BDA Project: Predicting Default Throughout Mexico

Andres Potapczynski (ap3635), Jongwoo Choi (jc4816), Yi Chen (yc3356)

12/10/2018

Abstract

We help a small mortgage lending start-up in Mexico understand how several demographical and economic variables predict default in different cities throughout the country. For this, we tested over 8 models. We find that the % of unemployment is the most relevant predictor. We also observe that the age and the credit risk behave as expected, where older and higher ranked individuals are less likely to default. Additionally, we see that there is a large uncertainty on the effect of income and the mortgage amount. We believe this is driven by the combination of different groups in the data set. A higher mortgage puts more financial pressure on individuals but it is also indicative that they were deemed reliable (by some factors that we do not observe in our data set). Finally, we observe that poorer states have a higher tendency to default. Nonetheless, this detrimental effect was not observed in the states that are currently in a drug war (which was something that the start-up expected).

We summarize our model approaches in the following table. Here we compare each model in both their predictive accuracy at the City and State level but also whether they successfully pass the two sets of PPCs that we defined (please see section 3.2 for the full details)

Model	City RMSE (sd)	State RMSE (sd)	PPC set 1	PPC set 2
Binomial	1.87 (0.24)	4.69 (0.98)	Yes	No
Binomial Hierarchical	2.16 (0.61)	5.23 (1.92)	Yes	Yes
Binomial ICAR	1.92 (0.28)	4.55 (1.22)	Yes	Yes
Binomial GP	1.91	5.20 (0.98)	Yes	No
Zero-inflated Binomial	1.811 (0.34)	NA	No	No

Therefore, as of now our leading model is the Binomial ICAR. We leave for future work the improvement of the final two alternatives.

Note: All the code and STAN models can be found in the following link [link][https://github.com/ChrisChen0429/BDA_project]

The structure of the document is:

1. **Research Question**
    - Stating the objective & setting a preamble
    - Acquiring some domain knowledge
2. **Data**
    - Understanding the data
    - Exploratory data analysis
3. **Models**
    - Failure of Logistic Regression
    - Overview of all models: changing the approach, variables included and evaluation (PPCs and CV)
    - Baseline Regression - comparing different model alternatives
    - Binomial Hierarchical Regression - including some hierarchy
    - Binomial ICAR - leveraging from spatial information

4. **Models in progress**
    - Gaussian Process Regression - testing nonparametric alternatives
    - Zero-inflated Binomial ICAR - adding a mixture component
    - Individual Level Binomial ICAR - a second take on individual data
5. **Conclusion and Learnings**
6. **Appendix**
    - Parameter Recovery
    - STAN Code
7. **References**

1. Research Question

1.1 Stating the objective & setting a preamble

Succinctly, our research question is the following:

- **What covariates influence / help predict the default rate at each city in Mexico?**

The context of why our research question is relevant is the following. **We are helping a small mortgage lending start-up in Mexico understand how separate cities vary in their risk to default on their mortgage**. This start-up has recently entered the market and wants us to help it asses how it should expand geographically throughout the country. Put differently, they want us to help them prioritize the expansion to cities that have shown the highest compliance as well as to understand how different variables predict such compliance.

It is also worth mentioning that someone defaulting generates an impact directly to the company's bottom line and, therefore, understanding why this happens (what variables influence this) and an accurate prediction of the number of defaults could potentially allow the company to flourish. **Put differently, inacurate predictions threaten the survival of this start-up**. Since the company is more concern about predicting the number of defaults than understanding why some individual might default, our models then focused on predicting the total number of defaults per city with city specific covariates (which are simply the average values of the covariates for the individuals residing in that city). As it will be shown below, this perspective reduced the noise for the individual level data and allowed us to successfully give an answer.

1.2 Acquiring some domain knowledge

In order to accomplish our objective, **we were provided with a large data set of 30,499 mortgages with over 90 covariates** (for a comprehensive explanation of the data set see the next section). This data set was provided by another player in the market which might acquire the start-up. We wanted to get some domain knowledge so **we interviewed the startup to understand their main hypothesis** of why different cities might have a diverging default behavior. Summarizing their points:

- **Some northern states in Mexico are fighting a drug war against different cartels and some cities have been specially damaged**. Thus, even though the northern region of Mexico is one of the wealthiest, cities belonging to the states of Sinaloa, Coahuila and Nuevo Leon might present higher default rates that cannot be explained with the covariates that we were given (since non are related to this event).
- **Poor southern states have a cultural tendency to fall easier into default**. States like Oaxaca might present a higher default rate even among high income individuals. The rational is that, given the low financial penetration in those states, individuals are less concern about their credit score and thus are more prone to abandon their properties if they fall into financial distress or even when they dislike their property.
- **The covariates related to employment should be the most predictive variables**. The start-up believes that the main driver of default is that people lose their jobs. Thus, the employment variables of our data set should be the most relevant.

2. Data

2.1 Understanding the data

In this section we examine three main elements of the data set. First, we expose the set of covariates that we were given. Next, we show the most important plots of exploratory data analysis. Finally, we detail the preprocessing steps that we took before pushing the data into `STAN`.

In terms of the covariates, as mentioned in the introduction, the data set consists of **30,499 mortgages with over 90 covariates** (where some of features are simply administrative and thus were not included in the analysis). **The average default rate is 6%**. Also, the data set provided was **their latest report available as of August 2018**. We group the covariates in the following categories:

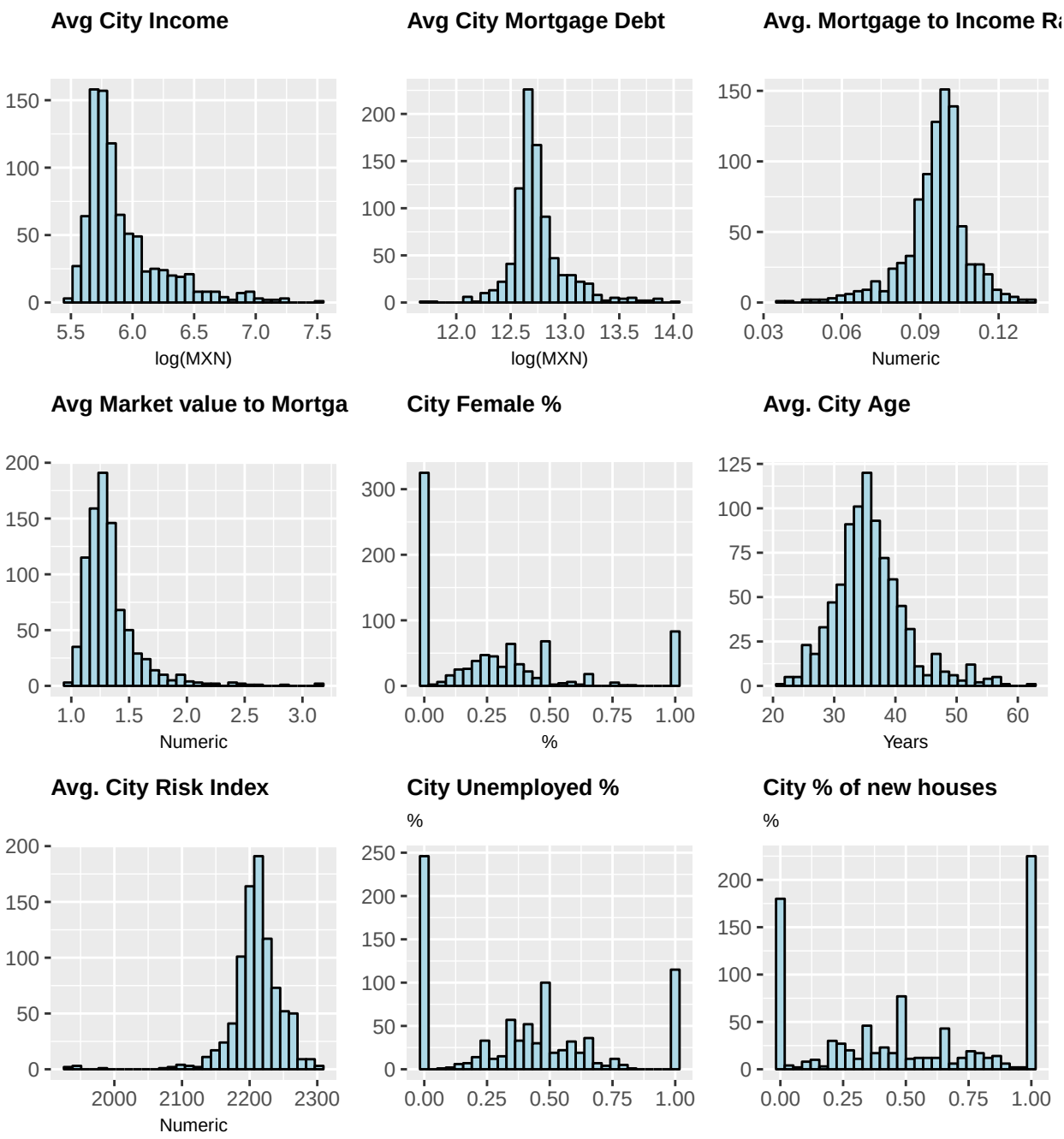
- [**Geographic**] `state`, `city` and `zip`: These features will allow us to understand if the client's behavior varies by geography. Also, we have this location features for both the house acquired and for the owner's location.
- [**Demographics**] `age`, `sex`, `ratio`, `risk_index`, `client_income` and `credit_score`. The feature `ratio` is the % of the client's income that the monthly payments for its mortgage represent. The `risk_index` is a variable that combines several metrics associated with her likelihood of paying a debt. The `client_income` feature is the monthly income that the person earned at that moment in time when she signed the mortgage. Finally, the ordinal variable `credit_score` evaluates how the client has performed in previous debts (related to `risk_index`).
- [**Asset features**] `vendor name`, `new_used`, and `appraisal_value`: These features tell us who was the vendor (either a construction company or an individual) and if the asset is new or not as well as the amount of money the person was given as a mortgage.
- [**Employment**] `employer_name` and `factor_employed`. The first feature conveys who is the employer and the second one informs about the client's status of employment.
- [**Payment Records**] `days_pay`, and `y`. The first feature counts the number of days that have passed between the last payment date and the date on which the mortgage started. The next feature is the target variable that is 1 if the mortgage has at least one month without a payment and 0 otherwise.

Note that features like `interest_rate` and `contract_length` are present but have no variance. The current *product offer* in the data base is a 12% interest rate mortgage for 30 years. Thus it is impossible to pose counter factual questions for those variables.

2.2 Exploratory Data Analysis

Our main tool for this analysis was to plot the relevant covariates. We also performed a clustering analysis using k-means but did not find the cluster to be much different (at the individual level). This was one of the indicators that told us that predicting default at the individual level (rather than city) was going to be hard.

The plots of the subset variables that we included in the analysis are:



Covariates	Data Type	Range	Mean	Standard Deviation	Kurtosis	Skewness
client_income	numeric	143.9 ~ 1887.2	409.8	300.8	11.8	3
appraisal_value	numeric	79521 ~ 1654602	371064	189052.8	14.5	3.2
asset_market_value	numeric	120000 ~ 451900	491311	301451.4	26.8	4
sex	character (binary)	“Male” or “Female”	“F”:0.3%			
age	numeric	18 ~ 65	34.3	9	2.8	0.8
risk_index	numeric	0 ~ 2370	2203	76.5	460.2	-16.6
factor_employed	numeric	0.52 ~ 196.2	2.08	7.9	259.8	15.1

2.2 Data Preprocessing

Below we list the main preprocessing steps that we did with the data

- **Aggregated the variables at the city level**. The data set that we were given was at the individual mortgage level. In order to make it suitable for the geographical analysis we aggregated the relevant variables at the city level. The majority by the mean but some by their sum. For example, individual income and age were transformed into mean income and mean age in that city. And the binary response when people default was summed into the count of people that defaulted in that city.
- **Transformed to log space the variables related to money**. As it is well-known, income distributions tend to be skewed and to resemble extreme value densities. Thus, we transformed those values into the log space to bring closer together the large values that were present.
- **Z-scored all the variables**. Finally we, z-scored all the variables to put them in the same scale and thus assuring that placing a similar prior in their slope coefficients (β_j) made sense.

3. Models

3.1 Failure of Logistic Regression

What appears most natural is to model this data with a logistic regression since we are trying to estimate a binary variable with information from several variables. However, as it was discussed previously, we find that the data is not informative at this level (there is a lot of noise). Even with nonparametric methods it is not possible to uncover a clear relationship. Remember that in this context, the data is highly unbalanced, thus predicting always that the person is not going to default ($Y = 0$) already achieves a 93% test accuracy. The results from the nonparametric methods that we ran were:

Classifier	5 fold Test Accuracy
KNN	94 %
Decision Tree	93 %
Always Predict Zero	93 %

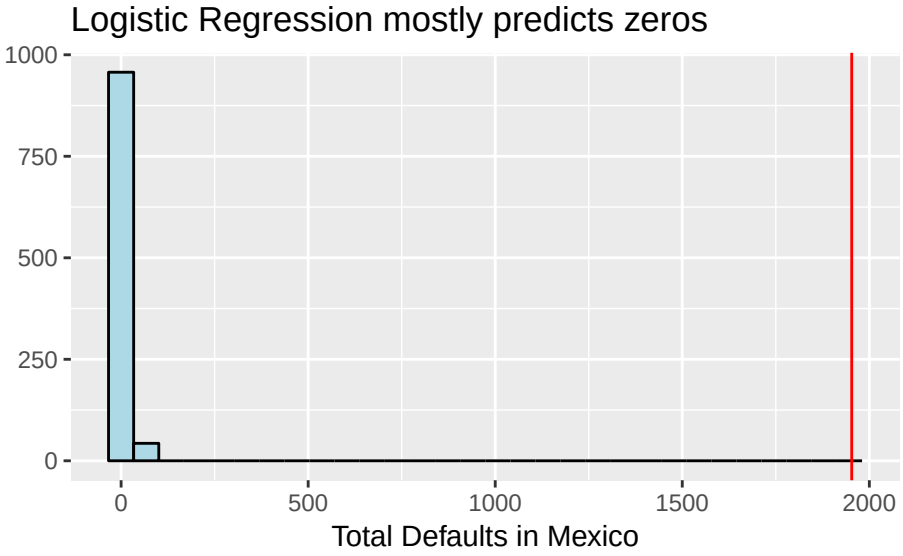
The main message is that even complex / flexible models are not able to extract signal from all the noise. We hypothesize that the data is composed of several cohorts of different individuals that each has a different response for each variables and, thus, when aggregated together the signal gets lost. At the end we explored some ideas like the zero-inflated model which gave us some progress but we did not get it to converge satisfactorily.

To display that logistic regression is not useful we estimate this model and show a PPC where it fails short. The generative process is:

$$\alpha \sim N(0, 5)$$
$$\beta \sim N(0, 5)$$
$$y_i \sim Ber(logit^{-1}(\alpha + X_i\beta))$$

the variables that are going to be included for the X are the following: `client_income`: β_1, `appraisal_value`: β_2, `app_2_inc`: β_3, `mar_2_app`: β_4, `sex_F`: β_5, `age`: β_6, `risk_index`: β_7, `employed_30`: β_8 and `condition_U`: β_9.

