INLA)
library(rstan)
```

Define a Custom Operator

```r
# Define a custom '%notin%' operator
`%notin%` <- Negate(`%in%`)
```

This code defines a custom operator `%notin%`, which is used to check if an element is not in a vector or list.

Set Working Directory and Load Data

The `setwd` function is used to set the working directory to the specified path. This directory is where data files are expected to be found.

This code sets a new working directory and loads spatial data from a shapefile named "2011_Dist.shp" using the `st_read` function from the `sf` package. The data is stored in the `India_Districts` object.

This command loads data from an RDA file named "India_Employment_withCensus2011_SampleSize.rda" into the R environment.

Set Working Directory for Source File

```r
setwd("/Users/user/Dropbox/Mac (2)/Desktop/Workshop Files")
source("icar-functions.R")
```

Here, the code sets another working directory, and then it sources an external R script file named "icar-functions.R." This file likely contains custom functions used later in the analysis.

Disable S2 for sf Package

```r
# Disable S2 for sf package
sf::sf_use_s2(FALSE)
```

This command disables the S2 geometry library for the `sf` package. S2 is an advanced library for geometric operations on spatial data.

Prepare Spatial Data

```r
## Load Spatial Data

India_Districts <- st_read("/Users/user/Dropbox/Mac (2)/Desktop/Workshop Files/Census_2011/2011_Dist.shp
```

```
## Reading layer `2011_Dist' from data source
##   `/Users/user/Dropbox/Mac (2)/Desktop/Workshop Files/Census_2011/2011_Dist.shp'
##   using driver `ESRI Shapefile'
## Simple feature collection with 641 features and 5 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 68.18625 ymin: 6.755953 xmax: 97.41529 ymax: 37.07827
## Geodetic CRS:  WGS 84
```

```r
# Extract geometry
sp <- India_Districts$geometry
```

Here, the code extracts the geometry (geographic shapes) from the loaded spatial data and stores it in the `sp` object.

Prepare Data for ICAR Function in Stan

```
C <- spdep::nb2mat(spdep::poly2nb(sp, queen = TRUE), style = "B", zero.policy = TRUE)
icar.data <- prep_icar_data(C)
```

This section prepares the data for the Integrated Conditional Autoregressive (ICAR) model. It calculates the binary neighbor matrix, `C`, based on the spatial relationships between districts. The `prep_icar_data` function processes this matrix to prepare data required for the ICAR model.

Calculate Scale Factors

```
# Notice that the scale_factor is just ones.
icar.data$inv_sqrt_scale_factor
```

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

```
# Calculate the scale factor for each connected group of nodes using scale_c function
k <- icar.data$k
scale_factor <- vector(mode = "numeric", length = k)
for (j in 1:k) {
  g.idx <- which(icar.data$comp_id == j)
  if (length(g.idx) == 1) {
    scale_factor[j] <- 1
    next
  }
  Cg <- C[g.idx, g.idx]
  scale_factor[j] <- scale_c(Cg)
}
```

This code calculates scale factors for each connected group of nodes in the spatial data. It iterates through the connected components (`comp_id`) and computes the scale factor using the `scale_c` function. The scale factors are then stored in `scale_factor`.

Update Data for Stan

```
# Update the data list for Stan
icar.data$inv_sqrt_scale_factor <- 1 / sqrt(scale_factor)

# Display the new scale factors
print(icar.data$inv_sqrt_scale_factor)
```

```
## [1] 1.124418 1.000000 1.000000 1.000000 1.000000
```

Here, the code updates the `inv_sqrt_scale_factor` in the data list `icar.data` by taking the reciprocal square root of the previously calculated scale factors. This step is likely necessary for model compatibility.

Data List for Stan

```
# Load data
load("India_Employment_withCensus2011_SampleSize.rda")

# Data list for Stan
n <- nrow(India_Districts)
```

This section defines the data list for the Stan model. It calculates the number of districts, `n`, which will be used in the Bayesian model.

Join Data Frames

