Prevalence of Childhood Lead Poisoning in Chicago Communities

Aaron Brake and Alvin Sheng

November 2020

Abstract

We fit various models of increasing complexity to the rates of childhood lead poisoning in 77 Chicago community areas. The data set has 46 features related to demographics and public health. We fitted the Beta, Negative Binomial, and SSVS models. We propose a hierarchical model to incorporate the spatial dependence of the spike-and-slab probabilities across community areas.

Honor Pledge: "We have neither given nor received any unauthorized aid on this assignment."
Aaron Brake, Alvin Sheng

1 Introduction

Modeling disease rates has become an important area of concern, especially considering the global pandemic ongoing. Being able to recognize factors that correspond with specific diseases can not only help with the general health of the population, but can also save municipalities money by being able make effective decisions to combat the disease. Our interest in this project was to model the rates of childhood lead poisoning in the Chicago area, using demographic information and public health indicators.

1.1 Data Description

This data has its origins in multiple sources. Firstly, the data contains information about 27 public health indicators from the city of Chicago. The health indicators include categories such as disease rates, economic information, crime incidents, and natality information. This data is segmented based on 77 communities located throughout Chicago. Secondly, information was gathered from a 2014 American Community survey. This included information like population estimates, ethnic and age demographics, education and employment rates, and poverty rates. Many of the variables in this survey were given as both counts and proportions based on community population. Again, this information was segmented at the same community level. Combined there were 86 variables, and 77 observations (one per community). Rates of childhood lead poisoning

are the response variable, and are measured as infections per 100. Thus, this reduced the dependency on community size.

1.2 Data Cleaning

Since many variables were given as both rates and proportions based on community size, there were many redundant and duplicate variables. Community size varied greatly between community area, so to "standardize" these variables the proportions were used. Thus, we would not see deceiving results based on community size. Additionally, there were some categories that added to one, thus only one was used. For example, the proportion of the population that is male and female adds to one so both columns are not needed, so one was dropped to avoid collinearity.

Further, there were some issues with missing variables in some of the variables, these were investigated and any columns with many missing values were dropped. There was one missing response in the childhood lead poisoning, but since JAGS can handle this automatically with PPD it was left in. The unit of childhood lead rate was in cases/100 people. This was transformed into two different response variables, one being a percentage between zero and one. The other was multiplying this percentage by the community population and then rounded to the nearest whole to get a count. This rounding should not change the counts dramatically, thus we are not worried about errors induced by the rounding. Furthermore, the data on other diseases were dropped to avoid over-fitting of the model. Lastly, the data was scaled and an intercept was added. After the data were cleaned, there were 43 variables remaining.

1.3 Incorporation of Spatial Data

In order to incorporate spatial data into our analysis, we engineered several features from the latitude-longitude coordinates of the boundaries of the community areas. First, we calculated the latitude and longitude of the centroid for each community area, Centroid Latitude and Centroid Longitude. Then, we created the variable Distance to Loop, which is the distance between the centroid of each community area to the centroid of Chicago's central business district, the Loop. We added these three spatial variables to our models.

Finally, we calculated the adjacency matrix A, in which $A_{ij} = 1$ if the boundaries of community areas i and j share at least one coordinate, and $A_{ij} = 0$ if $i = j$ or the community area boundaries do not share any coordinates. We will use the adjacency matrix to account for the spatial dependence in our model.

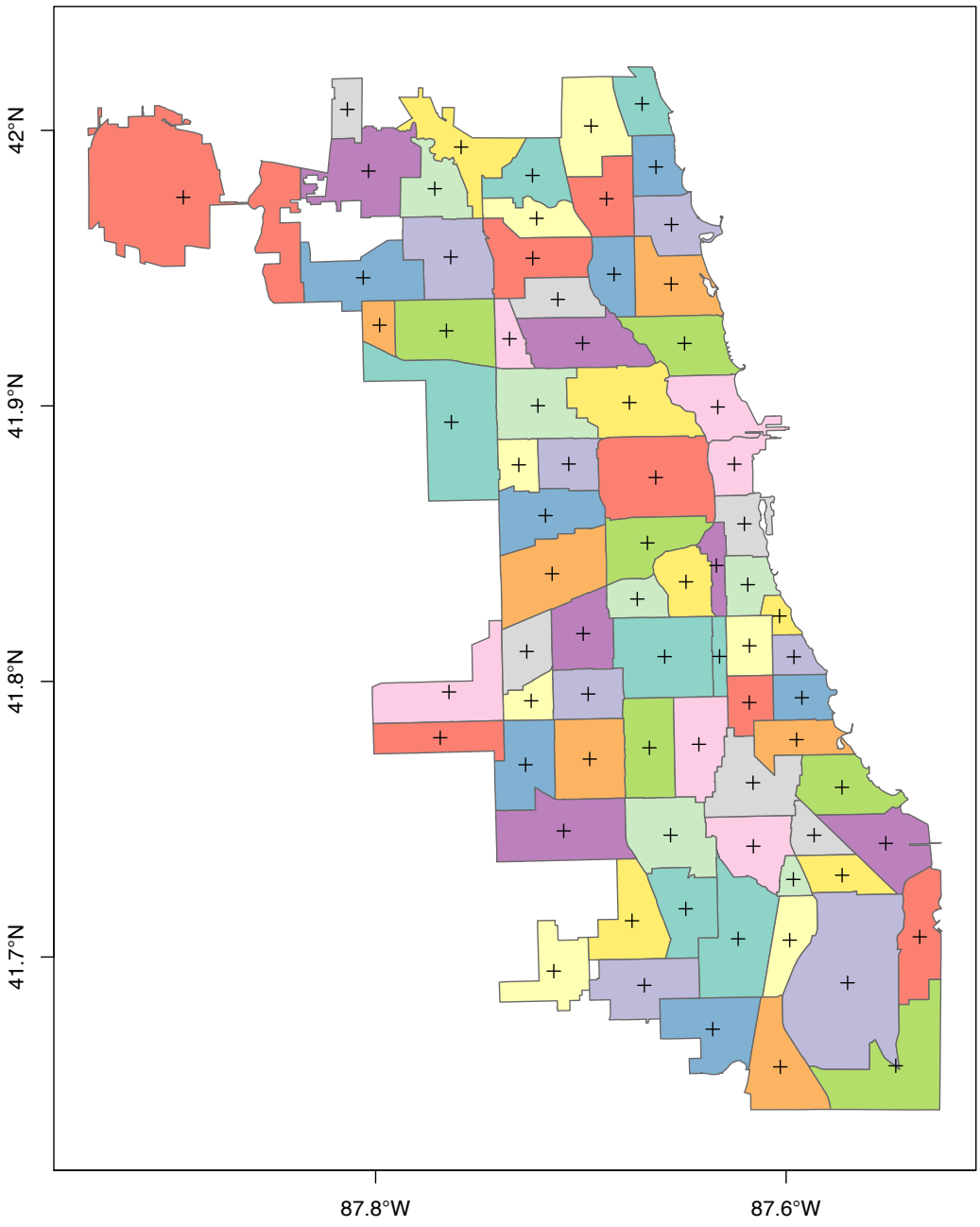

Figure 1: Map of Chicago community areas. The black plus signs indicate the centroids of the community areas.

2 Bayesian Model and Associated Methodology

We attempted to fit several models of increasing complexity.

2.1 Initial Models

Because the response variable is in the form of a proportion or count, we fit Beta, Poisson, and Negative Binomial regression models to the data. For the percentage response variable, the beta regression is appropriate. For the Beta regression, we used both the logistic and probit link functions.

We fit the Beta regression model. First, note that sing beta regression in JAGS does not allow for 0 or 1 values. Since there were 0 values in our data, a transformation is required to have these 0 values go away. A suitable transformation is proposed in Smithson and Verkuilen [1]

$$Y_i = \frac{Y_i(n-1) + 0.5}{n} \tag{1}$$

where n is the size, in this case 77. Now that the data has the correct support, the beta regression model can be run

$$\begin{aligned} Y_i|\beta, r &\sim Beta[r * \mu_i, r(1-\mu_i)] \\ logit(\mu_i) &= \mathbf{X}_i^T\beta \\ probit(\mu_i) &= \mathbf{X}_i^T\beta \\ \beta_j &\sim Normal(0, 0.01) \\ r &\sim Gamma(0.1, 0.1) \end{aligned} \tag{2}$$

where the link function is either logit or proit, and μ_i is the mean and the variance is $\mu_i(1-\mu_i)/(r+1)$, and $r > 0$. Thus, the r parameter can be seen as the concentration of the beta distribution around its mean.

