Leveraging Conjugacy in Dirichlet Process Poisson Mixture Models

STAT 447 Final Project: Caden Hewlett

Introduction

In this work, we present a non-parametric approach to the Gamma-Poisson (GP) model framework using weekly aggregation of daily crash count data from the Chicago Police Department (CPD 2024). As of writing, there is no out-of-the-box approach for the GP setup in the widely-used `dirichletprocess` package for Dirichlet Process Mixture Models (DPMMs). So, in addition to presenting a formalized Bayesian analysis of the CPD data set, this work develops a bespoke integration of the GP Bayesian Model with the aforementioned `R` package. In addition we provide a more formal analysis of these data considering that the closest investigation of the CPD data publicly available were summary statistics by (Johny 2018).

We will begin with a brief literature review, discussing probability measures and Dirichlet Processes (DPs) from a theoretical standpoint. In the Data Analysis section, we briefly recap the source data aggregation and imputation, inspect the plot of the count data and provide some observational analysis. We will proceed to detail the stick-breaking process used by the algorithm and formally define both the Bayesian model and hyper-parameter choices for the DPMM. In the Results section, we produce the *a posteriori* results which include the table of weighted posterior rates and the corresponding posterior plots. We then briefly interpret these results, discuss shortcomings, and establish ideas for future work.

Literature Review

Before we discuss Dirichlet Processes, we establish some groundwork in measure theory. We will briefly revisit the concepts of σ-algebra and probability measures. In the following, we present generalized definitions which are discussed rigorously in works such as (Billingsley 2012) and (Rudin 1986).

Measure Theory

Let $\mathbb{X}$ be a well-defined sample space. A σ-algebra $\mathcal{F} \subseteq P(\mathbb{X})$ is a set satisfying the following:

1. *Non-emptiness and universality*: The entire sample space $\mathbb{X}$ is in $\mathcal{F}$ and $\emptyset$ is in $\mathcal{F}$.
2. *Closure under complementation*: For all sets $A \in \mathcal{F}$, the complement $A^c \in \mathcal{F}$.
3. *Closure under countable unions*: For any countable collection of sets $\{A_i\}_{i \in I}$, where I is a countable index set, if $\forall i \in I, A_i \in \mathcal{F}$ then $\bigcup_{i \in I} A_i \in \mathcal{F}$.

For the sake of this work, we are more interested in *probability measures*, which are built on σ-algebra. A probability measure $\mu : \mathcal{F} \mapsto [0, 1]$ satisfies the following familiar axioms of probability:

1. *Non-negativity*: $\forall A \in \mathcal{F},\ \mu(A) \geq 0$. This is often extended to include the fact that $0 \leq \mu(A) \leq 1$.
2. *Unit measure*: For the sample space $\mathbb{X}$, $\mu(\mathbb{X}) = 1$, and $\mu(\emptyset) = 0$.
3. *Countable additivity*: For any countable set $\{A_i\} \subseteq \mathcal{F}$ where $\forall i \neq j, A_i \cap A_j = \emptyset$, $\mu\Big(\bigcup_{i=1}^{\infty} A_i\Big) = \sum_{i=1}^{\infty} \mu(A_i)$.

Before we move on to Dirichlet Processes, we acknowledge that the above definitions might be abstract for those new to measure theory. For clarity, the Appendix includes a finite example to illustrate σ-algebra properties and verify a simple probability measure μ.