The total number of people that default is a critical quantity to estimate for the start-up since there is a direct link to its profits. The logistic regression can get away with predicting zero all the time (this already gives a 93% accuracy) however it is worthless for predicting the total number of defaults as it is seen below.



3.2 Overview of all models: changing the approach, variables included and evaluation (PPCs and CV)

We changed our approach in the following manner. Rather than keep modelling the individual data, we aggregated it at the level on which the start-up was interested: city and state. The aggregation was mostly done by averaging the different variables for each individual. In this way, since we are now working with variables which are the combination of many small events (the individuals), the *CLT* then provides us with a more manageable data set since part of the noise gets pruned on the aggregation process.

With this new approach, we now focus on modelling a count variable: the number of defaults per city. **For each of the models that we tried below we do the following**:

- **Show the generative process of the data** (in order words, writing the model mathematically rather than in STAN code)
- **Show the parameters estimates using the actual data**
- **Evaluate with PPCs of important quantities and 5 fold CV**
- **Verify that it recovers the parameters with fake data**. We do not display the usual STAN print but rather use the `mcmc_recover_hist` function which displays the same information but in a visual manner (where we also input the true values). It is also worth mentioning that the simulate data is for the y variable. Thus the X covariates are held fixed. (We leave this for the appendix)

In terms of evaluation, we do the classical approaches of *Posterior Precitive Checks* (PPCs) and 5-fold *Cross-Validation* (CV) with an 80 / 20 train / held-out split. Doing PPCs for parameters that are already in the model (such as the mean or variance) is not as informative especially if the model is correctly replicating the data. Therefore we chose to do two sets of PPCs to test on all the models which mostly include variables that are not estimated directly by the model. The sets are the following

- **PPC set 1**
  - **Highest City Default Total**: Mathematically, it is just $max(y)$. It is important for the start-up to know what is, at most, the highest default count that they will get in any given city. Since all of our models below already have a mean and variance parameters this quantity is easier to get it right. Nonetheless, we left it as a safe guard to ensure that all our models do not fall short for this quantity.
  - **Total Defaults in Mexico**: Mathematically, this is now $\sum_i y_i$ where each i indexes a particular city. This is not a quantity modeled directly and similar to the previous discussion is a variable that tells the start-up which is going to be its expected overall loss.

  - **City Overlay**: This is simply the density of the y_{rep} over the observed y. This is a vital PPC since it summarizes how well the model is replicating the actual data (since we are directly modeling this). We see that all our models satisfactorily cover the observed value and that the blue lines are not so thick. Thus the uncertainty is not so high.
  - **State Overlay**: This is now a slightly more involved density overlay but it is similar in spirit to the previous one. It is more involved since we need to aggregate the y_{rep} at the state level and then create the densities. This is an important quantity that we are not modeling directly for our research question since we are understanding how default changes by geography. Also, because again the number of defaults per state is another variable of interest to the start-up.
- **PPC set 2**: The idea of this set of PPCs is to **ensure that we are estimating the total number of defaults by each of the 32 states (which we are not modelling directly)**. Again this is relevant for the start-up to ensure a proper geographical expansion. We see that our first model does a poor job here. However, the rest of our model extensions are able to correct for this.

Finally the variables that we included and their order that is shared across the following models are:

Coeff	Variable
α	Intercept
β_1	avg. city income
β_2	avg. city mortgage
β_3	avg. city mortgage to income ratio
β_4	avg. city asset market value to debt ratio
β_5	female % in city
β_6	avg. city age
β_7	avg. city risk index
β_8	% of unemployed
β_9	% of new houses

3.3 Baseline Regression - comparing different model alternatives

Since we have count data, we tried different models such as binomial, poisson and negative binomial. All of them yielded similar results: both for the values of α, β and for held-out RMSE in 5 fold CV. We opted to continue the analysis with the binomial specification since it was converging faster and with non numerical problems (in contrast with the poisson model when even when using the function `poisson_log` in `STAN` we had numerical problems for some chains).

3.3.1 Mathematical Model (Generative Process)

Note: The `STAN` code for this model can be found on the Appendix Section 1.

The generative process for our main model of this section is

$$\alpha \sim N(0, 5)$$
$$\beta \sim N(0, 5)$$
$$y_j \sim Bin(n_j, logit^{-1}(\alpha + X_j\beta))$$

where now $j = 1, \ldots, 880$ (total number of cities in the data set) and X_j is equal to the average of all the value of that variables for all the individuals in that city. The other model specifications that we tried only changed how y_j is distributed. For the poisson model we have

$$y_j \sim Poi(n_j \exp(\alpha + X_j\beta))$$

whereas for the negative binomial we have to add another positive parameter ϕ. We employed the mean-variance parameterization and finally,

$$y_j \sim NegBin(\exp(\alpha + X_j\beta), \phi)$$

It is worth mentioning that as part of a prior check, we also substitute the normal priors for

$$\alpha \sim Cauchy(0, 5)$$
$$\beta \sim Cauchy(0, 5)$$

3.3.2 Parameter Estimates

We now present the results for the binomial model (as a `STAN` print) but also in a comparative table the result for the rest of the models.

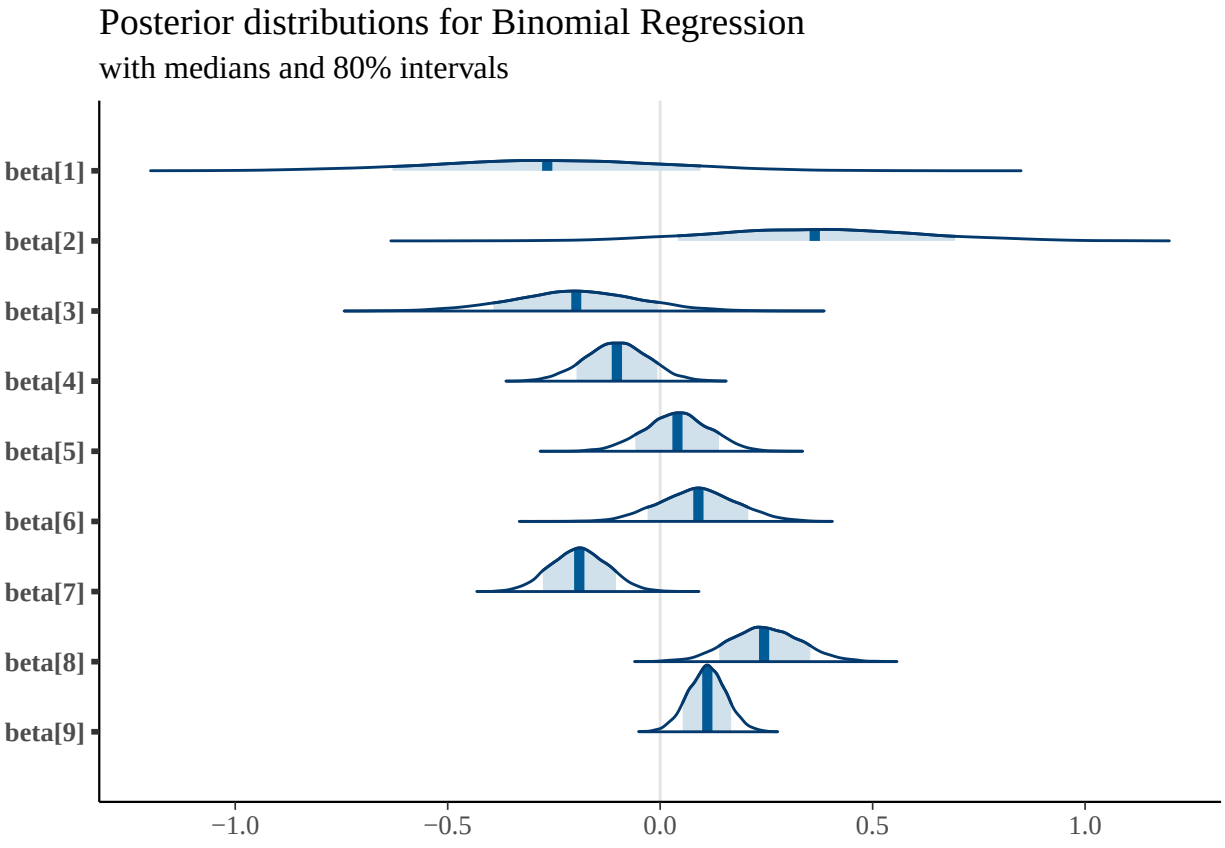
```
## Inference for Stan model: binomial.
## 8 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=8000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## alpha   -2.73    0.00 0.04 -2.82 -2.76 -2.73 -2.70 -2.65  5199    1
## beta[1] -0.27    0.01 0.28 -0.83 -0.45 -0.27 -0.08  0.27  3108    1
## beta[2]  0.36    0.00 0.25 -0.12  0.19  0.36  0.53  0.87  3354    1
## beta[3] -0.20    0.00 0.15 -0.50 -0.30 -0.20 -0.10  0.10  3434    1
## beta[4] -0.10    0.00 0.07 -0.25 -0.15 -0.10 -0.05  0.04  6036    1
## beta[5]  0.04    0.00 0.08 -0.11 -0.01  0.04  0.09  0.19  7279    1
## beta[6]  0.09    0.00 0.09 -0.09  0.03  0.09  0.15  0.27  5676    1
## beta[7] -0.19    0.00 0.07 -0.32 -0.24 -0.19 -0.14 -0.06  6747    1
## beta[8]  0.25    0.00 0.08  0.08  0.19  0.24  0.30  0.41  5838    1
```

```
## beta[9]  0.11    0.00 0.04  0.02  0.08  0.11  0.14  0.20  6767    1
##
## Samples were drawn using NUTS(diag_e) at Fri Dec  7 20:41:54 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Coeff	Binomial	Binomial Cauchy	Poisson	Negative Binomial
α	-2.73	-2.73	-2.8	-2.69
β_1	-0.27	-0.20	-0.23	-0.32
β_2	0.36	0.31	0.33	0.39
β_3	-0.2	-0.17	-0.17	-0.25
β_4	-0.10	-0.10	-0.09	-0.11
β_5	0.04	0.04	0.03	0.04
β_6	0.09	0.09	0.08	0.1
β_7	-0.19	-0.19	-0.17	-0.2
β_8	0.25	0.24	0.23	0.23
β_9	0.11	0.11	0.1	0.13

What we can conclude from the previous table is that our estimates are not perturbed by different model specifications. Moreover, the effect of adding a "robust" prior to the problem has little effect.

From the following plot we see that we have a lot of uncertainty in our estimates.



A few comment on some of the variable estimates:

- It is counter intuitive that the avg. income has the largest negative effect (β_1) however there is a lot of

uncertainty in the estimate (probably due to the fact that we are mixing together cities with different levels of income) so we cannot understand what it the true effect.

- In contrast, the average city mortgage has a positive effect as expected since higher debts puts more financial distress on the individuals. However, we hypothesize that the large variance and negative values might be due to confounding effects (where also individuals that are deemed worthy of a higher mortgage displayed an accountability metric that we do not observe in the data).
- The % of people unemployed in the city suggest an important variable for the start-up to consider. Also note how it is one of the estimates that has the lowest variance.

3.3.3 Evaluation

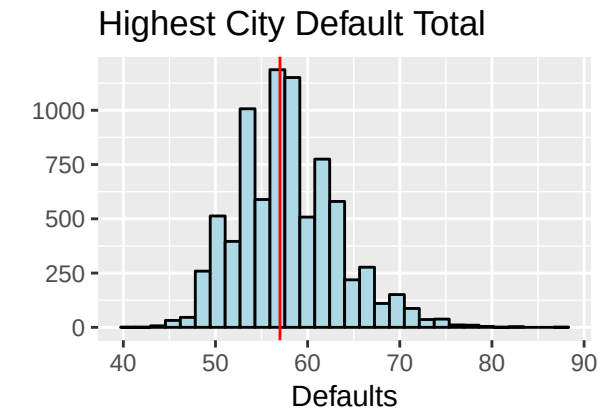
Before diving into the PPCs of the binomial model, we will present yet another comparative table between the models. Below is the RMSE for 5 fold CV with an 80 / 20 split each time

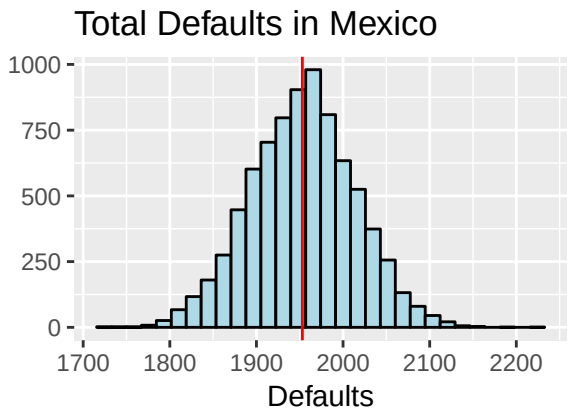
Model	5 fold RMSE (sd RMSE) at City
Predict always zero	7.174 (1.58)
Binomial	1.87 (0.239)
Binomial Cauchy	1.866 (0.24)
Poisson	1.87 (0.242)
Negative Binomial	1.8513 (0.2383)

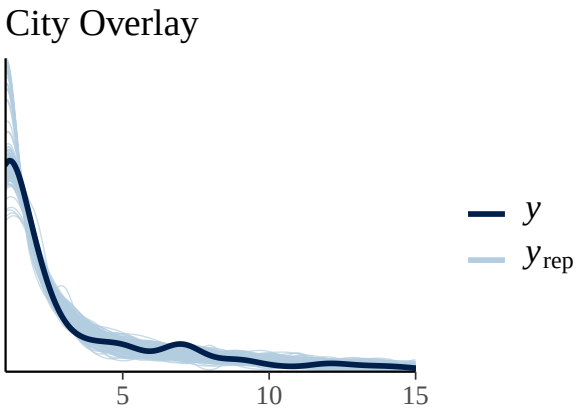
Model	5 fold RMSE (sd RMSE) at State
Predict always zero	23.63 (4.45)
Binomial	4.69 (0.981)
Binomial Cauchy	4.69 (0.977)
Poisson	4.69 (0.9814)
Negative Binomial	4.657 (0.9742)

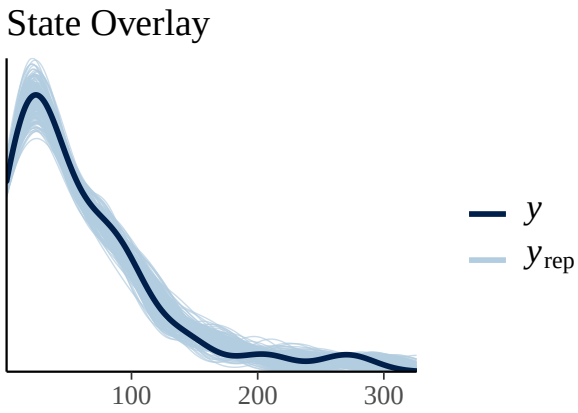
where we see that in CV the models are performing quite the same.

Again, we start by looking at the tail value of the sum of the total defaults predicted by the model. Now we check how well the model is able to replicate the per city counts.



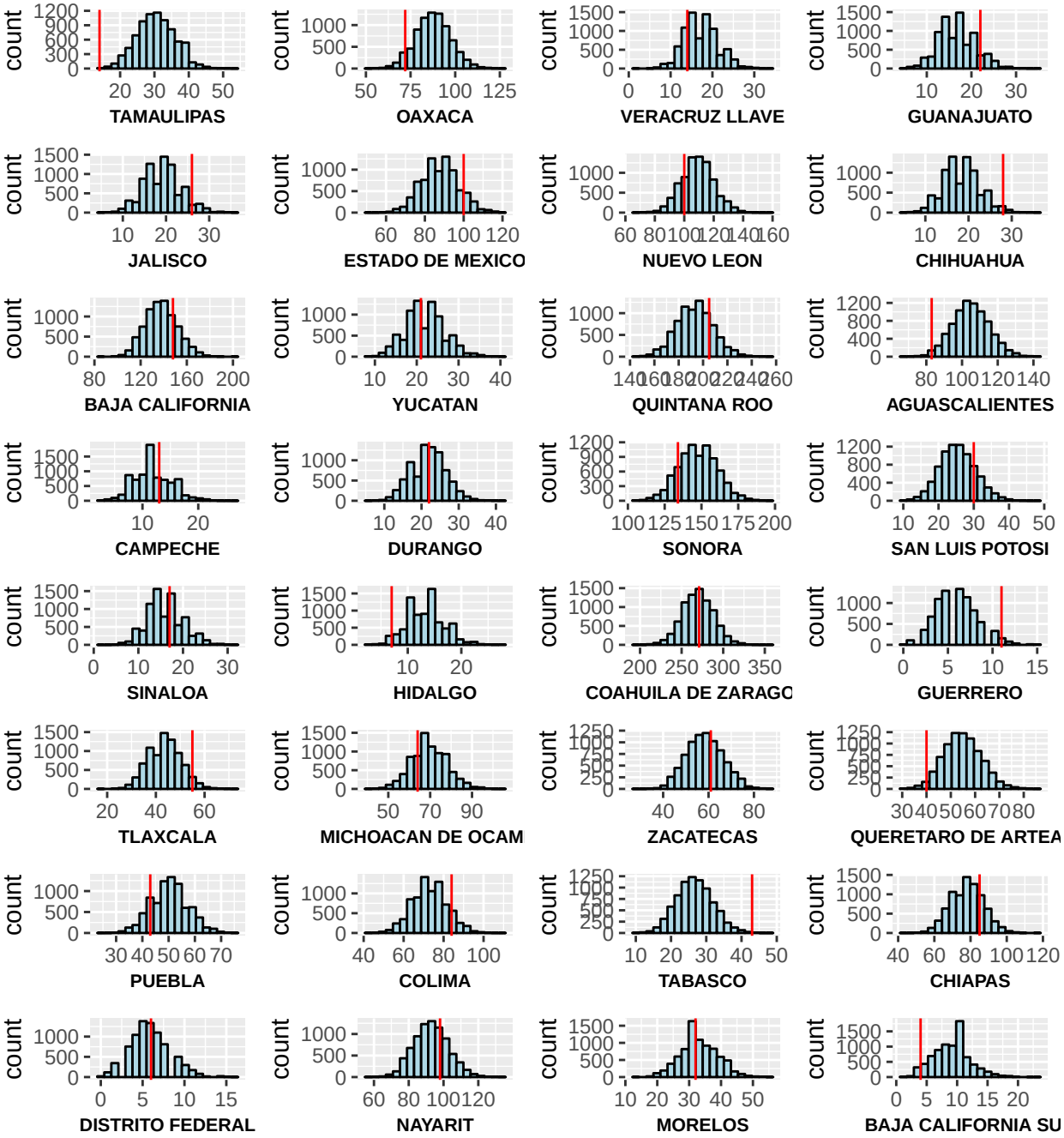






We observe that the model is able to satisfactorily mimic the distribution of the data. In addition, the model is also able to emulate the distribution of the data if we aggregate it at the state level.

Thus far our models are fitting the data appropriately. However, if we look at the per individual state graphs, we acknowledge some deficiency which leave some room for improvement.



3.4 Binomial Hierarchical Regression - Including some Hierarchy

The easiest way to fix a PPC is to model it directly. It appears as cheating. In the sense that we keep expanding the model to every eventuality that we encounter; nonetheless, if this eventuality is actually an aspect of the model that we do care then there is not evident harm on expanding the model to fit this aspect as well. This is what we do for the first model expansion. We introduce a hierarchical structure at the state level to each of the intercepts.

3.4.1 Mathematical Model (Generative Process)

The generative process now is the following

$$\alpha \sim N(0, 5)$$

then for each $s = 1, \ldots, 32$

$$\sigma_s \sim lognormal(\log(1), 1)$$

$$\alpha_s \sim N(\alpha, \sigma_j)$$

$$\beta \sim N(0, 5)$$

$$y_j \sim Bin(n_j, logit^{-1}(\alpha_{s_j} + X_j\beta))$$

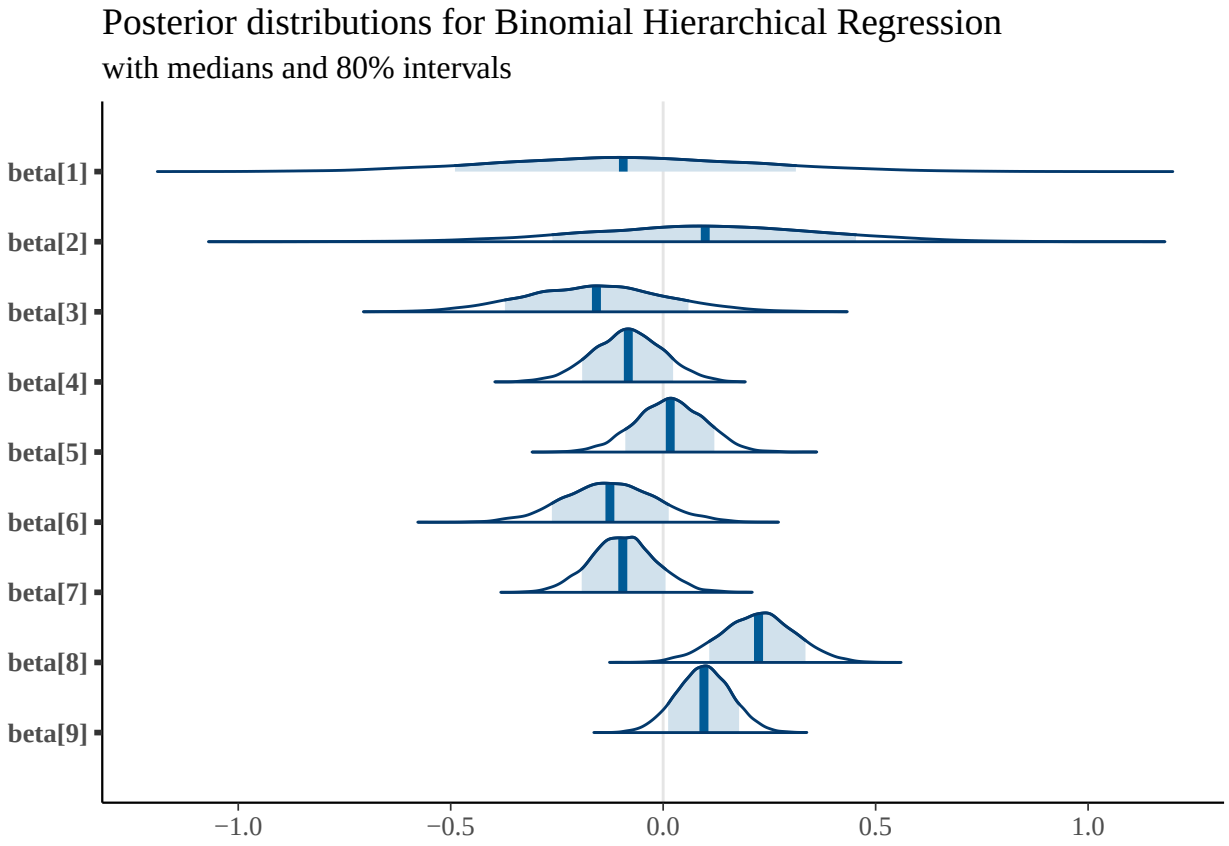
where again $j = 1, \ldots, 880$. Now we have an offset that should give us sufficient flexibility to estimate what we want to.

3.4.2 Parameter Estimates

The results of incorporating this hierarchical structure can be seen below.

