Spatial Statistics 2021 - 4H and 5M

Professor Duncan Lee

Computer lab 2 - areal unit modelling

1. Introduction

In this lab we analyse spatial data from a study investigating the spatial pattern in cancer risk in Greater Glasgow, Scotland, between 2001 and 2005. The study region is the Greater Glasgow and Clyde health board, which contains the city as well as the surrounding area. The health board is split up into $n = 271$ administrative units called Intermediate Geographies (IG), which have a median area of 124 hectares and a median population of 4,239. The disease data we will model are the numbers of new cancer cases diagnosed between 2001 and 2005 for each of the 271 intermediate geographies that comprise our study region. The expected numbers of cases are calculated by indirect standardisation, using age and sex adjusted rates for the whole of Scotland. A small number of covariates are available to describe the spatial variation in cancer risk across Greater Glasgow, and the data for this study come in the following two parts.

glasgowdata.csv - contains the response and covariate data. The data are as follows:

- **IG** - a unique identifier for each intermediate geography.
- **Y** - the observed number of cancer cases.
- **E** - the expected number of cancer cases based on national age and sex specific disease rates and computed by indirect standardisation.
- **pm10** - the average particulate matter air pollution concentration.
- **smoke** - the estimated percentage of the population that smoke.
- **ethnic** - the percentage of school children who are non-white.

SG - a shapefile containing multiple files with the same name but different file extensions (e.g. *.shp, .dbf, .shx,* etc) that contains the polygons that make up the set of Intermediate Geographies (IG) for all of Scotland.

To model areal data in *R* we use the *CARBayes* package (Lee (2013)). The package has a user guide that is provided with this lab handout.

Aim

In analysing these data our aims are to:

- Visualise the high-risk areas for cancer by mapping.
- Check whether the data exhibit residual spatial autocorrelation after covariate adjustment.
- Estimate the effects of the covariates on cancer risk whilst allowing for spatial autocorrelation.

2. Data input and formatting

First read in the data using the commands:

```
#### Read in the data
dat <- read.csv(file="glasgowdata.csv")
head(dat)
```

```
         IG   Y         E pm10 smoke ethnic
1 S02000260 133 106.17907 17.8  21.9   5.58
2 S02000261  38  62.43131 18.6  21.8   7.91
3 S02000262  97 120.00694 18.6  20.8   9.58
4 S02000263  80 109.10245 17.0  14.0  10.39
5 S02000264 181 149.77821 18.6  15.2   5.67
6 S02000265  77  82.31156 17.0  14.6   5.61
```

which shows that the data contain the variables listed above. As discussed in lectures the exploratory measure of disease risk is the standardised morbidity ratio (SMR), which for area i is given by

$$\text{SMR}_i = \frac{Y_i}{E_i}.$$

Compute the SMR for these data and add it as an additional column to the *data.frame dat* using the following code.

```
dat$smr <- dat$Y / dat$E
```

The shapefiles can be read in using the commands:

```
#### Read in the shapefile
library(sp)
library(rgdal)
shape <- readOGR(dsn = "SG.shp")
```

```
Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS =
dumpSRS, : Discarded datum OSGB_1936 in CRS definition: +proj=tmerc +lat_0=49
+lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +ellps=airy +units=m +no_defs
```

```
OGR data source with driver: ESRI Shapefile
Source: "/Users/duncanlee/OneDrive - University of Glasgow/teaching/Spatial statistics 2021/Computer la
with 1235 features
It has 5 fields
```

This object *shape* is a *SpatialPolygonsDataFrame* object, which can be seen from

```
class(shape)
```

```
[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"
```