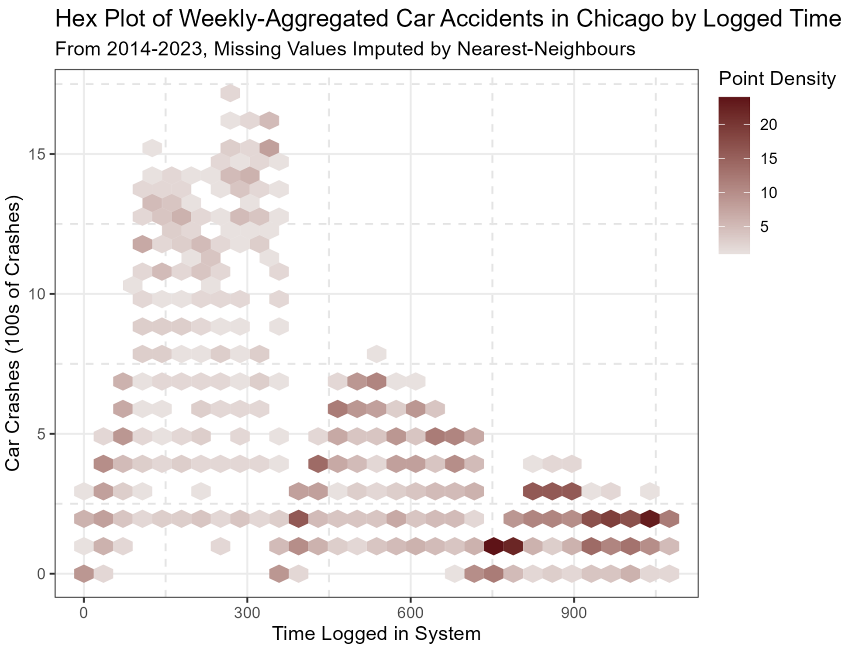
Dirichlett Process: A Distribution Over Measures

Now, we introduce the concept of the Dirichlet Process (DP), a distribution over probability measures. We adopt the nomenclature used in (Hannah 2011), where the DP is specified by its base measure $\mathbb{G}_0$ (commonly a distribution) and the concentration parameter, α. A random sample from a Dirichlet Process is a probability measure over a σ-algebra from sample space $\mathbb{X}$, which is often assumed to be countable. The resulting probability measure follows the structure of the base measure $\mathbb{G}_0$ with a degree of deviation controlled by α. A Dirichlet Process can also be considered conceptually as a Dirichlet Distribution whose support is an infinite-simplex, rather than a discretely-defined k-simplex (Ferguson 1973).

The α parameter can be thought of as the confidence in the base measure, where a higher α is indicative of less confidence and a greater dispersion about $\mathbb{G}_0$. Conversely, a lower α indicates more certainty in $\mathbb{G}_0$ resulting in fewer, larger clusters about the base measure (Sethuraman 1994). This property is explored in more detail in the discussion of finite approximation in the Methods section.

Data Analysis and Processing

Now, we will perform some exploratory data analysis and explain the data set used. The data is sourced from the Chicago Police Department (CPD 2024) and contains $794,956$ vehicle accident reports from 2014 to 2023. A hexplot of the aggregated results is below.



We applied an aggregation code framework modified from (StackOverflow 2023) in order to coerce the data into weekly crash counts over the time frame. Certain weeks in both 2014 and 2015 had missing counts. In these cases, we performed nearest-neighbours imputation for the missing values. In addition, we grouped the counts by severity to craft an environment in which reports were logged by monetary crash damage, with low-severity accidents ordered first. Furthermore, in an attempt to avoid overflow in training loops we measured crashes in hundreds. As the hexplot shows, there are three distinct categorizations of crash counts, with the count dispersion decreasing across the groups. In addition, as the plot shows, the point densities vary across these clusters. These characteristics of the aggregated data create a distinct multimodal plot, which makes these data a good candidate for Bayesian non-parametrics. The code for data processing, imputation and plotting are included in the appendix.

Methods

In this section, we define the stick-breaking discrete approximation of the Dirichlet Process implemented in this work and in the Gibbs sampler of the `dirichletprocess` package (Markwick 2023). This library implements the algorithm discussed in-depth in (Neal 2000). As Markwick notes in his work, using a conjugate base prior allows the algorithm to use an optimized method presented by Neal, and hence is generally preferable if fast mixing is desired. However, as of writing, the Gamma-Poisson conjugate pair is not implemented as a default model in the package. Hence, in addition to the Dirichlet Process finite approximation, we explicitly define the likelihood, conjugate posterior and posterior predictive for the Gamma-Poisson to be implemented as a mixing distribution in the DPMM Gibbs sampler. Further, we explicitly state the Bayesian model applied to the data from the previous section.

