

00_Leukemia_in_NY

Frances Lin

3/23/2022

Load packages

```
# Load packages
library(spdep)
```

```
## 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) # 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 3.3.5      v purrr  0.3.4
## v tibble  3.1.6      v dplyr  1.0.8
## 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(DCclusterm)
data(NY8)
```

The NY8 data

The NY8 data set contains the number of leukemia cases in an eight-country region of upstate New York from 1978-1982.

```
# 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, 1
## 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
0	3.083	4069	-67353	423391	4661502
1	4.083	4639	-66862	423961	4661993
2	1.087	5709	-66978	425031	4661878
3	1.065	7614	-65996	426935	4662859
4	3.06	7316	-67318	426638	4661537
5	1.064	8559	-66934	427880	4661921

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

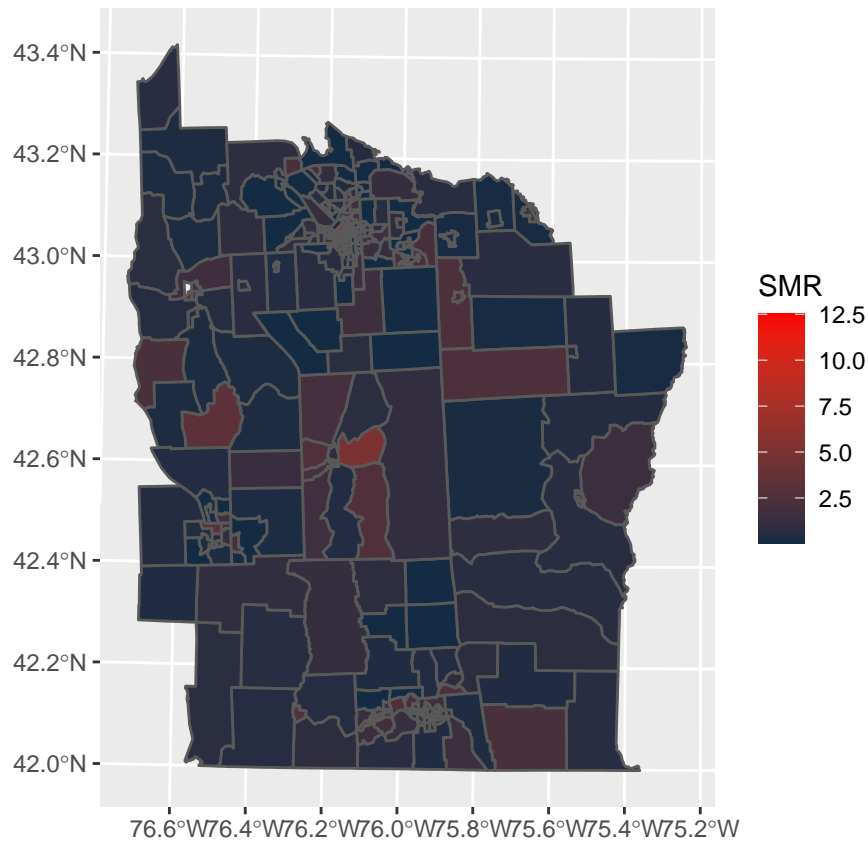
Plotting

