PREDICT ALIGN PREVENT

Little Rock Child Maltreatment: Predictive Analysis

UA Research Team: Drawve, Grant, Shaun Thomas, and Jyotishka Datta

July 6, 2020 (Updated October 14, 2020)

1 Predict Phase

1.1 Motivation

This report outlines a statistical prediction/inference framework and modeling structure for spatially referenced child maltreatment events in Little Rock between 2015 and 2018. The underlying premise is to aggregate multiple spatial events to a common underlying geography by looking at the counts of events in pre-defined raster cells, and build a statistical or machine learning framework, that accounts for both social and environmental (physical - built) factors as well as the spatial clustering of both the response events and other factors and resources.

To visualize the extent of spatial clustering, Figure 1.1 displays the count of maltreatment events between July 2015 and June 2018 by census tracts. This figure demonstrates that the spatial distribution of child maltreatment is far from uniformly distributed across tracts. Thus, a critical question is allocation of limited child welfare resources to the communities that are most severely affected.

To identify locations that will need the most resources we must first determine where child maltreatment events are clustered. While individual, family, and household level factors affect child maltreatment, extant research suggests that community and social factors play an important role in understanding where maltreatment may occur (Daley et al. 2016, Durlauf, 2004).

The maltreatment event clusters are visualized in Figure 1.1b, which maps the rate of child maltreatment events in Little Rock, AR between 2015 and 2018. Recent work indicates that variation in these spatial clusters can be predicted by environmental factors such as crime, blight, and bars and restaurants.

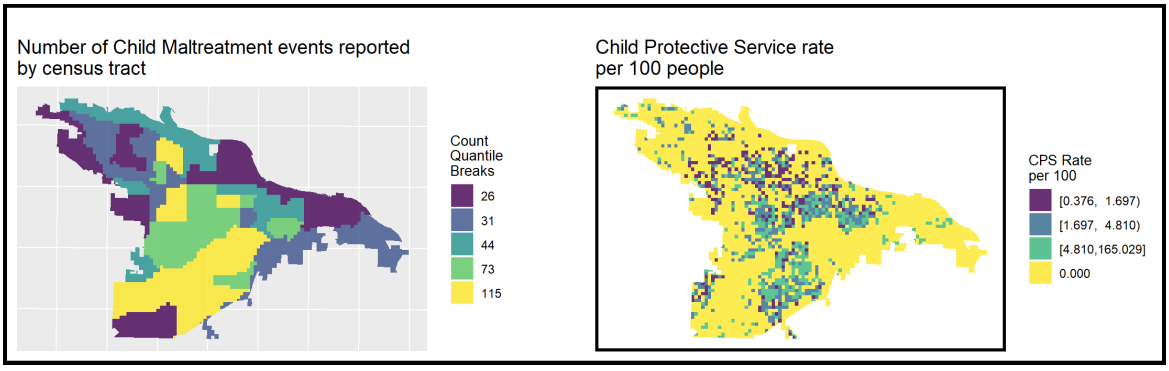


Figure 1.1 Distribution of child maltreatment events in Little Rock, AR.

1.2 Exploratory Analysis

In this section, we derive several descriptive insights from our data. To begin, maltreatment events across space and time are visualized and we discuss possible selection bias in the data. Next, hypotheses related to the clustering of maltreatment events across space are presented. Finally, we calculate a series of pairwise correlations between maltreatment and our selected features (I.e. features we expect to influence where child maltreatment occurs).

1.2.1 Analysis of maltreatment events over time and space

Figure 1.2 below plots the counts of child maltreatment events throughout the study period, which began midway into 2015 and ended midway through 2018. As we do not observe these reporting changes, time variation cannot be used for model validation.

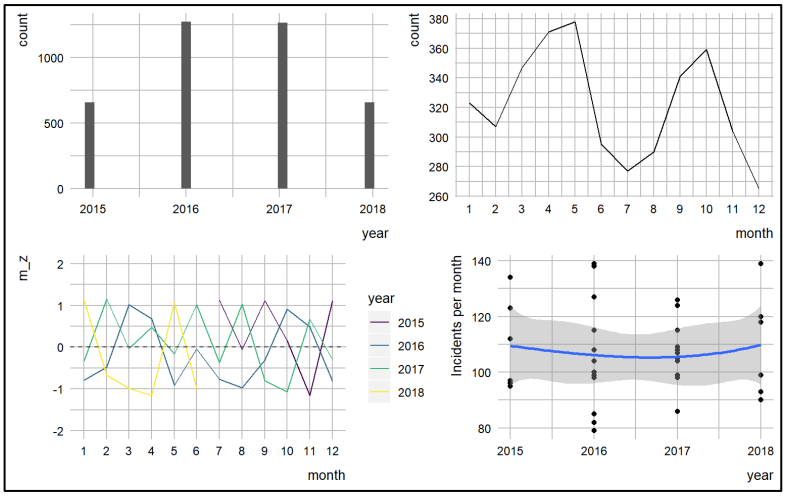


Figure 1.2a Temporal variation by year, month and weekdays.

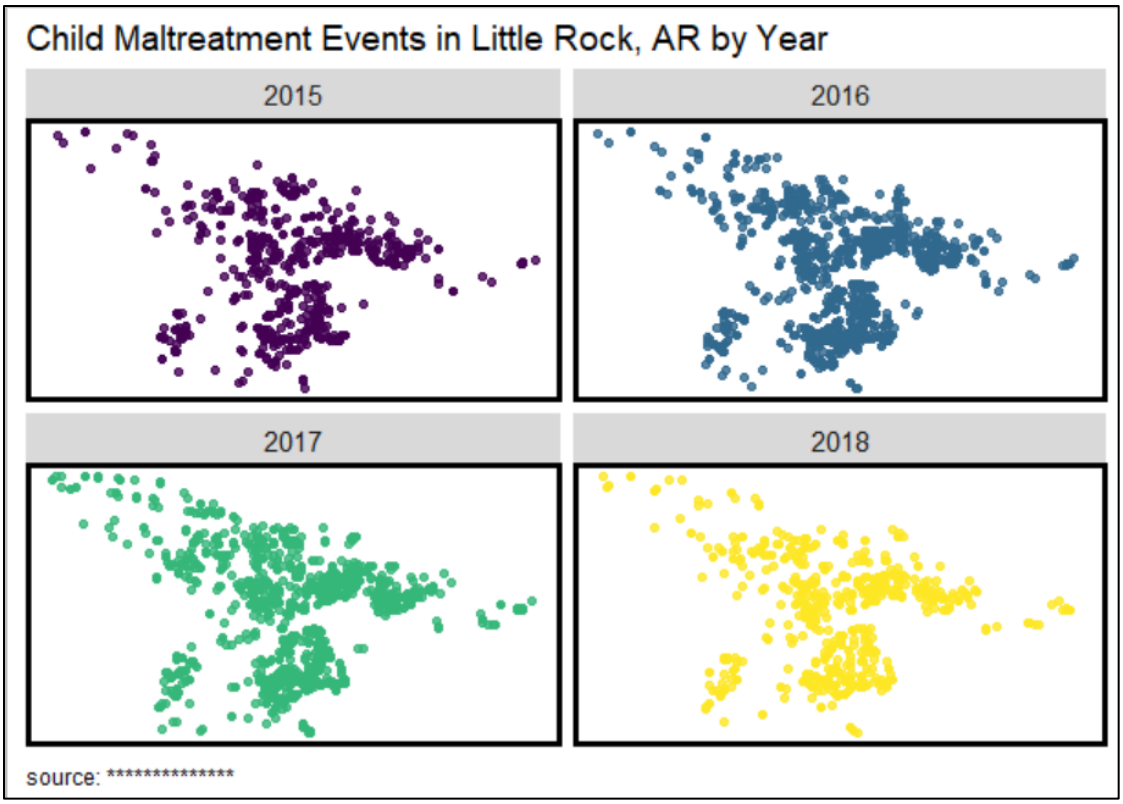


Figure 1.2b Yearly variation of child maltreatment events in Little Rock.

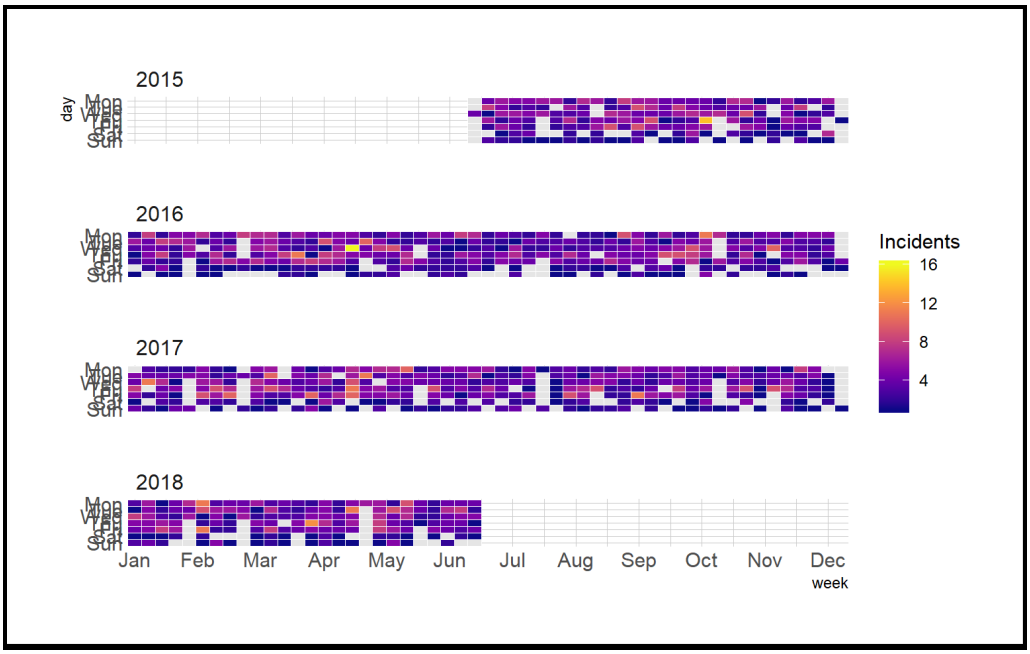


Figure 1.2c Weekly distribution of child maltreatment incidents.

1.2.2 Testing maltreatment events for clusters

One overarching assumption underlying our model is that maltreatment events are clustered in space. To test this hypothesis, the Local Moran's I statistic is employed.

The Local Moran's I statistic examines whether the local distribution in the rate of maltreatment events, defined by a spatial weights matrix, is more clustered than we would expect due to random chance alone (Bivand *et al.* 2008, Pebesma & Bivand, 2019). The next figure plots the Local Moran's I results, with the identification of the k nearest neighbors then assigning their respective weights were done using the package **spdep** (Bivand & Wong, 2018). The first panel displays the count of maltreatment events; Panel 2 shows the Local Moran's I value; and Panel 3 identifies areas that exhibit statistically significant clustering. Panel 3 suggest areas that resemble discrete clusters of maltreatment in space.

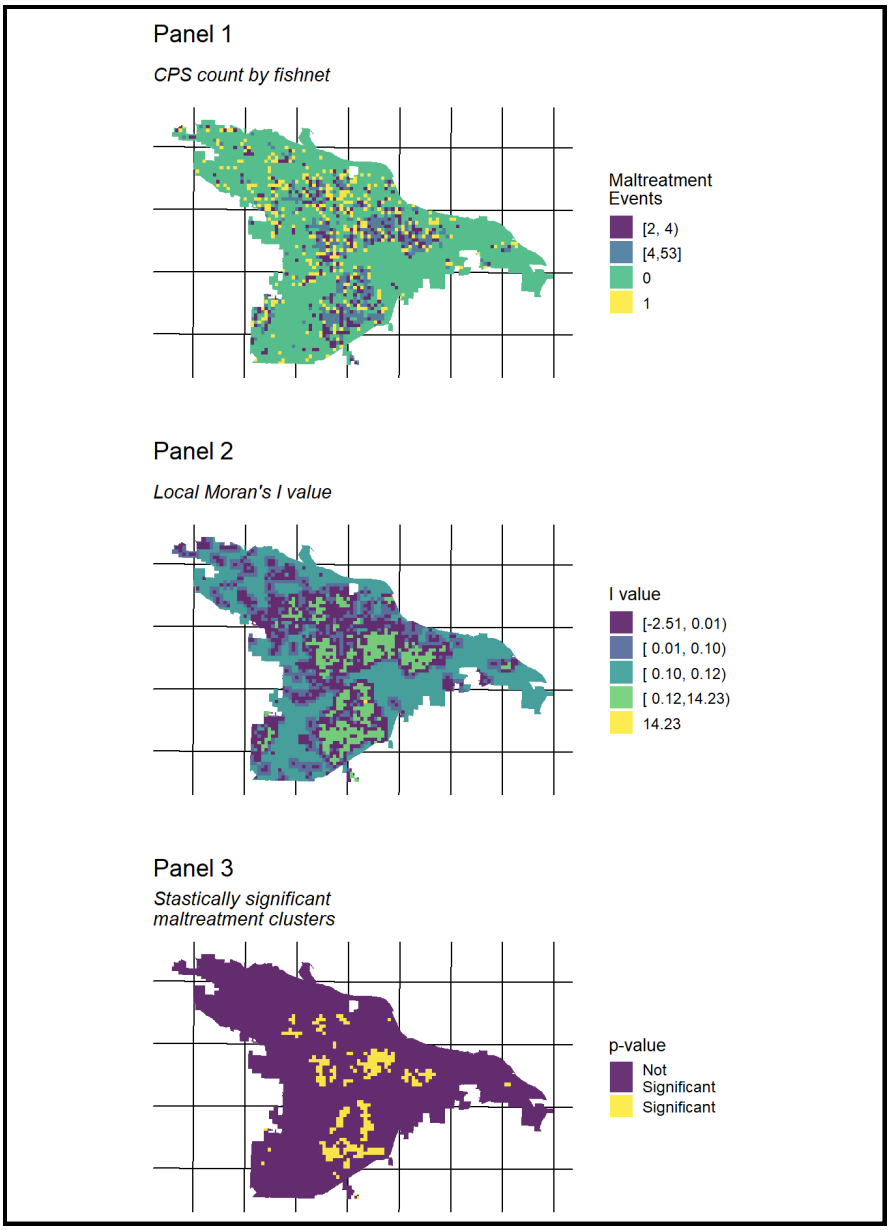


Figure 1.3 Spatial Autocorrelation via Moran's I

1.3 Feature Engineering

1.3.1 Pairwise correlations

In this section we seek to understand the extent of the linear relationship between the available features. Towards this end, we first create several features from the available data (i.e. built environment factors and census data). The two figures below visualize pairwise correlations for the most correlative risk and protective factors, respectively. Note the correlation coefficients associated with the maltreatment count (`cps_net`) and the maltreatment rate (`cps_rate`). The colors of the plot vary with the strength of the correlations, either positive or negative.

There are 3 different prefixes associated which each type of feature. `NN` refers to features calculated by taking the average distance between a fishnet grid cell and its k nearest risk/protective factor neighbor. `ed` refers to the Euclidean distance between a fishnet grid cell and its 1 nearest risk/protective factor neighbor. `agg` refers to the count of risk/protective factor events in each fishnet grid cell.

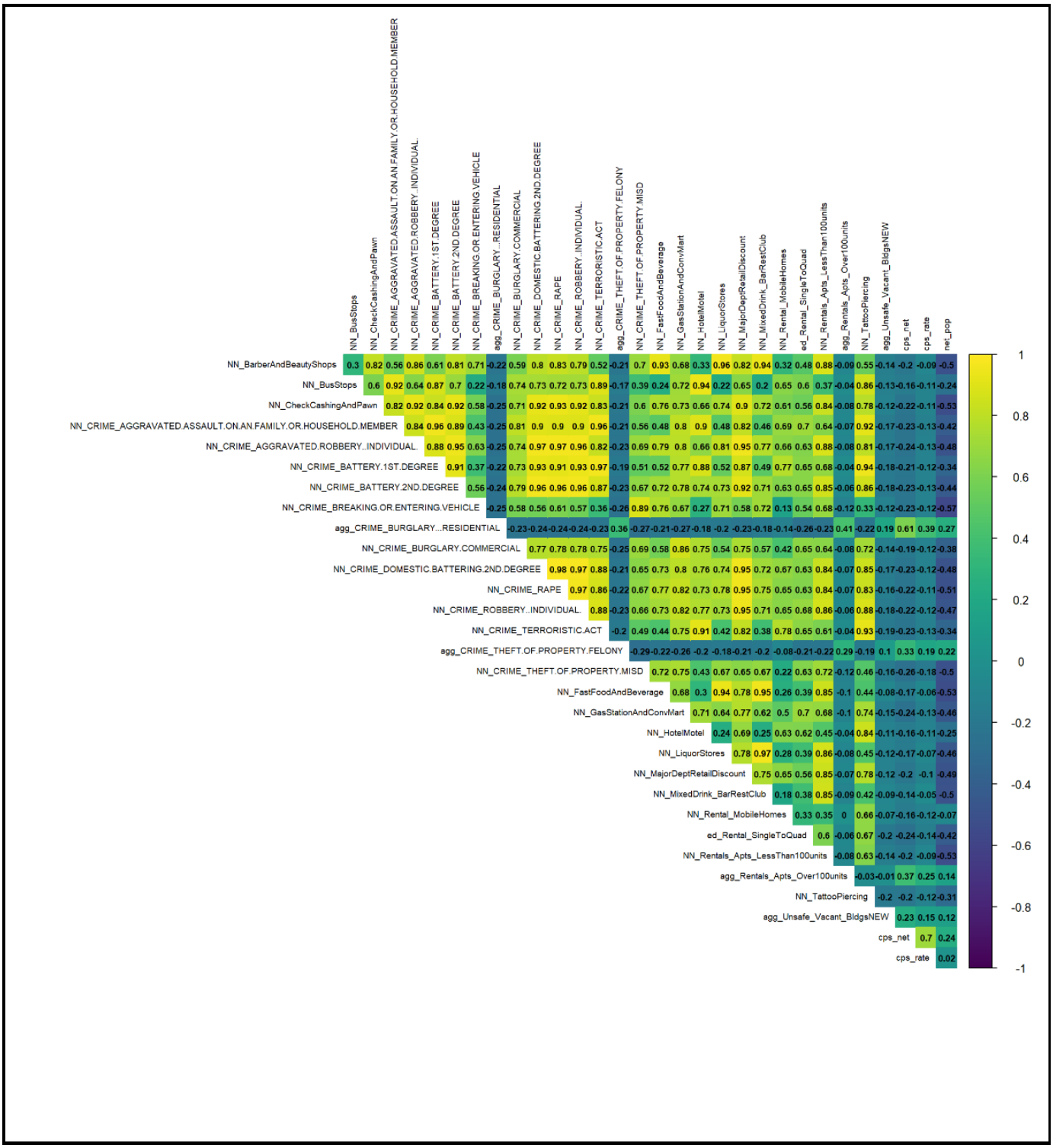


Figure 1.4 Correlation between risk factors

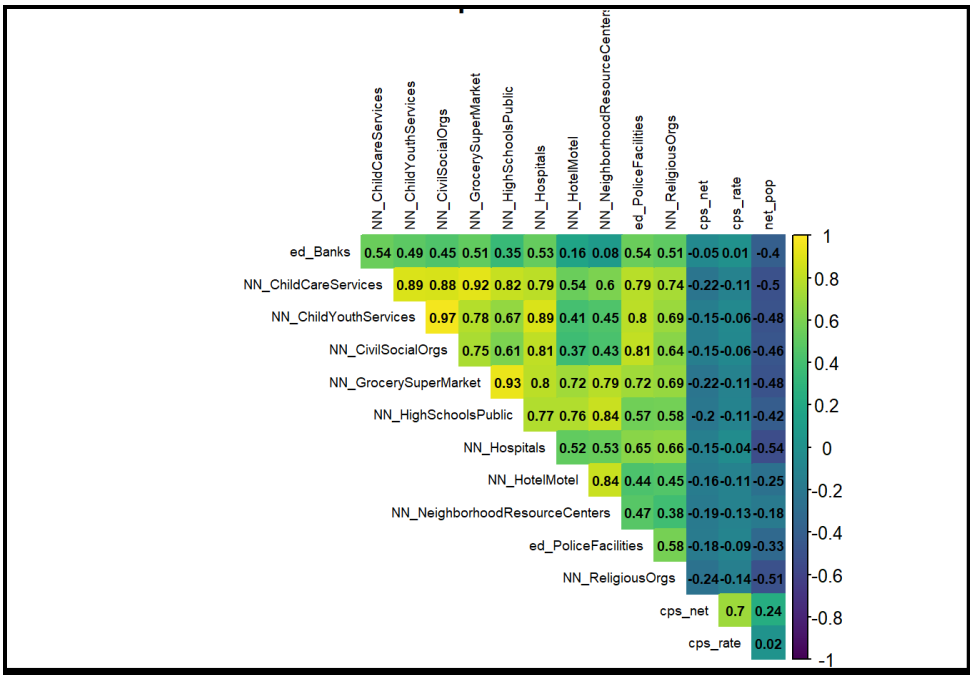


Figure 1.5 Correlation between protective factors

1.4 Model Fitting

1.4.1 Model fitting & stacking

Model 'fitting' describes the process by which a statistical algorithm learns about maltreatment risk by relating the interaction of risk and protective factors to maltreatment events across space. Once a model is fitted and validated, the learned pattern is applied back to the contributing features in each fishnet cell in order to predict the count of maltreatment events across space. This prediction highlights areas where maltreatment is present but unreported. This is similar to the risk terrain analysis methods (Caplan et al. (2011, 2015), Drawve, 2016) where a separate map layers, created for each predictor, are combined to produce a composite map where each factor can be evaluated in a model-based way. The first step in the model building process is to select the top few most statistically important risk and protective feature sets. We select across the different feature types (Euclidean distance, average nearest neighbor distance, and aggregate counts) based upon statistical correlation. These features comprise the final feature sets, which are then subjected to our models.

Three different algorithms are fit modeling different aspects of the spatial process and then combined into a fourth 'meta-model'. The three individual models are a Poisson Generalized Linear Model (Poisson GLM), a Random Forest model, and a Spatial Durbin Model (SDM). The final prediction of maltreatment events is produced from the meta-model which is created by applying the Random Forest algorithm to the predictions made by the sub-models. The use of three distinct model algorithms is an effort to understand different aspects of the highly complex system that contributes to the observation of a maltreatment event.

At each stage in this process, models are fit using a 'k-fold cross validation' routine (kFCV). kFCV splits the data into spatially explicit groups, in this case tracts, fits the models to all but (1/k)-th of the groups,

and predicts maltreatment event counts for the left-out folds of groups. This process, explained in greater detail below, tests how well the models generalize across neighborhoods. Below we explain the three sub-models and their inclusion in the final meta-model as well.

Poisson GLM

The Poisson GLM model, fit with the base R `glm` function, is an adaptation of linear regression that accounts for the characteristics of count data. The adaptations include modeling the residuals as Poisson distributed and transforming the linear model with a natural log function (Montgomery et al., 2006). As a result, the predictions from a Poisson model are positive and represent mean expected counts conditional on the contributing risk or protective features for each fishnet grid cell. In the meta-model, Poisson GLM predictions represent a linear model of a Poisson distributed count process.

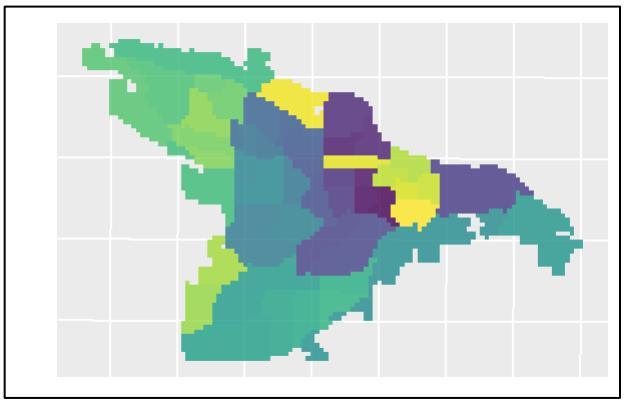


Figure 1.6 Spatial cross-validation scheme – the entire geography is divided into k different folds, where the model is trained on (k-1) folds and tested on the remaining k-th fold. In this figure, colors represent folds, i.e. like-colored tracts belong to the same cross-validatory fold.