```
# Join data frames
India_Data_Employment_Census2011 <- left_join(India_Districts, India_Sample_Employment, by = "censuscode
```

This code performs a left join between the spatial data frame (`India_Districts`) and a previously loaded data frame (`India_Sample_Employment`) based on the “censuscode” column. The result is stored in `India_Data_Employment_Census2011`.

Fill Missing Values

```
# Fill missing values with zero
India_Data_Employment_Census2011$Freq_State[is.na(India_Data_Employment_Census2011$Freq_State)] <- 0
India_Data_Employment_Census2011$Prop_Females[is.na(India_Data_Employment_Census2011$Prop_Females)] <- 0
India_Data_Employment_Census2011$Prop_Rural[is.na(India_Data_Employment_Census2011$Prop_Rural)] <- 0
```

Here, missing values in the “Freq_State” column of the `India_Data_Employment_Census2011` data frame are replaced with zeros.

Define Data List for Stan

```
# Define data list

dl <- list(
  n = nrow(India_Data_Employment_Census2011),
  y = India_Data_Employment_Census2011$Freq_State,
  x1 = India_Data_Employment_Census2011$Prop_Females,
  x2 = India_Data_Employment_Census2011$Prop_Rural,
  offset = rep(1, n),
  prior_only = 0
)


dl <- c(dl, icar.data)
```

This section defines the data list `dl` for use in the Stan model. It includes various components like the number of data points `n`, the observed data `y`, an offset vector, and additional data from `icar.data`.

Compile and Sample from Stan Model

```
#Compile the Stan model
BYM2 <- stan_model("/Users/user/Dropbox/Mac (2)/Desktop/Workshop Files/BYM2.stan")

