UNITED STATES MILITARY ACADEMY

MA478 FINAL PROJECT

MA478: GENERALIZED LINEAR MODELS

COL NICHOLAS CLARK

SECTION H2

BY

CADET EVAN ASUNCION ’25, B1

WEST POINT, NEW YORK

07MAY2024

EA ____ MY DOCUMENTATION IDENTIFIES ALL SOURCES USED AND ASSISTANCE RECEIVED IN COMPLETING THIS ASSIGNMENT.

_____ I DID NOT USE ANY SOURCES OR ASSISTANCE REQUIRING DOCUMENTATION IN COMPLETING THIS ASSIGNMENT.

SIGNATURE: Evan Asuncion _____

MA478 Final Project:
An Inferential Analysis of Burglary Rates Across Chicago

Evan Asuncion

April 2024

1 Abstract

Crime, particularly burglary, continues to plague Chicago. Constrained by limited resources, city officials must decide where to prioritize efforts to best combat crime. Assisting city officials, we conduct an inferential analysis to determine what factors most influence burglary rates across Chicago's 552 different Census blocks. Using a dataset with 39,744 observations of each block for each month from 2010-2015, we generate generalized linear models (GLMs) and generalized linear mixed models (GLMMs) that explain some variation within burglary rates across Chicago. As GLMMs allow us to include random, as opposed to fixed, effects as predictors, they potentially explain more variation. Since burglaries are count data, each GLM and GLMM considered models the Poisson family distribution. Since there is a large proportion of observations with no burglaries reported, we consider a Zero-Inflated Poisson (ZIP) model as an alternative. We also consider placing auto-regressive correlation structure on month because of seasonal similarities. Comparing models via AIC, we find a GLMM with wealth and year as fixed effects and Census block and month as random effects, to best explain variation of burglary rates across Chicago. We find each of these socio-economic and spatial-temporal variables to have significant effects. Interpreting the final GLMM, we recommend Chicago law enforcement to focus on "hotspot" Census blocks, and for policymakers to address wealth disparity. A future inferential analysis into urban burglary rates should focus on gathering more data as considering only a small number of predictors is a major limiting factor.

Key Words: Poisson regression, spatial-temporal statistics, generalized linear mixed models, Chicago crime

2 Introduction

The city of Chicago has one of the highest crime rates in the United States of America. Evident in multiple news articles, crime is a major problem for the Midwestern city that continues to grow. Burglaries in particular have risen every year since 2021 according to the Chicago P.D. [1]. As such, the following proposal suggests an inferential study focused on identifying what most influences the difference in count of burglaries throughout the 552 distinct Census blocks of Chicago. Our specific problem statement is to infer how socioeconomic characteristics-unemployment rate, wealth measure, and number of young males-and temporal characteristics-month and year-of different Census Block groups in Chicago affect the count of burglaries. Another goal of the inferential analysis is to delineate between fixed effect variables and random effect variables. That is, we want to see if any predictor is a random variable itself with its own distribution. Ultimately, the findings of our inferential analysis will serve as valuable tools for policymakers, law enforcement agencies, and urban planners, to implement policies and initiatives aimed at reducing burglary rates and enhancing the safety of Chicago residents.

3 Literature review

The amount of crime experienced in a particular area varies on a number of socio-economic and demographic factors. As these factors vary depending on where and when researchers record data, modeling crime must take into account the spatial-temporal realm. Urban crime in particular heightens the importance of spatial-temporal factors due to the diverse, melting-pot nature of cities. As such, past research into the modeling of crime rates across large cities roots in spatial-temporal statistics. Within the general field of spatial-temporal statistics and a particular focus on urban crime, studies can be grouped into predictive or inferential categories.

Predictive studies, aim to generate models that can predict how much crime a particular subsection of a city could experience. These predictive studies, such as [2] and [7], consider both demographic factors as well as spatial-temporal factors. Two interesting examples of spatial-temporal factors that help predict crime rates in the aforementioned studies include movement of individual people's mobile devices into different neighborhoods and a "Granger" network that connects crime incidents across time and space. The drawback from these predictive studies' models, especially the ones that yield the highest accuracy, is that they often are black boxes. Although these predictive models can help government agencies identify crime "hotspots" that need to be prioritized, we cannot use them to interpret why these certain areas are more likely to experience higher incidence of crime.

Inferential studies, on the other hand, allow us to determine what factors directly influence crime rates in different parts of the cities. They evaluate generated models' coefficients corresponding to potential explanatory variables and compare models with a different set of these factors in order to determine if which factors are most significant in influencing crime rates. Along with verifying the significant effects of socio-economic and demographic variables, recent inferential analyses have suggested interesting spatial-temporal effects that also have a significant effect across urban neighborhoods regarding crime rates. Relying on Generalized Linear Mixed Models (GLMM) Jung et al. suggest there to be a significant seasonal temperature effect on the number of assaults experienced in Seoul[6]. A drawback, however, is that many papers focus on different international cities and their crime, making it difficult to generalize conclusions to Chicago. However, the studies that do focus specifically on Chicago demonstrate similar findings. For example, Wang et al. demonstrate that factors relating to taxi flow between neighborhoods better explains variation of crime rate. Our study looks to expand on and unite these past studies of crime rate inferential analysis, using Generalized Linear Mixed Models to further identify distinct significant factors, delineating between fixed effects and random effects, and focusing specifically on burglaries specifically within the different Census blocks of Chicago.

4 Methodology

4.1 Data Collection and Transformation

To analyze the amount of burglaries in Chicago, we compile a dataset from raw data from the US Census. The raw data, pulled from a GitHub folder, includes burglary counts from 552 distinct Census block groups of Chicago measured from 2010-2015. The GitHub folder also provides separate raw .csv files regarding the following socioeconomic factors: population statistics, unemployment rates, wealth distribution, and the count of young males. To get a more robust dataset, one we can actually fit models on, we convert the count data from wide to long format and then join the socioeconomic data based on an observations given block. The final dataset that we fit models on contains 39,744 observations with the following variables: Census Block, population, wealth distribution, unemployment, count of young males, month, year, and observed count of burglaries. To avoid model overfitting, we split the dataset into a training set, 80% of the observations which we actually fit the models on, and a test set, the other 20% which can be used to test each models predictive capabilities. Finally, we transform the year variable to be on a 0-5 scale, zero being 2010, to make computation easier. For similar reasons, we also considered to standardize population but ultimately did not as the transformation made it difficult to fit mixed models with an offset term. Each

non-response variable contained within the data set is listed in Table 1.

4.2 Data Exploration

Before we actually fit any models, we conduct an initial exploratory data analysis to identify any trends or oddities within the dataset. In doing so, we have a better starting point in generating well-informed models which we can use to infer what most influences burglary rates across Chicago. Looking at the observed distribution of burglaries throughout all the 552 Chicago Census Blocks through all months and years observed, as in Figure 1, there is a large amount of observations with no burglaries reported. This could be due to either there being no burglaries in a block or if there were actually burglaries but they went unreported for whatever reason. To account for the excess of observations with zero burglary counts, one type of model we will consider is the Zero-inflated Poisson (ZIP) model.

Exploring potential covariates observed relationship with burglary counts, we find some initial trends that suggest we should include certain variables within our models. First, we look at the socio-economic potential factors. From looking at sample correlation coefficients and a pairs plot, Figure 2, it appears that all three socioeconomic variables within the dataset–wealth, unemployment, and count of young males–influence the count of burglaries. The suggested significance of these variables is in line with longstanding criminology literature [5]. However, as evident in Figure 3, there is potential multicollinearity caused by the direct relationship between the amount of young males in a given block and the given block's aggregate wealth. To avoid issues caused by multicollinearity, we only consider wealth and not count of young males in our model.

Exploring the spatial-temporal aspects–namely block, year, and month—-in our data, we find both apparent random and fixed effects on burglary count. When we look at the relationship between year and burglary counts, shown in Figure 4, there appears to be a fixed effect: from year-to-year, the number of burglaries decrease. Conversely, there is not such a direct linear relationship between month and burglary count. Evident in Figure 5, the amount of burglaries experienced in a given Census block increases as it gets into the summer months, peaks in August, and then decreases during the fall and winter months. Since this effect could be seasonal, we not only consider month as a random effect but also consider placing an autoregressive (AR1) correlation structure on the temporal variable. The AR1 term assumes that some months, months within the same season, are more closely related than others. In summary, our initial data exploration led us to well-informed starting points–Poisson Generalized Linear Model (GLM), Zero-Inflated Poisson (ZIP) model, and a Generalized Linear Mixed Model with a possible AR1 term relating to month–for model generation.

4.3 Model Generation

As we are modeling burglary counts, the initial model we consider is the simplest model for count data: a Poisson GLM. The initial predictor variables we place in the model are wealth, unemployment, and year. It is also important to note that we add population as an offset term. This way we model burglary count rate, taking into account differences in population, instead of solely modeling aggregate burglary counts. We consider adding more predictors, using likelihood ratio tests to verify variable significance. However the inital GLM only contains those three aforementioned quantitative varaibles and can be written out as follows:

$$y_{ij} \sim Pois(\lambda_{ij}) \tag{1}$$

$$log(\lambda_{ij}) = \eta_{ij}$$

$$\eta_{ij} = \beta_0 + \beta_1 Wealth_i + \beta_2 Unemployment_i + \beta_3 Year_j + log(Population_i)$$

where y_ij is the count of burglaries in Census block i and year j. Since Poisson regression assumes that $\lambda = \mu$, we consider λ_{ij} as the average count of burglaries in a given block and year. We implement the GLM

R Package "glmnet" which relies on Maximum Likelihood Estimation (MLE) to fit the model. fitted model coefficients can be interpreted as the multiplicative change in expected burglary counts per population. For example, e^{β_1} can be interpreted as the change in expected burglary rate due to a one unit increase in wealth. Another related assumption we take in fitting this Poisson GLM is that the mean and variance should be equal.

As the data exploration revealed, there is a large amount of Census blocks where we observe no burglaries within a given month and year. This might mean a zero-inflated Poisson (ZIP) model better captures the variance of burglary counts and is a better choice to infer the most significant predictors. The ZIP model we build can be written as follows:

$$y_{ij} \sim \begin{cases} 0 & \text{w/p } \phi_{ij} \\ Pois(\lambda_{ij}) & \text{w/p } (1-\phi_{ij}) \end{cases} \tag{2}$$

$$\begin{aligned} z_{ij} &\sim Bern(\phi_{ij}) \\ y|z_{ij} &= (\frac{e^{-\lambda_i j}\lambda^2}{y!})^{1-z_i} \\ log(\lambda_{ij}) &= \eta_{ij} \end{aligned}$$

$$\eta_{ij} = \beta_0 + \beta_1 Wealth_i + \beta_2 Unemployment_i + \beta_3 Year_j + log(Population_i)$$

Where, as with the prior model, y_{ij} is the count of burglaries in Census block i and year j and ϕ_{ij} is the probability of observing excess zeros in block i and year j. There are two components of a ZIP Model: the zero-inflated model and the count model. The zero-inflated model can be thought as representing a Bernoulli distribution. That is, we interpret the intercept coefficient as the likelihood of observing excess zeros. Having a low likelihood of observing excess zeros could suggest some sort of mechanism that leads to no burglaries being reported. The count model, on the other hand, assumes a regular Poisson GLM. As such, the coefficients of the count model can be interpreted the same as in Equation 1. It is also key to point out that we fit the ZIP model via MLE, making it possible to compare its output to that of the initial Poisson GLM.

A drawback from theses first two models–the Poisson GLM and the ZIP model–is that they do not account for random effects. Unlike fixed effects, which assume uncertainty only comes from one source, random effects account for situations where uncertainty comes from an individual predictor. Since our initial data exploration along with inherent instinct suggest there to be random effects due to both month recorded as well as uniqueness of each block, the next class of models we consider are GLMMs. The specific GLMM we initially fit can be written out as the following:

$$y_{ijk} \sim Pois(\lambda_{ijk}) \tag{3}$$

$$log(\lambda_{ijk}) = \eta_{ijk}$$

$$\eta_{ijk} = \beta_0 + \beta_1 Wealth_i + \beta_2 Unemployment_i + \beta_3 Year_j + \alpha_i + \gamma_k$$

$$\begin{aligned} \alpha_i &\sim MVN(0, \alpha_i^2) \\ \gamma_k &\sim MVN(0, \gamma_k^2) \end{aligned}$$

Where y_{ijk} is the count of burglaries in block i observed in month k and year j, α_i is a random effect due to block, and γ_k is a random effect due to month. Although this basic GLMM accounts for random variation within the block and months variable, placing auto-regressive correlation structure on these random effects could explain more variation within Chicago burglaries. Auto-regressive (AR-1) terms assume that some observations have variables that are more related compared to other observations. For example, an AR-1 term for the month variable assumes that some months are more closely related than others. With the three seasons and their differences, month as an AR-1 term makes sense. We can modify Equation 3's linear predictor to include the month AR-1 term:

$$\eta_{ijk} = \beta_0 + \beta_1 Wealth_i + \beta_2 Unemployment_i + \beta_3 Year_j + \alpha_i + \gamma_k \tag{4}$$

$$\gamma_k = \phi_{\gamma_k - 1} + u_k$$
$$u_k \sim MVN(0, \sigma_u^2)$$

To fit these GLMMs, we implement the "glmer" and "lme" functions from the "lme4" RPackage. Like those functions generating the prior models, these GLMM-fitting functions rely on MLE. Again, since all our models rely on MLE for fitting, we are able to directly compare them.

4.4 Model Selection

To conduct the most well-informed inferential analysis, we must interpret the output of the best model, the model which explains the most variance within burglaries observed throughout the 552 distinct. We judge and compare models by evaluating Aikaike Information Critera (AIC). AIC is found through the following equation:

$$AIC = -2l(\hat{\theta}) + 2p \tag{5}$$

Where $l(\hat{\theta})$ is the log-likelihood of the model and p is the number of predictors. AIC is a measure to compare models because it evaluates fit while also penalizing for complexity. Especially for an inferential analysis like this, we do not want a model that is to complex to interpret. With that, our best model will minimize AIC. Another test of performance that we implement, specifically for variable selection, are likelihood ratio Chi-squared tests. Tied to model deviance, the likelihood ratio Chi-squared test compares the deviance of a simpler model with the deviance of a more complex model, a model containing more predictors. Equation 9 describes $D(y, \hat{\mu})$, the deviance of a given model. Equation 10 outlines the underlying Null Hypothesis behind likelihood ratio Chi-squared tests.

$$D(y, \hat{\mu}) = 2\Sigma_{i=1}^{n}[y_i log(\frac{y_i}{\hat{\mu}_i}) - y_i + \hat{\mu_i}] \tag{6}$$

$$H_0 : D_1(y, \hat{\mu_1}) - D_2(y, \hat{\mu_2}) \sim \chi^2_{n-p} \tag{7}$$

Testing the probability of the difference of deviance between a "simple model" and a "complex model" with more predictors being observed under a χ^2_{n-p} distribution, we can derive a p-value. If the resultant p-value is significant, we suggest that the more complex model, the model with more predictors, is preferred. Likelihood ratio tests, however, can only be implemented in nested models. Although forward selection is oftentimes laborious, in this context the dataset only has six predictors to consider. As such, we implement the process to verify variable significance. Starting with the null model, we subsequently add variables, performing a likelihood ratio test after each variable addition. If the p-value is significant, then the variable in question should be included in the model.