We also fit the Poisson regression model

$$\begin{aligned} Y_i|\lambda_i &\sim Poisson(\lambda_i N_i) \\ \lambda_i &= \exp\left(\sum_{j=0}^{p} X_{ij}\beta_j\right) \\ \beta_j &\sim N(0, 0.01), \end{aligned} \tag{3}$$

where Y_i is the count of lead poisoning cases in community i and N_i is the population at community i. We then fit the negative binomial model

$$
\begin{aligned}
Y_i|\lambda_i, m &\sim NegBinomial(q_i, mN_i) \\
q_i &= \frac{m}{m+\lambda_i} \\
\lambda_i &= \exp\left(\sum_{j=0}^{p} X_{ij}\beta_j\right) \\
\beta_j &\sim N(0, 0.01) \\
m &\sim Gamma(0.1, 0.1).
\end{aligned} \tag{4}
$$

2.2 Stochastic Search Variable Selection

Next, we fit Beta and Negative Binomial regression models with stochastic search variable selection (SSVS). In this case, the models are the same, except the formulation and priors of β parameters are changed to the so-called "spike and slab" priors. The SSVS is defined as follows

$$
\begin{aligned}
\beta_j &= \gamma_j \delta_j \\
\gamma_j &\sim Bernoulli(0.5) \\
\delta_j &\sim Normal(0, \tau) \\
\tau &\sim Gamma(0.1, 0.1)
\end{aligned} \tag{5}
$$

Using the SSVS allows for variable selection, since if the posterior sample of $\gamma_j = 0$ then $\beta_j = 0$. Thus, some covariates are not included in the model. This is a different variable selection criterion than standard forwards or backwards selection, since SSVS is a stochastic process and is done within the MCMC process. To select variables, the Marginal Inclusion Probability (MIP) can be used, which is the proportion of MCMC samples where the β_j is included in the model. The cutoff for MIP is usually 0.5, meaning only variables with 50% probability to be included in the model.

2.3 Spatial SSVS

Next, we attempted to incorporate a prior on the probability that a given covariate is included in the model that varies by location and accounts for the spatial dependence among adjacent communities. We shall refer to the resulting hierarchical model as Spatial SSVS.

Our approach is informed by Andersen et al.'s application of spatio-temporal spike-and-slab priors to under-determined linear systems (Equation 6) with many features but few observations, such as in signal processing [2].

$$y = X\beta + \epsilon \tag{6}$$

They assume that the probabilities that the coefficients $\beta_1, \ldots, \beta_p$ are nonzero have a spatial dependence. Thus, they impose a transformed Gaussian process on the spike-and-slab probabilities that account for the spatial dependence. Our approach is similar, except that we allow the spike-and-slab probabilities to vary among the 77 community areas as well as among the coefficients. We would account for the spatial dependence of the spike-and-slab probabilities for a given coefficient via a transformed multivariate Gaussian. Our hierarchical model is given below, in several layers. Uninformative priors for the parameters for a given layer are shown in the same layer, for convenience. First is the Data layer:

$$\begin{aligned} Y_i|\lambda_i, m &\sim NegBinomial(q_i, mN_i) \\ q_i &= \frac{m}{m+\lambda_i} \\ \lambda_i &= \exp\left(\beta_0 + \sum_{j=1}^{p} X_{ij}\beta_{ij}\right) \\ m &\sim Gamma(0.1, 0.1). \end{aligned} \tag{7}$$

Then, we have the Spike-and-Slab layer:

$$\begin{aligned} \beta_0 &\sim N(0, 0.1) \\ \beta_{ij} &= \gamma_{ij}\delta_j \\ \gamma_{ij} &\sim Bern(\pi_i) \\ \pi_i &= \Phi(s_i) \\ \delta_j &\sim N(0, \tau_1) \\ \tau_1 &\sim Gamma(0.1, 0.1). \end{aligned} \tag{8}$$

Finally, we have the Latent Spatial layer:

$$\begin{aligned} \mathbf{s} &\sim MVN(\mathbf{0}, \tau_2\Omega) \\ \Omega &= M - \rho A \\ \tau_2 &\sim Gamma(0.1, 0.1) \\ \rho &\sim Beta(1, 1), \end{aligned} \tag{9}$$

where M is the diagonal matrix with the i^{th} diagonal element equal to the number of community areas

that neighbor community area i, and A is the adjacency matrix as detailed in Section 1.3. The approach of defining the spatial inverse covariance matrix Ω as in Equation 9 was used to analyze the effect of gun control laws among US states in [3].

3 Numerical Results

3.1 Initial Model Results

3.1.1 Negative Binomial and Poisson

For both the Poisson and Negative Binomial models, four chains were run, each with a burn-in of 100,000 iterations and 1,000,000 post-burn-in samples. However, despite the large number of burn-in and post-burn-in samples, the models were not able to converge, as some β parameters had effective sample sizes less than 20 in both models. This is likely due to having too many covariates in the model.

The over-dispersion parameter m for the negative binomial model in Equation 4 is very close to zero, with a posterior mean of 3.27×10^{-5} and time-series S.E. $< 10^{-7}$. This indicates that there is likely to be over-dispersion in the model. We confirm this via a hypothesis test proposed by Cameron and Trivedi where the null hypothesis of equidispersion is tested against the alternative hypothesis of overdispersion [4]. The p-value of the test was 2.82×10^{-11}.

Therefore, we focus on the negative binomial model instead of the Poisson model when increasing the complexity.

3.1.2 Beta

For both the probit and logit link models, four chains were run, with 20000 burn-in iterations and 100,000 post-burn-in-samples were used. The Gelman–Rubin statistics for the probit model are not cause for concern. Additionally, the effective size of each model was large, leading the a belief that this model converged, but was not perfect. The Gelman-Rubin statistic for the logit model was large, and the effective size was under 100 for a few β parameters. Thus, the model did not converge well. Increasing the number of iterations in the MCMC could help with this, although since there are so many parameters, it would make sense to use variable selection to reduce the number of covariates.

3.2 SSVS Results

The SSVS was run for the negative binomial and beta probit link models. Both models showed acceptable convergence. Using a MIP cutoff of 0.5 in the negative binomial model, 8 variables were included. See Table

Covariate	Inclusion Prob	Median	95% Interval
Birth Rate	0.55	0.08	(-0.08,0.37)
Fertility Rate	0.51	0.06	(-0.14,0.36)
1st Trimester Prenatal Care	0.74	-0.15	(-0.43,0.01)
Homicide/Assaults	0.5	0.05	(-0.19,0.38)
Caucasian Percentage	0.5	-0.05	(-0.37,0.16)
Black Percentage	0.54	0.07	(-0.15,0.45)
Below 185% poverty line	0.5	0.04	(-0.21,0.39)
Below 200% poverty line	0.51	0.05	(-0.19,0.40)

Table 1: Negative Binomial SSVS included variables and related statistics

1 for the included variables and their posterior medians and 95% credible intervals.

For the beta probit model there was only one variable that had a MIP above 0.5, meaning there was only one variable with inclusion probability of 50%. This variable was Teen Birth Rate. If the MIP minimum value was lowered to 0.4, homicide, firearm crime, and the poverty indicators appear in the model. With this cutoff, 9 variables are included in the model. This seems like a reasonable size, without lowering the MIP cutoff too low.

3.3 Spatial SSVS Results

To obtain the posterior samples of the 77 by 46 matrix of $(\gamma)_{ij}$, we ran two chains, each with a burn-in of 140,000 iterations and 1,000,000 post-burn-in samples. All the variables of interest were monitored: ρ, $\delta_j, j = 1, \ldots, 46$, $\pi_i, i = 1, \ldots, 77$, and $(\gamma)_{ij}$. Because storing all the samples of a large matrix takes up too much memory, the samples were thinned by 10, leaving 100,000 samples to approximate the posterior. The MCMC chains failed to converge, with the minimum effective sample size being 54.42047 and the third quartile of the upper limits of the potential scale reduction factors (PSRF) being 1.163. The main culprits of the poor convergence are the π_i, as they make up the majority of the parameters that have effective sample sizes less than 150.

To obtain the posterior samples of the other parameters of interest, i.e. ρ, δ_j, and π_i, we ran two chains, each with a burn-in of 1,140,000 iterations and 1,000,000 post-burn-in samples. This time, the matrix $(\gamma)_{ij}$ was not monitored. The MCMC chains also failed to converge, but the convergence diagnostics improved upon those for the MCMC chains monitoring the $(\gamma)_{ij}$ matrix. The minimum effective sample size was 150.3 and the third quartile of the upper limits of the PSRFs was 1.159. Again, the main culprits of the poor convergence are the π_i, as they are the only parameters that have effective sample sizes less than 500.

