Modelling spatial data in R with CARBayes

Part 2: Modelling spatial data

Duncan Lee and Eilidh Jack

---

1. Introduction

This practical session will show you how to undertake spatial modelling in R using CARBayes. Specifically, this session will cover:

- Fitting and checking convergence of spatial correlation models using the CARBayes software package.
- Summarising the results from a fitted spatial model.

2. Recap from the first practical session

Before we fit a spatial model to the data we need to read the data and shapefiles into R, create the spatialPolygonsDataFrame object, and create the neighbourhood matrix $\mathbf{W}$, all of which we did in the first practical session. The data and shapefiles can be read in using the following code.

```
#### Data
dat <- read.csv(file="Scotland spatial data.csv")

#### Shapefiles
library(shapefiles)
shp <- read.shp(shp.name = "ScotlandIZ.shp")
dbf <- read.dbf(dbf.name = "ScotlandIZ.dbf")
```

Then they can be combined together into a spatialPolygonsDataFrame object using the code:

```
library(sp)
library(CARBayes)
```

```
rownames(dat) <- dat$IZ
dat$IZ <- NULL
sp.dat <- combine.data.shapefile(data=dat, shp=shp, dbf=dbf)
```

Then finally the neighbourhood matrix can be constructed using the code:

```
library(spdep)
W.nb <- poly2nb(sp.dat, row.names = rownames(sp.dat@data))
W <- nb2mat(W.nb, style = "B")
```

3. Modelling spatial data with `CARBayes`

Recall that the data we have are:

```
head(sp.dat@data)
```

```
##             Y         E   jsa ethnic      no2
## S02000260 90  93.19477 4.600   7.54 16.13495
## S02000261 20  43.78443 1.775   6.27 15.26339
## S02000262 58  92.03014 1.800   9.73 15.26339
## S02000263 43  81.48188 1.200  16.12 17.25486
## S02000264 52 122.64095 2.150   5.83 16.00148
## S02000265 24  56.49576 2.000   7.44 15.42137
```

where

- `Y` - is the observed numbers of hospital admissions for respiratory disease and is the count variable response (dependent variable).
- `E` - is the estimated expected numbers of hospital admissions and is treated as a fixed offset.
- `jsa, ethnic, no2` - are three covariates (independent variables) representing poverty, non-white ethnicity and air pollution respectively.

Therefore given the response is a count, we wish to fit the following general model:

$$\begin{aligned} \mathtt{Y}_k &\sim \text{Poisson}(\mathtt{E}_k\theta_k) \\ \ln(\theta_k) &= \beta_1 + \beta_2 \mathtt{jsa}_k + \beta_3 \mathtt{ethnic}_k + \beta_4 \mathtt{no2}_k + \phi_k, \end{aligned} \tag{1}$$

where here θ_k is the risk of disease in area k relative to the expected number of admissions $\mathtt{E}_k$. It is on the same scale as the SMR, so here if $\theta_k = 1.1$ then area k has a 10% increased risk of respiratory hospitalisation compared to the expected counts, where as if $\theta_k = 0.8$ then there is a 20% reduced risk. To illustrate model fitting, we are going to fit the CAR model proposed by Leroux, Lei, and Breslow (2000) for ϕ_k, which is given by

$$\phi_k|\boldsymbol{\phi}_{-k}, \mathbf{W} \sim \mathrm{N}\left(\frac{\rho\sum_{j=1}^{K} w_{kj}\phi_j}{\rho\sum_{j=1}^{K} w_{kj} + 1 - \rho}, \frac{\tau^2}{\rho\sum_{j=1}^{K} w_{kj} + 1 - \rho}\right),$$

where here ρ is a spatial dependence parameter with

- $\rho = 0$ corresponding to independence and
- $\rho = 1$ corresponding to strong spatial correlation.

This model can be fitted in a Bayesian setting using Markov chain Monte Carlo (MCMC) simulation using the `S.CARleroux()` function from the `CARBayes` package, which requires (at a minimum) the following arguments.