5 Experimentation and Results

5.1 Poisson GLM

As the simplest model for count data, the output of the Poisson GLM is the easiest to interpret. Subsequent likelihood ratio tests as described in the Methodology section yielded 3 significant p-values, suggesting that wealth, unemployment, and year are all significant predictors of burglary count. The fitted linear predictor of Equation 1 is:

$$\hat{\eta}_{ij} = -6.41 - 0.18 * Wealth_i + 0.52 * Unemployment_i - 0.16 * Year_j + log(Population_i) \tag{8}$$

Interpreting these fitted coefficients, we suggest that a one unit increase in wealth results in a 0.84 factor decrease in burglaries per capita, a one unit increase in unemployment results in a 1.69 factor increase in burglaries per capita, and a one unit increase in year results in a 0.85 factor decrease in burglaries per capita. We are able to interpret the model because it yields a dispersion parameter $\phi = 1.6$ that is not too far off from the 1:1 mean to variance ratio assumption. A problem with this fitted model, however, is that it fails the Chi-square goodness of fit test. What this means is that the saturated model, containing all combinations of predictors in the dataset, better captures the variance within the model. As there are only six predictors in the data, the fitted saturated model is not too different from Equation 6. However, due to the addition of categorical variables and possible interactions, the saturated model is a lot harder to interpret.

5.2 ZIP Model

As shown in Table 2, the ZIP model yields a lower AIC than the Poisson GLM so it is a better choice to capture the variance of Chicago burglary rates. The summary of the fitted count model of the ZIP Model provides a very similar output to the fitted Poisson GLM:

$$log(\hat{\lambda}_{ij}) = -6.25 - 0.22 * Wealth_i + 0.47 * Unemployment_i - 0.13 * Year_j + log(Population_i) \tag{9}$$

The count model's coefficients can be interpreted the exact same way as the coefficients of the Poisson GLM in Equation 6. The zero-inflated model's coefficients, conversely, can be interpreted as the multiplicative change in the probability of observing excess zeros. We see these fitted coefficients of the zero-inflated model in Equation 8:

$$\hat{\eta_{ij}} = -8.69 - 0.57 * Wealth_i - 0.01 * Unemployment_i + 0.16 * Year_j + log(Population_i) \tag{10}$$

We can interpret the intercept term, $\hat{\beta}_0 = -8.69$, as a low probability of observing an excess amount of zero burglaries across Chicago. We can also interpret the coefficient corresponding to wealth and unemployment, which are both negative, as a sign that blocks are more likely to report a burglary if they have a wealthier population and face larger unemployment rates. With the coefficient corresponding to year being positive, we also see that there has been an increase in the probability of no burglaries being reported. Similar to the initial Poisson GLM, the ZIP model does not pass the Chi-squared goodness of fit test, suggesting that a saturated ZIP model is better. Another downside of the ZIP model is that, yielding a dispersion parameter of $\phi = 2.98$, it slightly violates the Poisson family assumption that the mean and variance are equal. Although the slight overdispersion suggests poor fit and explanation of variation, it does not take away from the interpretability of the ZIP model.

5.3 GLMMs

When we add random effects to our models, shifting from generating GLMs to generating GLMMs, the new GLMMs yield a lower AIC. Comparing AIC between different GLMMs, we find that adding both month and

block as random effects yield a lower AIC. This suggests that each random effect is significant in explaining the variation of Chicago burglary rates. However, unlike with previous models consider, a likelihood ratio test between a GLMM with and without unemployment yields an insignificant p-value, suggesting the simpler model to be better. That is, we find unemployment to be an insignificant factor within a GLMM. The fitted version of this GLMM with wealth and year as fixed effects and month and block as random effects, which is the best model as listed in Table 2, is written as follows:

$$\hat{\eta}_{ijk} = -6.44 - 0.22 * Wealth_i - 0.164 * Year_j + \alpha_i + \gamma_k \tag{11}$$

$$\hat{\alpha_i} \sim MVN(0, 0.2582)$$
$$\hat{\gamma_k} \sim MVN(0, 0.0318)$$

Like the past model, we can interpret the slope coefficients corresponding to the fixed effects of the GLMM as multiplicative changes to the expected number of burglaries per population. Yielding a dispersion parameter of $\phi = 1.28$, there are also no potential issues regarding overdisperson. When attempting to include an AR(1) term for month, the R package we use to fit, "lme4," does not permit offset terms when specifying correlation structure. Although we fit a model with $log(population)$ as a predictor, it is not the exact same as adding an offset term. When we do fit the model with an incorrectly applied offset, we are left with a fitted model with the lowest AIC yet, 112631. As such, we do not consider any output from the GLMM with a month AR(1) term in our inferential analysis takeaways.

6 Discussion and Conclusions

6.1 Conclusions

Although the GLMM, since it yielded the lowest AIC, was the "best" model to explain variation with Chicago burglary rates, it still failed the Chi-squared goodness of fit test. In fact, all models that we generated failed to be a better fit than the saturated model. Although this means that our models are not the best fits of Chicago burglary rates, it does not take away from their interpretability. As the saturated model has over 564 additional beta coefficients included in the model, due to adding categorical variables, it is a lot harder to interpret the model output. Furthermore, as the saturated model includes the young males count variable, we could suffer from multicollinearity issues. As such, we will rely on analyzing the simpler, more interpretable models in our suggestions to Chicago city officials.

Since every class of model considered suggested it to be the case, we conclude that wealth and year are statistically significant fixed effects. Specifically, we find that a wealthier block would expect a lower expected burglaries per capita. As such, we suggest Chicago P.D. to put more resources into to less wealthy areas and for city policymakers to create legislation focused on decreasing wealth disparity. We also find that the expected number of burglaries per capita significantly decreases over time. Although this might suggest a forward-looking trend, we should be wary to reach any definite conclusions. First, the dataset only contains 5 years worth of information. There is decades worth of other data that could suggest a separate, unrelated, or nonlinear trend between year and burglary rate. Second, the decrease in burglaries per capita could be due to a decrease in reporting. The output of the ZIP model supports this. That is, the coefficient corresponding to year within the zero-inflated model suggests that a one year increase in year leads to a higher probability of observing excess zeros. As such, it might be worthwhile to look into any reason or mechanism that explains the potential dip in burglary reporting. Since unemployment did not end up being a significant factor within the GLMM, we cannot come to any conclusion regarding the socio-economic variable.

As the GLMM containing both block and month as a random effect yielded the lowest AIC out of the models compared, we suggest that both of the spatio-temporal variables are significant random effects. That is each block has it own unique randomness which can influence the occurrence of a burglary. Similarly, we conclude that different months bring uniqueness to crime rates across Chicago. However, since our methods

were unable to fit models with both AR(1) and offset terms, we cannot come to any conclusion regarding the presence of any auto-regressive correlation structure.

6.2 Limitations

When considering our suggestions above, it is important to keep in mind the limitations of our inferential analysis. The primary factor limiting our study is the small amount of potential predictors within our dataset. There is no doubt that more factors that influence prevalence of burglaries throughout Chicago. These factors include, but in no way are limited to, gender ratio, ethnicity stats, and education ranking for each Census block. The small amount of factors in our dataset also likely had an effect on always failing the Chi-squared goodness of fit test. That is, there is simply too little a number of predictors for the saturated model to be very different than a non-saturated model.

Another limitation resulting from the data, was the limited range and diversity of observations recorded. Observations in the sample were only taken in a 5-year span. Even if the dataset spanned a longer range of time periods, we would suffer from the fact that all socio-economic variables are constant. That is, a given Census block's population, wealth, unemployment rate, or count of young males remains the same for all years recorded. This does not model reality, where all those values can change from year-to-year.

Regarding limitations relating to the models we build, we cannot fit any models through any process but MLE to ensure that we can accurately compare models via AIC. There exist other methods to fit models, to include Integrated nested Laplace approximations (INLA), which could result in better fitting models. Furthermore, as the ZIP Model leads to slight overdispersion, our conclusions from it are not completely valid. An alternative to the ZIP model is a Zero-inflated negative binomial (ZINB) model which models the same scenario of excess zeros but relaxes the 1:1 mean-to-variance ration assumptions. A downside to the ZINB model, however, is that is not as interpretable.

6.3 Future Work

Future work will directly improve upon this inferential analysis by confronting the limitations of this study. First, future work will consider a large dataset. Primarily in the sense that there are a greater number of predictors, but also in the sense that the range and diversity of the observations expands. Next, future work will consider fitting models through a process other than maximum likelihood estimation. INLA is a good choice since it implements Bayesian inference and estimates the posterior. In simple terms, INLA computes a current probability distribution based on prior distributions. INLA and Bayesian inference could be particularly useful in the spatio-temporal context of Chicago burglary rates. Another way future work could improve upon this inferential analysis is to look at different types of models, not just GLMs and GLMMs. Specifically, a GEE model could help capture the population level impact for burglary frequency in Chicago.

7 Ethical Implications

A key part of any statistical analysis, is ensuring that we adhere to inherent, globally-recognized moral and ethical standards. The following section discusses how this inferential analysis fits into all standards outlined in the Data Science Ethics Checklist [4].

7.1 Data Collection

As we are focused on Census blocks, non-human subjects, informed consent and limited PII exposure are not a concern. Since the source of the raw data is the Chicago Data Portal, and official City of Chicago website and a seemingly trustworthy source, we do not consider there to be major collection bias [3]. Regarding downstream bias outcomes, there is no variable which we consider that could lead to biased outcomes. Each Censuse block appears in the dataset the same amount of times, so no bias comes from unbalanced data.

7.2 Data Storage

The original source of data can is located at https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2/about_data. Along with the code within the GitHub folder and my RStudio, there is no harm in storing it. It can continue to be used to educate students enrolled in statistics courses. Regarding right to be forgotten, there is no specific personal information to be removed so it is not a concern. People interested in the Chicago burglary problem should, as years continue to go by, keep in mind that our models were trained on data from 2010-2015. This time period might not be generalizeable to future or past time periods.

7.3 Analysis

Regarding missing perspectives, we could benefit from hearing the opinions of people living in different types of Census blocks. These perspectives could reveal some of the randomness of the block effect as well as hint at some sort of mechanism behind burglaries not being reported. Regarding auditability, I would say that the methodology section of this paper is well documented and this inferential analysis is definitely reproducible. Again, I do not think this data brings any concern of bias, honest representation, or privacy. We have to trust in the source of the data, which I do.

7.4 Modeling

Overall, our modeling process upholds data science ethics. The dataset is balanced, regarding Census block observations, and there are no major signs indicating unfairness across groups. Proxy discrimination could exist if one specific identity group was over sampled within a given Census block. Again, we cannot verify this without more information on the original data collection methods. Our main performance metric, AIC, does not provide sufficient predictive information. We use it to compare models and determine which model is best to interpret and reach inferential conclusions. As such, there no real way to "optimize" AIC values. The prior Conclusions and Limitations section fulfill the checklist items of model explainability and the communication of limitations.

7.5 Deployment

Regarding deployment of this inferential analysis, it is up to how the city of Chicago processes and acts upon our conclusions. There is no roll black plan; once city officials see this paper, it is up to them to implement its suggestions fairly. Their city trusts them to lead so they should be trusted to not misuse the results of this statistical analysis.

References

[1] •. *WATCH LIVE: CPD, Johnson to announce 'strategy to address and prevent robberies' in Chicago*. NBC Chicago. Apr. 12, 2024. URL: https://www.nbcchicago.com/news/local/cpd-mayor-johnson-to-announce-strategy-to-address-and-prevent-robberies-in-chicago/3408596/ (visited on 04/27/2024).

[2] Andrey Bogomolov et al. "Once Upon a Crime: Towards Crime Prediction from Demographics and Mobile Data". In: *Proceedings of the 16th International Conference on Multimodal Interaction*. ICMI '14: INTERNATIONAL CONFERENCE ON MULTIMODAL INTERACTION. Istanbul Turkey: ACM, Nov. 12, 2014, pp. 427–434. ISBN: 978-1-4503-2885-2. DOI: 10.1145/2663204.2663254. URL: https://dl.acm.org/doi/10.1145/2663204.2663254 (visited on 05/03/2024).

[3] Nicholas John Clark. "Self-exciting spatio-temporal statistical models for count data with applications to modeling the spread of violence". Pages: 12318517. Doctor of Philosophy. Ames: Iowa State University, Digital Repository, 2018. DOI: 10.31274/etd-180810-5963. URL: https://lib.dr.iastate.edu/etd/16333/ (visited on 05/07/2024).

[4] Deon. *An ethics checklist for data scientists*. URL: https://deon.drivendata.org/#data-science-ethics-checklist (visited on 05/07/2024).

[5] Richard B. Freeman. "Chapter 52 The economics of crime". In: vol. 3. Handbook of Labor Economics. ISSN: 1573-4463. Elsevier, 1999, pp. 3529–3571. DOI: https://doi.org/10.1016/S1573-4463(99)30043-2. URL: https://www.sciencedirect.com/science/article/pii/S1573446399300432.

[6] Yeondae Jung, Dohyeong Kim, and Alex R. Piquero. "Spatiotemporal Association Between Temperature and Assaults: A Generalized Linear Mixed-Model Approach". In: *Crime & Delinquency* 66.2 (Feb. 2020), pp. 277–302. ISSN: 0011-1287, 1552-387X. DOI: 10.1177/0011128719834555. URL: http://journals.sagepub.com/doi/10.1177/0011128719834555 (visited on 05/03/2024).

[7] Victor Rotaru et al. "Event-level prediction of urban crime reveals a signature of enforcement bias in US cities". In: *Nature Human Behaviour* 6.8 (June 30, 2022), pp. 1056–1068. ISSN: 2397-3374. DOI: 10.1038/s41562-022-01372-0. URL: https://www.nature.com/articles/s41562-022-01372-0 (visited on 05/03/2024).

