Project

Spatial Statistics

Contents

List of Figures

Appendix

List of Tables

Appendix

Chapter 1

Exploratory Data Analysis using SIR

1.1 Introduction

The main purpose of the project is to identify the effect of air pollution and socio-economic variables on the risk of admission in the hospitals due to respiratory disease across electoral wards in Greater London. The data provided has information of 7 years from 2003 to 2009 describing hospital admissions across 624 electoral wards in Greater London. We will focus on the data for the year 2007 for model building and analysing the effect of variables on the hospital admissions.

1.2 SIR

Beginning with the exploratory analysis, we start by getting the subset of the data for the year 2007. To do so we first load the dataset from the file london.RData which loads as an sp object, we then convert it into an sf object to further take subset with respect to the year and do our analysis.

Since we are dealing with count of people at risk, we will use Standardised Incidence Ratio (SIR) which basically compares the number of annual admissions to what we expect given the national average rate (SIR=Observed/Expected). The Observed and the Expected values are available in the dataset so we directly take the ratio and add the values to new column “SIR”. We then further explore SIR values for each region using ggplot (see Fig 1.1) which would help us detect areas with elevated risk.

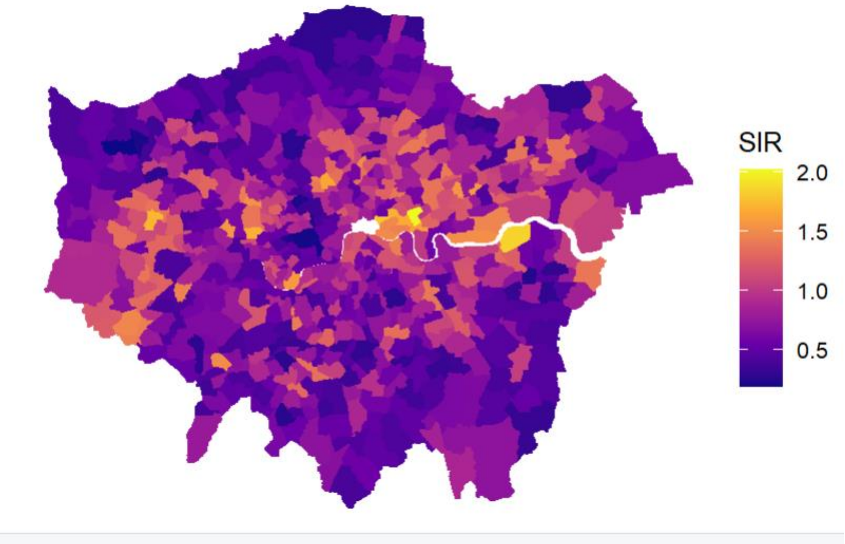


Figure 1.1: SIR values for Greater London Regions

The respiratory admissions seem to be higher in the few parts from east to west of greater London and areas near Greenwich and Tower hamlets regions seem be have elevated risks in the central wards and becomes less prominent as we move towards north and south wards. We have also checked if the data is skewed, see Appendix 1.1.

The SIR values distribution seems to be pretty decent (see Appendix 1.1) so no need to take transformations for the values. Still to have a better understanding we plotted the distribution of logged SIR values (see Appendix 1.2), it didn’t point out anything new compared to the original plot so we will go ahead with original values. We can further explore the effect of prominent variables like PM25 (Annual average concentration of fine particulate matter PM2.5 (µgl−1)), JSA (Proportion of electoral ward in receipt of jobseeker's allowance, an unemployment benefit) and Price (Annual average sale price of homes (logged GBP)) on the SIR values using scatter plots.

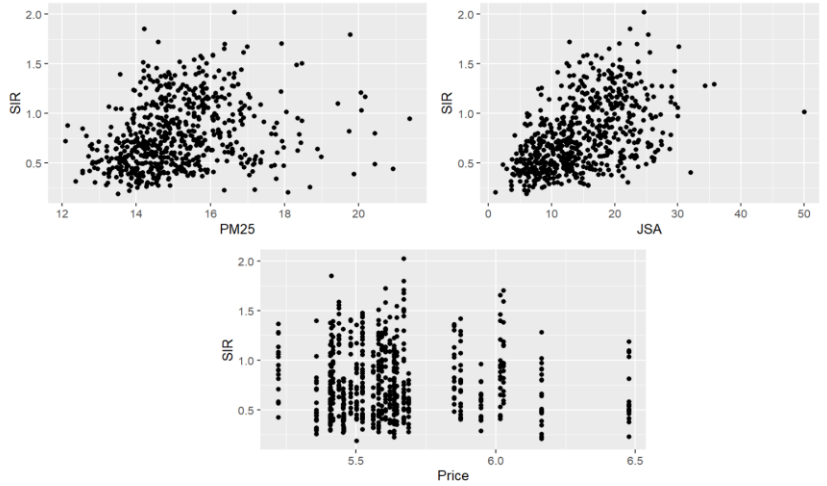


Figure 1.2: Scatter plots of PM25, JSA and Price with SIR

As you can see in Fig 1.2, PM25 and JSA has a clear linear correlation with the SIR values having only a few outliers that is it the variables have a clear impact on annual hospital admissions for respiratory diseases. The price of houses seems to have almost a linear line then going down still we will include this variable in our model and see how it impacts.

