Zero Inflated Models: Complete vs. Partial Pooling

REPORT

By

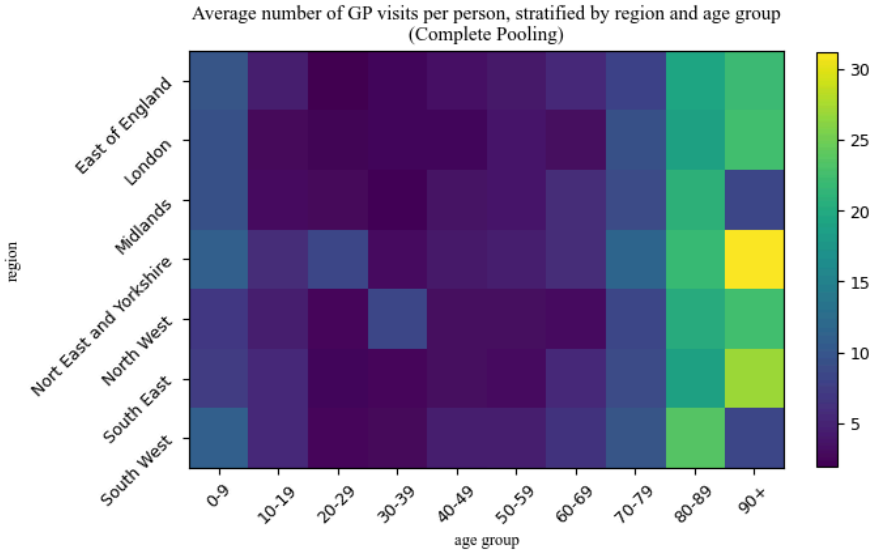
Giorgi Kukishvili

Minerva University

CS146 - Computational Methods for Bayesian Statistics

Prof. E. Volkan

March 16, 2024



Word count: 2843 Words

Strategy

We were provided with the data set in the shape of a 7x10x30 array for the 7 geographical regions, 10 age groups, and 30 samples per group. However, the data set had some missing/NaN values that comprised five age and region clusters. The missing data is visualized in the figure below.

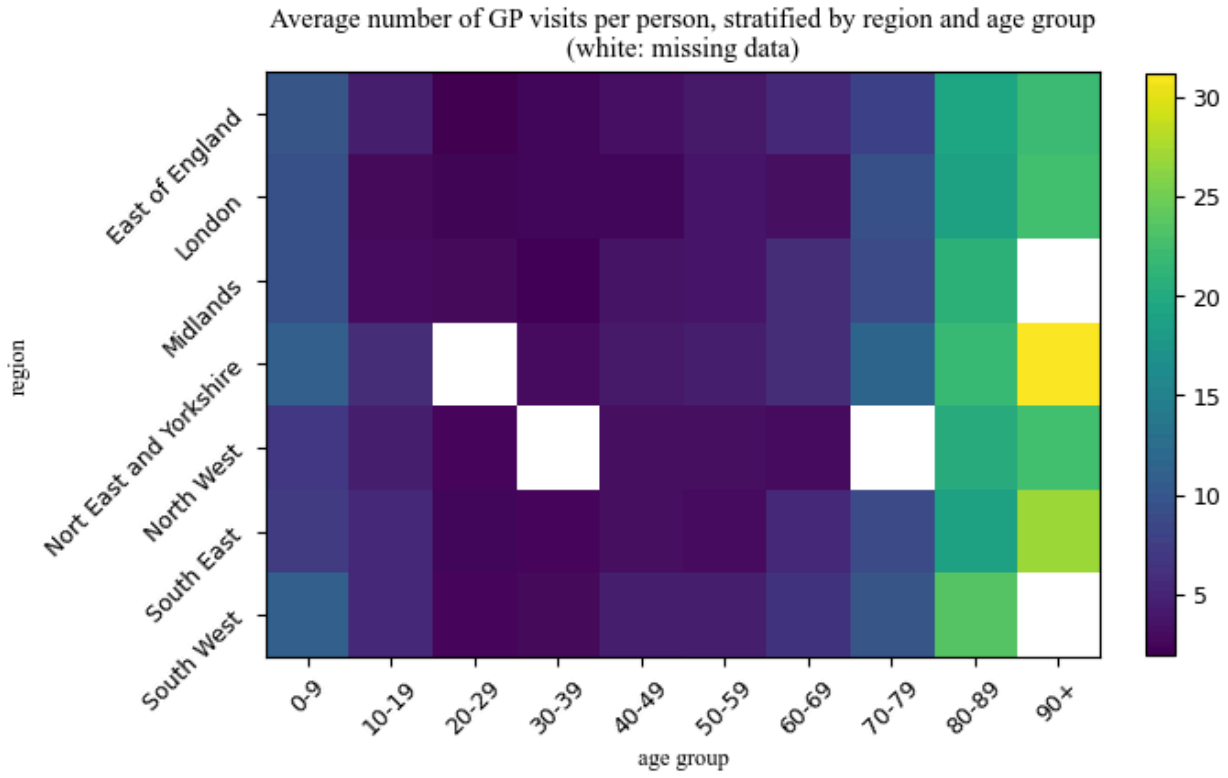


Figure 1. Average number of GP visits per person.

We are tasked with estimating the values of the missing groups. To do this we will first pre-process the data to make it easier to work with in PyMC. Next, we will implement a Zero-Inflated Poisson likelihood model with complete pooling. Following this, we will assume a hierarchical model where each group has a Zero-Inflated Poisson distribution with its own parameters and we will set up not only prior parameters but also hyperpriors for this hierarchical model. Finally, we will show the predictions of the values of

each missing group, explain the differences between the predictions of the complete and partial pooling models we created, and compare these models.

Data Pre-Processing

To make the data easier to work with in PyMC I will disaggregate the raw data and reorganize it into 3 numpy arrays: `Region`, `Age`, and `Counts`. The region array will be an integer from 0-6 indicating which geographical region each count is from, the Age array will be an integer for each age category, and the Count array will be the number of GP visits. Each array will have the same length and the *i*-th element of each array will correspond to the same person. This was accomplished by converting the lists of raw data to numpy arrays, after which I created a dictionary with corresponding column names and values, and then created a pandas dataframe from this dictionary. Below is the first couple of rows of the final dataset.

Table: 1

Index	Region	Age	Counts
0	0	0	4.0
1	0	0	11.0
2	0	0	14.0
3	0	0	13.0
4	0	0	12.0

After pre-processing and removing the NaN values from the data which had a shape of $7 \times 10 \times 30$ now has a shape of 1950×3 and is ready to be modeled using PyMC. However, before we set up the model let's see some summary statistics and distribution.