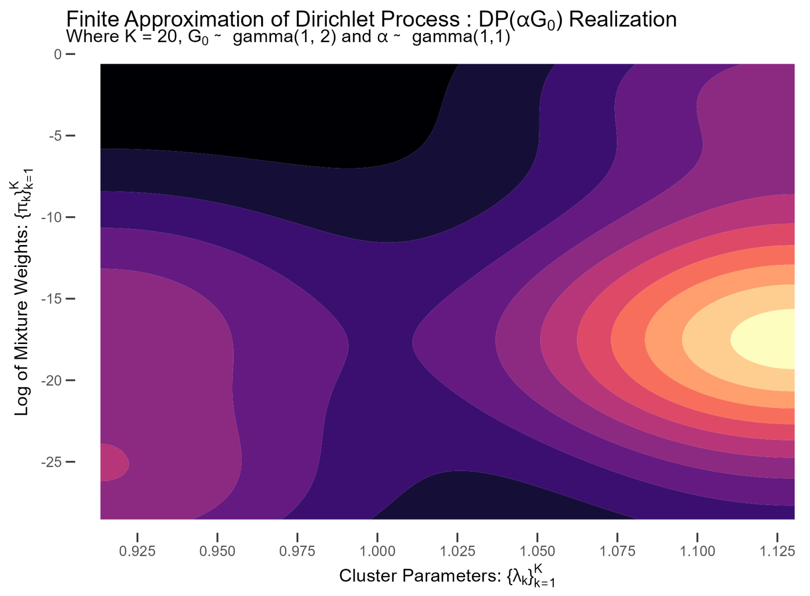
Finite Approximation

The finite approximation used in this work is known as a "stick-breaking" or Griffiths, Engen, and McCloskey (GEM) process. The purpose of the GEM process in terms of a Dirichlet Process is to generate weights $\{\pi_k\}$, which will be assigned to pulls from $\mathbb{G}_0$ to approximate a sampled measure from $\mathrm{DP}(\alpha\mathbb{G}_0)$.

The idea behind GEM weighing is to take a "stick" with unit length and break it at a location determined by the realization of a $\beta_1 \sim \mathrm{Beta}(1,\alpha)$ random variable, which we denote π_1. Then, we break the remaining stick length in two by a second $\beta_2 \sim \mathrm{Beta}(1,\alpha)$ sample. Hence, $\pi_2 = (1-\beta_1)\beta_2$, which can be understood as "the remaining stick length after the first break, broken at the second random break location." Then, we generalize this concept for discrete $k = \{1, 2, \dots, K\} \subseteq \mathbb{Z}$ as follows:

$$\pi \sim \mathrm{GEM}(\alpha) : \text{Let } \beta_k \overset{\text{iid}}{\sim} \mathrm{Beta}(1,\alpha), \text{ then } \pi_k = \beta_k \prod_{i=1}^{k-1}(1-\beta_i)$$

Where π_k can be considered the k-th value returned from the GEM process. The realization is then an estimation of a K-dimensional probability measure. For computational purposes, we treat $\mathrm{GEM}(\alpha)$ as a discrete probability distribution for a reasonably large choice of K, since $\lim_{K\to\infty}\left(\sum_{k=1}^{K}\pi_k\right) = 1$ which we prove in the Appendix. Further, the residual distance of the sum from 1 decreases exponentially, as noted in (Xing 2014) and other works.