Chapter 2

Models

2.1 Poisson Regression

We will be using Poisson regression to build a linear regression model for SIR values of respiratory admission in greater London using offset of log. I'm going to use Nimble to fit the model, using JSA, Price, PM25 as covariances to set the prior distribution on as they almost have a linear relationship with SIR. I will not be thinning the chain at all and running only 2 chains to check Gelman-Rubin diagnostic (Rhat) later for convergence. Also setting up initial values for the model since we are taking log transformations.

Once the model is run, we will check for convergence using trace plots initially.

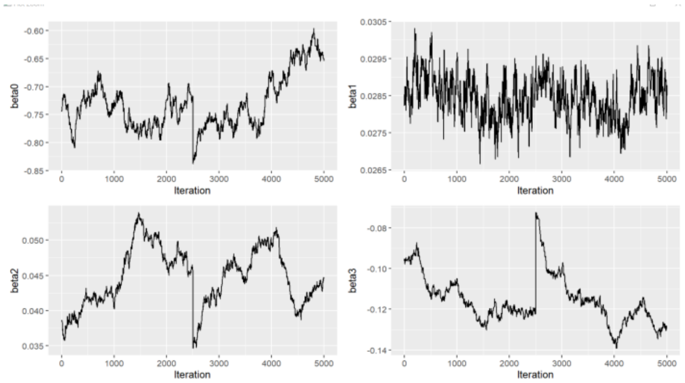


Figure 2.1: Trace plots for Poisson Regression

As seen in Fig. 2.1, the chains have not converged telling us the mean of the sample is not constant, also there is some correlation in chains particularly beta 2 and beta 3, I will try to mean centre the covariates and try again.

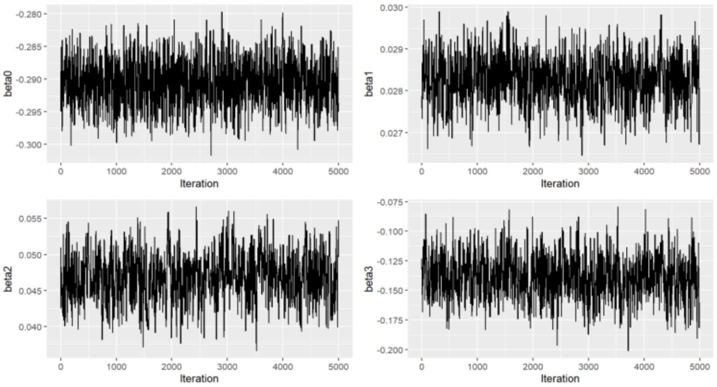


Figure 2.2: Trace plots for Poisson Regression with mean covariates

The convergence looks much better (see Fig. 2.2), we will now try with convergence diagnostics to further check the convergence.

For Gelman-Rubin diagnostic (Rhat) diagnostics all the values in Rhat is less than 1.2 (see Appendix 2.1), we can fairly assume convergence is fine still I will use geweke plots to investigate further.

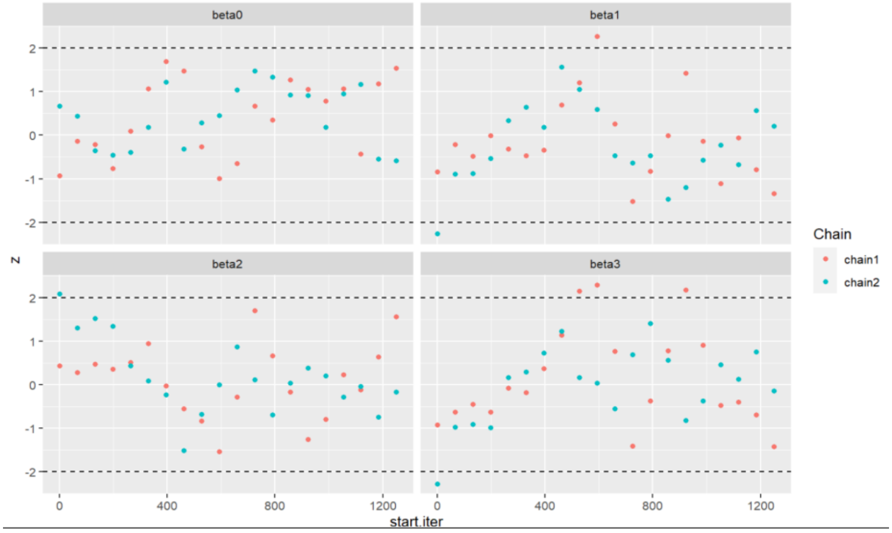


Figure 2.3: Geweke plots for 2 chains

There are only very few points outside the convergence index (see Fig. 2.3) so I think this is acceptable, nothing too concerning, I'll continue with this model fit.