- `formula` - specifies the response, covariates and offset to include in the model.
- `family` - what data likelihood model to fit, in this case a Poisson log-linear model.
- `data` - where the data (response, covariates, offset) are stored. This is not formally required if each variable is not stored in the same data set.
- `W` - the neighbourhood matrix $\mathbf{W}$.
- `burnin` - the number of samples to throw away as the burnin period.
- `n.sample` - the total number of samples to generate.

The first step in fitting the model is to specify the response, covariate and offset components, using the `formula` argument. This is done in the same way as the `lm()` and `glm()` functions in `R`. However fixed offsets (here `E`) are specified via the `offset()` function on the log scale. Thus we actually fit the equivalent model

$$\begin{aligned} \mathtt{Y}_k &\sim \text{Poisson}(\mu_k) \\ \ln(\mu_k) &= \ln(\mathtt{E}_k) + \beta_1 + \beta_2\mathtt{jsa}_k + \beta_3\mathtt{ethnic}_k + \beta_4\mathtt{no2}_k + \phi_k. \end{aligned} \tag{2}$$

This is the same as model (1) because $\mu_k = \mathtt{E}_k\theta_k$ is the fitted value for area k as before. The formula argument for this model can be specified via the following code:

```
formula <- Y ~ offset(log(E)) + jsa + ethnic + no2
```

The model can be fitted using the following code, where the `print()` function prints a summary of the model to the screen. The `verbose=FALSE` argument stops the function updating the user on its progress, which is purely done to make this document look nice! I recommend setting `verbose=TRUE` (the default value) so you can see how long the function has left to run.

```
model <- S.CARleroux(formula=formula, family="poisson", data=sp.dat@data, W=W,
    burnin=20000, n.sample=100000, verbose=FALSE)
print(model)
```

```
##
## #################
## #### Model fitted
```

```
## #################
## Likelihood model - Poisson (log link function)
## Random effects model - Leroux CAR
## Regression equation - Y ~ offset(log(E)) + jsa + ethnic + no2
## Number of missing observations - 0
##
## ############
## #### Results
## ############
## Posterior quantities and DIC
##
##              Median    2.5%   97.5% n.sample % accept n.effective
## (Intercept) -0.6682 -0.7685 -0.5644    80000     45.3      1610.8
## jsa          0.0953  0.0847  0.1057    80000     45.3      2508.8
## ethnic      -0.0004 -0.0030  0.0022    80000     45.3      1913.0
## no2          0.0077  0.0015  0.0139    80000     45.3      1286.9
## tau2         0.0594  0.0388  0.0898    80000    100.0      2670.6
## rho          0.3193  0.1130  0.6360    80000     45.8      2029.0
##             Geweke.diag
## (Intercept)         2.1
## jsa                -0.1
## ethnic             -0.5
## no2                -1.6
## tau2               -2.1
## rho                -1.3
##
## DIC =  2135.557       p.d =  182.8932       Percent deviance explained =  59.98
```

The output from the `print()` function is split into 2 sections. The first section `Model fitted` displays the model that has been fitted, which includes the choice of covariates, the data likelihood model and the random effects model. The second section presents the results, which includes both parameter summaries for key parameters and overall model fit criteria such as the Deviance Information Criterion (`DIC`, Spiegelhalter et al. (2002)) with the effective number of parameters (`p.d`), and the percentage of the deviance (variation) explained by the model. The summary table of the key model parameters (all parameters except the random effects ϕ) contains the following information:

- `Median` - point estimate for the parameter, which is the median of the samples generated.
- `(2.5%, 97.5%)` - 95% credible interval for the parameter.
- `n.sample` - the number of post burnin samples generated.
- `% accept` - the acceptance probability for the Markov chain.
- `n.effective` - the effective number of independent samples generated, as the set of samples generated are correlated.
- `Geweke.diag` - the convergence diagnostic for the samples proposed by Geweke (1992), which is in the form of a Z-score. Values within the interval (-1.96, 1.96) are indicative of convergence.