Table 2. Summary Statistics

Metric	Value
Mean number of GP visits	8
Median number of GP visits	4.0
Mode	0
Standard deviation	11
95% Interval	[0, 40]
Range	0.0 to 84.0

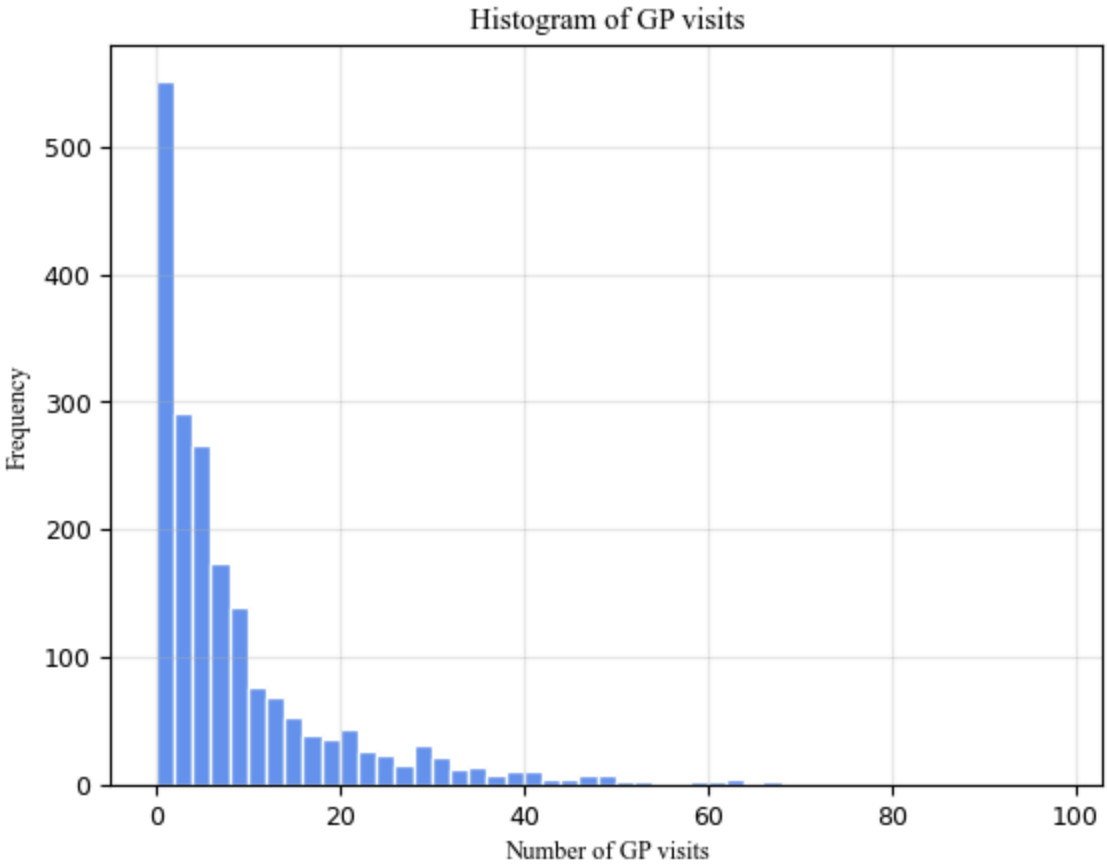


Figure 2. Histogram of GP Visits per year.

To quickly summarize the data, let's look into the summary statistics which reveal insights into GP visit patterns. On average, individuals made approximately 8 visits, with a median of 4.0, indicating a skewed distribution with many lower counts. The mode of 0 suggests a notable portion had no visits. The standard deviation of 11 highlights variability around the mean. The 95% interval spans 0 to 40, with a range from 0.0 to 84.0, indicating diverse visit frequencies, including both frequent users and those with no visits. The visits could be influenced by age group, availability of GPs in a given region, individual needs, economic and genetic predispositions, and many more. We will expand on this once we discuss the model.

Complete Pooling

Since a lot of people never visit the doctor we will use a Zero-Inflated Poisson likelihood function for this data set. The reasoning behind the high frequency of 0 visits can vary from individual to individual. Some people do not like doctors and do not take their health seriously, others procrastinate, don't have good health coverage, or are too busy to visit the GP. These can be some of the potential reasons behind the high occurrence of zero values in the distribution. Now speaking of the Zero-Inflated Poisson likelihood function this function is convenient for our story for two main reasons. Firstly, Poisson distribution is a discrete probability distribution and gives the probability of an event happening a certain number of times (k) within a given interval of time or space. This is convenient because our data is (1) discrete count data and (2) is measured within a given interval. This is the PMF of the Poisson function

$$p(x) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Where λ (lambda), the Poisson distribution's only one parameter, is the mean number of events. However, as we saw from the data and its distribution the poisson function won't be enough as its likelihood model and we will need something extra. We will use a mixture model for this which uses more than one simple probability distribution to model a mixture of causes. Our mixture model will be Zero-Inflated Poisson.

Model Analysis: Likelihood and Prior

The process of our data generation is binomial: individuals either visit the doctor or not, but it also has a large number of trials and very low probability, so the distribution tends towards Poisson. In our data, we have two predictor variables: age and region. Both of these variables can impact the outcome since young people tend to be healthier and not visit the doctor as much or visits can also be influenced by region since the available number of GPs varies based on region. Since people tend to only visit doctors when it is an emergency we will implement the probability of observing zero as the probability that an individual isn't sick and didn't go to the GP or the probability that an individual is sick and still didn't visit the GP (procrastinated). After following these assumptions we get this likelihood of observing a zero value.

$$P(0|p,\lambda) = P(\boldsymbol{sick}|p) + P(\neg \boldsymbol{sick}|p) \times P(0|\lambda)$$

The parameters for ZIPoisson distribution are the probability of observing zero values and lambda which is the expected Poisson count for the *i*-th individual. This is the mathematical representation of the mixture model:

$$\Pr(Y = 0) = \pi + (1 - \pi)e^{-\lambda}$$

$$\Pr(Y = y_i) = (1 - \pi)\frac{\lambda^{y_i} e^{-\lambda}}{y_i!}, \qquad y_i = 1, 2, 3, \dots$$

Where y is th outcome variable, λ is the expected poisson count for the i-th individual/observation and π is the probability of extra zeros. So the model takes the form:

$$y_i \sim ZIPoisson(p_i, \lambda)$$
$$logit(p_i) = \alpha_p + \beta_p x_i$$

We have two link functions for each process within ZIPoisson function. One for the probability of observing zeros: p_i and the other for expected poisson count: λ. In PyMC this is how we set the likelihood up:

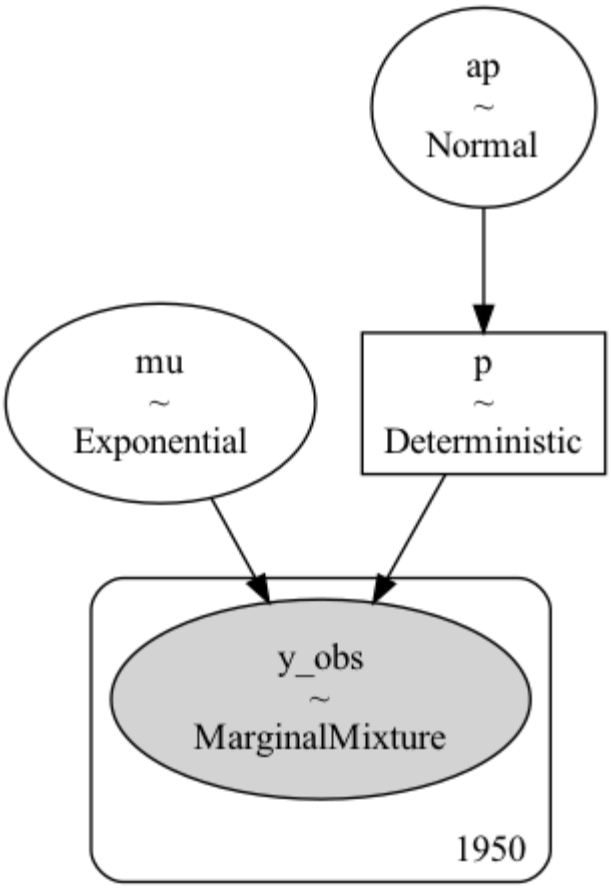


Figure 3. Complete pooling model.