Before I think about making inference from this model, I want to check the assumptions. I'll extract the fitted values, and calculate mu, then the Pearson residuals, calculating residual variance.

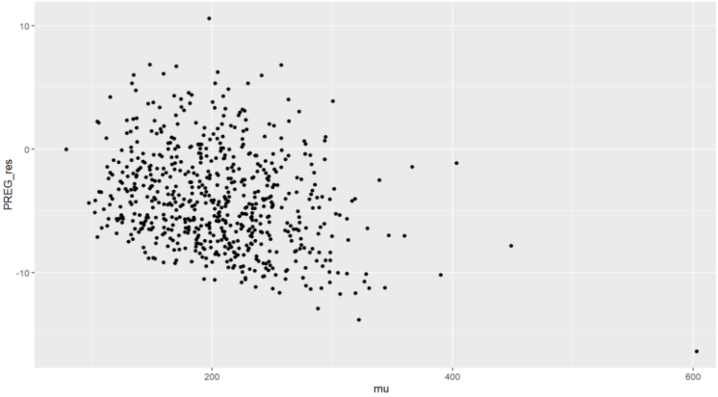


Figure 2.4: Pearson Residuals against fitted values

There is clear evidence of residual overdispersion (see Fig. 2.4). The Pearson residual variance is also very large at around 15 and there is quite strong fanning out in the plot of the Pearson residuals versus fitted values. We may be underfitting the model or missing out some important information. Further using Morain's I test to check spatial dependence with two-sided hypothesis.

Moran I statistic standard deviate = 8.9126, p-value < 2.2e-16

Table 2.1: Moran's I values for Poisson Regression

Moran I statistic	Expectation	Variance
0.2147978916	-0.0016051364	0.0005895414

The results of this test show that there is significant spatial dependence. The Observed Moran's I is greater than 0 at around 0.21 (see Table 2.1), the p-value is also very small at 2.2e-16 to reflect that, we therefore reject the null hypothesis that there is no spatial dependence.

2.2 Poisson CAR

We tried Poisson regression and assumptions are not met. Since there is high spatial dependence, we will add spatial random effect to account for the spatial dependence using CAR model. Setting up all the initial model parameters, running nimble model for 10000 iterations and chopping off 50% of the chain. The trace plots seen in Appendix Fig. 1.3 states that it is definitely not converged, even running the Gelman-Rubin diagnostic (Rhat) diagnostics gives values much greater than 1.2 (see Appendix 2.2- not adding mu values to table) further stating issues with convergence. We can try and thin the chain to make the convergence better.

Looking at effective sample sizes to decide the number of iterations to run (see Appendix 2.3) we can see that the smallest effective sample size is around 300, so we thin the chains by 5000/350, taking every 17th sample. That means that I'll need to run my sampler for 170000 iterations, with a burn in of 85000. Checking convergence using Gelman-Rubin diagnostic (Rhat) values.

Table 2.2: Gelman-Rubin diagnostic (Rhat) values for CAR model for 170000 iterations

	Point est.	Upper C.I.
beta0	1.00	1.00
beta1	1.02	1.06
beta2	1.06	1.20
beta3	1.04	1.10
Multivariate psrf	1.12	

Even after thinning the chains and running longer iterations there's a value of 1.2(see Table 2.2- not adding mu values to table) so we can state that convergence hasn't been fully met. Even the trace plots (see Appendix Fig 1.4) suggest convergence hasn't been met properly. Further, evaluating geweke plot.

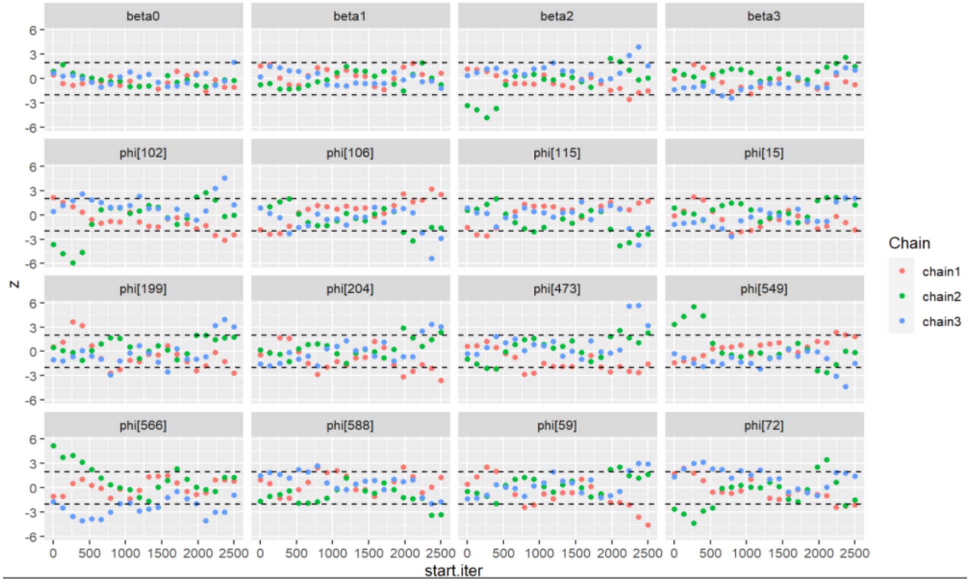


Figure 2.5: Geweke plots for CAR model

The Geweke plots(see Fig 2.5) also shows there are several points that lie outside the convergence in the beginning of the chain further suggesting convergence hasn't been fully met, although let's go ahead with this model to calculate Pearson residuals and check the model assumptions.