This essentially combines the set of polygons representing the IGs, with limited pre-loaded data. The data element can be viewed by:

```
head(shape@data)
```

```
    IZ_CODE                              IZ_NAME STDAREA_HA Shape_Leng
0 S02000001                           Cove South   97.37491   6346.393
1 S02000002     Kincorth, Leggart and Nigg South  601.39718  19559.865
2 S02000003                               Culter 1975.00798  32535.137
3 S02000004 Cults, Bieldside and Milltimber East  555.75168  18431.092
```

```
4 S02000005                           Cove North  675.58618  22568.155
5 S02000006     Kincorth, Leggart and Nigg North  122.84323   7955.546
  Shape_Area
0   952193.8
1  6005769.5
2 19724106.4
3  5525075.2
4  6738951.0
5  1205903.4
```

The next step is to transform the coordinate reference system from metres (easting and northing) to degrees (longitude and latitude), which makes it compatible with the data visualisation later on. This is achieved via the code:

```
proj4string(shape)
```

```
Warning in proj4string(shape): CRS object has comment, which is lost in output
```

```
[1] "+proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +ellps=airy +units=m +no_
```

```
shape2 <- spTransform(shape, CRS("+proj=longlat +datum=WGS84 +no_defs"))
proj4string(shape2)
```

```
Warning in proj4string(shape2): CRS object has comment, which is lost in output
```

```
[1] "+proj=longlat +datum=WGS84 +no_defs"
```

Then use the following code to combine the data set and shapefiles together.

```
sp.dat <- merge(shape2, dat, all.x=FALSE, by.x="IZ_CODE", by.y="IG")
class(sp.dat)
```

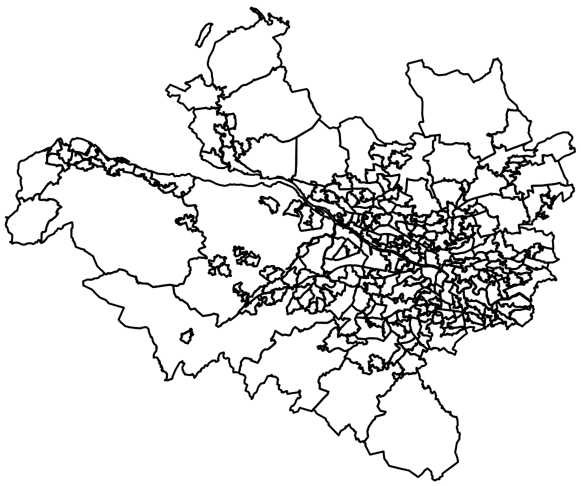
```
[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"
```

The resulting object *sp.dat* is also a *SpatialPolygonsDataFrame* object. Note that if you look at the data and you plot the object you get the following results

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

```
    IZ_CODE                       IZ_NAME STDAREA_HA Shape_Leng Shape_Area   Y
1 S02000260                    Auchinairn  112.29253   7751.798  1107840.6 133
2 S02000261                 Woodhill East  111.57629   6464.223  1104265.0  38
3 S02000262                 Woodhill West  107.23367   7316.999  1065621.6  97
4 S02000263                Westerton East  133.78761   5167.311  1324364.3  80
5 S02000264 Bishopbriggs West and Cadder  209.41393  13999.507  2068431.5 181
6 S02000265                Westerton West   73.94589   5863.684   718460.3  77
          E pm10 smoke ethnic       smr
1 106.17907 17.8  21.9   5.58 1.2526009
2  62.43131 18.6  21.8   7.91 0.6086690
3 120.00694 18.6  20.8   9.58 0.8082866
4 109.10245 17.0  14.0  10.39 0.7332558
5 149.77821 18.6  15.2   5.67 1.2084534
6  82.31156 17.0  14.6   5.61 0.9354700
```

```
plot(sp.dat)
```



This raises 2 points:

- The data set is now a combination of the pre-loaded data and the data from *dat*.
- The set of IGs from *shape2* relates to all of Scotland, but as *dat* only relates to the Greater Glasgow and Clyde health board, the combined object is also similarly geographically restricted.