Because the MCMC chains were not able to converge, the following results from the spatial SSVS model serve as a demonstration of the information that our proposed hierarchical model can provide, if it had a

sufficient number of posterior samples.

The posterior mean (time-series s.e.) of the spatial dependence parameter ρ, which ranges from $(0, 1)$, is 0.959 (0.001), which would indicate strong spatial dependence among the inclusion probabilities at different community areas.

As shown in Figure 2, one can see which community areas are more likely to have many included covariates based on the parameter π_i. One can also examine the MIP for a given covariate in each community area, as shown in Figures 3 and 4. The MIPs for the covariate Below 200% poverty line are very similar to the prior probabilities, π_i. This suggests that the $(\gamma)_{ij}$ may be sensitive to the prior. The MIPs for the covariate 1st Trimester Prenatal Care deviate slightly from the prior probabilities, π_i.

Additionally, one could focus on a community area and determine the covariates most likely to be included. For instance, for Hyde Park, the covariates with MIPs larger than 0.70 are the Birth Rate, General Fertility Rate, and Percentage of Population between 18 and 21.

4 Conclusion

We have fit various models of increasing complexity to the dataset. Among all of the models, the negative binomial SSVS is the most appropriate for the dataset. It is preferable to the Beta model, as it incorporates the population sizes of the community areas. Further, the MIP cutoff did not have to be lowered to get a model with multiple covariates. It was able to reduce the large feature space from 46 variables to eight. It should be noted that this variable selection should be seen as the best exploratory analysis model, and not a conclusive proposed final model.

Unfortunately, sufficiently exploring the posterior of our proposed hierarchical model was not able to be completed in a reasonable number of iterations. One way to overcome this issue is to use an Expectation Propagation algorithm to approximate the posterior, as in [2].

References

[1] Michael Smithson and Jay Verkuilen. A better lemon squeezer? maximum-likelihood regression with beta-distributed dependent variables. *Psychological Methods*, 11(1):54–71, 2006.

[2] Michael Riis Andersen, Aki Vehtari, Ole Winther, and Lars Kai Hansen. Bayesian inference for spatio-temporal spike-and-slab priors. *Journal of Machine Learning Research*, 18:1–58, 2017.

[3] Brian J. Reich and Sujit K. Ghosh. *Bayesian Statistical Methods*. CRC Press, 2019.

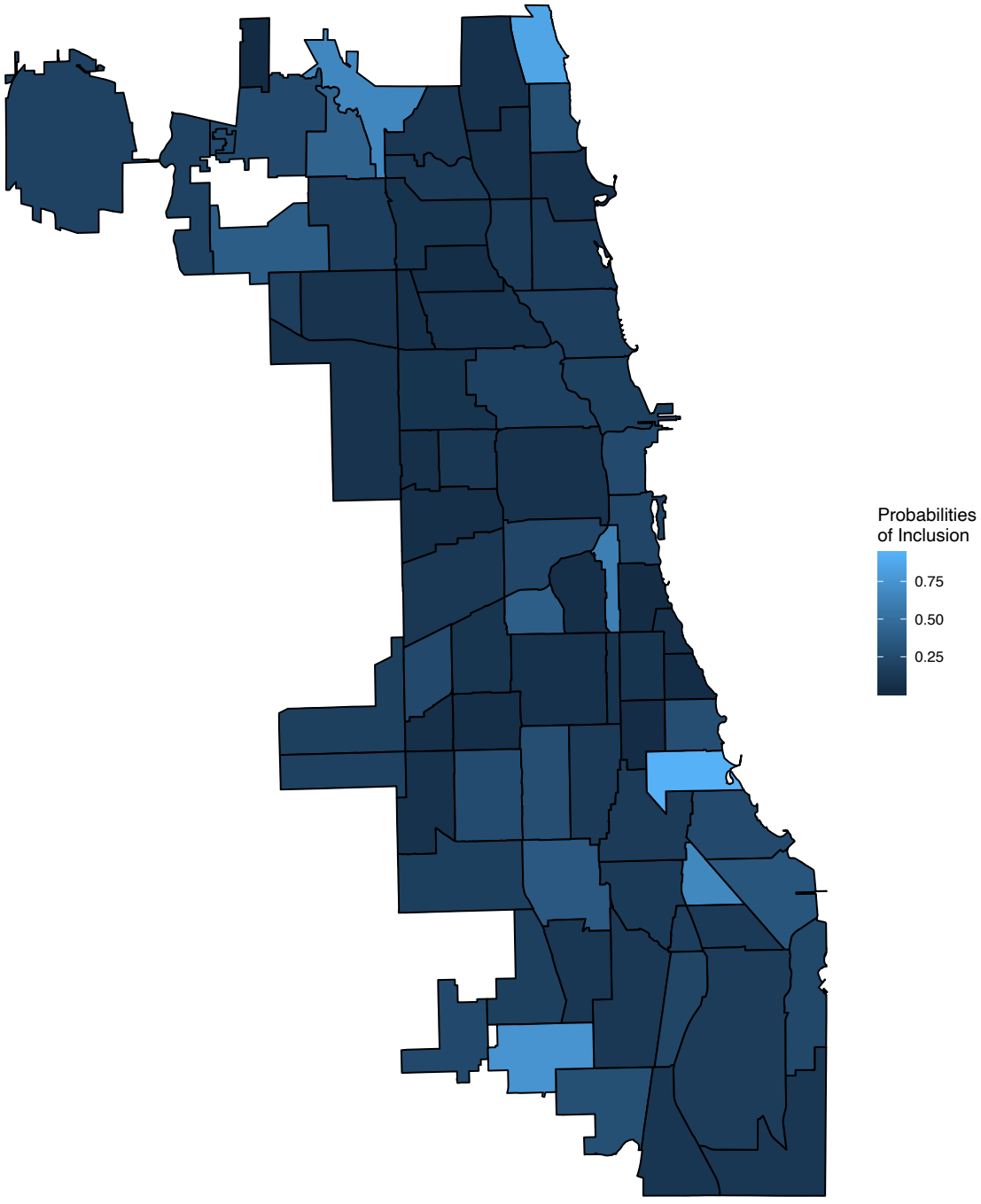

Figure 2: Posterior means of the probabilities of inclusion π_i for each community area. Maximum time-series standard error was 0.026.