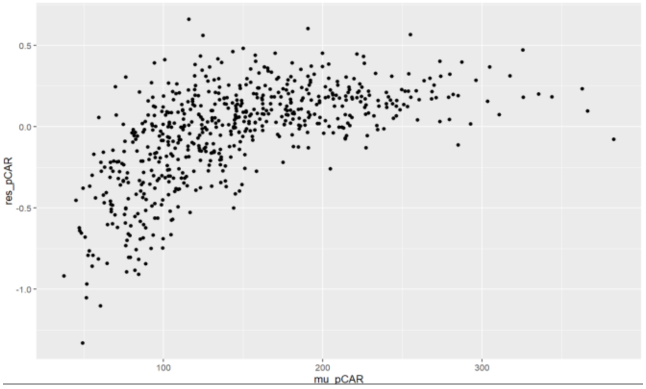


Figure 2.6: Pearson Residuals against fitted values for CAR

There is clear evidence of residual under dispersion (see Fig. 2.6). The Pearson residual variance is very small (0.09095632) and there is clear fanning in in the plot of residuals versus fitted values, so it states under dispersion. The Morain's I is also negative (see Table 2.3) which suggest relatively low spatial dependence though the p value is also small so we cannot fully save the independence assumption has met.

Moran I statistic standard deviate = -8.723, p-value < 2.2e-16

Table 2.3: Moran's I values for CAR

Moran I statistic	Expectation	Variance
-0.2132569570	-0.0016051364	0.0005887249

Chapter 3
Comparison and Interpretation

Calculating the DIC and effective number of parameters to choose the better model.

Table 3.1: Model comparisons

Model	Pearson Variance	Moran's I	P value	DIC	pD
Poisson Regression	15.59019	0.2147978916	2.2e-16	16822.32	3.925438
Poisson CAR	0.09095632	-0.2143345634	2.2e-16	5420.205	582.3789

As seen in Table 3.1, the DIC for the Poisson CAR model is very less compared to the Poisson regression model, despite pD suggesting that CAR model is more complex of the two models. Both the models don't really perform well in terms of the Pearson variance, the Poisson regression displays clear over dispersion, and the CAR model has under dispersion, which tells us maybe the Poisson distribution is does not suit well for the data that we have. Further adding more covariates would may also help with this. Moran's I for CAR indicates that it has reduced the spatial dependence though there is still under dispersion in the data which suggests the model is not performing as well. Comparing both the model we can say CAR model is better that Poisson regression model although both don't perform particularly well with the data.

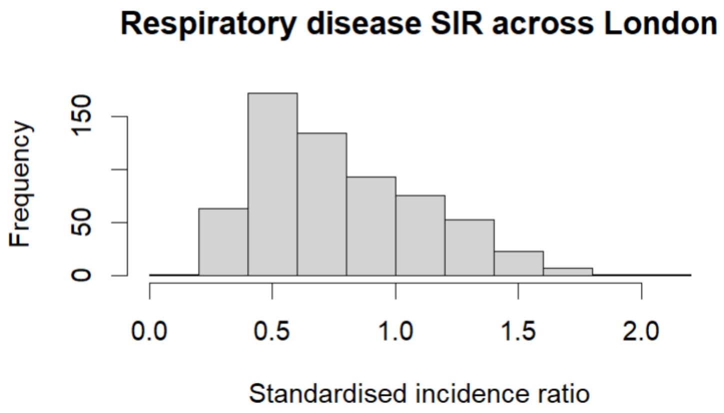
looking at the coefficient estimates for both the models:

Table 3.2: Coefficient Estimates comparisons

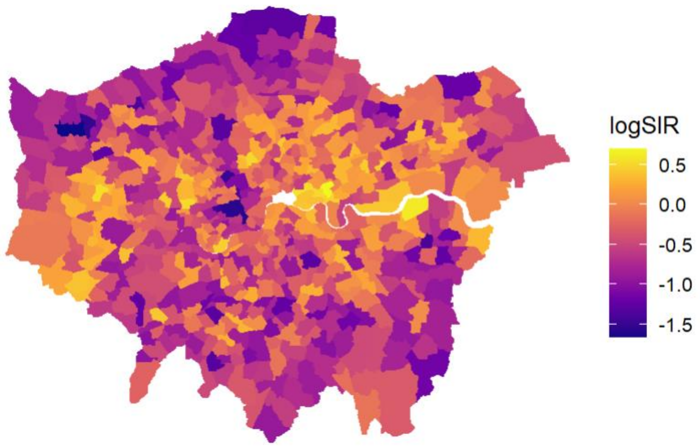
Model	Mean	95%CI_low	95%CI_upp
Poisson Regression	1.0481076	1.0418225	1.0548120
Poisson CAR	1.0026318	0.9318613	1.0779069

Considering CAR model as a better model we can state that, for each additional percentage increase of concertation of PM25 is there will be an estimate increase of 0.2% for hospital admissions for respiratory disease with actual increase can be anywhere between -6.82 to 7.7 with 95% probability.

