Modeling and Predicting Crime Rates in Chicago Using Bayesian Model

Colin Peng, Yonnie Chan, Youngmin Kim
University of Michigan

1. Problem statement

Crime rate prediction is one of the famous, interesting, and useful data science research topics. Using the crime rate prediction, we can allocate law enforcement and securities resources more efficiently and effectively. Also, it helps to make crime prevention strategies and policies such as installing surveillance cameras or formulating laws and regulations. Moreover, people can be more informed about their safety and make better decisions to make them safer.

Chicago is one of the largest cities which have a large crime rate. According to the Chicago Tribune, approximately 10 people are shot on average per day in Chicago. Also, the crime rate varies much among districts. This makes it more interesting and useful to predict the crime rate in Chicago.

Our goal is to use the Bayesian model to predict the crime rate of each sector in Chicago. We think that the Bayesian model is quite a useful technique for modeling crime rates. One advantage is that we can use historical crime rates as a prior distribution. Also, unlike the complex machine learning model, the Bayesian model is easy to interpret. Because it can be used to make some important decisions, interpretability is one of the most essential factors we should consider.

In particular, we try to answer these two questions with our models:

1. Are certain police districts more dangerous than the others? (in terms of having more crimes or crimes per resident)
2. Are certain months in a year more dangerous than the others?

2. Datasets

From the Chicago Data Portal, we found the dataset that matches our expectations exactly. The data was extracted from the Chicago Police Department's CLEAR(Citizen Law Enforcement Analysis and Reporting) system. The datasets contain all reported incidents of crime (with the exception of murders where data exists for each victim) that occurred in the City of Chicago from 2001 to the Present (updated daily).

To protect the privacy of crime victims, they provided only block-level addresses for location information. Because the data is supplied by the Police department, the preliminary crime classification can be changed after further investigation. Also, there may be human errors. Other than that, we believe that the source of data is reliable and contains sufficient information to achieve our goal.

Because the crime data can not be provided by other than public institutes, we don't have any other choice but to gather the data from the same source, the police department. Also, we think the data is pretty enough to apply the Bayesian model and predict crime rate. Therefore, we don't plan to gather additional data, but if it is needed while making the model and analysis, we will try to gather more data.

2.1. Data structure

The dataset contains 7,921,096 crime records and 22 attributes in .csv format. Here is a summary of the data structure:

Column_name	Data type	Column_name	Data type	Column_name	Data type	Column_name	Data type
id	int	location_description	string	fbi_code	string	primary_type	string
case_number	string	arrest	bool	x_coordinate	float	description	string
date	string	domestic	bool	y_coordinate	float	location	string
block	string	beat	int	year	int	latitude	float
iucr	string	district	float	updated_on	string	longitude	float
community_area	float	ward	float				

Table 1. column information

2.2. Handle missing data

We have 1,691,466 missing values in the whole dataset that are present in location_description, district, ward, community_area, x_coordinate, y_coordinate, latitude, longitude, and location.

Inspecting the features, we see that all the features that have a large count of missing values are features that relate to the geographical location of the crime scene. This is no surprise as the Chicago Crime Dataset is based on first-hand accounts of people involved in or around the crime.

Since most of these features are not direct numeric values, we can't use summary statistical functions to fill in the missing values. Hence, we decided to drop these values from the dataset, and 91.11% of the data has been retained. Also, the crime rate data of districts 3, 4, and 8 is only available for specific periods. Therefore, we eliminated these districts' data for better analysis and focused on the districts where we have complete data.

2.3. Detect anomalies

After identifying invalid data in the location data on the scatter plot, there is a point that should not be located in Chicago. Therefore, we have decided to remove these records.

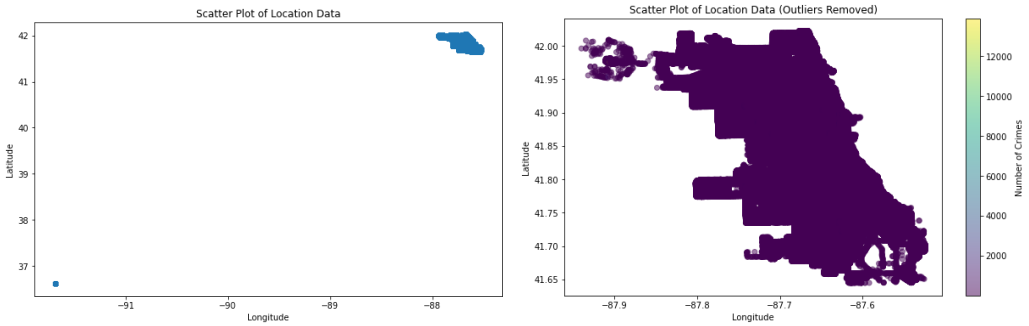

Figure 1. Scatter plot of location data before and after removing anomalies

2.4. Preliminary analysis of the data

- From 2001 to 2023, there has been a consistent decrease in the overall number of crimes.
- Theft and Battery emerged as the most prevalent offenses, with 1,528,215 and 1,322,986 reported cases, respectively.
- Over the past decade (2014-2023), a staggering 83.2% of criminals remained unarrested. Offenses such as burglary, criminal damage, criminal sexual assault, motor vehicle theft, and robbery saw over 90% of cases going unresolved. In contrast, crimes like prostitution, narcotics, gambling, and liquor law violations saw over 90% of cases leading to arrests.
- Crimes were most likely to occur on the street (25.85% of total cases) and in residences (16.37% of total cases). Community Area 25 exhibited the highest crime rate.
- Figure 2 shows that the South side and Southeast side of Chicago exhibit the highest numbers of reported crimes.
- The summer season, particularly in July and August, witnessed a surge in criminal activities. Conversely, winter recorded the lowest incidence of crimes.
- Crime rate slumps at midnight (01:00 - 07:00).

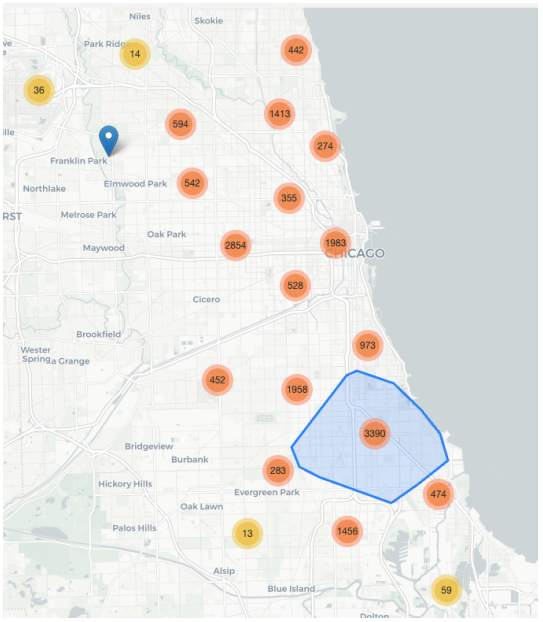

Figure 2. An overview of crime counts in Chicago

3. Approach

To find a police district and a month that are more dangerous than the others. We tried three different models, one linear regression model (Bayesian Poisson regression model) and two hierarchical models (Hierarchical Normal Model, Hierarchical Poisson-gamma model). The regression model is quite widely used to predict crime rate and, because each district or month belongs to a larger group called Chicago city or year, respectively, we thought that a hierarchical model would be helpful to utilize this structure. Also, since we don't know the possible distribution of crime rate, we decided to try two different hierarchical models.

We divided the data into two sets based on time period. We used the data from 2001.01 to 2010.12 to get the prior distribution and data from 2011.01 to 2023.10 were used as training data. We counted the number of crime rates for each month and district and used them as our data. After constructing each model and running MCMC, we compared the result (predicted confidence region of each parameter) to derive the final conclusion.

4. Models

4.1 Bayesian Poisson Regression Model

The Bayesian Poisson regression model allows us to not only estimate the parameters of the model but also quantify the uncertainty associated with these estimates, providing a more comprehensive understanding of the relationships between different features and crime incidence. The resulting posterior distribution from the sampling process enables us to draw inferences about the likely values of the parameters, offering a robust framework for statistical analysis. Moreover, the model's flexibility allows for the incorporation of prior knowledge, with the choice of priors influencing the final estimates. Through this modeling process, we aim to gain valuable insights into the factors influencing crime rates, considering both spatial and temporal dimensions in our analysis.

We conducted a Bayesian Poisson regression model on crime data, selecting the Poisson distribution due to its suitability for modeling count data, which aligns seamlessly with the discrete and non-negative nature of crime incident counts. The model can be expressed as:

$$Y_i \sim Poisson(\lambda_i)$$

$$log(\lambda_i) = \beta_0 + \beta_1 * Covariate_1 + \beta_2 * Covariate_2$$

Where:

Y_i represents the count of crimes in the sector i.