```
## Inference for Stan model: hier_binomial.
## 8 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=8000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## alpha   -2.73    0.01 0.15 -3.03 -2.84 -2.73 -2.63 -2.43   755 1.01
## beta[1] -0.09    0.00 0.31 -0.68 -0.30 -0.09  0.12  0.52  4262 1.00
## beta[2]  0.10    0.00 0.28 -0.45 -0.09  0.10  0.28  0.63  4322 1.00
## beta[3] -0.16    0.00 0.17 -0.48 -0.27 -0.16 -0.05  0.18  4504 1.00
## beta[4] -0.08    0.00 0.08 -0.25 -0.14 -0.08 -0.03  0.08  6589 1.00
## beta[5]  0.02    0.00 0.08 -0.15 -0.04  0.02  0.07  0.17  8828 1.00
## beta[6] -0.12    0.00 0.11 -0.34 -0.20 -0.13 -0.05  0.09  5563 1.00
## beta[7] -0.09    0.00 0.08 -0.24 -0.15 -0.09 -0.04  0.06  7248 1.00
## beta[8]  0.22    0.00 0.09  0.05  0.16  0.22  0.28  0.39  6508 1.00
## beta[9]  0.10    0.00 0.06 -0.03  0.05  0.10  0.14  0.22  7791 1.00
##
## Samples were drawn using NUTS(diag_e) at Sun Dec  9 13:15:16 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Visualizing the previous table

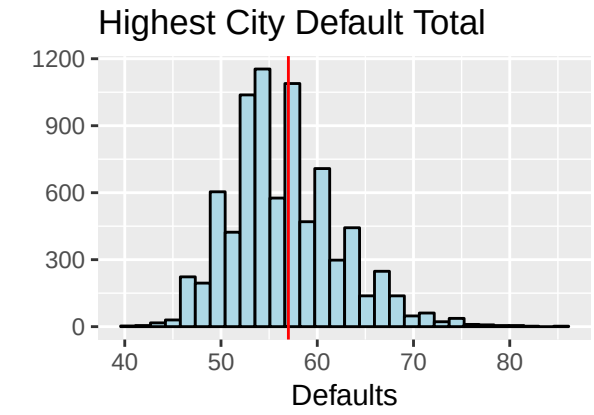


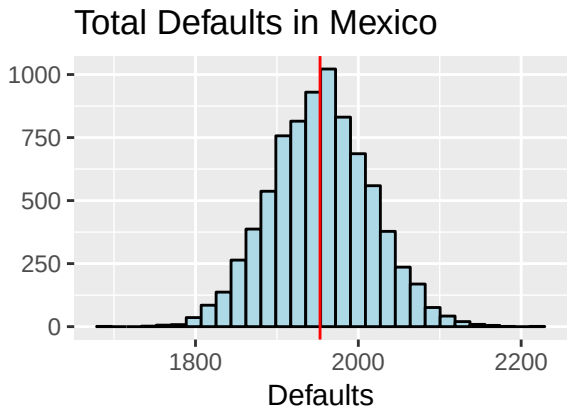
Some comments of the new estimates for this model:

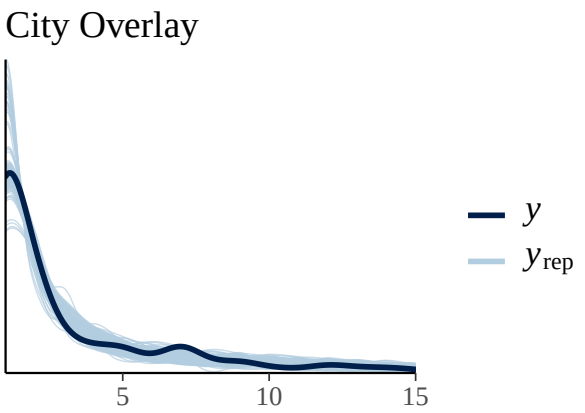
- Now the avg. city mortgage (β_2) has much more variance. Thus this variable is starting to become less important for the model.
- The % of females in a given city (β_5) is now pushed almost to zero. Thus suggesting that this variable is actually not important.
- The avg. city age (β_6) switches sides. Hence if we control for the cities fixed effects then the age now plays a different role. Where it makes sense that older people tend to be more responsible since they probably have a family to attend. Were possibly there was confounding.