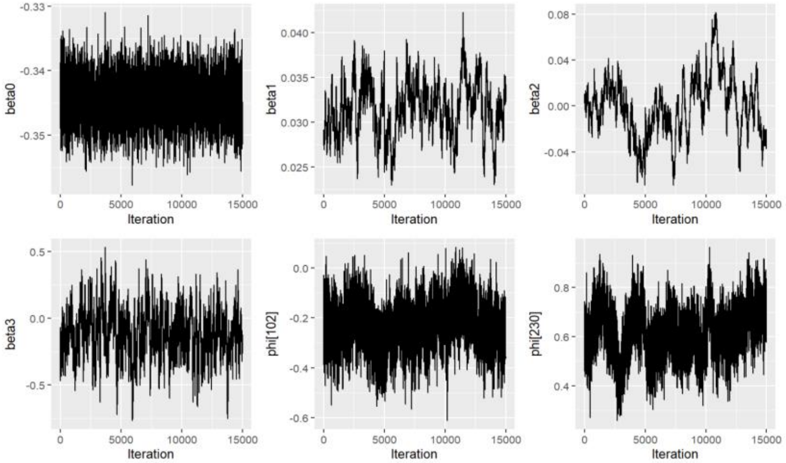
Appendix 1



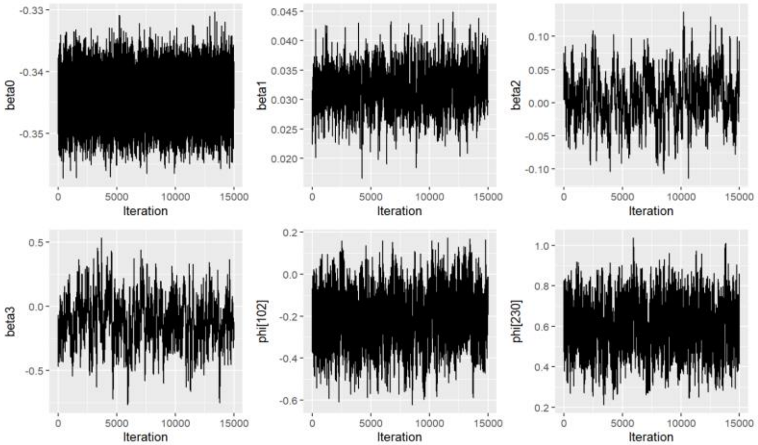
Appendix 1.1: Histogram for SIR values



Appendix 1.2 Log of SIR values for Greater London Regions



Appendix 1.3: Trace plots for CAR model



Appendix 1.3: Trace plots for CAR model for 170000 iterations

Appendix 2.1: Gelman-Rubin diagnostic (Rhat) values

	Point est.	Upper C.I.
beta0	1.00	1.02
beta1	1.00	1.01
beta2	1.01	1.04
beta3	1.01	1.02
Multivariate psrf	1.01	

Appendix 2.2: Gelman-Rubin diagnostic (Rhat) values for CAR model

	Point est.	Upper C.I.
beta0	1.00	1.02
beta1	1.05	1.11
beta2	1.15	1.44
beta3	1.62	2.94
Multivariate psrf	1.69	

Appendix 2.3: Effective Sample size(not including all the values due to page limit)

beta0	beta1	beta2	beta3	mu[1]	mu[2]	mu[3]	mu[4]
2836.5210	393.3222	371.8893	372.9368	3166.3151	3366.8136	3145.4046	3313.9742