As mentioned in the literature review, α is the concentration of the finite-approximated Dirichlet Process. With the additional context of the GEM distribution this property becomes more evident. For an arbitrary $\beta_k \sim \text{Beta}(1, \alpha)$, a larger α assigns more probability density to lower values in $\text{supp}(\beta_k) = (0, 1)$. This means that the breaks are more likely to happen "earlier on" along each stick, yielding smaller initial clusters and hence more dispersion in the density of pulls from $\mathbb{G}_0$. To contrast, a very low α may result in a β_1 break near 1, implying very low weights for the remaining $\{\beta_k\}_{k=1}^{K-1}$. For the model we apply in this work, we let $\alpha \sim \text{Gamma}(1, 1)$ and $\mathbb{G}_0 \sim \text{Gamma}(1, 2)$ define $\text{DP}(\alpha\mathbb{G}_0)$. In the figure above, we demonstrate a finite approximation of $\text{DP}(\alpha\mathbb{G}_0)$ at $K = 20$, where the vertical axis corresponds to the logarithm of stick-breaking weights $\{\pi_k\}_{k=1}^K$ and the horizontal axis denote the sampled values $\{\lambda_k\}_{k=1}^K$ from $\mathbb{G}_0$. The kernel density bin width for contour separation is 0.02, starting from $(0, 0.02]$ and ending at $(0.22, 0.24]$. The full code for the approximation and density plot is included in the appendix. For the full implementation of the DPMM we selected $K = 150$, a choice justified in (Ishwaran and James 2001).

Model Implementation

We implemented the Dirichlet Process Mixture Model by deriving an explicit expression for the mixing distribution. This custom implementation was necessary, since the standard Poisson-Gamma conjugate pair is not supported by default in the package for Gibbs sampling we utilized. This adaptation was accomplished through a set of four functions, which customize the sampling procedure. The first function, denoted F1, specifies the likelihood as Poisson. The second function, F2, generates n random draws from the Gamma base distribution $\mathbb{G}_0$ which are mandatory for the stick-breaking definition of the Dirichlet Process discussed in the previous section.

In addition, we define a function F3 which enables posterior updates for the Poisson rate parameter λ. This function simulates n random draws from the Gamma posterior distribution using the observations $\{y_i\}_{i=1}^n$, where N is the sample size. Letting α and β be the parameters of $\mathbb{G}_0$, the posterior distribution of λ is given by:

$$\lambda \mid \{y_i\}_{i=1}^N \sim \text{Gamma}(\alpha + \sum_{i=1}^N y_i, \beta + N) \tag{F3}$$

The complete derivation of the distribution above is in the Appendix. Finally, we derived a function F4 for the Posterior Predictive distribution given observation y_n. Letting y_{n+1} be the new observation, the predictive distribution is given as:

$$p(y_{n+1} \mid y_n) = \frac{(\beta+1)^{\alpha+y_n}\Gamma(\alpha + y_n + y_{n+1})}{\Gamma(\alpha + y_n) y_{n+1}! (\beta+2)^{\alpha+y_n+y_{n+1}}} \tag{F4}$$

Here, Γ indicates the gamma function. In the DPMM, the predictive is crucial for calculating the probability of the data being from the prior, as noted in (Markwick 2023). The complete mathematical derivation of this expression is included in the Appendix.

With this framework in place, we can define the theoretical infinite Poisson mixture model. Let π_k be the GEM weights, λ_k be the k-th mixture rate and $\{y_i\}_{i=1}^N$ be observed crash counts, in hundreds.

$$\begin{aligned} \{\pi_k\}_{k=1}^\infty &\sim \text{GEM}(\alpha) \\ \{\lambda_k\}_{k=1}^\infty &\sim \mathbb{G}_0 \\ y_i \mid \{\pi_k\}_{k=1}^\infty, \{\lambda_k\}_{k=1}^\infty &= \sum_{k=1}^\infty \pi_k \, \text{Poisson}(\lambda_k), \text{ for } i = 1, 2, \dots, N \end{aligned}$$