8 Appendices

8.1 A: Supplemental Tables and Figures

Variable	Type	Unit
Wealth	Quantitative	N/A (standardized)
Unemployment	Quantitative	N/A (standardized)
Young Males Count	Quantitative	N/A (standardized)
Census Block	Categorical	Block # (1-552)
Year	Quantitative	Year (2010-2015)
Month	Categorical	Month (JAN-DEC)
Population	Offset	# people

Table 1: Explanatory Variables within Dataset

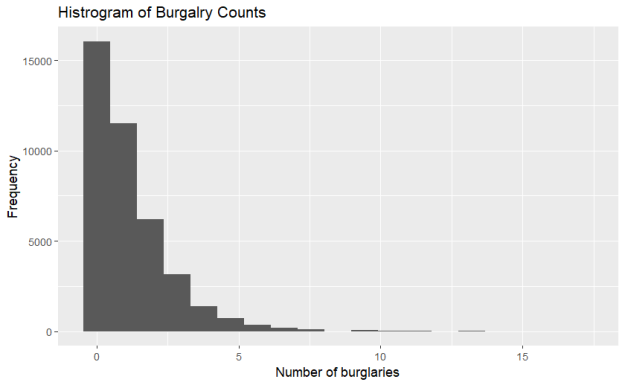


Figure 1: Distribution of Burglary Counts

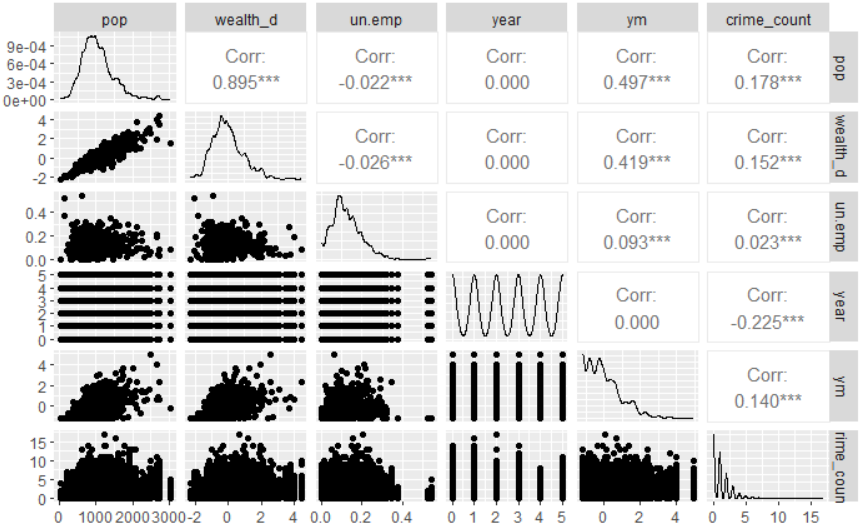


Figure 2: Pairs Plot Showing Potential Covariates

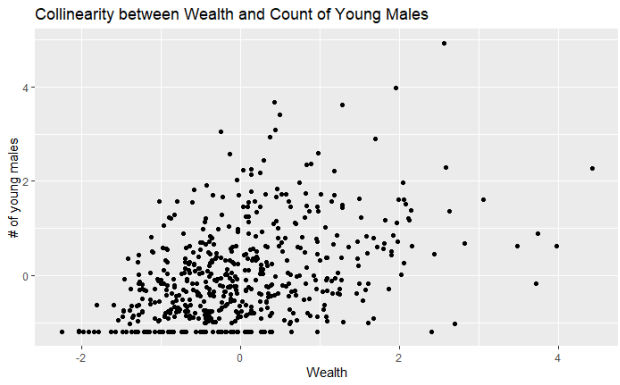


Figure 3: Relationship between Wealth and Count of Young Males

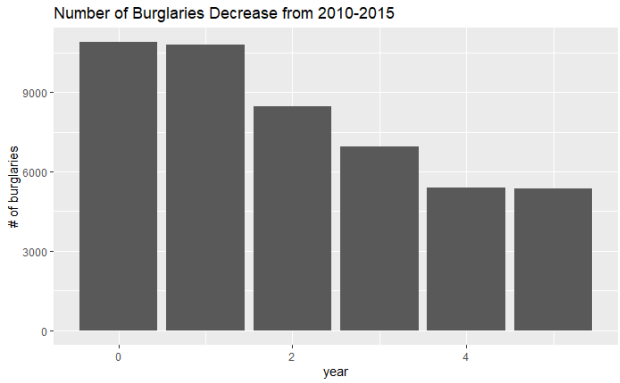


Figure 4: Relationship between Year and Burglary Count

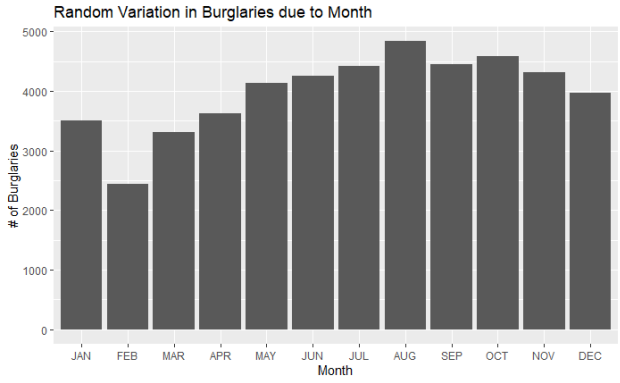


Figure 5: Relationship between Month and Burglary Count

Model	**AIC**	χ^2 **Goodness of Fit p-value**	**Dispersion Parameter** (ϕ)
Poisson GLM	95921	0	1.58
ZIP Model	94918	0	2.98*
GLMM	*89543*	0	1.28

Table 2: Model Comparison

8.2 B: RStudio Code Annex

MA478FinalProject

2024-04-07

```
# LOAD NECESSARY PACKAGES
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(INLA)
```

```
## Loading required package: sp
```

```
## Warning: package 'sp' was built under R version 4.3.3
```

```
## This is INLA_24.02.09 built 2024-02-09 03:35:28 UTC.
##  - See www.r-inla.org/contact-us for how to get help.
##  - List available models/likelihoods/etc with inla.list.models()
##  - Use inla.doc(<NAME>) to access documentation
```

```
library(lme4)
library(pscl)
```

```
## Classes and Methods for R originally developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University (2002-2015),
## by and under the direction of Simon Jackman.
## hurdle and zeroinfl functions by Achim Zeileis.
```

```
library(nlme)
```

```
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:lme4':
##
##     lmList
##
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
library(glmmTMB)
```

DATA CLEANING

```
## LOAD IN DATA ##
lsn_16_d <- read.csv("https://raw.githubusercontent.com/nick3703/Chicago-Data/master/crime.csv")[,-1]
wealth_d<- read.csv("https://raw.githubusercontent.com/nick3703/Chicago-Data/master/wealth.csv")[,-1]
wealth_d = as.data.frame(wealth_d)

#Population by Census Block Group
pop=read.csv("https://raw.githubusercontent.com/nick3703/Chicago-Data/master/pop.csv")[,-1]
pop = as.data.frame(pop)

#Percentage Unemployed by Census Block Group
un.emp=read.csv("https://raw.githubusercontent.com/nick3703/Chicago-Data/master/unemp.csv")[,-1]
un.emp = as.data.frame(un.emp)

#Number of Young Males by Census Block Group (15-20 yr olds)
ym= read.csv("https://raw.githubusercontent.com/nick3703/Chicago-Data/master/ym.csv")[,-1]
ym=(ym-mean(ym))/sqrt(var(ym))  # normalize variable
ym = as.data.frame(ym)

# Calculate average counts of burglaries for each census block
average_counts <- rowMeans(lsn_16_d, na.rm = TRUE)

# Create a data frame with census block ID and average counts
average_counts_df <- data.frame(Average_Counts = average_counts)

# Month
```

```r
# By Month and Year
by_month = gather(lsn_16_d, factor_key = TRUE)
block = c(seq(1,552))
for (x in seq(1, 71)){
  block = append(block, c(seq(1,552)))
}
by_month$block = block
```

```r
## MERGE DATA ##
merged_df <- cbind(wealth_d, pop, un.emp, ym)
merged_df$block = c(seq(1,552))
merged_df$block = as.factor(merged_df$block)

merged_df <- cbind(merged_df, by_month)
merged_df <- merged_df[,1:7]
merged_df<-rename(merged_df,crime_count = value, year_month = key)
```

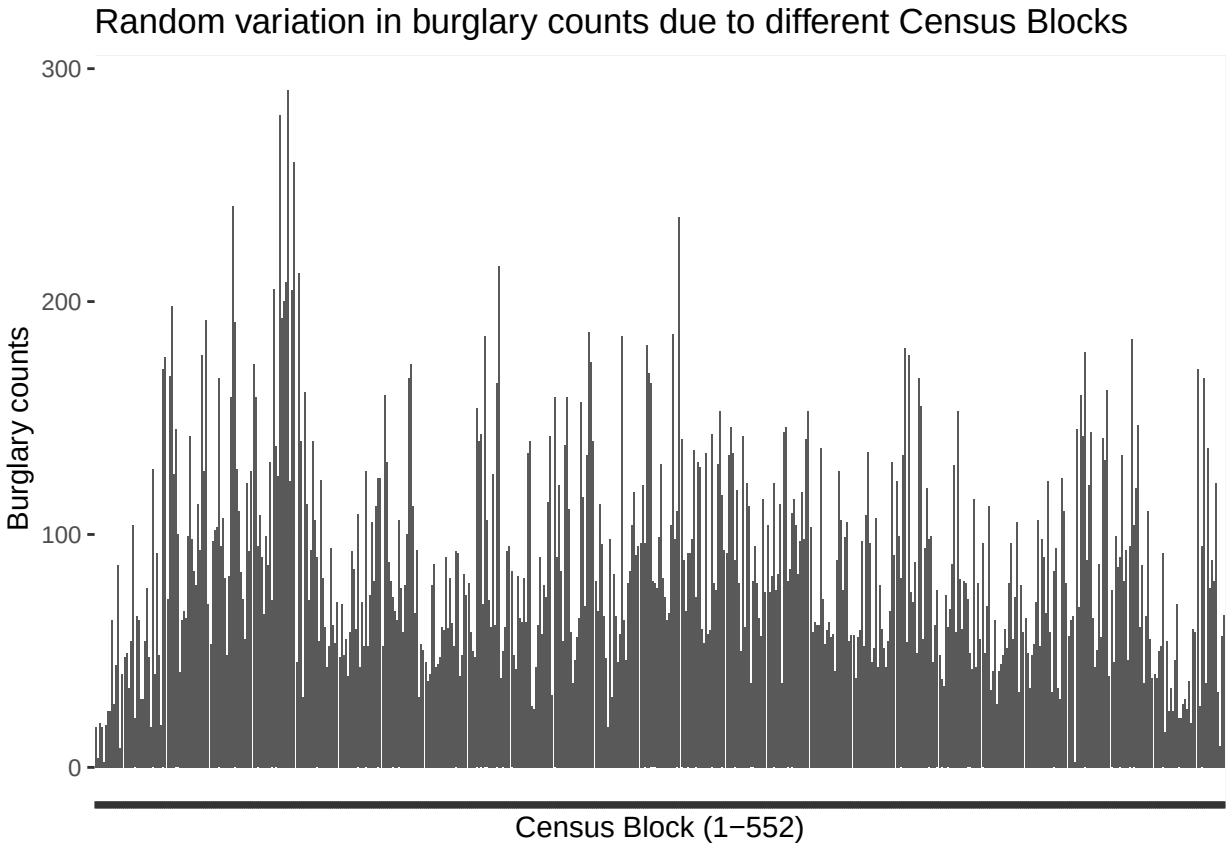
```r
test = as.character(merged_df$year_month)
merged_df$month = merged_df$year_month
merged_df$month = ifelse(substr(test, 11, 13) == '01', 'JAN', merged_df$month)
merged_df$month = ifelse(substr(test, 11, 13) == '02', 'FEB', merged_df$month)
merged_df$month = ifelse(substr(test, 11, 13) == '03', 'MAR', merged_df$month)
merged_df$month = ifelse(substr(test, 11, 13) == '04', 'APR', merged_df$month)
merged_df$month = ifelse(substr(test, 11, 13) == '05', 'MAY', merged_df$month)
merged_df$month = ifelse(substr(test, 11, 13) == '06', 'JUN', merged_df$month)
merged_df$month = ifelse(substr(test, 11, 13) == '07', 'JUL', merged_df$month)
merged_df$month = ifelse(substr(test, 11, 13) == '08', 'AUG', merged_df$month)
merged_df$month = ifelse(substr(test, 11, 13) == '09', 'SEP', merged_df$month)
merged_df$month = ifelse(substr(test, 11, 13) == '10', 'OCT', merged_df$month)
merged_df$month = ifelse(substr(test, 11, 13) == '11', 'NOV', merged_df$month)
merged_df$month = ifelse(substr(test, 11, 13) == '12', 'DEC', merged_df$month)
```

```r
merged_df$year = substr(merged_df$year_month, 7, 10)
merged_df$year = as.numeric(merged_df$year)
merged_df$year = merged_df$year-2010

# standardize population (better to not normalize pop because we are able to include it as offset)
#merged_df$pop = (mean(merged_df$pop)-merged_df$pop)/sd(merged_df$pop)
```

```r
# Split into test/train set
set.seed(1)
test_index <- sample(1:nrow(merged_df),0.2 * dim(merged_df)[1],replace = FALSE)
train_df = merged_df[-test_index,]
test_df = merged_df[test_index,]
```

```r
merged_df %>% group_by(block) %>% ggplot(aes(x = block, y = crime_count))+geom_bar(stat = 'identity')+la
      #axis.ticks.x=element_blank())
    )
```