λ_i is the expected rate of crimes in the sector i, which we want to estimate.

β_0 is the intercept term.

β_1 is the coefficient for the demographic factor (district) that influences the crime rate.

β_2 is the coefficient for the temporal factor (month) that influences the crime rate.

The link function in this case is the natural logarithm.

In this model, we have incorporated informative priors to provide additional structure. The priors serve as our initial beliefs about the likely values of the parameters before observing the data. In the model, we set up our priors as follows:

$$\beta_0 \sim Normal(\mu_0, \sigma_0^2)\ ,\ \beta_1 \sim Normal(\mu_1, \sigma_1^2),\ \beta_2 \sim Normal(\mu_2, \sigma_2^2)$$

For setting the prior distributions on the regression coefficients, intercept term (β_0), district-specific effects (β_1), and month-specific effects (β_2), we employed a normal prior with the means μ_0, μ_1, μ_2, ... and variances σ_0^2, σ_1^2, σ_2^2, ... based on the historical data. Prior to observing the current data, we utilized a PoissonRegressor to fit a Poisson regression model on the historical crime data. These priors suggest that, based on historical patterns, we expect these three coefficients to be centered around these mean values, with a reasonable variability.

The model was then trained on the training crime data, and samples were drawn from the posterior distribution using Markov Chain Monte Carlo (MCMC) methods in Python with PyMC3. After obtaining the samples from the posterior distribution, we summarized the parameter estimates and made predictions for crime rates in Chicago sectors.

4.2 Hierarchical Model

Since each district is geographically connected, it can be difficult to assume that the number of crimes in each district is completely independent. Also, the number of crimes each month can be influenced by the overall economic and social situation of the year. In response to this, we constructed and tested a hierarchical model to predict the number of crimes in each district or each month. We implemented and tested both the Hierarchical Normal Model and the Hierarchical Poisson Gamma Model.

4.2.1 Hierarchical Normal Model

The Hierarchical Normal Model is a model framed on the assumption that one or both parameters of the Normal model follow another Normal distribution. We assumed that the number of crimes in each district (or each month) follows a Normal distribution, and at the same time, we formulated a model on the assumption that the mean of the Normal distribution of each district (or each month) follows another Normal distribution. For the simple model configuration, we set a separate hyperparameter for the variance of the Normal model and constructed the model.

As seen in Figure 3., the mean of each District (or month) follows another Normal distribution ($N(\mu, \tau)$), and the number of crimes each month in each District is assumed to follow the Normal distribution($N(\mu_i,\sigma)$. In addition, it was assumed that σ follows a separate Gamma distribution. For the priors of μ, τ, each follows a separate Normal distribution and Gamma distribution respectively. Also, the hyperparameters used values predetermined. In the case of μ, the mean and variance values based on the data from 2001 to 2010 were designated as parameters of the Normal distribution, and in the case of τ, σ, all hyperparameters were set to 1, a weakly informative prior. This can be represented as follows in formulas.

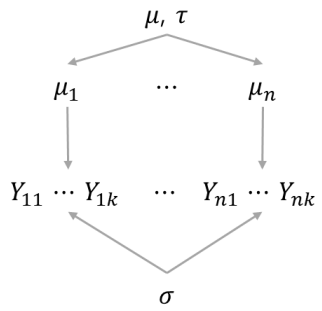

Figure 3. Model structure

- Sampling : $Y_{ij} \mid \mu_i, \sigma \sim \text{Normal}(\mu_i, \sigma)$
- Prior for μ_i : $\mu_i \mid \mu, \tau \sim \text{Normal}(\mu, \tau)$
- Prior for σ : $1/\sigma^2 \sim \text{Gamma}(1,1)$
- Prior for μ : $\mu \mid \mu_0, \tau_0 \sim \text{Normal}(\mu_o, \tau_0)$
- Prior for τ : $1/\tau^2 \sim \text{Gamma}(1,1)$

4.2.2 Hierarchical Poisson-Gamma Model

The Poisson-Gamma Bayesian hierarchical model is a powerful statistical framework that integrates Poisson and Gamma distributions within a hierarchical structure. This approach is particularly useful when analyzing data that exhibit variability beyond what can be explained by a simple Poisson distribution. In this model, the Poisson distribution is employed to capture the count of crime in our data, while the Gamma distribution serves as a prior for the Poisson parameter λ, allowing for the incorporation of different mean and variance for different months or districts. The hierarchical aspect of the model enables the

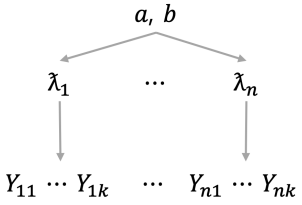

Figure 4. Model structure

sharing of information across different levels of the data hierarchy, providing a flexible and robust framework for Bayesian analysis that accommodates complex and correlated structures within our crime rate data.

■ Sampling : $Y_{ij} \mid \lambda_i \sim \text{Poisson}(\lambda_i)$ ■ Prior for λ_i : $\lambda_i \mid a, b \sim \text{Gamma}(a,b)$

Where i is the number of district for answering question 1, the number of month for answering question 2.

5. Result and discussion

5.1 Bayesian Poisson Regression Model

To infer the posterior distribution of the model parameters, the model is sampled using the No-U-Turn Sampler (NUTS) algorithm with 2500 draws and 500 tuning samples per chain, across four chains. Table 2. shows the mean, standard deviation, and variance of each parameter.

District				**Month**			
	mean	**sd**	**var**		**mean**	**sd**	**var**
beta[1]	-0.257	0.736	0.5417	**beta[1]**	-0.089	1.882	3.5419
beta[2]	-0.762	0.737	0.5432	**beta[2]**	-0.229	1.882	3.5419
beta[5]	0.207	0.737	0.5432	**beta[3]**	-0.068	1.882	3.5419
beta[6]	0.400	0.737	0.5432	**beta[4]**	-0.082	1.882	3.5419
beta[7]	-0.036	0.737	0.5432	**beta[5]**	0.033	1.882	3.5419
beta[9]	-0.131	0.737	0.5432	**beta[6]**	0.042	1.882	3.5419
beta[10]	0.194	0.737	0.5432	**beta[7]**	0.084	1.882	3.5419
beta[11]	-0.267	0.737	0.5432	**beta[8]**	0.070	1.882	3.5419
beta[12]	-0.098	0.736	0.5417	**beta[9]**	-0.003	1.882	3.5419
beta[13]	0.557	0.737	0.5432	**beta[10]**	-0.007	1.882	3.5419
beta[14]	0.248	0.737	0.5432	**beta[11]**	-0.104	1.882	3.5419
beta[15]	0.277	0.737	0.5432	**beta[12]**	-0.122	1.882	3.5419
beta[16]	0.629	0.737	0.5432	**Intercept**			
beta[17]	0.417	0.737	0.5432	**beta0**	6.762	2.367	5.6027
beta[18]	0.303	0.737	0.5432				
beta[19]	0.459	0.737	0.5432				
beta[20]	0.505	0.737	0.5432				
beta[21]	0.214	0.737	0.5432				
beta[22]	0.210	0.737	0.5432				
beta[23]	0.254	0.737	0.5432				
beta[24]	0.152	0.736	0.5417				
beta[25]	0.124	0.737	0.5432				

Table 2. sampling result of each hyperparameter.

From Table 2, we observe notable variations in the mean coefficients for each district and each month, indicating distinct or month relationships and the predicted crime count. However, the variances, which are consistent between different districts and months, didn't capture the degree of uncertainty or variability in these coefficient estimates.

Regarding the district aspect, it is evident that District 2 exhibits the most substantial negative coefficient, suggesting a strong negative relationship with the crime count. Conversely, District 16 boasts the highest positive coefficient, implying a potentially higher predicted crime incident. For the temporal aspect, month, the lowest coefficient is observed in February, suggesting a potential decrease in the expected crime count during this month. Conversely, the highest coefficient is recorded in July, indicating a potential surge in predicted crime incidents. It meets the result we gained from the preliminary analysis.

5.2 Hierarchical Normal Model

Using MCMC (total 16,000 iterations (adapt: 1,000, burn-in: 5,000), we computed the posterior distribution of each parameter. We ran twice, one to get the result of each district and one to get the result of each month. The following table shows the mean, variance, confidence interval, and standard deviation of each parameter.