The fitted model object `model` is an `R list` object, which contains the following elements as shown via the `summary` function.

```
summary(model)
```

```
##                      Length Class      Mode
## summary.results       42   -none-     numeric
## samples                6   -none-     list
## fitted.values        271   -none-     numeric
## residuals              3   data.frame list
## modelfit               7   -none-     numeric
## accept                 4   -none-     numeric
## localised.structure    0   -none-     NULL
## formula                3   formula    call
## model                  2   -none-     character
## X                   1084   -none-     numeric
```

A description of the key elements in this list is given below.

- `summary.results` - the summary table of results produced when using the `print()` function.
- `samples` - a list of the parameter samples generated by the model.
- `fitted.values` - a vector of fitted values (μ_k values).
- `residuals` - a matrix with 3 different types of residuals, response, pearson and deviance.
- `modelfit` - a vector containing model fit criteria including the DIC and LMPL.

4. Checking convergence of the MCMC simulation

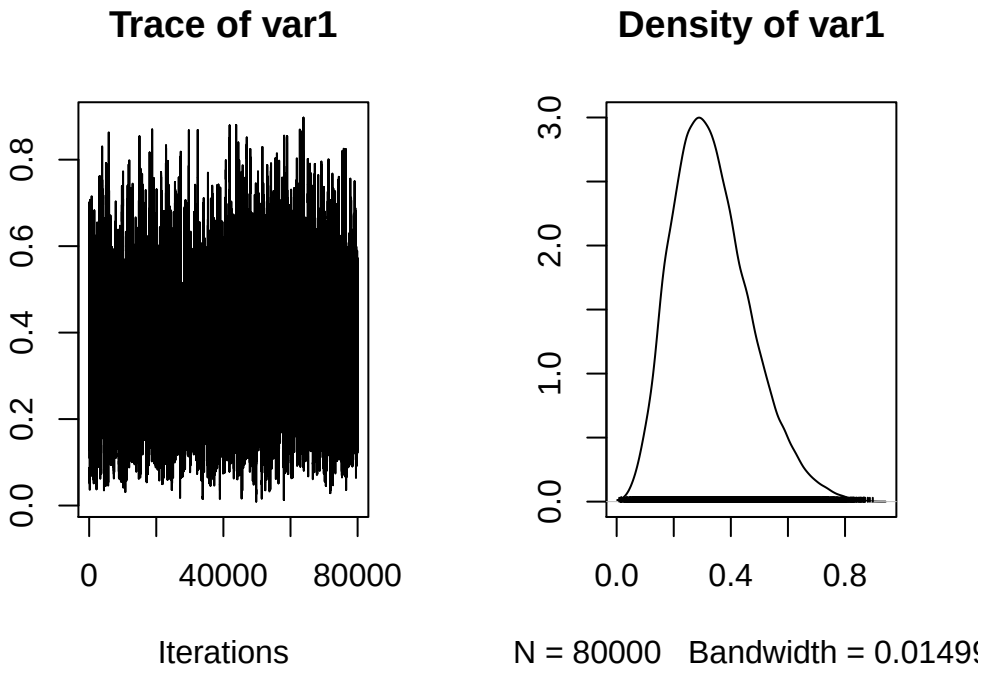
The convergence of the MCMC samples can be assessed by viewing traceplots of the samples for certain parameters. The set of samples for a given parameter have converged if they show no trend and random scatter above and below the average value. The samples are stored in the `samples` element of the `R list` object `model`, which for this model has elements

```
summary(model$samples)
```

```
##        Length   Class Mode
## beta     320000 mcmc  numeric
## phi    21680000 mcmc  numeric
## tau2      80000 mcmc  numeric
## rho       80000 mcmc  numeric
## fitted 21680000 mcmc  numeric
## Y             1 mcmc  logical
```

which correspond to the different parameters in the model. For example, to plot the traceplot for the spatial dependence parameter ρ use the following code.

```
plot(model$samples$rho)
```



The left plot is the traceplot which shows no trend and hence convergence, while the right plot shows a density estimate of the samples. Additionally, these samples show the estimated value of rho (ρ) is close to 0.3, suggesting the spatial dependence in these data after adjusting for the covariates is weak to moderate.