Appendix 2

```
#load packages
library(sp)
library(ggplot2)
library(patchwork)
require(sf)
library(rgeos)
library(rgdal)
library(spdep)
library(nimble)
library(coda)
library(sf)
library(R2WinBUGS)
library(dplyr)
library(tidyverse)

#loading london data
load("london.RData")
#the data loads as london_3, checking the data first
head(london_3@data)

#creating a subset with the year 2007
london_3@data <- (subset(london_3@data, year%in%2007))
#converting to sf
london_2007_sf <- st_as_sf(london_3)
head(london_2007_sf)
#calculating SIR
london_2007_sf$SIR <- london_2007_sf$Observed/london_2007_sf$Expected
#using ggplot
ggplot(london_2007_sf) +
  geom_sf(aes(fill=SIR, colour=SIR)) +
  scale_fill_viridis_c(option="C") +
  scale_colour_viridis_c(option="C") +
  theme_void()
## Check distribution of SIR
hist(london_2007_sf$SIR, main = "Respiratory disease SIR across London",
xlab = "Standardised incidence ratio")
#trying log of SIR
london_2007_sf$logSIR <- log(london_2007_sf$SIR)

ggplot(london_2007_sf) +
  geom_sf(aes(fill=logSIR, colour=logSIR)) +
  scale_fill_viridis_c(option="C") +
  scale_colour_viridis_c(option="C")+
  theme_void()
```

```
## Plot SIR against covariates, scatter plots:

ggplot(london_2007_sf) +
  geom_point(aes(PM25, SIR)) ->
  p1
ggplot(london_2007_sf) +
  geom_point(aes(JSA, SIR)) ->
  p2
ggplot(london_2007_sf) +
  geom_point(aes(Price, SIR)) ->
  p3

layout <-"
AABB
#CC#
"

p1+p2+p3 + plot_layout(design=layout)
```