Where ‘mu’ represents the prior for the expected count: λ in Poisson, ‘ap’ represents the prior distribution for ZIPoisson – the distribution from which we will generate zeros, and ‘p’ is a deterministic logistic function responsible for generating zeros with probabilities from ‘ap’ distribution. In terms of distributions, ‘mu’ follows exponential distribution since exponential distribution is the probability distribution modeling elapsed time before events. I chose λ=0.1 since in this prior, I am assuming that the number of GP visits in a year will come out to about $\frac{1}{0.1} = 10$ visits on average. Looking at the data average number of visits is 8 so this assumption is accurate. Next, for ‘ap’ prior which determines the distribution where zeros will come from I selected normal distribution $\mu = 0$ and $\sigma^2 = 1$ allowed for some flexibility in capturing varying degrees of zero inflation which then feeds into the logistic function.

Prior Predictive Distribution

Having discussed all of these now let’s visualize the prior-predictive distribution.

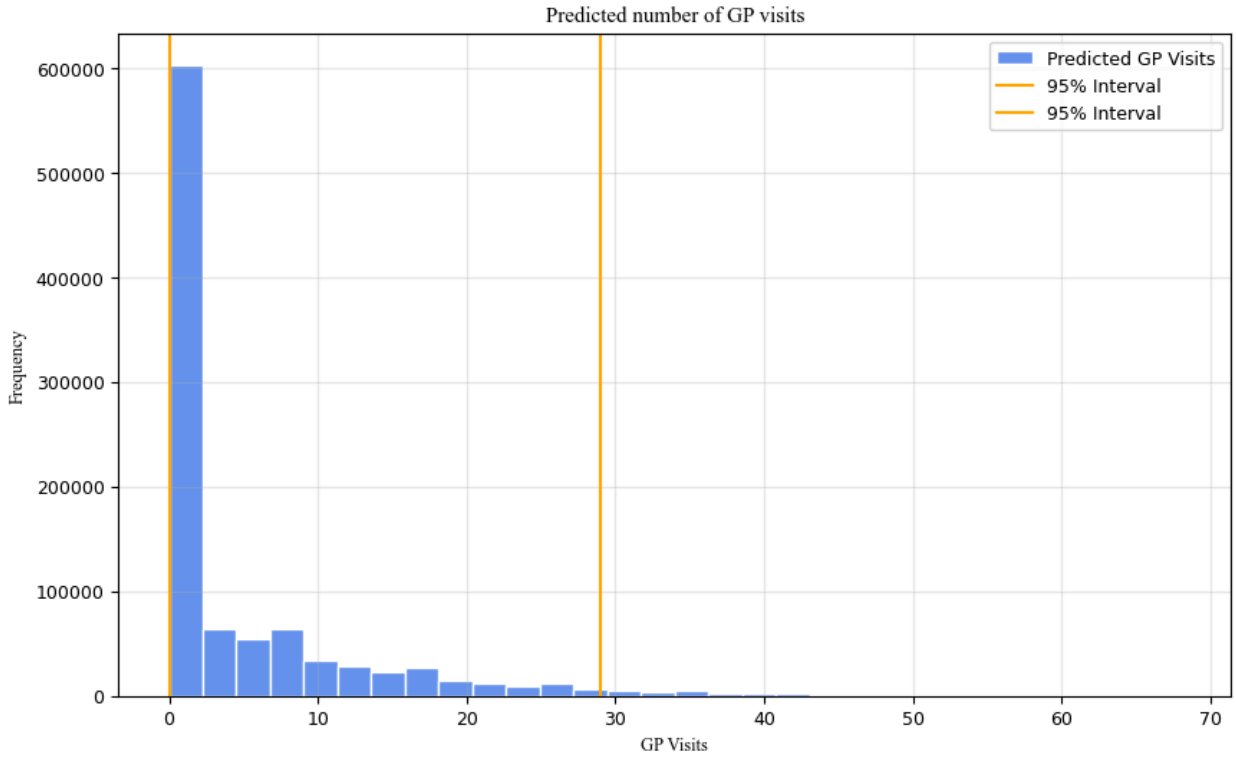


Figure 4. Prior-predictive distribution of complete pooling model.

In this visualization, it is clear that our model works and is reasonable with these summary statistics

Metric	Value
Mean number of GP visits	5
Standard Deviation	8
Mode	0
95% Interval	[0, 29]
Range	0.0 to 68.0

Indicating that our model is indeed accurate and its prior predictive values are plausible and can be used for analysis.

Posterior Predictive Sampler Check

Now let's go ahead with posterior predictive distribution and run the inference in PyMC.

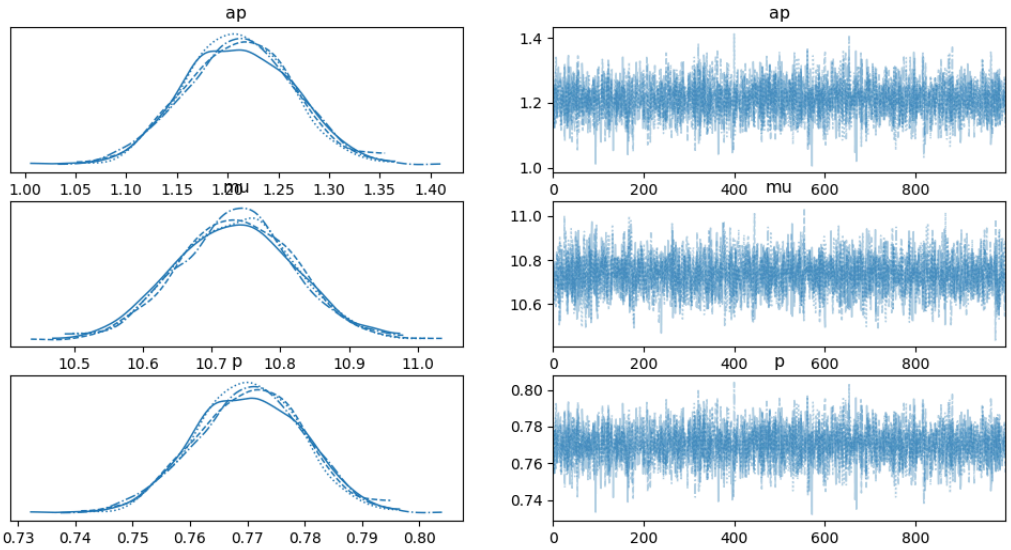


Figure 5. Posterior predictive trace plots.

From observing trace plots we can see that the chains converged, explored the entire posterior, and didn't get stuck. This is an indication that our sampler worked well and can be trusted with prediction. Next let's see the rank plots.