3. Mapping spatial data

Once the data have been read into *R* the natural first step is to draw a map, and in this example the SMR is the natural variable to visualise. *R* has a number of different packages for drawing maps, including *sp* and *ggplot2*, and we illustrate the *ggplot2* package here. First load the required packages using the code:

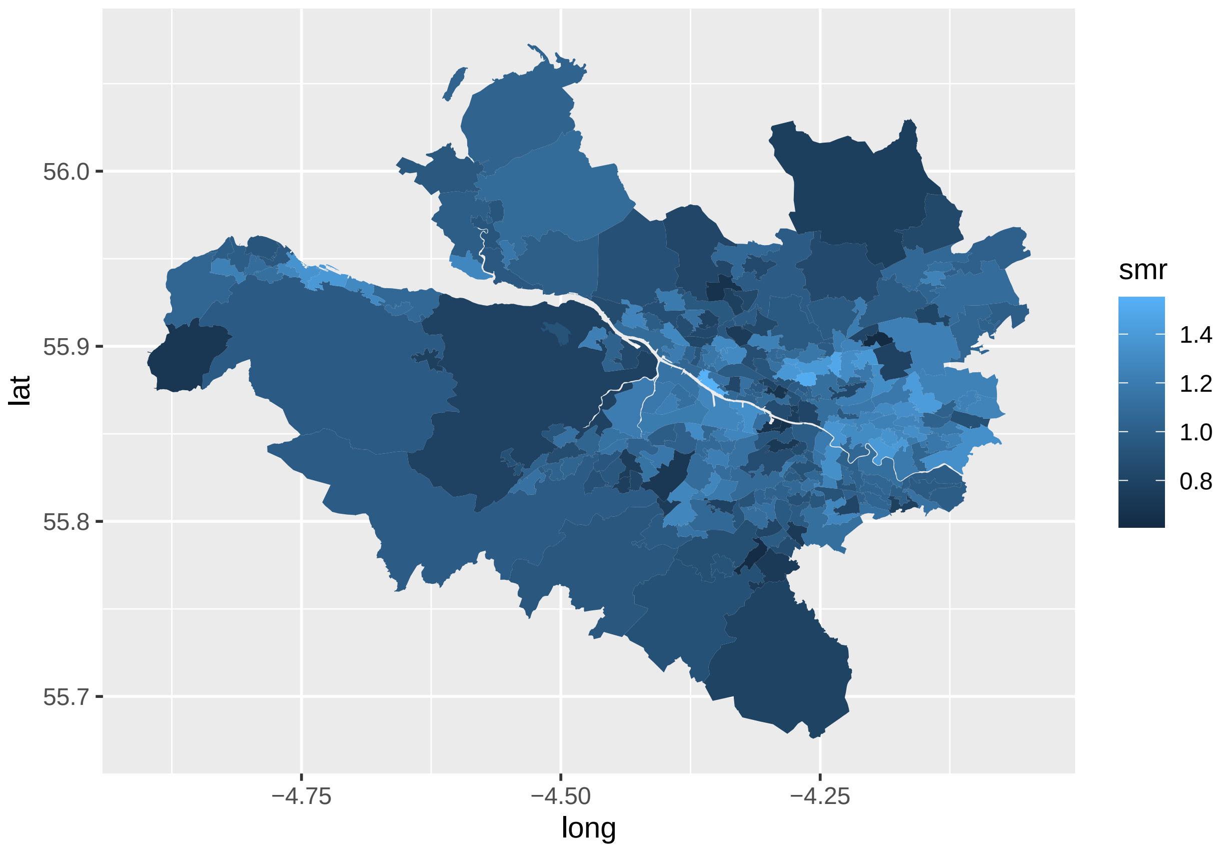
```
library(ggplot2)
library(rgeos)
library(maptools)
```

Before you can draw a map, *ggplot2* requires you to turn the *sp.dat SpatialPolygonsDataFrame* object into a *data.frame*. This can be done using the following code:

```
sp.dat@data$id <- rownames(sp.dat@data)
temp1 <- fortify(sp.dat, region = "id")
sp.dat2 <- merge(temp1, sp.dat@data, by = "id")
```