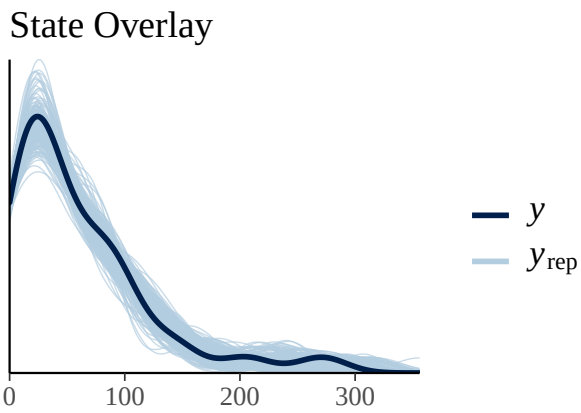
3.4.3 Evaluation

As always, we start with the PPC for the total number of defaults

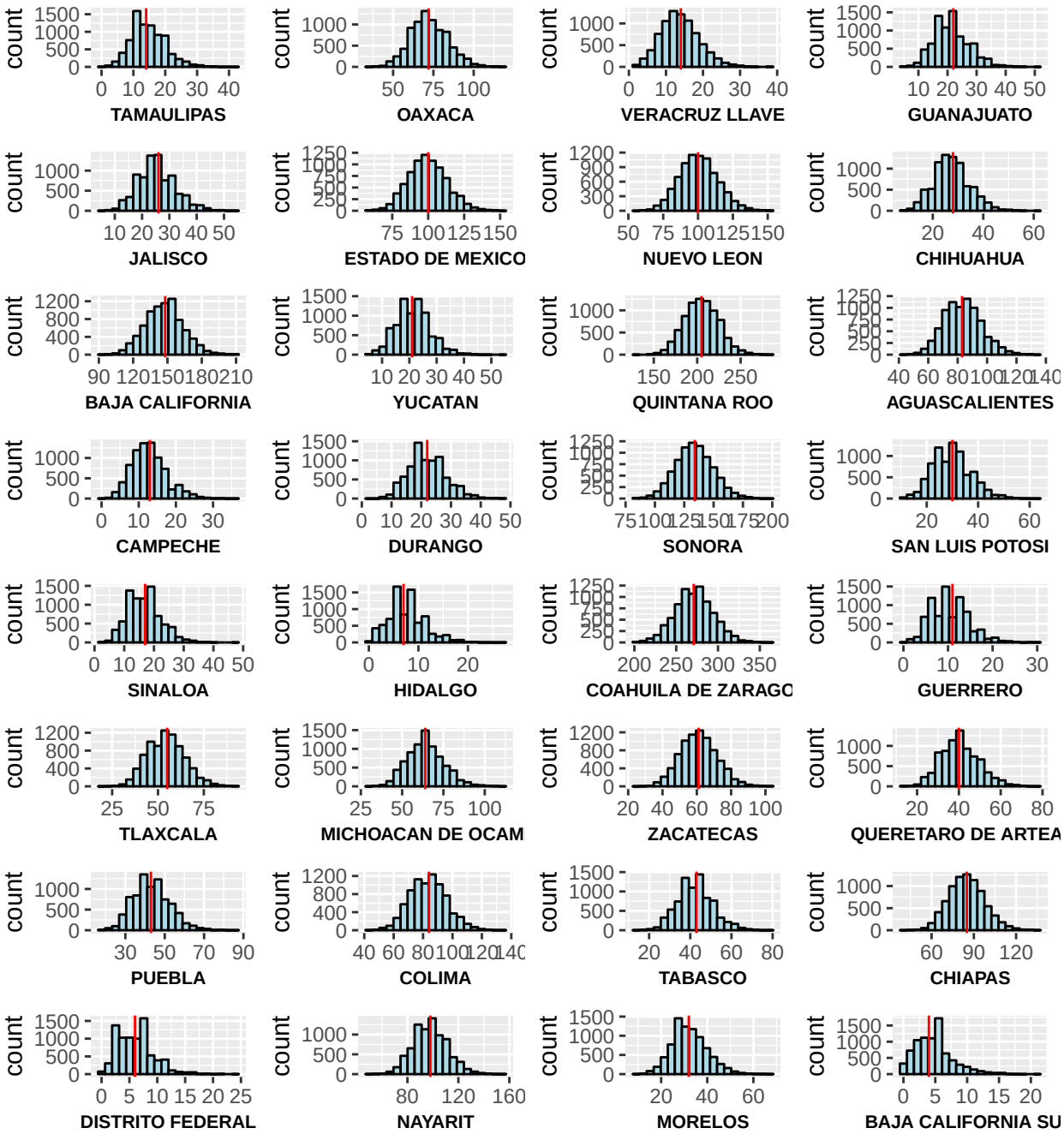








Now for the second set of PPCs, we see how our new model does much better in estimating the values at each state



Now **it is worth pointing out that in terms of 5 fold CV this extension worsens the performance. The 5 fold RMSE is: 2.1594 (0.61)**. Even though held out scores are a common currency to compare models, it is not necessarily true that we should always pick the one with the lowest error especially when the differences are not as substantial. Nonetheless, our next model fixes this loss of predictability and estimates the state default means as well as the model of this section.

3.5 Binomial ICAR - Leveraging from Spatial information

In this section we yet again expand the previous model. Now by incorporating spatial information. As seen before, the addition of a hierarchical structure into our model made the y^{rep} at the state level fit much better. Nonetheless, we suffered a decrease in held-out accuracy. Thus we explore if adding spatial information is able to alleviate that. The idea is to upgrade the hierarchical interaction of the α_s's where now they share information between neighboring states.

3.5.1 Mathematical Model (Generative Process)

One of the most popular ways to incorporate spatial random effects it to use Conditional Auto regressive (CAR) priors. Now instead of α_s we are going to switch to ϕ_s. Now the process is the following:

$$\phi_s \mid \phi_j \sim N(\alpha \sum_{j=1}^{n} b_{sj}\phi_j, \tau_s^{-1})$$

where each b_{sj} is an element that encodes the adjacency matrix information and τ_s is a spatially varying precision. By using Brook's lemma we can express simplify the previous expression as:

$$\phi \sim N(0, [D_\tau(I - \alpha B)]^{-1})$$

where

- $D = diag(m_s)$ is a 32×32 diagonal matrix where m_s is the number of the neighbors for the state s
- $D_\tau = \tau D$ and τ is the variance hyperparameter for the ϕs
- α is the parameter that controls spatial dependence.
- $B = D^{-1}W$ is the scaled adjacency matrix (discussed above). And W is the adjacency matrix. ($w_{ss} = 0$, $w_{ij} = 1$ if the state s is a neighbor of state j and zero otherwise)

However, to alleviate the computational burden, it is common to use the ICAR prior where, in the previous expression α is set to 1. The only problem with the ICAR prior is that it is improper.

The generative process is

$$\alpha \sim Cauchy(0, 10)$$
$$\beta \sim Cauchy(0, 2.5)$$

we picked those variables based on a recommendation from the STAN team

$$\tau \sim Gamma(2, 2)$$

then for each $s = 1, \ldots, 32$

$$\phi_s \sim N(0, [D_\tau(I - \alpha B)]^{-1})$$

and finally

$$y_j \sim Bin(n_j, logit^{-1}(\alpha + \phi_{j_s} + X_j\beta))$$

where again $j = 1, \ldots, 880$.

Before moving forward, it is important to make two comments:

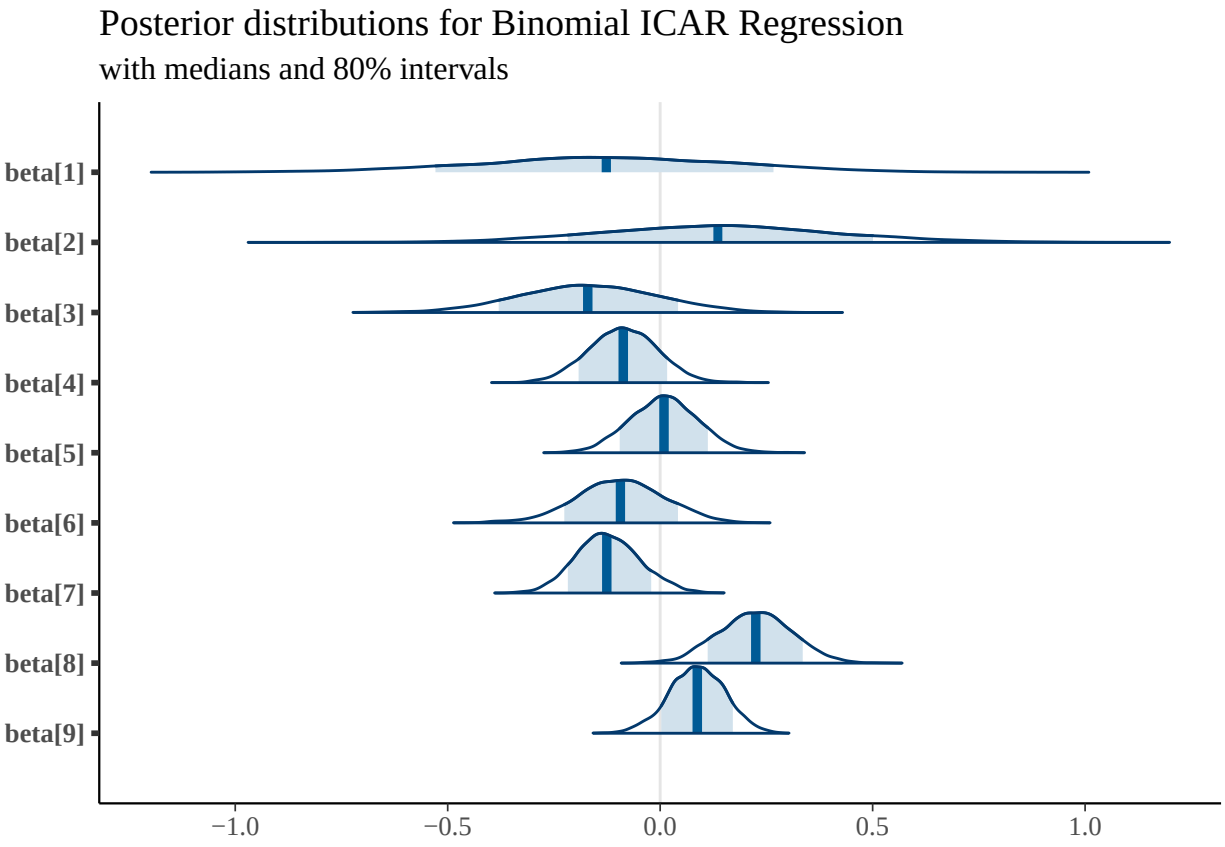
- This prior on the ϕ_s is more restrictive than the previous prior on α_s. Thus, we expect more regularization.
- This design of the model is better for interpretation and for the questions that the start-up wants to answer (since they have geographical concerns). With this model we can *learn* about geographical patterns.