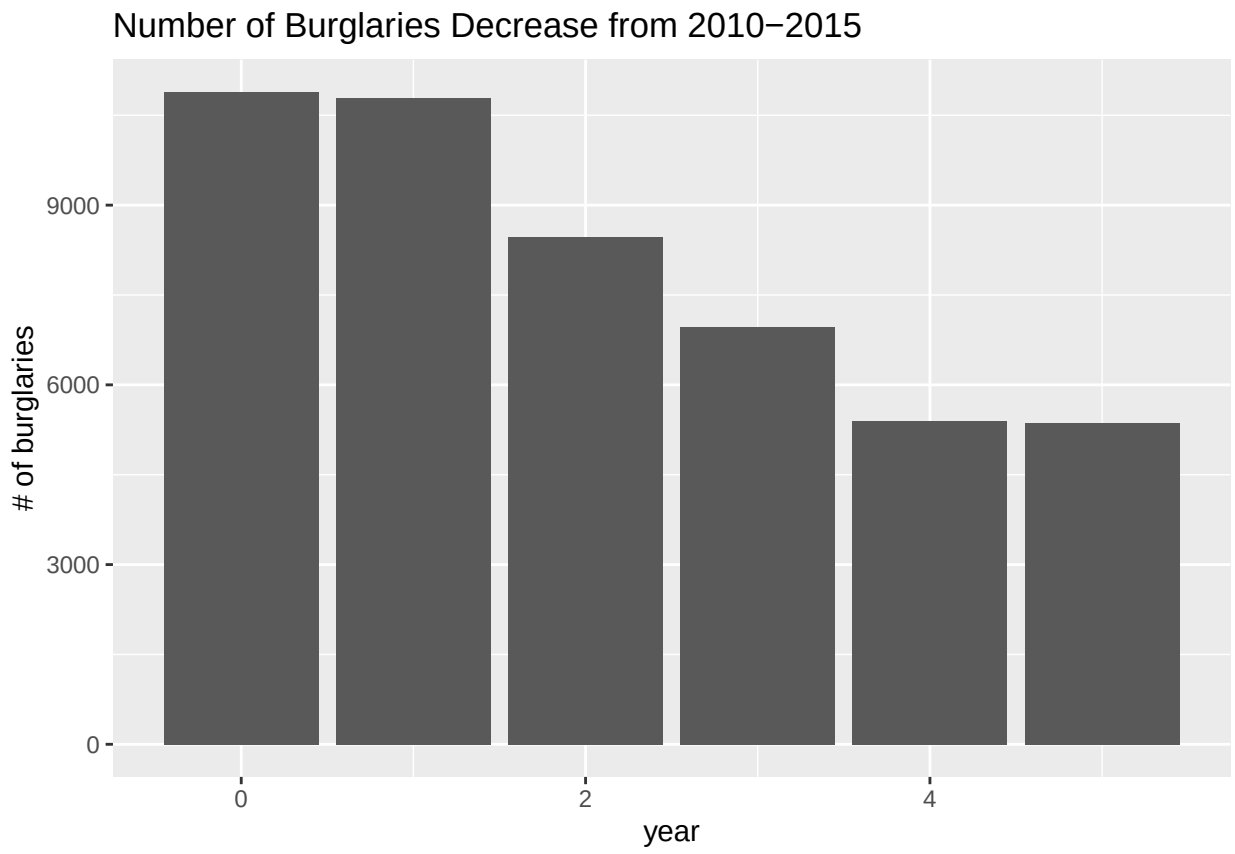

```
merged_df %>% summarize(n=n(), mean_crime = mean(crime_count), sd_crime = sd(crime_count))
```

```
##       n mean_crime sd_crime
## 1 39744   1.203603 1.480573
```

```
# No real benefit of showing summary stats within written report. Histogram is a better visual of obser
```

DATA EXPLORATION

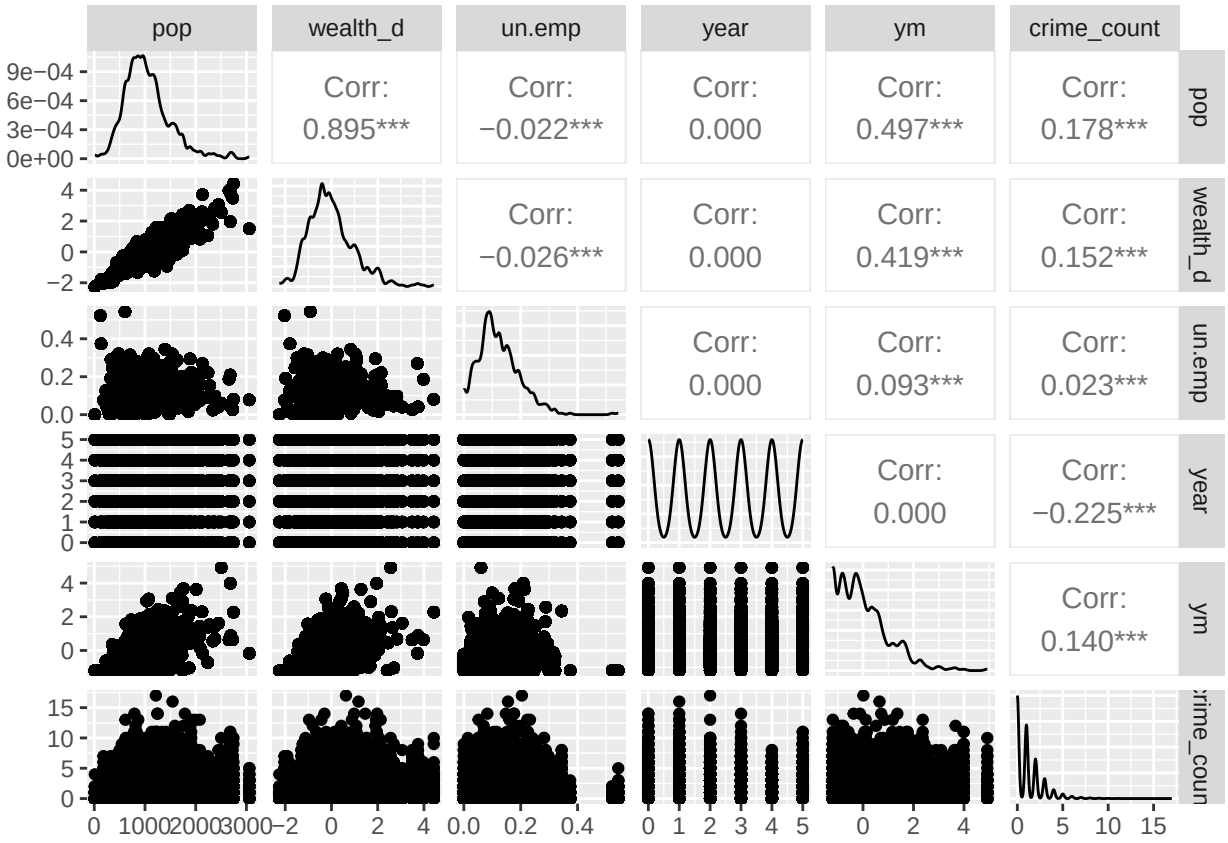
```
merged_df %>% group_by(year) %>%
  ggplot(aes(x = year, y = crime_count))+
  geom_bar(stat = 'identity')+
  labs(title = "Number of Burglaries Decrease from 2010-2015", y = '# of burglaries')
```



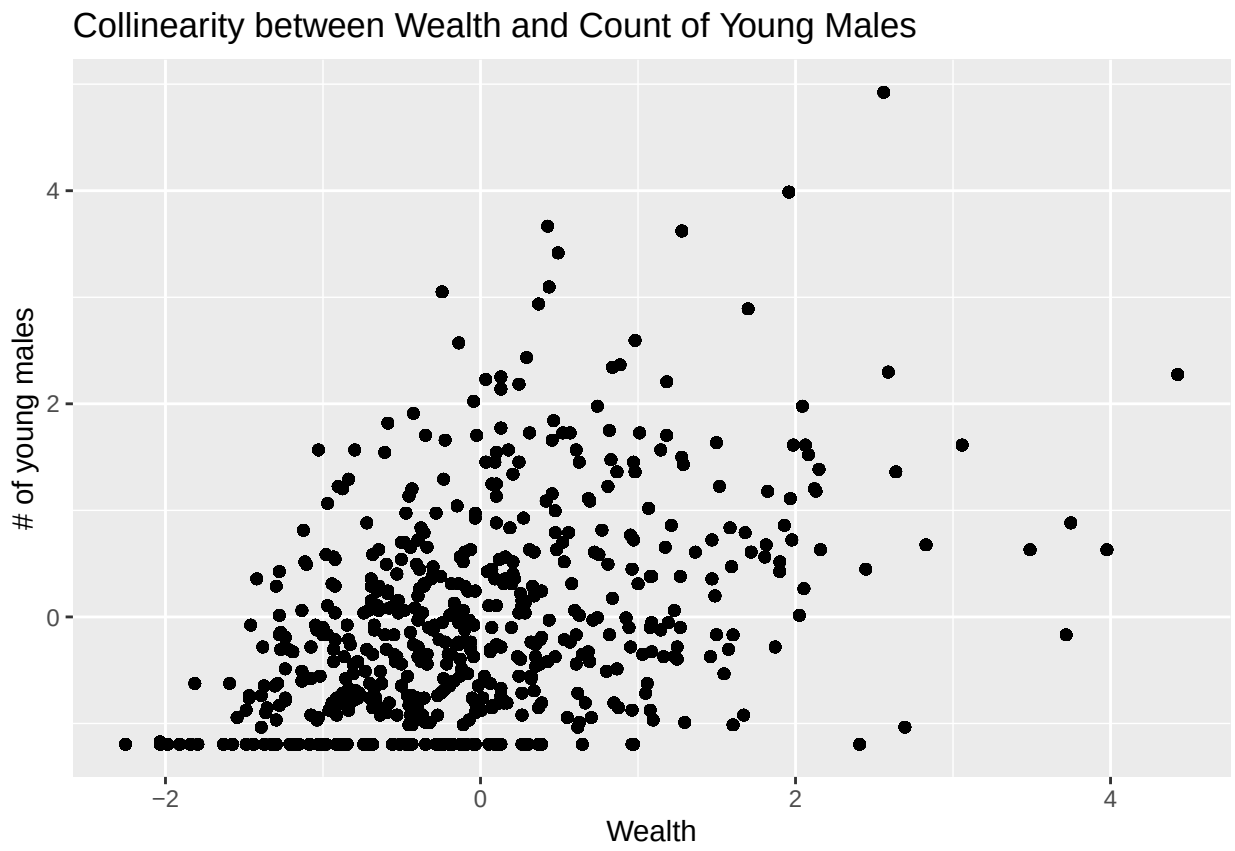

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

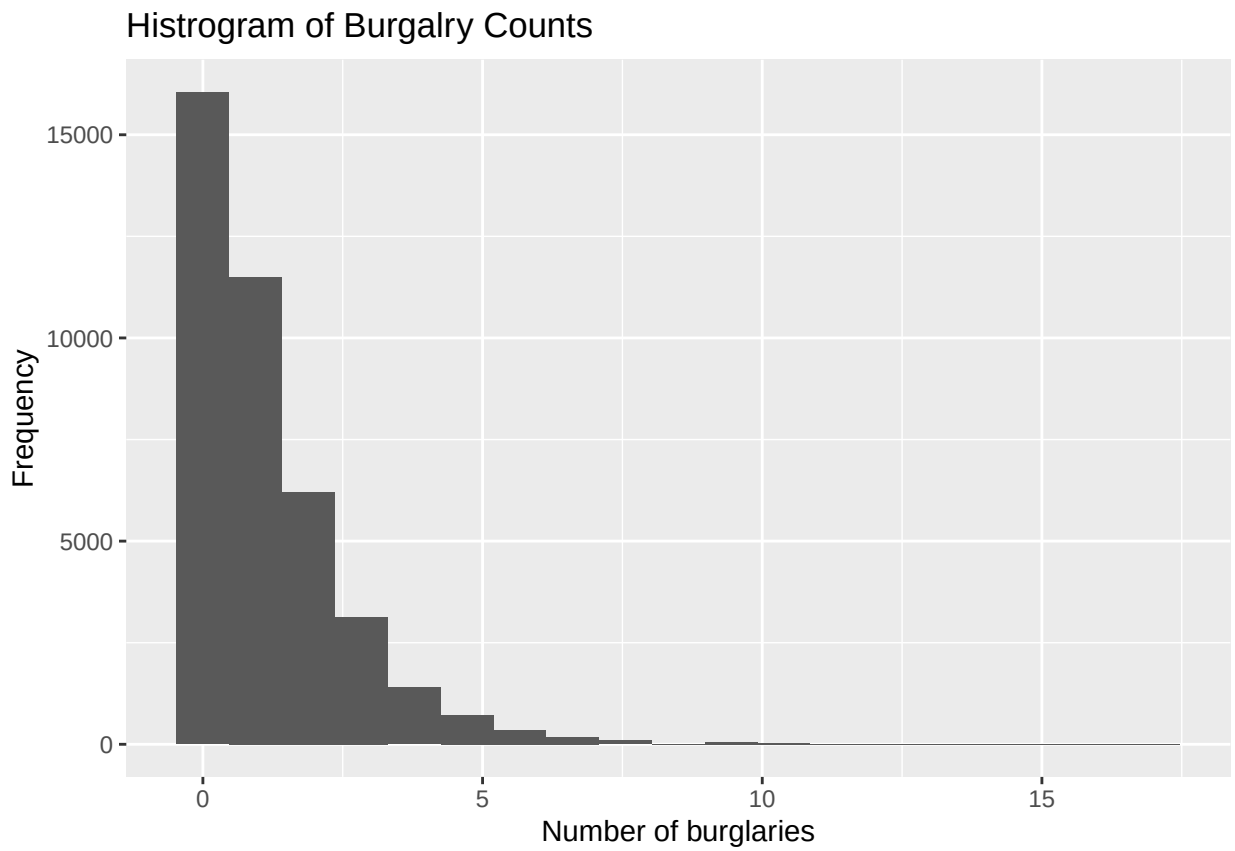
```
ggpairs(merged_df %>% select(pop, wealth_d, un.emp,year, ym, crime_count))
```