Then a basic map of the SIR can be created using the following code:

```
ggplot(data = sp.dat2, aes(x=long, y=lat, goup=group, fill = smr)) +
    geom_polygon()
```

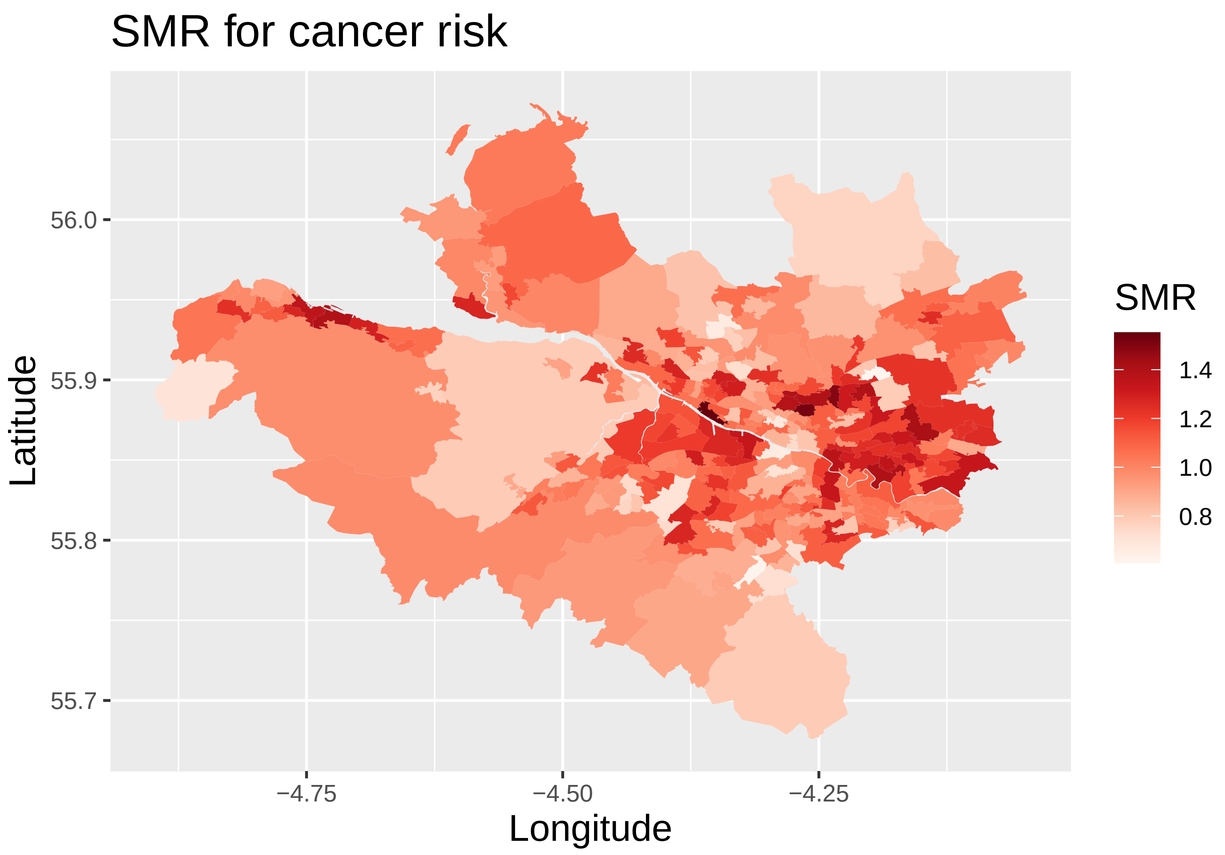


Here:

- *ggplot()* - specifies the data frame, the two variables (columns) to be used to create the plotting area, and the variable to be mapped.
- *geom_poylgon()* - adds the shaded areal units to the plot.