Remark: The Poisson GLM model is not an optimal choice as we have a moderately high-dimensional feature space. As such, it runs into problems of rank-deficiency and multicollinearity. Put another way, rank deficiency happens when the design matrix has less than full rank, leading to lack of unique solutions for the GLM, and multicollinearity implies columns of **X** matrix exhibiting high-degree of correlation, leading to inflated variance of the parameter estimates and instability. One fix is using a penalized regression that puts a budget on the parameter vector in the GLM framework, such as LASSO or Elastic Net, which reduces these issues to some extent.

Random Forest

In this study, the Random Forest algorithm is fit using the `ranger` library. The Random Forest algorithm builds a series of decision tree models to learn the relationship between maltreatment and exposure variables (Breiman, 2001). The stochastic approach to sampling from observed data ensures that each individual tree is different from the next. The Random Forest provides a 'wisdom of many' approach, contributing non-linear interactions between maltreatment and the corresponding features to the final meta-model.

Spatial Durbin Model

To model spatial interrelationships - also referred to as 'spatial autocorrelation' - a Spatial Durbin Model (SDM) is fit using the `errorsarlm` function of the `spdep` package in R. In the setting of this study, the interpretation of this model is that the rate of maltreatment events is affected by both the exogenous exposure factors as well as neighboring rates of maltreatment. Further, this model assumes that there may be latent features that impact the model errors but are not accounted by the exposure features (Elhorst, 2014). The key model input of spatial autocorrelation is a spatial weights matrix relating maltreatment in a given grid cell to its neighbors. Modeling the underlying spatial maltreatment process provides a powerful predictive story when input into the final meta-model. It is important to note that

the SDM is not fit with the 10FCV method due to the complications of subsetting a spatial weights matrix in a cross-validation setting.

Meta-Model

The final maltreatment count predictions are generated from a meta-model which combines predictions from the three sub-models. The process to combine the three models is straightforward as the predicted counts from each sub-model are used as input features of a new model fit with the Random Forest algorithm. Often referred to as model 'stacking', this technique seeks to average out the variance in the three separate models (Wolpert, 1992, Breiman, 1996). To reduce the risk of over-fitting, the stacked meta-model is fit and predicted using the same 10-fold cross-validation routine as the sub-models.

1.4.2 Model validation

Assessing the accuracy and spatial generalizability of model predictions is crucial when considering how to embed this model in the provision of child welfare services. A variety of approaches are used for model validation including k-fold cross validation (kFCV) and assorted goodness of fit metrics. Some of these metrics are statistical in nature, while others measure goodness of fit across space.

Goodness of fit metrics

Model error is defined simply as the difference between the observed count of maltreatment events and the predicted count for each grid cell. Complicating matters is that 297 models yield more than 567,000 grid cell level predictions. We derive several statistics to summarize and aggregate these errors in order to judge models and compare across them. We describe each below:

The Mean Absolute Error or MAE measures the average absolute difference between the observed and predicted values. An example interpretation of MAE is that, 'on average, the model is off by plus or minus 1.67 events.' MAE is simple to interpret in a policy context, however, it comes with some drawbacks, namely that the direction of the error is unknown and that every error is assumed to have the same severity. The MAE assumes that an error between a predicted count of 5 and an observed count of 7 events should be considered the same as a prediction of 23 and an observed value of 25 events. This metric is used here due to its obvious interpretation and common usage in the assessment of predictive models.

The second goodness of fit metric used in this study is called the **Logarithmic Score**. This metric is not as straightforward as the MAE, but it has qualities that make it well-suited to count-based predictions. The intuition of the Logarithmic Score is as follows: What is the likelihood of the observed count given the predicted count. More descriptively stated: if the model predicts 10 events and the observed count is 7 events, then what is the probability of observing those 7 events if the prediction of 10 is indeed the correct number. In this way, the Logarithmic Score measures the deviance between the predicted and observed counts. Specifically, this is measured by calculating the probability density of the observed value from a Poisson distribution centered on the predicted value. The goodness of fit measures below reports the negative log of the probability density so that the value should always be minimized. In the results portion of this report, the Logarithmic Score is converted back to a probability and aggregated. The result is a score that is interpreted as the 'average likelihood that the observed counts are true given the predicted maltreatment counts. Closer to one means a higher relative likelihood, 0.5 equates to maximum uncertainty, and a value near zero signifies very little likelihood.

1.4.3 Accuracy and generalization tradeoff

The purpose of the 10FCV and the goodness of fit metrics is to assess model errors on average and across space. A model that perfectly predicts the observed event counts for each fishnet grid cell would be very accurate but would not generalize well to other cells because exposure changes across the city. Conversely, a model that predicts the same count of maltreatment events for every cell would generalize well, but not be relevant to the conditions of any one cell. 10FCV and associated metrics help establish a balance between model accuracy and model generalization. Given the purpose of this study, it is important to create a model that is accurate enough to give confidence, but general enough to be applicable in areas where few cases are documented.

1.5 Results

1.5.1 Average goodness of fit results

Goodness of fit metrics the deviation between the distribution of observed count of maltreatment events and the distribution of the predicted count for each observational unit or grid cell. Since no single measurement is best for comparing forecasting accuracy [16, 19], we present several metrics to summarize and aggregate these errors in order to judge how well the data fit the models used and compare across the candidate models. We describe each of these metrics below:

Our first goodness of fit metric, the Mean Absolute Error (MAE), is defined as the absolute difference between observed and predicted maltreatment counts averaged over all grid cells, implying that a method with higher MAE has poorer predictive accuracy on average. The metric MAE_sd calculates the standard deviation of the Mean Absolute Error across each neighborhood holdout. A relatively high value of the MAE_sd suggests that the model accuracy is highly variable across different communities. A low value suggests errors are relatively consistent across neighborhoods.

For our study, the MAE values range from a minimum of 0.80 for the Spatial Durbin Model to a maximum of 1.17 for the GLM – Poisson model. The overall MAE for the Meta-Model is 0.475. Based on prior research, these MAE values are within acceptable ranges [17]. Furthermore, the relatively high standard deviation of MAE suggests that greater errors can be found in particular communities, primarily those with very high maltreatment counts.

Table 1. Comaprison between candidate method in terms of accuracy and model fit diagnostics.

Model Name	R2_mean	R2_sd	MAE_mean	MAE_sd	RMSE_mean	RMSE_sd	logdev_mean	logdev_sd
GLM - Poisson	0.331	0.114	1.166	0.416	6.452	8.966	0.583	0.066
Meta-Model	0.438	0.131	0.930	0.229	2.251	0.570	0.620	0.071

Random Forest	0.391	0.107	1.072	0.221	2.362	0.576	0.540	0.065
Spatial Durbin - sqrt	0.434	NaN	**0.803**	NaN	2.348	NaN	0.693	NaN

The second goodness of fit metric, the logarithmic score, captures the deviance between the predicted and observed counts measured as twice the log-likelihood ratio of a saturated model compared to a reduced, lower dimensional model [18]. For this metric, the likelihood is based on a Poisson distribution centered on the predicted value. The difference in the deviance metrics between two models can be loosely interpreted as the relative change in the average likelihood that the observed counts are true given the predicted maltreatment counts according to the model. Values closer to 1 is indicative of a higher relative likelihood, 0.5 equates to maximum uncertainty, and a value near zero signifies very little likelihood. The goodness of fit measures below report the negative log of the probability density so that the value should always be minimized. In the results portion of this report, the logarithmic score is converted back to a probability and then aggregated.

For the current study, the logarithmic score (reported as logdev) ranges from a low of 0.58 (GLM) to a high of 0.69 (Spatial Durbin). The mean logdev for the meta model is 0.62 with a standard deviation of 0.071. These values roughly approximate to a 95% confidence interval between 0.558 and 0.682 for the population average deviance. Substantively, these results suggest that, on average, the probability that the estimated model parameters are correct given the observed maltreatment counts is between 0.558 and 0.682. While the population average of errors from independent LOOCV estimates is helpful for assessing how the model generalized, it is equally important to know how these errors are distributed both statistically and across space.

The Root Mean Squared Error (RMSE) represents our third goodness of fit metric. In the current study, the RMSU ranges from a low of 2.25 (meta-model) to a high of 6.45 (GLM). Similar to the MAE, the RMSE is reported on the scale of the dependent variable, however, RMSE differs from MAE in that the metric is weighted heavily by larger errors. The coefficient of determination (i.e., R-squared) represents the proportion of the variance for a dependent variable that can be attributed to or explained by variation in an independent variable(s) in a regression model [18]. For example, an R-square value of 0.40 suggests that approximately 40 percent of the observed variation in the dependent variable is explained by variation in the independent variables included in the prediction model. Although r-square has become a standard, albeit limited, goodness of fit measure in linear based analyses. However, it is less applicable or appropriate to interpret in nonlinear modeling and thus not generally reported. As such, R-square is not appropriate for assessing goodness-of-fit for this current study. We include this as a point of reference without a detailed discussion.

The predictions were validated by testing the meta-model on July 2018 through June 2019 confirmed child maltreatment incidents. This represents the year following our model data and allows us to test the accuracy of our model. Citywide, there was a total of 3,671 cells and for comparison to administration units, there are, 51 census tracts and 155 block groups (depending on clipping) in Little Rock. This highlights the importance of using smaller units of analysis to identify greater variation in risk

at a smaller, more micro-unit of analysis. For instance, the highest risk category (5) had 368 cells, comprising only 10% of the area of Little Rock. At the same time, risk category 5 spatially predicted 59% of maltreatment incidents occurring the following year (i.e., hit rate; number of outcomes within category / total number of outcomes). As the risk category decreases, the overall area increases and the hit rate decreases.

To supplement these general findings, we used the predictive accuracy index commonly used within crime prediction studies (e.g., see Chainey, Tompson, & Uhlig, 2008; Drawve, 2016; Drawve & Wooditch, 2019) and even by the National Institute of Justice in their crime forecasting challenge. In short, the PAI is a metric that accounts for both the hit rate (numerator) and the predicted area percentage (denominator), allowing for meaningful comparisons. The larger the PAI, the more accurate that approach or selection is at forecasting spatial events. Starting with our highest risk category, 5, we found the PAI to be 5.89 and a decreasing PAI as the risk category decreased. Risk category 5 was about 4 times more accurate than risk category 4 (PAI = 1.42). Similar findings hold when updating the PAI calculation based on recommendations from Drawve and Wooditch (2019), risk category 5 remains about 4 times as accurate as risk category 4 This is important in understanding that a small area of Little Rock is able to accuractly forecast a majority of the following years maltreatment incidents.

1.5.2 Predicted Values and MAE maps

Figure 1.10 shows the predicted values for child maltreatment events as well as the mean absolute errors (MAE) overlaid on the map of Little Rock. Although the maps look fairly similar to each other at first glance, a close inspection reveals that there are differences in how tightly clustered or 'patchy' the prediction maps are as well as the relative quantification of uncertainty.

For example, the predicted counts for the highest bin in Poisson is very different from both random forest and spatial durbin model, which could be due to Poisson models inability to account for overdispersion in presence of clusters (also see Table 1).

As explained before, the meta-model is created by stacking the other candidates in a supervised set-up, the predicted counts as well as the error metrics for the meta-model fall somewhere in the range of the same for all of the other candidate methods. A key use of these maps is not just to provide a predictive hotspot-type visualization but also give a sense of the uncertainty attached and, by comparison with other environmental and social factors, allocate the available resources in a data-driven way. We address the issue of association of predicted maltreatment counts with ACS variables in the next section.

1.5.3 Generalizability

Further complementing these findings, Figure 1.11 shows the goodness of fit metrics broadened to the tract level for the meta-model estimates. The goodness of fit indicator was calculated by way of LOGOCV. The MAE and Logarithmic Score metrics follow a similar pattern with higher errors in tracts with higher rates of maltreatment events.

If the model were perfectly generalizable, model errors by tract would be randomly distributed. The map in Figure 1.11 demonstrates that MAE clusters slightly, see the yellow patches in Figure 1.11b.

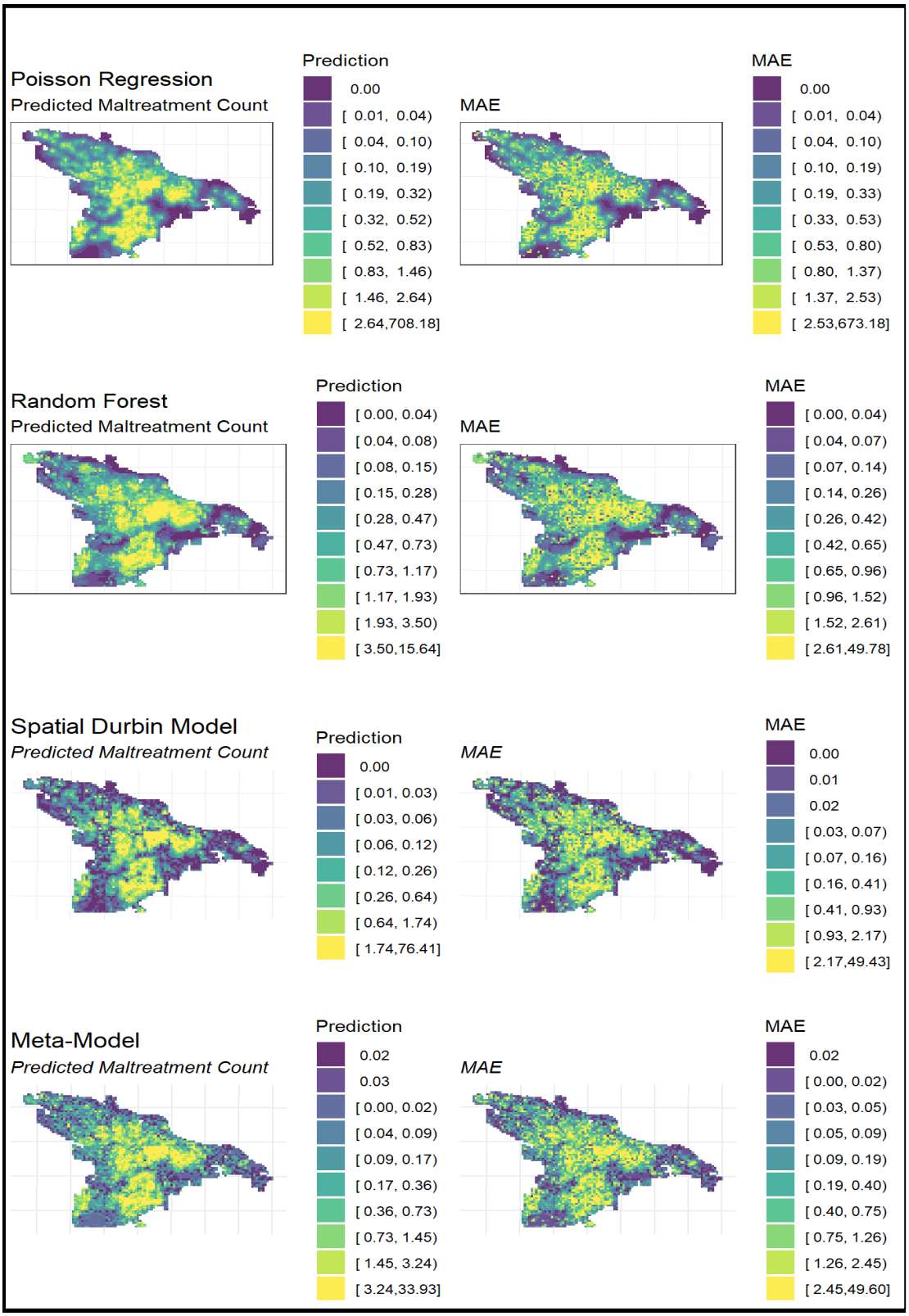


Figure 1.7 Map of Little Rock with Predicted maltreatment counts as well as MAE for dour different methods compared.

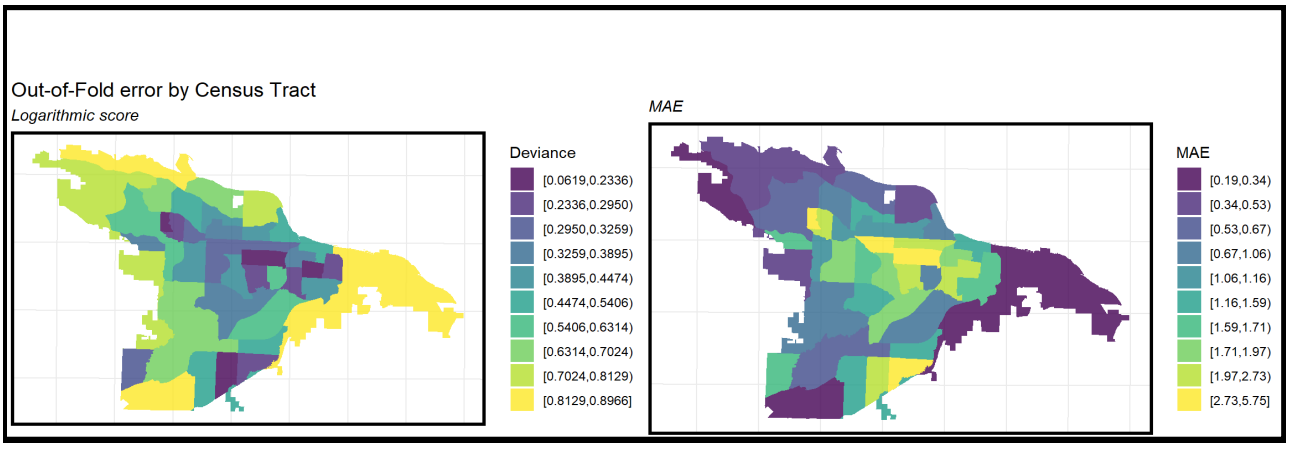


Figure 1.8 Out-of-fold error by tracts: logarithmic deviance and MAE.

1.5.4 Random Forest: Feature importance

Next, we attempt to look "under the hood" of the model. We do this first by visualizing feature importance. The below plot visualizes 'feature importance' for the Random Forest sub-model, showing which features make the greatest contribution in predicting maltreatment. We caution the reader that the feature importance for a random forest is not a formal probabilistic quantity, like the p-value or the posterior probability of a predictor being important.

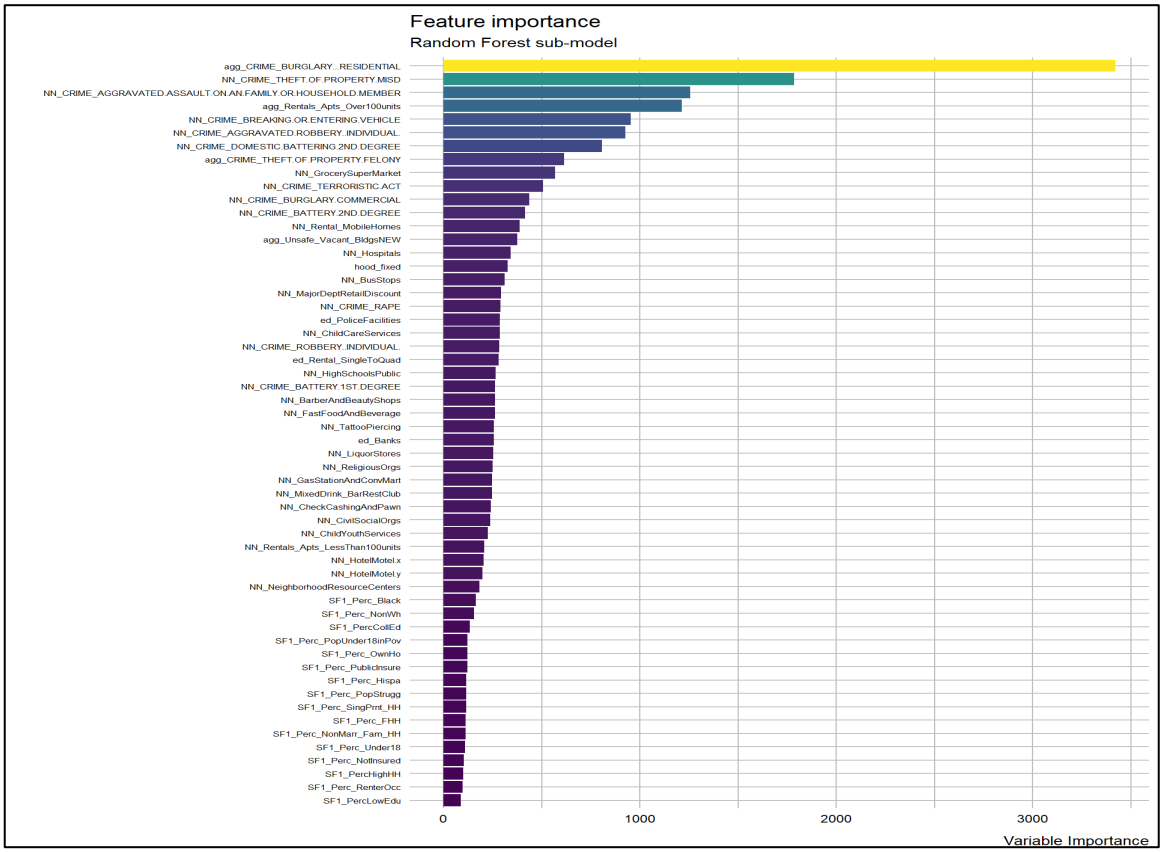


Figure 1.9 Feature importance for the random forest model.

1.5.5 Poisson GLM: Coefficients

For a Poisson GLM, the exponents of coefficients are equal to the incidence rate ratio (relative risk).

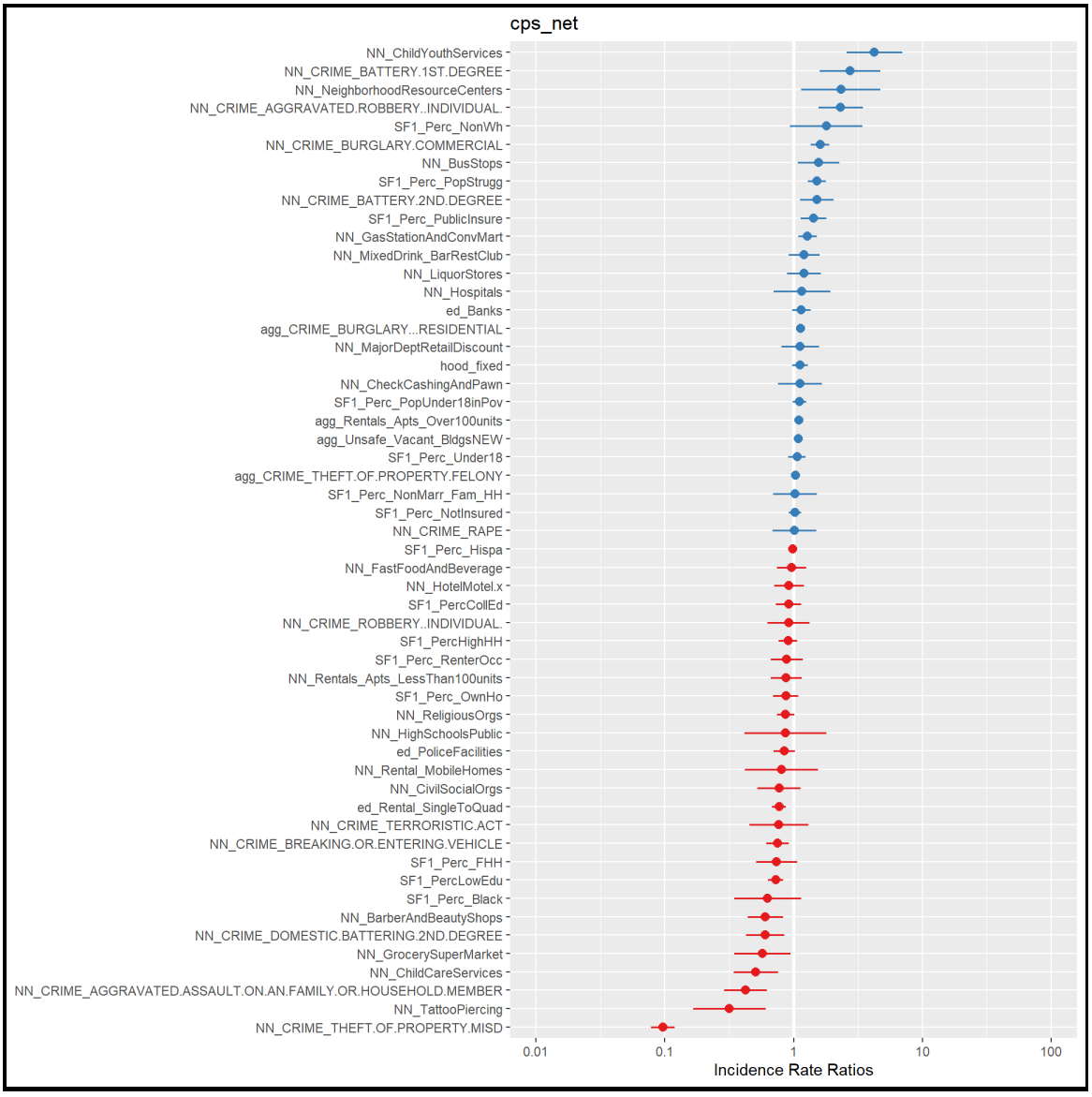


Figure 1.10 Poisson GLM coefficients

1.6 ACS Variables: Stat Area Category Plot

We look at poverty (population struggling) and non-whites rates by census tracts, classified by percentiles, and plot the spatial distribution (here '0' is less and '1' is more).