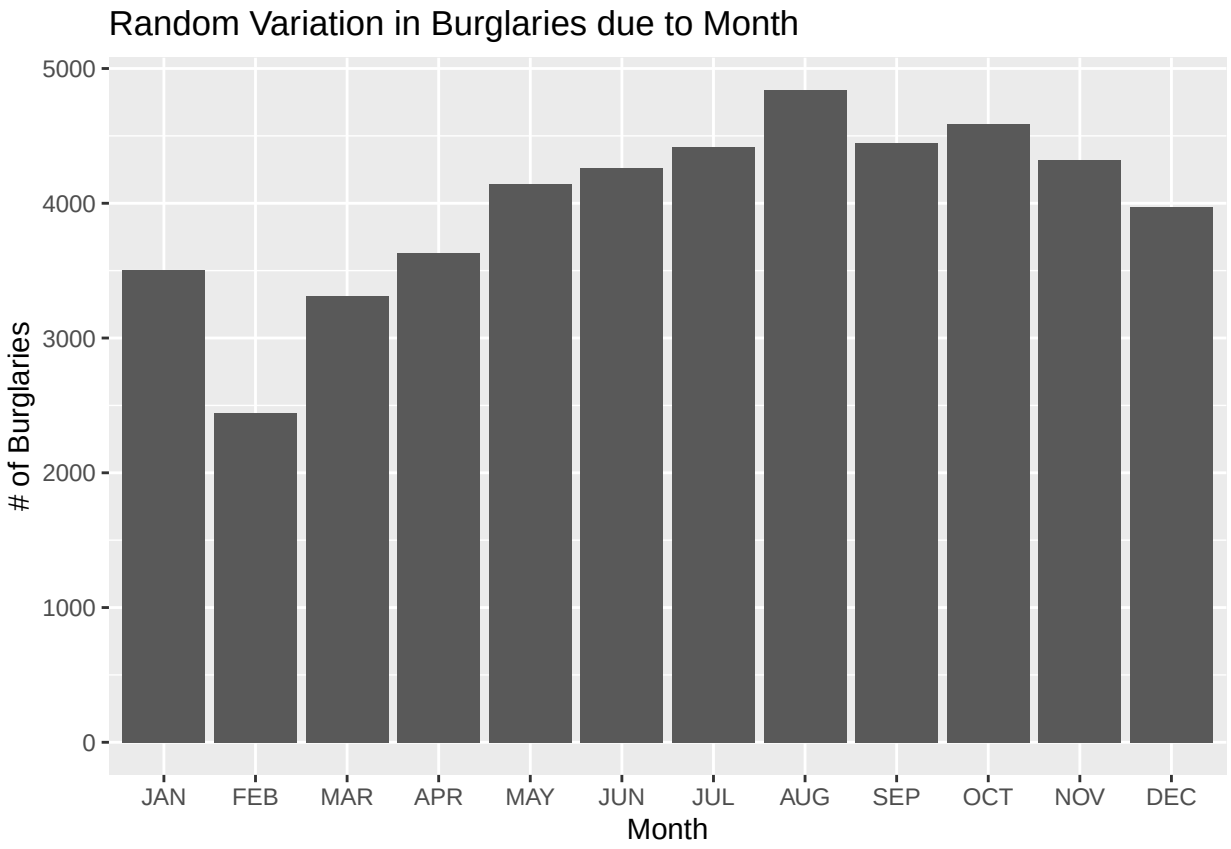

```
merged_df %>% group_by(block) %>%
  ggplot(aes(x = wealth_d, y = ym))+
  geom_point()+
  labs(title = "Collinearity between Wealth and Count of Young Males", x = "Wealth", y = '# of young ma
```




```
merged_df %>%
  ggplot(aes(x = crime_count))+
  geom_histogram(bins = 19)+
  labs(title = "Histrogram of Burgalry Counts", x = 'Number of burglaries', y = 'Frequency')
```




```
test_df = merged_df
test_df$month <- factor(merged_df$month, levels = c("JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AU
test_df %>%
  ggplot(aes(x = month, y = crime_count))+
  geom_bar(stat = "identity")+
  labs(title = "Random Variation in Burglaries due to Month", x = "Month", y = '# of Burglaries')
```