However, this map is unsatisfactory in a number of ways, and can be improved by adding additional commands to the *ggplot()* function separated by the + sign as shown below.

```
library(RColorBrewer)
ggplot(data = sp.dat2, aes(x=long, y=lat, goup=group, fill = smr)) +
    geom_polygon() +
    xlab("Longitude") +
    ylab("Latitude") +
    labs(title = "SMR for cancer risk", fill = "SMR") +
    theme(title = element_text(size=14)) +
    scale_fill_gradientn(colors=brewer.pal(n=9, name="Reds"))
```

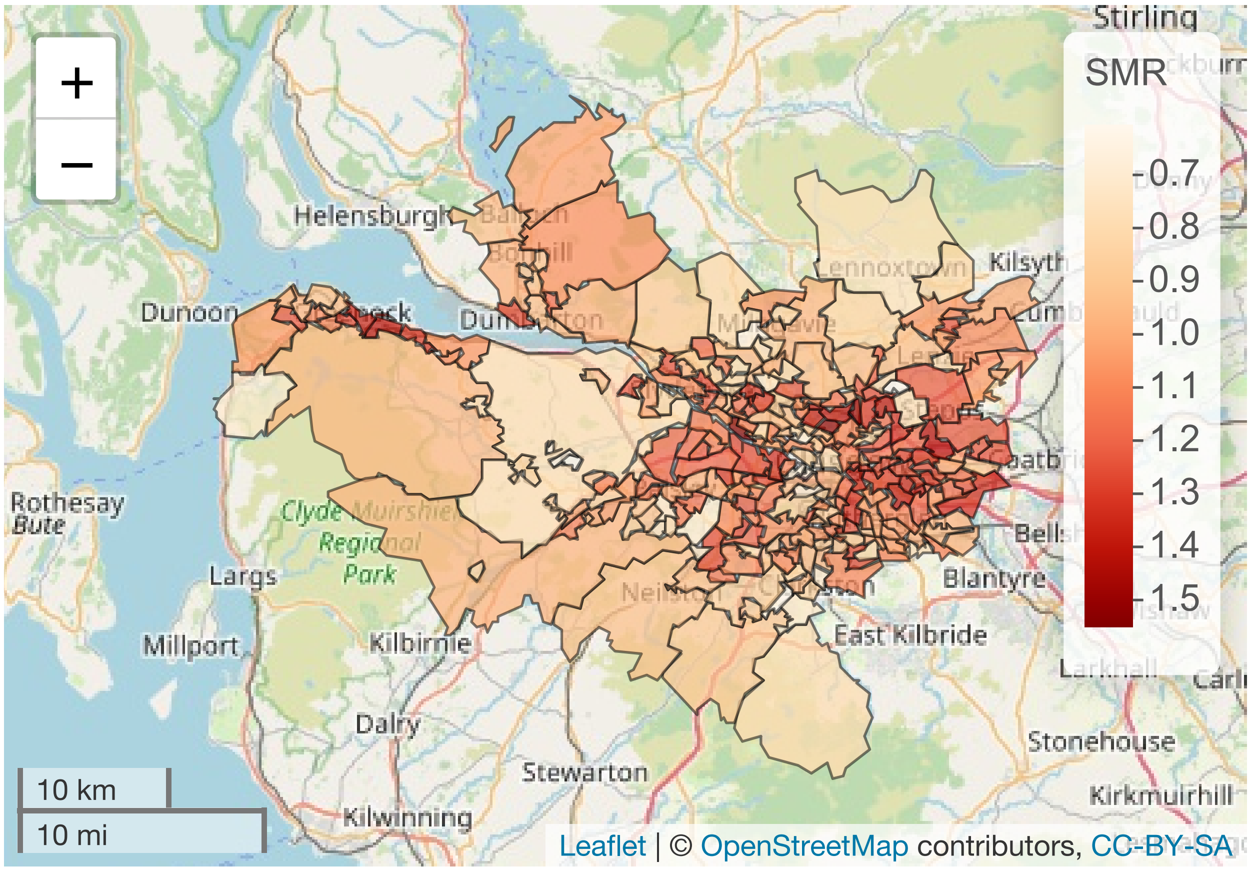


Here the new lines make the following changes:

- *xlab()* - specifies the horizontal axis label for the plot.
- *ylab()* - specifies the vertical axis label for the plot.
- *labs()* - adds titles to the plot and to the colour key.
- *theme()* - changes the typeface and size of the text on the plot.
- *scale_fill_gradientn()* - change the colour scale. The colour palette comes from the *RColorBrewer* library at http://colorbrewer2.org/.