3.5.2 Parameter Estimates

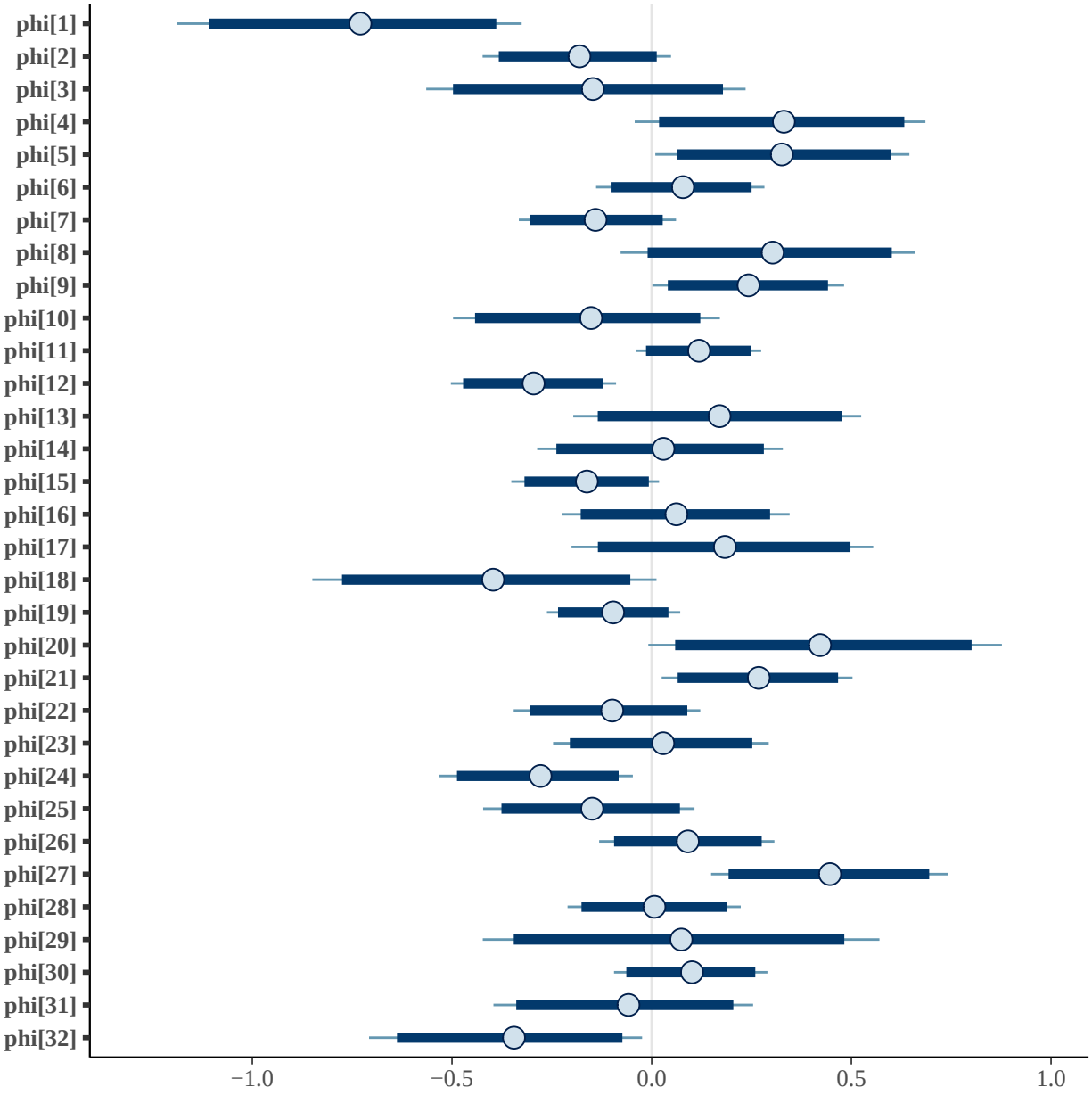
The `STAN` print for the parameters estimated is

```
## Inference for Stan model: car.
## 8 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=8000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## alpha   -2.73    0.00 0.05 -2.83 -2.76 -2.73 -2.70 -2.63   376 1.02
## beta[1] -0.13    0.02 0.31 -0.74 -0.32 -0.13  0.09  0.46   258 1.03
## beta[2]  0.14    0.02 0.28 -0.39 -0.05  0.14  0.32  0.70   268 1.03
## beta[3] -0.17    0.01 0.16 -0.49 -0.28 -0.17 -0.06  0.15   342 1.02
## beta[4] -0.09    0.00 0.08 -0.24 -0.14 -0.09 -0.03  0.07   600 1.01
## beta[5]  0.01    0.00 0.08 -0.15 -0.05  0.01  0.06  0.17   797 1.01
## beta[6] -0.09    0.01 0.10 -0.30 -0.16 -0.09 -0.02  0.11   361 1.01
## beta[7] -0.12    0.00 0.08 -0.26 -0.17 -0.13 -0.07  0.03   581 1.00
## beta[8]  0.22    0.00 0.09  0.06  0.17  0.23  0.28  0.40   525 1.02
## beta[9]  0.09    0.00 0.07 -0.05  0.04  0.09  0.13  0.21   554 1.03
##
## Samples were drawn using NUTS(diag_e) at Fri Dec  7 19:50:50 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Visualizing the previous information graphically we have



were we see that the parameter estimates are maintained almost unchanged from the previous model. Now, in terms of the intercepts estimated for each state we have the following plot

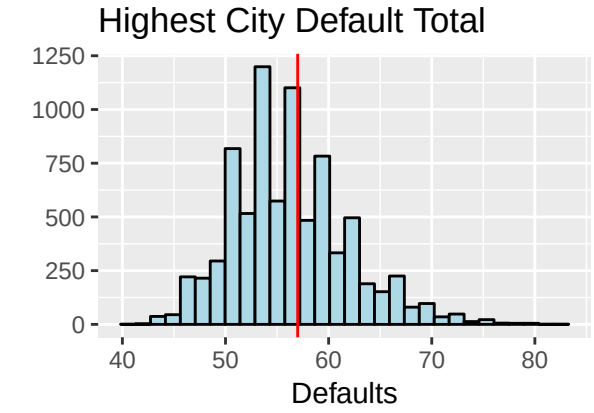


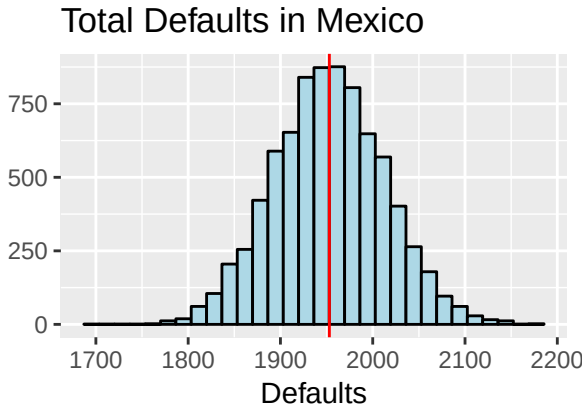
The states appear in alphabetical order. Some comments on the value of this coefficients:

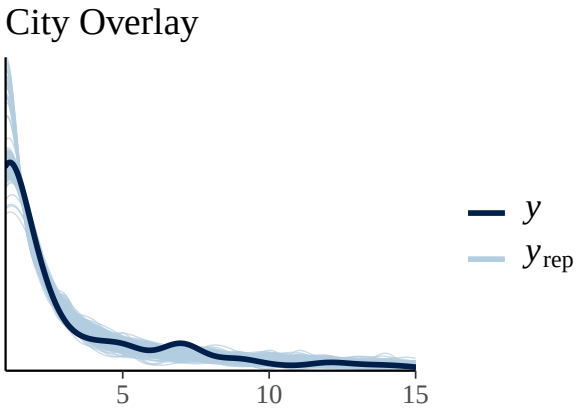
- **Poor states have positive a similar large effect**: Campeche (ϕ_4), Chiapas (ϕ_5), Colima (ϕ_8), Guerrero (ϕ_{13}), Oaxaca (ϕ_{20}) and Tabasco (ϕ_{27}).
- **States in Drug war do no have any effect**: Chihuahua (ϕ_6), Coahuila (ϕ_7), Michoacan (ϕ_{16}), Sinaloa (ϕ_{25}) and Tamaulipas (ϕ_{28}) none display an important fixed effect. **This is a relevant finding which was opposed to the initial hypothesis of the start-up**.

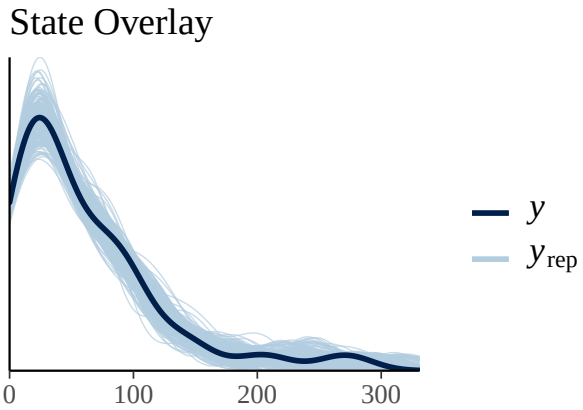
3.5.3 Evaluation

As always, we start with the PPC for the total number of defaults we see that this model does estimate well the important quantities evaluated by the PPC

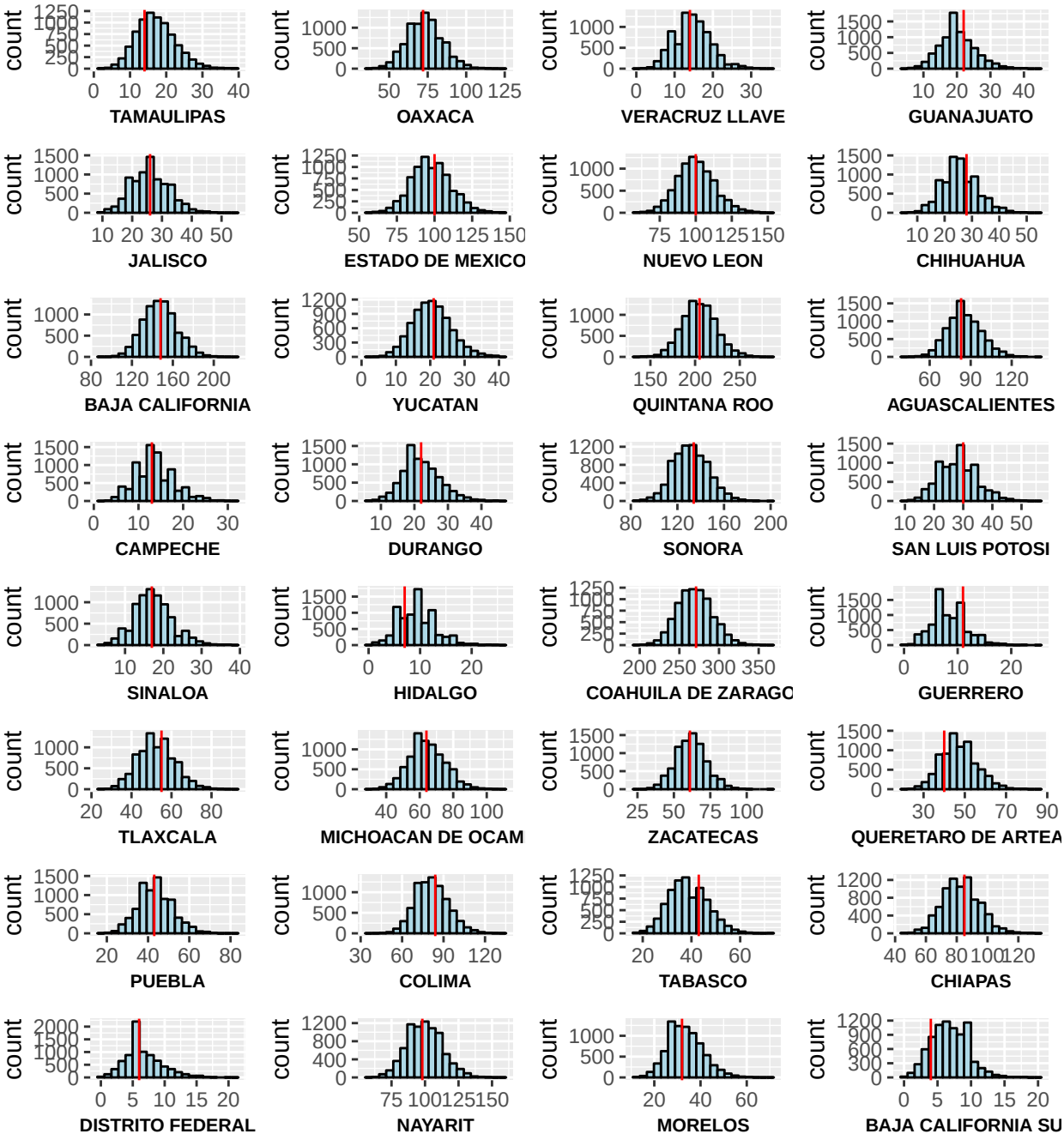








Moreover, it does sustain the estimation of the state level variables.



Also, we were glad to find that the RMSE for the CAR model is 1.9284 (0.28) which is slightly above of the other regression models but not by much. Moreover, it does reduce the standard held-out deviation to from 0.6 to 0.28.

4. Models in progress

This section of the write-up consists of the models that we are either still extending (like the GP) or that we are fixing its convergence (Zero-inflated Binomial ICAR)

The idea of this chapter is to show the ideas that we are still working on but not to dive fully into the details. So we avoid showing the parameter recovery for the simulated fake data as well as to show the parameter estimations.

4.1 Gaussian Process Regression - testing nonparametric alternatives

Finally, we incorporated a Gaussian Process into our analysis as an alternative route to understand if there were any interactions or nonlinearities in the data. Moreover, since this model fits well to the data, we leave as a future extension the addition of the ICAR prior to this model (although it would probably take days to fit).

4.1.1 Mathematical Model (Generative Process)

Nonparametrics do not follow the same generative interpretation from the previous discussions. Now, instead of a generative process over numbers, we have a probability distribution over functions. As the name suggests, the Gaussian process relies on the multivariate normal distribution where now instead of a mean and covariance matrix we have a mean *function* and a covariance *function*. Mathematically,

$$y \sim MVN(a + f(x), K(x|\theta))$$

where a is an intercept value we included, $f(x)$ is the realization of a function over the N inputs and the positive-definite matrix takes the common form of

$$K(x|\alpha, \rho, \sigma)_{i,j} = \alpha^2 \exp\left(-\frac{1}{2\rho^2}\sum_{d=1}^{D}(x_{i,d} - x_{j,d})^2\right) + \delta_{i,j}\sigma^2,$$

where α, ρ and σ are the hyperparametrs. In this context α is called the *marginal standard deviation* and it controls the magnitude of the range of the function modeled by the GP. Moreover, ρ is called the *length-scale* parameters and links to the smoothness of the function represented by the GP.

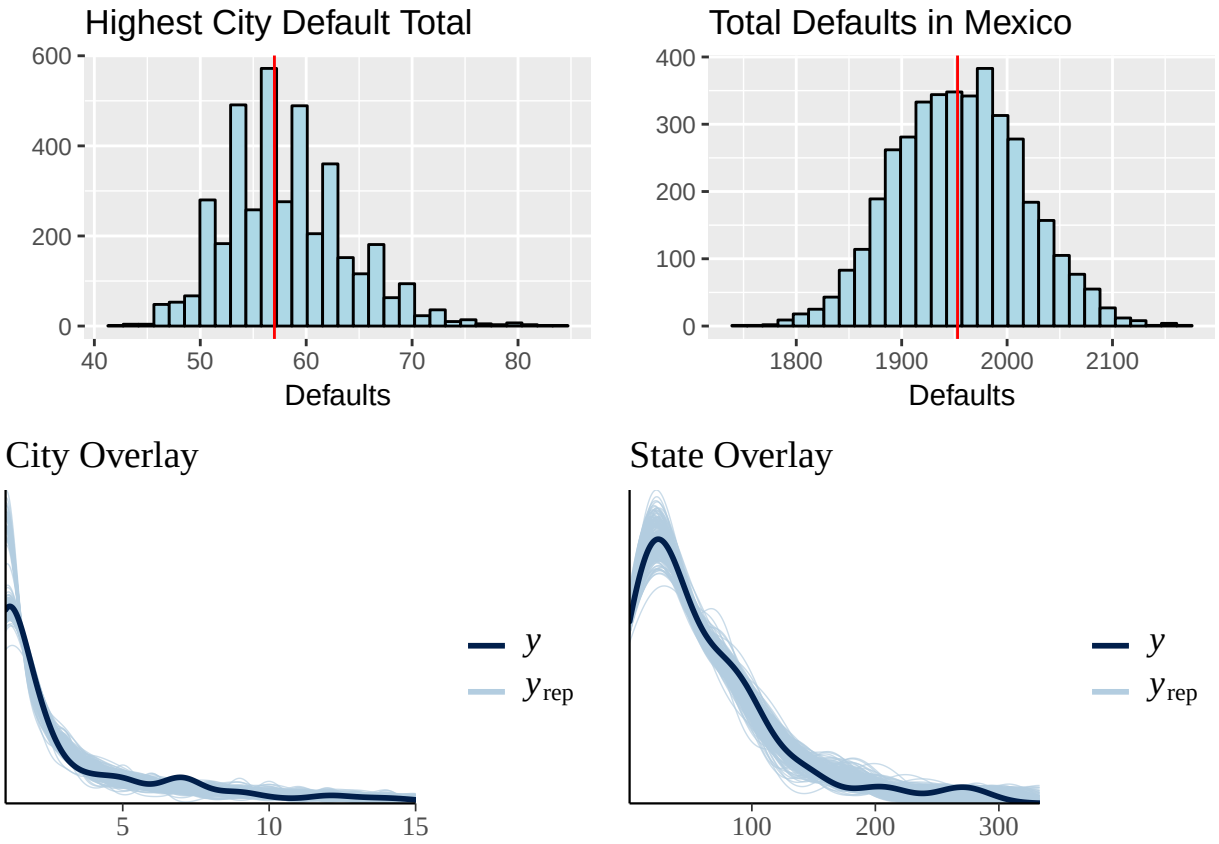
4.1.2 Parameter Estimates

```
## Inference for Stan model: binomial_GP.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##        mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## alpha  0.47    0.01 0.19  0.19  0.33  0.44  0.58  0.94  1289    1
## rho    2.18    0.03 0.78  1.04  1.63  2.06  2.59  4.03   955    1
## a     -2.40    0.01 0.29 -2.86 -2.59 -2.44 -2.25 -1.67  1468    1
##
## Samples were drawn using NUTS(diag_e) at Sat Dec  8 05:46:11 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

From the value of ρ we see that it is close to 1.5 and does indicates certain level of smoothness (which is evidence that our linear models were a defensible modelling assumption). Furthermore, the intercept a is much lower than in the previous models (where it was denoted as α). This is relevant to our analysis since it suggests that possible interactions of the given variables make take some of the explanatory effect from the intercept. This is reinforced by the fact that it is not the nonlinearities that are taking the weight-off the intercept (since ρ is close to 2) but rather some interactions.

4.1.3 Evaluation

Below is the performance of the GP over our set of four relevant PPCs



were we see that the model, at this levels, equally as good as the previous models. We were concerned that adding much more complexity could distort this. Moreover, we did a a 5 CV fold where the value was 1.9163 (0.228). Which shows that there is not a clear improvements.

At this point we were unsure on how to proceed extracting more value out of our GP analysis. We also leave for future work to understand if there were some interactions suggested by the model as well as the nonlinearities (if present).