MODEL GENERATION

```
# Ideas for Models (least complex to most complex)

# 1: Poisson GLM (EXP: wealth, offset population, block, unemployed?)
mod2 = glm(crime_count ~ wealth_d + year + un.emp + offset(log(pop)), data = train_df, family = "poisso
mod1 = glm(crime_count ~ wealth_d + year + offset(log(pop)), data = train_df, family = "poisson")
mod3 = glm(crime_count ~ wealth_d + year + un.emp + month + offset(log(pop)), data = train_df, family =
anova(mod2, mod3, test = 'Chisq') # all model 3 variables are significant (deleted arbitrary likelihood
```

```
## Analysis of Deviance Table
##
## Model 1: crime_count ~ wealth_d + year + un.emp + offset(log(pop))
## Model 2: crime_count ~ wealth_d + year + un.emp + month + offset(log(pop))
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1     31792      50077
## 2     31781      49044 11   1033.2 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(glm(crime_count ~ offset(log(pop)), data = train_df, family = "poisson"), mod2, test = 'Chisq') #
```

```
## Analysis of Deviance Table
```

```
##
## Model 1: crime_count ~ offset(log(pop))
## Model 2: crime_count ~ wealth_d + year + un.emp + offset(log(pop))
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1     31795      54400
## 2     31792      50077  3   4323.1 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# goodness of fit test
1-pchisq(deviance(mod3), df.residual(mod3))   # fails the goodness of fit test
```

```
## [1] 0
```

```
summary(mod2)
```

```
##
## Call:
## glm(formula = crime_count ~ wealth_d + year + un.emp + offset(log(pop)),
##     family = "poisson", data = train_df)
##
## Coefficients:
##              Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -6.407741   0.012140 -527.800  < 2e-16 ***
## wealth_d    -0.179799   0.005092  -35.308  < 2e-16 ***
## year        -0.164278   0.003068  -53.544  < 2e-16 ***
## un.emp       0.522584   0.072123    7.246  4.3e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 54400  on 31795  degrees of freedom
## Residual deviance: 50077  on 31792  degrees of freedom
## AIC: 96932
##
## Number of Fisher Scoring iterations: 5
```

```
deviance(mod2) / df.residual(mod2)
```

```
## [1] 1.575157
```

```
sat_mod = glm(crime_count ~. -pop + offset(log(pop)), data = train_df, family = "poisson")
summary(sat_mod)
```

```
##
## Call:
## glm(formula = crime_count ~ . - pop + offset(log(pop)), family = "poisson",
##     data = train_df)
##
## Coefficients: (15 not defined because of singularities)
```

```
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)               6.45104    3.32472   1.940 0.052340 .
## wealth_d                  2.00240    0.82408   2.430 0.015105 *
## un.emp                  -57.37094   12.68595  -4.522 6.11e-06 ***
## ym                        6.42990    1.60276   4.012 6.03e-05 ***
## block2                   -1.16410    0.80604  -1.444 0.148678
## block3                   -4.71838    1.04902  -4.498 6.86e-06 ***
## block4                    2.07851    0.65836   3.157 0.001594 **
## block5                  -10.32981    1.97402  -5.233 1.67e-07 ***
## block6                   -7.63538    1.91639  -3.984 6.77e-05 ***
## block7                   -5.94161    2.73134  -2.175 0.029604 *
## block8                    2.14618    0.62428   3.438 0.000586 ***
## block9                   -3.92706    1.26461  -3.105 0.001901 **
## block10                  20.52748    4.63408   4.430 9.44e-06 ***
## block11                   6.22262    1.13356   5.489 4.03e-08 ***
## block12                 -19.08058    5.31395  -3.591 0.000330 ***
## block13                  -2.53572    0.81089  -3.127 0.001766 **
## block14                  -4.21569    0.81347  -5.182 2.19e-07 ***
## block15                   1.42249    0.47169   3.016 0.002564 **
## block16                  -8.36005    2.03519  -4.108 4.00e-05 ***
## block17                  -0.81671    0.44628  -1.830 0.067245 .
## block18                  -2.63266    1.25230  -2.102 0.035531 *
## block19                   1.14599    0.61109   1.875 0.060748 .
## block20                  13.96334    3.45887   4.037 5.41e-05 ***
## block21                   0.68959    0.19123   3.606 0.000311 ***
## block22                  10.25369    2.48662   4.124 3.73e-05 ***
## block23                  -6.17399    1.26270  -4.890 1.01e-06 ***
## block24                  -7.34347    1.82099  -4.033 5.51e-05 ***
## block25                 -16.21298    4.37122  -3.709 0.000208 ***
## block26                  -8.80840    2.41679  -3.645 0.000268 ***
## block27                  -6.53875    1.70893  -3.826 0.000130 ***
## block28                  -7.24180    1.69495  -4.273 1.93e-05 ***
## block29                 -11.94854    3.28374  -3.639 0.000274 ***
## block30                 -11.10204    2.83067  -3.922 8.78e-05 ***
## block31                 -11.22578    2.97987  -3.767 0.000165 ***
## block32                  -7.47902    2.02236  -3.698 0.000217 ***
## block33                  -6.50058    1.59099  -4.086 4.39e-05 ***
## block34                 -15.68520    4.39821  -3.566 0.000362 ***
## block35                   7.61340    1.22602   6.210 5.30e-10 ***
## block36                   1.94880    0.52864   3.686 0.000227 ***
## block37                 -20.32028    5.33453  -3.809 0.000139 ***
## block38                  -2.42105    1.09441  -2.212 0.026953 *
## block39                  -8.00061    2.40066  -3.333 0.000860 ***
## block40                  -9.96015    3.15936  -3.153 0.001618 **
## block41                  -9.14728    2.54241  -3.598 0.000321 ***
## block42                  -5.81032    1.41581  -4.104 4.06e-05 ***
## block43                 -20.65325    5.33266  -3.873 0.000108 ***
## block44                 -12.08331    3.32808  -3.631 0.000283 ***
## block45                  -1.63080    0.72440  -2.251 0.024369 *
## block46                 -16.93191    4.63815  -3.651 0.000262 ***
## block47                  10.18268    1.95250   5.215 1.84e-07 ***
## block48                 -11.63022    3.19041  -3.645 0.000267 ***
## block49                   1.19624    0.56515   2.117 0.034288 *
## block50                   2.03772    0.32721   6.228 4.74e-10 ***
```

```
## block51              -11.92029    3.35720  -3.551 0.000384 ***
## block52                6.25645    1.36503   4.583 4.57e-06 ***
## block53                4.71407    0.66130   7.128 1.02e-12 ***
## block54              -14.45284    3.93916  -3.669 0.000243 ***
## block55               -8.65386    2.93884  -2.945 0.003233 **
## block56                4.43051    0.85553   5.179 2.23e-07 ***
## block57                5.29950    1.32983   3.985 6.75e-05 ***
## block58                2.54858    0.38157   6.679 2.40e-11 ***
## block59                7.67979    1.37170   5.599 2.16e-08 ***
## block60               -5.37497    1.57715  -3.408 0.000654 ***
## block61              -10.13598    2.97956  -3.402 0.000669 ***
## block62               -1.50106    0.63786  -2.353 0.018608 *
## block63               -1.90726    0.54716  -3.486 0.000491 ***
## block64               -4.49273    1.20958  -3.714 0.000204 ***
## block65                2.36625    0.61003   3.879 0.000105 ***
## block66               -2.22015    0.60106  -3.694 0.000221 ***
## block67               -0.86689    0.51900  -1.670 0.094854 .
## block68               -6.78525    2.05104  -3.308 0.000939 ***
## block69              -21.21973    5.72663  -3.705 0.000211 ***
## block70               -7.29256    2.34258  -3.113 0.001852 **
## block71                6.48449    1.26738   5.116 3.11e-07 ***
## block72               -1.58497    0.53483  -2.964 0.003042 **
## block73              -17.03201    4.36080  -3.906 9.39e-05 ***
## block74               -8.43839    2.35031  -3.590 0.000330 ***
## block75                2.43271    0.66592   3.653 0.000259 ***
## block76                0.56036    1.00229   0.559 0.576109
## block77               -2.71674    0.91253  -2.977 0.002910 **
## block78               -4.63212    1.47028  -3.151 0.001630 **
## block79              -15.88279    4.60489  -3.449 0.000562 ***
## block80                4.74263    0.97874   4.846 1.26e-06 ***
## block81               -9.27669    2.84606  -3.259 0.001116 **
## block82               -0.85619    0.42238  -2.027 0.042655 *
## block83               -3.51984    0.93560  -3.762 0.000168 ***
## block84               -5.80782    1.76729  -3.286 0.001015 **
## block85               -3.17038    1.09114  -2.906 0.003666 **
## block86              -10.69734    2.92330  -3.659 0.000253 ***
## block87                6.30738    1.40264   4.497 6.90e-06 ***
## block88               -4.71013    1.94738  -2.419 0.015576 *
## block89               -8.60136    2.83124  -3.038 0.002381 **
## block90                5.20383    1.12384   4.630 3.65e-06 ***
## block91               -9.96118    3.26374  -3.052 0.002273 **
## block92              -12.13882    3.58991  -3.381 0.000721 ***
## block93               -9.94880    2.97985  -3.339 0.000842 ***
## block94                2.69976    0.21010  12.850  < 2e-16 ***
## block95              -19.60207    5.34070  -3.670 0.000242 ***
## block96              -11.26504    3.22448  -3.494 0.000477 ***
## block97               -4.88967    1.36538  -3.581 0.000342 ***
## block98                1.00702    0.80367   1.253 0.210196
## block99               -1.74946    0.44569  -3.925 8.66e-05 ***
## block100             -11.37259    3.71885  -3.058 0.002227 **
## block101             -11.78066    3.15386  -3.735 0.000187 ***
## block102              11.77758    2.85976   4.118 3.82e-05 ***
## block103               4.34103    0.50267   8.636  < 2e-16 ***
## block104               0.16677    0.71721   0.233 0.816127
```

```
## block105                0.21333    0.42077   0.507 0.612146
## block106               -0.50479    0.22610  -2.233 0.025576 *
## block107                1.71027    0.25462   6.717 1.86e-11 ***
## block108              -13.47561    3.84408  -3.506 0.000456 ***
## block109               -4.54877    1.82024  -2.499 0.012455 *
## block110               -5.55129    1.50473  -3.689 0.000225 ***
## block111              -15.76876    4.49040  -3.512 0.000445 ***
## block112              -16.05111    4.46979  -3.591 0.000329 ***
## block113               -0.15860    0.36584  -0.434 0.664640
## block114               -9.59363    2.28067  -4.207 2.59e-05 ***
## block115              -11.51117    2.89807  -3.972 7.13e-05 ***
## block116              -13.40955    4.01143  -3.343 0.000829 ***
## block117               -9.66932    2.49310  -3.878 0.000105 ***
## block118               -0.66025    0.23759  -2.779 0.005454 **
## block119                3.62208    0.62646   5.782 7.39e-09 ***
## block120               -6.42156    1.58600  -4.049 5.15e-05 ***
## block121               -4.56354    1.89360  -2.410 0.015954 *
## block122               -2.09933    0.87367  -2.403 0.016267 *
## block123               -2.18669    0.71088  -3.076 0.002098 **
## block124               -4.04204    1.26236  -3.202 0.001365 **
## block125              -13.19736    3.55827  -3.709 0.000208 ***
## block126              -16.56822    4.47338  -3.704 0.000212 ***
## block127                8.13766    1.73292   4.696 2.65e-06 ***
## block128               -4.96914    1.38757  -3.581 0.000342 ***
## block129               -3.48956    1.40811  -2.478 0.013205 *
## block130               -0.78598    0.51582  -1.524 0.127568
## block131               -5.67879    1.52170  -3.732 0.000190 ***
## block132                4.52898    1.01709   4.453 8.47e-06 ***
## block133               -6.45348    1.90737  -3.383 0.000716 ***
## block134              -10.89613    2.65311  -4.107 4.01e-05 ***
## block135                5.63470    1.16847   4.822 1.42e-06 ***
## block136               -5.98201    2.01016  -2.976 0.002921 **
## block137               -6.42969    1.87008  -3.438 0.000586 ***
## block138              -14.74698    3.87797  -3.803 0.000143 ***
## block139              -22.26684    5.82739  -3.821 0.000133 ***
## block140               -2.39278    0.89603  -2.670 0.007575 **
## block141                3.84051    0.70077   5.480 4.24e-08 ***
## block142               -5.04712    1.94632  -2.593 0.009510 **
## block143              -12.23532    3.56664  -3.430 0.000602 ***
## block144               -6.59435    1.83140  -3.601 0.000317 ***
## block145               -5.49971    1.44511  -3.806 0.000141 ***
## block146              -12.79898    3.24459  -3.945 7.99e-05 ***
## block147                0.90334    0.36015   2.508 0.012133 *
## block148               -7.11214    1.98957  -3.575 0.000351 ***
## block149              -20.20621    5.18018  -3.901 9.59e-05 ***
## block150               -8.44283    2.13524  -3.954 7.68e-05 ***
## block151                0.52675    0.27688   1.902 0.057112 .
## block152               -2.98417    1.01080  -2.952 0.003154 **
## block153               -5.95826    1.74792  -3.409 0.000653 ***
## block154              -11.52939    3.26673  -3.529 0.000417 ***
## block155               -5.65509    2.00504  -2.820 0.004796 **
## block156              -13.73628    4.09099  -3.358 0.000786 ***
## block157               -6.64965    1.97654  -3.364 0.000767 ***
## block158               -5.26192    1.67928  -3.133 0.001728 **
```

```
## block159              -3.10844    0.67408  -4.611 4.00e-06 ***
## block160               0.79472    0.22695   3.502 0.000462 ***
## block161              -2.96435    1.06922  -2.772 0.005564 **
## block162               0.01683    0.59589   0.028 0.977473
## block163              -4.40077    0.98116  -4.485 7.28e-06 ***
## block164               8.13811    1.82028   4.471 7.79e-06 ***
## block165             -13.33750    3.87481  -3.442 0.000577 ***
## block166               7.82224    1.48179   5.279 1.30e-07 ***
## block167              -6.49130    1.89204  -3.431 0.000602 ***
## block168              -1.06750    0.63191  -1.689 0.091157 .
## block169              -1.51786    1.08517  -1.399 0.161894
## block170             -20.81305    5.49241  -3.789 0.000151 ***
## block171              -5.06177    1.60534  -3.153 0.001616 **
## block172              -0.89820    0.46319  -1.939 0.052483 .
## block173             -13.79277    3.48007  -3.963 7.39e-05 ***
## block174             -15.84323    4.16068  -3.808 0.000140 ***
## block175             -11.87070    3.26087  -3.640 0.000272 ***
## block176             -19.54776    4.99013  -3.917 8.96e-05 ***
## block177             -10.64298    2.86708  -3.712 0.000206 ***
## block178              -8.56876    2.62301  -3.267 0.001088 **
## block179             -20.26540    5.00000  -4.053 5.05e-05 ***
## block180               0.54783    0.23158   2.366 0.018000 *
## block181             -15.92084    4.33418  -3.673 0.000239 ***
## block182              -7.20431    2.05266  -3.510 0.000449 ***
## block183              -7.80162    2.16239  -3.608 0.000309 ***
## block184              -2.86407    0.91912  -3.116 0.001833 **
## block185              -6.11574    1.59660  -3.830 0.000128 ***
## block186               1.85417    0.56282   3.294 0.000986 ***
## block187             -11.98569    3.67478  -3.262 0.001108 **
## block188              -7.11293    2.01651  -3.527 0.000420 ***
## block189              -0.30396    0.96122  -0.316 0.751833
## block190              -7.39980    1.81863  -4.069 4.72e-05 ***
## block191             -22.84943    6.62887  -3.447 0.000567 ***
## block192               5.83135    0.99738   5.847 5.02e-09 ***
## block193              -4.25631    1.36932  -3.108 0.001881 **
## block194             -11.92820    3.00492  -3.970 7.20e-05 ***
## block195              -0.88693    0.75228  -1.179 0.238403
## block196              -2.47847    0.95313  -2.600 0.009313 **
## block197             -12.67809    3.82150  -3.318 0.000908 ***
## block198             -26.16485    7.01310  -3.731 0.000191 ***
## block199             -14.55718    3.52212  -4.133 3.58e-05 ***
## block200             -16.28142    4.05274  -4.017 5.88e-05 ***
## block201              -0.80932    0.51377  -1.575 0.115197
## block202              -2.85601    1.32735  -2.152 0.031424 *
## block203              -8.06552    2.15766  -3.738 0.000185 ***
## block204             -13.49972    3.70885  -3.640 0.000273 ***
## block205               1.50275    0.39900   3.766 0.000166 ***
## block206              -0.80119    0.27442  -2.920 0.003505 **
## block207              -1.87897    0.97398  -1.929 0.053711 .
## block208              -7.69223    2.04340  -3.764 0.000167 ***
## block209              -7.46202    1.94201  -3.842 0.000122 ***
## block210              -6.57863    1.75819  -3.742 0.000183 ***
## block211              -3.16560    0.64564  -4.903 9.43e-07 ***
## block212             -12.90791    3.42572  -3.768 0.000165 ***
```

```