Finally, you can do an interactive data visualisation using the *leaflet* package as shown below, which overlays your data on an OpenStreetMap.

```
library(leaflet)
colours <- colorNumeric(palette = "OrRd", domain = sp.dat@data$smr, reverse=FALSE)
leaflet(data=sp.dat) %>%
    addTiles() %>%
    addPolygons(fillColor = ~colours(smr),
                color="black", weight=1,
                fillOpacity = 0.7) %>%
    addLegend(pal = colours, values = sp.dat@data$smr,
              opacity = 1, title="SMR") %>%
    addScaleBar(position="bottomleft")
```



Note, that this data visualistaion tool works with the original *SpatialPolygonsDataFrame* object *sp.dat*.

4. Initial non-spatial modelling

When fitting a regression model it is spatial autocorrelation in the residuals after adjusting for any covariates that should be checked, so the next step is to fit a simple covariate model and check the residuals for spatial autocorrelation. Given the response data are counts, the following Poisson log-linear model is appropriate. Here the expected numbers of cases (E_i) is included as an offset term on the log-scale (as the linear predictor is on the log scale).

$$Y_i \sim \text{Poisson}(\mu_i)$$
$$\ln(\mu_i) = \ln(E_i) + \beta_1 + \beta_2 pm10_i + \beta_3 smoke_i + \beta_4 ethnic_i.$$

This model is fitted and the results visualised using the following code:

```
form <- Y~offset(log(E))+pm10+smoke+ethnic
model1 <- glm(formula=form, family=poisson, data=dat)
summary(model1)


Call:
glm(formula = form, family = poisson, data = dat)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.1688  -0.9560  -0.0885   0.8845   4.0877
```

```
Coefficients:
              Estimate Std. Error z value Pr(>|z|)    
(Intercept) -0.5470147  0.0565970  -9.665  < 2e-16 ***
pm10         0.0227687  0.0035544   6.406  1.5e-10 ***
smoke        0.0082125  0.0006203  13.239  < 2e-16 ***
ethnic      -0.0049302  0.0005853  -8.423  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 972.94  on 270  degrees of freedom
Residual deviance: 616.93  on 267  degrees of freedom
AIC: 2401.9

Number of Fisher Scoring iterations: 4
```

5. Assess the presence of spatial autocorrelation in the residuals

Moran's I statistic for measuring the amount of spatial autocorrelation can be calculated using functionality from the *spdep* package. Specifically, the statistic is computed and a hypothesis test (where the null hypothesis H_0 is independence) conducted using the *moran.mc()* function. However, first the spatial neighbourhood information is required in the form of a list type object, which is created from the *sp.dat* object using the following code:

```
library(spdep)
W.nb <- poly2nb(sp.dat, row.names = rownames(sp.dat@data))
W.list <- nb2listw(W.nb, style = "B")
```

Then the presence of spatial autocorrelation can be checked using the following code:

```
moran.mc(x = residuals(model1), listw = W.list, nsim = 10000)


    Monte-Carlo simulation of Moran I

data:  residuals(model1) 
weights: W.list  
number of simulations + 1: 10001 

statistic = 0.084654, observed rank = 9884, p-value = 0.0117
alternative hypothesis: greater
```

From which it can be seen that correlation is present and hence needs to be modelled.