Sampling result for each district						sampling result for each month					
	Lower95	**Median**	**Upper95**	**Mean**	**SD**		**Lower95**	**Median**	**Upper95**	**Mean**	**SD**
mu	1663	1663	1663	1663	0.0443	**mu**	1663	1663	1663	1663	0.0451
tau	504	675	893	685	103	**tau**	406	609	883	628	129
mu_j[1]	608	646	683	646	19.3	**mu_j[1]**	931	975	1019	975	22.3
mu_j[2]	353	391	429	391	19.4	**mu_j[2]**	804	848	890	848	22.2
mu_j[5]	989	1027	1065	1026	19.4	**mu_j[3]**	953	995	1039	996	22.3
mu_j[6]	1205	1245	1281	1245	19.3	**mu_j[4]**	940	982	1027	982	22.2
mu_j[7]	767	806	843	806	19.4	**mu_j[5]**	1057	1101	1145	1100	22.5
mu_j[9]	695	732	770	732	19.4	**mu_j[6]**	1067	1110	1154	1110	22.3
mu_j[10]	976	1014	1052	1014	19.6	**mu_j[7]**	1115	1158	1201	1158	22.2
mu_j[11]	602	640	677	640	19.4	**mu_j[8]**	1098	1142	1187	1142	22.6
mu_j[12]	718	756	794	756	19.6	**mu_j[9]**	1018	1061	1106	1062	22.3
mu_j[13]	1420	1457	1496	1457	19.4	**mu_j[10]**	1014	1058	1101	1058	22.4
mu_j[14]	1032	1070	1107	1069	19.4	**mu_j[11]**	918	960	1007	960	22.6
mu_j[15]	1063	1100	1139	1101	19.4	**mu_j[12]**	900	944	987	944	22.3
mu_j[16]	1531	1567	1606	1567	19.5	**sigma**	339	348	357	348	4.61
mu_j[17]	1227	1266	1304	1266	19.5						
mu_j[18]	1090	1130	1166	1130	19.3						
mu_j[19]	1281	1321	1358	1321	19.4						
mu_j[20]	1345	1383	1421	1383	19.3						
mu_j[21]	995	1034	1070	1034	19.3						
mu_j[22]	993	1030	1068	1030	19.3						
mu_j[23]	1039	1076	1114	1076	19.3						
mu_j[24]	931	971	1007	971	19.5						
mu_j[25]	907	945	982	945	19.2						
sigma	217	223	229	223	2.96						

Table 3. sampling result of each hyperparameter

Table 3 shows that the mean of each district and each month are different from other districts or months. The district with the lowest predicted monthly average number of crimes(District 2) had a crime count of 25.0% (=391/1567) of the level of the district with the most crimes(District 16). On the other hand, in terms of months, the month with the lowest predicted monthly average number of crimes(February) had a crime count of 73.2% (-848/1158) of the month with the most crimes(July). This difference was relatively small compared to that in districts. Through this, we found that location has a more significant impact on crime than the month does.

We computed the R-value to see whether the majority of total variability came from the differences between groups or within groups. R is computed by $R = \frac{\tau^2}{\tau^2 + \sigma^2}$, and if R is close to 1 it means that the majority of total variability came from the differences among groups. The following graph shows the result.

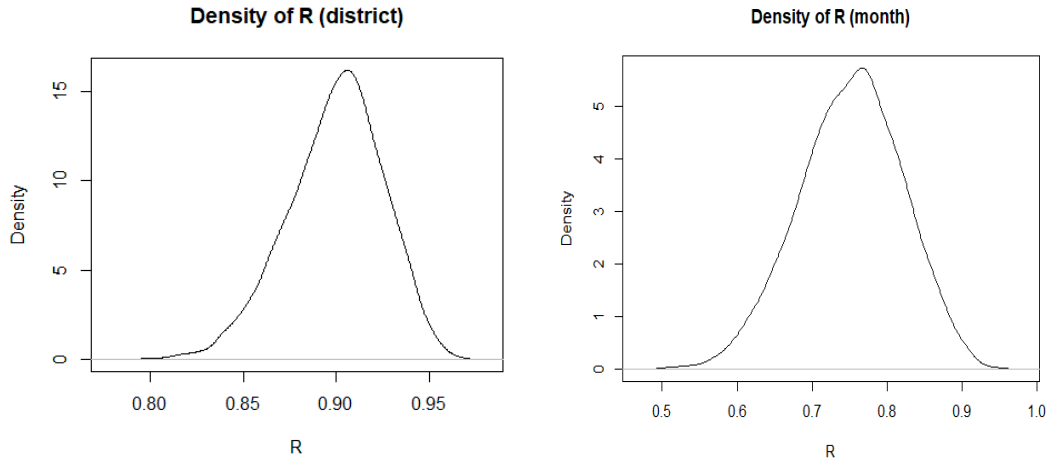

Figure 5. R-value of samples(left: district, right: month)

Figure 5 shows that the R-value is very close to 1 in both cases (district, month) this means that most variability came from between groups. Also, the R-value is greater when modeling districts. Therefore, we discovered that the number of crimes is more affected by district than by month.

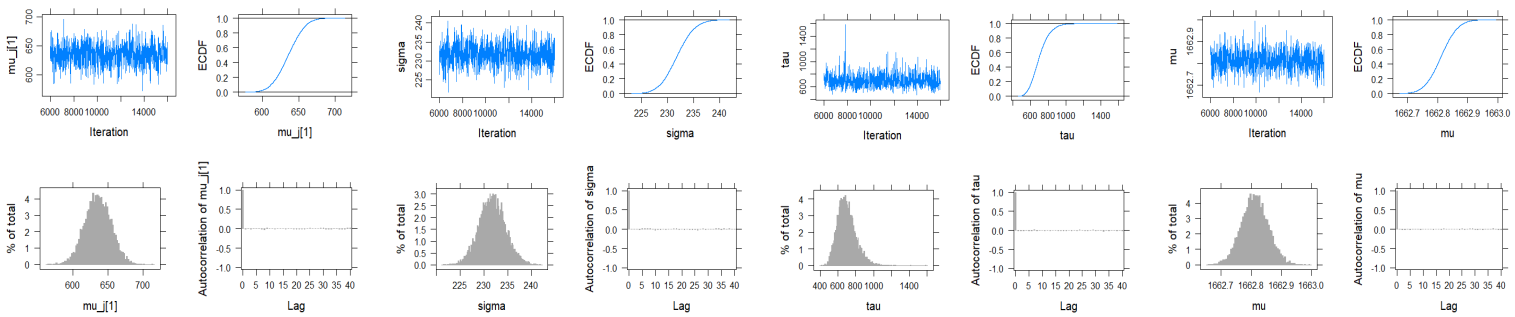

Figure 6. Diagnostic plots of simulated draws of mu_1, sigma, tau, mu

Figure 6 is diagnostic plots of simulated draws of mu_1, sigma, tau, mu. As you can see the value is converged to a certain distribution and looks normally distributed. Also, there is no correlation between time lag. Therefore, we can conclude that our MCMC samples have converged.

5.3 Hierarchical Poisson-Gamma Model

Using MCMC(total 16,000 iterations (adapt: 1,000, burn-in: 5,000), we computed the posterior distribution of each parameter. We ran twice, once using district index as j to get the result of each district and once using the month as j to get the result of each month. Table 4 shows the posterior mean, variance, confidence interval, and standard deviation of each parameter. From the table, we derive similar conclusions that February is the month with the least crime count, July is the month with the most crime count, District 2 is the district with the least crime count, and District 16 is the district with the most crime count.

Using district as j						Using month as j					
	Lower95	**Median**	**Upper95**	**Mean**	**SD**		**Lower95**	**Median**	**Upper95**	**Mean**	**SD**
lambda[1]	631	635	639	635	2	**lambda[1]**	947	951	954	951	1.82
lambda[2]	391	394	397	394	1.59	**lambda[2]**	829	832	836	832	1.69
lambda[5]	1019	1024	1029	1024	2.58	**lambda[3]**	972	976	979	976	1.84
lambda[6]	1204	1210	1215	1210	2.78	**lambda[4]**	960	964	967	964	1.84
lambda[7]	782	786	791	786	2.24	**lambda[5]**	1073	1077	1081	1077	1.96
lambda[9]	712	716	720	716	2.15	**lambda[6]**	1085	1088	1092	1088	1.95
lambda[10]	976	981	986	981	2.52	**lambda[7]**	1134	1138	1142	1138	1.99
lambda[11]	639	643	647	643	2.04	**lambda[8]**	1122	1126	1130	1126	1.99
lambda[12]	745	750	754	750	2.17	**lambda[9]**	1047	1051	1055	1051	1.92
lambda[13]	1418	1424	1430	1424	3.05	**lambda[10]**	1048	1051	1055	1051	1.92
lambda[14]	1055	1060	1065	1060	2.64	**lambda[11]**	889	892	896	892	1.76
lambda[15]	1108	1113	1118	1113	2.68	**lambda[12]**	929	933	936	933	1.88
lambda[16]	1490	1496	1502	1496	3.13						
lambda[17]	1197	1202	1208	1202	2.79						
lambda[18]	1103	1109	1114	1109	2.66						
lambda[19]	1290	1296	1302	1296	2.93						
lambda[20]	1352	1358	1363	1358	2.97						
lambda[21]	998	1003	1008	1003	2.54						
lambda[22]	1029	1034	1039	1034	2.58						
lambda[23]	1041	1046	1051	1046	2.6						
lambda[24]	970	975	980	975	2.52						
lambda[25]	895	900	905	900	2.41						

Table 4. sampling result of each hyperparameter.