5. Inference from the model

The first quantity you may be interested in are overall measures of model fit, particularly if you are interested in comparing different models. These can be accessed via the code:

```
model$modelfit
```

```
##                             DIC                             p.d
##                      2135.55730                       182.89322
##                            WAIC                             p.w
##                      2121.04642                       124.33723
##                            LMPL                      loglikelihood
##                      -912.98041                        -884.88543
## Percentage deviance explained
##                       59.98039
```

The next quantities you may be interested in are the fitted values and the residuals, which can be accessed as described below.

- fitted values using `model$fitted.values` or the `fitted.values(model)` function.

- residuals of various types (response, pearson, deviance) using `model$residuals` or using the `residuals(model, type=...)` function.

For example, you can assess the presence of spatial correlation in the residuals using the following code.

```
W.list <- nb2listw(W.nb, style = "B")
moran.mc(x = residuals(model, type="pearson"), listw = W.list, nsim = 10000)
```

```
##
##  Monte-Carlo simulation of Moran I
##
## data:  residuals(model, type = "pearson")
## weights: W.list
## number of simulations + 1: 10001
##
## statistic = -0.038282, observed rank = 1790, p-value = 0.821
## alternative hypothesis: greater
```

One element of interest from fitting this model are the effects of the covariates on disease risk, which are typically presented as relative risks. For example, estimated relative risks and 95% credible intervals for a 1 unit increase in each covariate can be obtained via the code:

```
exp(model$summary.results[2:4 , 1:3])
```

```
##           Median      2.5%    97.5%
## jsa    1.0999888 1.0883905 1.111488
## ethnic 0.9996001 0.9970045 1.002202
## no2    1.0077297 1.0015011 1.013997
```

So for example, a 1% increase in the percentage of the working age population claiming Job Seekers Allowance (JSA) is associated with a 10% increase in disease risk.

The next quantity of interest is the estimated risks, which are computed as $\theta_k = \mu_k/E_k$, where μ_k are the fitted values. We add the estimated risks to the spatial data set to enable plotting using the code:

```
sp.dat@data$risk <- model$fitted.values / sp.dat@data$E
```

The other quantity that is often mapped is the posterior exceedence probability (PEP), which is the probability that each area exceeds the average risk of one given the data. These can be computed using the code below, where the first two lines compute the posterior distributions of risk for each area $(\theta_1, \ldots, \theta_K)$, while the third line uses the function `summarise.samples()` (from the `CARBayes` package) to compute the exceedence probabilities and add them to the `sp.dat` object.

```
m <- nrow(model$samples$fitted)
risk <- model$samples$fitted / matrix(rep(sp.dat@data$E,m),
    nrow=m, byrow=T)
```

```
sp.dat@data$PEP <- as.numeric(summarise.samples(risk, exceedences=1)
    $exceedences)
```

Now these two quantities have been added to the spatial data set, we transform it to a `data.frame` object using code similar to that seen earlier.