4.2 Zero-inflated Binomial ICAR - adding a mixture component

In this section, we will analyze the dataset on individual level again. The model is logistic regression with CAR prior. The likelihood of the individual i from the state j is:

$$y_{ij} \sim Bernoulli(logit^{-1}(a + \phi_j + x_{ij}\beta))$$

The zero-inflated extension could also be applied in logsitic regression:

$$p(y_{ij}|\theta, a, \beta, \phi) = \begin{cases} \theta + (1-\theta) \times bernoulli(y_{ij}|a, \beta, \phi) & , y_{ij} = 0 \\ (1-\theta) \times bernoulli(y_{ij}|a, \beta, \phi) & , y_{ij} > 0 \end{cases}$$

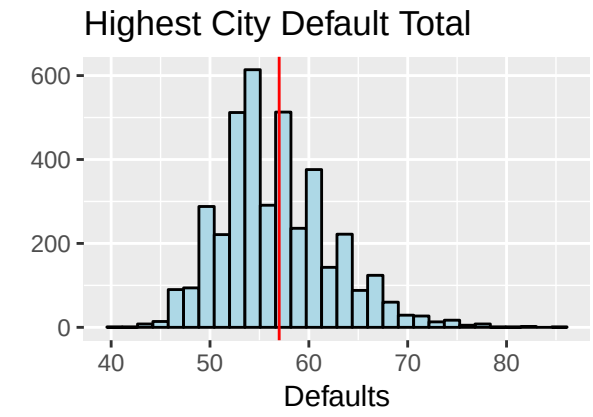
Part of the model result is in the following:

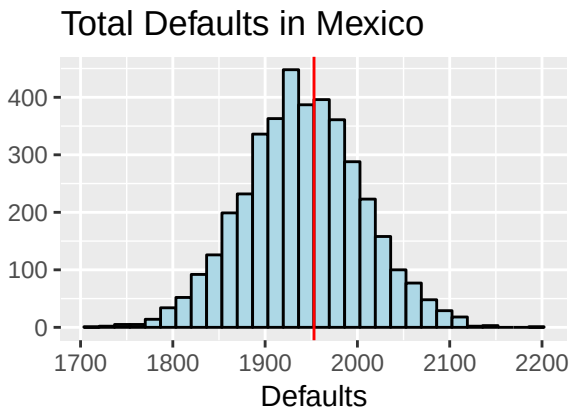
Parameters	mean	se_mean	sd	1%	neff	Rhat
alpha	-0.82	0.18	1.09	-2.64	36	1.11
beta[1]	-0.13	0.02	0.18	-0.56	100	1.04
beta[2]	0.08	0.04	0.25	-0.26	32	1.13
beta[3]	0.03	0.06	0.27	-1.08	23	1.14
beta[4]	0.01	0.03	0.15	-0.29	27	1.14
beta[5]	-0.08	0.05	0.27	-0.67	30	1.13
beta[6]	-0.27	0.04	0.23	-1.03	30	1.13
beta[7]	-0.02	0.00	0.06	-0.22	154	1.01
theta	0.69	0.03	0.23	0.05	64	1.06
lp___	-7215.11	2.34	13.51	-7251.54	33	1.11

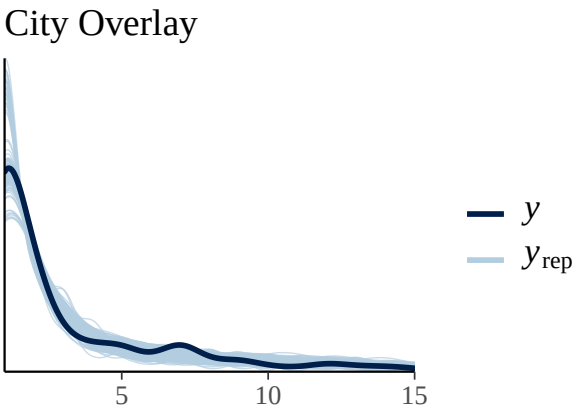
The individual level model did not converge as well as the city level model and it takes more time. However, θ is estimated to be 0.69 which means about 69% of the individual would not default at all. This indicates the zero-inflated extension on the individual level would be more useful.

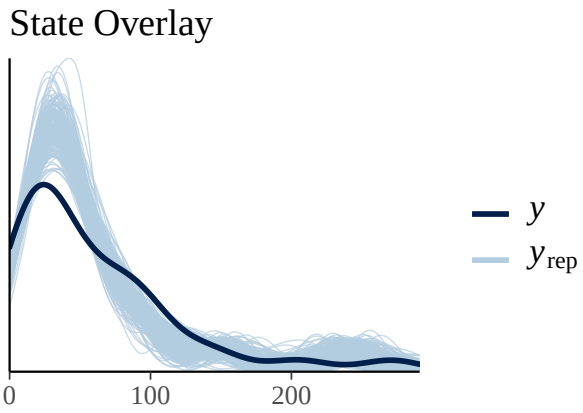
4.2.1 Evaluation

On the PPCs is where we start to see the detrimental effects of the lack of convergence that we saw. First, note how the density overlay for state worsens

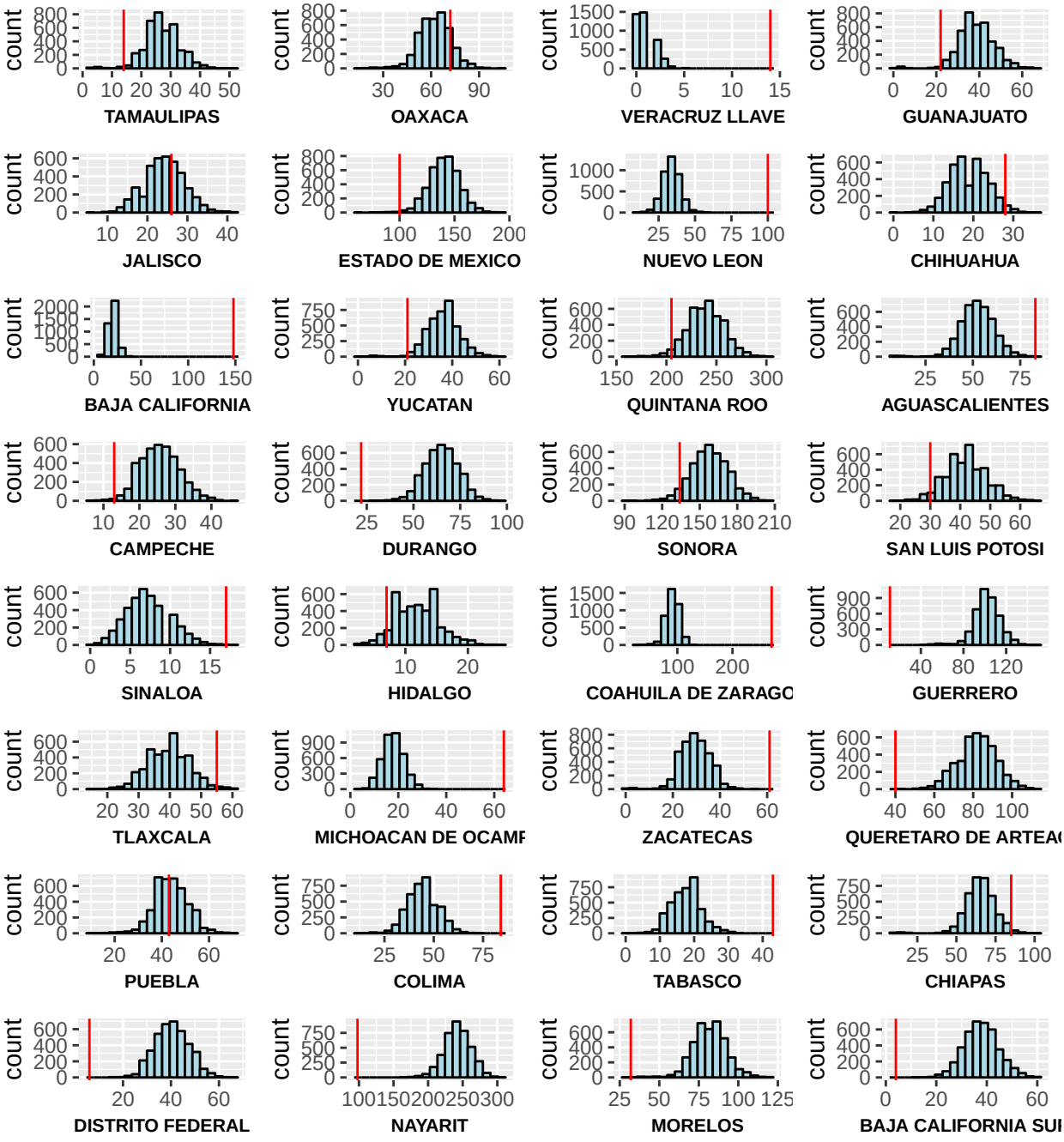








Finally, look how the PPCs for the states are completely off. Which is something that should now happen since we are including a coefficient for them. Thus it displays the lack of convergence.


count
TAMAULIPAS
OAXACA
VERACRUZ LLAVE
GUANAJUATO
JALISCO
ESTADO DE MEXICO
NUEVO LEON
CHIHUAHUA
BAJA CALIFORNIA
YUCATAN
QUINTANA ROO
AGUASCALIENTES
CAMPECHE
DURANGO
SONORA
SAN LUIS POTOSI
SINALOA
HIDALGO
COAHUILA DE ZARAGO
GUERRERO
TLAXCALA
MICHOACAN DE OCAMP
ZACATECAS
QUERETARO DE ARTEA
PUEBLA
COLIMA
TABASCO
CHIAPAS
DISTRITO FEDERAL
NAYARIT
MORELOS
BAJA CALIFORNIA SUI

5. Conclusion and Learnings

5.1 Conclusion

From our previous finding we derived the following recommendations to the start-up:

- **Prioritize the expansion to cities that do not belong to a poor state**. We saw from the ICAR model that there was a positive fix effect for all the poor states which indicates an unfavorable inclination for default. We understand that this recommendation might perpetuate a vicious cycle where poorer states do not receive credit. However, without any further variables for screening, the start up should refrain from expanding to states that might threaten its development.
- **Focus on measuring additional employment variables for new applicants but also obtain additional credit score variables**. The % of unemployed people in a city was the most relevant predictor as well as the avg. risk index. Thus, we recommend that the start-up places more emphasis on the recollection of variables related to the inherent risk of the applicant as well as its employment status. (Rather than income and mortgage which had great uncertainty).
- **Understand further why the percentage of new houses in the city is a relevant predictor**. We hypothesize that maybe individuals that buy new houses do so, in the majority of the cases, as an investment. Therefore, they might be more likely to leave the house when they are under financial distress; in contrast to someone who is actually residing in that house with its family.

5.2 Learning Experiences

We summarize our learning experiences.

Andres' Learnings:

- **Defining PPCs it not a mechanical excercise**. We can always check for quantities such as the mean, sd, max, min and so on. However, if our model has a mean and variance parameter we should not be too off for those measurements (specially when the densities properly overlay) thus they are uninformative. In contrast, coming up with test aspects where the model is not fitted does require ingenuity but also a deep understanding of the problem.
- **Trying out different models to the data is a great way to gain insight about it**. It was until we tried different model specifications that we saw aspects of the data that were not clear in the very beginning. Therefore, most of the initial models that we tried were instrumental in understanding what aspects of the data were important to model.
- **Simulating fake data is vital to understanding how influencial are our priors**. Simulating fake data for different hyperparameter values allowed us to understand what the impact of the scale was (say how informative or not was to say that parameter had an sd of 1) but also to see the limitation of noise that our model can handle with the present data.
- **CV accuracy is not the definitive metric to chose a model**. Our Binomial ICAR model did not had the best 5 fold RMSE of all our models but it was both close to the minimum and, most importantly, it had a good performance over the two sets of PPCs which we were interested in getting right. Also, CV is an approximation to the data generating density which we do not observe, but it is biased anyways. Hence, we feel more secure and comfortable with a model whose data is able to replicate the quantities which we are most interested on.
- **Z-scoring the variables is fundamental for improving running time**. A Gaussian Process is a really intensive model. I made the mistake of running this model without standardizing the variables and it took over 25 hours. However, once I standardized those variables the running time was reduced to less than 10 hours. In this respect it was also an important learning to see how intensive where the nonparametric methods were we had to set-up a cluster on the cloud to leave the models running.

Jongwoo's Learnings:

1. This project helped me to understand more about simple binomial models.
2. I learned about how to expand from simple bayesian model to complicated models.
3. It increased my programming skills such as R and Stan.
4. I learned more about bayesian data analysis outside the book.
5. I learned about basic bayesian workflow, from simulate fake data to perform model checking and validation.

Yi's Learnings:

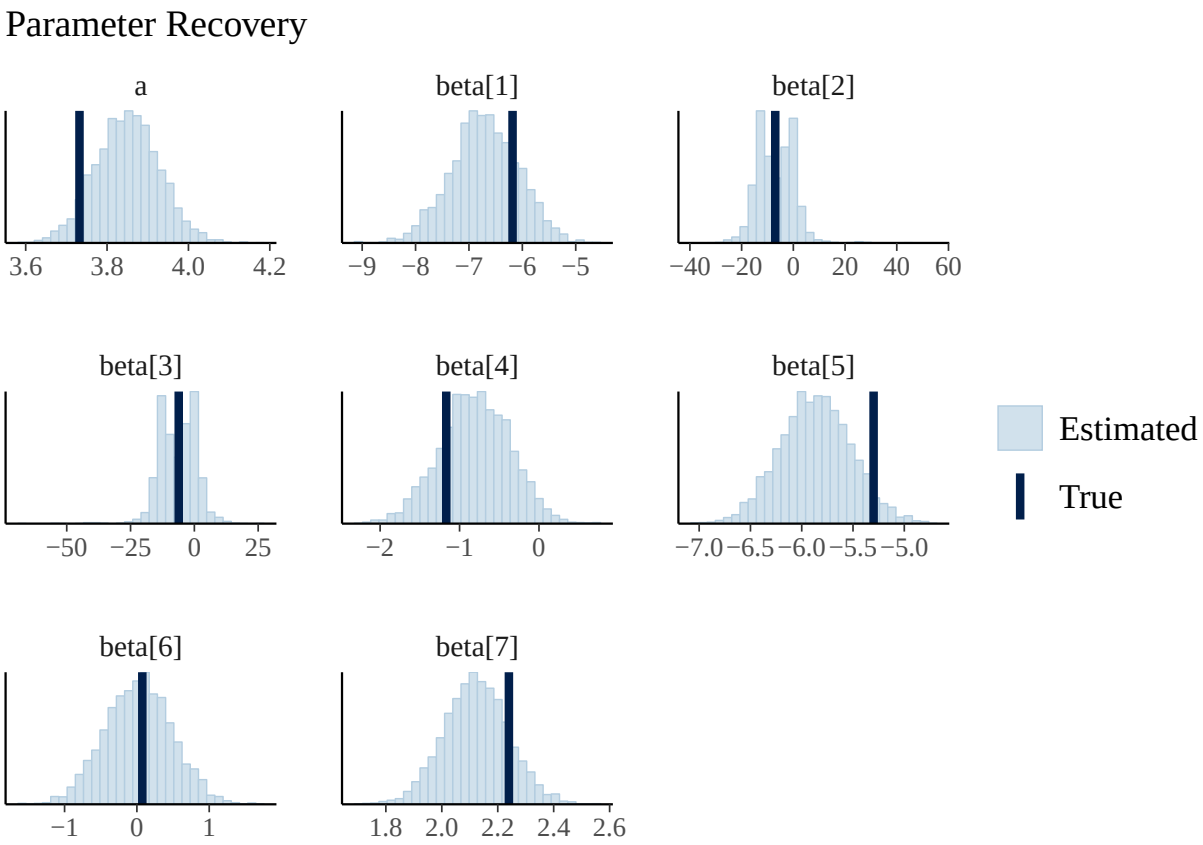
1. I learn more about mix models, in particular, zero-inflated model, and how to apply these models using R and stan.
2. I learn more about the CAR models, in particular, ICAR model, and how to incorporate the geographic information into the model.
3. I learn more about cross-validation in Bayesian Statistics and how to use RMSE to compare the models.
4. I learn more about how to use 'bayesplot' and 'shinystan' packages for model estimation, diagnosis, and checking.
5. I learn more about how count regressions, including poisson, logistic, binomial and negative binomial regression.