```
# 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
)
```

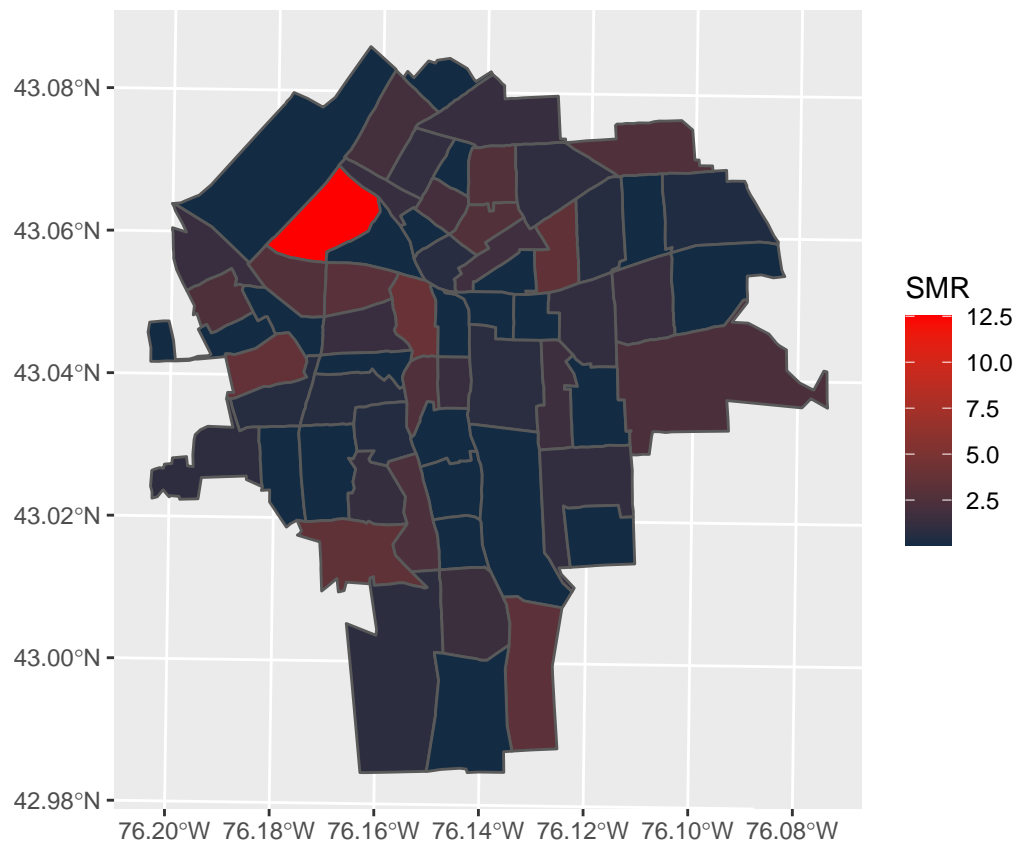
```
# Plot SMR
ggplot(NY8_sf) + geom_sf(aes(fill = SMR)) + # Look nice!
  scale_fill_gradient(high = "red")
```



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
m1 <- 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(m1) %>% pander # very bad!
```

```
summary(m1)
```

```
##
## 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 = 3.22, Running = 0.33, Post = 0.0288, Total = 3.58
## 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

m2 <- 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(m2)
```