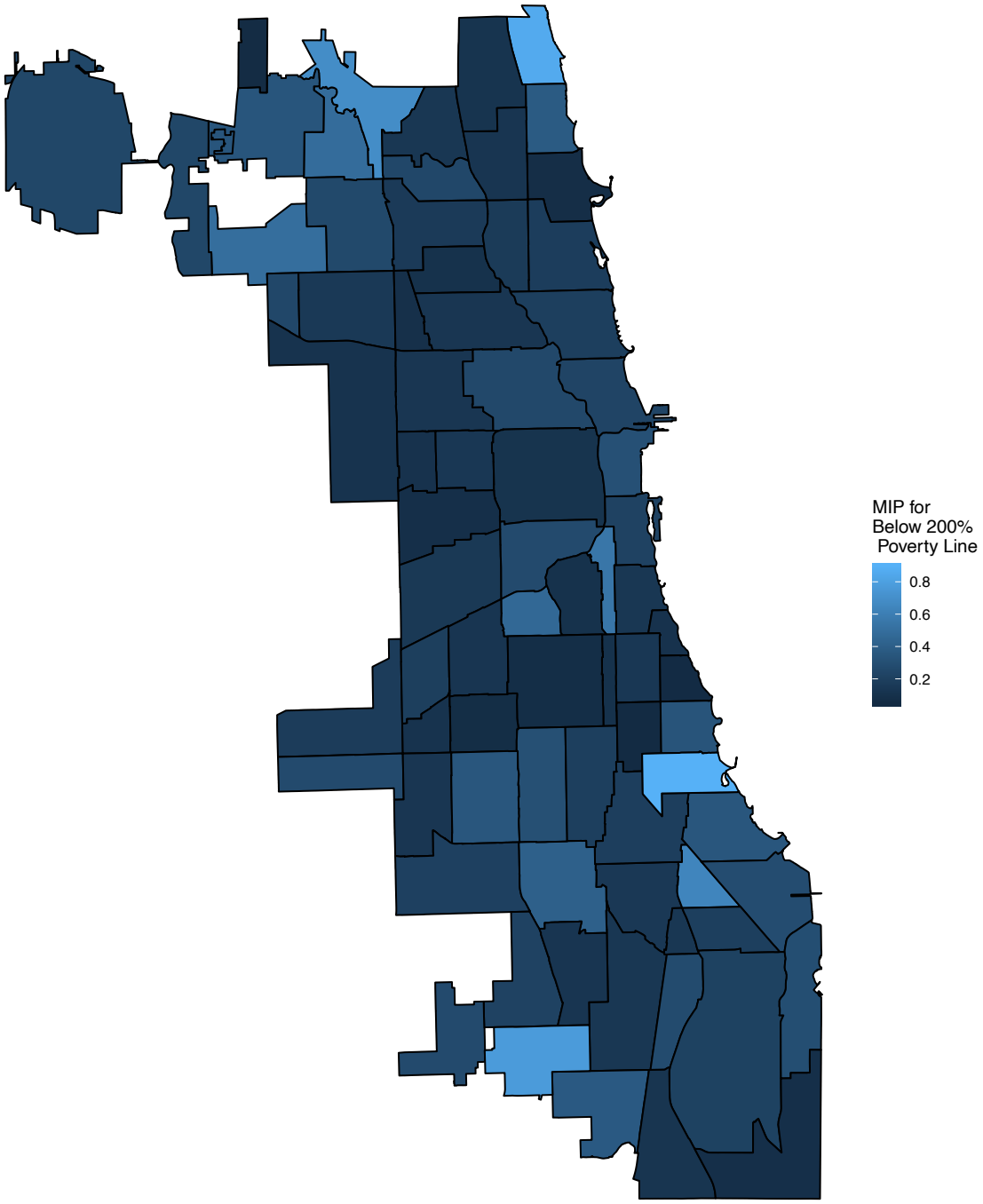

Figure 3: Marginal inclusion probabilities for Below 200% Poverty Line across community areas.

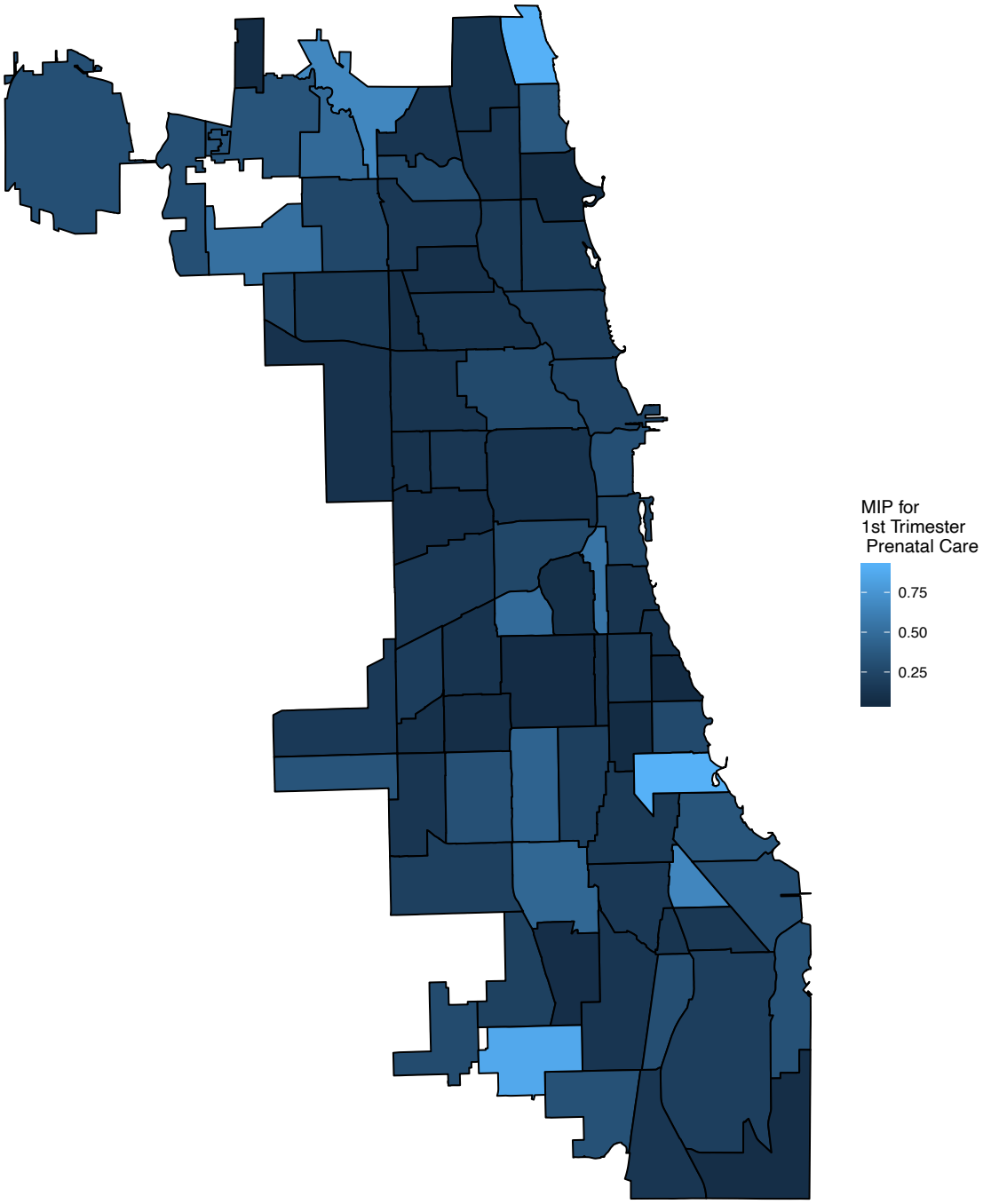

Figure 4: Marginal inclusion probabilities for 1st Trimester Prenatal Care across community areas.