6. STAN Code Appendix

6.1 Parameter Recovery

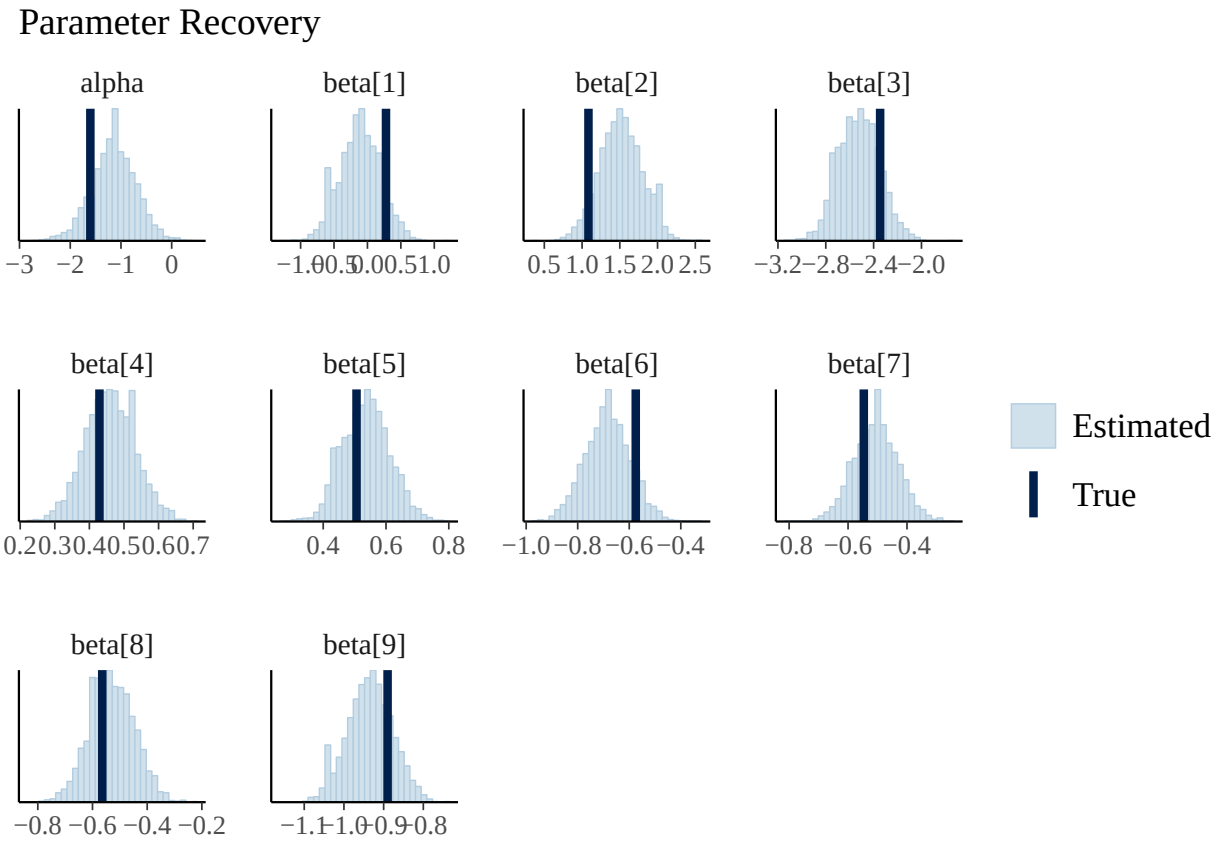
6.1 Parameter Recovery - Binomial Regression

We simulated fake data for the binomial model and we verified that it recovered the parameters. The results are summarized in the histograms below.



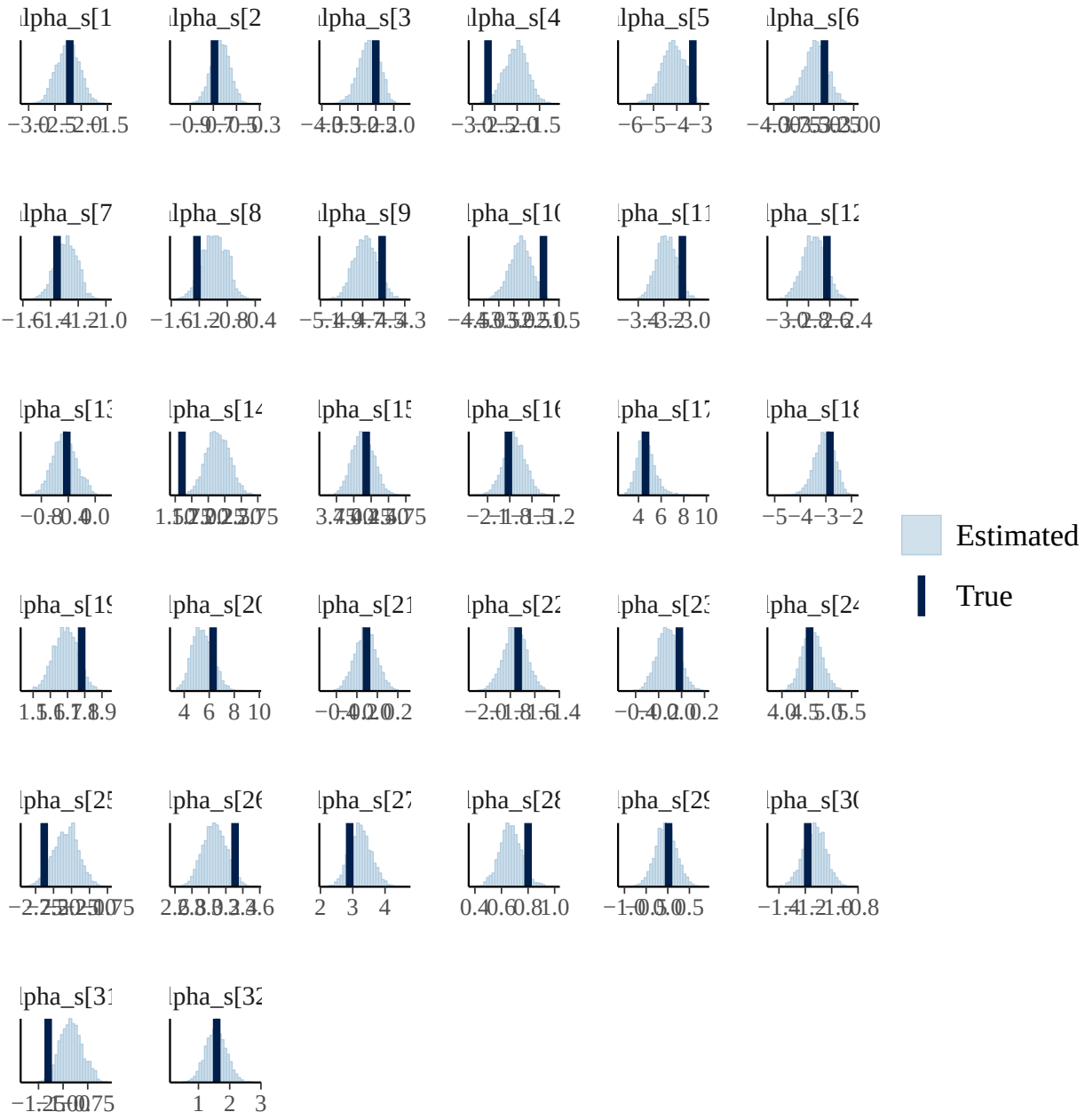
where we see that we have recovered the parameters successfully.

6.2 Parameter Recovery - Hierarchical Binomial Regression

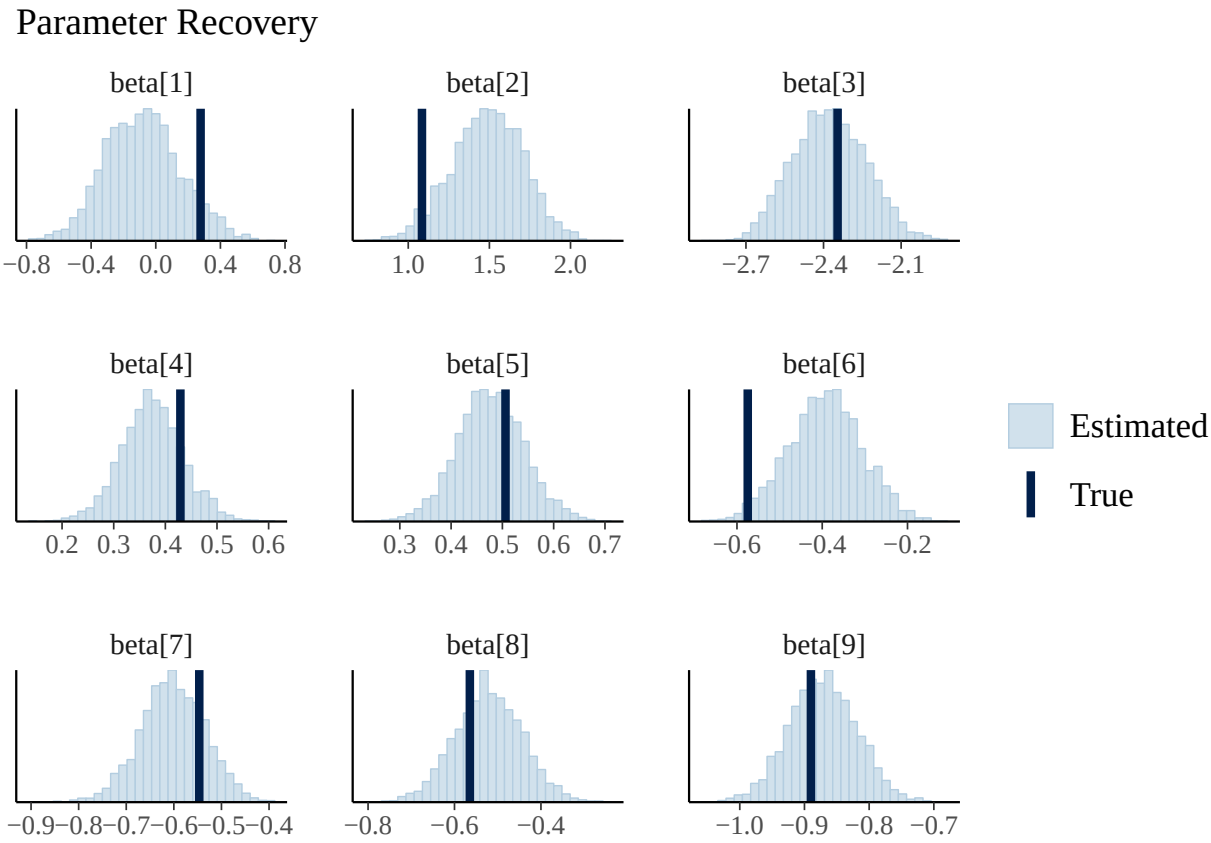


And for each individual alpha

Parameter Recovery



6.3 Parameter Recovery - Binomial ICAR Regression



6.2 STAN Code - Binomial

All data

```
data {
  int<lower=1> N;
  int<lower=1> D;
  int<lower=0> city_n[N];
  matrix[N, D] X;
  int<lower=0> y[N];
```

```
} parameters {
  real alpha;
  vector[D] beta;
} model {
  alpha ~ normal(0, 5);
  beta ~ normal(0, 5);
  y ~ binomial_logit(city_n, alpha + X * beta);
} generated quantities {
  int<lower=0> y_rep [N];
  for (i in 1:N){
    y_rep[i] = binomial_rng(city_n[i],
    inv_logit(alpha + X[i, :] * beta));
  }
}
```

For Training

```
data {
  int<lower=1> N_train;
  int<lower=1> N_test;
  int<lower=1> D;
  int<lower=0> city_n_train[N_train];
  int<lower=0> city_n_test[N_test];
  matrix[N_train, D] X_train;
  matrix[N_test, D] X_test;
  int<lower=0> y_train[N_train];
} parameters {
  real alpha;
  vector[D] beta;
} model {
  alpha ~ normal(0, 5);
  beta ~ normal(0, 5);
  y_train ~ binomial_logit(city_n_train, alpha + X_train * beta);
} generated quantities {
  int<lower=0> y_rep_train [N_train];
  int<lower=0> y_rep_test [N_test];
  for (i in 1:N_train){
    y_rep_train[i] = binomial_rng(city_n_train[i],
    inv_logit(alpha + X_train[i, :] * beta));
  }
  for (i in 1:N_test){
    y_rep_test[i] = binomial_rng(city_n_test[i],
    inv_logit(alpha + X_test[i, :] * beta));
  }
}
```

6.2 STAN Code - Hierarchical Binomial

All data

```
data {
  int<lower=1> N;
  int<lower=1> D;
  int<lower=1> S;
  int<lower=1> city_n[N];
  int<lower=1, upper=S> state[N];
  matrix[N, D] X;
  int<lower=0> y[N];
} parameters {
  real alpha;
  vector[D] beta;
  vector[S] alpha_s_raw;
  vector<lower=0>[S] sigma_alpha_s;
} transformed parameters {
  vector[S] alpha_s;
  vector[S] ones = rep_vector(1, S);
  alpha_s = alpha * ones + alpha_s_raw .* sigma_alpha_s;
} model {
  alpha ~ normal(0, 5);
  beta ~ normal(0, 5);
  sigma_alpha_s ~ lognormal(1, 1);
  alpha_s_raw ~ std_normal();
  y ~ binomial_logit(city_n, alpha_s[state] + X * beta);
} generated quantities {
  int<lower=0> y_rep[N];
  for (i in 1:N){
    y_rep[i] = binomial_rng(city_n[i],
    inv_logit(alpha_s[state[i]] + X[i, :] * beta));
  }
}
```

For Training

```
data {
  int<lower=1> N_train;
  int<lower=1> N_test;
  int<lower=1> D;
  int<lower=1> S;
  int<lower=1> city_n_train[N_train];
  int<lower=1> city_n_test[N_test];
  int<lower=1, upper=S> state_train[N_train];
  int<lower=1, upper=S> state_test[N_test];
  matrix[N_train, D] X_train;
  matrix[N_test, D] X_test;
  int<lower=0> y_train[N_train];
} parameters {
  real alpha;
  vector[D] beta;
```

```
  vector[S] alpha_s_raw;
  vector<lower=0>[S] sigma_alpha_s;
} transformed parameters {
  vector[S] alpha_s;
  vector[S] ones = rep_vector(1, S);
  alpha_s = alpha * ones + alpha_s_raw .* sigma_alpha_s;
} model {
  alpha ~ normal(0, 5);
  beta ~ normal(0, 5);
  sigma_alpha_s ~ lognormal(1, 1);
  alpha_s_raw ~ std_normal();
  y_train ~ binomial_logit(city_n_train,
  alpha_s[state_train] + X_train * beta);
} generated quantities {
  int<lower=0> y_rep_train[N_train];
  int<lower=0> y_rep_test[N_test];
  for (i in 1:N_train){
    y_rep_train[i] = binomial_rng(city_n_train[i],
    inv_logit(alpha_s[state_train[i]] + X_train[i, :] * beta));
  }
  for (i in 1:N_test){
    y_rep_test[i] = binomial_rng(city_n_test[i],
    inv_logit(alpha_s[state_test[i]] + X_test[i, :] * beta));
  }
}
```