Similar to 5.2, we also check the diagnostic plots of each parameter to make sure that they have converged to the posterior distribution. In figure 7, we provide a sample of such diagnostic plots.

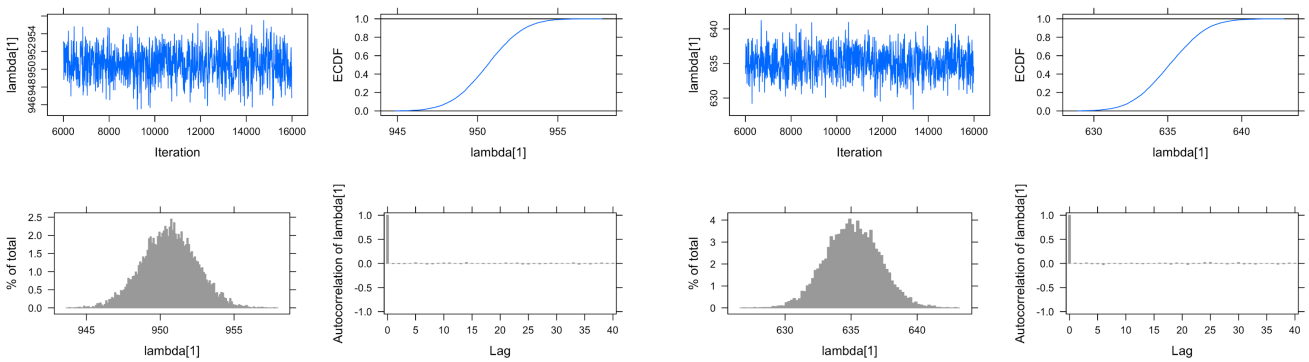

Figure 7. Diagnostic plots of simulated draws of lambda_1 for district and month

5.4 Discussion

In this section, we aim to provide answers to our two main questions by plotting out the estimate of the corresponding "mean" parameter of the three models. For the hierarchical models, we do so by plotting the corresponding μ or λ parameter. To make things comparable to the hierarchical models, for the Poisson regression model, we do so by computing the corresponding mean coefficient

by averaging out the coefficients of the unrelated category. For instance, to compute the mean parameter of district one, we will use the μ_1 or λ_1 for the hierarchical model and use $beta_0 + beta_{district1} + avg(beta_{month1}, \ldots, beta_{month12})$.

5.4.1 Which districts are more dangerous than the others?

The first question we are trying to answer with these models is "Which districts are more dangerous than the others?" by plotting the estimate of the corresponding "mean" parameters. In Figure 8 to 10, we see that the police districts with the most crime count are districts 15, 20, 5, and 6. However, since some districts may be bigger and have more people in them, it is unfair to compare the count without considering the population in it. As a result, we also compute the crime count per population in Figure 11 to 13. In terms of crime count per population, our model shows that districts 16, 13, 20, and 19 are the ones with the highest crime rate.

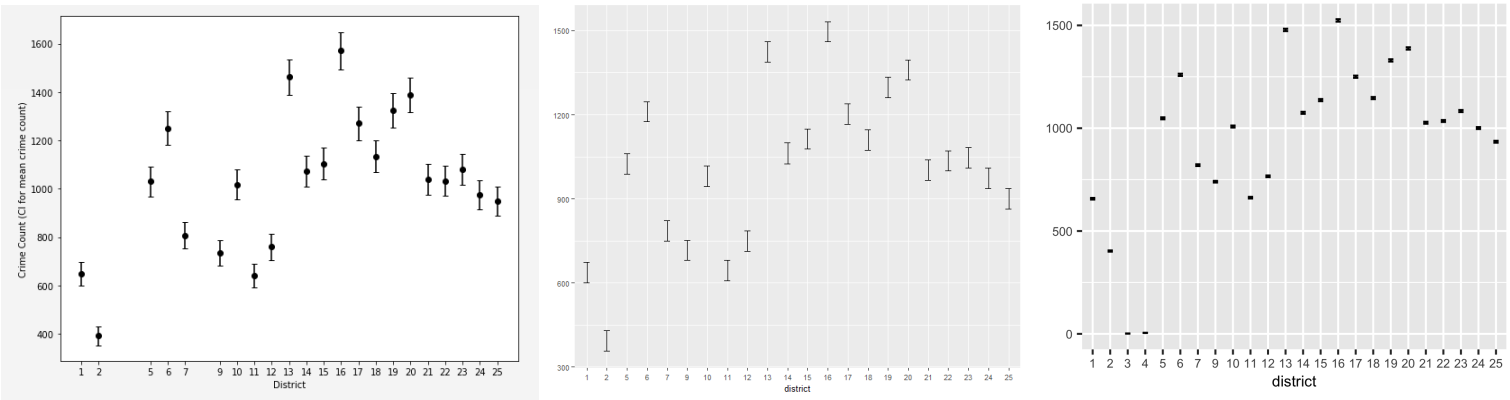

Figure 8 to 10. 95% CI for the mean of crime count by the three models (Poisson regression model, Hierarchical normal model, and Hierarchical Poisson model). The upper left plot the average mean in each district by averaging out the different months in a year. The upper right and lower ones plot the corresponding μ or λ for each district.

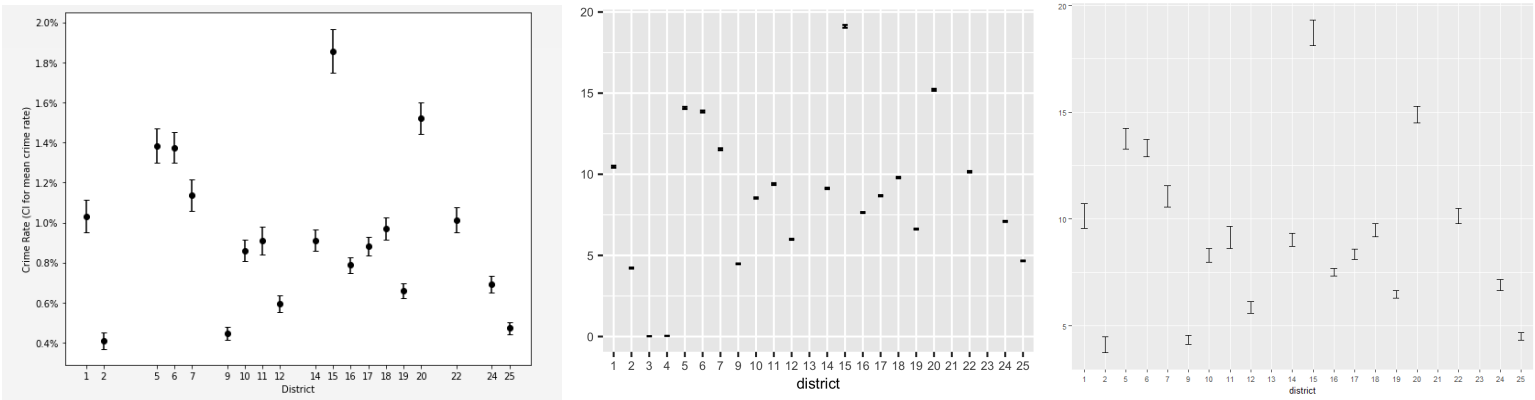

Figure 11 to 13. 95% CI for the mean of crime count by the three models (Poisson regression model, Hierarchical normal model, and Hierarchical Poisson model), divided by the population in the corresponding district. The upper left plot the average mean per person in each district by averaging out the different months in a year. The upper right and lower ones plot the corresponding μ or λ for each district per 1000 population.

5.4.2 Which months in a year have more crime than the others?

To address this question, we visualize the estimated means and 95% CI of the temporal parameters. In Figure 14 to 16, the plotted estimated data reveal compelling insights into the seasonal dynamics of crime incidents. July emerges as a standout month with a notably highest estimated means, suggesting a peak in crime incidents during this period. Additionally, August and July could observe a relatively high estimated mean. Conversely, February should be highlighted because of its remarkably lowest estimated crime count. Notice that we may sometimes see slightly different results in our regression model and hierarchical models, this is

because the regression model controls for the factor of the district, so if certain crime count in November, for instance, also happens to appear in the same district, the model may attribute the factor to the certain district instead of the factor of being in November.