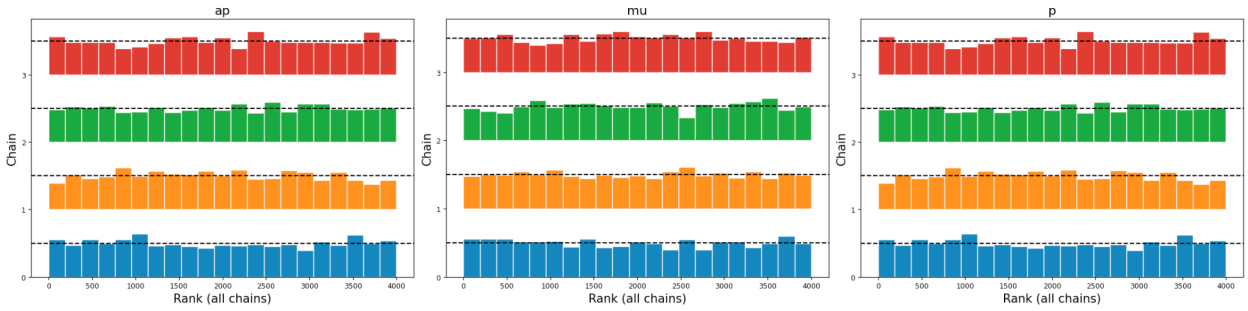


Figure 6. Rank plots.

From the rank plots we can see that all chains are uniformly distributed exploring the entire distribution and are close to the dashed black line. Which is another essential indication that the sampler worked.

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
ap	1.211	0.054	1.114	1.312	0.001	0.001	4090.0	3119.0	1.0
mu	10.735	0.085	10.583	10.897	0.001	0.001	4236.0	2765.0	1.0
p	0.770	0.010	0.753	0.788	0.000	0.000	4090.0	3119.0	1.0

Figure 7. Summary table.

The summary table shows that R_hat=1.0 which is exactly what we wanted. Anything greater than 1.01 would not have been okay. Secondly, the summary table also shows that ESS values are all above 10% of our sampling size. The ESS measures how successful the sampling algorithm was in ensuring that consecutive samples are uncorrelated. Next, it is also evident that MCSE_mean and MCSE_sd are both very low which indicates that there is a low correlation in Markow Chain. Lastly, we have mean values for each of the parameters, their sd and HDI values.

To see how our model performed let's compare actual data with posterior predictive distribution. To do this I will showcase two plots in figures eight and nine. Figure eight shows the posterior predictive distribution of zero counts compared to the actual counts of zeros in the dataset. As it is evident from the visualization the proportion of zeros is almost the exact in both posterior predictive distribution and the actual dataset. I also compared the mean of posterior and actual distribution which comes out to 8 visits per year for both. These two custom metrics indicate that the model predicts data accurately and reliably.

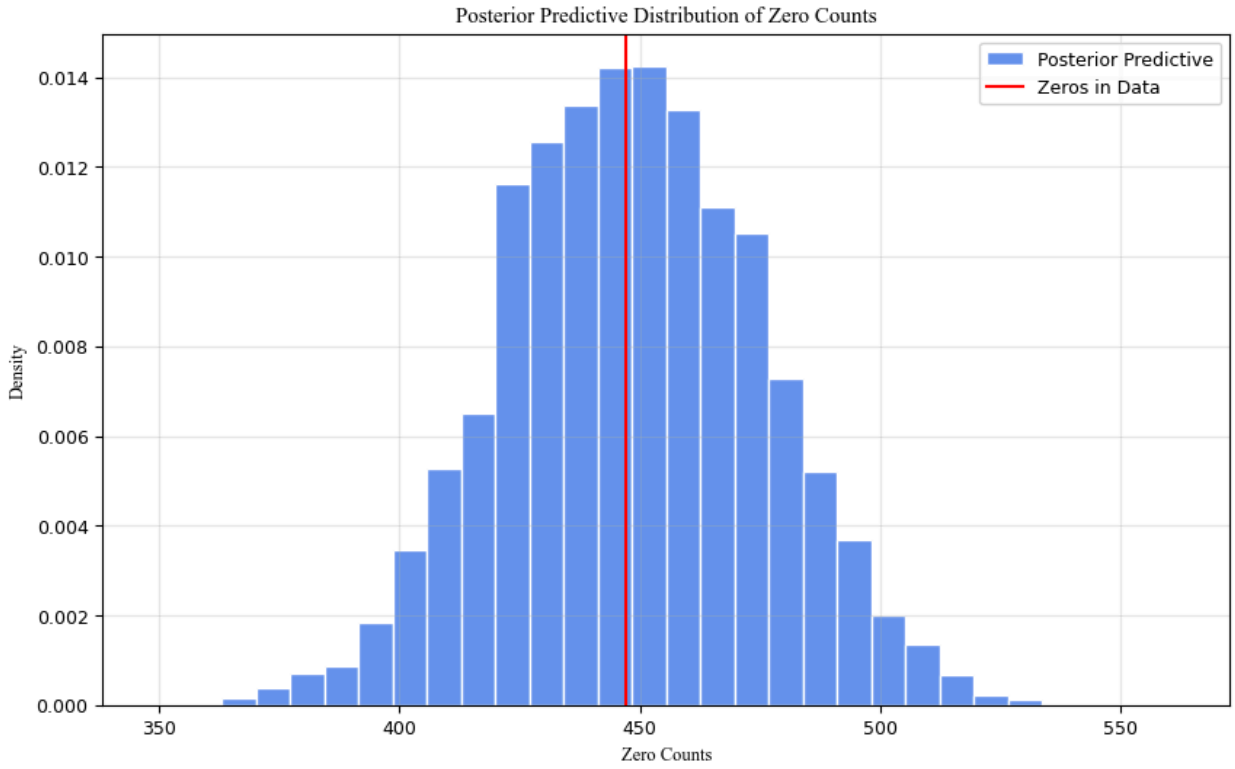


Figure 8. Posterior predictive distribution of zeros in the model and zeros in the actual data.

Now we can look at figure nine which is the full posterior predictive distribution (not only zero counts) its compatibility interval and mean value which is compared to the actual mean. We can see that our model captures the trend of the data accurately and produces a similar distribution with proportional zero inflation, mean number of GP visits, and similar compatibility interval as the actual data.