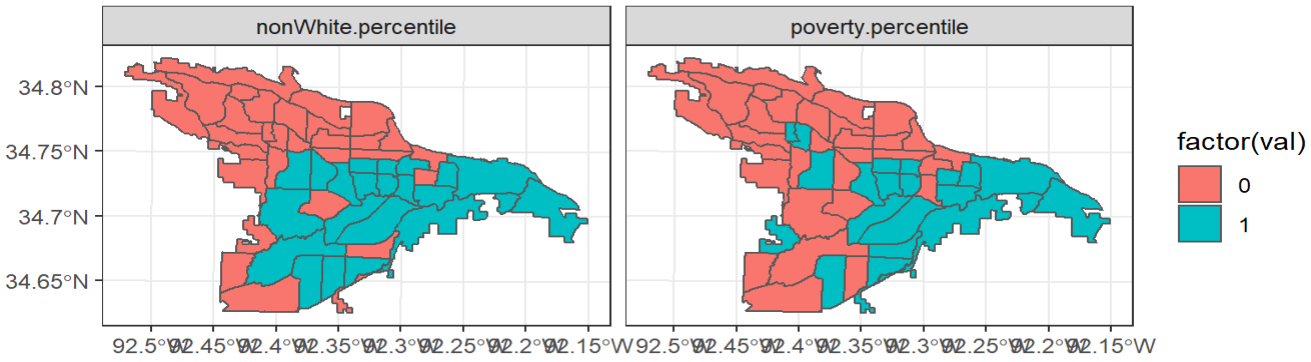


Figure 1.11 Poverty and non-white percentiles by tracts.

1.6.1 Census-tract typology comparison

1.6.1.1 Model Errors by Poverty and Non-white percentage

Next, we aggregate mean errors to statistical areas (med_dev, and med_MAE), aggregate sum of Child maltreatment incidents (med_CPS) to statarea, and (a) group by poverty (pop-struggle) and get median of stat-area aggregate errors, and (b) group by nonwhite and get median of statarea aggregate errors.

Income and race are inextricably linked to many of the census and exposure features used in the model, but no variables directly measuring race or income are included in the models. While feature importance provides some glimpses into how the model predicts, the best way to understand the inner workings of a model is to look for patterns in how it predicts. Our approach for doing so, tests how well the model generalizes across both wealthy and poor areal units as well as predominately white and predominately minority areal units.

Two census attributes are selected for these purposes including percent living below poverty and percent non-white.

nonWhite.percentile	Median log-dev	Median MAE	Median child maltreatment counts
0	0.507	1.091	1,465
1	0.398	1.710	2,361

Next, median meta-model predictions are calculated for each NSA and goodness of fit is compared between high and low areas. Table below lists the median Logarithmic Score for both the high and low classes for each of the census variables. If the model generalizes well to both tract typologies, the

Logarithmic Score should be comparable across high and low categories. We find a small but non-negligible difference between the log-scores between race-related categories across the city but slightly less so for poverty-categories.

Poverty Percentile	Median log-dev	Median MAE	Median child maltreatment counts
0	0.540	1.05	1,800
1	0.415	1.71	2,026

1.7 Comparing meta-model predictions to Kernel Density

Perhaps the strongest method for assessing the usefulness of a predictive model is to compare its predictive power to that of the current resource allocation strategy. Here we compare our model to another common spatial targeting algorithm - Kernel Density Estimation (KDE).

KDE is a simple spatial interpolation technique that calculates 'predicted risk' by taking a weighted local density of maltreatment events. No risk/protective factors are used, and no measures of statistical significance can be calculated. To compare between the meta-model predictions and KDE, predictions from both are divided into five risk categories for the purposes of comparison. We then overlay held out maltreatment events that were not used to fit the original model and calculate the percent of observed maltreatment events that fall into each predicted risk category.

Figure 1.15 maps the comparison. The KDE clearly picks up the main areas of recorded events, but also interpolates high predictions for maltreatment in the areas between and beyond. The meta-model is far more targeted.

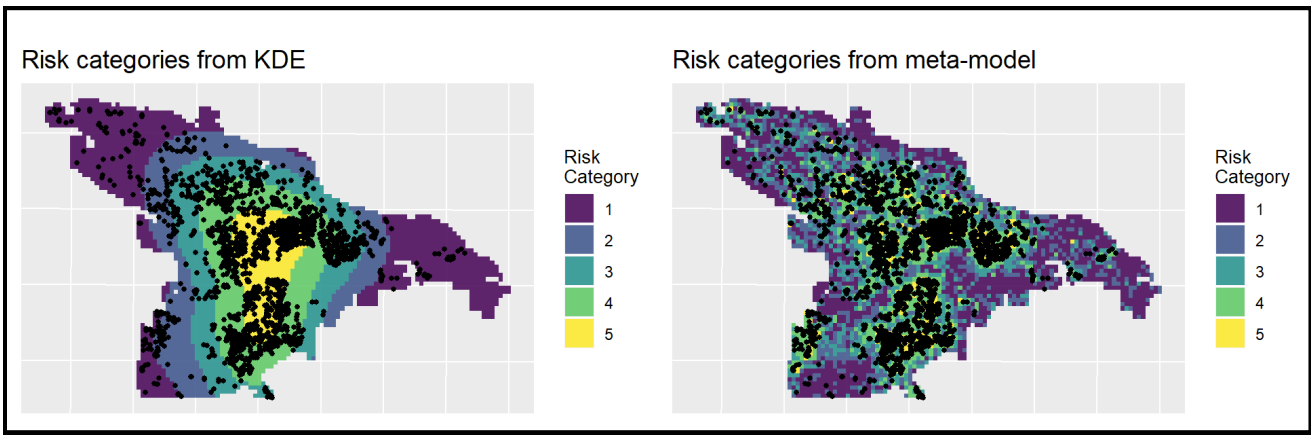


Figure 12 Risk categories from Kernel Density Estimates and the meta model.

Figure 1.16 formalizes the comparison in chart form. *The highest risk category for the meta-model captures approximately 60% of the recorded maltreatment events, whereas the KDE captures only about 35%.* This suggests that the spatial risk model vastly outperforms KDE.

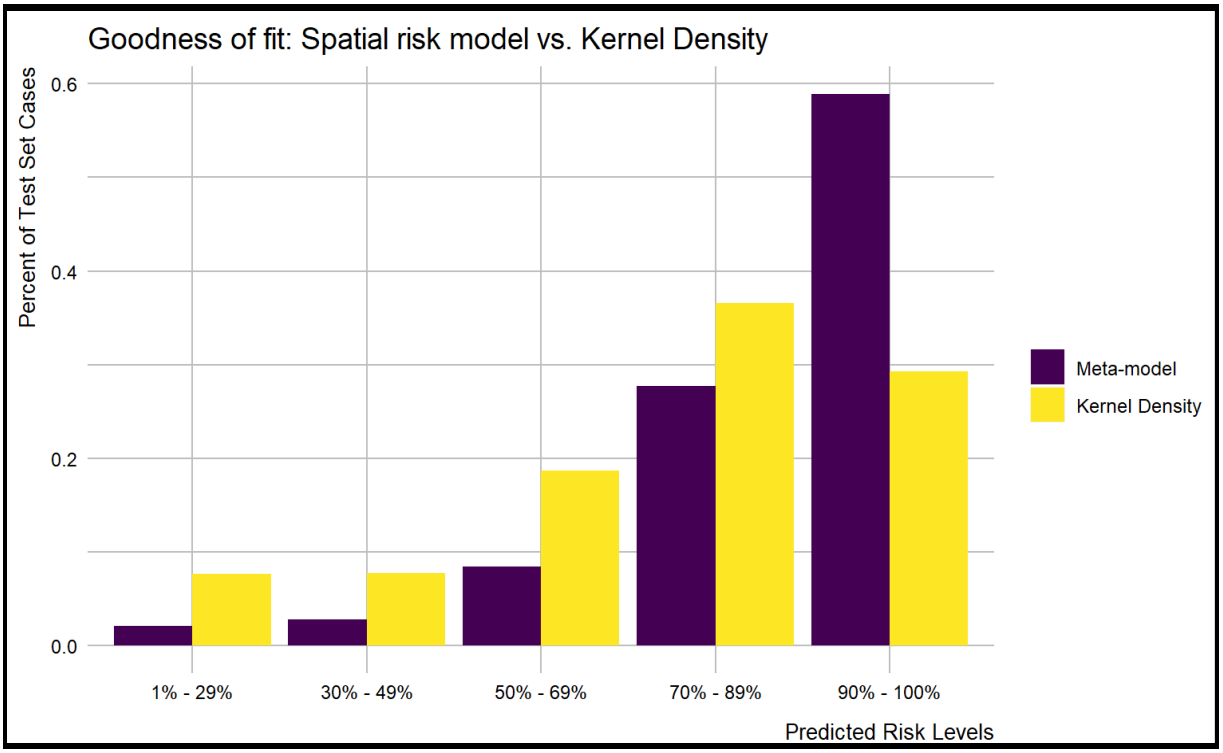


Figure 1.13 Spatial risk model versus Kernel Density Estimate: goodness-of-fit comparison.

2 Align

2.1 Risk category population totals

The demand for child welfare services is related to the number of people living in high risk areas. Figure 2.1 shows that approximately 33,210 people live in the highest risk category, covering 15.8% of the total population in Little Rock. Another 28.1% of the population (59,221) reside in the second highest risk category, indicating, in total, over 44% of the population live in areas of potentially high demand for child welfare services.

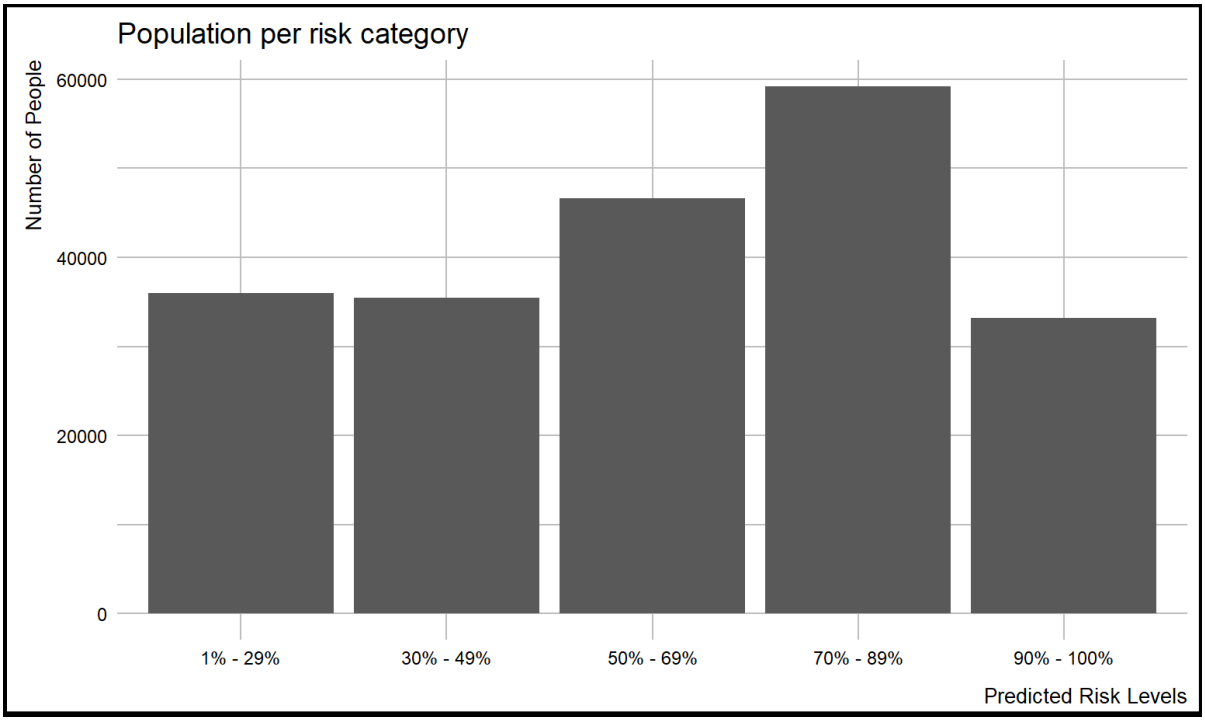


Figure 2.1 Population per risk category.

2.2 Is poverty related to predicted maltreatment events?

Figure 2.2 maps the weighted poverty rate by fishnet grid cell. How does the distribution of poverty relate to maltreatment events?

Figure 2.3 illustrates the relationship between poverty rate and predicted maltreatment count. The scatter plot demonstrates that the correlation between poverty and predictive risk is marginal. This visual relationship is confirmed by a correlation coefficient of 0.15. The weak relationship persists even when the zero count grid cells are removed.

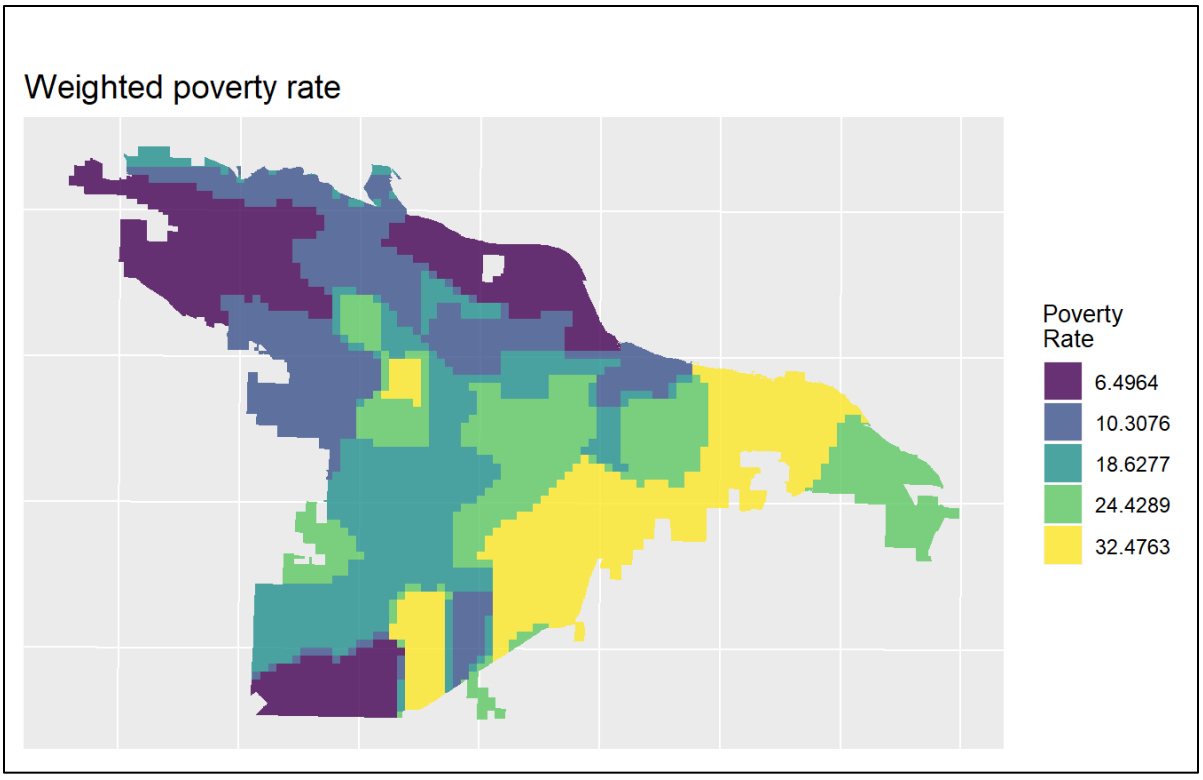


Figure 2.2 Poverty rate by census tract

2.3

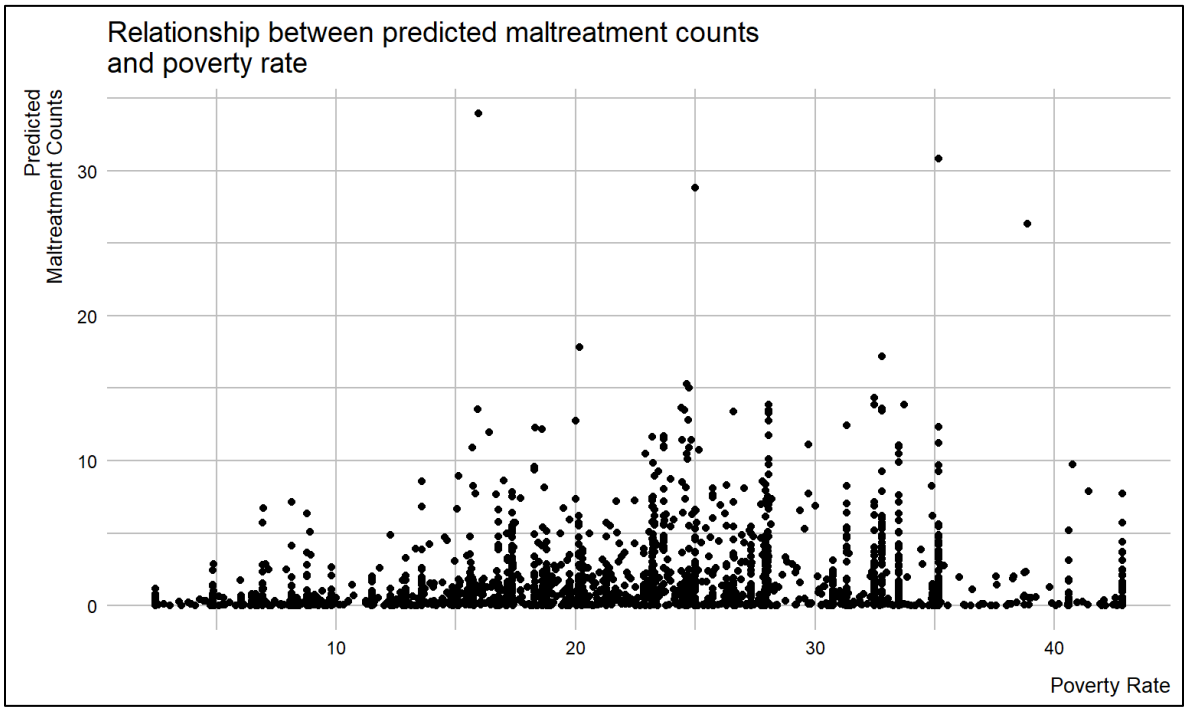


Figure 2.3 Predicted maltreatment counts versus poverty rate.

Maltreatment risk and child fatalities

Figure 2.4 maps locations of child fatalities over the predicted maltreatment risk categories. The associated bar plot shows most child fatalities are occurring in the highest risk categories.

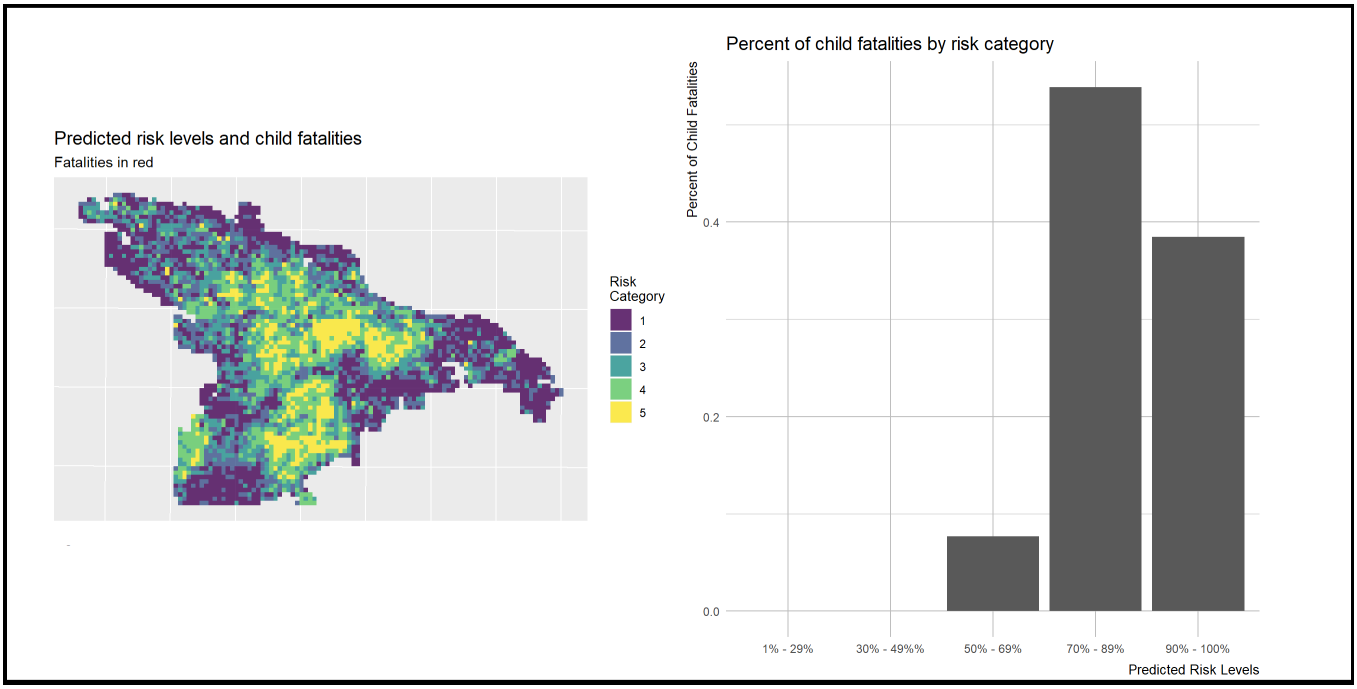


Figure 2.4 Predicted risk categories vs child fatalities.

2.4 Protective land uses

The maltreatment predictions suggest where education, outreach, and prevention efforts should occur. What resources are available at these locations? To answer this question, we use a subset of the original protective factors data aggregated for the model building exercise. Stakeholders can replicate this approach on any list of sites that are suitable to host these interventions.