[4] A. Colin Cameron and Pravin K. Trivedi. Regression-based tests for overdispersion in the poisson model. *Journal of Econometrics*, 46:347–364, 1990.

```
####################### Data Cleaning ######################



# Aaron's data cleaning code to get comareaCleaned.RData



comarea_init = read.csv("~/Documents/PublicHealthCh.csv")
orig_health_indic = read.csv("~/Documents/ComArea_ACS14_f.csv")
names(orig_health_indic) = clean_names

comarea_merge <- comarea_init[, -c(66:86)]
names(comarea_merge)[1] <- "ComAreaID"
names(orig_health_indic)[1] <- "ComAreaID"
comarea_merge <- merge(comarea_merge, orig_health_indic)
comarea <- comarea_merge[, -which(names(comarea_merge) %in%
                                    c("Community.Area.Name" ,
"Below.Poverty.Level", "No.High.School.Diploma", "Per.Capita.Income",
"Unemployment"))]

#Drop useless/repeated Columns
#Rename the Childhood lead poisoning column
comarea$ChlLeadP = comarea$Childhood.Lead.Poisoning


#Get the column numbers of the columns to keep
keepCols = c(1:8,13,14,23:25,31,33,37,39,41,43,45, 47,49, 51,53,55,57,
             59,61,63,68:74,76,78,80,82,84:88, 106)
#Make the dataframe with the correct columns
comareaCleaned = cbind(comarea[,keepCols],CentroidLat, CentroidLon)
#Make a column with percents of Childhood lead
comareaCleaned$ChlLeadPerc = comareaCleaned$ChlLeadP / 100
#Make the column of childhood lead counts (rounded to nearest whole)
comareaCleaned$ChlLeadPois = round(comareaCleaned$ChlLeadPerc *
comarea$Pop2014)




library(tidyverse)

### continuation of data cleaning

load("comareaCleaned.Rdata")
```

```
# Under18P Over18P add up to 100
# So make separate proportion categories for age

comareaCleaned$Bw5and18P <- comareaCleaned$Under18P - comareaCleaned$Under5P

comareaCleaned$Bw18and21P <- comareaCleaned$Over18P - comareaCleaned$Over21P

comareaCleaned$Bw21and65P <- comareaCleaned$Over21P - comareaCleaned$Over65P

# the rest of the people should be over 65. I just keep Under5P among the
original variables.



# add in Dist2Loop

load("spatial_vars.RData")

comareaCleaned$Dist2Loop <- Dist2Loop



# put PerCInc14 and Pov14 (as proportion) back in

comarea_init <- read.csv("ComArea_ACS14_f.csv")
comarea_init <- comarea_init[order(comarea_init$ComAreaID.N.18.0),]

comareaCleaned$PerCInc14 <- comarea_init$PerCInc14.N.18.0

# making Pov14P a percentage
comareaCleaned$Pov14P <- comarea_init$Pov14.N.18.0 /
comarea_init$Pop2014.N.18.0 * 100



# I was thinking to make variables indicating ranges like for age (e.g.
Bw18and21P)
# for poverty (Pov50P to Pov200P). However, I'll just leave poverty variables
as it is.
# Shouldn't make a difference.



# Make ChldPov to Unemp into percentages

vars_to_divide <- c("ChldPov14", "NoHS14", "HSGrad14", "SmClg14",
"ClgGrad14",
                    "LaborFrc", "Unemp14")
```

```
comareaCleaned[, which(names(comareaCleaned) %in% vars_to_divide)] <-
  comareaCleaned[, which(names(comareaCleaned) %in% vars_to_divide)] /
comarea_init$Pop2014.N.18.0 * 100

# names(comareaCleaned[, which(names(comareaCleaned) %in% vars_to_divide)])
<-
#   c("ChldPov14P", "NoHS14P", "HSGrad14P", "SmClg14P", "ClgGrad14P",
#     "LaborFrcP", "Unemp14P")

comareaCleaned <- rename(comareaCleaned, ChldPov14P = ChldPov14, NoHS14P =
NoHS14,
       HSGrad14P = HSGrad14, SmClg14P = SmClg14, ClgGrad14P = ClgGrad14,
       LaborFrcP = LaborFrc, Unemp14P = Unemp14)



# rename Perc to Dec for childhood blood levels

comareaCleaned <- rename(comareaCleaned, ChlLeadDec = ChlLeadPerc)



# make sure to exclude ComAreaID and community,
# as well as Under18P, Over18P, Over21P, and Over65P. Just keep Under5P among
the ages
# drop ChlLeadP as well

drop_col_names <- c("ComAreaID", "community", "PopChng", "Under18P",
                    "Over18P", "Over21P", "Over65P", "ChlLeadP")

comareaCleaned <- comareaCleaned[, !(names(comareaCleaned) %in%
drop_col_names)]



# reorder column names

# desired order

# for ease of copy/paste
paste(names(comareaCleaned), collapse = "', '")

colname_order <- c('Birth.Rate', 'General.Fertility.Rate',
'Low.Birth.Weight',
                   'Prenatal.Care.Beginning.in.First.Trimester',
'Preterm.Births',
                   'Teen.Birth.Rate', 'Assault..Homicide.',
'Firearm.related',
                   'Infant.Mortality.Rate', 'Crowded.Housing', 'Dependency',
```

```
                    'PopMP', 'Under5P', 'Bw5and18P', 'Bw18and21P',
'Bw21and65P',
                    'Wht14P', 'Blk14P', 'AI14P', 'AS14P',
                    'NHP14P', 'Oth14P', 'Hisp14P', 'PropCrRt', 'VlntCrRt',
                    'ChldPov14P', 'NoHS14P', 'HSGrad14P', 'SmClg14P',
'ClgGrad14P',
                    'LaborFrcP', 'Unemp14P', 'Pov14P', 'Pov50P', 'Pov125P',
'Pov150P',
                    'Pov185P', 'Pov200P', 'COIave', 'HISave', 'SESave',
                    'Hlitave', 'CentroidLat', 'CentroidLon', 'Dist2Loop',
                    'PerCInc14', 'ChlLeadDec', 'ChlLeadPois')

comareaCleaned <- comareaCleaned[colname_order]

# save it as comarea, instead of comareaCleaned

comarea <- comareaCleaned

save(comarea, file = "comarea.RData")



com_popl <- comarea_init$Pop2014.N.18.0

save(com_popl, file = "com_popl.R")
```

```
######################## Spatial Data Extraction ########################



library(sf) # to read in shapefile
library(geosphere) # to calculate centroids

# Can you get the points of the perimeter of each community area?

com_boundary_init <- st_read("comarea/ComArea_ACS14_f.shp")

# sort com_boundary by ComAreaID

com_boundary_order <- com_boundary_init[order(com_boundary_init$ComAreaID),]

# compare with comarea. Looks like these arbitrarily selected columns match
up.

# all.equal(comarea$shape_area.N.33.31, com_boundary_order$shape_area)
# all.equal(comarea$Over21P.N.18.4, com_boundary_order$Over21P)
# all.equal(comarea$Blk14.N.18.0, com_boundary_order$Blk14)
# all.equal(comarea$Hlitave.N.18.4, com_boundary_order$Hlitave)
# all.equal(comarea$Tuberculosis, com_boundary_order$Tuberc)

# grab the column names of com_boundary_order, to use for comarea
clean_names <- names(com_boundary_order)[-87]

# save(clean_names, file = "clean_names.RData")



# now, delete the columns already in comarea--don't need those (except for
ComAreaID and community name).

com_boundary_omit <- com_boundary_order[, -c(6:86)]

com_boundary <- com_boundary_omit

num_com <- nrow(com_boundary)



# now, to find adjacencies.

# adjacency algorithm:

# for a community, go through all the communities with a higher ComAreaID
# if they share at least one coordinate, call them adjacent.
```

```
com_coords <- vector("list", num_com)

# list of coordinates for each community area

cb_geom <- com_boundary$geometry

for (i in 1:num_com) {

    com_coords[[i]] <- cb_geom[[i]][[1]][[1]]

    # there's one community area, Norwood Park, that has two inner lists
    # cb_geom[[10]][[1]][[1]] and cb_geom[[10]][[1]][[2]]. I ignore
    # the second one, because that's the hole inside the outer perimeter.

    # As it turns out, O'Hare has two tiny regions in addition to one big
region,
    # cb_geom[[10]][[2]][[1]] and cb_geom[[10]][[3]][[1]]. They're
negligible, so I ignore those.

}



# approach: paste the long lat coordinates together.

com_coords_paste <- vector("list", num_com)

for (i in 1:num_com) {

  com_coords_paste[[i]] <- apply(com_coords[[i]], 1, paste, collapse=" ")

}



# returns 0 if two communities aren't adjacent, 1 if they are
adjacent_com <- function(com1, com2, coords_paste) {

  shared_coords <- intersect(com_coords_paste[[com1]],
com_coords_paste[[com2]])

  if (length(shared_coords) == 0) {
    return(0)
  } else if (length(shared_coords) > 0) {
    return(1)
  } else {
    return(NA)
  }

}
```

```