# Sample from the model
fit <- sampling(BYM2, data = dl, chains = 4, cores = 4,
                control=list(adapt_delta = 0.97, stepsize = 0.1),
                warmup=9000, iter=10000, save_warmup=FALSE)
```

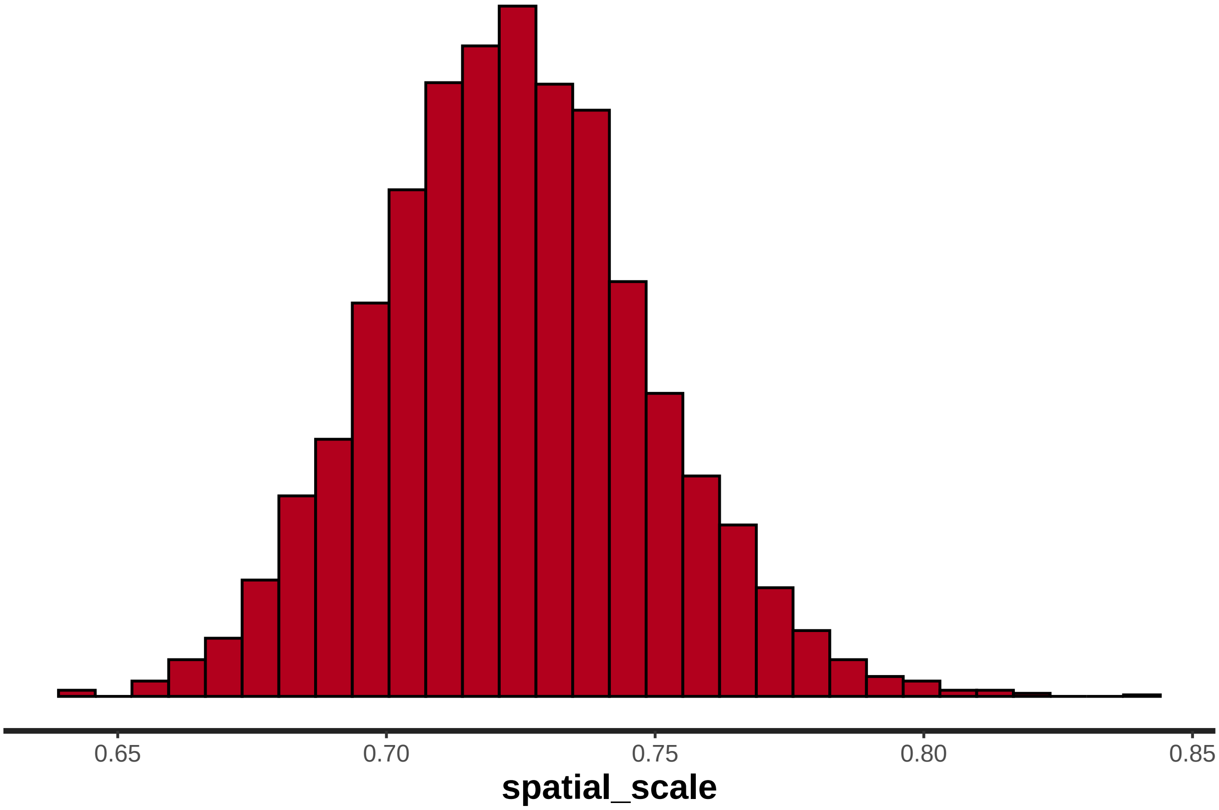
This code compiles a Stan model specified in the "BYM2.stan" file and then samples from the model using the `sampling` function. It specifies four chains with four cores for parallel processing.

View and Plot Results

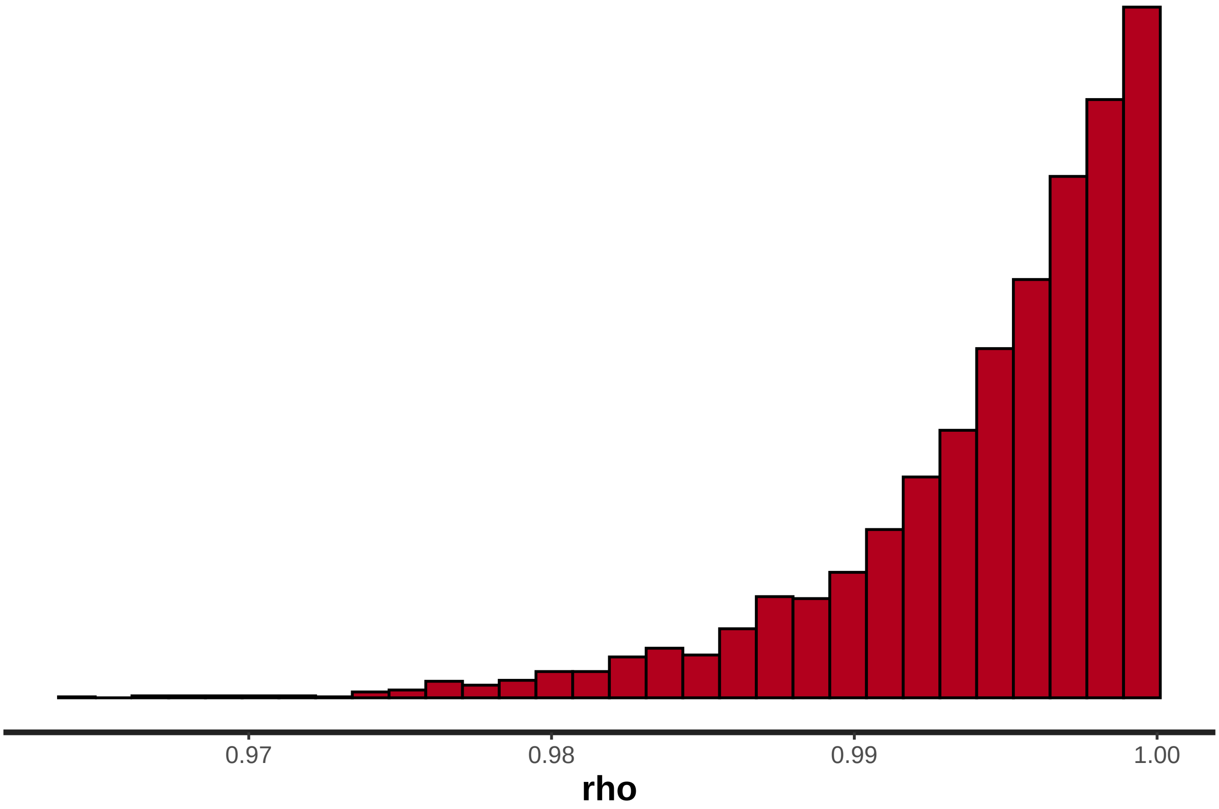
```
# View some results
#summary(fit)