We then calculate a relative measure of risk exposure for each facility by drawing quarter mile buffers around each site and taking the mean count of predicted events. Figure 2.5 plots these buffers and the relative measure of risk exposure. The table that follows lists the top 20 individual facilities sorted by type and mean predicted count. The rationale for choosing a quarter mile buffer is that it is a walkable distance, making it feasible for residents with transport limitation to reach by foot.

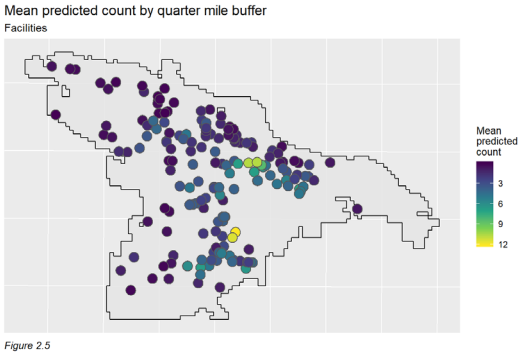


Figure 2.5 Mean predicted counts by 1/4-mile buffer.

The following table lists the ten most optimally located facilities based on mean predicted count of maltreatment events within a quarter mile.

Facility Name	Facility Type	Address	Mean Predicted Count
Kennedy Headstart	Child Care Center	4901 West 65th Street	12
Above And Beyond Child Care	Child Care Center	80 Westminister	11
Buttons & Bows Learning Center Inc	Child Care Center	4124 West 11th Street	11
Loving Care DCFH	Registered Child Care Family Home	3505 West 10th	11
Redeemed for Christ	Child Care Center	3105 W 12TH ST	8
Washington Magnet School	Child Care Center	2700 South Main Street	7
Lrsd Care Program Washington Elementary	Child Care Center	2700 South Main street	7
Stephens Elementary	Child Care Center	3700 West 18th Street	7
In A Childs World	Child Care Center	9420 Chicot Road	7
From A to Z Preschool	Child Care Center	4923 W 12th Street	7

To dig deeper, we will repeat this process for specific types of protective factors, e.g. we can calculate relative measure of risk exposure for Child Care Centers.

2.5 Optimally located protective land uses

2.5.1 Optimally Located Child Care Centers

Figure 2.6 plots the relative measure of risk exposure for quarter mile buffers around childcare centers.

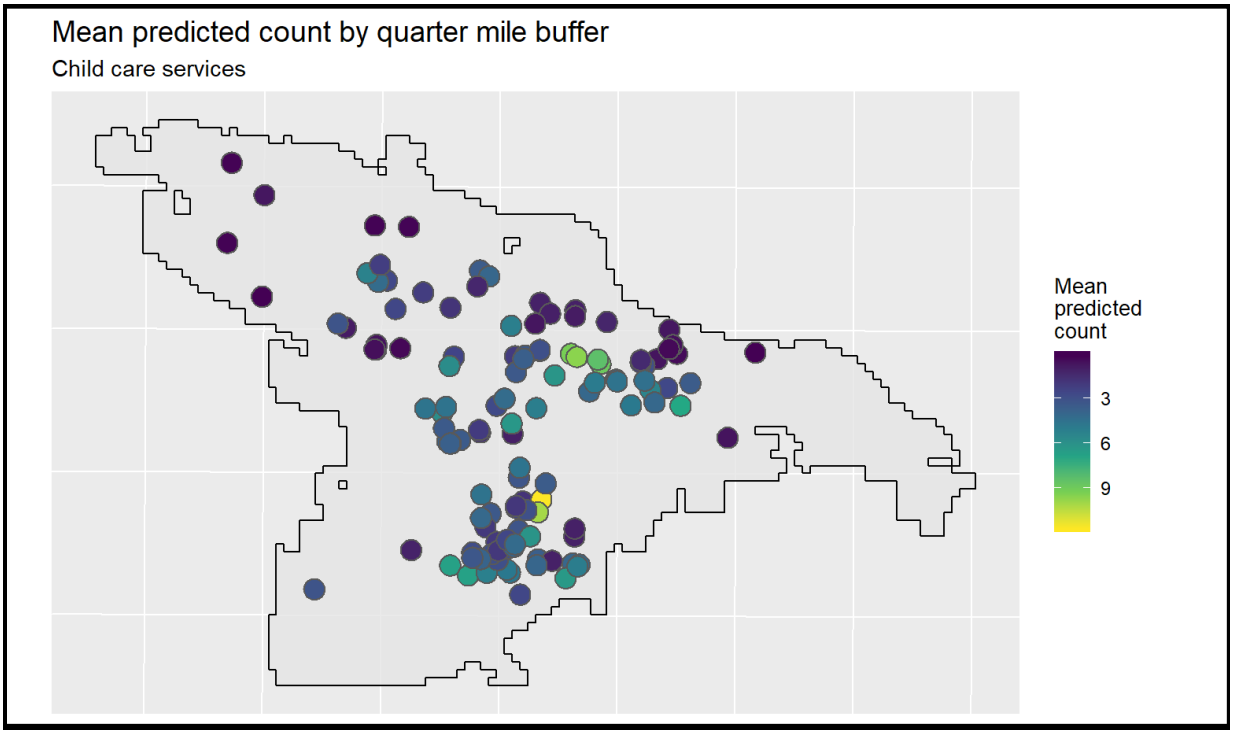


Figure 2.6 14Mean predicted count by childcare services

The following table lists the 5 most optimally located childcare facilities based on mean predicted count of maltreatment events within a quarter mile.

Facility Name	Address	Mean Predicted Count
TOUCHED BY AN ANGEL	6402 BUTLER RD # E	12
KINDLE LOVIN CARE	3914 W 12TH ST	10
ABOVE & BEYOND CHILD CARE CTR	80 WESTMINISTER DR	10
BUTTONS & BOWS LEARNING CTR	4124 W 11TH ST	9
YOUNG'S DAYCARE CTR	1314 BOOKER ST	9

2.5.2 Neighborhood Resource Centers

Figure 2.7 plots the relative measure of risk exposure for quarter mile buffers around neighborhood resource centers.

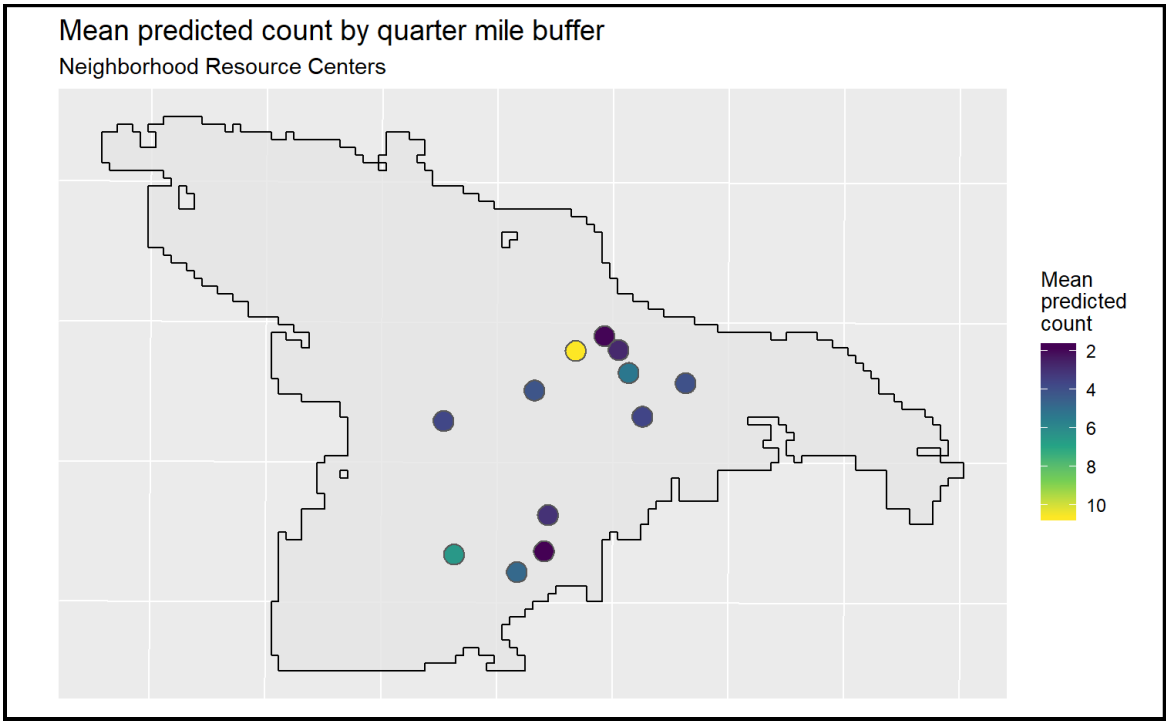


Figure 2.7 15Mean predicted count by neighborhood resource centers

The following table lists the five most optimally located neighborhood resource centers based on mean predicted count of maltreatment events within a quarter mile.

Facility Name	Address	Mean Predicted Count
Willie L Hinton Neighborhood Resource Center	3805 W 12th Street	11
West Baseline Neighborhood Resource Center	9209 Mann Road	7
Wright Avenue Neighborhood Resource Center	1813 Wright Avenue	5
Southwest Little Rock Neighborhood Resource Center	5621 Valley Drive	5
Oak Forest Neighborhood Resource Center	2823 Tyler Street	4

2.5.3 Child Youth Centers

Figure 2.8 plots the relative measure of risk exposure for quarter mile buffers around child youth centers.

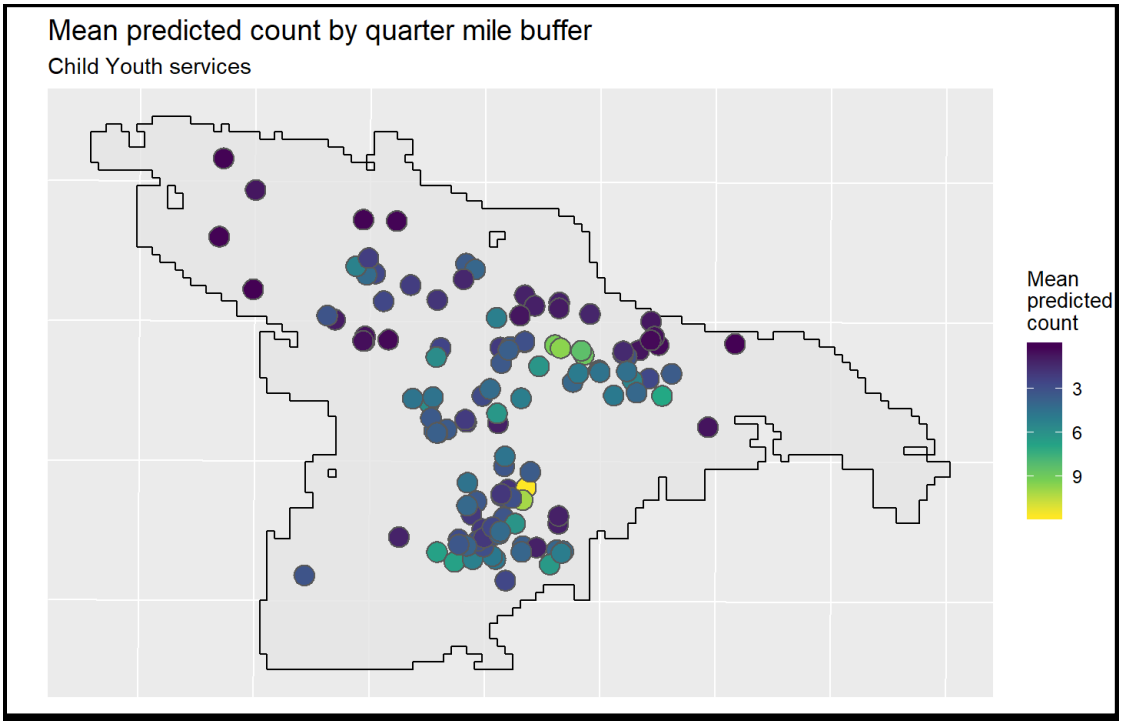


Figure 2.8 16Mean predicted count by child and youth services

The following table lists the 5 most optimally located child youth facilities based on mean predicted count of maltreatment events within a quarter mile.

Facility Name	Address	Mean Predicted Count
DALTON WHETSTONE BOYS & GIRLS	46 HARROW DR	11
B C D YOUTH CTR	4000 W 13TH ST	10
EXTRAORDINARY YOUTH CTR	6105 LANCASTER RD	7
HINTON GRAHAM & ASSOC	100 S UNIVERSITY AVE # 207	5
COUNSELING & PSYCHOLOGY SVC	100 S UNIVERSITY AVE # 200	5

2.5.4 Civil & Social Organizations

Figure 2.9 plots the relative measure of risk exposure for quarter mile buffers around civil social organizations.

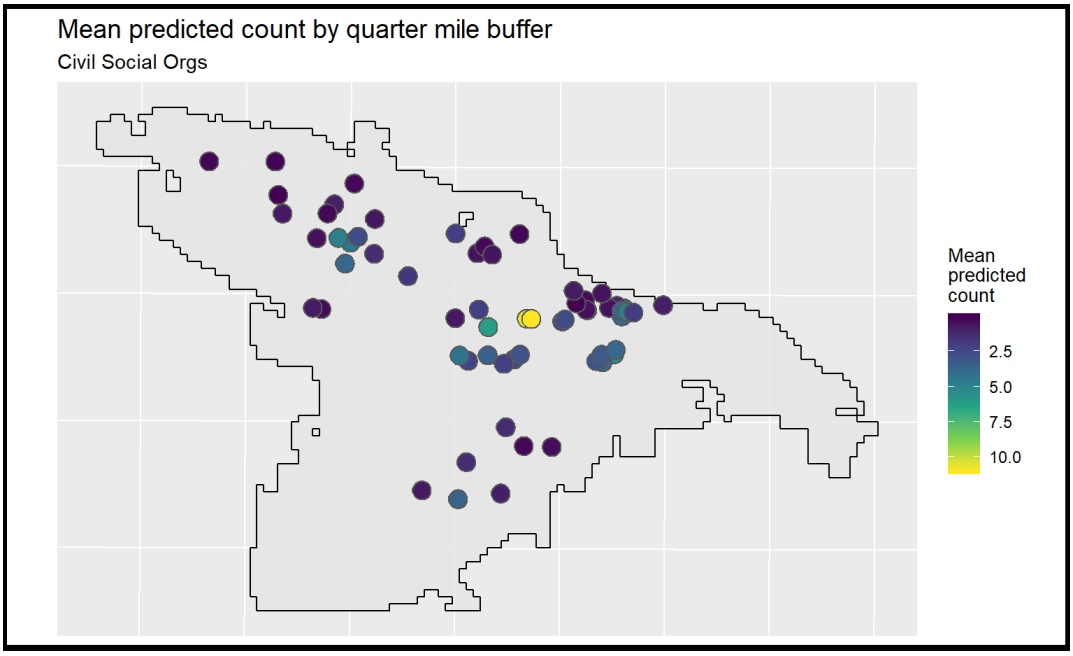


Figure 2.9 17Mean predicted count by civil social organizations

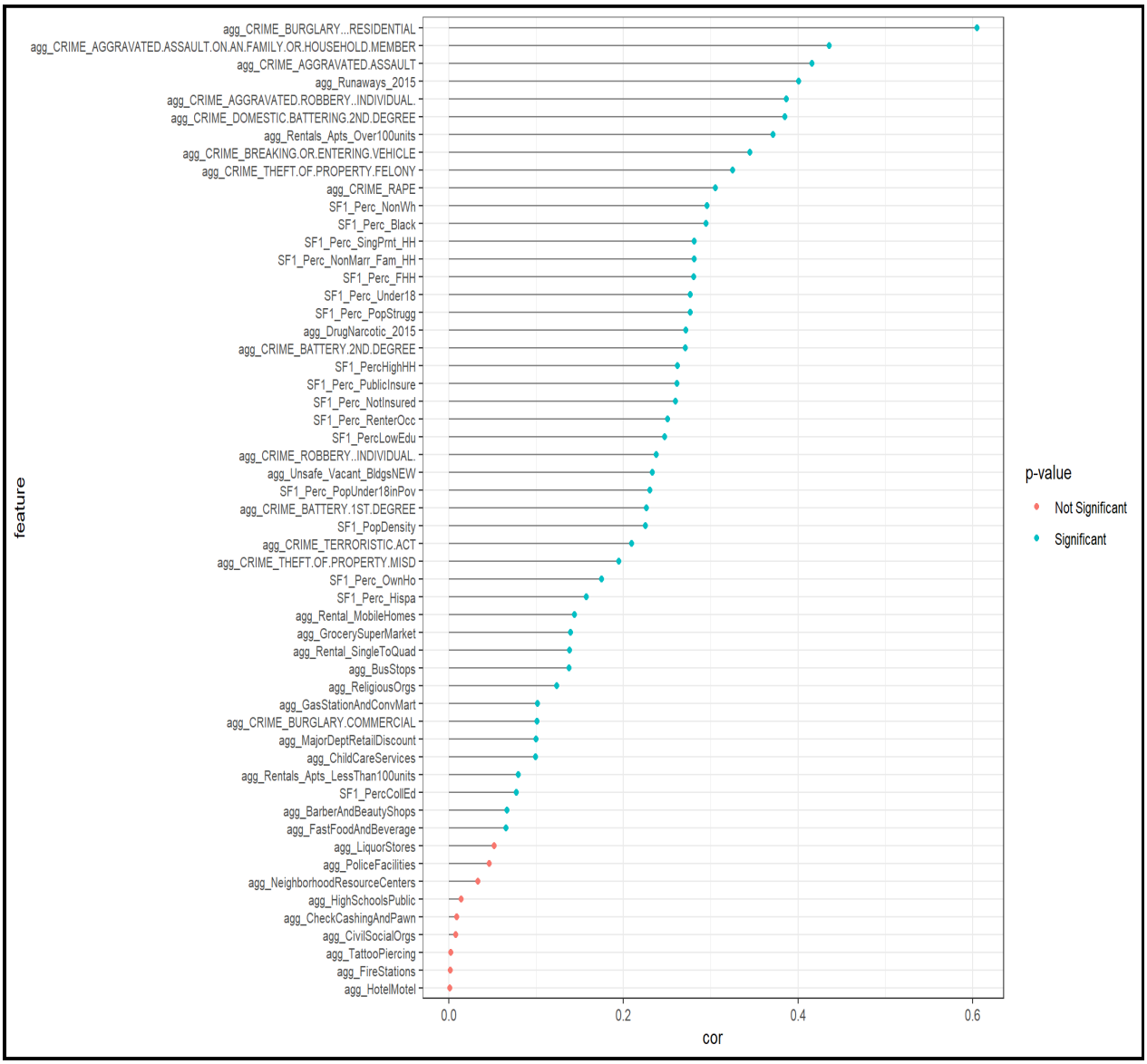
The following table lists the 5 most optimally located civil social organizations based on mean predicted count of maltreatment events within a quarter mile.

Facility Name	Address	Mean Predicted Count
WOMEN'S PROJECT	3802 W 12TH ST	11
B C D INC	3604 W 12TH ST	11
VINE & VILLAGE	1605 FAIR PARK BLVD	6
AMERICAN LEGION MM EBERTS	315 E CAPITOL AVE	5
WOMEN'S PROJECT	2224 MAIN ST	5

