

# 00\_Leukemia\_in\_NY

Frances Lin

3/23/2022

This project fits 6 models (fixed effects, random effects (iid), SLM, ICAR, BYM and Leroux et al.), produces summary results and plots results.

## Load packages

```
# Load packages
library(spdep)      # for spatial weights matrix objects

## Loading required package: sp

## Loading required package: spData

## Loading required package: sf

## Linking to GEOS 3.9.1, GDAL 3.4.0, PROJ 8.1.1; sf_use_s2() is TRUE

library(DCcluster) # for data

## Loading required package: parallel

## Loading required package: spacetime

## Loading required package: DCcluster

## Loading required package: boot

## Loading required package: MASS

library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.1 --

## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.7      v dplyr   1.0.9
## v tidyr   1.2.0      v stringr 1.4.0
## v readr   2.1.2      v forcats 0.5.1
```

```
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag() masks stats::lag()
## x dplyr::select() masks MASS::select()
```

```
library(pander)
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
## combine
```

```
library(sjmisc) # transpose df
```

```
##
## Attaching package: 'sjmisc'
```

```
## The following object is masked from 'package:purrr':
##
## is_empty
```

```
## The following object is masked from 'package:tidyr':
##
## replace_na
```

```
## The following object is masked from 'package:tibble':
##
## add_case
```

```
library(here)
```

```
## here() starts at /Users/franceslinyc/INLA-with-Spatial-data-2022
```

```
# Load data
library(DCclusterm)
data(NY8)
```

## The NY8 data

The NY8 data set contains the number of incident leukemia cases per census tract in an eight-country region of upstate New York from 1978-1982 (Waller & Gotway, 2004; Bivand et al., 2008). The NY8 data set can be accessed from the **R** package `DCclusterm`.

```

# Load data
data(NY8)

# View data
#head(NY8)
NY8

## class      : SpatialPolygonsDataFrame
## features   : 281
## extent     : 358241.9, 480393.1, 4649755, 4808545 (xmin, xmax, ymin, ymax)
## crs        : +proj=utm +zone=18 +ellps=WGS84 +units=m +no_defs
## variables  : 17
## names      : AREANAME, AREAKEY, X, Y, POP8, TRACTCAS, PROPCAS, PCTOWNHOME, I
## min values : Auburn city, 36007000100, -55.4823, -75.2907, 9, 0, 0, 0.00082237, 0
## max values : Vestal town, 36109992300, 53.5086, 56.41013, 13015, 9.29, 0.006993, 1, 0

# Check class
class(NY8)

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

# Convert it to a df?
# https://www.paulamoraga.com/book-geospatial/sec-spatialdataandCRS.html
NY8@data %>% head %>% pander

```

Table 1: Table continues below

	AREANAME	AREAKEY	X	Y	POP8	TRACTCAS
0	Binghamton city	36007000100	4.069	-67.35	3540	3.08
1	Binghamton city	36007000200	4.639	-66.86	3560	4.08
2	Binghamton city	36007000300	5.709	-66.98	3739	1.09
3	Binghamton city	36007000400	7.614	-66	2784	1.07
4	Binghamton city	36007000500	7.316	-67.32	2571	3.06
5	Binghamton city	36007000600	8.559	-66.93	2729	1.06

Table 2: Table continues below

	PROPCAS	PCTOWNHOME	PCTAGE65P	Z	AVGIDIST	PEXPOSURE
0	0.00087	0.3277	0.1466	0.142	0.2374	3.167
1	0.001146	0.4268	0.2351	0.3555	0.2087	3.039
2	0.000292	0.3377	0.138	-0.5817	0.1709	2.838
3	0.000384	0.4616	0.1189	-0.2963	0.1406	2.643
4	0.00119	0.1924	0.1416	0.4569	0.1578	2.759
5	0.000388	0.3652	0.1411	-0.2812	0.1726	2.848

	Cases	Xm	Ym	Xshift	Yshift
<b>0</b>	3.083	4069	-67353	423391	4661502
<b>1</b>	4.083	4639	-66862	423961	4661993
<b>2</b>	1.087	5709	-66978	425031	4661878
<b>3</b>	1.065	7614	-65996	426935	4662859
<b>4</b>	3.06	7316	-67318	426638	4661537
<b>5</b>	1.064	8559	-66934	427880	4661921

```
# # Plot it
# plot(NY8) # Just the map now.
```

```
# Check dimension
NY8@data %>% dim
```

```
## [1] 281 17
```

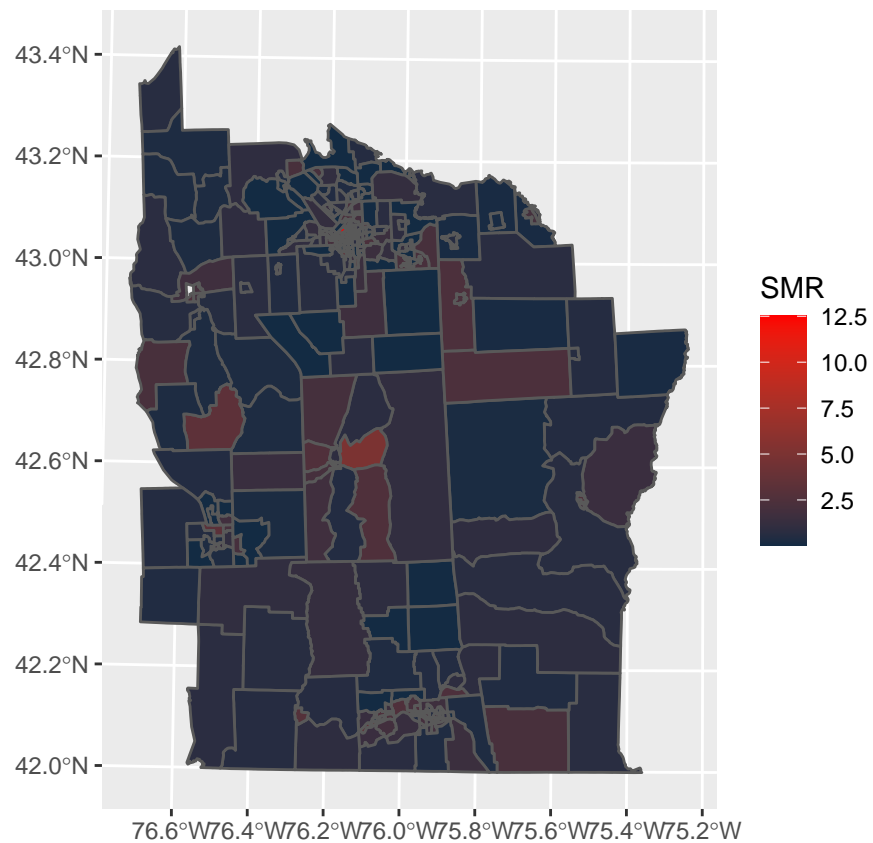
## Plotting

```
# Convert to sf
library(sf)
NY8_sf <- st_as_sf(NY8)
```

```
# Create the standardized mortality ratio (SMR) variable
# https://www.r-bloggers.com/2019/11/spatial-data-analysis-with-inla/
rate <- sum(NY8_sf$Cases) / sum(NY8_sf$POP8)
```