com_adj_mat <- matrix(0, nrow = num_com, ncol = num_com)

for (i in 1:(num_com - 1)) {

  for (j in (i + 1):num_com) {

    com_adj_mat[i, j] <- adjacent_com(i, j, com_coords_paste)

  }

}

com_adj_mat_half <- com_adj_mat[upper.tri(com_adj_mat)]
com_adj_mat <- t(com_adj_mat)
com_adj_mat[upper.tri(com_adj_mat)] <- com_adj_mat_half

save(com_adj_mat, file = "com_adj_mat.RData")



# checking the adjacency matrix

comarea_names <- com_boundary$community

# adjacency is good for Roger's Park
comarea_names[which(com_adj_mat[1,] == 1)]

# adjacency is good for O'Hare
comarea_names[which(com_adj_mat[76,] == 1)]

# adjacency is good for East Garfield Park
comarea_names[which(com_adj_mat[27,] == 1)]

# adjacency is good for Uptown
comarea_names[which(com_adj_mat[3,] == 1)]





# Can you get the centroids of the community areas?

centroid(com_coords[[72]])
```

```
CentroidLat <- rep(NA, num_com)

CentroidLon <- rep(NA, num_com)



for (i in 1:num_com) {

  cent <- centroid(com_coords[[i]])

  CentroidLat[i] <- cent[2]

  CentroidLon[i] <- cent[1]

}



# Find the distance from the Loop (Chicago's central business district) to
each centroid

Dist2Loop <- rep(NA, num_com)

loop_cent <- c(CentroidLat[32], CentroidLon[32])

for (i in 1:num_com) {

  Dist2Loop[i] <- sqrt(sum((c(CentroidLat[i], CentroidLon[i]) -
loop_cent)^2))

}



# 3 new variables to merge in:
#' CentroidLat
#' CentroidLon
#' Dist2Loop
```

```
######################## Count Models ########################



library(rjags)

library(AER)



load("comarea.RData")

load("com_popl.R")


#####

# Exploring overdispersion in the data using a hypothesis test

poisson_glm <- glm(ChlLeadPois ~ ., data = comarea[, names(comarea) !=
"ChlLeadDec"], family = poisson)

dispersiontest(poisson_glm)

#####


Y = comarea$ChlLeadPois

X <- cbind(1, comarea[, !(names(comarea) %in% c("ChlLeadDec",
"ChlLeadPois"))])

n = nrow(X)

p = ncol(X)



# making the Poisson model

dat <- list(Y = Y, X = X,
            N = com_popl, n = n, p = p)

model_str_poi <- textConnection("model {

                                # Likelihood
                                for (i in 1:n) {
                                Y[i] ~ dpois(lambda[i] * N[i])
                                log(lambda[i]) <- inprod(X[i,], beta[])
```

```
                                }

                                # Priors
                                for (j in 1:p) {
                                beta[j] ~ dnorm(0, 0.01)
                                }

                                }")

params <- c("beta")

n_burnin <- 100000

n_iter <- 1000000

# inits <- list(list(beta = rnorm(p, 0, 100)),
#               list(beta = rnorm(p, 0, 100)),
#               list(beta = rnorm(p, 0, 100)),
#               list(beta = rnorm(p, 0, 100)))

inits <- list(list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 819),
              list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 820),
              list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 821),
              list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 822))

model_poi <- jags.model(model_str_poi, data = dat, n.chains = 4, n.adapt =
2000,
                        inits = inits)

update(model_poi, n_burnin)

samples_poi <- coda.samples(model_poi, variable.names = params, n.iter =
n_iter)

save(samples_poi, file = "alvin_model_output/samples_poi.RData")

(ess_poi <- effectiveSize(samples_poi))

(gr_poi <- gelman.diag(samples_poi))

summary_poi <- summary(samples_poi)

round(summary_poi$quantiles, 4)

# plot(samples_poi)

(DIC_pois <- dic.samples(model_poi, n.iter = 20000))

# Mean deviance:  3875
# penalty 45.95
```

```
# Penalized deviance: 3920

save(DIC_pois, file = "alvin_model_output/DIC_pois.RData")



# making the negative binomial model

dat <- list(Y = Y, X = X,
            N = com_popl, n = n, p = p)

model_str_nb <- textConnection("model {

                                # Likelihood
                                for (i in 1:n) {
                                Y[i] ~ dnegbin(q[i], N[i] * m)
                                q[i] <- m / (m + lambda[i])
                                log(lambda[i]) <- inprod(X[i,], beta[])
                                }

                                # Priors
                                for (j in 1:p) {
                                beta[j] ~ dnorm(0, 0.01)
                                }
                                m ~ dgamma(0.1, 0.1)

                                }")

n_burnin <- 100000

n_iter <- 1000000

inits <- list(list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 926),
              list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 927),
              list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 928),
              list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 929))

model_nb <- jags.model(model_str_nb, data = dat, n.chains = 4, n.adapt =
2000,
                       inits = inits)

update(model_nb, n_burnin)

params <- c("beta", "m")

samples_nb <- coda.samples(model_nb, variable.names = params, n.iter =
n_iter)

save(samples_nb, file = "alvin_model_output/samples_nb.RData")
```

```
(ess_nb <- effectiveSize(samples_nb))

(gr_nb <- gelman.diag(samples_nb))

summary_nb <- summary(samples_nb)

round(summary_nb$quantiles, 4)



(DIC_nb <- dic.samples(model_nb, n.iter = 20000))

# Mean deviance:  984.1
# penalty 382.6
# Penalized deviance: 1367

save(DIC_nb, file = "alvin_model_output/DIC_nb.RData")
```

```
######################## Beta Models and Beta SSVS ########################


####Modeling####
library(rjags)

Y = comarea$ChlLeadDec
#Transform the data (Given in Smithson M, Verkuilen J (2006))
Y = (Y*(length(Y)-1) + 0.5) / length(Y)

X = comarea[,c(1:46)]
X = cbind(1, scale(X))

n_burnin <- 20000
n_iter <- 100000
####1####
#Full Beta Regression (Logit link)

dta = list(Y = Y, X = X, p = ncol(X), n = length(Y))
params = c('beta', "r")
model_str <- textConnection("model{
for(i in 1:n){
Y[i] ~ dbeta(r*mu[i],r*(1-mu[i]))
logit(mu[i]) <- inprod(X[i,],beta[])
}
for(j in 1:p){beta[j] ~ dnorm(0,0.01)}
r ~ dgamma(0.1,0.1)
}")

model_logit <- jags.model(model_str,data = dta, n.chains=4, n.adapt = 2000)
update(model_logit, n_burnin)
samples_logit <- coda.samples(model_logit, variable.names=params,
n.iter=n_iter)

DIC.logit = dic.samples(model_logit, n.iter = 20000)

#Save the data
save(samples_logit, file = 'samples_betaLogit.Rdata')
save(DIC.logit, file = "DIC_logit.Rdata")

########################################
#Full Beta Regression (Probit link)

dta = list(Y = Y, X = X, p = ncol(X), n = length(Y))
params = c('beta', "r")
model_str <- textConnection("model{
for(i in 1:n){
Y[i] ~ dbeta(r*mu[i],r*(1-mu[i]))
probit(mu[i]) <- inprod(X[i,],beta[])
}
```

```
for(j in 1:p){beta[j] ~ dnorm(0,0.01)}
r ~ dgamma(0.1,0.1)
}")

model_probit <- jags.model(model_str,data = dta, n.chains=4)
update(model_probit, n_burnin)
samples_probit <- coda.samples(model_probit, variable.names=params, n.iter =
n_iter)

DIC.probit= dic.samples(model_probit, n.iter = 20000)

save(samples_probit, file = "samples_betaProbit.Rdata")
save(DIC.probit, file = 'DIC_probit.RData')


####2####
#Spike and Slab Regression

##SSVS with logit link
params = c("gamma", "beta", "r", "tau2")
model_str <- textConnection("model{
for(i in 1:n){
Y[i] ~ dbeta(r*mu[i],r*(1-mu[i]))
probit(mu[i]) <- inprod(X[i,],beta[])
}
beta[1] ~ dnorm(0,0.01)
for(j in 2:p){
beta[j]  = delta[j]*gamma[j]
gamma[j] ~ dbern(0.5)
delta[j] ~ dnorm(0,tau2)
}
r ~ dgamma(0.1,0.1)
tau2 ~ dgamma(0.1,0.1)
}")

model_ssvs <- jags.model(model_str,data = dta, n.chains=4)
update(model_ssvs, n_burnin)
samples_ssvs <- coda.samples(model_ssvs, variable.names=params, n.iter =