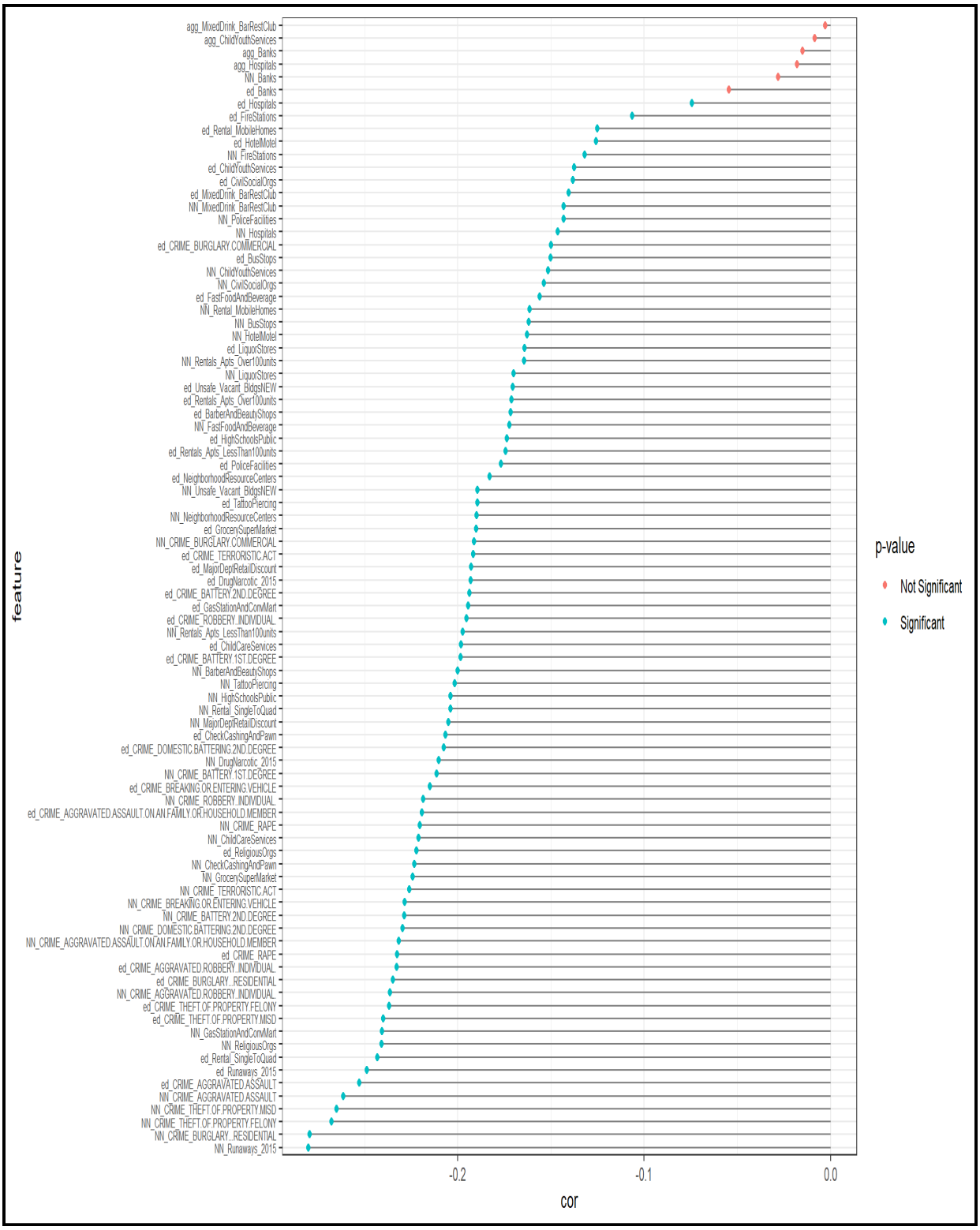
2.6 Supplementary Material

2.6.1 Correlation

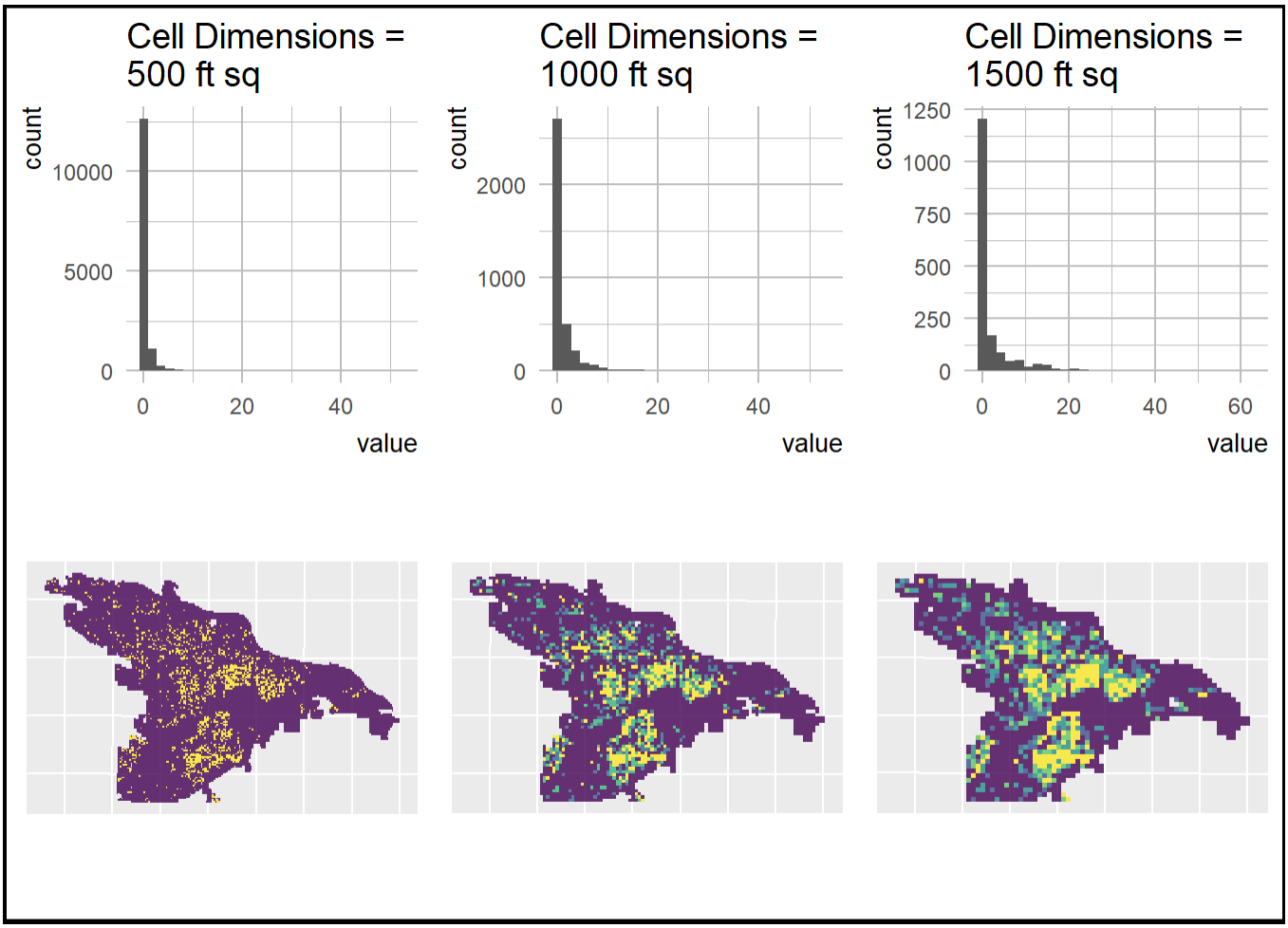
2.6.1.1 Positively correlated variables



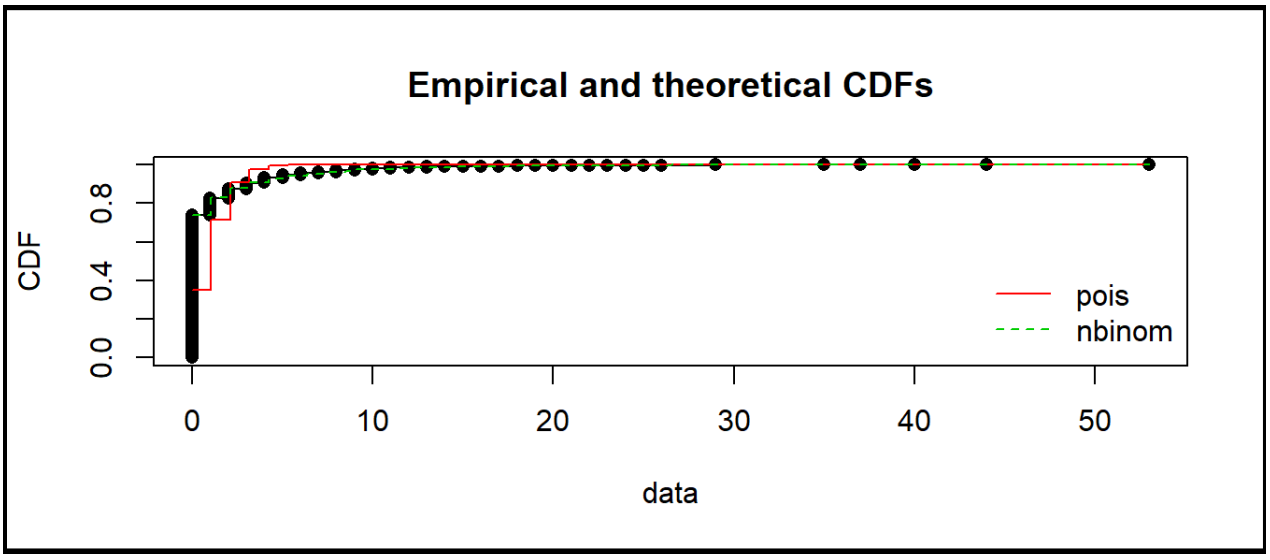
2.6.1.2 Negatively correlated variables

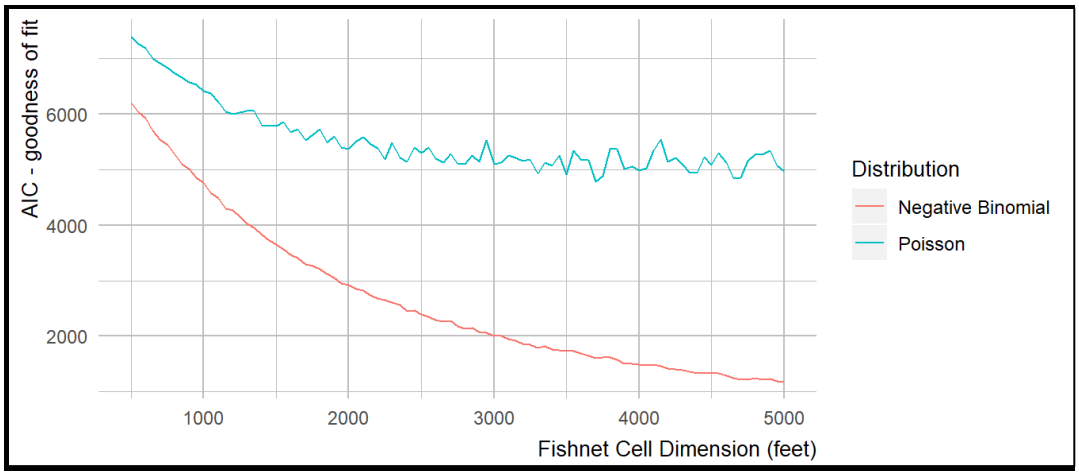


2.6.2 Choosing a fishnet grid size

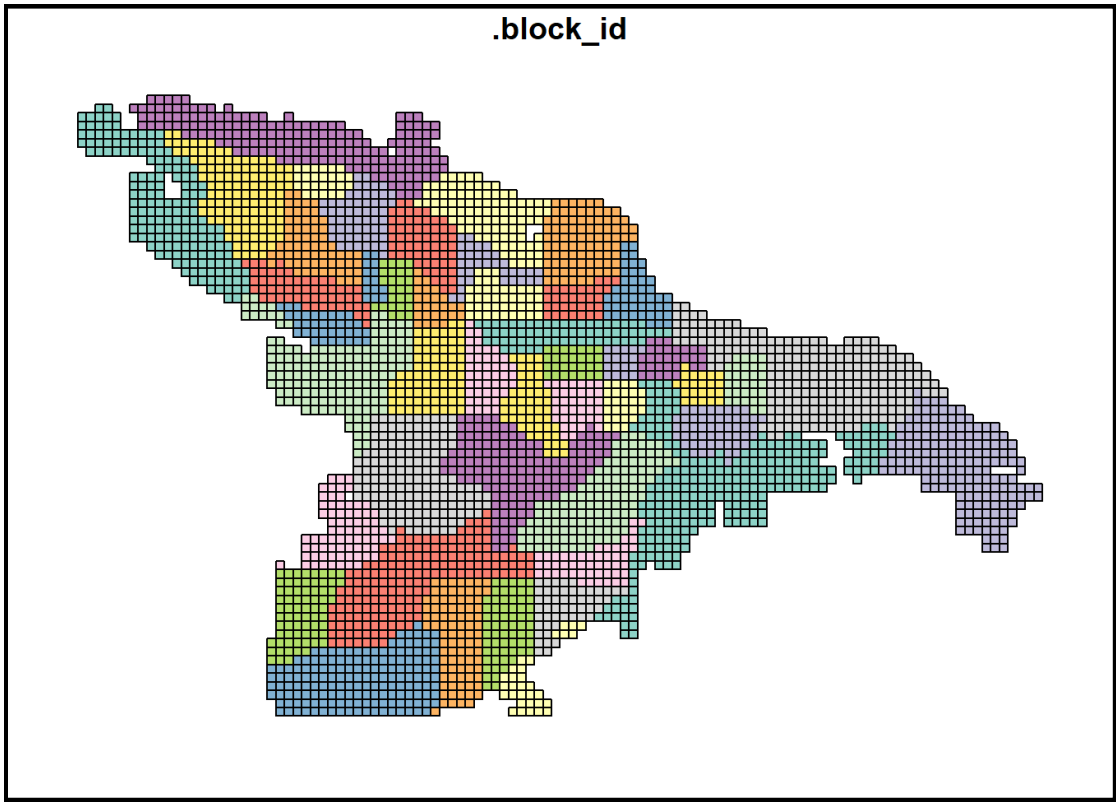


2.6.3 Goodness-of-fit Tests





2.6.4 Tract fixed effects



2.7 Appendix : R Codes

R codes can be found at this location: https://github.com/DattaHub/PAP-child

```
if(T){
  library("sf")            # Spatial data objects and methods
  library("mapview")       # Interactive Map Viewing
  library("ggmap")         # ggplot2 addon for base maps
  library("cowplot")
  library("spatstat")      # KDE and other spatial functions
  library("raster")        # cell-based spatial operations
  library("tidyverse")     # data manipulation framework
  library("Hmisc")         # using cut2() functions for ggplot legends
  library("fitdistrplus")  # Distribution fitting functions
  library("lubridate")     # Power tools for handling dates
  library("tidycensus")
  library("lwgeom")
  library("Hmisc")
  library("hrbrthemes")
  library("gridExtra")
  library("patchwork")
  library("spdep")         # KNN functions
  library("foreach")
  library("doParallel")
  library("corrplot")
  library("ranger")        # randomforest implimentation
  library("glmnet")        # for Ridge and Lasso Regression
  library("knitr")         # for kable table
  library("kableExtra")
  library("FNN")           # KNN for CPS vs. NN plots
  library("groupdata2")
  library("htmltools")
  library("viridis")
  library("viridisLite")
}
mapTheme <- function() {
  theme(
```



```
    plot.title = element_text(size = 14, family = "sans", face = "plain", hjust = 0),
    plot.subtitle=element_text(size = 11, family = "sans", hjust = 0),
    plot.caption=element_text(size = 10, family = "sans", face = "italic", hjust = 0),
    axis.text = element_blank(),
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    axis.ticks = element_blank(),
    axis.line = element_blank(),
    legend.title = element_text(size = 10, family = "sans"),
    legend.text = element_text(size = 9, family = "sans"),
    panel.border = element_blank()
  )
}


plotTheme <- function() {
  theme(
    plot.title = element_text(size = 14, family = "sans", face = "plain", hjust = 0),
    plot.subtitle=element_text(size = 11, family = "sans", hjust = 0),
    plot.caption=element_text(size = 10, family = "sans", face = "italic", hjust = 0),
    axis.title.x = element_text(size = 10, family = "sans", face = "plain", hjust = 1,
vjust = -0.5),
    axis.title.y = element_text(size = 10, family = "sans", face = "plain", hjust = 1,
vjust = 1),
    axis.text = element_text(size = 7, family = "sans", face = "plain"),
    panel.background = element_blank(),
    panel.grid.minor = element_line(colour = "gray"),
    panel.grid.major = element_line(colour = "gray"),
    axis.ticks = element_blank(),
    legend.title = element_text(size = 10, family = "sans"),
    legend.text = element_text(size = 9, family = "sans"),
    axis.line = element_blank()
  )
}
if(T){
  mapviewOptions(basemaps = c("Stamen.TonerLite", "OpenStreetMap.DE"))
  base_dir = "C:/Users/jd033/Box/LR-Project"
  fishnet_grid_dim = 1000
```

```
  k_direction = 8 # 4 = rook, 8 = queen
  k_nearest_neighbors = 5
  # Either k (e.g. 5 or 10) or "LOOCV"
  n_folds = "LOOCV"
  # threshold quntile for statArea grouping
  stat_area_quantile = 0.60
  # Number of simulations for CPS vs. NN
  simulations = 1000
  # Number of neighbors for CPS vs. NN
  k = 5
  # random seed
  set.seed(11235)
}
## Source


source('C:/Users/jd033/Box/Child Maltreatment/R-codes/FUNCTIONS_VAPAP_LR.R', echo = FA
LSE, keep.source = TRUE)
source('C:/Users/jd033/Box/Child Maltreatment/R-codes/FEA_CREATE_VARIABLES_LR_2.R', ec
ho = TRUE, keep.source = TRUE)
```

2.7.1 LR Tracts

```
## LR tracts data
lr_tract = var_list[["LR_Tracts_Working51"]]


TRACT_AREA_plot <- lr_tract %>%
    ggplot(aes(fill = Area)) +
    geom_sf(color = NA) +
    coord_sf(crs = 2765) +
    scale_fill_viridis_c(option = "plasma") +
    labs(title = "Area")


lr_tract_diss <- lr_tract %>%
  mutate(dissolve = 1) %>%
  # get rid of slivers
  st_buffer(., dist = 0.1) %>%
```

```
  group_by(dissolve) %>%
  summarise()


lr_rast_SP <- raster(as(lr_tract_diss, "Spatial"), nrows = 2000, ncol = 2000)
```

2.7.2 Basemap

```
var_list[["CPS_Accepted"]] <-  var_list[["CM_LR_Matched_Centerline_3857"]]


idx_2 = which(var_names == "CM_LR_Matched_Centerline_3857")
var_list[[idx_2]] <- NULL


cps base map   <- get stamenmap(bbox = c(left = -92.52091, bottom = 34.62606, right =
-92.15494, top = 34.82195),
                                    maptype = "toner-lite")


# ggmap(cps_base_map)


## Alternative


lr_base_map <- st_union(lr_tract) %>%
  ggplot()+geom_sf(aes(), fill = "grey85", color = NA, size = 1) +
  mapTheme()


### get CPS_Accepted values (add 1 column for dissolving)
cps_dissolve <- var_list[["CPS_Accepted"]] %>%
  mutate(value = 1) %>%
  dplyr::select(value)
```

2.7.3 Count CPS incidents per net cell

```
net <- st_make_grid(lr_tract, cellsize = fishnet_grid_dim) #%>%st_transform(2756)


# cps_dissolve %>% st_crs() == lr_tract %>% st_crs()


# count CPS incidents per net cell - really just to get net raster into sf polygon for
mat
```

```
net_agg <- aggregate(cps_dissolve, net , sum) %>%
    tibble::rowid_to_column(.,"net_id")

net_agg_vals = net_agg$value[!is.na(net_agg$value)]

# summary(net_agg_vals)

# list of net cells IDs that intersect with Little Rock
net_intersect <- st_intersects(lr_tract, net_agg)

# extract Little Rock net cells based on intersect ID
net_littlerock <- net_agg[unique(unlist(net_intersect)),]
net_hood <- st_join(net_littlerock, lr_tract, largest = TRUE)
listw <- nb2listw(poly2nb(as(net_littlerock, "Spatial"), queen = TRUE))

## Plot fishnet

net_littlerock$value[is.na(net_littlerock$value)] <- 0

FISHNET_plot <- net_littlerock %>%
  ggplot(aes(fill = value)) +
  geom_sf(color = "grey60") +
  coord_sf(crs = 2765) +
  scale_fill_viridis_c(option = "magma") +
  labs(title = "CM per grid-cell")+
  plotTheme()
```

2.7.4 Population and Other Census Data

```
acs <- var_list[["LR_BG_Tracts_ACS_DataJoined"]]

acs = acs %>% dplyr::select(Incident_C, TotPopSize, NLTotPop, PopDensity,Perc_Under,
                            Perc_Black, Perc_NonWh, Perc_Hispa, Perc_NonMa,
                            Perc_FHH,Perc_SingP,PercLowEdu, Perc_Rente,
                            Perc_PopUn, Perc_PopSt, Perc_NotIn, Perc_Publi,
                            PercHighHH, PercCollEd, Perc_OwnHo)
```

```
acs <- acs %>% rename( Incident_Count_sum = Incident_C,
                       Perc_Under18 = Perc_Under,
                       Perc_NonMarr_Fam_HH = Perc_NonMa,
                       Perc_SingPrnt_HH = Perc_SingP,
                       Perc_RenterOcc = Perc_Rente,
                       Perc_PopUnder18inPov = Perc_PopUn,
                       Perc_PopStrugg = Perc_PopSt,
                       Perc_NotInsured = Perc_NotIn,
                       Perc_PublicInsure = Perc_Publi)



acs_pop <- acs %>% dplyr::select(TotPopSize)


## The number 2.29568e-5 is sq ft to acre conversion.
## if you download from ACS using tidycensus, the variable name is "value", here it's
TotPopSize.


acs_pop <- acs_pop %>%
  mutate(acre = as.numeric(st_area(acs)*2.29568e-5),
         # acre = units::set_units(acre, acre),
         pop_acre_rate = TotPopSize / acre)



POP_ACRE_RATE_plot <- acs_pop %>%
  ggplot(aes(fill = pop_acre_rate)) +
  geom_sf(color = NA) +
  coord_sf(crs = 2765) +
  scale_fill_viridis_c(option = "magma") +
  labs(title = "Population per acre")+
  plotTheme()
net_blocks_intersect <- st_intersection(acs_pop, net_littlerock)


# group by cell and calc block stats.
net_blocks_intersect <- net_blocks_intersect %>%
  mutate(intersect_area_acres = as.numeric(st_area(net_blocks_intersect)*2.29568e-5))
%>%
  group_by(net_id) %>%
```



```
  mutate(cnt = n(),
         pcnt_of_block = intersect_area_acres/acre,
         intersect_pop = TotPopSize * pcnt_of_block) %>%
  arrange(net_id)
```

2.7.5 Summarize population

```
## Summarize pop
fishnet_pop <- net_blocks_intersect %>% # xcc
  group_by(net_id) %>%
  summarise(net_pop = sum(intersect_pop)) %>%
  filter(net_pop > 0)   # <-  zeros or no zeros!!!!

BASIC_FISHNET_plot <-fishnet_pop %>%
  ggplot(aes(fill = net_pop)) +
  geom_sf(color = NA) +
  scale_fill_viridis_c(option = "plasma") +
  labs(title = "Net Pop")

######### MAKE NET AND RATE FOR ALL CPS VARS
CPS_vars <- grep("CPS_",names(var_list), value = TRUE)
CPS_agg <- NULL

for(i in seq_along(CPS_vars)){
  var_name <- paste0("net_",CPS_vars[i])
  cat(var_name,"\n")

  CPS_dat <- var_list[[CPS_vars[i]]] %>%
    mutate(value = 1) %>%
    dplyr::select(value)

  fishnet_CPS_var <- aggregate(x = CPS_dat, by = fishnet_pop, FUN = sum) %>%
    st_drop_geometry() %>%
    mutate(Feature = var_name) %>%
    dplyr::select(Feature,value)

  CPS_agg <- rbind(CPS_agg, fishnet_CPS_var)
```

```
}


CPS_agg <- CPS_agg %>%
  mutate(id = rep(seq(1:nrow(fishnet_pop)),length(CPS_vars))) %>%
  spread(Feature, value) %>%
  dplyr::select(-id) %>%
  mutate(geometry = fishnet_pop$geometry) %>%
  st_as_sf()


#### Spatial join of fishnet_pop and fishnet_cps to then calculate rate for all CPS fe
atures


fishnet_pop_cps <- st_join(fishnet_pop, CPS_agg, join = st_equals) %>%
  mutate_at(vars(paste0("net_",CPS_vars)), funs(rate = ./(net_pop/100)))  %>% # cps pe
r 100 person
  # rename_at(vars( contains( "_rate")), .funs = list(paste("rate", gsub("net_|_rate",
"", .), sep = "_"))) %>%
  replace(is.na(.), 0) # replace NA with zero


fishnet_coords <- fishnet_pop_cps %>%
  st_centroid() %>%
  st_coordinates() %>%
  as.matrix()
```

2.7.6 CM Count by fishnet

```
fishnet_pop_cps_cut <- fishnet_pop_cps %>%
  mutate(net_CPS_Accepted = ifelse(is.na(net_CPS_Accepted), 0, net_CPS_Accepted)) %>%
  make_cuts(., "net_CPS_Accepted", cuts = "breaks", n_breaks = 10)


CPS_COUNT_BY_FISHNET_PLOT <- lr_base_map + #ggplot() + #ggmap(cps_base_map) +
  geom_sf(data = ll(fishnet_pop_cps_cut), aes(fill = cut_val), inherit.aes = FALSE, co
lor = NA, alpha = 0.8) +
  labs(title = "CPS count per\nfishnet cell") +
  scale_fill_viridis_d(na.value = NA, option = "D", direction = 1, name = "CPS Count")
+
  mapTheme() +
  theme(plot.title = element_text(size = 14, family = "sans", face = "plain", hjust =
0),
```

```
        plot.subtitle=element_text(size = 11, family = "sans", hjust = 0),
        plot.caption=element_text(size = 10, family = "sans", face = "italic", hjust = 0),
        axis.line = element_blank(),
        legend.title = element_text(size = 10, family = "sans"),
        legend.text = element_text(size = 9, family = "sans"))
fishnet_pop_cps <- fishnet_pop_cps %>% rename(rate_CPS_Accepted = rate)


fishnet_pop_cps_rate_cut <- fishnet_pop_cps %>%
  mutate(rate CPS Accepted = ifelse(is.na(rate CPS Accepted), 0, rate CPS Accepted)) %>%
  make_cuts(., "rate_CPS_Accepted", cuts = "breaks", n_breaks = 10)


CPS_RATE_BY_FISHNET_PLOT <- ggplot() + #ggmap(cps_base_map) +
  geom_sf(data = ll(fishnet_pop_cps_rate_cut), aes(fill = cut_val), inherit.aes = FALSE, color = NA, alpha = 0.8) +
  labs(title = "Child Protective Service rate\nper 100 people") +
  scale_fill_viridis_d(na.value = NA, option = "D", direction = 1, name = "CPS Rate\nper 100") +
  mapTheme() +
  theme(plot.title = element_text(size = 14, family = "sans", face = "plain", hjust = 0),
        plot.subtitle=element_text(size = 11, family = "sans", hjust = 0),
        plot.caption=element_text(size = 10, family = "sans", face = "italic", hjust = 0),
        axis.line = element_blank(),
        legend.title = element_text(size = 10, family = "sans"),
        legend.text = element_text(size = 9, family = "sans"))
```

2.7.7 CPS counts by month and year

```
CPS_Counts_Year_table  <- table(lubridate::year(var_list[["CPS_Accepted"]]$Referral_D))

CPS_Counts_Month_table <- table(lubridate::month(var_list[["CPS_Accepted"]]$Referral_D))
```

2.7.8 CPS histogram by date

```
CPS_by_year <- lubridate::year(var_list[["CPS_Accepted"]]$Referral_D) %>%
  data.frame(year = .)
```

```
CPS_HIST_BY_DATE <- ggplot(CPS_by_year, aes(x = year)) +
  geom_histogram() +
  plotTheme()
```