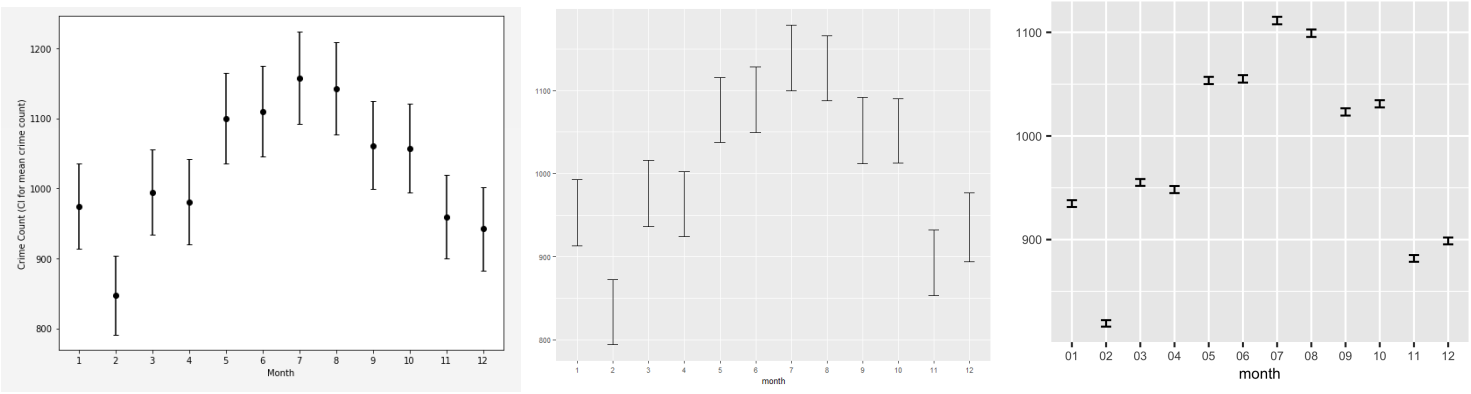

Figure 14 to 16. 95% CI for the mean of crime count by the three models for each month by the three models (Poisson regression model, Hierarchical normal model, and Hierarchical Poisson model). The upper left plot the average mean in each month by averaging out the different districts in Chicago. The upper right and lower ones plot the corresponding μ or λ for each month representing the population mean.

6. Conclusion

In this project, we implemented three different Bayesian models to answer the two research questions: Which police districts have more crime count than the others? Which months in a year do we see more crime in other months? We believe that successfully estimating the temporal and demographic pattern of crime occurrence can not only help us avoid dangerous time and place as a citizen but also help the police force allocate resources more efficiently. By using the three Bayesian models, we successfully located the districts or months with high crime rates that we can avoid or tackle with more resources.

7. References

- *Probability and Bayesian Modeling, Jim Albert and Jingchen Hu*
- *Crimes - 2001 to Present, Chicago Data Portal*
- *Tracking Chicago shooting victims: 2,021 so far this year, 164 more than in 2020, Chicagotribune.com*
- *Chicago race and ethnicity data by police district, John Keefe*
- *Bayesian Poisson Regression, Turing.jl*

Appendix. Contribution and Roles

Colin (Yiwei) Peng is responsible for the Hierarchial Poisson Model, Discussion, and Conclusion, Yongmin Kim is responsible for the problem statement, dataset description, and Hierarchical normal Model and Yonnie Chan is responsible for the exploratory data analysis, Discussion, and Poisson Regression model. In short, we thought that every member of the team contributed equally.

Code 1. EDA

```
import pandas as pd
import matplotlib.pyplot as plt
import math
import folium
from folium import Marker
from folium.plugins import MarkerCluster

df = pd.read_csv('Crimes_-_2001_to_Present_20231027.csv')
df.columns = [x.lower() for x in df.columns]
df.columns = df.columns.str.replace(' ', '_')
# What are the total missing values in the dataset
print("Number of Missing Values in the whole dataset : ", df.isna().sum().sum())
df_ = df.dropna().reset_index()
print(round(df_.shape[0]/df.shape[0] * 100,2), "percentage of the data has been retained.")

def extract_feature_from_date(df):
   df['date'] = pd.to_datetime(df['date'])

   df['date_month'] = df['date'].dt.month
   df['date_dayofmonth'] = df['date'].dt.day
   df['date_dayofweek'] = df['date'].dt.dayofweek
   df['date_weekofyear'] = df['date'].dt.isocalendar().week.astype(int)
   df['date_dayofyear'] = df['date'].dt.dayofyear
   df['date_hour'] = df['date'].dt.hour
   df['date_time'] = df['date'].dt.hour*60 + df['date'].dt.minute

   return df

df_ = extract_feature_from_date(df_)

df_temp = df_.groupby(['beat', 'date_month']).count()['id'].reset_index()
# 'id' is the column name resulting from the count, you can rename it if needed
df_temp.columns = ['beat', 'date_month', 'count']
# Create a scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(df_['longitude'], df_['latitude'], alpha=0.5)
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Scatter Plot of Location Data')
plt.show()

# Filter the DataFrame to remove outliers
df_ = df_[(df_['longitude'] > -91) & (df_['latitude'] > 37)]
```

```
# Create a scatter plot with filtered data
plt.figure(figsize=(10, 6))
plt.scatter(df_['longitude'], df_['latitude'], alpha=0.5)
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Scatter Plot of Location Data (Outliers Removed)')
plt.show()

# Top 10 crimes in the City of Chicago
top_crimes = df_['primary_type'].value_counts()
top_crimes.head(10)

# Get arrest information
arrest_info = df_['arrest'].value_counts()
arrest_df = pd.DataFrame({'Arrest?':arrest_info.index,'Number of Crimes':arrest_info.values})
df_crime = df_[df['year']>2013]
# Group the data by year and arrest status, and count the occurrences
crime_counts = df_crime.groupby(['year', 'arrest']).size().unstack()
# Calculate the total counts for arrested and non-arrested crimes
arrested_counts = crime_counts[True].sum()
unarrested_counts = crime_counts[False].sum()
# Create a pie chart for arrested and unarrested crimes
labels = ['Arrested', 'Unarrested']
sizes = [arrested_counts, unarrested_counts]
# explode the 'Arrested' slice
explode = (0.1, 0)
plt.figure(figsize=(8, 6))
plt.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Arrested and Unarrested Crimes from 2014-2023')
plt.axis('equal')
plt.show()

# Top 5 crimes that are more likely to be unarrested
unarrest_crime = df_.groupby('primary_type')['arrest'].value_counts(normalize = True).unstack()
# Filter rows where arrest_crime is True
unarrest = unarrest_crime[False]
# Sort arrest by decending order
sort_unarrest= unarrest.sort_values(ascending = False)
# print out the crime types
sort_unarrest.head(5)

# Top 5 crimes that are more likely to result in an arrest
arrest_crime = df_.groupby('primary_type')['arrest'].value_counts(normalize = True).unstack()
# Filter rows where arrest_crime is True
arrest = arrest_crime[True]
# Sort arrest by decending order
```

```
sort_arrest= arrest.sort_values(ascending = False)
# print out the crime types
sort_arrest.head(5)

# Group the data by Year and Crime_Category, and count the number of crimes
crime_counts_by_year = df_.groupby(['year']).count()['id']
# Create a line chart
plt.figure(figsize=(20, 6))
plt.plot(crime_counts_by_year.index, crime_counts_by_year.values)
plt.xlabel('Year')
plt.ylabel('Number of Crimes')
plt.title('Number of Crimes Over the Years')
plt.legend()
plt.show()

# Top 10 Community areas with the highest crime rate
community_crime = df_['community_area'].value_counts()
dangerous_df = community_crime.head(10)
area_crime = pd.DataFrame({'Community Area Number':dangerous_df.index,'Number of
Crimes':dangerous_df.values})
# Extract the month from the date
df_['month'] = df_['date'].dt.month

# Map the month values to corresponding season names
seasons_mapping = {
   1: 'Winter',
   2: 'Winter',
   3: 'Spring',
   4: 'Spring',
   5: 'Spring',
   6: 'Summer',
   7: 'Summer',
   8: 'Summer',
   9: 'Autumn',
   10: 'Autumn',
   11: 'Autumn',
   12: 'Winter'
}
df_['season'] = df_['month'].map(seasons_mapping)

# Group the data by season and count the occurrences of crimes
crime_counts = df_.groupby('season').size()

# Plot the crime rates by season
plt.figure(figsize=(10, 6))
crime_counts.plot(kind='bar', color='purple', alpha=0.7)
```

```
plt.title('Crime Rates by Season')
plt.xlabel('Season')
plt.ylabel('Number of Crimes')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()

# Calculate the crime counts for each month
crime_counts = df_['month'].value_counts().sort_index()
# Get the month names for labeling the pie chart
month_names = ['January', 'February', 'March', 'April', 'May', 'June',
               'July', 'August', 'September', 'October', 'November', 'December']
# Create a pie chart for monthly crime rate
plt.figure(figsize=(8, 8))
plt.pie(crime_counts, labels= month_names, autopct= '%1.1f%%', startangle=90)
plt.title('Monthly crime rate of the year')
plt.axis('equal')
plt.show()

# Calculate the crime counts for each month
hour_counts = df_['date_hour'].value_counts().sort_index()
# Create a bar plot for hourly crime rate
plt.figure(figsize=(10, 6))
plt.bar(hour_counts.index, hour_counts.values, width=0.4, color='skyblue')
plt.xlabel('Hour of the Day')
plt.ylabel('Number of Crimes')
plt.title('Hourly Crime Rate')
plt.xticks(range(24))  # Ensure x-axis ticks include all 24 hours
plt.show()

# Create the map for assault crime cases in 2023
assault_2023 = df_[((df_.primary_type == 'ASSAULT') & (df_.year == 2023))]
# Create the map
m1 = folium.Map(location=[41.877565108, -87.68479102], tiles='cartodbpositron', zoom_start=12)
# Add points to the map
mc = MarkerCluster()
for idx, row in assault_2023.iterrows():
   if not math.isnan(row['longitude']) and not math.isnan(row['latitude']):
       mc.add_child(Marker([row['latitude'], row['longitude']]))
m1.add_child(mc)
# Display the map
m1
```