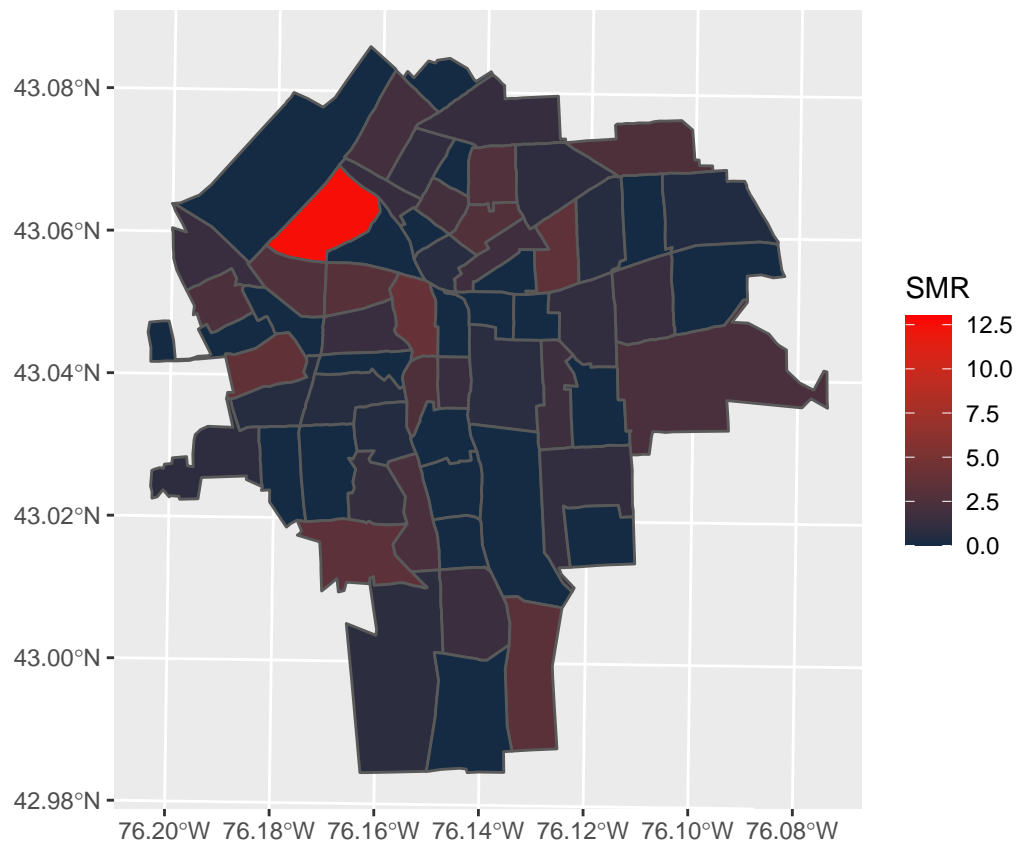
```
##
## Call:
## c("inla.core(formula = formula, family = family, contrasts = contrasts,
## ", " data = data, quantiles = quantiles, E = E, offset = offset, ", "
## scale = scale, weights = weights, Ntrials = Ntrials, strata = strata,
## ", " lp.scale = lp.scale, link.covariates = link.covariates, 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 = 3.26, Running = 0.415, Post = 0.0308, Total = 3.7
## 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 6.11 2.92      3.11    5.42      13.34 4.63
##
## Deviance Information Criterion (DIC) .....: 979.27
## Deviance Information Criterion (DIC, saturated) .....: -686.45
## Effective number of parameters .....: 73.42
##
## Watanabe-Akaike information criterion (WAIC) ....: 983.63
## 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)')
```