6.3 STAN Code - Binomial ICAR

All data

```
functions {
  real sparse_iar_lpdf(vector phi,
                       real tau,
                       int[,] W_sparse,
                       vector D_sparse,
                       vector lambda,
                       int S,
                       int W_n) {
      row_vector[S] phit_D;
      row_vector[S] phit_W;
      vector[S] ldet_terms;

      phit_D = (phi .* D_sparse)'';
      phit_W = rep_row_vector(0, S);
      for (i in 1:W_n) {
        phit_W[W_sparse[i, 1]] = phit_W[W_sparse[i, 1]] + phi[W_sparse[i, 2]];
        phit_W[W_sparse[i, 2]] = phit_W[W_sparse[i, 2]] + phi[W_sparse[i, 1]];
      }

      return 0.5 * ((S-1) * log(tau)
                    - tau * (phit_D * phi - (phit_W * phi)));
  }
} data {
  int<lower = 1> N;
  int<lower = 1> D;
  int<lower = 1> S;
  int<lower = 1> city_n[N];
  int<lower=1> state[N];    // state indicator for train set
  matrix[N, D] X;           // raw design matrix for train set
  int<lower = 0> y[N];      // response for train set
  matrix<lower = 0, upper = 1>[S, S] W; // adjacency matrix
  int W_n;                              // number of adjacent region pairs
} transformed data {
  int W_sparse[W_n, 2];   // adjacency pairs
  vector[S] D_sparse;     // diagonal of D (number of neigbors for each site)
  vector[S] lambda;       // eigenvalues of invsqrtD * W * invsqrtD
  { // generate sparse representation for W
  int counter;
  counter = 1;
    for (i in 1:(S - 1)) {
      for (j in (i + 1):S) {
        if (W[i, j] == 1) {
          W_sparse[counter, 1] = i;
          W_sparse[counter, 2] = j;
          counter = counter + 1;
        }
      }
    }
  }
```

```
  for (i in 1:S) D_sparse[i] = sum(W[i]);
  {
    vector[S] invsqrtD;
    for (i in 1:S) {
      invsqrtD[i] = 1 / sqrt(D_sparse[i]);
    }
    lambda = eigenvalues_sym(quad_form(W, diag_matrix(invsqrtD)));
  }
} parameters {
  vector[D] beta;
  vector[S] phi_unscaled;
  real<lower = 0> tau;
  real alpha;
} transformed parameters {
  vector[S] phi;
  phi = phi_unscaled - mean(phi_unscaled);
} model {
  tau ~ gamma(2, 2);
  phi_unscaled ~ sparse_iar(tau, W_sparse, D_sparse, lambda, S, W_n);
  beta ~ normal(0, 5);
  alpha ~ normal(0, 5);
  y ~ binomial_logit(city_n,
  alpha + phi[state] + X * beta);
} generated quantities {
  int<lower=0> y_rep[N];
  for (i in 1:N){
    y_rep[i] = binomial_rng(city_n[i],
    inv_logit(alpha + phi[state[i]] + X[i, :] * beta));
  }
}
```

For Training

```
functions {
  real sparse_iar_lpdf(vector phi,
                       real tau,
                       int[,] W_sparse,
                       vector D_sparse,
                       vector lambda,
                       int S,
                       int W_n) {
      row_vector[S] phit_D;
      row_vector[S] phit_W;
      vector[S] ldet_terms;

      phit_D = (phi .* D_sparse)'';
      phit_W = rep_row_vector(0, S);
      for (i in 1:W_n) {
        phit_W[W_sparse[i, 1]] = phit_W[W_sparse[i, 1]] + phi[W_sparse[i, 2]];
        phit_W[W_sparse[i, 2]] = phit_W[W_sparse[i, 2]] + phi[W_sparse[i, 1]];
      }
```

```
      return 0.5 * ((S-1) * log(tau)
                    - tau * (phit_D * phi - (phit_W * phi)));
  }
} data {
  int<lower = 1> N_train;               // number of record of train set
  int<lower = 1> N_test;                // number of record of test set
  int<lower = 1> D;                     // number of non-geo parameter
  int<lower = 1> S;                     // number of state
  int<lower=1> city_n_train[N_train];   // number of record for city k in
  int<lower=1> city_n_test[N_test];     // number of record for city k in
  int<lower=1> state_train[N_train];    // state indicator for train set
  int<lower=1> state_test[N_test];      // state indicator for test set
  matrix[N_train, D] X_train;           // raw design matrix for train set
  matrix[N_test, D] X_test;             // raw design matrix for train set
  int<lower = 0> y_train[N_train];      // response for train set
  matrix<lower = 0, upper = 1>[S, S] W; // adjacency matrix
  int W_n;                              // number of adjacent region pairs
} transformed data {
  int W_sparse[W_n, 2];   // adjacency pairs
  vector[S] D_sparse;     // diagonal of D (number of neigbors for each site)
  vector[S] lambda;       // eigenvalues of invsqrtD * W * invsqrtD
  { // generate sparse representation for W
  int counter;
  counter = 1;
  // loop over upper triangular part of W to identify neighbor pairs
    for (i in 1:(S - 1)) {
      for (j in (i + 1):S) {
        if (W[i, j] == 1) {
          W_sparse[counter, 1] = i;
          W_sparse[counter, 2] = j;
          counter = counter + 1;
        }
      }
    }
  }
  for (i in 1:S) D_sparse[i] = sum(W[i]);
  {
    vector[S] invsqrtD;
    for (i in 1:S) {
      invsqrtD[i] = 1 / sqrt(D_sparse[i]);
    }
    lambda = eigenvalues_sym(quad_form(W, diag_matrix(invsqrtD)));
  }
} parameters {
  vector[D] beta;
  vector[S] phi_unscaled;
  real<lower = 0> tau;
  real<lower=0, upper=1> theta;                 // probability of draw a zero
  real alpha;
} transformed parameters {
  vector[S] phi;                                // brute force centering
  phi = phi_unscaled - mean(phi_unscaled);
} model {
```

```
  tau ~ gamma(2, 2);
  phi_unscaled ~ sparse_iar(tau, W_sparse, D_sparse, lambda, S, W_n);
  beta ~ normal(0, 5);
  alpha ~ normal(0, 5);
  y_train ~ binomial_logit(city_n_train,
  alpha + phi[state_train] + X_train * beta);
} generated quantities {
  int<lower=0> y_rep_train[N_train];
  int<lower=0> y_rep_test[N_test];
  for (i in 1:N_train){
    y_rep_train[i] = binomial_rng(city_n_train[i],
    inv_logit(alpha + phi[state_train[i]] + X_train[i, :] * beta));
  }
  for (i in 1:N_test){
    y_rep_test[i] = binomial_rng(city_n_test[i],
    inv_logit(alpha + phi[state_test[i]] + X_test[i, :] * beta));
  }
}
```

6.4 STAN Code - Gaussian GP

All data

```
data {
  int<lower=0> N;
  int<lower=0> city_n[N];
  int<lower=0> D;
  vector[D] X[N];
  int<lower=0> y[N];
} transformed data {
real delta = 1e-9;
} parameters {
  real a;
  real<lower=0> alpha;
  real<lower=0> rho;
  vector[N] eta;
} transformed parameters {
  vector[N] f;
  {
    matrix[N, N] L_K;
    matrix[N, N] Ker = cov_exp_quad(X, alpha, rho);

    for (n in 1:N) {
      Ker[n, n] = Ker[n, n] + delta;
    }

    L_K = cholesky_decompose(Ker);
    f = L_K * eta;
  }
} model {
  a ~ std_normal();
  alpha ~ std_normal();
  rho ~ inv_gamma(7, 7);
  eta ~ std_normal();
  y ~ binomial_logit(city_n, a + f[1:N]);
} generated quantities {
  int<lower=0> y_rep[N];
  for (i in 1:N) {
    yrep[i] = binomial_rng(city_n[i], inv_logit(a + f[i]));
    }
}
```

For Training

```
data {
  int<lower=0> N_train;
  int<lower=0> N_test;
  int<lower=0> city_n_train[N_train];
  int<lower=0> city_n_test[N_test];
  int<lower=0> D;
  vector[D] X_train[N_train];
```

```
  vector[D] X_test[N_test];
  int<lower=0> y_train[N_train];
} transformed data {
  real delta = 1e-9;
  int<lower=0> N = N_train + N_test;
  vector[D] X[N];
  for (i in 1:N_train) X[i, ] = X_train[i, ];
  for (i in 1:N_test) X[N_train + i, ] = X_test[i, ];
} parameters {
  real a;
  real<lower=0> alpha;
  real<lower=0> rho;
  vector[N] eta;
} transformed parameters {
  vector[N] f;
  {
    matrix[N, N] L_K;
    matrix[N, N] Ker = cov_exp_quad(X, alpha, rho);

    for (n in 1:N) {
      Ker[n, n] = Ker[n, n] + delta;
    }

    L_K = cholesky_decompose(Ker);
    f = L_K * eta;
  }
} model {
  a ~ std_normal();
  alpha ~ std_normal();
  rho ~ inv_gamma(7, 7);
  eta ~ std_normal();
  y_train ~ binomial_logit(city_n_train, a + f[1:N_train]);
} generated quantities {
  int<lower=0> y_rep_train[N_train];
  int<lower=0> y_rep_test[N_test];
  for (i in 1:N_train) {
    y_rep_train[i] = binomial_rng(city_n_train[i],
    inv_logit(a + f[i]));
    }
  for (i in 1:N_test) {
    y_rep_test[i] = binomial_rng(city_n_test[i],
    inv_logit(a + f[N_train + i]));
  }
}
```

6.5 STAN Code - Zero-inflated Binomial ICAR

```
functions {
  real sparse_iar_lpdf(vector phi, real tau, int[,] W_sparse, vector D_sparse, vector lambda, int S, in
      row_vector[S] phit_D;
      row_vector[S] phit_W;
      vector[S] ldet_terms;

      phit_D = (phi .* D_sparse)'';
      phit_W = rep_row_vector(0, S);
      for (i in 1:W_n) {
        phit_W[W_sparse[i, 1]] = phit_W[W_sparse[i, 1]] + phi[W_sparse[i, 2]];
        phit_W[W_sparse[i, 2]] = phit_W[W_sparse[i, 2]] + phi[W_sparse[i, 1]];
      }

      return 0.5 * ((S-1) * log(tau)
                    - tau * (phit_D * phi - (phit_W * phi)));
  }
}

data {
  int<lower = 1> S;                       // number of state
  int<lower = 1> N_train;                 // number of record of train set
  int<lower = 1> D;                       // number of non-geo parameter
  matrix[N_train, D] X_train;             // raw design matrix for train set
  int<lower = 0> y[N_train];              // response for train set
  int<lower=1> state_train[N_train];      // state indicator for train set
  matrix<lower = 0, upper = 1>[S, S] W; // adjacency matrix
  int W_n;                                // number of adjacent region pairs
  int<lower=1> n_city_train[N_train];     // number of record for city k in training set
}

transformed data {
  int W_sparse[W_n, 2];   // adjacency pairs
  vector[S] D_sparse;     // diagonal of D (number of neigbors for each site)
  vector[S] lambda;       // eigenvalues of invsqrtD * W * invsqrtD
  { // generate sparse representation for W
  int counter;
  counter = 1;
  // loop over upper triangular part of W to identify neighbor pairs
    for (i in 1:(S - 1)) {
      for (j in (i + 1):S) {
        if (W[i, j] == 1) {
          W_sparse[counter, 1] = i;
          W_sparse[counter, 2] = j;
          counter = counter + 1;
        }
      }
    }
  }
  for (i in 1:S) D_sparse[i] = sum(W[i]);
  {
    vector[S] invsqrtD;
```

```
    for (i in 1:S) {
      invsqrtD[i] = 1 / sqrt(D_sparse[i]);
    }
    lambda = eigenvalues_sym(quad_form(W, diag_matrix(invsqrtD)));
  }
}

parameters {
  vector[D] beta;
  vector[S] phi_unscaled;
  real<lower = 0> tau;
  real<lower=0, upper=1> theta;                  // probability of draw a zero
  real alpha;
}

transformed parameters {
  vector[S] phi;                                 // brute force centering
  phi = phi_unscaled - mean(phi_unscaled);
}

model {
  tau ~ gamma(2, 2);
  phi_unscaled ~ sparse_iar(tau, W_sparse, D_sparse, lambda, S, W_n);
  beta ~ cauchy(0, 2.5);
  alpha ~ cauchy(0, 10);
  for (j in 1:N_train){
    if (y[j] == 0){
      target += log_sum_exp(bernoulli_lpmf(1 | theta), bernoulli_lpmf(0 | theta) + binomial_logit_lpmf(
      }
    else{
      target += bernoulli_lpmf(0 | theta) + binomial_logit_lpmf(y[j] | n_city_train[j],alpha + phi[stat
  }
}
generated quantities{
  int y_rep[N_train];
  real<lower =0,upper=1> zero_train[N_train];
  for (i in 1:N_train){
    zero_train[i] = uniform_rng(0,1);
    if (zero_train[i] < theta){
      y_rep[i] = 0;
      }
    else{
      y_rep[i] = binomial_rng(n_city_train[i],inv_logit(alpha + phi[state_train[i]] + X_train[i,] * beta
      }
  }
}
```

7. References

- Besag, Julian, Jeremy York, and Annie Mollié. (1991) Bayesian image restoration, with two applications in spatial statistics. *Annals of the Institute of statistical mathematics*, 43.1: 1-20.

- Gelfand, Alan E., and Penelope Vounatsou. (2003) Proper multivariate conditional autoregressive models for spatial data analysis. *Biostatistics* 4.1: 11-15.
- Jin, Xiaoping, Bradley P. Carlin, and Sudipto Banerjee. (2005) Generalized hierarchical multivariate CAR models for real data. *Biometrics* 61.4: 950-961.
- Bob Carpenter. (2018) Predator-Prey Population Dynamics: the Lotka-Volterra model in Stan. *Rstan Document For Example*.
- Max Joseph. (2011) Exact sparse CAR models in Stan. *Rstan Document For Example*.
- Stan Development Team (2017) *Stan Modeling Language Users Guide and Reference Manual*, Version 2.17, http://mc-stan.org.
- Vehtari, A., Gelman, A. & Gabry, J. (2017) Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Journal of Statistics and Computing* 27(5):1413–1432.
- Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, Donald B. Rubin (2014) *Bayesian Data Analysis*
- Stan Development Team (2018) *Bayesian Statistics Using Stan*