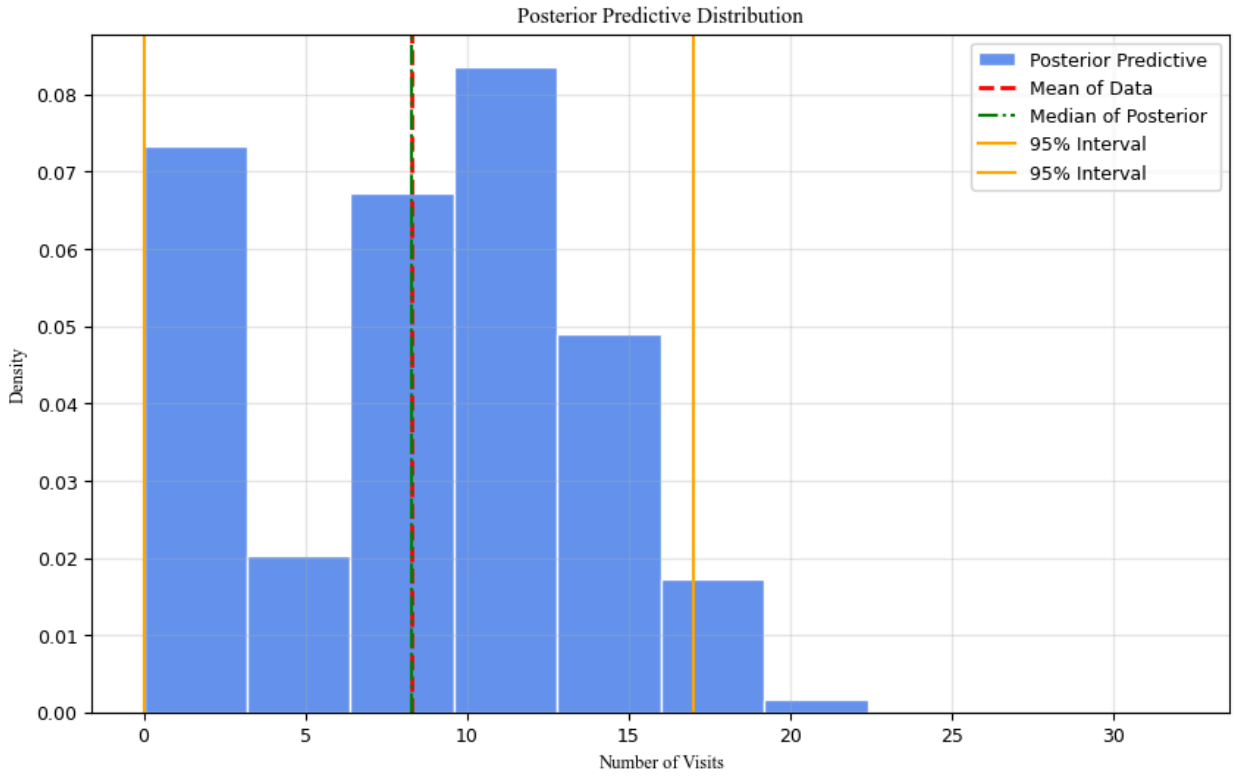


Figure 9. Posterior predictive distribution.

Partial pooling model

Our previous model used a complete pooling approach which ignores the varying intercepts and just uses the overall mean across all clusters. For reference, we have 65 total clusters in our dataset. Each cluster is a unique combination of 'Age' and 'Region' clusters. We have 65 clusters because five of them are NaN values (clusters we are estimating). Now the complete pooling approach pooled data from all the clusters to produce a single estimate that is applied to every cluster assuming that the variation among clusters is zero. This is quite a big assumption since it neglects the potential relationships between age, region, and GP visits. To make this better we can estimate varying intercepts by using partial pooling. Partial pooling is a method that takes information from each cluster and produces estimates for each of them. This model is less likely to underfit than the complete pooling model and less likely to overfit than the no-pooling approach which we don't even consider for this model since its the worst out of the three approaches. Partial pooling tends to better estimate the true per-cluster means. So to set up our model I will use hierarchical modeling and set up hyperparameters and include a shape argument which will account for the 65 clusters I discussed.

Data Pre-processing

Before I show you the model I want to mention that to create 65 clusters I used the pandas groupby() function to create individual index for each cluster by grouping the counts based on Region and Age. By doing this I essentially created a new 'Cluster' column in my pandas dataframe which now looks like this:

Index	Region	Age	Counts	Cluster
0	0	0	4.0	1
1	0	0	11.0	1
2	0	0	14.0	1
3	0	0	13.0	1
4	0	0	12.0	1

Now we can set up a partial pooling model like this:

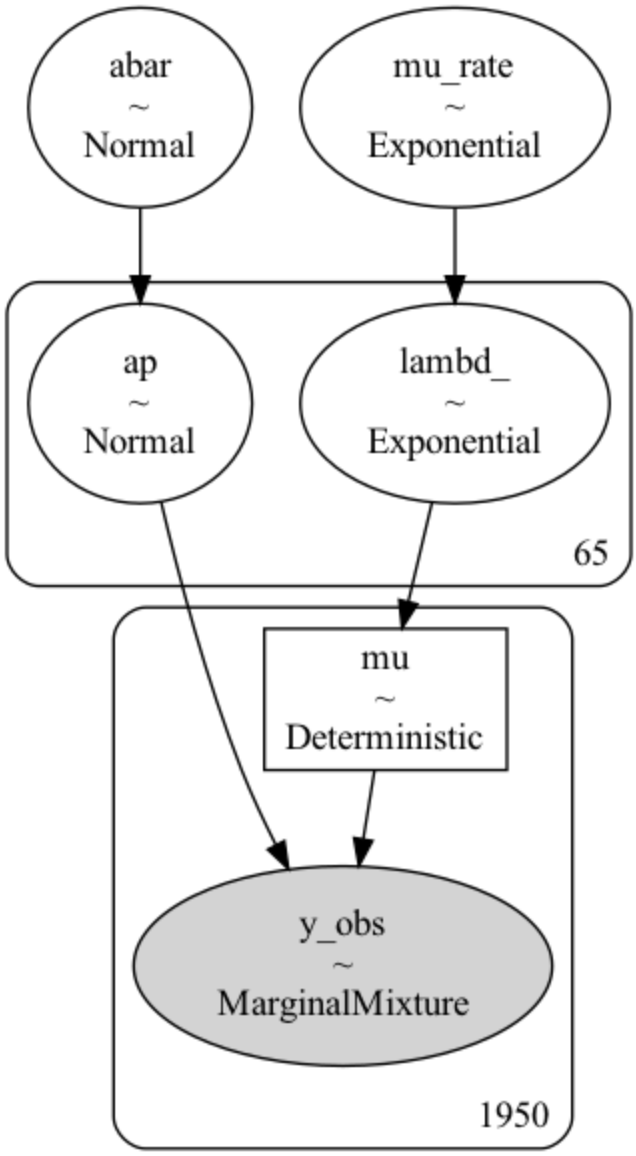


Figure 10. Partial pooling hierarchical mixture model.

As you can this model now has new parameters which are called hyperparameters. Hyperparameters are parameters for parameters and their priors are hyperpriors. I used hyperpriors to embed different populations of clusters within different regions and age groups. Notice that I have three hyperpriors ‘mu_rate,’ ‘lambd_,’ and ‘abar.’ These hyperpriors are priors for ‘mu’ and ‘ap’ distributions that I

discussed above. Hyeprior ‘mu_rate’ follows an exponential distribution $\lambda = 2.5$ which determines the distribution of λ values within ‘lambd_’ hyperprior. ‘Lambd_’ hyperprior itself follows an exponential distribution with a shape of 65 clusters meaning that it creates varying intercepts for the mu function. Justification for exponential distribution again is straightforward: it captures the distribution of events that occur at a constant rate and is a conjugate of the Poisson function in the likelihood distribution. Finally, ‘abar’ is a hyperprior for the probability of zero within the logistic model. ‘Abar’ follows a normal distribution with $\mu = 0$ and $\sigma^2 = 1$ values. These hyperpriors control the variability of the individual cluster rates and I set this variability at a medium level to allow the posterior distribution to learn from the data. Let’s visualize the prior predictive to see that our model works well.