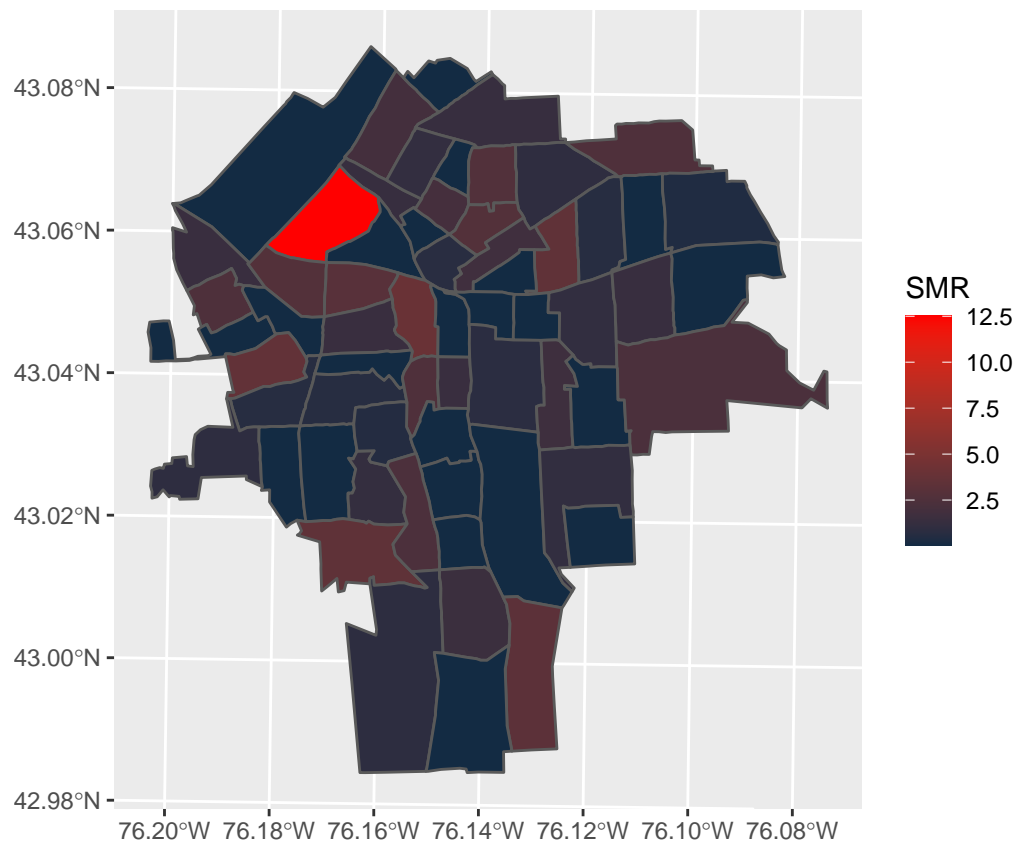
```
NY8_sf <- NY8_sf %>% mutate(
  Expected = POP8 * rate,
  SMR = Cases / Expected
)
```

```
# Plot SMR
ggplot(NY8_sf) + geom_sf(aes(fill = SMR)) + # Look nice!
  scale_fill_gradient(high = "red")
```



## Subsetting then plotting

```
# Subset to include Syracuse city only  
syracuse <- which(NY8$AREANAME == "Syracuse city")  
  
# Plot it  
ggplot(NY8_sf[syracuse, ]) + geom_sf(aes(fill = SMR)) +  
  scale_fill_gradient(high = "red")
```



## Poisson Models

### Fitting a Poisson regression model

```
#install.packages("INLA") # run once
#not available for this R version...
#install.packages("INLA", repos=c(getOption("repos"), INLA="https://inla.r-inla-download.org/R/stable")
library(INLA) # Now it works.
```

```
## Loading required package: Matrix
```

```
##
```

```
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
```

```
##
```

```
##      expand, pack, unpack
```

```
## Loading required package: foreach
```

```
##
```

```
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##   accumulate, when

## This is INLA_22.03.16 built 2022-03-16 13:24:07 UTC.
## - See www.r-inla.org/contact-us for how to get help.
```

Let's work on some toy examples first before coming to fix the issue. Toy examples work fine. Issues seem to related to Cases. Rounding Cases work but results differ a bit.

```
# Fit a Poisson regression model
m_fixed <- inla(round(Cases) ~ 1 + AVGIDIST,
               data = NY8_sf,
               family = "poisson",
               E = NY8_sf$Expected,
               control.predictor = list(compute = TRUE),
               control.compute = list(dic = TRUE, waic = TRUE))
```

```
# summary(m_fixed) %>% pander # very bad!
```

```
summary(m_fixed)
```

```
##
## Call:
##   c("inla.core(formula = formula, family = family, contrasts = contrasts,
##   ", " data = data, quantiles = quantiles, E = E, offset = offset, ", "
##   scale = scale, weights = weights, Ntrials = Ntrials, strata = strata,
##   ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose =
##   verbose, ", " lincomb = lincomb, selection = selection, control.compute
##   = control.compute, ", " control.predictor = control.predictor,
##   control.family = control.family, ", " control.inla = control.inla,
##   control.fixed = control.fixed, ", " control.mode = control.mode,
##   control.expert = control.expert, ", " control.hazard = control.hazard,
##   control.lincomb = control.lincomb, ", " control.update =
##   control.update, control.lp.scale = control.lp.scale, ", "
##   control.pardiso = control.pardiso, only.hyperparam = only.hyperparam,
##   ", " inla.call = inla.call, inla.arg = inla.arg, num.threads =
##   num.threads, ", " blas.num.threads = blas.num.threads, keep = keep,
##   working.directory = working.directory, ", " silent = silent, inla.mode
##   = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame =
##   .parent.frame)")
## Time used:
##   Pre = 2.74, Running = 0.223, Post = 0.0195, Total = 2.98
## Fixed effects:
##           mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept) -0.097 0.046    -0.188   -0.096    -0.008 -0.096   0
## AVGIDIST      0.324 0.078     0.163    0.327     0.471 0.332   0
##
## Deviance Information Criterion (DIC) .....: 1016.44
## Deviance Information Criterion (DIC, saturated) ....: -649.28
## Effective number of parameters .....: 2.00
##
```

```
## Watanabe-Akaike information criterion (WAIC) ...: 1017.37
## Effective number of parameters .....: 2.69
##
## Marginal log-Likelihood: -514.42
## is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

## Fitting a Poisson regression model with random effects

```
# Fit a Poisson regression model with random effects
```

```
NY8_sf <- NY8_sf %>% mutate(
  ID = 1:nrow(NY8)) # Use ID as the random effect

m_random <- inla(round(Cases) ~ 1 + AVGIDIST + f(ID, model = "iid"),
  data = NY8_sf,
  family = "poisson",
  E = NY8_sf$Expected,
  control.predictor = list(compute = TRUE),
  control.compute = list(dic = TRUE, waic = TRUE))
```