6. Fitting a spatial autocorrelation model to the data

We allow for the spatial autocorrelation by fitting the extended model:

$$\begin{aligned} Y_i &\sim \text{Poisson}(\mu_i) \\ \ln(\mu_i) &= \ln(E_i) + \beta_1 + \beta_2 pm10_i + \beta_3 smoke_i + \beta_4 ethnic_i + \phi_i. \end{aligned}$$

The random effects ϕ_i are modelled by the CAR model proposed by Leroux, Lei, and Breslow (2000), which is given by

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

where here ρ is a spatial dependence parameter with

- $\rho = 0$ corresponding to independence, that is $\phi_i|\boldsymbol{\phi}_{-i} \sim \mathrm{N}(0,\tau^2)$ and
- $\rho = 1$ corresponding to strong spatial correlation, that is $\phi_i|\boldsymbol{\phi}_{-i} \sim \mathrm{N}\left(\frac{\sum_{j=1}^{n} w_{ij}\phi_j}{\sum_{j=1}^{n} w_{ij}}, \frac{\tau^2}{\sum_{j=1}^{n} w_{ij}}\right)$.

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.
- *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.
- *thin* - how many to thin the MCMC samples by to reduce their autocorrelation.

The only one of these we need to construct is `W`, the adjacency or neighbourhood matrix, which can be constructed from the spatial object using the following code:

```
W <- nb2mat(W.nb, style = "B")
```

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.

```
library(CARBayes)
model2 <- S.CARleroux(formula=form, family="poisson", data=sp.dat@data,
W=W, burnin=20000, n.sample=120000, thin=10, verbose=FALSE)
print(model2)


#################
#### Model fitted
#################
Likelihood model - Poisson (log link function)
Random effects model - Leroux CAR
Regression equation - Y ~ offset(log(E)) + pm10 + smoke + ethnic
Number of missing observations - 0

############
#### Results
############
Posterior quantities and DIC

             Median    2.5%   97.5% n.effective Geweke.diag
(Intercept) -0.5529 -0.7897 -0.3004       856.8         0.9
pm10         0.0236  0.0081  0.0386       806.8        -0.7
smoke        0.0074  0.0053  0.0095      1390.2        -0.5
ethnic      -0.0047 -0.0066 -0.0028      1295.5        -0.7
tau2         0.0213  0.0130  0.0331      2521.1        -0.5
rho          0.3692  0.1001  0.7304      1754.0         0.2

DIC =  2209.79       p.d =  141.8381       LMPL =  -1137.02
```

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*). The summary table of the key model parameters (all parameters except the random effects $\boldsymbol{\phi}$) contains the following information:

- *Median* - point estimate for the parameter, which is the posterior median of the samples generated.
- *(2.5%, 97.5%)* - 95% posterior credible interval for the parameter.
- *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 *model2* is an *R list* object, which contains the following elements as shown via the *summary* function.

```
summary(model2)

                    Length Class      Mode
summary.results       42   -none-     numeric
samples                6   -none-     list
fitted.values        271   -none-     numeric
residuals              2   data.frame list
modelfit               6   -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_i values) from the model.
- *residuals* - a matrix with 2 different types of residuals, response and Pearson.
- *modelfit* - a vector containing model fit criteria including the DIC.

7. Checking convergence of the MCMC simulation

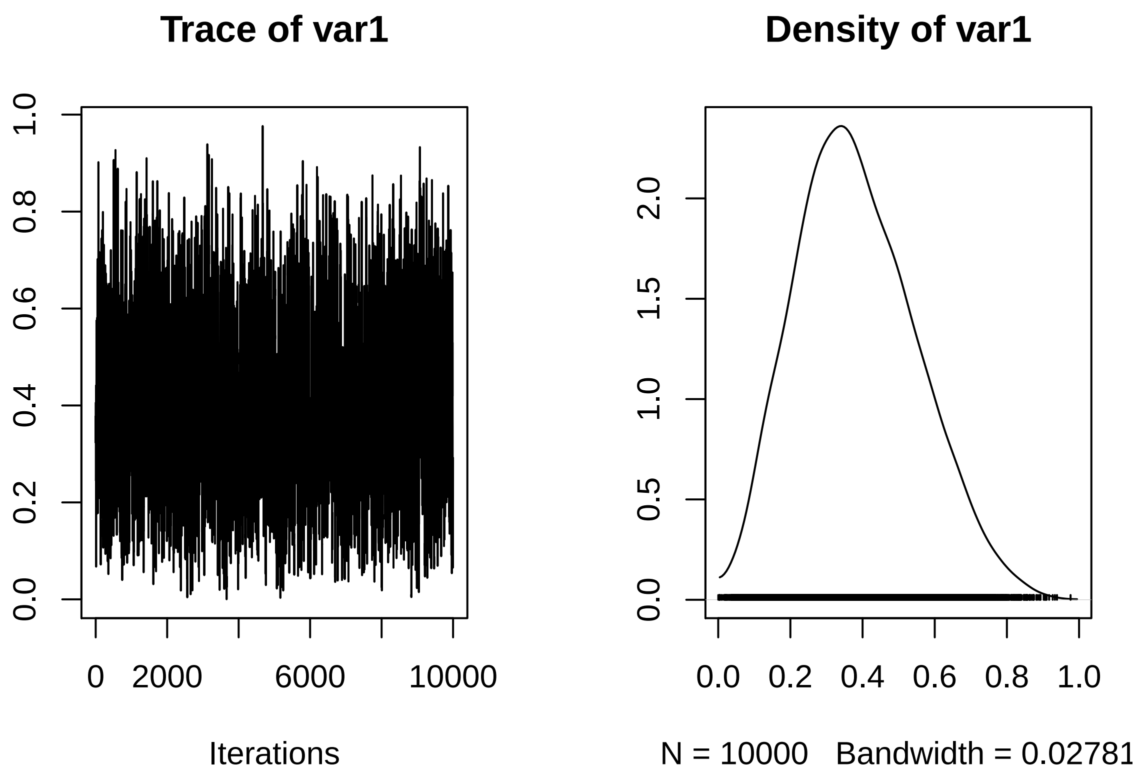
The convergence of the MCMC samples can be assessed by viewing traceplots of the samples for certain parameters. The set of samples for a given parameter have converged if they show no trend and random scatter above and below the average value. The samples are stored in the *samples* element of the *R list* object *model2*, which for this model has elements

```
summary(model2$samples)
```

```
       Length  Class Mode
beta     40000 mcmc  numeric
phi    2710000 mcmc  numeric
tau2     10000 mcmc  numeric
rho      10000 mcmc  numeric
fitted 2710000 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(model2$samples$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 0.4, suggesting the spatial dependence in these data after adjusting for the covariates is moderate.

8. Inference from the model

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(model2$summary.results[2:4 , 1:3])
```

```
          Median      2.5%     97.5%
pm10   1.023881 1.0081329 1.0393547
smoke  1.007427 1.0053141 1.0095453
ethnic 0.995311 0.9934217 0.9972039
```

So for example, a 1 unit increase in the PM_{10} concentrations is associated with a 2.4% increase in disease risk. But why does PM_{10} have a larger increase compared to smoking, which is counter-intuitive?

The reason is that the variation in these variables is different, for example:

```
summary(dat$pm10)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  13.10   15.97   17.30   17.14   18.43   20.50
```

```
summary(dat$smoke)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.50   22.55   29.60   29.67   36.40   52.40
```

So a 1 unit increase in PM_{10} covers more of the variation in that variable than a 1 unit increase in smoking. Computing the relative risks for a standard deviation increase in each one gives:

```
exp(sd(dat$pm10) * model2$summary.results[2 , 1:3])
```

```
  Median     2.5%    97.5%
1.043179 1.014615 1.071587
```

```
exp(sd(dat$smoke) * model2$summary.results[3 , 1:3])
```

```
  Median     2.5%    97.5%
1.073919 1.052403 1.095874
```

So now smoking has the larger effect 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.

Lee, D. 2013. “CARBayes: An R Package for Bayesian Spatial Modelling with Conditional Autoregressive Priors.” *Journal of Statistical Software* 55: 13.

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.