Prior Predictive Check

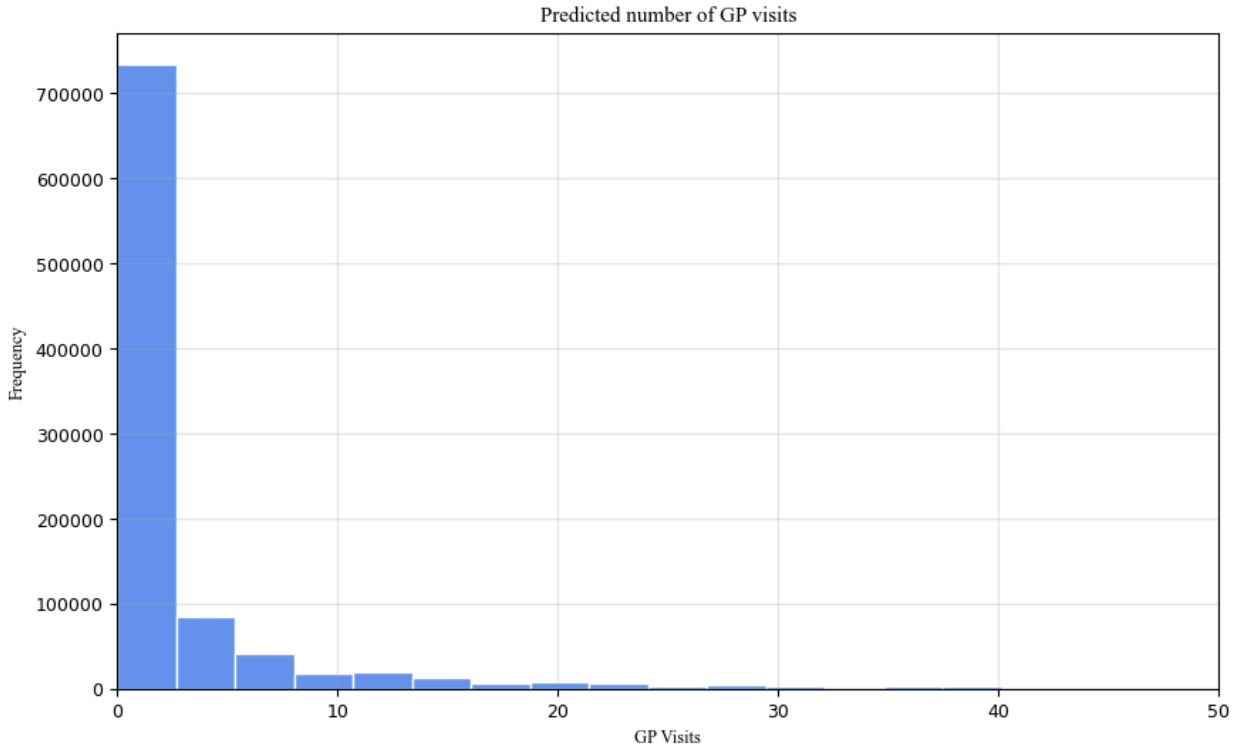


Figure 11. Prior predictive check.

We can see from prior predictive checks that our model has inflation around zero counts and has a similar distribution as the real data. This is an indicator that our model is accurately set up and now we can start sampling the posterior and visualize posterior predictive values.

Posterior Predictive Sampler Check

We can see from the summary table, rank, and trace plots that the sampler has worked well and it sampled parameter values for each cluster resulting in 65 unique parameter values for each cluster. It is also clear that these parameter values have accurately converged, with R_hat values of 1.0, low MCSE values, and ESS values of above 10% of the sample. Observe the plots below to confirm this.

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
abar	1.350	0.136	1.090	1.598	0.002	0.002	3708.0	3198.0	1.0
lambd_[0]	10.392	0.616	9.283	11.539	0.008	0.006	6114.0	3192.0	1.0
lambd_[1]	5.539	0.478	4.640	6.416	0.006	0.004	6697.0	3015.0	1.0
lambd_[2]	2.539	0.384	1.819	3.245	0.006	0.004	4417.0	2945.0	1.0
lambd_[3]	2.815	0.343	2.207	3.491	0.005	0.003	5566.0	3095.0	1.0
...	...	...	...	...	...	...	...	...	...
mu[1945]	27.300	1.015	25.427	29.151	0.013	0.010	5754.0	2726.0	1.0
mu[1946]	27.300	1.015	25.427	29.151	0.013	0.010	5754.0	2726.0	1.0
mu[1947]	27.300	1.015	25.427	29.151	0.013	0.010	5754.0	2726.0	1.0
mu[1948]	27.300	1.015	25.427	29.151	0.013	0.010	5754.0	2726.0	1.0
mu[1949]	27.300	1.015	25.427	29.151	0.013	0.010	5754.0	2726.0	1.0

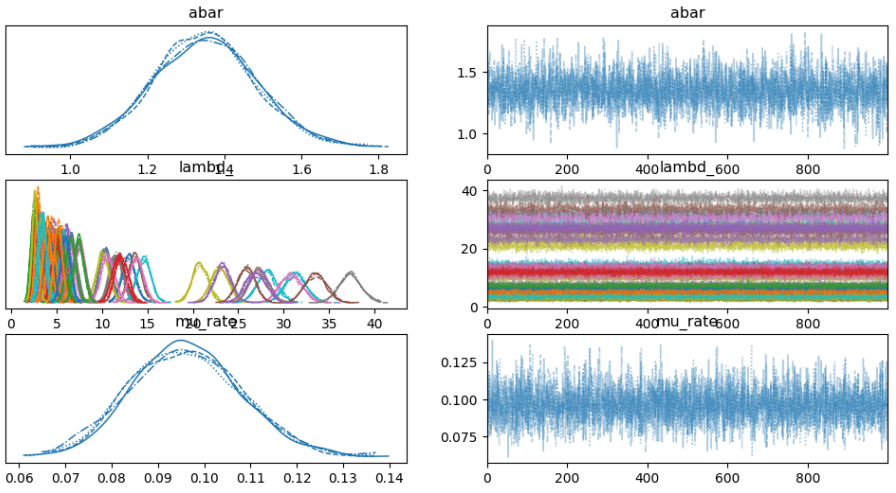


Figure 12. Summary table and trace plots

Figure 13. Rank plot (maximum number of ranks plots displayed – for more modify the jupyter code using var_names=[] argument in az.plot_rank() call).

Now let's create a similar visualization to complete the model above to evaluate how the posterior predictive model performs compared to real data. We can see that our model is quite close to the real observations since the proportion of zeros and the mean number of GP visits are both similar.