## block213              -11.29003    2.91270  -3.876 0.000106 ***
## block214                1.06354    0.39511   2.692 0.007107 **
## block215                4.33695    1.10968   3.908 9.30e-05 ***
## block216               -2.41670    0.64630  -3.739 0.000185 ***
## block217              -13.98287    4.00396  -3.492 0.000479 ***
## block218               -0.36760    0.78980  -0.465 0.641621
## block219               -7.74594    2.15213  -3.599 0.000319 ***
## block220               -8.18539    2.24468  -3.647 0.000266 ***
## block221               -9.29755    2.51051  -3.703 0.000213 ***
## block222               -9.04206    2.75813  -3.278 0.001044 **
## block223               -2.63679    1.08874  -2.422 0.015441 *
## block224              -13.25118    3.13474  -4.227 2.37e-05 ***
## block225              -24.33428    7.02148  -3.466 0.000529 ***
## block226               -8.23429    2.31574  -3.556 0.000377 ***
## block227               -0.22639    0.66449  -0.341 0.733336
## block228              -11.10243    2.95179  -3.761 0.000169 ***
## block229              -14.81512    3.66411  -4.043 5.27e-05 ***
## block230              -30.31185    7.85582  -3.859 0.000114 ***
## block231              -13.42550    3.86076  -3.477 0.000506 ***
## block232               -1.70296    0.68367  -2.491 0.012741 *
## block233               -2.78607    0.95801  -2.908 0.003635 **
## block234                0.40093    0.27807   1.442 0.149356
## block235              -20.12907    5.35614  -3.758 0.000171 ***
## block236               -5.43433    1.60075  -3.395 0.000687 ***
## block237               -8.72607    2.49617  -3.496 0.000473 ***
## block238               -2.35500    0.91710  -2.568 0.010232 *
## block239              -19.45398    5.24260  -3.711 0.000207 ***
## block240               -7.03668    1.81748  -3.872 0.000108 ***
## block241              -22.62358    5.94285  -3.807 0.000141 ***
## block242              -20.52125    5.71268  -3.592 0.000328 ***
## block243              -16.16975    4.67941  -3.456 0.000549 ***
## block244              -46.80602   12.19197  -3.839 0.000123 ***
## block245               -4.84498    1.76086  -2.751 0.005933 **
## block246               -8.52128    2.17696  -3.914 9.07e-05 ***
## block247               -1.98614    1.02226  -1.943 0.052030 .
## block248               -9.33921    2.81627  -3.316 0.000913 ***
## block249                2.76640    0.47466   5.828 5.60e-09 ***
## block250               -3.07395    0.94694  -3.246 0.001170 **
## block251                1.84781    0.63747   2.899 0.003748 **
## block252                4.40192    0.82179   5.356 8.49e-08 ***
## block253               -4.42344    1.78050  -2.484 0.012978 *
## block254               -4.86412    1.56546  -3.107 0.001889 **
## block255               -4.91176    1.38688  -3.542 0.000398 ***
## block256               15.56486    3.49737   4.450 8.57e-06 ***
## block257                2.66748    0.63417   4.206 2.60e-05 ***
## block258              -17.17413    4.93206  -3.482 0.000497 ***
## block259               -1.54320    0.66703  -2.314 0.020692 *
## block260               27.35745    6.11007   4.477 7.55e-06 ***
## block261              -10.51973    2.74848  -3.827 0.000129 ***
## block262              -18.29406    4.88416  -3.746 0.000180 ***
## block263               -5.67720    1.70884  -3.322 0.000893 ***
## block264               -4.46418    1.09817  -4.065 4.80e-05 ***
## block265               -9.73397    2.63740  -3.691 0.000224 ***
## block266               -4.26423    1.35471  -3.148 0.001646 **
```

```
## block267              -8.38225    2.35209  -3.564 0.000366 ***
## block268             -17.63315    4.88440  -3.610 0.000306 ***
## block269               0.26232    0.19526   1.343 0.179127
## block270             -15.39838    4.66918  -3.298 0.000974 ***
## block271             -10.22981    3.06973  -3.332 0.000861 ***
## block272             -19.89091    5.30756  -3.748 0.000178 ***
## block273               3.33140    0.54555   6.107 1.02e-09 ***
## block274               3.11242    0.81760   3.807 0.000141 ***
## block275             -11.45966    3.00649  -3.812 0.000138 ***
## block276             -14.40445    4.16444  -3.459 0.000542 ***
## block277             -12.70668    4.01373  -3.166 0.001547 **
## block278              -8.52226    2.58259  -3.300 0.000967 ***
## block279              -8.81833    2.24923  -3.921 8.83e-05 ***
## block280             -17.04063    4.37421  -3.896 9.79e-05 ***
## block281              -2.78253    1.12398  -2.476 0.013302 *
## block282              -3.86291    1.11055  -3.478 0.000504 ***
## block283              -9.68616    3.40223  -2.847 0.004413 **
## block284              -5.62682    1.72456  -3.263 0.001103 **
## block285               3.14742    0.48884   6.438 1.21e-10 ***
## block286             -30.62626    8.43203  -3.632 0.000281 ***
## block287              -4.97909    1.75400  -2.839 0.004530 **
## block288              -8.23520    2.48626  -3.312 0.000925 ***
## block289              -0.91025    0.63977  -1.423 0.154803
## block290             -14.29238    3.76375  -3.797 0.000146 ***
## block291               3.93343    0.90485   4.347 1.38e-05 ***
## block292              -1.23590    0.55185  -2.240 0.025121 *
## block293               3.59783    0.53975   6.666 2.63e-11 ***
## block294             -12.36096    3.44770  -3.585 0.000337 ***
## block295               5.28237    0.91002   5.805 6.45e-09 ***
## block296             -14.47882    4.11320  -3.520 0.000431 ***
## block297              -4.42570    1.13330  -3.905 9.42e-05 ***
## block298               7.36101    1.60421   4.589 4.46e-06 ***
## block299             -26.89439    7.02828  -3.827 0.000130 ***
## block300               2.75119    0.60657   4.536 5.74e-06 ***
## block301              12.22446    2.68267   4.557 5.19e-06 ***
## block302              -3.03051    1.07895  -2.809 0.004973 **
## block303              -9.47046    2.55038  -3.713 0.000205 ***
## block304              -1.63481    1.00782  -1.622 0.104777
## block305             -13.92352    4.30468  -3.235 0.001219 **
## block306              -3.19981    1.49228  -2.144 0.032013 *
## block307               6.64485    1.24953   5.318 1.05e-07 ***
## block308              -7.54187    2.53344  -2.977 0.002911 **
## block309               3.72769    0.66534   5.603 2.11e-08 ***
## block310              -1.13672    1.00101  -1.136 0.256135
## block311             -11.54556    3.44911  -3.347 0.000816 ***
## block312             -26.94042    7.32266  -3.679 0.000234 ***
## block313              -0.39443    0.42500  -0.928 0.353369
## block314              -8.24618    2.28314  -3.612 0.000304 ***
## block315              -4.14237    1.36190  -3.042 0.002353 **
## block316               0.65207    0.45050   1.447 0.147774
## block317              -8.19091    2.39123  -3.425 0.000614 ***
## block318               1.15635    0.48078   2.405 0.016165 *
## block319              -2.04211    0.91898  -2.222 0.026273 *
## block320              -5.65506    1.53978  -3.673 0.000240 ***
```

```
## block321              -4.55119    1.05903  -4.298 1.73e-05 ***
## block322              -0.95675    0.50917  -1.879 0.060239 .
## block323             -14.57772    4.10233  -3.554 0.000380 ***
## block324              -5.61354    1.76662  -3.178 0.001485 **
## block325              -3.35483    1.00595  -3.335 0.000853 ***
## block326              -7.45547    1.83275  -4.068 4.74e-05 ***
## block327              -7.94805    2.08871  -3.805 0.000142 ***
## block328              -9.55693    2.53569  -3.769 0.000164 ***
## block329               2.52748    0.60492   4.178 2.94e-05 ***
## block330               5.59126    1.23570   4.525 6.05e-06 ***
## block331              -8.51187    2.29799  -3.704 0.000212 ***
## block332              -1.88264    0.80410  -2.341 0.019216 *
## block333               5.23461    1.13943   4.594 4.35e-06 ***
## block334              -2.56361    0.78225  -3.277 0.001048 **
## block335              -6.79084    2.25213  -3.015 0.002567 **
## block336              -5.29930    1.07582  -4.926 8.40e-07 ***
## block337             -16.65682    4.50091  -3.701 0.000215 ***
## block338               5.47726    1.01219   5.411 6.26e-08 ***
## block339               0.49562    0.30394   1.631 0.102963
## block340               3.26610    0.85008   3.842 0.000122 ***
## block341              -4.63282    1.30825  -3.541 0.000398 ***
## block342              -3.83655    1.23103  -3.117 0.001830 **
## block343              -4.93321    1.56806  -3.146 0.001655 **
## block344               9.07862    1.88503   4.816 1.46e-06 ***
## block345             -10.41130    2.83256  -3.676 0.000237 ***
## block346               1.47280    0.30787   4.784 1.72e-06 ***
## block347              -6.25754    1.75981  -3.556 0.000377 ***
## block348               5.26819    0.80646   6.532 6.47e-11 ***
## block349               2.25965    0.30989   7.292 3.06e-13 ***
## block350              -1.62572    0.89849  -1.809 0.070390 .
## block351              -3.74346    1.13097  -3.310 0.000933 ***
## block352              -3.74062    0.85288  -4.386 1.16e-05 ***
## block353              -5.07810    1.52807  -3.323 0.000890 ***
## block354              -4.69317    1.37907  -3.403 0.000666 ***
## block355              -5.15303    1.38154  -3.730 0.000192 ***
## block356              -6.27983    1.68147  -3.735 0.000188 ***
## block357             -10.76338    2.60327  -4.135 3.56e-05 ***
## block358              -5.06487    1.14375  -4.428 9.50e-06 ***
## block359               9.17258    1.99293   4.603 4.17e-06 ***
## block360               7.48238    1.64564   4.547 5.45e-06 ***
## block361               4.79913    1.04884   4.576 4.75e-06 ***
## block362              -3.39258    1.16991  -2.900 0.003733 **
## block363              -7.05273    1.96037  -3.598 0.000321 ***
## block364             -16.59234    4.57528  -3.627 0.000287 ***
## block365              -0.64378    0.49425  -1.303 0.192731
## block366              -1.57699    0.67798  -2.326 0.020017 *
## block367              -4.10782    1.10627  -3.713 0.000205 ***
## block368             -19.16642    5.29192  -3.622 0.000293 ***
## block369              -7.23534    1.78782  -4.047 5.19e-05 ***
## block370              -5.07118    1.26737  -4.001 6.30e-05 ***
## block371             -17.58288    4.36346  -4.030 5.59e-05 ***
## block372              -6.97790    1.90775  -3.658 0.000255 ***
## block373              -1.32139    0.38381  -3.443 0.000576 ***
## block374              -3.95774    1.11129  -3.561 0.000369 ***
```

```
## block375              -8.81020    2.58199  -3.412 0.000644 ***
## block376               4.58049    1.02299   4.478 7.55e-06 ***
## block377             -27.94735    7.61581  -3.670 0.000243 ***
## block378             -11.78797    3.18568  -3.700 0.000215 ***
## block379             -29.38895    7.71863  -3.808 0.000140 ***
## block380             -24.22051    6.02488  -4.020 5.82e-05 ***
## block381             -28.77206    7.64368  -3.764 0.000167 ***
## block382              -3.58536    1.44486  -2.481 0.013084 *
## block383              -1.68725    0.43664  -3.864 0.000111 ***
## block384             -23.20931    6.36909  -3.644 0.000268 ***
## block385             -18.99672    5.19574  -3.656 0.000256 ***
## block386              -8.09440    2.19423  -3.689 0.000225 ***
## block387             -12.40380    3.15718  -3.929 8.54e-05 ***
## block388             -20.52933    5.61984  -3.653 0.000259 ***
## block389             -12.10492    3.38811  -3.573 0.000353 ***
## block390              -2.19119    1.26902  -1.727 0.084227 .
## block391             -17.50700    4.70598  -3.720 0.000199 ***
## block392             -10.87684    3.09640  -3.513 0.000444 ***
## block393               9.19574    1.76513   5.210 1.89e-07 ***
## block394               1.44159    0.39633   3.637 0.000275 ***
## block395             -13.36679    3.94529  -3.388 0.000704 ***
## block396             -27.05549    7.23958  -3.737 0.000186 ***
## block397               4.93361    1.03042   4.788 1.68e-06 ***
## block398              -0.42539    0.85637  -0.497 0.619376
## block399               1.50263    0.20470   7.341 2.13e-13 ***
## block400               5.18685    1.16410   4.456 8.36e-06 ***
## block401             -22.41116    5.94676  -3.769 0.000164 ***
## block402             -10.02452    2.81200  -3.565 0.000364 ***
## block403              -2.32530    1.26586  -1.837 0.066220 .
## block404              -6.42152    2.27545  -2.822 0.004771 **
## block405               0.67873    0.33292   2.039 0.041479 *
## block406              -8.21405    2.36425  -3.474 0.000512 ***
## block407             -16.61028    4.45413  -3.729 0.000192 ***
## block408              -9.27451    2.64697  -3.504 0.000459 ***
## block409               2.13023    0.67061   3.177 0.001490 **
## block410               0.05744    0.47470   0.121 0.903691
## block411             -13.82341    3.82991  -3.609 0.000307 ***
## block412              -2.14145    0.79096  -2.707 0.006781 **
## block413               9.06835    2.09100   4.337 1.45e-05 ***
## block414              -6.22575    1.57962  -3.941 8.10e-05 ***
## block415              -8.21403    2.19344  -3.745 0.000181 ***
## block416             -12.33294    3.31408  -3.721 0.000198 ***
## block417               3.85408    0.99154   3.887 0.000102 ***
## block418              -4.03985    0.99428  -4.063 4.84e-05 ***
## block419             -10.82090    2.80131  -3.863 0.000112 ***
## block420             -31.02458    8.21776  -3.775 0.000160 ***
## block421              -5.12531    1.38040  -3.713 0.000205 ***
## block422              -3.41421    1.84238  -1.853 0.063861 .
## block423               1.91795    0.30473   6.294 3.09e-10 ***
## block424             -14.66760    4.08977  -3.586 0.000335 ***
## block425              -8.61858    2.24547  -3.838 0.000124 ***
## block426               0.42241    0.54524   0.775 0.438502
## block427              -1.98692    0.94362  -2.106 0.035236 *
## block428               0.85138    0.28120   3.028 0.002465 **
```

```
## block429              -2.49977    1.08972  -2.294 0.021793 *
## block430              -9.53973    3.35321  -2.845 0.004442 **
## block431             -11.70522    3.06689  -3.817 0.000135 ***
## block432              -7.86620    2.21579  -3.550 0.000385 ***
## block433              -4.58509    1.36812  -3.351 0.000804 ***
## block434             -22.03062    5.85284  -3.764 0.000167 ***
## block435              -7.11917    1.81011  -3.933 8.39e-05 ***
## block436               4.88184    0.71112   6.865 6.65e-12 ***
## block437              -7.41083    2.17067  -3.414 0.000640 ***
## block438              -6.84748    1.82285  -3.756 0.000172 ***
## block439             -18.76143    4.74322  -3.955 7.64e-05 ***
## block440              -9.93635    2.65580  -3.741 0.000183 ***
## block441              -1.39464    0.47146  -2.958 0.003095 **
## block442              -3.38739    0.97555  -3.472 0.000516 ***
## block443               2.58152    0.39210   6.584 4.59e-11 ***
## block444             -11.38015    3.05054  -3.731 0.000191 ***
## block445              -8.70325    2.35588  -3.694 0.000221 ***
## block446              -5.93147    1.51364  -3.919 8.90e-05 ***
## block447             -16.16450    4.54626  -3.556 0.000377 ***
## block448              -9.42756    2.67707  -3.522 0.000429 ***
## block449              -6.60212    1.81492  -3.638 0.000275 ***
## block450             -21.74808    5.86112  -3.711 0.000207 ***
## block451             -17.09759    4.88275  -3.502 0.000462 ***
## block452              -2.13966    0.52381  -4.085 4.41e-05 ***
## block453             -21.78887    5.59448  -3.895 9.83e-05 ***
## block454              -2.74347    1.01106  -2.713 0.006658 **
## block455             -11.17710    3.07440  -3.636 0.000277 ***
## block456              -1.63684    0.52365  -3.126 0.001773 **
## block457              -5.43038    1.39419  -3.895 9.82e-05 ***
## block458             -10.57344    2.66768  -3.964 7.38e-05 ***
## block459               2.82165    0.66032   4.273 1.93e-05 ***
## block460              -9.17119    2.30999  -3.970 7.18e-05 ***
## block461             -22.07016    5.70121  -3.871 0.000108 ***
## block462              -0.24655    0.25846  -0.954 0.340118
## block463              -1.11009    0.42570  -2.608 0.009116 **
## block464              -4.06362    1.75813  -2.311 0.020815 *
## block465               3.90468    0.65233   5.986 2.15e-09 ***
## block466              -8.61960    2.43260  -3.543 0.000395 ***
## block467              -7.51114    2.23857  -3.355 0.000793 ***
## block468             -11.97274    2.91024  -4.114 3.89e-05 ***
## block469               1.93154    0.57838   3.340 0.000839 ***
## block470             -10.45651    3.09442  -3.379 0.000727 ***
## block471               1.56657    0.62842   2.493 0.012672 *
## block472              -7.51699    1.65771  -4.535 5.77e-06 ***
## block473              -4.57019    1.73554  -2.633 0.008456 **
## block474             -12.75989    3.74427  -3.408 0.000655 ***
## block475              -8.77371    2.44438  -3.589 0.000332 ***
## block476              -1.35371    0.36884  -3.670 0.000242 ***
## block477              -6.31803    1.72958  -3.653 0.000259 ***
## block478               4.47265    0.88025   5.081 3.75e-07 ***
## block479             -22.53481    5.38254  -4.187 2.83e-05 ***
## block480             -32.41105    9.16724  -3.536 0.000407 ***
## block481             -19.03186    4.84259  -3.930 8.49e-05 ***
## block482             -14.94532    4.33147  -3.450 0.000560 ***
```

```