##Question 2

```
## Fit the model # nimble

model_code_poisson <- nimbleCode({
  for(i in 1:N){
    Y[i] ~dpois(mu[i])
    log(mu[i]) <- log(E[i]) + beta0 + beta1*JSA[i]+ beta2*PM25[i]+
beta3*Price[i]
  }

  beta0~dnorm(0,0.01)
  beta1~dnorm(0,0.01)
  beta2~dnorm(0,0.01)
  beta3~dnorm(0,0.01)
})

Data_poisson <- list(Y = london_2007_sf$Observed,
             E = london_2007_sf$Expected,
             JSA = london_2007_sf$JSA,
             PM25 = london_2007_sf$PM25,
             Price = london_2007_sf$Price)
Data_poisson <- list(Y = london_2007_sf$Observed,
             E = london_2007_sf$Expected,
             JSA = london_2007_sf$JSA - mean(london_2007_sf$JSA),
             PM25 = london_2007_sf$PM25 - mean(london_2007_sf$PM25),
             Price = london_2007_sf$Price - mean(london_2007_sf$Price))
Constants_poisson <- list(N = nrow(london_2007_sf))
```

```
Inits_poisson <- list(list(beta0 = 0, beta1 = 0, beta2 = 0, beta3 = 0),
              list(beta0 = 0, beta1 = 0, beta2 = 0, beta3 = 0))

preg_mod <- nimbleMCMC(data = Data_poisson,
                       constants=Constants_poisson,
                       code=model_code_poisson,
                       monitors = c("beta0", "beta1", "beta2", "beta3"),
                       inits = Inits_poisson,
                       niter = 5000,
                       nburnin = 2500,
                       thin = 1,
                       nchains = 2,
                       summary=TRUE,
                       samplesAsCodaMCMC=TRUE)

#with mu to calculate DIC
preg_mod_mean <- nimbleMCMC(data = Data_poisson,
                       constants=Constants_poisson,
                       code=model_code_poisson,
                       monitors = c("beta0", "beta1", "beta2",
"beta3","mu"),
                       inits = Inits_poisson,
                       niter = 5000,
                       nburnin = 2500,
                       thin = 1,
                       nchains = 2,
                       summary=TRUE,
                       samplesAsCodaMCMC=TRUE)

sims <- as.data.frame(Reduce("rbind", preg_mod$samples))
sims_mean <- as.data.frame(Reduce("rbind", preg_mod_mean$samples))


#sims <- as.data.frame(preg_mod$samples)
ggplot(sims)+
  geom_line(aes(1:nrow(sims), beta0))+
  labs(x="Iteration")->
  p1

ggplot(sims)+
  geom_line(aes(1:nrow(sims), beta1))+
  labs(x="Iteration") ->
  p2

ggplot(sims)+
  geom_line(aes(1:nrow(sims), beta2))+
  labs(x="Iteration") ->
  p3

ggplot(sims)+
  geom_line(aes(1:nrow(sims), beta3))+
  labs(x="Iteration") ->
  p4

p1+p2+p3+p4


# Gelman-Rubin diagnostic (Rhat)
gelman.diag(preg_mod$samples)
# use coda package to get geweke values #Checking convergence
for_geweke <- coda::geweke.plot(preg_mod$samples)

# This is a data frame with a row for each point on the geweke plot and a
column for each parameter
for_geweke <- as.data.frame(for_geweke)

for_geweke %>%
  # Transform to each row being for a particular point and a particular
parameter (so we can use  facet_wrap)
  pivot_longer(-start.iter, names_to=c("Variable","Chain"), values_to="z",
names_prefix="z\\.", names_sep="\\.(?=c)")%>%
  # Make the variable names nicer (Some beautiful regex there!)
  mutate(Variable = str_replace(Variable, "\\.1$",""),
         Variable=str_replace(Variable, "\\.", "["),
         Variable=str_replace(Variable, "\\.", "]")) %>%
  ggplot() +
  # Plot the points
  geom_point(aes(start.iter, z, color=Chain)) +
  # Add dashed lines at +/-2
  geom_hline(aes(yintercept=-2),linetype="dashed") +
  geom_hline(aes(yintercept=2),linetype="dashed") +
  # Separate plots for each parameter
  facet_wrap(~Variable)

#check assumption-  Pearson residuals
# extract the beta samples
# grepl() looks for the presence of "beta" in the data frame names
beta_samples <- apply(sims[,grepl("beta", names(sims))], 2, median)

#log_rate <- mean(sims$beta0) +
#  mean(sims$beta1) * london_2007_sf$JSA+
#  mean(sims$beta2) * london_2007_sf$PM25+
#  mean(sims$beta3) * london_2007_sf$Price
# three covariates in this model
lp <- beta_samples[1] + beta_samples[2]*london_2007_sf$JSA +
  beta_samples[3]*london_2007_sf$PM25 +
beta_samples[4]*london_2007_sf$Price

mu <- london_2007_sf$Expected*exp(lp)

PREG_res <- (london_2007_sf$Observed - mu)/sqrt(mu)
#I want to check the mean=variance assumption first:

var(PREG_res)
ggplot() +
  geom_point(aes(mu, PREG_res))

#independence assumption
moran.test(PREG_res, nb2listw(poly2nb(london_3)), alternative="two.sided")


###Question 3

# Produce the adjacency matrix, and various pieces of associated
W <- nb2mat(poly2nb(london_3), style = "B")
inds <- lapply(1:nrow(W), function(i) which(W[i, ] == 1))
Adj <- Reduce("c", inds)
Num.Adj <- rowSums(W)
SumNumNeigh <- sum(Num.Adj)

# combine all of the data, and constants into a single list
pCAR_code <- nimbleCode({
  for(i in 1:N){
    Y[i] ~ dpois(mu[i])
    log(mu[i]) <- log(E[i]) + beta0 + beta1*JSA[i]+ beta2*PM25[i]+
beta3*Price[i] + phi[i]
  }

  phi[1:N] ~ dcar_normal(Adj[1:L], weights[1:L], Num[1:N], tau,
zero_mean=1)

  beta0~dnorm(0,0.01)
  beta1~dnorm(0,0.01)
  beta2~dnorm(0,0.01)
  beta3~dnorm(0,0.01)
  tau ~ dgamma(0.01,0.01)
})

Data_car <- list(Y = london_2007_sf$Observed,
             E = london_2007_sf$Expected,
             JSA = london_2007_sf$JSA - mean(london_2007_sf$JSA),
             PM25 = london_2007_sf$PM25 - mean(london_2007_sf$PM25),
             Price = london_2007_sf$Price - mean(london_2007_sf$Price))
```

```
Constants_car <- list(N = nrow(london_2007_sf),
                  Adj = Adj,