n_iter)

DIC.ssvs= dic.samples(model_ssvs, n.iter = 20000)


save(samples_ssvs, file = "samples_bSsvsProbit.Rdata")
save(DIC.ssvs, file = 'DIC_ssvsProbit.RData')

#Determine which vars to keep
full_params = rbind(samples_ssvs[[1]],
samples_ssvs[[2]],samples_ssvs[[3]],samples_ssvs[[4]])
gamma = full_params[,48:93]
```

```
model <- ""
for(j in 1:46){
  temp  <- ifelse(gamma[,j]==1,colnames(X)[j],"")
  model <- paste(model," ",temp)
}

most_common_model <- which.max(table(model))
colnames(gamma)   <-colnames(X[,-1])
marg_inc_probs    <- colMeans(gamma)

round(marg_inc_probs,2)

marg_inc_probs[round(marg_inc_probs,2) >= 0.4]


#Get the coefs of included variables
round(coefs,2)[c(2,3,5,8,18,19,38,39),]
#get the 95% interval
```

```
######################## Negative Binomial SSVS ########################


Y = comarea$ChlLeadPois
X = comarea[,c(1:46)]
X = cbind(1, scale(X))
n = nrow(X)
p = ncol(X)
N = comarea_full$Pop2014.N.18.0

n_burnin <- 20000
n_iter <- 100000
#Make the data
dat <- list(Y = Y, X = X,
            N = N, n = n, p = p)

model_str = textConnection('model{

#Likelihood
# Likelihood
for (i in 1:n) {
Y[i] ~ dnegbin(q[i], N[i] * m)
q[i] <- m / (m + lambda[i])
log(lambda[i]) <- inprod(X[i,], beta[])
}

#Ssvs Priors
beta[1] ~ dnorm(0,0.01)

for(j in 2:p){
beta[j]  = delta[j]*gamma[j]
gamma[j] ~ dbern(0.5)
delta[j] ~ dnorm(0,tau2)
}
m ~ dgamma(0.1, 0.1)
tau2 ~ dgamma(0.1,0.1)
}')

params <- c("gamma","beta", "m", 'tau2')

model_ssvs <- jags.model(model_str,data = dat, n.chains=4)
update(model_ssvs, n_burnin)
samples_ssvs <- coda.samples(model_ssvs, variable.names=params, n.iter =
n_iter)

save(samples_ssvs, file = "samples_nbSsvs2.Rdata")
save(DIC.ssvs, file = "DIC_nbssvs.Rdata")
DIC.ssvs= dic.samples(model_ssvs, n.iter = 20000)
```

```
(ess_nbSsvs <- effectiveSize(samples_ssvs))
gelman.diag(samples_ssvs)

save(samples_ssvs, file = "samples_nbSsvs2.Rdata")
```

```
full_params = rbind(samples_ssvs[[1]],
samples_ssvs[[2]],samples_ssvs[[3]],samples_ssvs[[4]])
gamma = full_params[,48:93]
model <- ""
for(j in 1:46){
  temp  <- ifelse(gamma[,j]==1,colnames(X)[j],"")
  model <- paste(model," ",temp)
}

most_common_model <- which.max(table(model))
colnames(gamma)   <-colnames(X[,-1])
marg_inc_probs    <- colMeans(gamma)


round(marg_inc_probs,2)

round(marg_inc_probs[round(marg_inc_probs,2) >= 0.5], 2)

#Get the coefs of included variables
round(coefs,2)[c(2,3,5,8,18,19,38,39),]
#get the 95% interval
```

```
######################## Spatial Spike and Slab ########################


library(rjags)

# Spatial Spike and Slab Hierarchical model

load("comarea.RData")

load("com_popl.R")

load("com_adj_mat.RData")


Y = comarea$ChlLeadPois

# I'll account for the intercept as beta0

X <- scale(comarea[, !(names(comarea) %in% c("ChlLeadDec", "ChlLeadPois"))])

# to make presentation of variable selection results easier

covnames <- colnames(X)

n = nrow(X)

p = ncol(X)

# adjacency matrix, with Aij = 0 if i = j or i and j aren't neighbors. Aij =
1 otherwise.

A <- com_adj_mat

# diagonal matrix with number of neighbors down the diagonal

M <- diag(rowSums(com_adj_mat))

dat <- list(Y = Y, X = X, N = com_popl, n = n, p = p, A = A, M = M)


# making the spatial spike and slab model

cat("model {

# Likelihood
for (i in 1:n) {
  Y[i] ~ dnegbin(q[i], N[i] * m)
```

```
  q[i] <- m / (m + lambda[i])
  log(lambda[i]) <- beta0 + inprod(X[i,], B[i,])
}
# Likelihood prior
m ~ dgamma(0.1, 0.1)

# Spike and Slab
beta0 ~ dnorm(0, 0.1)
for (j in 1:p) {
  for (i in 1:n) {
    B[i, j] <- gamma[i, j] * delta[j]
    gamma[i, j] ~ dbern(prob[i])
  }
  delta[j] ~ dnorm(0, tau)
}
for (i in 1:n) {
  prob[i] <- pnorm(l[i], 0, 1)
}
# Slab prior
tau ~ dgamma(0.1, 0.1)

# Latent Spatial Variable
l[1:n] ~ dmnorm(mu[], tau2 * Sigma_inv[,])
Sigma_inv <- M - rho * A
# Latent Spatial prior
for (i in 1:n) {
  mu[i] <- 0
}
tau2 ~ dgamma(0.1, 0.1)
rho ~ dbeta(1, 1)
                                }", file="model_spatial_spike_slab.txt")

params <- c("beta0", "delta", "prob", "gamma", "rho")

n_burnin <- 1000000

n_iter <- 10000

inits <- list(list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 1120),
              list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 1121))

model_spatial_ssvs <- jags.model("model_spatial_spike_slab.txt", data = dat,
n.chains = 2, n.adapt = 2000,
                       inits = inits)

update(model_spatial_ssvs, n_burnin)

samples_spatial_ssvs <- coda.samples(model_spatial_ssvs, variable.names =
params, n.iter = n_iter)
```

```
save(samples_spatial_ssvs, file =
"alvin_model_output/samples_spatial_ssvs.RData")

ess_spatial_ssvs <- effectiveSize(samples_spatial_ssvs)

gr_spatial_ssvs <- gelman.diag(samples_spatial_ssvs)

save(ess_spatial_ssvs, gr_spatial_ssvs, file =
"alvin_model_output/convergence_diagnostics_spatial_ssvs.RData")

summary_spatial_ssvs <- summary(samples_spatial_ssvs)

save(summary_spatial_ssvs, file =
"alvin_model_output/summary_spatial_ssvs.RData")



(DIC_spatial_ssvs <- dic.samples(model_spatial_ssvs, n.iter = 20000))

# put results here

save(DIC_spatial_ssvs, file = "alvin_model_output/DIC_spatial_ssvs.RData")
```

```
######################## Running Spatial SSVS ########################



# ran code in spatial_spike_slab_negbin.R for
# burn in 1000
# iterations 100

# I keep track of the effective sample sizes
summary(ess_spatial_ssvs)

# summary(ess_spatial_ssvs)
# Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
# 3.447  196.483  200.000  209.264  210.683 1172.358

# Consider saving the model
# save(model_spatial_ssvs, file = "model_spatial_ssvs.RData")

# 100 more iterations
samples_spatial_ssvs2 <- coda.samples(model_spatial_ssvs, variable.names =
params, n.iter = n_iter)

# 1000 more iterations
samples_spatial_ssvs3 <- coda.samples(model_spatial_ssvs, variable.names =
params, n.iter = 1000)

ess_spatial_ssvs3 <- effectiveSize(samples_spatial_ssvs3)

summary(ess_spatial_ssvs3)

# > summary(ess_spatial_ssvs3)
# Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
# 4.151 1861.558 2000.000 1883.060 2000.000 3382.907

# gr_spatial_ssvs3 <- gelman.diag(samples_spatial_ssvs3)

summary_spatial_ssvs3 <- summary(samples_spatial_ssvs3)



# I've run 1200 iterations so far. I'll update for 8800 more iterations as
burn in,
# for a total of 10000 burn in

update(model_spatial_ssvs, 8800)

# 1000 more iterations
samples_spatial_ssvs4 <- coda.samples(model_spatial_ssvs, variable.names =
params, n.iter = 1000)
```

```
ess_spatial_ssvs4 <- effectiveSize(samples_spatial_ssvs4)

# > summary(ess_spatial_ssvs4)
# Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
# 4.144 1821.874 2000.000 1857.706 2000.000 2972.865



# I've run 11000 iterations so far. I'll update for 9000 more iterations as
burn in,
# for a total of 20000 burn in

update(model_spatial_ssvs, 9000)

# 1000 more iterations
samples_spatial_ssvs5 <- coda.samples(model_spatial_ssvs, variable.names =
params, n.iter = 1000)

ess_spatial_ssvs5 <- effectiveSize(samples_spatial_ssvs5)

# > summary(ess_spatial_ssvs5)
# Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
# 2.993 1738.257 1909.519 1804.122 2000.000 3448.258

save(samples_spatial_ssvs5, file =
"alvin_model_output/samples_spatial_ssvs5.RData")

save(model_spatial_ssvs, file = "model_spatial_ssvs1.RData")

# tested changing from pnorm to probit. No change to results.

# get the model back

load("model_spatial_ssvs1.RData")

params <- c("beta0", "delta", "prob", "gamma", "rho")

model_spatial_ssvs$recompile()

# I've run 21000 iterations so far. I'll update for 9000 more iterations as
burn in,
# for a total of 30000 burn in

update(model_spatial_ssvs, 9000)

# 10000 more iterations. I guess it's time to roll with it, and use the
results I do have, somehow.