2.7.9 CPS points by month plot

```
months <- c("January", "February", "March", "April",
            "May", "June", "July","August",
            "September", "October", "November", "December")
cps <- var_list[["CPS_Accepted"]] %>%
  mutate(year  = lubridate::year(Referral_D),
         month = lubridate::month(Referral_D),
         month = months[month],
         month = fct_relevel(month, months))

CPS_POINT_BY_MONTH_plot <- ggplot()+#ggmap(cps_base_map) +
  geom_point(data = data.frame(st_coordinates(ll(cps)), year = cps$year),
             aes(x=X, y=Y, color = as.factor(year)), size=1.5, alpha = 0.8) +
  scale_color_viridis_d(name = "Year") +
  labs(title = "CPS Accepted in Little Rock, AR by Year",
       caption = "source: **************") +
  facet_wrap(~year) +
  mapTheme() +
  theme(
    legend.key = element_rect(fill = "white"),
    strip.text = element_text(face = "plain", size = 11),
    legend.position = c(0.85, 0.25) # or "none
  )
```

2.7.10 CPS KDE by Year plot

```
variable = "year"
values <- unique(cps[[variable]])
year_dat <- list()
brks <- 9
window_cps <- get_window(cps, buff_dist = 10000)
for(i in seq_along(values)){
```

```
  dat <- filter(cps, !!as.name(variable) == values[i])
  points.ppp <- as.ppp(st_coordinates(ll(dat)),window_cps)
  densityRaster <- raster(density(points.ppp, scalekernel=TRUE, sigma = 0.005))
  dens_data <- gplot_data(densityRaster, maxpixels = 2500) %>%
    mutate(!!as.name(variable) := values[i])
  year_dat[[i]] <- dens_data
}
year_dat <- do.call(rbind, year_dat)

CPS_KDE_BY_YEAR_plot <- ggplot() + #ggmap(cps_base_map) +
  geom_tile(data = year_dat,
            aes(x,y,fill = as.factor(ntile(value,brks)),
                group = !!as.name(variable)), alpha=0.8) +
  scale_fill_viridis_d(name = variable) +
  labs(title = "CPS accepted in Little Rock, VA by year",
       caption = "Figure 5.2") +
  facet_wrap(vars(!!as.name(variable))) +
  mapTheme() +
  theme(
    legend.key = element_rect(fill = "white"),
    strip.text = element_text(face = "plain", size = 11, hjust = 0),
    strip.background = element_rect(fill = "white"),
    legend.position = "none"
  )
```

2.7.11 CPS trend by month and year

```
# CPS_KDE_BY_YEAR_plot

CPS_by_year_month <- st_drop_geometry(var_list[["CPS_Accepted"]]) %>%
  mutate(month = lubridate::month(Referral_D),
         year  = lubridate::year(Referral_D))%>%
  dplyr::select(month, year) %>%
  group_by(month, year) %>%
  mutate(m_count = n()) %>%
  distinct() %>%
  ungroup()
```

```
CPS_TREND_BY_MONTH_YEAR_plot <- ggplot(CPS_by_year_month, aes(x = year, y = m_count))
+
  geom_point() +
  geom_smooth(method = lm, formula = y ~ splines::bs(x, 3)) +
  labs(y="Incidents per month") +
  plotTheme()
```

2.7.12 CPS Line Agg by Month

```
CPS_agg_by_month <- st_drop_geometry(var_list[["CPS_Accepted"]]) %>%
  mutate(month = lubridate::month(Referral_D),
         year  = lubridate::year(Referral_D))%>%
  group_by(month) %>%
  summarise(count = n())

CPS_LINE_AGG_BY_MOTNH_plot <- ggplot(CPS_agg_by_month, aes(x = month, y = count)) +
  scale_x_continuous(breaks = seq(1,12), labels = seq(1,12)) +
  geom_line() +
  plotTheme()
CPS_normalized_by_month <- st_drop_geometry(var_list[["CPS_Accepted"]]) %>%
  mutate(month = lubridate::month(Referral_D),
         year  = lubridate::year(Referral_D)) %>%
  group_by(year, month) %>%
  summarise(m_total = n()) %>%
  arrange(month, year) %>%
  dplyr::select(month, year, m_total) %>%
  ungroup() %>%
  group_by(month) %>%
  mutate(m_mean = mean(m_total),
         m_sd   = sd(m_total),
         m_z    = (m_total - m_mean) / m_sd)

CPS_LINE_NORMALIZED_plot <- ggplot(CPS_normalized_by_month, aes(x = as.factor(month),
                                                                y = m_z, group = year,
                                                                color = as.factor(year
))) +
```

```
  geom_line() +
  geom_hline(yintercept = 0, color = "gray20", linetype = "dashed") +
  scale_color_viridis_d(name = "year") +
  labs(x = "month") +
  scale_y_continuous(limits = c(-2,2)) +
  plotTheme()

CPS_LINE_NORMALIZED_plot
```

2.7.13 CPS Calendar plot

```
CPS_agg_cal <- st_drop_geometry(var_list[["CPS_Accepted"]]) %>%
  mutate() %>%
  mutate(day = factor(weekdays(Referral_D,T),
                      levels = rev(c("Mon", "Tue", "Wed", "Thu","Fri", "Sat", "Sun")))
,
         week = week(Referral_D),
         month = month(Referral_D),
         year  = year(Referral_D)) %>%
  dplyr::select(day, week, month, year) %>%
  group_by(day, week, month, year) %>%
  summarise(day_cnt = n()) %>%
  complete(day, week, month, year)

CPS_CALENDAR_plot <- ggplot(CPS_agg_cal, aes(x = week, y = day, fill = day_cnt)) +
  viridis::scale_fill_viridis(name="Incidents",
                              option = 'C',
                              direction = 1,
                              na.value = "gray90") +
  geom_tile(color = 'white', size = 0.1) +
  facet_wrap('year', ncol = 1) +
  scale_x_continuous(
    expand = c(0, 0),
    breaks = seq(1, 52, length = 12),
    labels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
               "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")) +
  theme_ipsum_rc()
```

2.7.14 CPS compare fishnet grid size

```
grid_seq <- c(500,1000,1500)
p_loc_l  <- vector(mode = "list", length = length(grid_seq))
p_hist_l <- vector(mode = "list", length = length(grid_seq))
for(i in seq_along(grid_seq)){
  cat(grid_seq[i], "\n")
  net_i <- st_make_grid(lr_tract, cellsize = grid_seq[i])
  net_agg_i <- aggregate(cps_dissolve, net_i, sum) %>%
    mutate(value = ifelse(is.na(value),0,value))

  net_intersect_i <- st_intersects(lr_tract, net_agg_i)
  # extract Little Rocks net cells based on intersect ID
  net_littlerock_i <- net_agg_i[unique(unlist(net_intersect_i)),]

  net_littlerock_i$class <- Hmisc::cut2(net_littlerock_i$value, g = 9)
  p_loc <- ggplot() + #ggmap(cps_base_map) +
    geom_sf(data = ll(net_littlerock_i), aes(fill = class),
            color = NA, inherit.aes = FALSE, size = 0.5, alpha = 0.8) +
    scale_fill_viridis_d(na.value=NA,
                         name   = paste0("Values","\n[quantiles]"),
                         breaks = levels(net_agg_i$class),
                         labels = levels(net_agg_i$class)) +
    mapTheme()

  p_loc_l[[i]] <- p_loc

  p_hist <- ggplot(net_littlerock_i, aes(x=value)) +
    geom_histogram(bins = 30) +
    # scale_x_continuous(limits = c(-1,100)) +
    # scale_y_continuous(limits = c(0,15)) +
    labs(title = paste0("Cell Dimensions =\n",grid_seq[i]," ft sq")) +
    plotTheme()

  p_hist_l[[i]] <- p_hist
}
```

```
CPS_COMPARE_FISHNET_GRID_SIZE_3x2_plot <- grid.arrange(p_hist_l[[1]], p_hist_l[[2]], p
_hist_l[[3]], p_loc_l[[1]], p_loc_l[[2]], p_loc_l[[3]], ncol = 3)


CPS_COMPARE_FISHNET_GRID_SIZE_3x2_plot
number <- as.numeric(na.omit(fishnet_pop_cps$net_CPS_Accepted))
fitp <- fitdist(number,"pois", discrete = TRUE)
fitnb <- fitdist(number,"nbinom", discrete = TRUE)
cdfcomp(list(fitp,fitnb)) # plot
gof <- gofstat(list(fitp,fitnb))
```

2.7.15 AIC calculation for Poisson and Negative Binomial

```
net_cell_dims <- seq(500,5000,50)
aic_results <- matrix(nrow=length(net_cell_dims), ncol = 3)
colnames(aic_results) <- c("cell_dim","pois","nbinom")
for(i in seq_along(net_cell_dims)){
  net <- st_make_grid(lr_tract,cellsize=net_cell_dims[i])

  cps_cnt <- aggregate(cps_dissolve, net, sum)

  number <- as.numeric(na.omit(cps_cnt$value))
  fitp <- fitdist(number,"pois", discrete = TRUE)
  fitnb <- fitdist(number,"nbinom", discrete = TRUE)
  gof <- gofstat(list(fitp,fitnb))
  aic_results[i,1] <- net_cell_dims[i]
  aic_results[i,2] <- as.numeric(gof$bic[1])
  aic_results[i,3] <- as.numeric(gof$bic[2])
}

AIC_LINE_FITDISTR_plot <- data.frame(aic_results) %>%
  gather(dist, aic, -cell_dim) %>%
  rename("Distribution" = dist) %>%
  mutate(Distribution = case_when(
    Distribution == "nbinom" ~ "Negative Binomial",
    Distribution == "pois"   ~ "Poisson"
  )) %>%
```

```
  ggplot(., aes(x = cell_dim, y = aic, group = Distribution, color = Distribution)) +
  geom_line() +
  labs(y = "AIC - goodness of fit",
       x = "Fishnet Cell Dimension (feet)") +
  plotTheme()
```

2.7.16 Protective and Risk Variables

```
protective_names <-  c("Banks",
                       "GrocerySuperMarket",
                       "HighSchoolsPublic",
                       "HotelMotel",
                       "ChildCareServices",
                       "ChildYouthServices",
                       "CivilSocialOrgs",
                       "Hospitals",
                       "NeighborhoodResourceCenters",
                       "PoliceFacilities",
                       "ReligiousOrgs")

risk_names <- c( "CRIME_THEFT OF PROPERTY FELONY",
                 "CRIME_BURGLARY - RESIDENTIAL",
                 "CRIME_AGGRAVATED ASSAULT",
                 "CRIME_TERRORISTIC ACT",
                "CRIME_THEFT OF PROPERTY MISD",
                "CRIME_RAPE",
                "CRIME_BATTERY 2ND DEGREE",
                "CRIME_DOMESTIC BATTERING 2ND DEGREE",
                "CRIME_BREAKING OR ENTERING VEHICLE" ,
                "CRIME_AGGRAVATED ROBBERY (INDIVIDUAL)",
                "CRIME_ROBBERY (INDIVIDUAL)" ,
                "CRIME_AGGRAVATED ASSAULT ON AN FAMILY OR HOUSEHOLD MEMBER",
                "CRIME_BURGLARY COMMERCIAL" ,
                "CRIME_BATTERY 1ST DEGREE",
                "BarberAndBeautyShops",
                "BusStops",
                "CheckCashingAndPawn",
```

```
                "FastFoodAndBeverage",
                "GasStationAndConvMart",
                "HotelMotel",
                "LiquorStores",
                "MajorDeptRetailDiscount",
                "MixedDrink_BarRestClub",
                "Rental_MobileHomes",
                "Rental_SingleToQuad",
                "Rentals_Apts_LessThan100units",
                "Rentals_Apts_Over100units",
                "TattooPiercing",
                "Unsafe_Vacant_BldgsNEW")


risk_var_list <- var_list[grep(paste(risk_names,collapse="|"), names(var_list), value
= TRUE)]
protective_var_list <- var_list[grep(paste(protective_names,collapse="|"), names(var_l
ist), value = TRUE)]
risk_plot_dat <- list()
brks <- 9
window_cps <- get_window(cps, buff_dist = 10000)
for(i in seq_along(risk_var_list)){
  var_dat <- risk_var_list[[i]]
  points.ppp <- as.ppp(st_coordinates(ll(var_dat)),window_cps)
  densityRaster <- raster(density(points.ppp, scalekernel=TRUE, sigma = 0.005))
  dens_data <- gplot_data(densityRaster, maxpixels = 2500) %>%
    mutate(variable = names(risk_var_list)[i])
  risk_plot_dat[[i]] <- dens_data
}
risk_plot_dat <- do.call(rbind, risk_plot_dat)


# one-liner to extract all 'geometry' cols from list and rbind
risk_compile <- sf::st_as_sf(data.table::rbindlist(lapply(risk_var_list, '[', "geometr
y")))
risk.points.ppp <- as.ppp(st_coordinates(ll(risk_compile)),window_cps)
risk_densityRaster <- raster(density(risk.points.ppp, scalekernel=TRUE, sigma = 0.005)
)
risk_aggregate_plot_data <- gplot_data(risk_densityRaster, maxpixels = 2500) %>%
  mutate(variable = "Risk")
```



```
## KDE Risk
# risk_plot_xy <- risk_plot_dat %>% dplyr::select(x,y)
# library(proj4)
# proj4string <- "+proj=utm +zone=19 +south +ellps=WGS84 +datum=WGS84 +units=m +no_def
s "
# pj = project(risk_plot_xy, proj4string, inverse=TRUE)
# risk_latlon <- data.frame(lat=pj$y, lon=pj$x)
# risk_plot_latlon = cbind(risk_plot_dat, risk_latlon)

RISK_KDE_FACET_PLOT <- ggplot() + #ggmap(cps_base_map) +
  geom_tile(data = risk_plot_dat,
            aes(x,y, fill = as.factor(ntile(value,brks)),
                group = variable), alpha=0.8) +
  scale_fill_viridis_d(name = variable) +
  facet_wrap(~variable) +
  labs(title = "Spatial density of risk factors",
       caption = "Figure 5.4") +
  mapTheme() +
  theme(
    legend.key = element_rect(fill = "white"),
    strip.text = element_text(face = "plain", size = 11, hjust = 0),
    legend.position = "none",
    strip.background = element_rect(fill = "white")
  )

# RISK_KDE_FACET_PLOT

RISK_KDE_PLOT <- ggplot() + #ggmap(cps_base_map) +
  geom_tile(data = risk_aggregate_plot_data,
            aes(x,y,fill = as.factor(ntile(value,brks)),
                group = variable), alpha=0.6) +
  scale_fill_viridis_d(name = variable) +
  #facet_wrap(~variable) +
  mapTheme() +
  theme(
    legend.key = element_rect(fill = "white"),
    strip.text = element_text(face = "plain", size = 11),
```

```
    legend.position = "none"
  )


# RISK_KDE_PLOT
protective_plot_dat <- list()
window_cps <- get_window(cps, buff_dist = 10000)
for(i in seq_along(protective_var_list)){
  var_dat <- protective_var_list[[i]]
  points.ppp <- as.ppp(st_coordinates(ll(var_dat)),window_cps)
  densityRaster <- raster(density(points.ppp, scalekernel=TRUE, sigma = 0.005))
  dens_data <- gplot_data(densityRaster, maxpixels = 2500) %>%
    mutate(variable = names(protective_var_list)[i])
  protective_plot_dat[[i]] <- dens_data
}
protective_plot_dat <- do.call(rbind, protective_plot_dat)


# one-liner to extract all 'geometry' cols from list and rbind
protective_compile <- sf::st_as_sf(data.table::rbindlist(lapply(protective_var_list, '
[', "geometry")))
protective.points.ppp <- as.ppp(st_coordinates(ll(protective_compile)),window_cps)
protective_densityRaster <- raster(density(protective.points.ppp, scalekernel=TRUE, si
gma = 0.005))
protective_aggregate_plot_data <- gplot_data(protective_densityRaster, maxpixels = 250
0) %>%
  mutate(variable = "Protective")
PROTECTIVE_KDE_FACET_PLOT <- ggplot() + #ggmap(cps_base_map) +
  # geom_point(data = data.frame(st_coordinates(ll(cps)),
  #                              month = cps[[variable]]),
  #            aes(x=X, y=Y), size = 1, color = "gray30", alpha = 0.75) +
  geom_tile(data = protective_plot_dat,
            aes(x,y,fill = as.factor(ntile(value,brks)),
                group = variable), alpha=0.8) +
  scale_fill_viridis_d(name = variable) +
  facet_wrap(~variable) +
  labs(title = "Spatial density of protective factors",
       caption = "Figure 5.3") +
  mapTheme() +
  theme(
```



```
    legend.key = element_rect(fill = "white"),
    strip.text = element_text(face = "plain", size = 11, hjust = 0),
    legend.position = "none",
    strip.background = element_rect(fill = "white")
  )


PROTECTIVE_KDE_PLOT <- ggplot()+  #ggmap(cps_base_map) +
  geom_tile(data = protective_aggregate_plot_data,
            aes(x,y,fill = as.factor(ntile(value,brks)),
                group = variable), alpha=0.6) +
  scale_fill_viridis_d(name = variable) +
  #facet_wrap(~variable) +
  mapTheme() +
  theme(
    legend.key = element_rect(fill = "white"),
    strip.text = element_text(face = "plain", size = 11),
    legend.position = "none"
  )


# PROTECTIVE_KDE_PLOT
fishnet_knn <- knn2nb(knearneigh(fishnet_coords, k_direction))
fishnet_Weights <- nb2listw(fishnet_knn, style="W")
localMorans  <- as.data.frame(localmoran(fishnet_pop_cps$net_CPS_Accepted, fishnet_Wei
ghts))
globalMorans <- moran.mc(fishnet_pop_cps$net_CPS_Accepted, fishnet_Weights, nsim=999)
GLOBAL_MORANS_PERMUTATION_plot <- ggplot(data.frame(res = globalMorans$res)[1:999,,0],
aes(res)) +
  geom_histogram(binwidth = 0.01) +
  geom_vline(aes(xintercept = globalMorans$statistic), colour = "red",size=1) +
  scale_x_continuous(limits = c(-1, 1)) +
  labs(title="Observed and permuted Moran's I",
       x = "Simulated Moran's I Value") +
  plotTheme()
fishnet_pop_cps_morans <- fishnet_pop_cps
fishnet_pop_cps_morans$Ii <- localMorans$Ii
fishnet_pop_cps_morans$pvalue <- localMorans$`Pr(z > 0)`
fishnet_pop_cps_morans <- cbind(fishnet_coords, fishnet_pop_cps_morans)
```

```
fishnet_pop_cps_morans_cut <- make_cuts(fishnet_pop_cps_morans, "net_CPS_Accepted",
                                        cuts = "breaks", n_breaks = 10)


## Next chunk
plot_cps <- lr_base_map + #ggplot() + #ggmap(cps_base_map) +
  geom_sf(data = ll(fishnet_pop_cps_morans_cut), aes(fill = cut_val),
          color = NA, inherit.aes = FALSE, alpha = 0.8) +
  scale_fill_viridis_d(na.value=NA, name = "Maltreatment\nEvents") +
  labs(title = "Panel 1",
       subtitle = "CPS count by fishnet") +
  mapTheme() +
  theme(plot.title = element_text(size = 14, family = "sans", face = "plain", hjust =
0),
        plot.subtitle=element_text(size = 11, family = "sans", hjust = 0),
        plot.caption=element_text(size = 10, family = "sans", face = "italic", hjust =
0),
        axis.line = element_blank(),
        legend.title = element_text(size = 10, family = "sans"),
        legend.text = element_text(size = 9, family = "sans"))


Ii_cut <- fishnet_pop_cps_morans %>%
  mutate(Ii_cut_val = as.character(Hmisc::cut2(.$Ii,
                                               cuts = as.numeric(quantile(round(fishne
t_pop_cps_morans$Ii,2),
                                                                          na.rm=T, p =
seq(0,1,0.25))))))
plot_Ii <- lr_base_map + #ggplot() + #ggmap(cps_base_map) +
  geom_sf(data = ll(Ii_cut), aes(fill = Ii_cut_val),
          color = NA, inherit.aes = FALSE, alpha = 0.8) +
  scale_fill_viridis_d(na.value=NA, name = "I value", option = "D") +
  labs(title = "Panel 2",
       subtitle = "Local Moran's I value") +
  mapTheme() +
  theme(plot.title = element_text(size = 14, family = "sans", face = "plain", hjust =
0),
        plot.subtitle=element_text(size = 11, family = "sans", hjust = 0),
        plot.caption=element_text(size = 10, family = "sans", face = "italic", hjust =
0),
        axis.line = element_blank(),
```



```
        legend.title = element_text(size = 10, family = "sans"),
        legend.text = element_text(size = 9, family = "sans"))


p_cut <- fishnet_pop_cps_morans %>%
  mutate(pval_cut = ifelse(pvalue > 0.05, "Not\nSignificant", "Significant"))


plot_p <- lr_base_map + #ggplot() + #ggmap(cps_base_map) +
  geom_sf(data = ll(p_cut), aes(fill = pval_cut),
          color = NA, inherit.aes = FALSE, alpha = 0.8) +
  scale_fill_viridis_d(na.value=NA, name = "p-value", option = "D") +
  labs(title = "Panel 3",
       subtitle = "Stastically significant\nmaltreatment clusters",
       caption = "Figure 5.5") +
  mapTheme() +
  theme(plot.title = element text(size = 14, family = "sans", face = "plain", hjust =
0),
        plot.subtitle=element_text(size = 11, family = "sans", hjust = 0),
        plot.caption=element_text(size = 10, family = "sans", face = "italic", hjust =
0),
        axis.line = element_blank(),
        legend.title = element_text(size = 10, family = "sans"),
        legend.text = element_text(size = 9, family = "sans"))


MORANS_I_P_plot <- cowplot::plot_grid(plot_cps, plot_Ii, plot_p, ncol =1, align = "hv"
, axis = "lrbt")
#cowplot::plot_grid(plot_cps, plot_Ii, plot_p,rel_widths = c(0.9,0.9,0.9),ncol = 1, al
ign = "v")


# MORANS_I_P_plot
```

2.7.17 Aggregate features

```
cl <- makePSOCKcluster(3)
registerDoParallel(cl)
source('C:/Users/jd033/Box/Child Maltreatment/R-codes/FUNCTIONS_VAPAP_LR.R')
agg_results <- Aggregate_points_Features(var_list, net_littlerock)
ED_results <- Euclidean_point_features(var_list,
                                       lr_rast_SP,
```



```
                                        lr_tract_diss,
                                        net_littlerock)
NN_results <- NN_point_features(var_list, net_littlerock, k_nearest_neighbors)
stopCluster(cl)
```