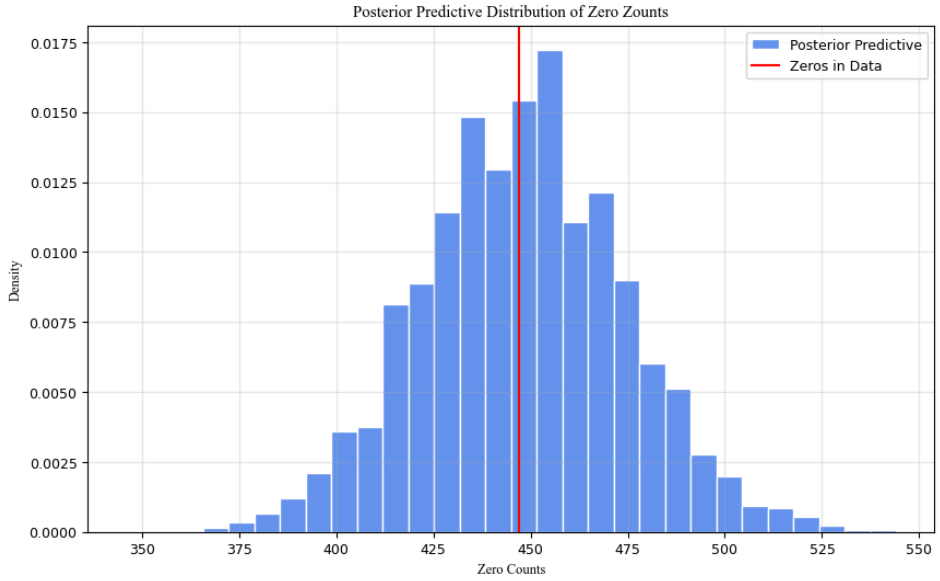


Figure 14. Posterior predictive distribution of zeros in the model and zeros in the actual data.

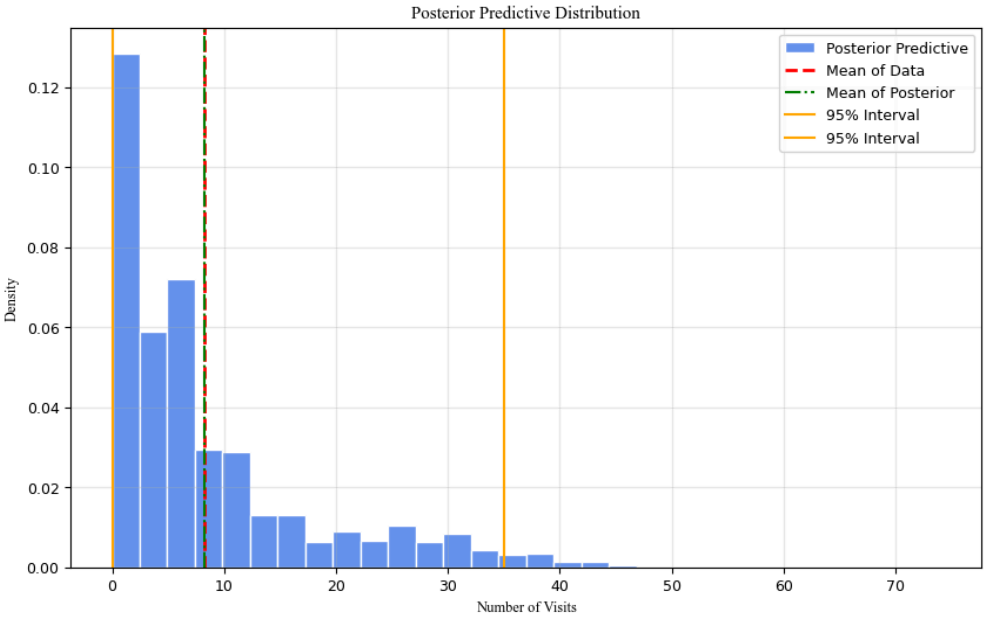


Figure 15. Posterior predictive distribution.

From the posterior predictive distribution, we clearly see that this model is not that different from the previous one. So what is the advantage of the partial pooling method? To observe the advantage of the partial pooling approach let's first take a couple of random samples from both posteriors and plot them on the same axes to see the difference between each model's variability.

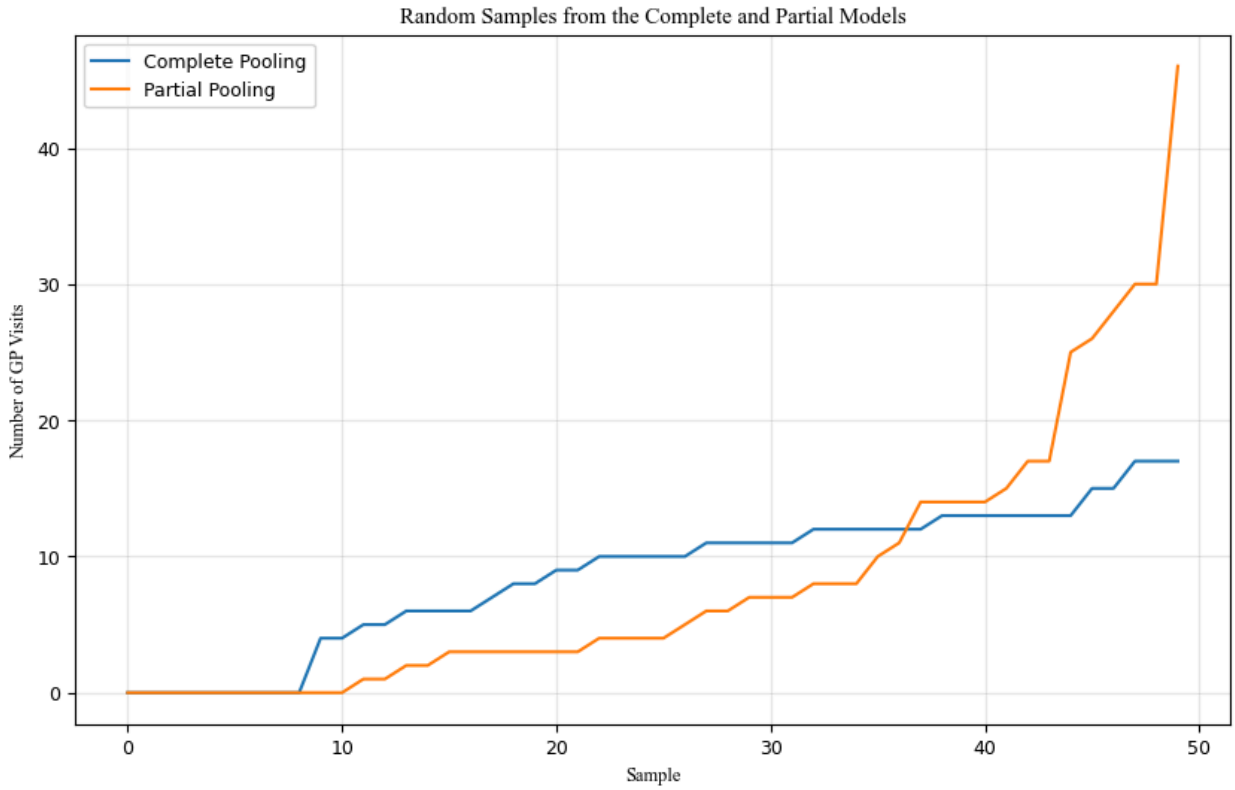


Figure 16. Comparing variance of the random samples from partial and complete pooling models.