samples_spatial_ssvs6 <- coda.samples(model_spatial_ssvs, variable.names =
params, n.iter = 10000)
```

```
ess_spatial_ssvs6 <- effectiveSize(samples_spatial_ssvs6)

# The ess is slowly getting kind of better, at least
# > summary(ess_spatial_ssvs6)
# Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
# 21.07 16115.33 18396.65 17105.12 19485.66 21114.64

save(samples_spatial_ssvs6, file =
"alvin_model_output/samples_spatial_ssvs6.RData")

save(model_spatial_ssvs, file = "model_spatial_ssvs2.RData")



load("model_spatial_ssvs2.RData")

params <- c("beta0", "delta", "prob", "gamma", "rho")

model_spatial_ssvs$recompile()

# test <- coda.samples(model_spatial_ssvs, variable.names = params, n.iter =
100)



# I've run 40000 iterations so far.

# sample 100000 posterior samples.
samples_spatial_ssvs7 <- coda.samples(model_spatial_ssvs, variable.names =
params, n.iter = 100000)

save(samples_spatial_ssvs7, file =
"alvin_model_output/samples_spatial_ssvs7.RData")

ess_spatial_ssvs7 <- effectiveSize(samples_spatial_ssvs7)

# > summary(ess_spatial_ssvs7)
# Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
# 22.91  6558.51  9711.00 10011.31 13089.44 24259.59

gr_spatial_ssvs7 <- gelman.diag(samples_spatial_ssvs7)

save(ess_spatial_ssvs7, gr_spatial_ssvs7, file =
"alvin_model_output/convergence_diagnostics_spatial_ssvs7.RData")

save(model_spatial_ssvs, file = "model_spatial_ssvs3.RData")

summary_spatial_ssvs7 <- summary(samples_spatial_ssvs7)
```

```
save(summary_spatial_ssvs7, file =
"alvin_model_output/summary_spatial_ssvs7.RData")
```

```
# sample 1000000 posterior samples, with thinning.
samples_spatial_ssvs8 <- coda.samples(model_spatial_ssvs, variable.names =
params, n.iter = 1000000, thin = 10)

save(samples_spatial_ssvs8, file =
"alvin_model_output/samples_spatial_ssvs8.RData")

save(model_spatial_ssvs, file = "model_spatial_ssvs4.RData")

ess_spatial_ssvs8 <- effectiveSize(samples_spatial_ssvs8)

gr_spatial_ssvs8 <- gelman.diag(samples_spatial_ssvs8)

save(ess_spatial_ssvs8, gr_spatial_ssvs8, file =
"alvin_model_output/convergence_diagnostics_spatial_ssvs8.RData")

summary_spatial_ssvs8 <- summary(samples_spatial_ssvs8)

save(summary_spatial_ssvs8, file =
"alvin_model_output/summary_spatial_ssvs8.RData")

# > summary(ess_spatial_ssvs8)
# Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
# 50.92   835.02  1165.05  1690.34  1969.15 29048.22
```

```
load("model_spatial_ssvs4.RData")

params <- c("beta0", "delta", "prob", "rho")

model_spatial_ssvs$recompile()

# sample 1000000 posterior samples, with no thinning. This time, don't track
gamma, so you can go for more iterations
samples_spatial_ssvs9 <- coda.samples(model_spatial_ssvs, variable.names =
params, n.iter = 1000000)

save(samples_spatial_ssvs9, file =
"alvin_model_output/samples_spatial_ssvs9.RData")
```

```
save(model_spatial_ssvs, file = "model_spatial_ssvs5.RData")

ess_spatial_ssvs9 <- effectiveSize(samples_spatial_ssvs9)

gr_spatial_ssvs9 <- gelman.diag(samples_spatial_ssvs9)

save(ess_spatial_ssvs9, gr_spatial_ssvs9, file =
"alvin_model_output/convergence_diagnostics_spatial_ssvs9.RData")

summary_spatial_ssvs9 <- summary(samples_spatial_ssvs9)

save(summary_spatial_ssvs9, file =
"alvin_model_output/summary_spatial_ssvs9.RData")



(DIC_spatial_ssvs <- dic.samples(model_spatial_ssvs, n.iter = 20000))

# Mean deviance:  795.7
# penalty 172.8
# Penalized deviance: 968.5

save(DIC_spatial_ssvs, file = "alvin_model_output/DIC_spatial_ssvs.RData")
```

```
######################## Exploring Spatial SSVS Results ########################


# Gamma results

# Convergence Diagnostics

load("alvin_model_output/convergence_diagnostics_spatial_ssvs8.RData")

summary(ess_spatial_ssvs8)

ess_spatial_ssvs8[ess_spatial_ssvs8 < 150]



psrf_upper_limit8 <- gr_spatial_ssvs8$psrf[, 2]

summary(psrf_upper_limit8)




# Model Results

load("summary_spatial_ssvs8.RData")

param_names8 <- rownames(summary_spatial_ssvs8$statistics)


post_means8 <- summary_spatial_ssvs8$statistics[, 1]

gamma_post_means <- post_means8[startsWith(param_names8, "gamma")]


gamma_post_means_mat <- matrix(gamma_post_means, nrow = 77, ncol = 46)


load("alvin_model_output/samples_spatial_ssvs8.RData")

samples_total <- rbind(samples_spatial_ssvs8[[1]],
samples_spatial_ssvs8[[2]])
```

```
gamma_idx <- param_names8[startsWith(param_names8, "gamma")]
mips <- apply(samples_total[, gamma_idx], MARGIN = 2, mean)
mips_mat <- matrix(mips, nrow = 77, ncol = 46)

# Results for the other parameters
# Convergence Diagnostics
load("alvin_model_output/convergence_diagnostics_spatial_ssvs9.RData")
summary(ess_spatial_ssvs9)
ess_spatial_ssvs9[ess_spatial_ssvs9 < 200]
psrf_upper_limit9 <- gr_spatial_ssvs9$psrf[, 2]
summary(psrf_upper_limit9)
ess_spatial_ssvs9[ess_spatial_ssvs9 < 500]

# Model Results
load("summary_spatial_ssvs9.RData")
param_names9 <- rownames(summary_spatial_ssvs9$statistics)

post_means9 <- summary_spatial_ssvs9$statistics[, 1]
beta0_post_means <- post_means9[param_names9 == "beta0"]
delta_post_means <- post_means9[startsWith(param_names9, "delta")]
prob_post_means <- post_means9[startsWith(param_names9, "prob")]
rho_post_mean <- post_means9[param_names9 == "rho"]

# rho posterior mean and sd
summary_spatial_ssvs9$statistics[param_names9 == "rho", ]
```

```
summary_spatial_ssvs9$statistics[startsWith(param_names9, "prob"),]

max(summary_spatial_ssvs9$statistics[startsWith(param_names9, "prob"), 4])


mips_mat[41, ]

which(mips_mat[41, ] > .7)
```

```
######################## Plotting Maps ########################



library(sf) # to read in shapefile
library(ggplot2)

com_boundary_init <- st_read("comarea/ComArea_ACS14_f.shp")

com_boundary_order <- com_boundary_init[order(com_boundary_init$ComAreaID),]

cb_geom <- com_boundary_order$geometry

# just plots the boundaries
plot(cb_geom)


plot(st_geometry(cb_geom), col = sf.colors(12, categorical = TRUE), border =
'dimgray',
     axes = TRUE)

plot(st_geometry(st_centroid(cb_geom)), pch = 3, col = 'black', add = TRUE)




library(rgdal)
library(ggplot2)
library(dplyr)



shp <- readOGR(dsn = "comarea/ComArea_ACS14_f.shp")



# map <- ggplot() + geom_polygon(data = shp, aes(x = long, y = lat, group =
group), colour = "black", fill = NA)
#
# map + theme_void()



# order the shp@data

shp@data$ComAreaID <- as.numeric(as.character(shp@data$ComAreaID))

shp@data <- shp@data[order(shp@data$ComAreaID),]
```

```
shp@data$ComAreaID <- as.character(shp@data$ComAreaID)



shp@data <- shp@data %>% mutate(probs = prob_post_means)

shp_df <- broom::tidy(shp, region = "ComAreaID")
shp_df <- shp_df %>% left_join(shp@data, by = c("id" = "ComAreaID"))

map <- ggplot() + geom_polygon(data = shp_df, aes(x = long, y = lat, group =
group, fill = probs), colour = "black") + theme_void()
map + scale_fill_continuous(name="Prior Probabilities\nof Inclusion")




shp@data <- shp@data %>% mutate(probs = prob_post_means, mips_prenatal =
mips_mat[, 4])

shp_df <- broom::tidy(shp, region = "ComAreaID")
shp_df <- shp_df %>% left_join(shp@data, by = c("id" = "ComAreaID"))

map <- ggplot() + geom_polygon(data = shp_df, aes(x = long, y = lat, group =
group, fill = mips_prenatal), colour = "black") + theme_void()
map + scale_fill_continuous(name="MIP for\n1st Trimester \n Prenatal Care")



shp@data <- shp@data %>% mutate(probs = prob_post_means, mips_poverty =
mips_mat[, 38])

shp_df <- broom::tidy(shp, region = "ComAreaID")
shp_df <- shp_df %>% left_join(shp@data, by = c("id" = "ComAreaID"))

map <- ggplot() + geom_polygon(data = shp_df, aes(x = long, y = lat, group =
group, fill = mips_poverty), colour = "black") + theme_void()
map + scale_fill_continuous(name="MIP for\nBelow 200% \n Poverty Line")
```