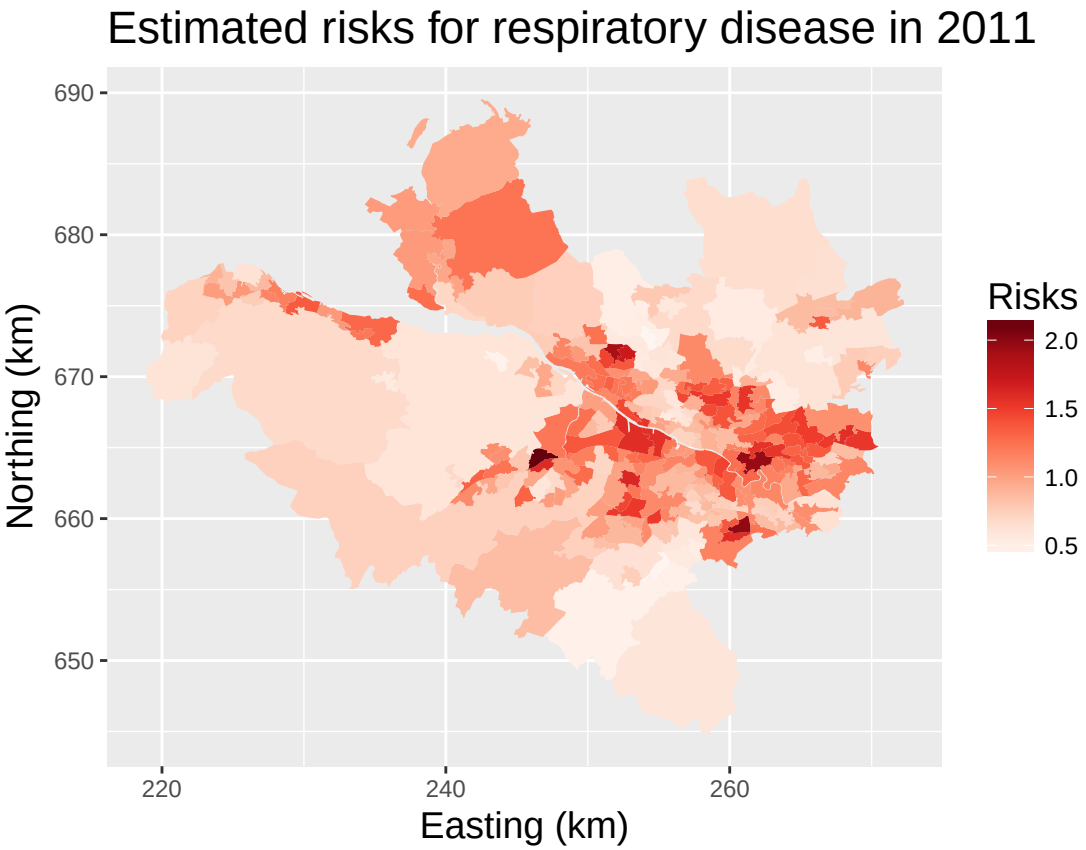
```
#### Load the libraries required
library(ggplot2)
library(rgeos)
library(maptools)

#### Turn into a data.frame
sp.dat@data$id <- rownames(sp.dat@data)
temp1 <- fortify(sp.dat, region = "id")
sp.dat2 <- merge(temp1, sp.dat@data, by = "id")


#### Change the scale to kilometres
sp.dat2$long <- sp.dat2$long / 1000
sp.dat2$lat <- sp.dat2$lat / 1000
```