Code 2. Poisson Regression Model

```
import pandas as pd
import numpy as np
import pymc3 as pm
import matplotlib.pyplot as plt
from sklearn import linear_model
from matplotlib.ticker import FuncFormatter
np.random.seed(551)

crime_df = pd.read_csv('crime_count_by_district.csv')
crime_df.columns = [x.lower() for x in crime_df.columns]

def extract_feature_from_date(df):
    df['date'] = pd.to_datetime(df['date'])
    df['date_year'] = df['date'].dt.year
    df['date_month'] = df['date'].dt.month
    return df
df = extract_feature_from_date(crime_df)

# Create prior, training, and evaluation datasets based on the specified time periods
prior_df = df[(df['date_year'] >= 2001) & (df['date_year'] <= 2010)].copy()
train_df = df[(df['date_year'] > 2010) & (df['date_year'] <= 2021)].copy()
eval_df = df[(df['date_year'] > 2021) & (df['date_year'] <= 2023)].copy()

prior_df = prior_df[(prior_df['district'] != 3) & (prior_df['district'] != 4) & (prior_df['district'] !=
8)]
train_df = train_df[(train_df['district'] != 3) & (train_df['district'] != 4) & (train_df['district'] !=
8)]
eval_df = eval_df[(eval_df['district'] != 3) & (eval_df['district'] != 4) & (eval_df['district'] != 8)]

unique_districts = train_df['district'].unique()

eval_df['district'] = eval_df['district'].apply(lambda x: eval_df['district'].unique().tolist().index(x))
eval_df['date_month'] = eval_df['date_month'].apply(lambda x:
eval_df['date_month'].unique().tolist().index(x))

# Set prior based on prior data
X_prior = prior_df[['district', 'date_month']].values
y_prior = prior_df['crime_count'].values

clf_prior = linear_model.PoissonRegressor()
clf_prior.fit(X_prior, y_prior)

prior_intercept_mean = clf_prior.intercept_
prior_distict_mean = clf_prior.coef_[0]
prior_month_mean = clf_prior.coef_[1]
```

```
prior_intercept_std = np.std(clf_prior.predict(X_prior) - y_prior)
prior_district_std = np.std(clf_prior.predict(X_prior))
prior_month_std = np.std(clf_prior.predict(X_prior))

# Encode categorical variables if needed
train_df['district'] = pd.Categorical(train_df['district'])
train_df['district_code'] = train_df['district'].cat.codes

# Encode date-related variables if needed
train_df['date_month'] = pd.Categorical(train_df['date_month'])
train_df['date_month'] = train_df['date_month'].cat.codes

# Define the model
with pm.Model() as poisson_regression:
   b0 = pm.Normal('b0', mu=prior_intercept_mean, sd=prior_intercept_std)
   b_district = pm.Normal('b_district', mu=prior_distict_mean, sd=prior_district_std,
shape=len(train_df['district'].unique()))
   b_month = pm.Normal('b_month', mu=prior_month_mean, sd=prior_month_std,
shape=len(train_df['date_month'].unique()))

   # Model the Poisson likelihood
   theta = (
       b0 +
       b_district[train_df['district_code']] +
       b_month[train_df['date_month']]
   )

   y_obs = pm.Poisson('y_obs', mu=np.exp(theta), observed=train_df['crime_count'])

# Sample from the model
with poisson_regression:
   trace = pm.sample(draws=2500, tune=500, chains=4)

# Print or analyze the trace as needed
print(trace)

summary = pm.summary(trace)
# Replace the index with district values
summary.index = ['b0'] + [f'b_district_{i}' for i in unique_districts] + [f'b_month_{i}' for i in range(1,
13)]
summary['var'] = round(summary['sd']**2, 4)
summary[['mean', 'sd', 'var']]

# Extracting traces for parameters from the training data
b0_trace = trace['b0']
b_district_trace = trace['b_district']
```

```
b_month_trace = trace['b_month']

# Extract the mean values for parameters
b0_trace_mean = np.mean(b0_trace, axis=0)
b_district_trace_mean = np.mean(b_district_trace, axis=0)
b_month_trace_mean = np.mean(b_month_trace, axis=0)
# Model the Poisson likelihood for evaluation data
theta_eval = (
   b0_trace_mean +
   b_district_trace_mean[eval_df['district']] +
   b_month_trace_mean[eval_df['date_month']]
)
# Calculate the predicted crime counts
y_pred_eval = np.exp(theta_eval)
# Add the predicted values to the evaluation DataFrame
eval_df['crime_cnt_pred'] = y_pred_eval
# Display the result
eval_df[['district', 'date_month', 'crime_count', 'crime_cnt_pred']]

# Extract posterior samples for the predicted counts
posterior_samples = np.random.poisson(np.exp(theta_eval), size=(len(trace), len(eval_df)))

# Calculate 95% credible interval
lower_bound = np.percentile(posterior_samples, 2.5, axis=0)
upper_bound = np.percentile(posterior_samples, 97.5, axis=0)

# Add the predicted values and confidence interval to the evaluation DataFrame
eval_df['crime_cnt_pred'] = y_pred_eval
eval_df['lower_bound'] = lower_bound
eval_df['upper_bound'] = upper_bound

# Display the result
result_df = eval_df[['district', 'date_month', 'crime_count', 'crime_cnt_pred', 'lower_bound',
'upper_bound']]

result_district = result_df.groupby('district').mean().reset_index()
result_district['district'] = [f'{i}' for i in unique_districts]
result_district[['district', 'crime_cnt_pred', 'lower_bound', 'upper_bound']]

# Add population data for every district to calculate crime rate
population_df = pd.read_csv('chicago_2010blocks_2020policedistricts_population.csv')
# Group by 'dist_num' and calculate the sum of 'P003001' for each group
population_by_dist = population_df.groupby('dist_num')['P003001'].sum().reset_index(name='population')

# Convert 'district' column in result_district to int64
result_district['district'] = result_district['district'].astype('int64')
```

```
# Merge the DataFrames
merged_df = result_district.merge(population_by_dist, left_on='district', right_on='dist_num', how='left')
# Drop the duplicate 'dist_num' column if needed
merged_df = merged_df.drop(columns='dist_num')

merged_df['crime_rate_mean'] = merged_df['crime_cnt_pred']/merged_df['population']
merged_df['crime_rate_upper'] = merged_df['upper_bound']/merged_df['population']
merged_df['crime_rate_lower'] = merged_df['lower_bound']/merged_df['population']
merged_df = merged_df.dropna(subset=['population'])

# Draw estimated mean/CI for District vs crime rate
plt.figure(figsize=(8, 6))  # Adjust the figure size as needed
# Plot the error bars
plt.errorbar(merged_df['district'], merged_df['crime_rate_mean'],
             yerr=[merged_df['crime_rate_mean']-merged_df['crime_rate_lower'],
merged_df['crime_rate_upper']-merged_df['crime_rate_mean']], fmt='o', color='black', capsize=3)
# Define a function to format y-axis as percentage
def percent_formatter(x, pos):
    return f'{x * 100:.1f}%'
# Create a formatter for the y-axis
percent_formatter = FuncFormatter(percent_formatter)
# Set y-axis formatter to percentage
plt.gca().yaxis.set_major_formatter(percent_formatter)
plt.xlabel('District')
plt.ylabel('Crime Rate (CI for mean crime rate)')
plt.xticks(merged_df['district'])
plt.tight_layout()
plt.show()

# Draw estimated mean/CI for District vs crime count
plt.figure(figsize=(8, 6))  # Adjust the figure size as needed
# Plot the error bars
plt.errorbar(result_district['district'], result_district['crime_cnt_pred'],
             yerr=[result_district['crime_cnt_pred']-result_district['lower_bound'],
result_district['upper_bound']-result_district['crime_cnt_pred']], fmt='o', color='black', capsize=3)
plt.xlabel('District')
plt.ylabel('Crime Count (CI for mean crime count)')
plt.xticks(result_district['district'])
plt.tight_layout()
plt.show()

# Draw estimated mean/CI for Month vs crime count
result_month = result_df.groupby('date_month').mean().reset_index()
result_month['date_month'] = [f'{i}' for i in range(1, 13)]
result_month[['date_month', 'crime_cnt_pred', 'lower_bound', 'upper_bound']]
```

```
plt.figure(figsize=(8, 6))  # Adjust the figure size as needed
# Plot the error bars
plt.errorbar(result_month['date_month'], result_month['crime_cnt_pred'],
             yerr=[result_month['crime_cnt_pred']-result_month['lower_bound'],
result_month['upper_bound']-result_month['crime_cnt_pred']], fmt='o', color='black', capsize=3)
plt.xlabel('Month')
plt.ylabel('Crime Count (CI for mean crime count)')
plt.tight_layout()
plt.show()

# Draw 22 line plots and compare predicted value vs true value over 12 months for 22 districts
# Sort the DataFrame by district and date_month for plotting
result_df_sorted = result_df.sort_values(by=['district', 'date_month'])
# Get unique districts
unique_district = result_df_sorted['district'].unique()
# Create a district_mapping dictionary
district_mapping = dict(zip(unique_district, unique_districts))
district_month_pre_true = result_df_sorted.groupby(['district', 'date_month']).mean()[['crime_count',
'crime_cnt_pred']].reset_index()

# Plotting
plt.figure(figsize=(20, 8))  # Adjust the figure size as needed
# Iterate over each district
for i, (district, group) in enumerate(district_month_pre_true.groupby('district'), start=1):
    plt.subplot(5, 8, i)  # Assuming you want a 5x5 grid of subplots
    plt.plot(group['date_month'], group['crime_count'], color='red', linewidth=1, label='True')
    plt.plot(group['date_month'], group['crime_cnt_pred'], color='blue', linewidth=1, label='Predicted')
    # Use the mapping to get the new index for the title
    real_district = district_mapping[district]
    plt.title(f'District {real_district}')
    plt.xlabel('Month')
    plt.ylabel('Crime Count')
    # Set y-axis limits
    plt.ylim(bottom=0)
plt.tight_layout()
plt.show()

# Draw 12 line plots and compare predicted value vs true value over 22 districts for 12 months
month_district_pre_true = result_df_sorted.groupby(['date_month', 'district']).mean()[['crime_count',
'crime_cnt_pred']].reset_index()
# Plotting
plt.figure(figsize=(20, 4))  # Adjust the figure size as needed
# Iterate over each month
for i, (month, group) in enumerate(month_district_pre_true.groupby('date_month'), start=1):
    plt.subplot(2, 6, i)  # Assuming you want a 3x4 grid of subplots
```

```
    plt.plot(group['district'], group['crime_count'], color='red', linewidth=1, label='True')
    plt.plot(group['district'], group['crime_cnt_pred'], color='blue', linewidth=1, label='Predicted')
    # Update the title with the range(1, 13) for months
    plt.title(f'Month {list(range(1, 13))[i-1]}')
    plt.xlabel('District')
    plt.ylabel('Crime Count')
    # Set y-axis limits
    plt.ylim(bottom=0)
plt.tight_layout()
plt.show()
```