However, the model description provided above is purely theoretical; in practice, it is impossible to construct a mixture model with infinite components. Thus, we implement a finite approximation of the model. We let π_k, λ_k and K be defined as earlier and y_i be the observed crash counts up to $N = 1,076$. We explicitly define the distributions of α and $\mathbb{G}_0$ describing $\text{DP}(\alpha\mathbb{G}_0)$. The Bayesian model and graphical representation are below:

$$\begin{aligned}
\alpha &\sim \text{Gamma}(1,1) \\
\{\lambda_k\}_{k=1}^K &\sim \mathbb{G}_0 \\
\{\pi_k\}_{k=1}^K \mid \alpha &\sim \text{GEM}(\alpha) \\
z_i \mid \{\pi_k\}_{k=1}^K &\sim \text{Categorical}(\{1,2,\dots,K\},\{\pi_k\}_{k=1}^K) \\
y_i \mid z_i, \{\lambda_k\}_{k=1}^K &\sim \text{Poisson}(\lambda_{z_i}),\ \text{for } i = 1,2,\dots,N
\end{aligned}$$

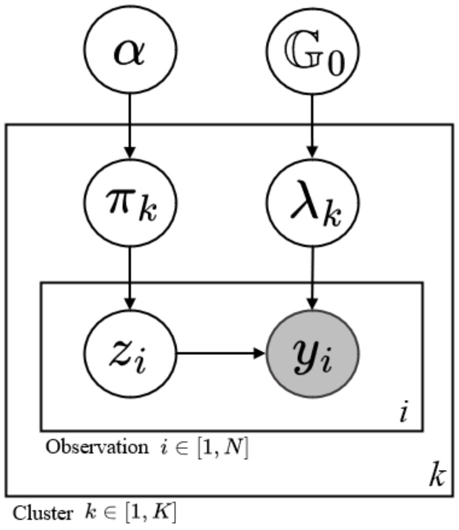
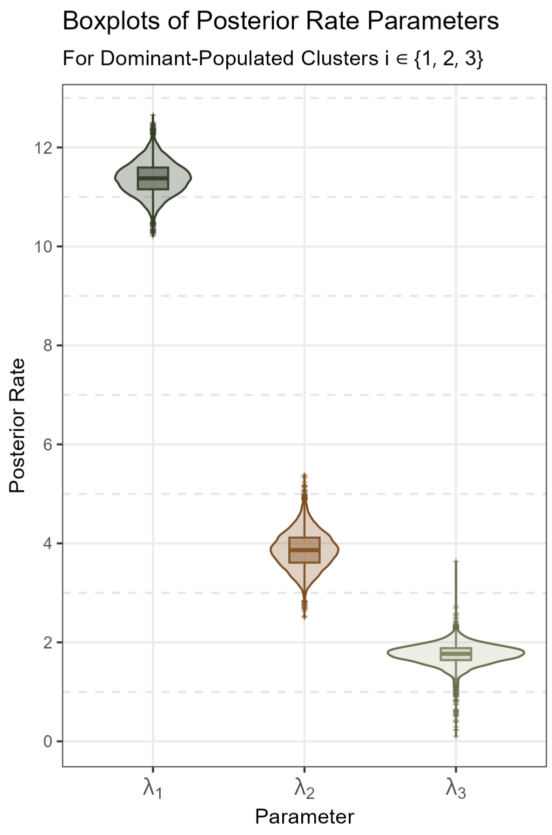


Figure 1: Graphical Model

In the model described in the equations above and in Figure 1, the base measure $\mathbb{G}_0$ is Gamma-distributed with shape 1 and rate 2. With this DPMM structure, z_i denotes the i-th cluster where the probability of selecting cluster $k \in [1, K]$ is determined by the stick-breaking weights $\{\pi_k\}_{k=1}^K$. As a consequence, the Poisson Likelihood is determined by the rate parameter corresponding to the k-th randomly sampled weight index. This indexing procedure is enabled by the use of a categorical distribution to select clusters. While the categorical distribution could select directly from λ_k, the cluster-based approach allows the algorithm to utilize information on the assignments z_i, which can be applied in nonparametric clustering methods such as those discussed in (Zhang, 2019).

Results

The Gibbs sampling procedure on 10,000 iterations with a burn-in of 100 completed in 703.87 seconds. Across multiple trials and randomization seeds, the procedure converged to similar clusters and posterior rates, supporting the consistency and stability of the chain.