```
summary(m_random)
```

```
##
## Call:
## c("inla.core(formula = formula, family = family, contrasts = contrasts,
## ", " data = data, quantiles = quantiles, E = E, offset = offset, ", "
## scale = scale, weights = weights, Ntrials = Ntrials, strata = strata,
## ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose =
## verbose, ", " lincomb = lincomb, selection = selection, control.compute
## = control.compute, ", " control.predictor = control.predictor,
## control.family = control.family, ", " control.inla = control.inla,
## control.fixed = control.fixed, ", " control.mode = control.mode,
## control.expert = control.expert, ", " control.hazard = control.hazard,
## control.lincomb = control.lincomb, ", " control.update =
## control.update, control.lp.scale = control.lp.scale, ", "
## control.pardiso = control.pardiso, only.hyperparam = only.hyperparam,
## ", " inla.call = inla.call, inla.arg = inla.arg, num.threads =
## num.threads, ", " blas.num.threads = blas.num.threads, keep = keep,
## working.directory = working.directory, ", " silent = silent, inla.mode
## = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame =
## .parent.frame)")
## Time used:
## Pre = 2.47, Running = 0.287, Post = 0.0203, Total = 2.78
## Fixed effects:
##          mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept) -0.184 0.062    -0.311   -0.182    -0.066 -0.179   0
## AVGIDIST      0.363 0.114      0.137    0.363     0.586 0.365   0
##
## Random effects:
##   Name      Model
##   ID IID model
```

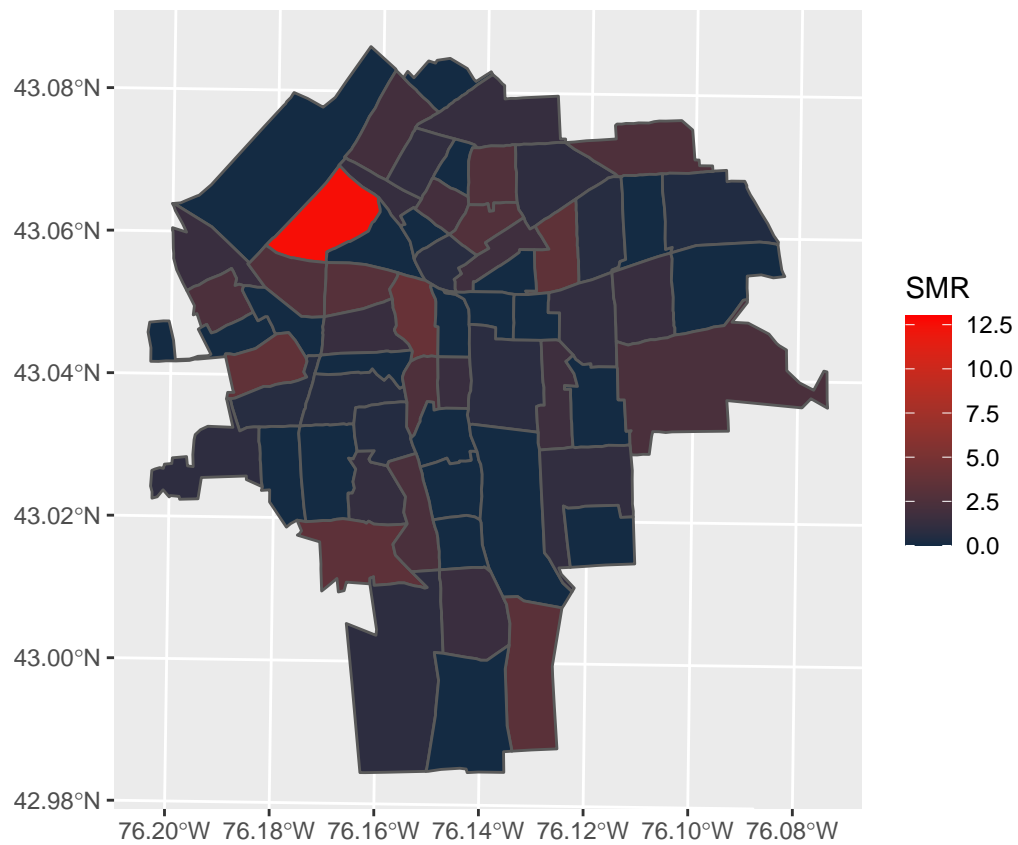


```
##
## Model hyperparameters:
##           mean    sd 0.025quant 0.5quant 0.975quant mode
## Precision for ID 6.10 2.92      3.11    5.42      13.32 4.63
##
## Deviance Information Criterion (DIC) .....: 979.25
## Deviance Information Criterion (DIC, saturated) .....: -686.48
## Effective number of parameters .....: 73.40
##
## Watanabe-Akaike information criterion (WAIC) ...: 983.62
## Effective number of parameters .....: 64.16
##
## Marginal log-Likelihood: -512.10
## is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

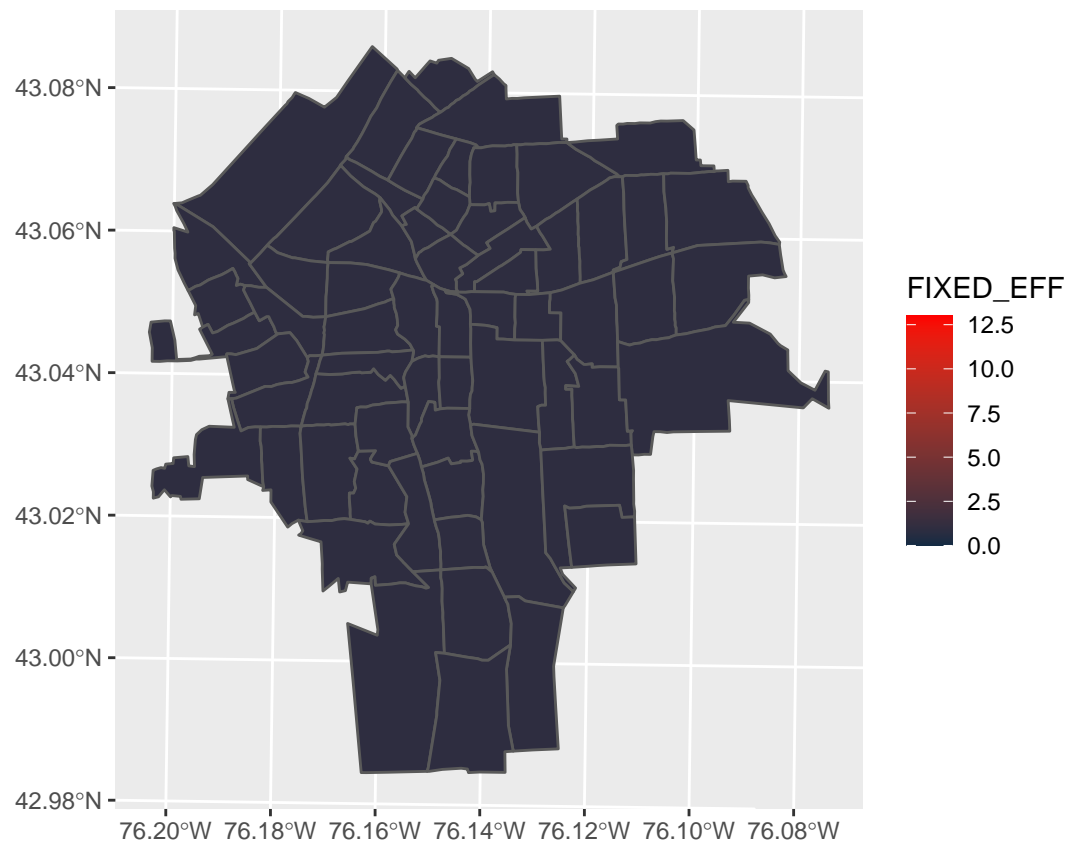
## Plotting

```
# Add fitted values for both m1 & m2
NY8_sf <- NY8_sf %>% mutate(
  FIXED_EFF = m_fixed$summary.fitted[, "mean"],
  IID_EFF = m_random$summary.fitted[, "mean"]
)

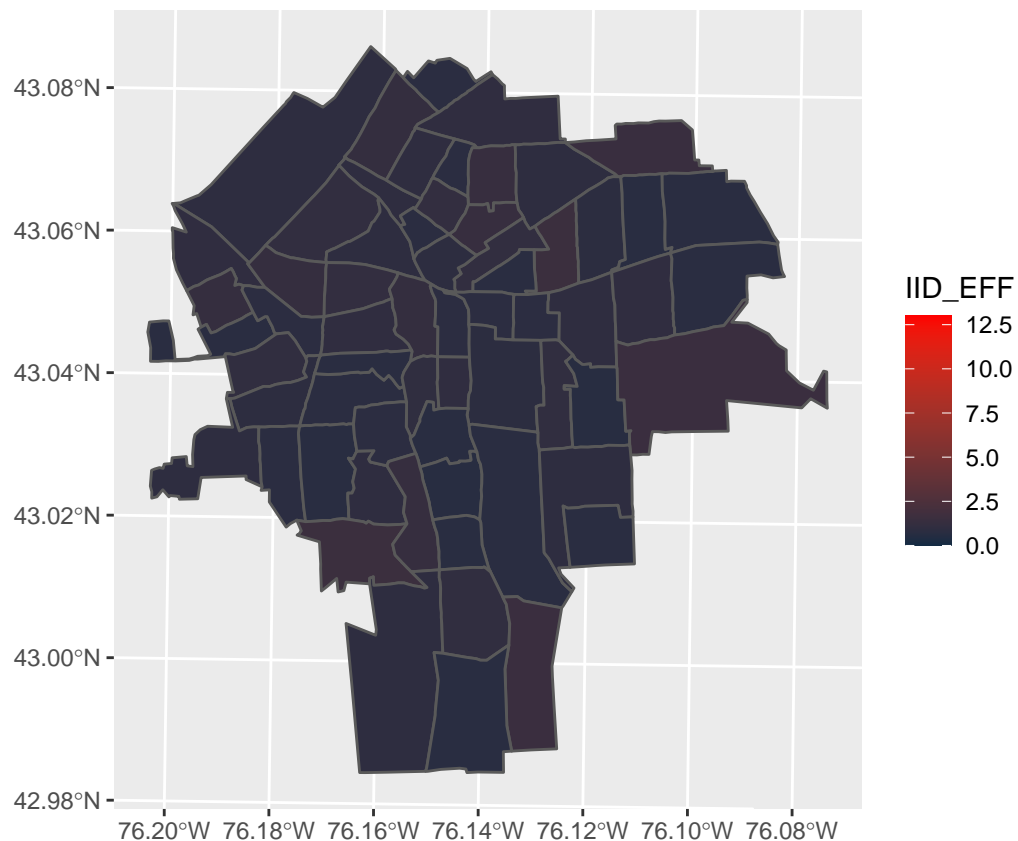
# Plot them but for Syracuse city only
ggplot(NY8_sf[syracuse, ]) + geom_sf(aes(fill = SMR)) +
  scale_fill_gradient(high = "red", limits = c(0, 13)) -> p_m0
p_m0
```



```
ggplot(NY8_sf[syracuse, ]) + geom_sf(aes(fill = FIXED_EFF)) + #, show.legend = FALSE) +
  scale_fill_gradient(high = "red", limits = c(0, 13)) -> p_m1
p_m1
```



```
ggplot(NY8_sf[syracuse, ]) + geom_sf(aes(fill = IID_EFF)) + # , show.legend = FALSE) +
  scale_fill_gradient(high = "red", limits = c(0, 13)) -> p_m2
p_m2
```