print(fit, pars=c("alpha", "beta1", "beta2"), probs=c(0.025, 0.5, 0.975));
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=10000; warmup=9000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##        mean se_mean   sd  2.5%   50% 97.5% n_eff Rhat
## alpha -1.27    0.01 0.31 -1.88 -1.26 -0.68  1582    1
## beta1  4.21    0.01 0.32  3.61  4.20  4.83  1658    1
## beta2  2.42    0.02 0.46  1.51  2.42  3.32   653    1
##
## Samples were drawn using NUTS(diag_e) at Tue Sep 12 10:52:14 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
# View some results from the joint prior probability
#plot(fit, pars = "phi_tilde")
#plot(fit, pars = "convolution")
plot(fit, pars = "spatial_scale", plotfun = "hist")
```



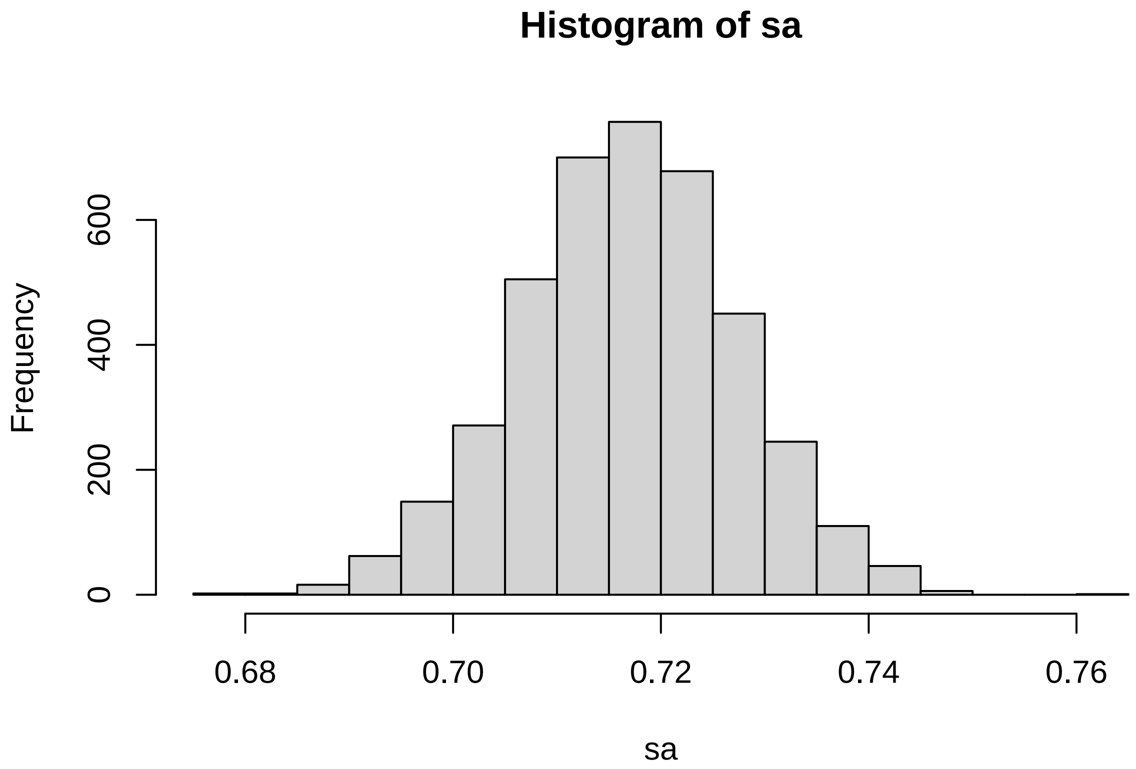
```
plot(fit, pars = "rho", plotfun = "hist")
```