Code 3. Hierarchical Normal Model

```
# Code for Hierarchical Normal Model
# We borrowed some come from the book
# "Probability and Bayesian Modeling,  Jim Albert and Jingchen Hu" Chapter 10


#setting the working directory and
setwd("C:/Users/na/Desktop/2023 FALL/Stats 551/Proj")
set.seed(123987346)

# load the library
library('runjags')
library(coda)

# preprocessing the data
 fulldata <- read.csv('Crimes_-_2001_to_Present(selected).csv')
 Date_temp <- strptime(fulldata$Date, format = "%m/%d/%Y")
 Date_value <- format(Date_temp, "%Y-%m")
 dataset <- data.frame(ID = fulldata$ID, Districts = fulldata$Police.Districts,
                       Date = Date_value)
 crime_count <- aggregate(dataset$ID, by = list(dataset$Districts, dataset$Date),
                          FUN= length)
 names(crime_count)[1] <- "District"
 names(crime_count)[2] <- "Date"
 names(crime_count)[3] <- "crime_count"
 write.csv(crime_count, "crime_count.csv", row.names=FALSE)

#Load data
crime_count <- read.csv('crime_count.csv')

#Remove district 3,4,8 (only available for specific period)
crime_count <- subset(crime_count, District != 3)
crime_count <- subset(crime_count, District != 4)
crime_count <- subset(crime_count, District != 8)

#Change District 23,24,25 to 3,4,8 temporally
crime_count$District[crime_count$District == 23] <- 3
crime_count$District[crime_count$District == 24] <- 4
crime_count$District[crime_count$District == 25] <- 8

#Divide the data_set
```

```
before2011 = crime_count[(crime_count$Date < '2011-00'),]
train_Data = crime_count[(crime_count$Date > '2011-00'),]

#set prior for upper hierarchy
mu_0 = mean(before2011$crime_count)
tau_0 = sqrt(var(before2011$crime_count))

#setting the model(by district)
modelString <-"
model {
## sampling
for (i in 1:N){
  y[i] ~ dnorm(mu_j[DistrictIndex[i]], invsigma2)
}
## priors
for (j in 1:J){
  mu_j[j] ~ dnorm(mu, invtau2)
}
invsigma2 ~ dgamma(a_s, b_s)
sigma <- sqrt(pow(invsigma2, -1))
## hyperpriors
mu ~ dnorm(mu0, g0)
invtau2 ~ dgamma(a_t, b_t)
tau <- sqrt(pow(invtau2, -1))
}
"

#set the value of the model
y <- train_Data$crime_count
DistrictIndex <- train_Data$District
N <- length(y)
J <- length(unique(DistrictIndex))

#define the model data
the_data <- list("y" = y, "DistrictIndex" = DistrictIndex,
                 "N" = N, "J" = J,
                 "mu0" = mu_0, "g0" = tau_0,
                 "a_t" = 1, "b_t" = 1,
                 "a_s" = 1, "b_s" = 1)

# run model
posterior <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("mu", "tau", "mu_j", "sigma"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 10000)

# print MCMC result
print(posterior, digits = 3)

# plot diagnastic plots
plot(posterior, vars="tau")
plot(posterior, vars="sigma")
```

```
plot(posterior, vars="mu")
plot(posterior, vars="mu_j")

# Compute R value and plot
tau_draws <- as.mcmc(posterior, vars = "tau")
sigma_draws <- as.mcmc(posterior, vars = "sigma")
R <- tau_draws ^ 2 / (tau_draws ^ 2 + sigma_draws ^ 2)
par(mfrow=c(1,1))
plot(density(R), main = "Density of R (district)",
     xlab="R", ylab="Density")

# Extract the posterior mean of mu_j
district_mean_sd = posterior$summaries[3:24, c("Lower95","Upper95","Mean","SD")]

## Draw CI plots
library(dplyr)
library(tidyr)
library(ggplot2)
district_mean_sd= as.data.frame(district_mean_sd)
district_mean_sd$district <- c(1:2, 23,24,5:7,25,9:22)
mcmc.confi.int= district_mean_sd
colnames(mcmc.confi.int) <- c("2.5%", "97.5%", "50%", "SD", 'district')
#merge
data_w_population <- merge(mcmc.confi.int, population_by_dist, by.x = "district", by.y = "dist_num", all.x = TRUE)
data_w_population$"2.5%/population" <- 1000*data_w_population$`2.5%`/data_w_population$population
data_w_population$"50%/population" <- 1000*data_w_population$`50%`/data_w_population$population
data_w_population$"97.5%/population" <- 1000*data_w_population$`97.5%`/data_w_population$population

#plot
data_w_population$district <- factor(data_w_population$district, levels = 1:25)
ggplot(data_w_population, aes(x = district, y = `50%/population`)) +
  geom_errorbar(aes(ymax = `97.5%/population`, ymin = `2.5%/population`), width = 0.3)  +
  theme(text = element_text(size=8)) +
  labs(y = NULL)

ggplot(data_w_population, aes(x = district, y = `50%`)) +
  geom_errorbar(aes(ymax = `97.5%`, ymin = `2.5%`), width = 0.3)+
  theme(text = element_text(size=8)) +
  labs(y = NULL)


## ---------- months ---------- ##
# same thing again, only change district to month#

#set prior for upper hierarchy
mu_0 = mean(before2011$crime_count)
tau_0 = sqrt(var(before2011$crime_count))

# change year-month to month

for (i in 1:9){
  grep_str = paste0("^\\d{4}-0", i, "$")
  train_Data$Date[grepl(grep_str, train_Data$Date)] <- i
}
```

```
for (i in 10:12){
  grep_str = paste0("^\\d{4}-", i, "$")
  train_Data$Date[grepl(grep_str, train_Data$Date)] <- i
}


#setting the model
modelString <-"
model {
## sampling
for (i in 1:N){
   y[i] ~ dnorm(mu_j[MonthIndex[i]], invsigma2)
}
## priors
for (j in 1:J){
   mu_j[j] ~ dnorm(mu, invtau2)
}
invsigma2 ~ dgamma(a_s, b_s)
sigma <- sqrt(pow(invsigma2, -1))
## hyperpriors
mu ~ dnorm(mu0, g0)
invtau2 ~ dgamma(a_t, b_t)
tau <- sqrt(pow(invtau2, -1))
}
"
y <- train_Data$crime_count
MonthIndex <- as.numeric(train_Data$Date)
N <- length(y)
J <- length(unique(MonthIndex))
the_data <- list("y" = y, "MonthIndex" = MonthIndex,
                 "N" = N, "J" = J,
                 "mu0" = mu_0, "g0" = tau_0,
                 "a_t" = 1, "b_t" = 1,
                 "a_s" = 1, "b_s" = 1)

posterior <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("mu", "tau", "mu_j", "sigma"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 10000)
print(posterior, digits = 3)

par(mfrow=c(1,1))
plot(posterior, vars="tau")
tau_draws <- as.mcmc(posterior, vars = "tau")
sigma_draws <- as.mcmc(posterior, vars = "sigma")
R <- tau_draws ^ 2 / (tau_draws ^ 2 + sigma_draws ^ 2)
plot(density(R), main = "Density of R (month)",
     xlab="R", ylab="Density")

# simulation to infer
# Extract the posterior mean of mu_j
month_mean_sd = posterior$summaries[3:14, c("Mean","SD")]
```

```
district_mean_sd = posterior$summaries[3:14, c("Lower95","Upper95","Mean","SD")]

district_mean_sd= as.data.frame(district_mean_sd)
district_mean_sd$month <- c(1:12)
mcmc.confi.int= district_mean_sd
colnames(mcmc.confi.int) <- c("2.5%", "97.5%", "50%", "SD", 'month')
#merge
mcmc.confi.int$month <- factor(mcmc.confi.int$month, levels = 1:12)
ggplot(mcmc.confi.int, aes(x = month, y = `50%`)) +
  geom_errorbar(aes(ymax = `97.5%`, ymin = `2.5%`), width = 0.3)+
  theme(text = element_text(size=8)) +
  labs(y = NULL)
```