We might want them plotted with the same scale.

```
#grid.arrange(p_m0, p_m1, p_m2, nrow = 3, ncol = 1)
```

## Spatial Models for Areal (or Lattice) Data

### Plot spatial neighbors

An adjacency (or neighbour) matrix  $W$  is often used to describe spatial proximity in areal (lattice) data. Element  $W_{ij}$  is non-zero, if area  $i$  and  $j$  are neighbors. Element  $W_{ij}$  is zero, otherwise.

```
# Compute adjacency matrix
NY8.nb <- poly2nb(NY8) # construct the neighbours list / neighbour matrix
NY8.nb
```

```
## Neighbour list object:
## Number of regions: 281
## Number of nonzero links: 1624
## Percentage nonzero weights: 2.056712
## Average number of links: 5.779359
```

```
class(NY8.nb)
```

```
## [1] "nb"
```

## Plot spatial neighbors using ggplot2

```
# Plot spatial neighbors using ggplot2
# https://mbjoseph.github.io/posts/2018-12-27-plotting-spatial-neighbors-in-ggplot2/
NY8_sp <- as(NY8_sf, 'Spatial') # NY8_sf is a "sf" "data.frame"
class(NY8_sp) # Now is a "SpatialPolygonsDataFrame"
```

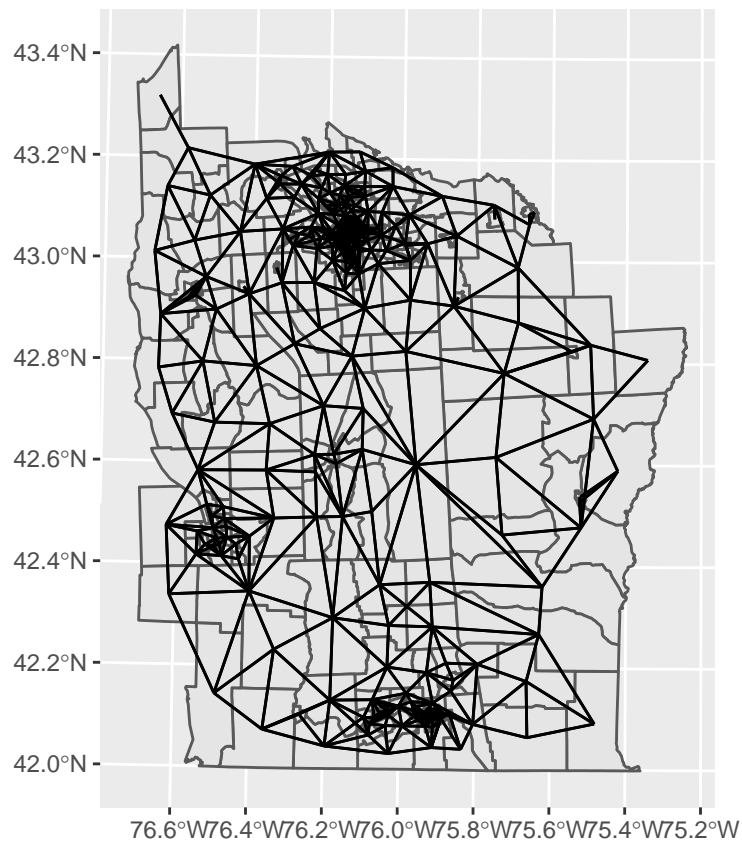
```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
neighbors <- poly2nb(NY8) # construct the neighbours list
neighbors_sf <- as(nb2lines(neighbors, coords = coordinates(NY8_sp)), 'sf')
```

```
## Warning in CRS(proj4string): CRS: projargs should not be NULL; set to NA
```

```
neighbors_sf <- st_set_crs(neighbors_sf, st_crs(NY8_sf))
```

```
ggplot(NY8_sf) +
  geom_sf() + # remove aes(fill = SMR)
  geom_sf(data = neighbors_sf)
```



```
#plot(NY8)
```

```
# plot(NY8)  
# plot(NY8.nb, coordinates(NY8), add = TRUE, pch = ".", col = "gray")
```

```
# Create sparse adjacency matrix  
# Or use the function nb2INLA to generate spatial neighbours for INLA  
NY8.mat <- as(nb2mat(NY8.nb, style = "B"), "Matrix") # generate a weights matrix for a neighbours list  
# Use this (NY8.mat) for the graph argument in the function inla  
#NY8.mat  
class(NY8.mat)
```

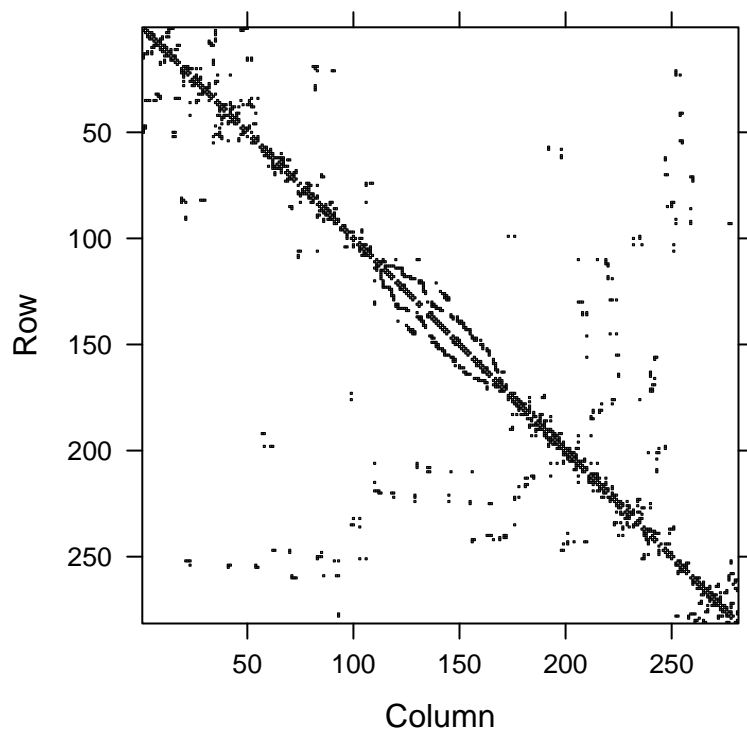
```
## [1] "dgCMatrix"  
## attr("package")  
## [1] "Matrix"
```

Here is a post that discusses the function `poly2nb` vs. `nd2INLA`. We might also need to check this tutorial to do more.