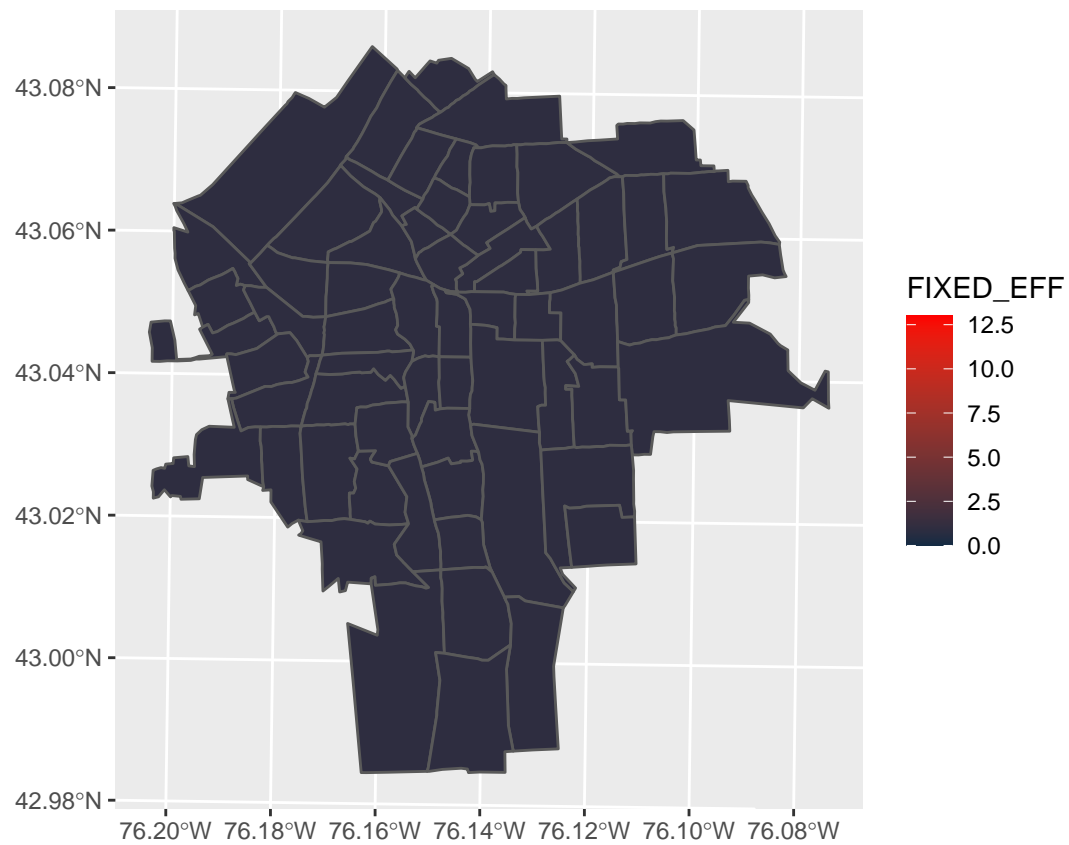
Plotting

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

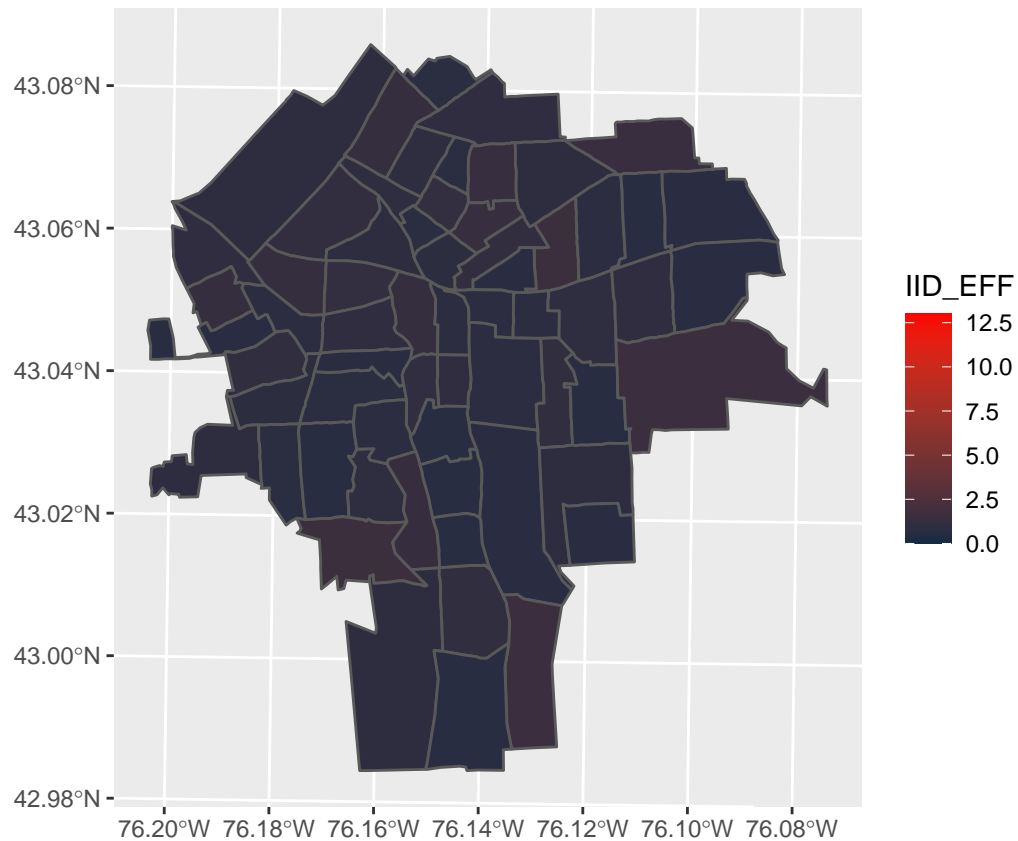
# Plot them but for Syracuse city only
ggplot(NY8_sf[syracuse, ]) + geom_sf(aes(fill = SMR)) +
  scale_fill_gradient(high = "red", limits = c(0, 13)) -> p_m0
p_m0
```