Code 4. Hierarchical Poisson-Gamma Model

```
library(rjags)
library(runjags)
library(dplyr)
library(tidyr)
library(ggplot2)
getwd()
setwd("/Users/colin/Desktop")
#data <- read.csv("Crimes_-_2001_to_Present(selected).csv")
data <- read.csv("crime_count.csv")
data <- data[data$District != 3, ]
data <- data[data$District != 4, ]
data <- data[data$District != 8, ]

data_prior <- data[data["Date"] <= "2010-12",]
data_train <- data[data["Date"] >= "2011-01",]

prior_var <- var(data_prior$crime_count)
prior_mean <- mean(data_prior$crime_count)
prior_alpha <- (prior_mean**2)/prior_var
prior_beta <- (prior_mean)/prior_var

data_wide <- pivot_wider(data_train, names_from = Date, values_from = crime_count)
#sort
data_wide <- data_wide[order(data_wide$District),]

modelString <-"
model {
## likelihood
for (i in 1:n_dist){
  for (j in 1:n_date){
    y[i, j] ~ dpois(lambda[i])
  }
}
## priors
for (i in 1:n_dist){
   lambda[i] ~ dgamma(a, b)
}
}
```

```
"
y <- as.matrix(data_wide[, 2:ncol(data_wide)])
n_dist <- nrow(data_wide)
n_date <- ncol(data_wide)-1
the_data <- list("y" = y, "a" = prior_alpha,
                 "n_dist" = n_dist,
                 "n_date" = n_date,
                 "b" = prior_beta)
posterior <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("lambda"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 10000)

plot(posterior)
print(posterior, digits = 3)

library(coda)
mcmc <- as.data.frame(as.mcmc.list(posterior)[[1]])
mcmc.sorted <- apply(mcmc, 2, sort, decreasing=F)
conf_int_index <- floor(nrow(mcmc.sorted)*c(0.025, 0.5, 0.975))
mcmc.confi.int <- as.data.frame(t(mcmc.sorted[conf_int_index,]))
colnames(mcmc.confi.int) <- c("2.5%", "50%", "97.5%")
mcmc.confi.int$district <- data_wide$District

#get population data
population <- read.csv("chicago/chicago_2010blocks_2020policedistricts_population.csv")
population_by_dist <- population %>%
  group_by(dist_num) %>%
  summarise(population = sum(P003001))

#merge
data_w_population <- merge(mcmc.confi.int, population_by_dist, by.x = "district", by.y = "dist_num", all.x = TRUE)
data_w_population$"2.5%/population" <- 1000*data_w_population$`2.5%`/data_w_population$population
data_w_population$"50%/population" <- 1000*data_w_population$`50%`/data_w_population$population
data_w_population$"97.5%/population" <- 1000*data_w_population$`97.5%`/data_w_population$population

#plot
data_w_population$district <- factor(data_w_population$district, levels = 1:25)
ggplot(data_w_population, aes(x = district, y = `50%/population`)) +
  geom_errorbar(aes(ymax = `97.5%/population`, ymin = `2.5%/population`), width = 0.3)  +
  theme(text = element_text(size=8)) +
  labs(y = NULL)

ggplot(data_w_population, aes(x = district, y = `50%`)) +
  geom_errorbar(aes(ymax = `97.5%`, ymin = `2.5%`), width = 0.3)+
  theme(text = element_text(size=8)) +
  labs(y = NULL)
```

```
#group by month
data_train_m <- as_tibble(data_train) %>%
  separate(Date, c("year", "month"), sep = "-")
data_wide <- pivot_wider(data_train_m, names_from = c("District", "year"), values_from = crime_count)

modelString <-"
model {
## likelihood
for (i in 1:n_month){
  for (j in 1:n_obs){
    y[i, j] ~ dpois(lambda[i])
  }
}
## priors
for (i in 1:n_month){
   lambda[i] ~ dgamma(a, b)
}
}
"
y <- as.matrix(data_wide[, 2:ncol(data_wide)])
n_obs <- ncol(data_wide)-1
n_month <- nrow(data_wide)
the_data <- list("y" = y, "a" = prior_alpha,
                 "n_obs" = n_obs,
                 "n_month" = n_month,
                 "b" = prior_beta)
posterior <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("lambda"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 10000)

plot(posterior)
print(posterior, digits = 3)

library(coda)
mcmc <- as.data.frame(as.mcmc.list(posterior)[[1]])
mcmc.sorted <- apply(mcmc, 2, sort, decreasing=F)
conf_int_index <- floor(nrow(mcmc.sorted)*c(0.025, 0.5, 0.975))
mcmc.confi.int <- as.data.frame(t(mcmc.sorted[conf_int_index,]))
colnames(mcmc.confi.int) <- c("2.5%", "50%", "97.5%")
mcmc.confi.int$month <- data_wide$month

ggplot(mcmc.confi.int, aes(x = month, y = `50%`)) +
  geom_errorbar(aes(ymax = `97.5%`, ymin = `2.5%`), width = 0.3) +
  theme(text = element_text(size=8)) +
  labs(y = NULL)
```