### Plot the adjacency matrix

Is the adjacency matrix the same as spatial neighbor?

```
# Plot the adjacency matrix  
image(NY8.mat)
```



**Dimensions: 281 x 281**

```
# summ <- summary(NY8.mat)
# #summ
# NY8.mat.df <- data.frame(
#   Origin = rownames(NY8.mat)[summ$i],
#   Destination = colnames(NY8.mat)[summ$j],
#   Weight = NY8.mat$x)
```

## Generalized Linear Models With Spatial Random Effects

The GLMs have the following form:

$$Y = X\beta + Z\alpha + \varepsilon,$$

where  $\beta$  is a vector of fixed effects with design matrix  $X$ ,  $\alpha$  is a vector of random effects with design matrix  $Z$ , and  $\varepsilon$  is an error term, where it is assumed that  $\varepsilon_i \sim N(0, \sigma^2)$ ,  $i = 1, \dots, n$ .

The vector of random effects  $\alpha$  is modeled as MVN (it is assumed that)

$$\alpha \sim N(0, \sigma_\alpha^2 \Sigma),$$

where  $\Sigma$  is defined such that it induces higher correlation with adjacent areas.

There are a few ways to include spatial dependence in  $\Sigma$ :

1. SAR (Simultaneous autoregressive)

$$\Sigma^{-1} = ((I - \rho W)^T (I - \rho W)),$$

where  $I$  is the identity matrix,  $\rho$  is a spatial autocorrelation parameter, and  $W$  is the adjacency matrix.

2. CAR (Conditional autoregressive)

$$\Sigma^{-1} = (I - \rho W)$$

3. ICAR (Intrinsic CAR):

$$\Sigma^{-1} = \text{diag}(n_i) - W,$$

where  $n_i$  is the number of neighbors of area  $i$ .

4. Mixture of matrices (Leroux et al.'s model)

$$\Sigma^{-1} = ((1 - \lambda)I_n + \lambda M), \lambda \in (0, 1)$$

where  $M$  is precision of intrinsic CAR specification.

Note.  $\Sigma^{-1} = Q$  is the precision matrix.

### Fit a SLM (spatial lag model)

This one seems a bit complicated. Let's wait till the last or skip it altogether.

## Fit a ICAR (Intrinsic CAR) model

```
# Setup model
# NY8.mat <- as(nb2mat(NY8.nb, style = "B"), "Matrix") # Already define earlier
```

```
# Fit model
start.time <- Sys.time()
m_icar <- inla(round(Cases) ~ 1 + AVGIDIST +
              f(ID, model = "besag", graph = NY8.mat),
              data = NY8_sf,
              family = "poisson",
              E = NY8_sf$Expected,
              control.predictor = list(compute = TRUE),
              control.compute = list(dic = TRUE, waic = TRUE))
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken # 3.155386 secs
```

```
## Time difference of 3.066259 secs
```

```
summary(m_icar)
```

```
##
## Call:
## c("inla.core(formula = formula, family = family, contrasts = contrasts,
## ", " data = data, quantiles = quantiles, E = E, offset = offset, ", "
## scale = scale, weights = weights, Ntrials = Ntrials, strata = strata,
## ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose =
## verbose, ", " lincomb = lincomb, selection = selection, control.compute
## = control.compute, ", " control.predictor = control.predictor,
## control.family = control.family, ", " control.inla = control.inla,
## control.fixed = control.fixed, ", " control.mode = control.mode,
## control.expert = control.expert, ", " control.hazard = control.hazard,
## control.lincomb = control.lincomb, ", " control.update =
## control.update, control.lp.scale = control.lp.scale, ", "
## control.pardiso = control.pardiso, only.hyperparam = only.hyperparam,
## ", " inla.call = inla.call, inla.arg = inla.arg, num.threads =
## num.threads, ", " blas.num.threads = blas.num.threads, keep = keep,
## working.directory = working.directory, ", " silent = silent, inla.mode
## = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame =
## .parent.frame)")
## Time used:
## Pre = 2.48, Running = 0.402, Post = 0.0171, Total = 2.9
## Fixed effects:
##          mean      sd 0.025quant 0.5quant 0.975quant  mode kld
## (Intercept) -0.163 0.054      -0.271   -0.162      -0.060 -0.160  0
## AVGIDIST      0.322 0.125        0.070    0.323        0.563  0.327  0
##
## Random effects:
##   Name      Model
##   ID Besags ICAR model
##
```



```
## Model hyperparameters:
##           mean    sd 0.025quant 0.5quant 0.975quant mode
## Precision for ID 2.76 1.36      1.25    2.43      6.26 2.00
##
## Deviance Information Criterion (DIC) .....: 968.26
## Deviance Information Criterion (DIC, saturated) .....: -697.47
## Effective number of parameters .....: 51.25
##
## Watanabe-Akaike information criterion (WAIC) ...: 972.20
## Effective number of parameters .....: 47.60
##
## Marginal log-Likelihood: -718.91
## is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

Later, we want to repeat these for all models. Perhaps consider a function?

```
# Get components from the results
# 2.4 Model assessment and model choice
# https://becarioprecario.bitbucket.io/inla-gitbook/ch-INLA.html#sec:modelassess
m_icar$mlik
```

```
##                                     [,1]
## log marginal-likelihood (integration) -718.8629
## log marginal-likelihood (Gaussian)    -718.9074
```

```
# m_icar$mlik[2,1]
m_icar$mlik[[2,1]]
```

```
## [1] -718.9074
```

```
m_icar$dic$dic
```

```
## [1] 968.2557
```

```
m_icar$waic$waic
```

```
## [1] 972.203
```

## Fit a BYM (Besag-York-Mollié) model

The BYM (Besag-York-Mollié) model is a convolution model of an ICAR (intrinsic CAR) effect and an iid Gaussian latent effect.

```
# Fit model
start.time <- Sys.time()
m_bym = inla(round(Cases) ~ 1 + AVGIDIST +
             f(ID, model = "bym", graph = NY8.mat),
             data = NY8_sf,
```

```

        family = "poisson",
        E = NY8_sf$Expected,
        control.predictor = list(compute = TRUE),
        control.compute = list(dic = TRUE, waic = TRUE))
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken # Time difference of 3.797915 secs

```

```
## Time difference of 3.657247 secs
```

```
summary(m_bym)
```

```

##
## Call:
##   c("inla.core(formula = formula, family = family, contrasts = contrasts,
##   ", " data = data, quantiles = quantiles, E = E, offset = offset, ", "
##   scale = scale, weights = weights, Ntrials = Ntrials, strata = strata,
##   ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose =
##   verbose, ", " lincomb = lincomb, selection = selection, control.compute
##   = control.compute, ", " control.predictor = control.predictor,
##   control.family = control.family, ", " control.inla = control.inla,
##   control.fixed = control.fixed, ", " control.mode = control.mode,
##   control.expert = control.expert, ", " control.hazard = control.hazard,
##   control.lincomb = control.lincomb, ", " control.update =
##   control.update, control.lp.scale = control.lp.scale, ", "
##   control.pardiso = control.pardiso, only.hyperparam = only.hyperparam,
##   ", " inla.call = inla.call, inla.arg = inla.arg, num.threads =
##   num.threads, ", " blas.num.threads = blas.num.threads, keep = keep,
##   working.directory = working.directory, ", " silent = silent, inla.mode
##   = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame =
##   .parent.frame)")
## Time used:
##   Pre = 2.49, Running = 0.976, Post = 0.0261, Total = 3.49
## Fixed effects:
##           mean      sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept) -0.163 0.054      -0.271   -0.163     -0.060 -0.161   0
## AVGIDIST      0.322 0.125       0.070    0.324      0.563  0.327   0
##
## Random effects:
##   Name      Model
##   ID BYM model
##
## Model hyperparameters:
##
##           mean      sd 0.025quant 0.5quant
## Precision for ID (iid component) 1772.87 1759.70   113.36 1249.70
## Precision for ID (spatial component) 2.70 1.14    1.21 2.46
##           0.975quant   mode
## Precision for ID (iid component) 6482.92 304.49
## Precision for ID (spatial component) 5.58 2.06
##
## Deviance Information Criterion (DIC) .....: 967.43
## Deviance Information Criterion (DIC, saturated) ....: -698.29

```