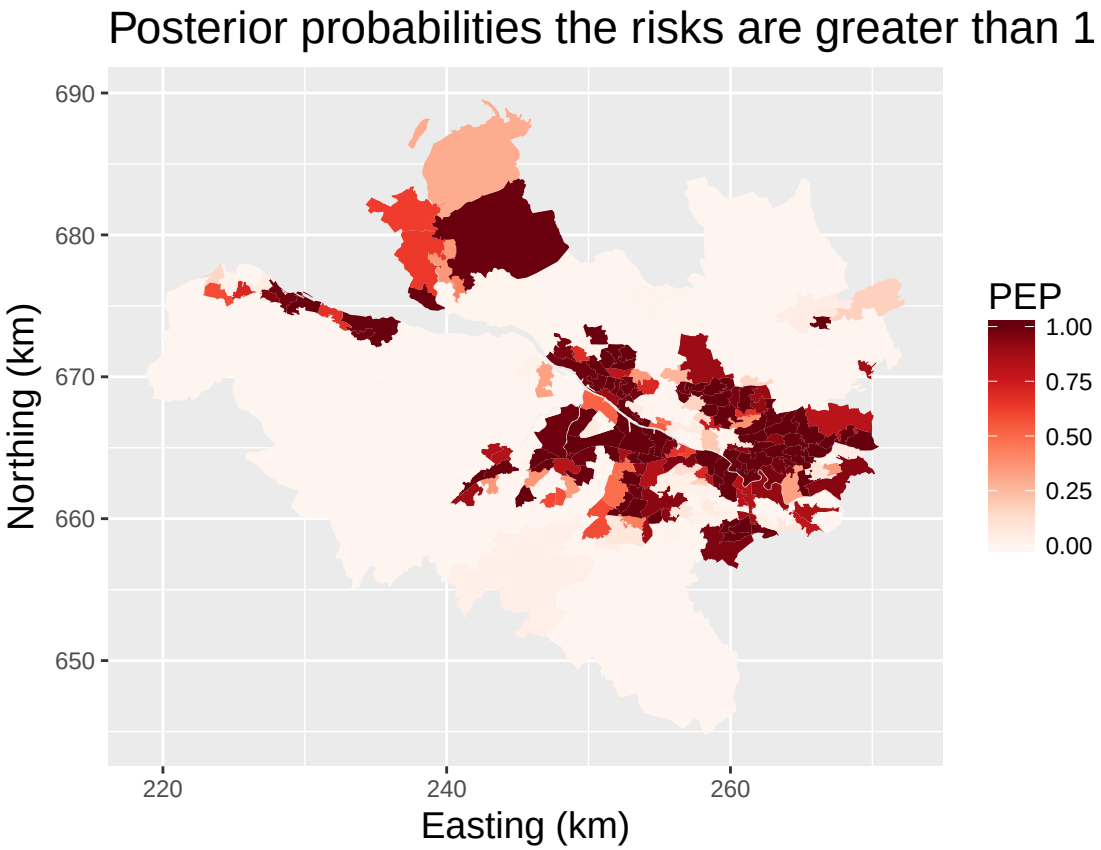
Then the estimated risk surface can be plotted using the following code:

```
library(RColorBrewer)
ggplot(data = sp.dat2, aes(x=long, y=lat, group=group, fill = risk)) +
    geom_polygon() +
    coord_equal() +
    xlab("Easting (km)") +
    ylab("Northing (km)") +
    labs(title = "Estimated risks for respiratory disease in 2011",
    fill = "Risks") +
    theme(title = element_text(size=14)) +
    scale_fill_gradientn(colors=brewer.pal(n=9, name="Reds"))
```



The map is essentially a smoother version of the SMR from the first practical session. Finally, the estimated posterior exceedence probabilities can be mapped using the following code.

```
ggplot(data = sp.dat2, aes(x=long, y=lat, goup=group, fill = PEP)) +
  geom_polygon() +
  coord_equal() +
  xlab("Easting (km)") +
  ylab("Northing (km)") +
  labs(title = "Posterior probabilities the risks are greater than 1",
  fill = "PEP") +
  theme(title = element_text(size=14)) +
  scale_fill_gradientn(colors=brewer.pal(n=9, name="Reds"))
```



The figure shows the model is generally very sure if the risk is greater or less than 1, with the majority of the PEP values being close to 0 or close to 1. The spatial pattern in this map corresponds to that in the estimated risk map above as expected.

References

Geweke, John. 1992. "Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments." In *Bayesian Statistics*, 169–93. University Press.

Leroux, Brian G., Xingye Lei, and Norman Breslow. 2000. "Statistical Models in Epidemiology, the Environment, and Clinical Trials." In, 179–91. Springer-Verlag, New York. http://dx.doi.org/10.1007/978-1-4612-1284-3_4.

Spiegelhalter, D, N Best, B Carlin, and A Van der Linde. 2002. "Bayesian Measures of Model Complexity and Fit." *Journal of the Royal Statistical Society B* 64: 583–639.