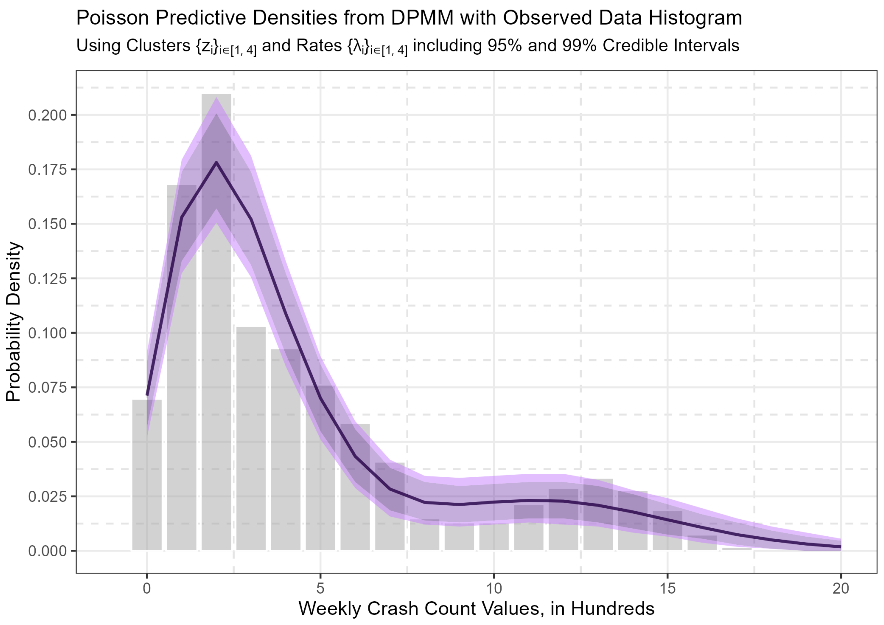
We tested the model on varying sets of synthetic data of size N, and observed consistent recovery of the true clustering and approximate rates of the data. In addition, we observed similarity in convergence times between the synthetic and actual data, suggesting the chain is mixing well on the real-world dataset. In the table below, we report the posterior rate estimates and the associated mixing weights for the Chicago Police data set, where columns $k \in \{1, 2, 3, 4\}$ indicate the cluster.

Table 1: Posterior Parameters and Weights

Rate λ_k	11.964	3.880	2.034	0.718
Weight π_k	0.198	0.329	0.466	0.007
Cluster Size z_k	213	354	501	8

From the table above, we see that the majority of the weight is assigned to three distinct rate parameters. The estimates for these parameters are notably different from another, as shown in the boxplots to the left. This implies that the DPMM is correctly identifying the varying rates in the source data; specifically, it is locating the three groupings of the crash counts we discussed in the Data Analysis section. It should be noted that the fourth rate parameter with $\lambda_4 \approx 0.718$ has a very small cluster size, which indicates minimal contribution to the overall mixture model. In addition, we found the average cluster size to be approximately 4.6 and the proportion of clusters of size 3 improved to around 35% by the final 500 iterations from 21% in the first 500 post burn-in. In all, the DPMM seems to performing well with respect to rate identification of the intrinsically clustered count data.

To further assess the model's ability to capture the characteristics of the true data, we simulated $10,000$ draws of size N from the posterior predictive distribution. These samples used the posterior rates $\{\lambda_i\}_{i=1}^4$ and corresponding weights $\{\pi_i\}_{i=1}^4$, as reported in Table 1. We compared these simulations to the original data by tabulating the frequencies of simulated counts and plotting them alongside a histogram of the observed crash counts. The resulting plot, shown to the right, suggests that the model captures the general shape of the true data well. It correctly identifies the varying density and heavier tails seen in the histogram. While the true values seem to occasionally fall beyond the 95% Credible Interval for the posterior predictive, they consistently remain within the 99% Interval save for one observed bin. In addition, the interval width seems to grow near areas where the densities shift in the source data; this makes sense, as we would expect higher variability along the boundaries of clusters. From this visual evaluation of the posterior, it seems unlikely that DPMM is mis-specified; the similar shape of the posterior predictive to the true observed counts and the consistent identification of distinct rates suggest that our implementation of the non-parametric Poisson mixture is sound.