```
## Effective number of parameters .....: 51.67
##
## Watanabe-Akaike information criterion (WAIC) ...: 971.93
## Effective number of parameters .....: 48.34
##
## Marginal log-Likelihood: -458.85
## is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

Fit a mixture (Leroux et al.) model

```
# Setup model
ICARmatrix <- Diagonal(nrow(NY8.mat), apply(NY8.mat, 1, sum)) - NY8.mat
Cmatrix <- Diagonal(nrow(NY8), 1) - ICARmatrix
```

```
# Fit model
m_1er = inla(round(Cases) ~ 1 + AVGIDIST +
              f(ID, model = "generic1", Cmatrix = Cmatrix),
              data = NY8_sf,
              family = "poisson",
              E = NY8_sf$Expected,
              control.predictor = list(compute = TRUE),
              control.compute = list(dic = TRUE, waic = TRUE))
summary(m_1er)
```

```
##
## Call:
## c("inla.core(formula = formula, family = family, contrasts = contrasts,
## ", " data = data, quantiles = quantiles, E = E, offset = offset, ", "
## scale = scale, weights = weights, Ntrials = Ntrials, strata = strata,
## ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose =
## verbose, ", " lincomb = lincomb, selection = selection, control.compute
## = control.compute, ", " control.predictor = control.predictor,
## control.family = control.family, ", " control.inla = control.inla,
## control.fixed = control.fixed, ", " control.mode = control.mode,
## control.expert = control.expert, ", " control.hazard = control.hazard,
## control.lincomb = control.lincomb, ", " control.update =
## control.update, control.lp.scale = control.lp.scale, ", "
## control.pardiso = control.pardiso, only.hyperparam = only.hyperparam,
## ", " inla.call = inla.call, inla.arg = inla.arg, num.threads =
## num.threads, ", " blas.num.threads = blas.num.threads, keep = keep,
## working.directory = working.directory, ", " silent = silent, inla.mode
## = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame =
## .parent.frame)")
## Time used:
## Pre = 2.47, Running = 0.821, Post = 0.0217, Total = 3.31
## Fixed effects:
##          mean    sd 0.025quant 0.5quant 0.975quant  mode  kld
## (Intercept) -0.169 0.491    -1.034   -0.170     0.698 -0.168 0.05
## AVGIDIST     0.330 0.126     0.076    0.331     0.574 0.335 0.00
##
```

```
## Random effects:
##   Name      Model
##   ID Generic1 model
##
## Model hyperparameters:
##           mean      sd 0.025quant 0.5quant 0.975quant  mode
## Precision for ID 2.358 0.915      1.118    2.173      4.643 1.861
## Beta for ID      0.849 0.152      0.431    0.902      0.996 0.993
##
## Deviance Information Criterion (DIC) .....: 967.34
## Deviance Information Criterion (DIC, saturated) ....: -698.38
## Effective number of parameters .....: 56.46
##
## Watanabe-Akaike information criterion (WAIC) ...: 971.32
## Effective number of parameters .....: 51.48
##
## Marginal log-Likelihood: -508.29
## is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

Results for all models differ a bit.

## Get results for all models

```
# Get criteria
get_criteria <- function(model){
  mlik = model$mlik[[2,1]]
  dic = model$dic$dic
  waic = model$waic$waic
  criteria = c(mlik, dic, waic)
  return(criteria)
}
```

```
# Test
x = get_criteria(m_icar)
x
```

```
## [1] -718.9074 968.2557 972.2030
```

```
# # Still need SLM
# get_criteria(m_fixed)
# get_criteria(m_random)
# get_criteria(m_icar)
# get_criteria(m_bym)
# get_criteria(m_ler)
```

```
# Put into a df and consider saving to the folder results
criteria_df <- tibble(
  "fixed" = get_criteria(m_fixed),
  "iid" = get_criteria(m_random),
```

```

"ICAR" = get_criteria(m_icar),
"BYM" = get_criteria(m_bym),
"Leroux" = get_criteria(m_ler)
)
criteria_df

```

```

## # A tibble: 3 x 5
##   fixed iid ICAR BYM Leroux
##   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 -514. -512. -719. -459. -508.
## 2 1016.  979.  968.  967.  967.
## 3 1017.  984.  972.  972.  971.

```

We want DIC (deviance information criterion) and WAIC (Watanabe-Akaike information criterion (WAIC)) to be low since lower DIC or WAIC value indicates better fit of the model.

```

criteria_df <- criteria_df %>% rotate_df() %>% rename(
  Marg_logLik = V1,
  DIC = V2,
  WAIC = V3
) # %>% pander # won't work. %>% pander later.

```

```

# Write to the results folder
write_rds(criteria_df, here("results", "criteria_df.rds"))

```

## Plot results for all models

Can we also create a function for plotting?