2.7.18 Aggregating all features, creating correlation plots and fitting three different models

```
sf1_tract <- acs %>% dplyr::select(-Incident_Count_sum, -TotPopSize, -NLTotPop)


sf1_tract <- sf1_tract %>%
  mutate(acre = as.numeric(st_area(acs)*2.29568e-5))


vars_sf1_desc <- sf1_tract %>% st_drop_geometry() %>% dplyr::select(starts_with("P"))%
>% names()


net_blocks_intersect <- st_intersection(sf1_tract, net_littlerock)


# group by cell and calc block stats.
net_blocks_intersect2 <- net_blocks_intersect %>%
  mutate(intersect_area_acres = as.numeric(st_area(net_blocks_intersect)*2.29568e-5))
%>%
  group_by(net_id) %>%
  mutate(cnt = n(),
         pcnt_of_block = intersect_area_acres/acre) %>%
  # intersect_pop = value * pcnt_of_block) %>%
  arrange(net_id) %>%
  mutate_at(vars(matches("^P|^T")), funs(.* pcnt_of_block))


### summarise intersect pops to each net cell and create pop rates for some


fishnet_sf1 <- net_blocks_intersect2 %>% # xcc
  group_by(net_id) %>%
  summarise_at(vars(matches("^P|^H")), funs(sum)) %>%
  dplyr::select(-pcnt_of_block)


## cast data frame to list of variables
```

```
sf1_results <- fishnet_sf1 %>%
  gather(variable, value, -net_id, -geometry) %>%
  mutate(feature_name = paste0("SF1_",variable)) %>%
  group_by(variable) %>%
  nest() %>%
  pull(data)
names(sf1_results) <- paste0("SF1_",setdiff(colnames(fishnet_sf1), c("net_id","geometry")))


##


fishnet_pop_cps_net <- fishnet_pop_cps %>%
  dplyr::select(net_id, net_pop, rate_CPS_Accepted, net_CPS_Accepted) %>%
  rename(cps_rate = rate_CPS_Accepted,
         cps_net  = net_CPS_Accepted)


## NN features combine


features <- data.frame(net_id = NN_results[[1]]$net_id, stringsAsFactors = FALSE)
for(i in  seq_along(NN_results)){
  feat_i <- NN_results[[i]] %>%
    st_drop_geometry() %>%
    dplyr::select(net_id, feature_name, value) %>%
    spread(feature_name, value)
  features <- left_join(features, feat_i, by = "net_id")
}
# join features to our target of cps_rate
NN_features <- features %>%
  left_join(., st_drop_geometry(fishnet_pop_cps_net), by = "net_id")


##


features <- data.frame(net_id = ED_results[[1]][[1]]$net_id, stringsAsFactors = FALSE)
for(i in  seq_along(ED_results[[1]])){
  feat_i <- ED_results[[1]][[i]] %>%
    st_drop_geometry() %>%
    dplyr::select(net_id, feature_name, value = mean_dist ) %>% ### mean_dist  !!!
```

```
    spread(feature_name, value)
  features <- left_join(features, feat_i, by = "net_id")
}
# join features to our target of cps_rate
ED_features <- features %>%
  left_join(., st_drop_geometry(fishnet_pop_cps_net), by = "net_id")

## agg_feature_combine

features <- data.frame(net_id = agg_results[[1]]$net_id, stringsAsFactors = FALSE)
for(i in  seq_along(ED_results[[1]])){
  feat_i <- agg_results[[i]] %>%
    st_drop_geometry() %>%
    dplyr::select(net_id, feature_name, value) %>%
    spread(feature_name, value)
  features <- left_join(features, feat_i, by = "net_id")
}
# join features to our target of cps_rate
agg_features <- features %>%
  left_join(., st_drop_geometry(fishnet_pop_cps_net), by = "net_id")

## sf1-features-combine

features <- data.frame(net_id = sf1_results[[1]]$net_id, stringsAsFactors = FALSE)
for(i in  seq_along(sf1_results)){
  feat_i <- sf1_results[[i]] %>%
    st_drop_geometry() %>%
    dplyr::select(net_id, feature_name, value) %>%
    spread(feature_name, value)
  features <- left_join(features, feat_i, by = "net_id")
}
# join features to our target of cps_rate
sf1_features <- features %>%
  left_join(., st_drop_geometry(fishnet_pop_cps_net), by = "net_id")

## corr feature remove NA
```

```
cor_NN_features <- NN_features %>%
  mutate_all(funs(replace(., is.na(.), 0))) %>%
  dplyr::select(-net_id)

cor_agg_features <- agg_features %>%
  mutate_all(funs(replace(., is.na(.), 0))) %>%
  dplyr::select(-net_id)

cor_ED_features <- ED_features %>%
  mutate(cps_rate = ifelse(is.na(cps_rate),0,cps_rate),
         net_pop = ifelse(is.na(net_pop),0,net_pop)) %>%
  na.omit() %>%
  dplyr::select(-net_id)

cor_sf1_features <- sf1_features %>%
  mutate(cps_rate = ifelse(is.na(cps_rate),0,cps_rate),
         net_pop = ifelse(is.na(net_pop),0,net_pop)) %>%
  na.omit() %>%
  dplyr::select(-net_id)

## combine all features

ALL_FEATURES <- full_join(NN_features, agg_features, by = "net_id") %>%
  full_join(.,ED_features, by = "net_id") %>%
  full_join(.,sf1_features, by = "net_id")
all.equal(ALL_FEATURES$cps_rate.x, ALL_FEATURES$cps_rate.y,
          ALL_FEATURES$cps_rate.x.x, ALL_FEATURES$cps_rate.y.y)

NN_CPS_Accepted <- ALL_FEATURES$NN_CPS_Accepted

ALL_FEATURES <- ALL_FEATURES %>%
  dplyr::select(-cps_rate.y, -cps_rate.x.x, -cps_rate.y.y,
                -cps_net.y, -cps_net.x.x, -cps_net.y.y,
                -net_pop.y, -net_pop.x.x, -net_pop.y.y) %>%
  dplyr::select(-contains("_CPS_")) %>%
  dplyr::rename(cps_net  = cps_net.x,
                cps_rate = cps_rate.x,
```



```
                net_pop  = net_pop.x) %>%
  mutate_all(funs(replace(., is.na(.), 0)))  %>%
  dplyr::rename_all(funs(make.names(.)))
## add NN_CPS_Accepted back in to ALL_FEATURES
ALL_FEATURES$NN_CPS_Accepted <- NN_CPS_Accepted



## Corr all plot

cps_cor_ALL <- cor(ALL_FEATURES)
All_cors <- cps_cor_ALL[,"cps_net"]

p.mat_ALL <- cor.mtest(ALL_FEATURES)$p
p.mat_ALL <- p.mat_ALL[,which(colnames(cps_cor_ALL)=="cps_net")]

cor_ALL_plot <- data.frame(feature = names(All_cors),
                           cor = as.numeric(All_cors),
                           p_value   = p.mat_ALL) %>%
  filter(!(feature %in% c("cps_rate","cps_net","net_pop","net_cps","net_id"))) %>%
  filter(!(feature %in% grep("CPS", names(All_cors),value=T))) %>%
  arrange(desc(cor)) %>%
  mutate(p_value = ifelse(p_value >= 0.05, "Not Significant", "Significant"))

cor_ALL_plot$feature <- factor(cor_ALL_plot$feature,
                               levels=cor_ALL_plot[order(cor_ALL_plot$cor,
                                                         decreasing=F),]$feature)
## corr line positive feature

CORR_LINE_POSITIVE_FEATURE_plot <- ggplot(dplyr::filter(cor_ALL_plot,cor >= 0),
                                          aes(x = feature, y = cor, color = factor(p_v
alue))) +
  geom_segment(aes(x = feature, y = 0, xend = feature, yend = cor), color = "grey50")
+
  geom_point() +
  coord_flip() +
  scale_color_discrete(name = "p-value") +
  theme_bw()+
```



```
  theme(axis.text.y = element_text(size=8))


CORR_LINE_POSITIVE_FEATURE_plot


## corr line negative feature plot


CORR_LINE_NEGATIVE_FEATURE_plot <- ggplot(dplyr::filter(cor_ALL_plot,cor <= 0),
                                          aes(x = feature, y = cor, color = factor(p_v
alue))) +
  geom_segment(aes(x = feature, y = 0, xend = feature, yend = cor), color = "grey50")
+
  geom_point() +
  coord_flip() +
  scale_color_discrete(name = "p-value") +
  theme_bw()+
  theme(axis.text.y = element_text(size=6))


CORR_LINE_NEGATIVE_FEATURE_plot


## features corr strong


features_cor <- cor_ALL_plot %>%
  mutate(feature = as.character(feature)) %>%
  arrange(desc(cor)) %>%
  pull(feature)
top_n <- head(features_cor,10)
bottom_n <- tail(features_cor,10)


features_strong_cor <- ALL_FEATURES %>%
  dplyr::select(top_n, bottom_n, cps_net, cps_rate, net_pop, net_id) %>%
  base::identity()



### Now this line has to be modified to meet our needs
features_protective_all <- ALL_FEATURES %>%
  dplyr::select(contains("Banks"),
                contains("GrocerySuperMarket"),
```



```
                contains("HighSchoolsPublic"),
                contains("HotelMotel"),
                contains("ChildCareServices"),
                contains("ChildYouthServices"),
                contains("CivilSocialOrgs"),
                        contains("Hospitals"),
                        contains("NeighborhoodResourceCenters"),
                        contains("PoliceFacilities"),
                        contains("ReligiousOrgs"),
                NN_CPS_Accepted,
                cps_net, cps_rate, net_pop, net_id)

features_strong_protective_names <- cor_ALL_plot %>%
  filter(feature %in% names(features_protective_all)) %>%
  mutate(prefix = str_extract(feature, "^[^_]+(?=_)"),
         suffix = str_extract(feature, "(?<=_)[^_].*"),
         feature = as.character(feature)) %>%
  group_by(suffix) %>%
  slice(which.max(abs(cor)))

features_protective_strong <- features_protective_all %>%
  dplyr::select(features_strong_protective_names$feature,
                NN_CPS_Accepted,
                cps_net, cps_rate, net_pop, net_id) %>%
  base::identity()


## risk features all

features_risk_all <- ALL_FEATURES %>%
  dplyr::select(contains("CRIME_THEFT.OF.PROPERTY.FELONY"),
                contains("CRIME_BURGLARY...RESIDENTIAL"),
                contains("CRIME_TERRORISTIC.ACT"),
                contains("NN_CRIME_THEFT.OF.PROPERTY.MISD"),
                contains("NN_CRIME_RAPE"),
                contains("NN_CRIME_BATTERY.2ND.DEGREE" ),
                contains("NN_CRIME_DOMESTIC.BATTERING.2ND.DEGREE"),
```



```
                contains("NN_CRIME_BREAKING.OR.ENTERING.VEHICLE" ),
                contains("NN_CRIME_AGGRAVATED.ROBBERY..INDIVIDUAL." ),
                contains("NN_CRIME_ROBBERY..INDIVIDUAL."  ),
                contains("NN_CRIME_AGGRAVATED.ASSAULT.ON.AN.FAMILY.OR.HOUSEHOLD.MEMBER
" ),
                contains("NN_CRIME_BURGLARY.COMMERCIAL" ),
                contains("NN_CRIME_BATTERY.1ST.DEGREE"),
                contains("BarberAndBeautyShops"),
                contains("BusStops"),
                contains("CheckCashingAndPawn"),
                contains("FastFoodAndBeverage"),
                contains("GasStationAndConvMart"),
                contains("HotelMotel"),
                contains("LiquorStores"),
                contains("MajorDeptRetailDiscount"),
                contains("MixedDrink_BarRestClub"),
                contains("Rental_MobileHomes"),
                contains("Rental_SingleToQuad"),
                contains("Rentals_Apts_LessThan100units"),
                contains("Rentals_Apts_Over100units"),
                contains("TattooPiercing"),
                contains("Unsafe_Vacant_BldgsNEW"),
                NN_CPS_Accepted,
                cps_net, cps_rate, net_pop, net_id)

## features_risk_strong

features_risk_strong_names <- cor_ALL_plot %>%
  filter(feature %in% names(features_risk_all)) %>%
  mutate(prefix = str_extract(feature, "^[^_]+(?=_)"),
         suffix = str_extract(feature, "(?<=_)[^_].*"),
         feature = as.character(feature)) %>%
  group_by(suffix) %>%
  slice(which.max(abs(cor)))

features_risk_strong <- features_risk_all %>%
  dplyr::select(features_risk_strong_names$feature,
```



```
                NN_CPS_Accepted,
                cps_net, cps_rate, net_pop, net_id) %>%
  base::identity()

# features_census_select

features_census_select <- ALL_FEATURES %>%
  dplyr::select(SF1_Perc_Under18,
                SF1_Perc_Black,
                SF1_Perc_NonWh,
                SF1_Perc_Hispa,
                SF1_Perc_NonMarr_Fam_HH,
                SF1_Perc_FHH,
                SF1_Perc_SingPrnt_HH,
                SF1_PercLowEdu,
                SF1_Perc_RenterOcc,
                SF1_Perc_PopUnder18inPov,
                SF1_Perc_PopStrugg,
                SF1_Perc_NotInsured,
                SF1_Perc_PublicInsure,
                SF1_PercHighHH,
                SF1_PercCollEd,
                SF1_Perc_OwnHo,
                cps_net, cps_rate, net_pop, net_id)

features_risk_strong_plot <- features_risk_strong %>%
  dplyr::select(-net_id)

CORR_RISK_FEATURES_plot <- feature_corrplot(features_risk_strong_plot, "Correlation of
Risk Features")

features_protective_strong_plot <- features_protective_strong %>%
  dplyr::select(-net_id)
CORR_PROTECTIVE_FEATURES_plot <- feature_corrplot(features_protective_strong_plot, "Co
rrelation of Protective Features")

## Line 1096 is Corr-protective-features-plot
```



```
## Line 1560 is feature prep
## We can safely ignore bunch of plots in between these two chunks
## feature prep

target_var <- "cps_net"
features_protective_strong2 <- dplyr::select(features_protective_strong, -cps_rate, -net_pop)
features_risk_strong2 <- dplyr::select(features_risk_strong, -cps_rate, -net_pop)
features_census_select2     <- dplyr::select(features_census_select, -cps_rate, -net_pop)



## model data prep

full_join(features_risk_strong, features_census_select, by = "net_id") %>%
  full_join(., features_protective_strong, by = "net_id") %>% names()

og_dat <- full_join(features_risk_strong, features_census_select, by = "net_id") %>%
  full_join(., features_protective_strong, by = "net_id") %>%
  dplyr::select(-net_pop.y, -cps_net.y, -cps_rate.y,
                -net_pop.x, -cps_net.x, -cps_rate.x)



dat    <- og_dat %>% dplyr::select(-cps_rate, -net_pop, -net_id) %>%
  mutate_at(vars(-cps_net), scale_this) %>%
  identity() # line ender (does nothing)

net_hood <- st_join(net_littlerock, lr_tract, largest = TRUE)
all.equal(net_hood$net_id, og_dat$net_id)
og_dat$.block_id <- net_hood$NAME

## tract fixed effects

hood_matrix <- model.matrix(cps_net~.block_id,og_dat)
hood_model <- lm(sqrt(og_dat$cps_net) ~ hood_matrix)
dat$hood_fixed <- predict(hood_model, type = "response")^2
og_dat$hood_fixed <- predict(hood_model, type = "response")^2
```



```
## create cv fold_tibble

n_folds = 5

target_var <- "cps_net"

all_hoods <- length(unique(net_hood$name))
n_folds = ifelse(n_folds == "LOOCV", all_hoods, n_folds)
folds_index <- groupdata2::fold(og_dat, k = n_folds, id_col = '.block_id')$.folds

cv_tbl <- tibble(folds = seq_len(n_folds),
                 train = NA, train_y = NA, train_index = NA, train_net_id = NA,
                 test  = NA, test_y  = NA, test_index  = NA, test_net_id  = NA)

for(k in seq_len(n_folds)){
  fold_i  <- which(folds_index == k)
  cv_tbl[k,]$train         <- list(dat[-fold_i,])
  cv_tbl[k,]$test          <- list(dat[ fold_i,])
  cv_tbl[k,]$train_y       <- list(og_dat[-fold_i,target_var])
  cv_tbl[k,]$test_y        <- list(og_dat[ fold_i,target_var])
  cv_tbl[k,]$train_index   <- list(setdiff(seq(1:nrow(dat)),fold_i))
  cv_tbl[k,]$test_index    <- list(fold_i)
  cv_tbl[k,]$train_net_id  <- list(og_dat[-fold_i,"net_id"])
  cv_tbl[k,]$test_net_id   <- list(og_dat[ fold_i,"net_id"])
}

cv_sf <- left_join(og_dat, net_littlerock, by = "net_id") %>%
  st_as_sf() %>%
  dplyr::select(.block_id)
NEIGHBORHOOD_FOLDS_plot <- plot(cv_sf)

## Poisson regression

po_cv_tbl <- cv_tbl %>%
  mutate(fit   = map(train, glm_fit,
```



```
                      formula =  paste("cps_net ~ ."),
                      family = "poisson"),
         pred  = map2(fit, test, lm_predict, sqrt = FALSE),
         mdl_nam = "GLM - Poisson") %>%
  score_model()
cat("Test Set MAE:",mean(po_cv_tbl$MAE),"\n")
cat("Test Set logdev:",mean(po_cv_tbl$logdev, na.rm=TRUE),"\n")


## Poisson Regression fit plot


POISSON_REGRESSION_FIT_plot <- plot_fold_pred(po_cv_tbl$pred, po_cv_tbl$test_y, type =
"fit")


POISSON_REGRESSION_FIT_plot


## Random Forest


rf_cv_tbl <- cv_tbl %>%
  mutate(fit   = map(train, rf_fit, formula = "cps_net ~ .", mtry_add = 2, importance
= "impurity"),
         pred  = map2(fit, test, lm_predict),
         mdl_nam = "Random Forest") %>%
  score_model()
cat("Test Set MAE:",mean(rf_cv_tbl$MAE),"\n")
cat("Test Set logdev:",mean(rf_cv_tbl$logdev, na.rm=TRUE),"\n")



## Random Forest Var Imp Plot


varimp_dat <- data.frame(importance = rf_cv_tbl$fit[[1]]$variable.importance) %>%
  rownames_to_column("variable")


RF_VARIMP_PLOT <- ggplot(varimp_dat, aes(x=reorder(variable,importance), y=importance,
fill=importance))+
  geom_bar(stat="identity", position="dodge")+ coord_flip()+
  labs(y = "Variable Importance",
       x = " ",
       title = "Feature importance",
```



```
       subtitle = "Random Forest sub-model",
       caption = "Figure 6.4") +
  guides(fill=F)+
  scale_fill_viridis_c() +
  plotTheme()+
  theme(axis.text.y = element_text(size = 6))

RF_VARIMP_PLOT

ggsave(file = "RF-Var-Imp-Plot.png",RF_VARIMP_PLOT, height = 9, width = 7)

RANDOM_FOREST_FIT_plot <- plot_fold_pred(rf_cv_tbl$pred, rf_cv_tbl$test_y, type = "fit
")

RANDOM_FOREST_FIT_plot

## Spatial Error Regression

spat_durbin <- errorsarlm(sqrt(cps_net) ~ ., data = dat, listw, etype ="emixed")
spat_durbin_tbl <- tibble(
  fit   = list(spat_durbin),
  pred  = map(fit, sar_pred),
  test_y= list(dat$cps_net),
  test_net_id = list(og_dat$net_id),
  mdl_nam = "Spatial Durbin - sqrt") %>%
  score_model()
cat("Test Set MAE:",mean(spat_durbin_tbl$MAE),"\n")
cat("Test Set logdev:",mean(spat_durbin_tbl$logdev, na.rm=TRUE),"\n")

SPATIAL_ERROR_FIT_plot <- plot_fold_pred(spat_durbin_tbl$pred, dat$cps_net, type = "fi
t")

POISSON_REGRESSION_FIT_plot+RANDOM_FOREST_FIT_plot+SPATIAL_ERROR_FIT_plot



## Gather oof prediction
```

```
po_pred_dat <- po_cv_tbl %>%
  unnest(pred) %>%
  mutate(test_y = po_cv_tbl %>% unnest(test_y) %>% pull(test_y),
         test_net_id = po_cv_tbl %>% unnest(test_net_id) %>% pull(test_net_id))

po_pred_geoplot <- model_pred_geoplot(po_pred_dat$pred,
                                      po_pred_dat$test_y,
                                      po_pred_dat$test_net_id,
                                      net_littlerock, cps_base_map, "po")

rf_pred_dat <- rf_cv_tbl %>%
  unnest(pred) %>%
  mutate(test_y = rf_cv_tbl %>% unnest(test_y) %>% pull(test_y),
         test_net_id = rf_cv_tbl %>% unnest(test_net_id) %>% pull(test_net_id))

rf_pred_geoplot <- model_pred_geoplot(rf_pred_dat$pred,
                                      rf_pred_dat$test_y,
                                      rf_pred_dat$test_net_id,
                                      net_littlerock, cps_base_map,
                                      "Rand
                                      om Forest")
### One special
if(T){
  pred_dat <- data.frame(pred = rf_pred_dat$pred,
                         obs  = rf_pred_dat$test_y,
                         net_id = rf_pred_dat$test_net_id)

  MAE_geoplot <- net_littlerock %>%
    left_join(., pred_dat, by = "net_id") %>%
    mutate(MAE = round(abs(pred - obs),2),
           feature_name = paste0("RF"," ", "MAE")) %>%
    make_cuts(., "MAE")

  MAE_geoplot %>%  ggplot(aes(fill = MAE)) +
    geom_sf(color = NA) +
    scale_fill_viridis_d() +
    labs(title = "MAE")
```

```
}



###
sarlm_pred_dat <- spat_durbin_tbl %>%
  unnest(pred) %>%
  mutate(test_y = spat_durbin_tbl %>% unnest(test_y) %>% pull(test_y),
         test_net_id = spat_durbin_tbl %>% unnest(test_net_id) %>% pull(test_net_id))

sarlm_pred_geoplot <- model_pred_geoplot(sarlm_pred_dat$pred,
                                         sarlm_pred_dat$test_y,
                                         sarlm_pred_dat$test_net_id,
                                         net_littlerock, cps_base_map,
                                         "SARLM")



cps_preds <- og_dat %>%
  dplyr::select(net_id, cps_net) %>%
  left_join(., dplyr::select(po_pred_dat,
                             net_id = test_net_id,
                             pred_lm = pred), by = "net_id") %>%
  left_join(., dplyr::select(rf_pred_dat,
                             net_id = test_net_id,
                             pred_rf = pred), by = "net_id") %>%
  left_join(., dplyr::select(sarlm_pred_dat,
                             net_id = test_net_id,
                             pred_sarlm = pred), by = "net_id") %>%
  mutate_if(is.double, round, 2)



## Meta model stacking


if(all.equal(cps_preds$net_id, net_hood$net_id)){
  cat("Predictions and spatial data are in same order, GOOD to go!", "\n")
} else {
  cat("There is a PROBLEM with order of predictions and spatial data; Likely Errors!",
"\n")
```



```
}

cps_preds_cv_dat <- dplyr::select(cps_preds, -net_id)
ens_cv_tbl <- tibble(folds = seq_len(n_folds),
                     train = NA, train_y = NA, train_index = NA, train_net_id = NA,
                     test  = NA, test_y  = NA, test_index  = NA, test_net_id  = NA)
for(k in seq_len(n_folds)){
  fold_i  <- which(folds_index == k)
  ens_cv_tbl[k,]$train         <- list(cps_preds_cv_dat[-fold_i,])
  ens_cv_tbl[k,]$test          <- list(cps_preds_cv_dat[ fold_i,])
  ens_cv_tbl[k,]$train_y       <- list(cps_preds_cv_dat[-fold_i,target_var])
  ens_cv_tbl[k,]$test_y        <- list(cps_preds_cv_dat[ fold_i,target_var])
  ens_cv_tbl[k,]$train_index   <- list(setdiff(seq(1:nrow(cps_preds_cv_dat)),fold_i))
  ens_cv_tbl[k,]$test_index    <- list(fold_i)
  ens_cv_tbl[k,]$train_net_id  <- list(cps_preds[-fold_i,"net_id"])
  ens_cv_tbl[k,]$test_net_id   <- list(cps_preds[ fold_i,"net_id"])
}

ens_cv_tbl <- ens_cv_tbl %>%
  mutate(fit   = map(train, rf_fit, formula = "cps_net ~ pred_rf + pred_sarlm"),
         pred  = map2(fit, test, lm_predict),
         # pred  = map(pred, round),
         mdl_nam = "Meta-Model") %>%
  score_model()

cat("Test Set MAE:",mean(ens_cv_tbl$MAE),"\n")
cat("Test Set logdev:",mean(ens_cv_tbl$logdev),"\n")

## Meta model fit plot

META_MODEL_FIT_plot <- plot_fold_pred(ens_cv_tbl$pred, ens_cv_tbl$test_y, type = "fit"
) +
  labs(x = "Observed Maltreatment Counts",
       y = "Predicted Maltreatment Counts",
       title = "Predicted vs. observed maltreatment counts",
       caption = "Figure 1.7") +
  plotTheme() +
```



```
  theme(panel.border = element_blank())

## join meta moodel predictions

ens_pred_dat <- ens_cv_tbl %>%
  unnest(pred) %>%
  mutate(test_y = ens_cv_tbl %>% unnest(test_y) %>% pull(test_y),
         test_net_id = ens_cv_tbl %>% unnest(test_net_id) %>% pull(test_net_id))

ens_pred_geoplot <- model_pred_geoplot(ens_pred_dat$pred,
                                       ens_pred_dat$test_y,
                                       ens_pred_dat$test_net_id,
                                       net_littlerock, cps_base_map,
                                       "Meta-Model")
cps_preds2 <- cps_preds %>%
  left_join(., dplyr::select(ens_pred_dat,
                             net_id = test_net_id,
                             pred_ens = pred) %>%
              mutate(pred_ens = round(pred_ens,2)), by = "net_id")

## PREDICTION_MAP_plots

POISSON_MODEL_PREDICTION_MAP_plot <- cowplot::plot_grid(po_pred_geoplot[[2]] +
                                                          labs(title = "Poisson Regres
sion",
                                                               subtitle = "Predicted M
altreatment Count"),
                                                        po_pred_geoplot[[1]] +
                                                          labs(subtitle = "MAE") +
                                                          scale_fill_viridis_d(name =
"MAE"),
                                                        align = "h")

POISSON_MODEL_PREDICTION_MAP_plot

RF_MODEL_PREDICTION_MAP_plot <- cowplot::plot_grid(rf_pred_geoplot[[2]] +
                                                     labs(title = "Random Forest",
```

```
                                                        subtitle = "Predicted Maltre
atment Count"),
                                                rf_pred_geoplot[[1]] +
                                                  labs(subtitle = "MAE") +
                                                  scale_fill_viridis_d(name = "MAE"
),
                                                align = "h")

RF_MODEL_PREDICTION_MAP_plot


SARLM_MODEL_PREDICTION_MAP_plot <- cowplot::plot_grid(sarlm_pred_geoplot[[2]] +
                                                    labs(title = "Spatial Durbin M
odel",
                                                         subtitle = "Predicted Mal
treatment Count") +
                                                    mapTheme() +
                                                    theme(panel.border = element_b
lank()),
                                                  sarlm_pred_geoplot[[1]] +
                                                    labs(subtitle = "MAE") +
                                                    scale_fill_viridis_d(name = "M
AE") +
                                                    mapTheme() +
                                                    theme(panel.border = element_b
lank()),
                                                  align = "h")

META_MODEL_PREDICTION_MAP_plot <- cowplot::plot_grid(ens_pred_geoplot[[2]] +
                                                   labs(title = "Meta-Model",
                                                        subtitle = "Predicted Malt
reatment Count",
                                                        caption = "Figure 6.2") +
                                                   mapTheme() +
                                                   theme(panel.border = element_bl
ank()),
                                                 ens_pred_geoplot[[1]] +
                                                   labs(subtitle = "MAE") +
                                                   scale_fill_viridis_d(name = "MA
E") +
                                                   mapTheme() +
```



```
                                                        theme(panel.border = element_bl
ank()),
                                                    align = "h")

### r model_error_by_decile
models <- bind_rows(rf_cv_tbl, spat_durbin_tbl, ens_cv_tbl, po_cv_tbl)

CV_preds_long <- models %>%
  group_by(mdl_nam) %>%
  unnest(pred, test_y)

## map over all quantiles to get error metrics
quantile_errors <- CV_preds_long %>%
  nest(-mdl_nam) %>%
  mutate(q      = list(seq(0,1,0.01)),
         pred   = map(data, "pred"),
         test_y = map(data, "test_y")) %>%
  dplyr::select(-data) %>%
  unnest(q, .preserve = c(pred, test_y)) %>%
  filter(q != 0) %>%
  mutate(q_dat  = pmap(list(pred, test_y, q), quantile_error),
         q_pred = map(q_dat, "pred"),
         q_obs  = map(q_dat, "obs"),
         q_RMSE = map2_dbl(q_pred, q_obs, rmse),
         q_MAE  = map2_dbl(q_pred, q_obs, mae),
         q_logdev  = map2_dbl(q_pred, q_obs, logdev_p),
         y_max  = quantile(seq(0,max(dat$cps_net)), q),
         q_cnt  = nrow(og_dat) - map_int(q_dat, nrow))

q_error_plotdat <- quantile_errors %>%
  dplyr::select(mdl_nam, q, q_RMSE, q_MAE, q_logdev)
q_cnt_plotdat <- quantile_errors %>%
  dplyr::select(mdl_nam, q, y_max, q_cnt) %>%
  filter(q != 0) %>%
  mutate(q_pcnt = (q_cnt / nrow(og_dat)))
q_error_mean <- q_error_plotdat %>%
  group_by(mdl_nam) %>%
```



```
  summarise(mean_RMSE = mean(q_RMSE, na.rm = TRUE),
            mean_MAE  = mean(q_MAE, na.rm = TRUE),
            mean_logdev  = mean(q_logdev, na.rm = TRUE)) %>%
  arrange(desc(mean_logdev))
print(q_error_mean)


## Appendix 4e - Model Error Table


# Helper function for quantile error
quantile_error <- function(pred,obs,quant){
  preds <- data.frame(pred = pred, obs = obs) %>%
    filter(quantile(seq(0,max(obs)), quant)>obs)
  return(preds)
}
# Join/bind model prediction tables
models <- bind_rows(rf_cv_tbl, spat_durbin_tbl, ens_cv_tbl, po_cv_tbl)
# Unnest predictions by model
CV_preds_long <- models %>%
  group_by(mdl_nam) %>%
  unnest(pred, test_y)
## Map over all quantiles to get error metrics
quantile_errors <- CV_preds_long %>%
  nest(-mdl_nam) %>%
  mutate(q      = list(seq(0,1,0.01)),
         pred   = map(data, "pred"),
         test_y = map(data, "test_y")) %>%
  dplyr::select(-data) %>%
  unnest(q, .preserve = c(pred, test_y)) %>%
  filter(q != 0) %>%
  mutate(q_dat  = pmap(list(pred, test_y, q), quantile_error),
         q_pred = map(q_dat, "pred"),
         q_obs  = map(q_dat, "obs"),
         q_RMSE = map2_dbl(q_pred, q_obs, rmse),
         q_MAE  = map2_dbl(q_pred, q_obs, mae),
         q_logdev  = map2_dbl(q_pred, q_obs, logdev_p),
```

```
          y_max  = quantile(seq(0,max(dat$cps_net)), q),
          q_cnt  = nrow(og_dat) - map_int(q_dat, nrow))


# Map over all predictions grouped by model to calculate mean and sd for error metrics
model_results <- models %>%
  dplyr::select("Model Name" = mdl_nam, R2, RMSE, MAE, logdev) %>%
  group_by(`Model Name`) %>%
  arrange(`Model Name`) %>%
  summarise(R2_mean      = mean(R2, na.rm=TRUE),
            R2_sd        = sd(R2, na.rm=TRUE),
            MAE_mean     = mean(MAE, na.rm=TRUE),
            MAE_sd       = sd(MAE, na.rm=TRUE),
            RMSE_mean    = mean(RMSE, na.rm=TRUE),
            RMSE_sd      = sd(RMSE, na.rm=TRUE),
            logdev_mean  = mean(logdev, na.rm=TRUE),
            logdev_sd    = sd(logdev, na.rm=TRUE))
Model_Error_Results_table <- model_results %>%
  kable(., format = "html", digits = 3) %>%
  kable_styling()
##
meta_log_mean <- model_results[which(model_results$`Model Name` == "Meta-Model"),"logdev_mean",drop=TRUE]
meta_log_sd <- model_results[which(model_results$`Model Name` == "Meta-Model"),"logdev_sd",drop=TRUE]
meta_log_error <- qnorm(0.975)*meta_log_sd/sqrt(nrow(ens_cv_tbl))
meta_log_error_lower <- round(meta_log_mean - meta_log_error,3)
meta_log_error_upper <- round(meta_log_mean + meta_log_error,3)


meta_MAE_mean <- model_results[which(model_results$`Model Name` == "Meta-Model"),"MAE_mean",drop=TRUE]
meta_MAE_sd <- model_results[which(model_results$`Model Name` == "Meta-Model"),"MAE_sd",drop=TRUE]
meta_MAE_error <- qnorm(0.975)*meta_MAE_sd/sqrt(nrow(ens_cv_tbl))
meta_MAE_error_lower <- round(meta_MAE_mean - meta_MAE_error,3)
meta_MAE_error_upper <- round(meta_MAE_mean + meta_MAE_error,3)



## chunk aggregate_model_errors_to_neighborhood
```



```
error_geoplot <-  net_littlerock %>%
  left_join(., ens_pred_dat, by = c("net_id" = "test_net_id"),
            feature_name = paste0("Meta-Model", "dev")) %>%
  score_model() %>%
  mutate(dev_p_inv = 1 - logdev) %>%
  make_cuts(., "logdev", cuts = "breaks", n_breaks = 5)

# error metrics to points
error_points <- st_centroid(error_geoplot) %>%
  dplyr::select(logdev, MAE, test_y) ##add net_id

# aggreate mean errors to neighborhoods
neighborhood_metric_logdev <- error_points %>%
  aggregate(., lr_tract, mean) %>%
  dplyr::select(logdev) %>%
  make_cuts(., "logdev")  ## lr_tract replaces nbr

neighborhood_metric_MAE<- error_points %>%
  aggregate(., lr_tract, mean) %>%
  dplyr::select(MAE) %>%
  mutate(MAE = round(MAE,2)) %>%
  make_cuts(., "MAE")

## model error by decile

models <- bind_rows(rf_cv_tbl, spat_durbin_tbl, ens_cv_tbl, po_cv_tbl)

CV_preds_long <- models %>%
  group_by(mdl_nam) %>%
  unnest(pred, test_y)

## map over all quantiles to get error metrics
quantile_errors <- CV_preds_long %>%
  nest(-mdl_nam) %>%
  mutate(q      = list(seq(0,1,0.01)),
         pred   = map(data, "pred"),
```



```
         test_y = map(data, "test_y")) %>%
  dplyr::select(-data) %>%
  unnest(q, .preserve = c(pred, test_y)) %>%
  filter(q != 0) %>%
  mutate(q_dat  = pmap(list(pred, test_y, q), quantile_error),
         q_pred = map(q_dat, "pred"),
         q_obs  = map(q_dat, "obs"),
         q_RMSE = map2_dbl(q_pred, q_obs, rmse),
         q_MAE  = map2_dbl(q_pred, q_obs, mae),
         q_logdev  = map2_dbl(q_pred, q_obs, logdev_p),
         y_max  = quantile(seq(0,max(dat$cps_net)), q),
         q_cnt  = nrow(og_dat) - map_int(q_dat, nrow))

q_error_plotdat <- quantile_errors %>%
  dplyr::select(mdl_nam, q, q_RMSE, q_MAE, q_logdev)
q_cnt_plotdat <- quantile_errors %>%
  dplyr::select(mdl_nam, q, y_max, q_cnt) %>%
  filter(q != 0) %>%
  mutate(q_pcnt = (q_cnt / nrow(og_dat)))
q_error_mean <- q_error_plotdat %>%
  group_by(mdl_nam) %>%
  summarise(mean_RMSE = mean(q_RMSE, na.rm = TRUE),
            mean_MAE  = mean(q_MAE, na.rm = TRUE),
            mean_logdev  = mean(q_logdev, na.rm = TRUE)) %>%
  arrange(desc(mean_logdev))
print(q_error_mean)

### Error decile plots

LOGDEV_MODEL_ERROR_BY_DECILE_plot <- ggplot(data = q_error_plotdat, aes(x=q, y=q_logde
v, group = mdl_nam, color = factor(mdl_nam))) +
  geom_line(size = 1) +
  scale_color_viridis_d(name = "Model") +
  scale_y_continuous(limits=c(0,1)) +
  labs(y = "Logarithmic Score",
       caption = "Figure 6.2 - Goodness of fit by decile") +
  plotTheme() +
```



```
  theme(legend.position = "right")


MAE_MODEL_ERROR_BY_DECILE_plot <- ggplot(data = q_error_plotdat, aes(x=q, y=q_MAE, gro
up = mdl_nam, color = factor(mdl_nam))) +
  geom_line(size = 1) +
  scale_color_viridis_d(name = "Model") +
  labs(y = "MAE") +
  plotTheme()  +
  theme(legend.position = "right")


COUNT_BY_DECILE_plot <- ggplot(data = q_cnt_plotdat,
                               aes(x=q, y=q_cnt, group = mdl_nam, color = factor(mdl_n
am))) +
  geom_line(size = 1) +
  scale_x_continuous(breaks=seq(0,1,0.1), labels = seq(0,1,0.1)) +
  scale_color_viridis_d(name = "Model") +
  labs(y = "Number of Predictions in Each Decile",
       x = "Decile") +
  plotTheme()  +
  theme(legend.position = "right")



legend <- get_legend(LOGDEV_MODEL_ERROR_BY_DECILE_plot + plotTheme() + theme(legend.po
sition = "right"))


### Model_Error_Results_table


model_results <- models %>%
  dplyr::select("Model Name" = mdl_nam, R2, RMSE, MAE, logdev) %>%
  group_by(`Model Name`) %>%
  arrange(`Model Name`) %>%
  summarise(R2_mean      = mean(R2, na.rm=TRUE),
            R2_sd        = sd(R2, na.rm=TRUE),
            MAE_mean     = mean(MAE, na.rm=TRUE),
            MAE_sd       = sd(MAE, na.rm=TRUE),
            RMSE_mean    = mean(RMSE, na.rm=TRUE),
            RMSE_sd      = sd(RMSE, na.rm=TRUE),
            logdev_mean  = mean(logdev, na.rm=TRUE),
```

```
            logdev_sd    = sd(logdev, na.rm=TRUE))

Model_Error_Results_table <- model_results %>%

  kable(., format = "html", digits = 3) %>%

  kable_styling()


meta_log_mean <- model_results[which(model_results$`Model Name` == "Meta-Model"),"logd
ev_mean",drop=TRUE]

meta_log_sd <- model_results[which(model_results$`Model Name` == "Meta-Model"),"logdev
_sd",drop=TRUE]

meta_log_error <- qnorm(0.975)*meta_log_sd/sqrt(nrow(ens_cv_tbl))

meta_log_error_lower <- round(meta_log_mean - meta_log_error,3)

meta_log_error_upper <- round(meta_log_mean + meta_log_error,3)


meta_MAE_mean <- model_results[which(model_results$`Model Name` == "Meta-Model"),"MAE_
mean",drop=TRUE]

meta_MAE_sd <- model_results[which(model_results$`Model Name` == "Meta-Model"),"MAE_sd
",drop=TRUE]

meta_MAE_error <- qnorm(0.975)*meta_MAE_sd/sqrt(nrow(ens_cv_tbl))

meta_MAE_error_lower <- round(meta_MAE_mean - meta_MAE_error,3)

meta_MAE_error_upper <- round(meta_MAE_mean + meta_MAE_error,3)


## Line 1944 : change nbr to lr_tract

## Chunk aggregate_model_errors_to_neighborhood


error_geoplot <-  net_littlerock %>%

  left_join(., ens_pred_dat, by = c("net_id" = "test_net_id"),

            feature_name = paste0("Meta-Model", "dev")) %>%

  score_model() %>%

  mutate(dev_p_inv = 1 - logdev) %>%

  make_cuts(., "logdev", cuts = "breaks", n_breaks = 5)


# error metrics to points

error_points <- st_centroid(error_geoplot) %>%

  dplyr::select(logdev, MAE, test_y)


# aggreate mean errors to neighborhoods

neighborhood_metric_logdev <- error_points %>%

  aggregate(., lr_tract, mean) %>%
```



```
  dplyr::select(logdev) %>%
  make_cuts(., "logdev")


neighborhood_metric_MAE<- error_points %>%
  aggregate(., lr_tract, mean) %>%
  dplyr::select(MAE) %>%
  mutate(MAE = round(MAE,2)) %>%
  make_cuts(., "MAE")


##MODEL_ERROR_BY_NEIGHBORHOOD_plots


LOGDEV_BY_NEIGHBORHOOD_plot <- make_fishnet_dist_plot(neighborhood_metric_logdev, cps_
base_map, legend = "right",
                                                      direction = 1, var_name = "Devia
nce",
                                                      title = "Out-of-Fold error by Ce
nsus Tract") +
  labs(caption = "Figure 6.3",
       subtitle = "Logarithmic score") +
  mapTheme()


MAE_BY_NEIGHBORHOOD_plot <- make_fishnet_dist_plot(neighborhood_metric_MAE, cps_base_m
ap, legend = "right",
                                                   direction = 1, var_name = "MAE") +
  labs(subtitle = "MAE") +
  mapTheme()


MAE_BY_NEIGHBORHOOD_plot


plot_fold_pred(ens_cv_tbl$pred, ens_cv_tbl$test_y, type = "fit") +
  labs(x = "Observed Maltreatment Counts",
       y = "Predicted Maltreatment Counts",
       title = "Predicted vs. observed maltreatment counts",
       caption = "Figure 6.1") +
  plotTheme() +
  theme(panel.border = element_blank())


plot_fold_pred(ens_cv_tbl$pred, ens_cv_tbl$test_y, type = "fit") +
```



```
  labs(x = "Observed Maltreatment Counts",
       y = "Predicted Maltreatment Counts",
       title = "Predicted vs. observed maltreatment counts",
       caption = "Figure 6.1") +
  plotTheme() +
  theme(panel.border = element_blank())
## Below we calculate poverty and nonWhite rates by neighborhood by converting tracts
to centroids and spatial joining with
#neighborhoods statistical areas.

lr_tract_statAreas <- lr_tract %>% mutate(stat_area_id = GEOID)
tract10 <- var_list[["LR_BG_Tracts_ACS_DataJoined"]]

tract10 <- tract10 %>% dplyr::select(TotPopSize, Perc_NonWh, Perc_PopSt,
                                     Perc_NotIn, Perc_PopUn,geometry)
tract10 <- tract10 %>% dplyr::rename( Perc_PopUnder18inPov = Perc_PopUn,
            Perc_PopStrugg = Perc_PopSt,
            Perc_NotInsured = Perc_NotIn,
            TotalPop = TotPopSize) %>%
            dplyr::mutate(tract_id = dplyr::row_number(),
                          NumberWhites = ifelse(TotalPop > 0, (TotalPop*(1-Perc_NonWh)
),0),
                          TotalPoverty = TotalPop*Perc_PopStrugg)



tract10$tract_area <- st_area(tract10)
## Line 2015 ## chunk: census_statistical_area_spatial_intersection

#do the spatial join, create poverty and non whites rates by district. Create a dummy
for rates >= stat_area_quantile percentile
# create intersection of tract10 and statareas
lr_tract_statAreas.intersect <- st_intersection(tract10, lr_tract_statAreas)

# get % tract in statares and mulitply by pop totals from each tract
# result is the total tract pops distributed to the statarea by % of tract in statare
lr_tract_statAreas.spJoin <- lr_tract_statAreas.intersect %>%
  mutate(intersect_area = st_area(lr_tract_statAreas.intersect)) %>%
  # get % of tract and multiply totals by percent area of tract in statarea
```

```
  group_by(tract_id) %>%
  mutate(intersect_pcnt_of_tract = as.numeric(intersect_area) / as.numeric(tract_area)
,
         intersect_TotalPop = round(TotalPop * intersect_pcnt_of_tract, 1),
         intersect_NumberWhites = round(NumberWhites * intersect_pcnt_of_tract, 1),
         intersect_TotalPoverty = round(TotalPoverty * intersect_pcnt_of_tract, 1)) %>
%
  ungroup() %>%
  # sum the fraction of pop totals up to statarea
  group_by(stat_area_id) %>%
  summarise(statarea_TotalPop = sum(intersect_TotalPop),
            statarea_NumberWhites = sum(intersect_NumberWhites),
            statarea_TotalPoverty = sum(intersect_TotalPoverty)) %>%
  # make quantites of interest
  mutate(percentNonWhite = ifelse(statarea_TotalPop > 0,
                                  ((statarea_TotalPop - statarea_NumberWhites) / stata
rea_TotalPop),0),
         percentPoverty = ifelse(statarea_TotalPop > 0,
                                 statarea_TotalPoverty / statarea_TotalPop, 0))
# classify by quantile and make dummy variable
lr_tract_statAreas.spJoin <- lr_tract_statAreas.spJoin %>%
  mutate(poverty.percentile = ifelse(percentPoverty >=
                                       quantile(lr_tract_statAreas.spJoin$percentPover
ty,
                                                p = stat_area_quantile, na.rm=T),"1",0
),
         nonWhite.percentile = ifelse(percentNonWhite >=
                                        quantile(lr_tract_statAreas.spJoin$percentNonW
hite,
                                                 p = stat_area_quantile, na.rm=T),1,0)
)
```