                  Num = Num.Adj,
                  L = SumNumNeigh,
                  weights=rep(1, SumNumNeigh))
# Initial Values
Inits_car <- list(list(beta0=0, beta1=0, beta2=0, beta3=0, tau=1,
phi=rep(1, nrow(london_2007_sf))),
              list(beta0=0, beta1=0, beta2=0, beta3=0, tau=1, phi=rep(1,
nrow(london_2007_sf))),
              list(beta0=0, beta1=0, beta2=0, beta3=0, tau=1, phi=rep(1,
nrow(london_2007_sf))))

pCAR <- nimbleMCMC(data=Data_car,
                   constants = Constants_car,
                   code=pCAR_code,
                   monitors=c(paste0("beta", 0:3), "phi", "mu"),
                   inits=Inits_car,
                   nchains = 3,
                   niter=10000,
                   nburnin=5000,
                   summary=TRUE,
                   samplesAsCodaMCMC = TRUE)
car.sims <- as.data.frame(Reduce("rbind",pCAR$samples))
ggplot(car.sims) +
  geom_line(aes(1:nrow(car.sims), beta0)) +
  labs(x="Iteration") ->
  p1
ggplot(car.sims) +
  geom_line(aes(1:nrow(car.sims), beta1)) +
  labs(x="Iteration") ->
  p2
ggplot(car.sims) +
  geom_line(aes(1:nrow(car.sims), beta2)) +
  labs(x="Iteration") ->
  p3
ggplot(pCar_samples) +
  geom_line(aes(1:nrow(pCar_samples), beta3)) +
  labs(x="Iteration") ->
  p4
ggplot(car.sims) +
  geom_line(aes(1:nrow(car.sims), `phi[102]`)) +
  labs(x="Iteration") ->
  p5
ggplot(car.sims) +
  geom_line(aes(1:nrow(car.sims), `phi[230]`)) +
  labs(x="Iteration") ->
  p6
p1+p2+p3+p4+p5+p6
# Gelman-Rubin diagnostic (Rhat)
for_diag <- lapply(pCAR$samples, function(x) x[,!grepl("mu",
names(x[1,]))])
coda::gelman.diag(for_diag)


Neff <- function(samples, t=20){
  N <- nrow(samples)
  sum_cor <- apply(samples, 2,function(x){
    sum(sapply(1:t, function(y){
      abs(cor(x[-(1:y)], x[-((N-y+1):N)]))
    }))
  })
  return(N/(1+2*sum_cor))
}
Neff(as.data.frame(Reduce("rbind", pCAR$samples)))

# Run the model
pCAR <- nimbleMCMC(data=Data_car,
                   constants = Constants_car,
                   code=pCAR_code,
                   monitors=c(paste0("beta", 0:3), "phi", "mu"),
                   inits=Inits_car,
                   nchains = 3,
                   niter=170000,
                   nburnin=85000,
                   thin=17,
                   summary=TRUE,
                   samplesAsCodaMCMC = TRUE)

coda::gelman.diag(pCAR$samples, multivariate=FALSE)
pCar_samples <- as.data.frame(Reduce("rbind", pCAR$samples))
#lets check the trace plots
ggplot(pCar_samples) +
  geom_line(aes(1:nrow(pCar_samples), beta0)) +
  labs(x="Iteration") ->
  p1
ggplot(pCar_samples) +
  geom_line(aes(1:nrow(pCar_samples), beta1)) +
  labs(x="Iteration") ->
  p2
ggplot(pCar_samples) +
  geom_line(aes(1:nrow(pCar_samples), beta2)) +
  labs(x="Iteration") ->
  p3
ggplot(pCar_samples) +
  geom_line(aes(1:nrow(pCar_samples), beta3)) +
  labs(x="Iteration") ->
  p4
ggplot(pCar_samples) +
  geom_line(aes(1:nrow(pCar_samples), `phi[102]`)) +
  labs(x="Iteration") ->
  p5
ggplot(pCar_samples) +
  geom_line(aes(1:nrow(pCar_samples), `phi[230]`)) +
  labs(x="Iteration") ->
  p6
p1+p2+p3+p4+p5+p6

#let's check geweke plot for convergence
# use coda package to get geweke values #Checking convergence
for_diag <- lapply(pCAR$samples, function(x) x[,!grepl("mu",
names(x[1,]))])
to_plot <- c("beta0","beta1","beta2","beta3", paste0("phi[",
sample(1:nrow(london_2007_sf), 12, replace=FALSE), "]"))

for_geweke <- coda::geweke.plot(for_diag)
# This is a data frame with a row for each point on the geweke plot and a
column for each parameter
for_geweke <- as.data.frame(for_geweke)
for_geweke %>%
  # Transform to each row being for a particular point and a particular
parameter (so we can use  facet_wrap)
  pivot_longer(-start.iter, names_to=c("Variable","Chain"), values_to="z",
names_prefix="z\\.", names_sep="\\.(?=c)")%>%
  # Make the variable names nicer (Some beautiful regex there!)
  mutate(Variable = str_replace(Variable, "\\.1$",""),
         Variable=str_replace(Variable, "\\.", "["),
         Variable=str_replace(Variable, "\\.", "]")) %>%
  filter(Variable%in%to_plot) %>%
  ggplot() +
  # Plot the points
  geom_point(aes(start.iter, z, color=Chain)) +
  # Add dashed lines at +/-2
  geom_hline(aes(yintercept=-2),linetype="dashed") +
  geom_hline(aes(yintercept=2),linetype="dashed") +
  # Separate plots for each parameter
  facet_wrap(~Variable)


#check assumption-  Pearson residuals
#I want to check the mean=variance assumption first:

mu_pCAR <- pCAR$summary$all.chains[grepl("mu",
rownames(pCAR$summary$all.chains)),1]
res_pCAR <- (london_2007_sf$Observed - mu_pCAR)/sqrt(mu_pCAR)
var(res_pCAR)
ggplot() +
  geom_point(aes(mu_pCAR, res_pCAR))

#independence assumption
moran.test(res_pCAR, nb2listw(poly2nb(london_3)), alternative="two.sided")
```

##Question 4

```
DIC <- function(Observed, Fitted, name="mu"){
  Fitted <- Fitted[,grepl(name, names(Fitted))]
  deviance <- sum(log(dpois(Observed, lambda=apply(Fitted,2,median))))

  ave_dev <- mean(sapply(1:nrow(Fitted), function(x)
sum(log(dpois(Observed, lambda=as.numeric(Fitted[x,]))))))

  pD <- 2*(deviance - ave_dev)
  DIC <- -2*deviance + 2*pD
  list(pD = pD, DIC = DIC)
}
preg_DIC <- DIC(london_2007_sf$Observed, sims_mean[grepl("mu",
names(sims_mean))])
pCAR_DIC <- DIC(london_2007_sf$Observed, pCar_samples[grepl("mu",
names(pCar_samples))])

exp(preg_mod$summary$all.chains[grepl("beta",rownames(preg_mod$summary$all.
chains)),c(1,4,5)])
exp(pCAR$summary$all.chains[grepl("beta",rownames(pCAR$summary$all.chains))
,c(1,4,5)])
```