```

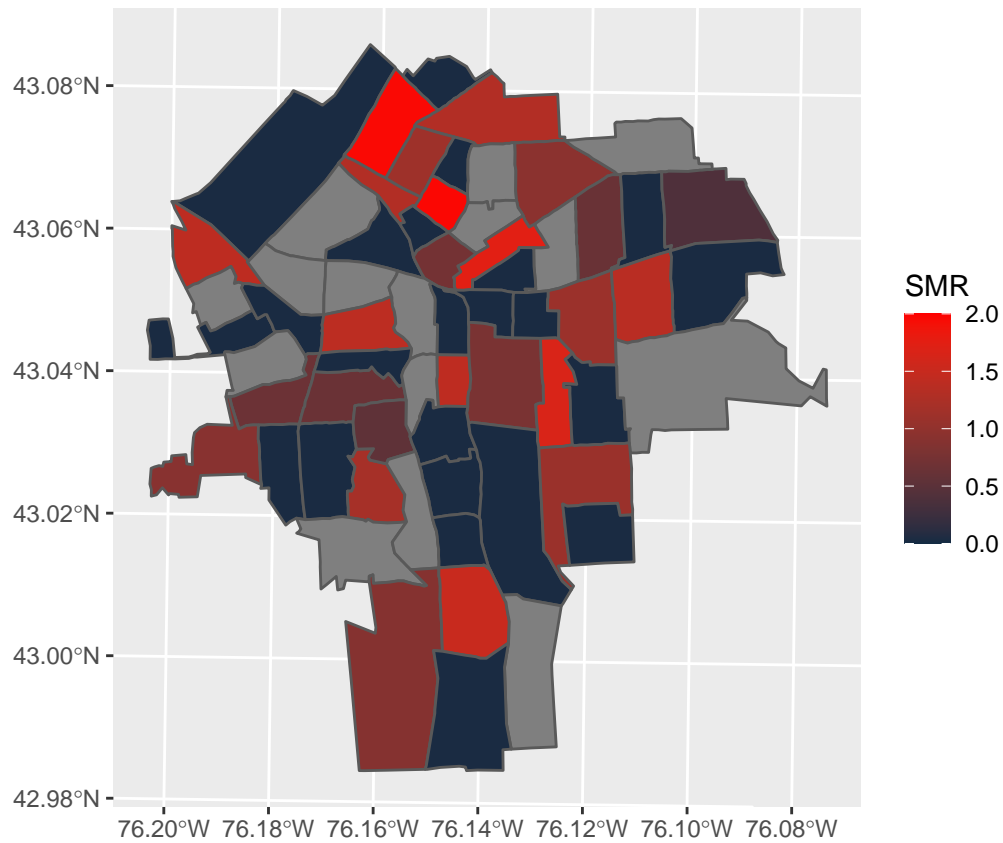
# Create sf for plotting
NY8_sf <- NY8_sf %>% mutate(
  FIXED_EFF = m_fixed$summary.fitted[, "mean"],
  IID_EFF = m_random$summary.fitted[, "mean"],
  ICAR = m_icar$summary.fitted[, "mean"],
  BYM = m_bym$summary.fitted[, "mean"],
  LER = m_ler$summary.fitted[, "mean"]
)

```

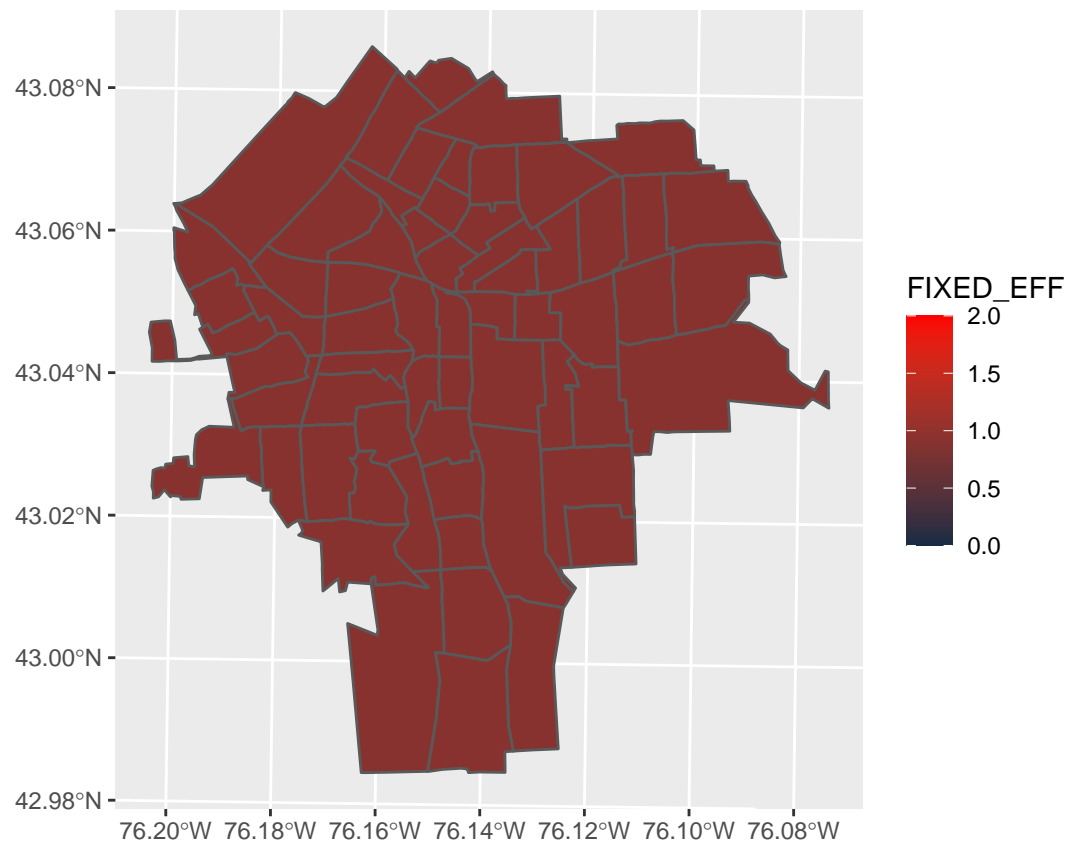
```

# Plot base case
ggplot(NY8_sf[syracuse, ]) +
  geom_sf(aes(fill = SMR)) +
  scale_fill_gradient(high = "red", limits = c(0, 2)) # Change limits but why?