This section displays summary statistics and plots related to the Bayesian model fit using Stan. It provides insights into the model's parameters and convergence.

Calculate Spatial Autocorrelation

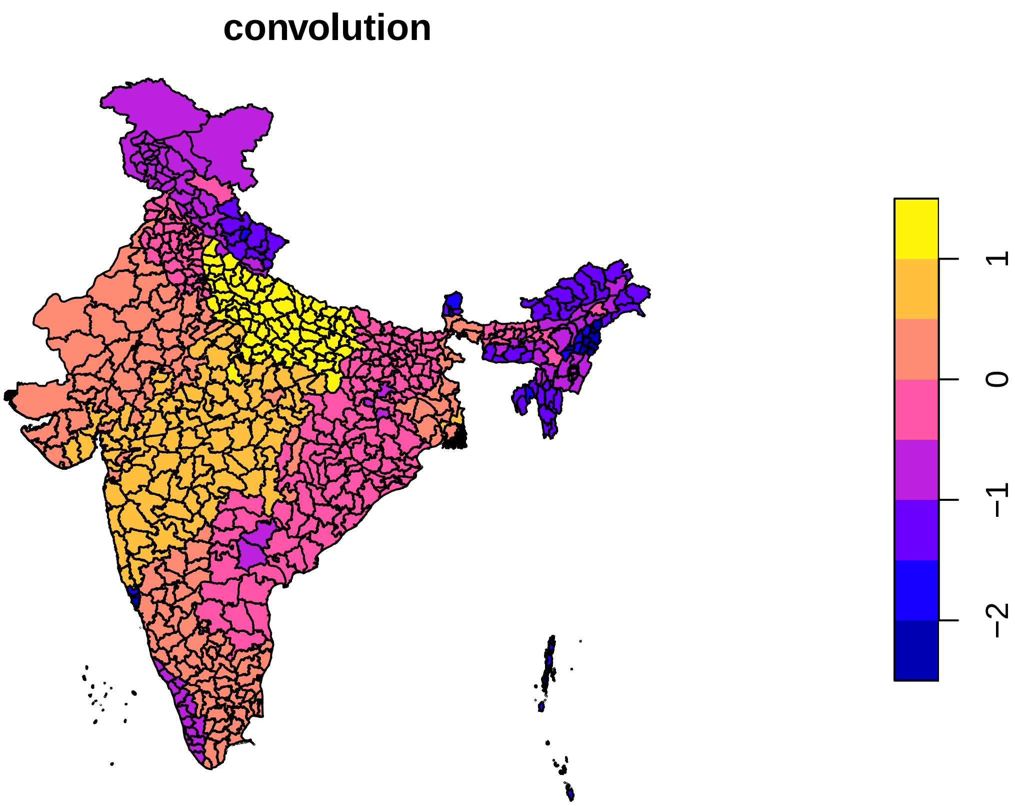
```r
# Calculate the degree of spatial autocorrelation (SA) in the convolution term
convolution <- as.matrix(fit, pars = "convolution")
sa <- apply(convolution, 1, mc, w = C)
hist(sa)
```



Here, the code calculates the degree of spatial autocorrelation (SA) in the convolution term of the model and plots a histogram of SA values.

Create a Posterior Mean Map

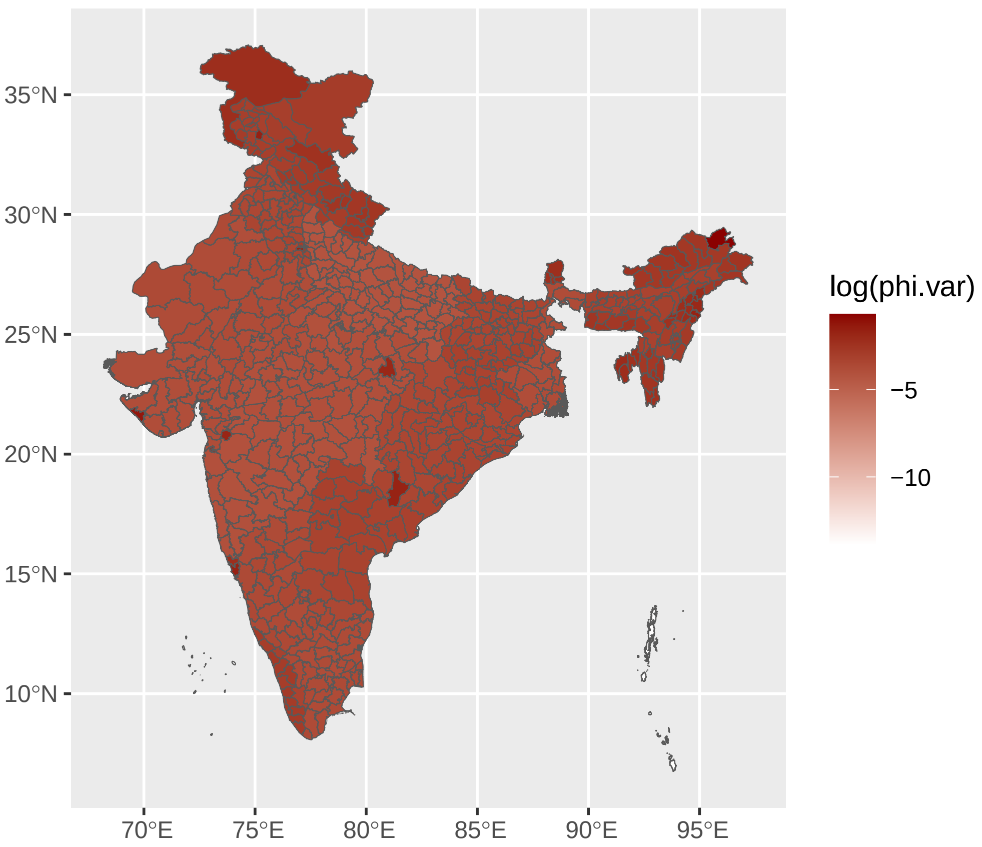
```
# Create a simple map of the posterior mean of the convolution term
spx <- st_as_sf(sp)
spx$convolution <- apply(convolution, 2, mean)
plot(spx[, "convolution"])
```



This section creates a simple map using the spatial data and the posterior mean of the convolution term from the Bayesian model. The result is a map representing the posterior mean.

Plot a Map with Variance

```
# Calculate D_diag and phi.var for plotting
D_diag <- rowSums(C)
phi.samples <- as.matrix(fit, pars = "phi_tilde")
phi.var <- apply(phi.samples, 2, var)

# Plot a map
ggplot(India_Districts) +
  geom_sf(aes(fill = log(phi.var))) +
  scale_fill_gradient(
    low = "white",
    high = "darkred"
  )
```



In this final section, the code calculates `D_diag` and `phi.var` for plotting purposes and then creates a map using the `ggplot2` package. The map color represents the logarithm of the variance of the `phi` parameter.

Overall, this R code conducts spatial data analysis and Bayesian modeling on Indian employment data, including spatial autocorrelation analysis, model fitting, and visualization of results.