```
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 Lattice Data

Plot spatial neighbors

An adjacency matrix is often used to describe spatial proximity in lattice data.

```
# Compute adjacency matrix
NY8.nb <- poly2nb(NY8) # neighbors
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
# 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"
```

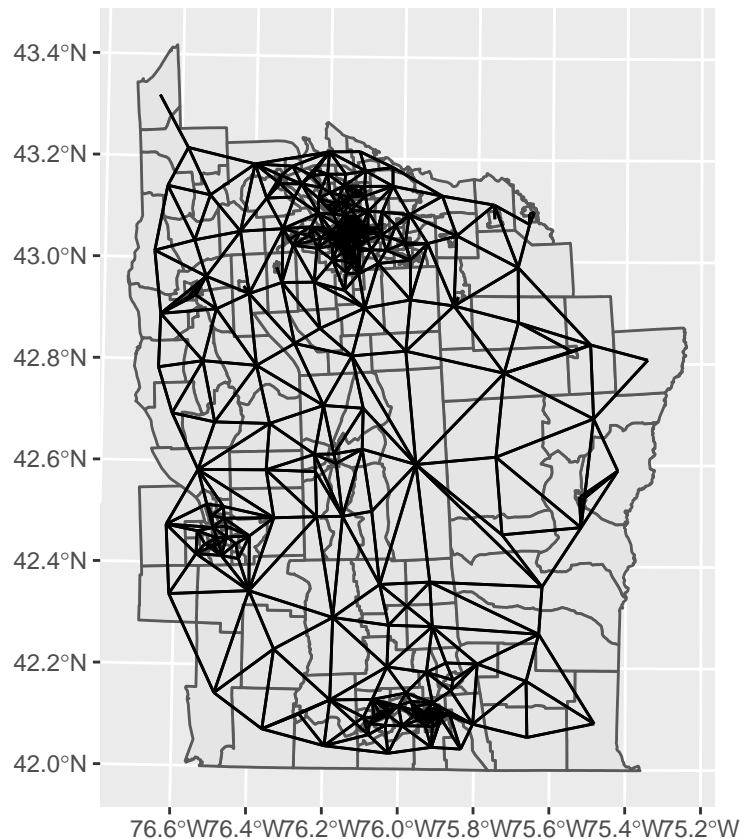
```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
neighbors <- poly2nb(NY8)
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")
```

Generalized Linear Models With Spatial Random Effects

The GLMs have the following form:

$$Y = X\beta + Zu + \varepsilon,$$

where β is a vector of fixed effects, u is a vector of random effects,

The vector of random effects u is modeled as MVN:

$$u \sim N(0, \sigma_u^2 \Sigma),$$

where Σ is defined s.t. it induces higher correlation with adjacent areas.

There are a few ways to include spatial dependence in Σ :

1. SAR (Simultaneous autoregressive):
2. CAR (Conditional autoregressive):
3. ICAR (Intrinsic CAR): , etc.

Fit

Fit

```
# m.bym = inla(Cases ~ 1 + AVGIDIST +
#             f(ID, model = "bym", graph = NY8.mat),
#             data = as.data.frame(NY8), E = NY8$Expected, family = "poisson",
#             control.predictor = list(compute = TRUE),
#             control.compute = list(dic = TRUE, waic = TRUE))
```

Reference

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