From Figure 16 you can see that the partial pooling model explores a much broader range of values than the complete model due to the way partial pooling varies intercepts for each cluster. This is the big difference between the models. While partial pooling considers the ingroup variance in outcomes complete pooling approach fails to do this and creates one mean for all of the clusters. This way the complete pooling approach loses the ability to 'adapt' to each cluster and give accurate estimates. This obviously shows up when we compare models using the leave-one-out cross-validation technique below.

	rank	elpd_loo	p_loo	elpd_diff	weight	se	dse	warning	scale
partial	0	12408.652727	255.490162	0.000000	0.872889	225.792189	0.000000	True	deviance
complete	1	21935.853002	13.184472	9527.200275	0.127111	669.036986	564.893357	False	deviance

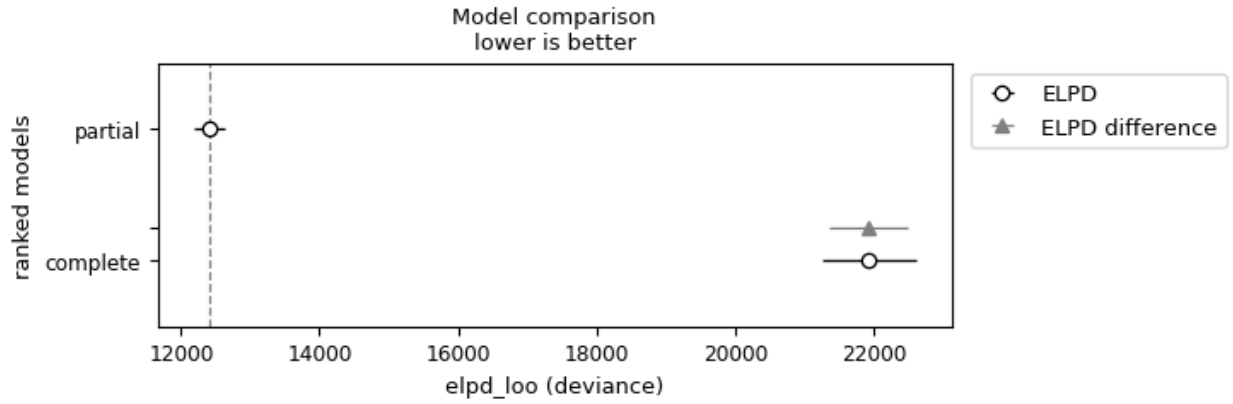


Figure 17. Model comparison.

Since we disaggregated the data at the beginning of our model we can confidently and comfortably use the leave-one-out cross-validation method to compare these two models. From the table, we can see that the partial pooling approach is clearly a favorite since it outperforms the complete pooling model. The weight column shows the probability of the partial model being a better model is about 87% and it ranks higher with lower standard error and lower expected log pointwise predictive density which are indicators that this model performs better with the data. We can also see this from figure 17 which is a visual representation of two models and shows that partial is clearly the better one.

Final Visualization

Finally let's fill in the missing data using both models and see how our first visualization changed.

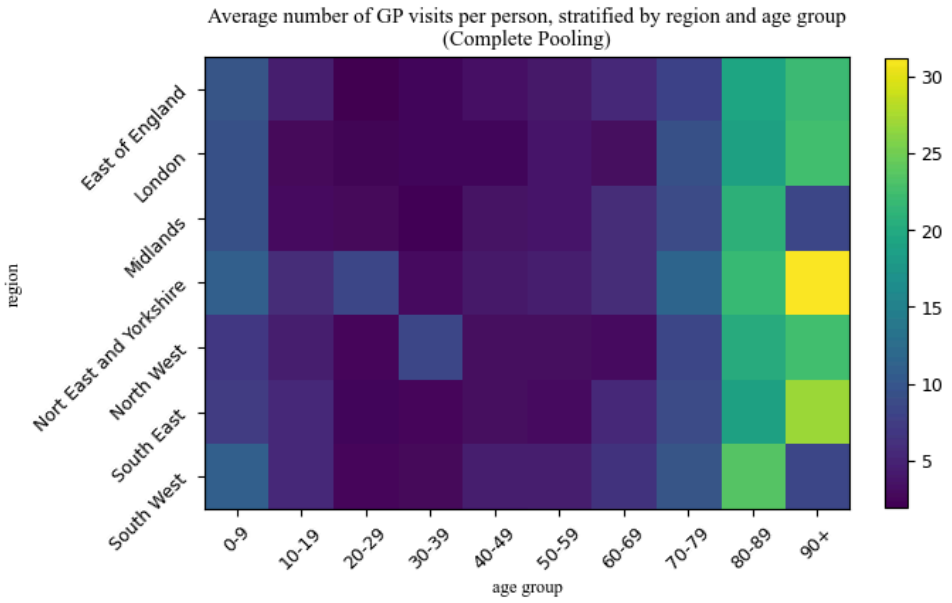


Figure 18. Filled NaNs with complete pooling.

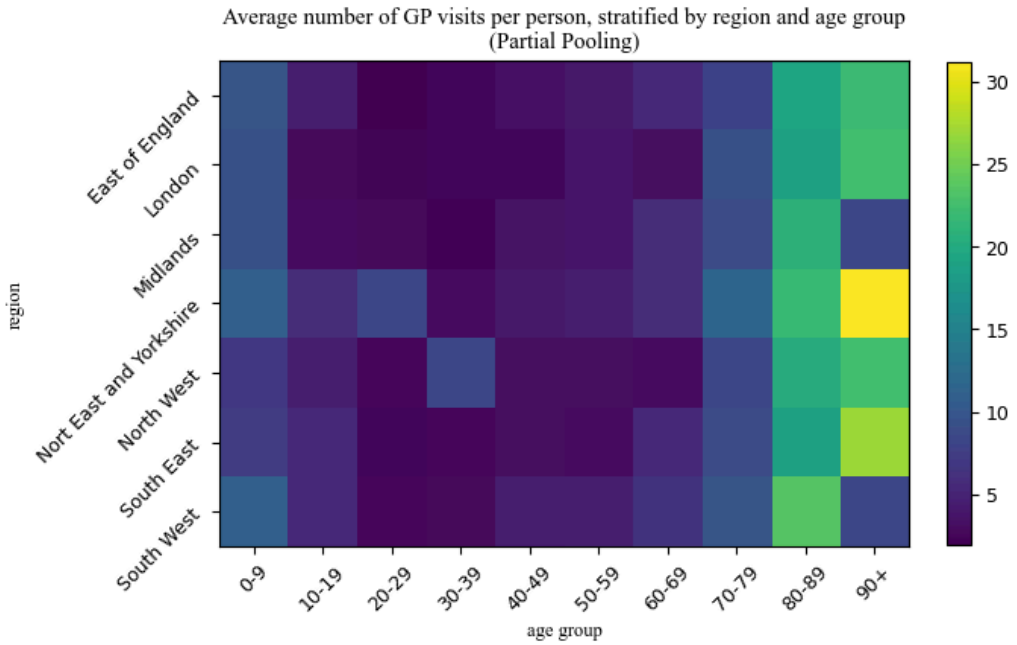


Figure 19. Filled NaNs with partial pooling.

Conclusion

We can see that when filling the NaN values from the posterior predictive distribution we can't see the clear difference between which model performs better and since both models blend well. However, since we already discussed LPPD, compared models, and determined that the partial model was better, I conclude that we should prefer the partial model since it has greater variability to capture the unique characteristics of each cluster and account for cluster variant data. While the complete pooling model overlooks the cluster variability and picks one mean for the entire dataset. Secondly, the partial pooling model has a higher ELPD value which means that it has better predictive accuracy and this is what matters when we are trying to account for missing data since ELPD describes whether the predicted values match the actual values of the target field within the incertitude due to statistical fluctuations and noise in the input data values.

AI Statement

I used Grammarly to check for typos and grammar mistakes. I did not use any other AI resource for the content of this assignment.