References

1. Daley, Dyann, Michael Bachmann, Brittany A. Bachmann, Christian Pedigo, Minh-Thuy Bui, and Jamye Coffman. "Risk terrain modeling predicts child maltreatment." *Child abuse & neglect* 62 (2016): 29-38.
2. Durlauf, Steven N. "Neighborhood effects." In *Handbook of regional and urban economics*, vol. 4, pp. 2173-2242. Elsevier, 2004.
3. Daley, Dyann, Michael Bachmann, Brittany A. Bachmann, Christian Pedigo, Minh-Thuy Bui, and Jamye Coffman. "Risk terrain modeling predicts child maltreatment." *Child abuse & neglect* 62 (2016): 29-38.
4. Bivand, R. & Wong, D. W. S. (2018). Comparing implementations of global and local indicators of spatial association. *TEST* **27**, 716–748.
5. Bivand, R. S., Pebesma, E. J., Gomez-Rubio, V. & Pebesma, E. J. (2008). *Applied spatial data analysis with R*, vol. 747248717. Springer.
6. Breiman, L. (2001). Random forests. *Machine learning* **45**, 5–32.
7. Caplan, J. M., Kennedy, L. W., Barnum, J. D. & Piza, E. L. (2015). Risk terrain modeling for spatial risk assessment. *Cityscape* **17**, 7–16.
8. Caplan, J. M., Kennedy, L. W. & Miller, J. (2011). Risk terrain modeling: Brokering criminological theory and gis methods for crime forecasting. *Justice Quarterly* **28**, 360–381.
9. Drawve, G. (2016). A metric comparison of predictive hot spot techniques and RTM. *Justice Quarterly* **33**, 369–397.
10. Elhorst, J. (2014). *Spatial econometrics: from cross-sectional data to spatial panels*. Springer.
11. James, G., Witten, D., Hastie, T. & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Inc.
12. Montgomery, D. C., Peck, E. A. & Vining, G. G. (2006). *Introduction to Linear Regression Analysis* (4th ed.). Wiley & Sons.
13. Pebesma, E. J. & Bivand, R. (2019). *Spatial Data Science*.
14. Breiman, L. (1996). Stacked regressions. *Machine learning*, *24*(1), 49-64.
15. Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, *5*(2), 241-259.
16. Armstrong, J. S., & Collopy, F. (1993). Error measures for generalizing about forecasting methods: Empirical comparisons: International Journal of Forecasting, 8 (1), 69–80 (June 1992). *Long Range Planning*, *26*(1), 150.
17. Dustin, D., Liscio, E., & Eng, P. (2016). Accuracy and repeatability of the laser scanner and total station for crime and accident scene documentation. *J Assoc Crime Scene Reconstr*, *20*, 57-67.
18. McCullagh, P., & Nelder, J. (1989). Generalized Linear Models Second edition Chapman & Hall.
19. Gorr, W., Olligschlaeger, A., & Thompson, Y. (2003). Short-term forecasting of crime. *International Journal of Forecasting*, *19*(4), 579-594.