## block483              -9.04518    2.45533  -3.684 0.000230 ***
## block484              -4.38417    2.30799  -1.900 0.057491 .
## block485              -2.03670    1.03272  -1.972 0.048590 *
## block486              -5.86578    1.72889  -3.393 0.000692 ***
## block487             -22.58120    6.30552  -3.581 0.000342 ***
## block488               1.36258    0.49386   2.759 0.005797 **
## block489              -5.83746    1.63575  -3.569 0.000359 ***
## block490              -6.86505    2.29911  -2.986 0.002827 **
## block491             -14.66790    3.96849  -3.696 0.000219 ***
## block492              -1.60944    0.49044  -3.282 0.001032 **
## block493              -3.92291    1.59808  -2.455 0.014098 *
## block494             -17.43867    4.71344  -3.700 0.000216 ***
## block495              -6.26532    2.27240  -2.757 0.005831 **
## block496              -4.18452    1.10465  -3.788 0.000152 ***
## block497             -11.39401    2.99822  -3.800 0.000145 ***
## block498              -9.30180    2.29394  -4.055 5.01e-05 ***
## block499              -7.32865    2.18334  -3.357 0.000789 ***
## block500              -9.18227    2.41859  -3.797 0.000147 ***
## block501               6.41233    1.36819   4.687 2.78e-06 ***
## block502             -12.07554    3.37117  -3.582 0.000341 ***
## block503              -0.15351    0.54673  -0.281 0.778881
## block504              -3.54313    0.92026  -3.850 0.000118 ***
## block505             -15.89690    4.00042  -3.974 7.07e-05 ***
## block506               4.88487    0.89711   5.445 5.18e-08 ***
## block507             -17.03530    4.73385  -3.599 0.000320 ***
## block508             -15.58033    4.20349  -3.707 0.000210 ***
## block509              -0.89710    0.40886  -2.194 0.028223 *
## block510             -18.24568    4.77648  -3.820 0.000134 ***
## block511               0.59321    0.40842   1.452 0.146378
## block512              -7.67335    2.22559  -3.448 0.000565 ***
## block513               5.73421    1.36253   4.209 2.57e-05 ***
## block514              -0.28434    0.65108  -0.437 0.662319
## block515             -10.95998    3.12338  -3.509 0.000450 ***
## block516              -6.03646    1.76664  -3.417 0.000633 ***
## block517             -11.20651    2.83708  -3.950 7.81e-05 ***
## block518             -14.33148    3.66655  -3.909 9.28e-05 ***
## block519              -2.11451    0.78574  -2.691 0.007122 **
## block520              -1.52005    0.51708  -2.940 0.003286 **
## block521              -7.41320    2.14889  -3.450 0.000561 ***
## block522               0.10862    0.38867   0.279 0.779887
## block523              28.81283    6.65424   4.330 1.49e-05 ***
## block524              -5.84628    1.68594  -3.468 0.000525 ***
## block525             -11.08180    2.77657  -3.991 6.57e-05 ***
## block526             -10.66590    2.51310  -4.244 2.19e-05 ***
## block527               0.62215    0.37760   1.648 0.099431 .
## block528              -2.45901    1.19001  -2.066 0.038794 *
## block529              -0.14034    0.41605  -0.337 0.735872
## block530               8.02243    1.69797   4.725 2.30e-06 ***
## block531             -10.08397    2.37843  -4.240 2.24e-05 ***
## block532              -6.71186    1.73495  -3.869 0.000109 ***
## block533              -3.49946    0.69815  -5.012 5.37e-07 ***
## block534               0.12922    0.54569   0.237 0.812817
## block535             -11.30527    2.78694  -4.057 4.98e-05 ***
## block536              -4.23765    0.73937  -5.731 9.96e-09 ***
```

```
## block537                -1.01994    0.33804  -3.017 0.002551 **
## block538                -4.49347    1.24615  -3.606 0.000311 ***
## block539               -18.83663    5.42268  -3.474 0.000513 ***
## block540                -3.68206    0.78471  -4.692 2.70e-06 ***
## block541                -4.82366    1.50534  -3.204 0.001354 **
## block542                -5.30958    2.12418  -2.500 0.012434 *
## block543                -1.13446    0.72858  -1.557 0.119451
## block544               -14.95996    5.02246  -2.979 0.002896 **
## block545                -3.48979    1.38326  -2.523 0.011640 *
## block546               -12.86373    3.47983  -3.697 0.000218 ***
## block547                -3.68152    1.32250  -2.784 0.005373 **
## block548                 6.66572    1.26968   5.250 1.52e-07 ***
## block549                 1.20355    0.67761   1.776 0.075708 .
## block550                      NA         NA      NA       NA
## block551                      NA         NA      NA       NA
## block552                      NA         NA      NA       NA
## year_monthcount.201002  -0.31653    0.06408  -4.939 7.84e-07 ***
## year_monthcount.201003   0.11105    0.05672   1.958 0.050237 .
## year_monthcount.201004   0.11467    0.05644   2.032 0.042169 *
## year_monthcount.201005   0.28635    0.05443   5.261 1.43e-07 ***
## year_monthcount.201006   0.22821    0.05508   4.143 3.43e-05 ***
## year_monthcount.201007   0.24789    0.05465   4.536 5.73e-06 ***
## year_monthcount.201008   0.48851    0.05230   9.341  < 2e-16 ***
## year_monthcount.201009   0.33808    0.05383   6.281 3.37e-10 ***
## year_monthcount.201010   0.39437    0.05322   7.410 1.26e-13 ***
## year_monthcount.201011   0.36520    0.05415   6.745 1.53e-11 ***
## year_monthcount.201012   0.10407    0.05682   1.832 0.066986 .
## year_monthcount.201101  -0.01310    0.05791  -0.226 0.820986
## year_monthcount.201102  -0.33698    0.06359  -5.299 1.16e-07 ***
## year_monthcount.201103  -0.06312    0.05967  -1.058 0.290106
## year_monthcount.201104   0.14379    0.05630   2.554 0.010659 *
## year_monthcount.201105   0.24226    0.05489   4.414 1.02e-05 ***
## year_monthcount.201106   0.30812    0.05407   5.698 1.21e-08 ***
## year_monthcount.201107   0.22978    0.05487   4.187 2.82e-05 ***
## year_monthcount.201108   0.40931    0.05301   7.722 1.14e-14 ***
## year_monthcount.201109   0.34750    0.05337   6.511 7.45e-11 ***
## year_monthcount.201110   0.28592    0.05444   5.252 1.50e-07 ***
## year_monthcount.201111   0.32060    0.05416   5.919 3.24e-09 ***
## year_monthcount.201112   0.24020    0.05481   4.382 1.17e-05 ***
## year_monthcount.201201  -0.02670    0.05912  -0.452 0.651547
## year_monthcount.201202  -0.39554    0.06470  -6.113 9.76e-10 ***
## year_monthcount.201203  -0.10541    0.05884  -1.791 0.073233 .
## year_monthcount.201204  -0.12554    0.05995  -2.094 0.036252 *
## year_monthcount.201205   0.02738    0.05835   0.469 0.638906
## year_monthcount.201206   0.08223    0.05698   1.443 0.148938
## year_monthcount.201207   0.06412    0.05769   1.112 0.266328
## year_monthcount.201208   0.07132    0.05669   1.258 0.208429
## year_monthcount.201209  -0.07870    0.05931  -1.327 0.184531
## year_monthcount.201210   0.03241    0.05756   0.563 0.573376
## year_monthcount.201211   0.02685    0.05768   0.466 0.641566
## year_monthcount.201212  -0.07346    0.05942  -1.236 0.216334
## year_monthcount.201301  -0.19140    0.06098  -3.139 0.001698 **
## year_monthcount.201302  -0.75685    0.07143 -10.596  < 2e-16 ***
## year_monthcount.201303  -0.55804    0.06870  -8.123 4.55e-16 ***
```

```
## year_monthcount.201304  -0.32550    0.06354  -5.123 3.01e-07 ***
## year_monthcount.201305  -0.23331    0.06284  -3.713 0.000205 ***
## year_monthcount.201306  -0.28457    0.06303  -4.515 6.33e-06 ***
## year_monthcount.201307  -0.10732    0.06000  -1.789 0.073649 .
## year_monthcount.201308   0.01686    0.05824   0.290 0.772133
## year_monthcount.201309  -0.11958    0.05938  -2.014 0.044048 *
## year_monthcount.201310  -0.12571    0.06030  -2.085 0.037098 *
## year_monthcount.201311  -0.17097    0.06022  -2.839 0.004525 **
## year_monthcount.201312  -0.26517    0.06231  -4.256 2.08e-05 ***
## year_monthcount.201401  -0.58115    0.06882  -8.444  < 2e-16 ***
## year_monthcount.201402  -0.94629    0.07726 -12.248  < 2e-16 ***
## year_monthcount.201403  -0.68122    0.07153  -9.523  < 2e-16 ***
## year_monthcount.201404  -0.64002    0.07011  -9.129  < 2e-16 ***
## year_monthcount.201405  -0.40837    0.06508  -6.275 3.51e-10 ***
## year_monthcount.201406  -0.41578    0.06534  -6.363 1.97e-10 ***
## year_monthcount.201407  -0.29923    0.06308  -4.744 2.10e-06 ***
## year_monthcount.201408  -0.36600    0.06435  -5.687 1.29e-08 ***
## year_monthcount.201409  -0.37502    0.06500  -5.769 7.96e-09 ***
## year_monthcount.201410  -0.35821    0.06361  -5.631 1.79e-08 ***
## year_monthcount.201411  -0.59582    0.06883  -8.657  < 2e-16 ***
## year_monthcount.201412  -0.33330    0.06436  -5.178 2.24e-07 ***
## year_monthcount.201501  -0.62642    0.06946  -9.019  < 2e-16 ***
## year_monthcount.201502  -1.07933    0.08204 -13.156  < 2e-16 ***
## year_monthcount.201503  -0.73597    0.07212 -10.205  < 2e-16 ***
## year_monthcount.201504  -0.60748    0.07005  -8.673  < 2e-16 ***
## year_monthcount.201505  -0.52470    0.06748  -7.775 7.53e-15 ***
## year_monthcount.201506  -0.43752    0.06587  -6.643 3.08e-11 ***
## year_monthcount.201507  -0.44890    0.06610  -6.792 1.11e-11 ***
## year_monthcount.201508  -0.30157    0.06265  -4.813 1.48e-06 ***
## year_monthcount.201509  -0.35701    0.06349  -5.623 1.87e-08 ***
## year_monthcount.201510  -0.27554    0.06254  -4.406 1.05e-05 ***
## year_monthcount.201511  -0.40606    0.06474  -6.272 3.57e-10 ***
## year_monthcount.201512  -0.33863    0.06442  -5.257 1.47e-07 ***
## monthAUG                      NA         NA      NA       NA
## monthDEC                      NA         NA      NA       NA
## monthFEB                      NA         NA      NA       NA
## monthJAN                      NA         NA      NA       NA
## monthJUL                      NA         NA      NA       NA
## monthJUN                      NA         NA      NA       NA
## monthMAR                      NA         NA      NA       NA
## monthMAY                      NA         NA      NA       NA
## monthNOV                      NA         NA      NA       NA
## monthOCT                      NA         NA      NA       NA
## monthSEP                      NA         NA      NA       NA
## year                          NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 54400  on 31795  degrees of freedom
## Residual deviance: 40238  on 31173  degrees of freedom
## AIC: 88331
##
```

```
## Number of Fisher Scoring iterations: 6
```

```
# 2: ZIP Model

zipmod = zeroinfl(crime_count ~ wealth_d + offset(log(pop)) + un.emp + year , data = train_df)
zipmod2 = zeroinfl(crime_count ~ wealth_d + offset(log(pop)) + un.emp + year + month , data = train_df)
#zip_aic = 2*2 - 2 *zipmod$loglik
#zip_aic
```

```
summary(zipmod)
```

```
##
## Call:
## zeroinfl(formula = crime_count ~ wealth_d + offset(log(pop)) + un.emp +
##     year, data = train_df)
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -1.4420 -0.8369 -0.2580  0.4917 13.3483
##
## Count model coefficients (poisson with log link):
##              Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -6.246961   0.015561 -401.455  < 2e-16 ***
## wealth_d    -0.217610   0.006558  -33.183  < 2e-16 ***
## un.emp       0.473899   0.093731    5.056 4.28e-07 ***
## year        -0.133378   0.004060  -32.850  < 2e-16 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##              Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -8.696068   0.067763 -128.330   <2e-16 ***
## wealth_d    -0.571401   0.033013  -17.308   <2e-16 ***
## un.emp      -0.009703   0.390075   -0.025     0.98
## year         0.160139   0.015756   10.164   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 17
## Log-likelihood: -4.745e+04 on 8 Df
```

```
AIC(zipmod) > AIC(zipmod2)    # better without month (month as fixed effect unclear since month was sig
```

```
## [1] TRUE
```

```
AIC(zipmod)
```

```
## [1] 94918.25
```

```
1-pchisq(-2*zipmod$loglik, zipmod$df.residual)   # fails the goodness of fit test
```

```
## [1] 0
```

```r
-2*zipmod$loglik / zipmod$df.residual # possible overdispersion
```

```
## [1] 2.985474
```

```r
# 3:
mixmod1 = glmer(crime_count ~ wealth_d + un.emp + year + (1|block) + offset(log(pop)), control = glmerC
summary(mixmod1)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula:
## crime_count ~ wealth_d + un.emp + year + (1 | block) + offset(log(pop))
##    Data: train_df
## Control: glmerControl(optCtrl = list(maxfun = 10000))
##
##      AIC      BIC   logLik deviance df.resid
##  90607.3  90649.1 -45298.6  90597.3    31791
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.4369 -0.8435 -0.3175  0.5635  9.9385
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  block  (Intercept) 0.2572   0.5072
## Number of obs: 31796, groups:  block, 552
##
## Fixed effects:
##              Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -6.457652   0.044549 -144.956   <2e-16 ***
## wealth_d    -0.220200   0.022454   -9.807   <2e-16 ***
## un.emp       0.252686   0.307874    0.821    0.412
## year        -0.164372   0.003065  -53.628   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) wlth_d un.emp
## wealth_d -0.030
## un.emp   -0.853  0.028
## year     -0.139  0.000 -0.001
```

```r
AIC(mixmod1) < AIC(zipmod)
```

```
## [1] TRUE
```

```r
mixmod2 = glmer(crime_count ~ wealth_d + un.emp +year + (1|month) + (1|block) + offset(log(pop)), nAGQ =
summary(mixmod2)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
```

```
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: crime_count ~ wealth_d + un.emp + year + (1 | month) + (1 | block) +
##     offset(log(pop))
##    Data: train_df
##
##      AIC      BIC   logLik deviance df.resid
##  89644.6  89694.8 -44816.3  89632.6    31790
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.4340 -0.8327 -0.3278  0.5641 13.4216
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  block  (Intercept) 0.2577   0.5076
##  month  (Intercept) 0.0318   0.1783
## Number of obs: 31796, groups:  block, 552; month, 12
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.471908   0.068099 -95.037   <2e-16 ***
## wealth_d    -0.220661   0.022471  -9.820   <2e-16 ***
## un.emp       0.251711   0.307892   0.818    0.414
## year        -0.164237   0.003064 -53.607   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) wlth_d un.emp
## wealth_d -0.020
## un.emp   -0.558  0.028
## year     -0.091  0.000  0.000
```

```
anova(mixmod1, mixmod2, test= 'Chisq') # month is significant random effect
```

```
## Data: train_df
## Models:
## mixmod1: crime_count ~ wealth_d + un.emp + year + (1 | block) + offset(log(pop))
## mixmod2: crime_count ~ wealth_d + un.emp + year + (1 | month) + (1 | block) + offset(log(pop))
##         npar   AIC   BIC logLik deviance Chisq Df Pr(>Chisq)
## mixmod1    5 90607 90649 -45299    90597
## mixmod2    6 89645 89695 -44816    89633 964.7  1  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mixmod3 = glmer(crime_count ~ wealth_d + year + (1|month) + (1|block) + offset(log(pop)), nAGQ = 1, fami
anova(mixmod3, mixmod2, test= 'Chisq')  # unemployment insignficant according to GLMM
```

```
## Data: train_df
## Models:
## mixmod3: crime_count ~ wealth_d + year + (1 | month) + (1 | block) + offset(log(pop))
## mixmod2: crime_count ~ wealth_d + un.emp + year + (1 | month) + (1 | block) + offset(log(pop))
```

```
##         npar   AIC   BIC logLik deviance  Chisq Df Pr(>Chisq)
## mixmod3    5 89643 89685 -44817    89633
## mixmod2    6 89645 89695 -44816    89633 0.6738  1     0.4117
```

```
summary(mixmod3)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula:
## crime_count ~ wealth_d + year + (1 | month) + (1 | block) + offset(log(pop))
##    Data: train_df
##
##      AIC      BIC   logLik deviance df.resid
##  89643.2  89685.1 -44816.6  89633.2    31791
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.4343 -0.8327 -0.3283  0.5640 13.4266
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  block  (Intercept) 0.2582   0.5082
##  month  (Intercept) 0.0318   0.1783
## Number of obs: 31796, groups:  block, 552; month, 12
##
## Fixed effects:
##              Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -6.440848   0.056493 -114.012   <2e-16 ***
## wealth_d    -0.221183   0.022484   -9.837   <2e-16 ***
## year        -0.164236   0.003064  -53.607   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) wlth_d
## wealth_d -0.005
## year     -0.110  0.000
```

```
# goodness of fit (fails)
1-pchisq(deviance(mixmod3), df.residual(mixmod3))
```

```
## [1] 0
```

```
deviance(mixmod3)/df.residual(mixmod3)    # No over dispersion
```

```
## [1] 1.278817
```

```
# Poisson GLMM Model (AR-1 term because there is an apparent effect from month)
#weights = 1 / log(pop)
# lme assumes response is normally distributed
```

```r
#compare_glm = glm(crime_count ~ wealth_d + year + un.emp, data = train_df, family = "poisson")

#ar_mod = lme(crime_count ~ wealth_d + un.emp + year,
 #               data = train_df,
  #              random =~ 1|month + 1|block,
   #             correlation = corAR1(form =~ 1|month))  # specify AR-1 correlation

# Not sure what is going on here
```

```r
#ar_mod_test = lme(crime_count ~ wealth_d + un.emp + log(pop),
                #data = train_df,
                #random =~ 1|month,
                #correlation = corAR1(form = ~ 1 | month))
# Cannot do offset terms like before
```

```r
#summary(ar_mod)
#summary(ar_mod2)
#AIC(compare_glm)
```

```r
ar_mod2 = lme(crime_count ~ wealth_d + un.emp + log(pop),
                data = train_df,
                random =~ 1|month,
                correlation = corAR1(form = ~ 1 | month))  # specify AR-1 correlation
# COL Clark says to avoid straight log(pop) but best option we have. Will mention in the write-up.
```

```r
summary(ar_mod2)
```

```
## Linear mixed-effects model fit by REML
##   Data: train_df
##        AIC      BIC    logLik
##   112631.1 112689.7 -56308.55
##
## Random effects:
##  Formula: ~1 | month
##         (Intercept) Residual
## StdDev:   0.1965172 1.444408
##
## Correlation Structure: AR(1)
##  Formula: ~1 | month
##  Parameter estimate(s):
##       Phi
## 0.1808729
## Fixed effects:  crime_count ~ wealth_d + un.emp + log(pop)
##                  Value  Std.Error    DF   t-value p-value
## (Intercept) -0.9818849 0.18759180 31781 -5.234157       0
## wealth_d     0.1082012 0.01346533 31781  8.035541       0
## un.emp       0.4483385 0.10890851 31781  4.116653       0
## log(pop)     0.3111442 0.02604785 31781 11.945100       0
##  Correlation:
##          (Intr) wlth_d un.emp
## wealth_d  0.764
## un.emp   -0.048  0.032
```

```
## log(pop) -0.949 -0.806 -0.025
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -1.5591351 -0.7406433 -0.2220277  0.4548783 11.1271015
##
## Number of Observations: 31796
## Number of Groups: 12
```