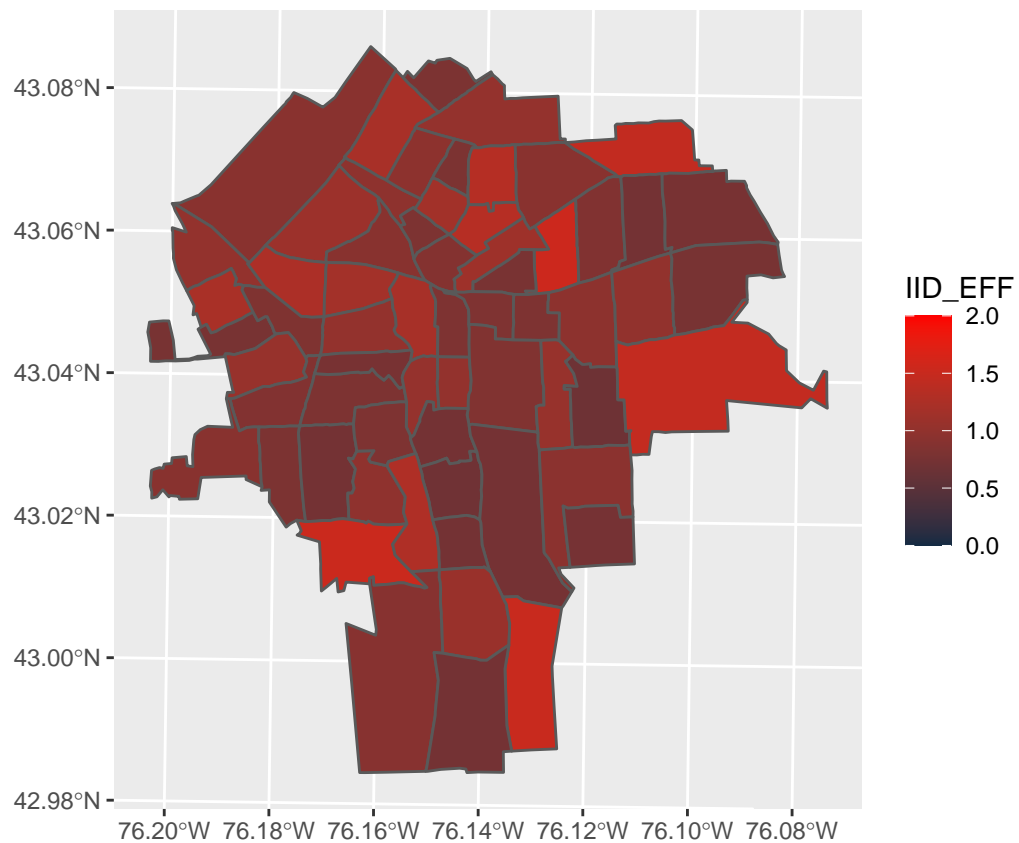
```



```
ggplot(NY8_sf[syracuse, ]) +
  geom_sf(aes(fill = FIXED_EFF)) +
  scale_fill_gradient(high = "red", limits = c(0, 2))
```



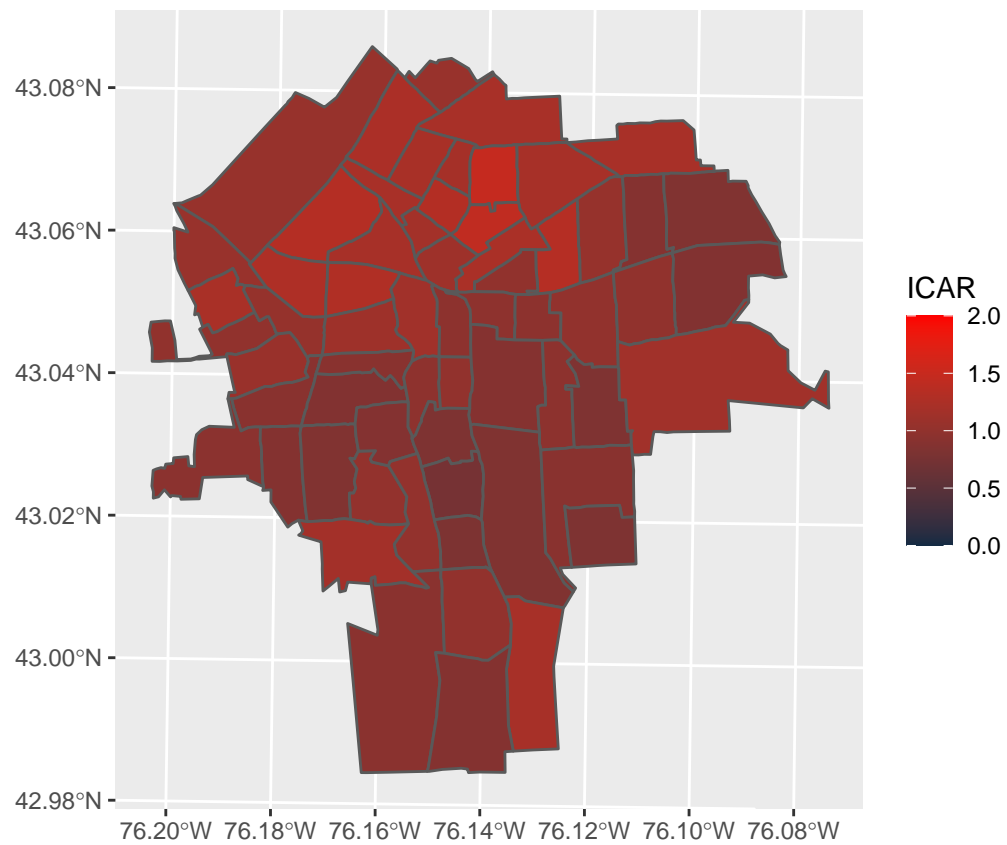
```
ggplot(NY8_sf[syracuse, ]) +  
  geom_sf(aes(fill = IID_EFF)) +  
  scale_fill_gradient(high = "red", limits = c(0, 2))
```



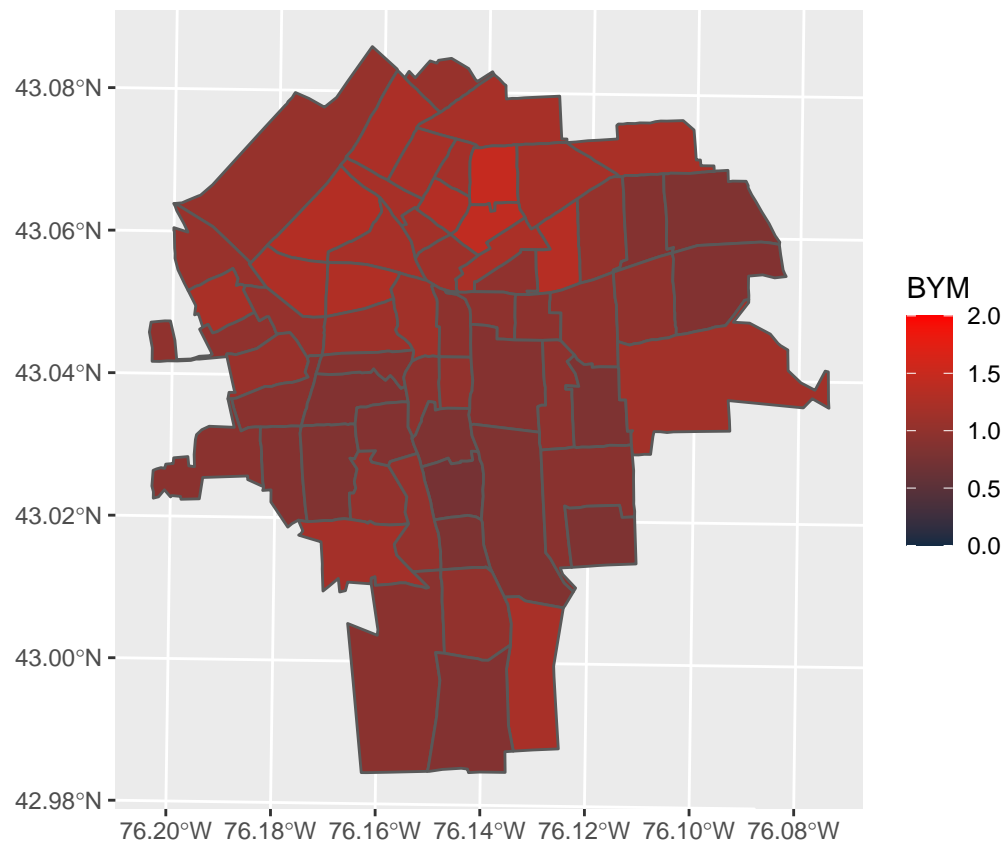
Something is off.

```
ggplot(NY8_sf[syracuse, ]) +
  geom_sf(aes(fill = ICAR)) +
  scale_fill_gradient(high = "red", limits = c(0, 2))
```

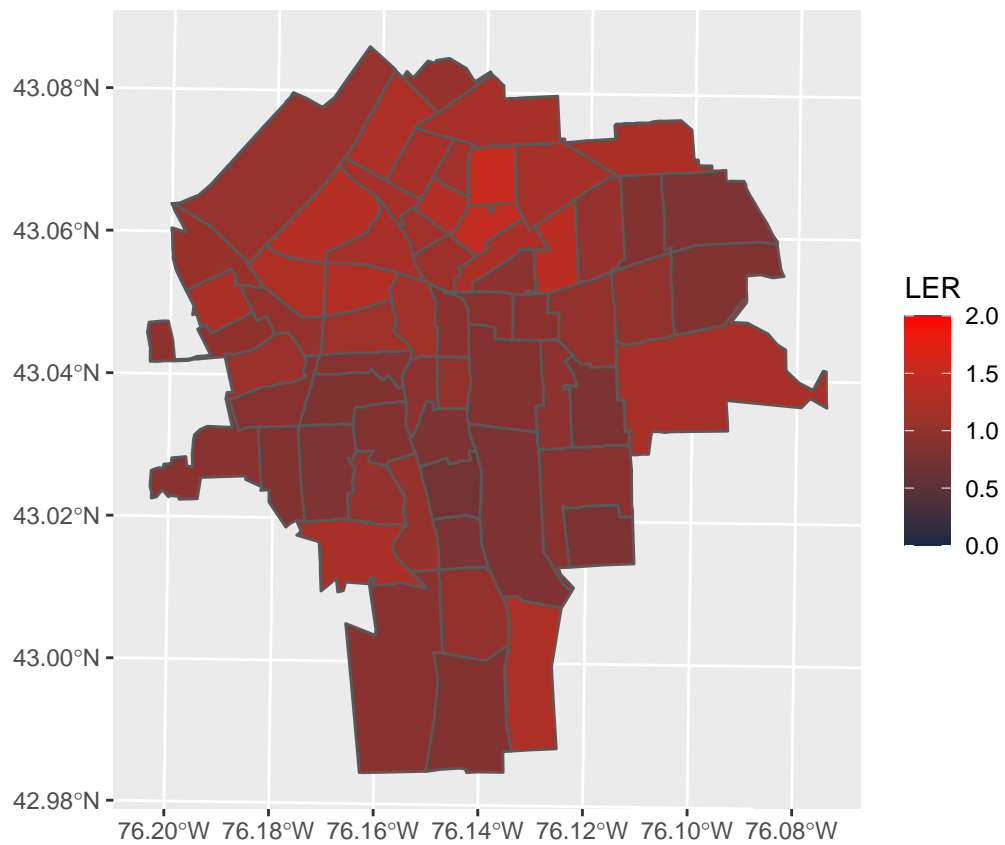




```
ggplot(NY8_sf[syracuse, ]) +  
  geom_sf(aes(fill = BYM)) +  
  scale_fill_gradient(high = "red", limits = c(0, 2))
```



```
ggplot(NY8_sf[syracuse, ]) +
  geom_sf(aes(fill = LER)) +
  scale_fill_gradient(high = "red", limits = c(0, 2))
```



## Reference

Gómez-Rubio, V. (2019). R-bloggers. Spatial Data Analysis with INLA. <https://www.r-bloggers.com/2019/11/spatial-data-analysis-with-inla/>.