Discussion and Further Work

While the posterior results seemed generally favourable, the methodology applied here is not without its shortcomings. Despite the fact that the proportion of clusters of size 3 increased as we ran the chain, the relatively high overall mean cluster size of 4.6, notably different from the clustering seen in the source data, suggests potential problems in cluster assignments. The boxplot in the results section revealed that the rates of the second and third clusters are relatively close, a claim that is supported by the overlap seen in the hexplot reported in the Data Analysis section. The closeness of these groups is a potential culprit for the larger than expected number of clusters, as the algorithm may favour group assignments which overfit the number of distinct rates. This could theoretically incentivize more lightly-weighted clusters with slightly different rates rather than a small number of heavily weighted assignments. However, such a hypothesis would need to be formally tested. Regardless, these findings bring into question whether or not the model would have performed better on a different set of data. To that end, it is tough to say whether a purely non-parametric approach is necessary in this setting; in situations where assumptions on the prior are rather lax, a more simple hierarchical Bayesian model can often suffice instead of a rigorous non-parametric fit.

With all of this in mind, the implementation of the Dirichlet Process Poisson Mixture Model using a customized `R` function with existing package framework was still an overall success. The ability to add customized models to the `dirichletprocess` package opens the door for novel `R`-based software for more complex Bayesian non-parametrics such as DP-GLMs. In addition, the process used here could be expanded to facilitate a potential CRAN-based extension to the existing package by developing more out-of-the-box options for DPMMs. A direct extension to this work would be to implement another conjugate pair such as a Dirichlet-distributed base measure with a Categorical likelihood. Such a model would have interesting applications in classification problems, as the non-parametric framework allows for an arbitrary number of categories.

Appendices

The appendices are divided into three sections: Proofs and Examples, `R` Code, and Large Images of Figures. These groupings are by importance. While the entire appendix is not strictly necessary, the proofs in Appendix 1 and the first code cell in Appendix 2 showing the custom mixing distribution are recommended to read.

Appendix 1: Proofs and Examples

Proofs are in order of reference, with examples being included afterwards. Miscellaneous information such as knowledge on sets, power sets, subsets and countability from (Demirbas and Rechnitzer 2023).

Proof of Convergence of Summation of GEM Process to 1

We recall the definition of the GEM process, as discussed in the Methods section.

$$\pi \sim \text{GEM}(\alpha) : \text{Let } \beta_k \overset{\text{iid}}{\sim} \text{Beta}(1,\alpha), \text{ then } \pi_k = \beta_k \prod_{i=1}^{k-1}(1-\beta_i)$$

We wish to show the following:

$$\lim_{K\to\infty}\left(\sum_{k=1}^{K}\pi_k\right) = \sum_{k=1}^{\infty}\pi_k = 1$$

We will achieve this by showing that the expected value of the sum converges to 1. We will evaluate the expectation of the sum directly, using properties of expectation, independence, the beta distribution and geometric series.

$$\begin{aligned}
\mathbb{E}\left(\sum_{k=1}^{\infty}\pi_k\right) &= \mathbb{E}\left(\sum_{k=1}^{\infty}\left(\beta_k\prod_{i=1}^{k-1}(1-\beta_i)\right)\right)\\
\mathbb{E}\left(\sum_{k=1}^{\infty}\pi_k\right) &= \sum_{k=1}^{\infty}\left(\mathbb{E}(\beta_k)\prod_{i=1}^{k-1}\left(1-\mathbb{E}(\beta_i)\right)\right), \text{ by independence and linearity}\\
\mathbb{E}\left(\sum_{k=1}^{\infty}\pi_k\right) &= \sum_{k=1}^{\infty}\left(\left(\frac{1}{1+\alpha}\right)\prod_{i=1}^{k-1}\left(1-\frac{1}{1+\alpha}\right)\right), \text{ since } \beta_k \overset{\text{iid}}{\sim} \text{Beta}(1,\alpha)\\
\mathbb{E}\left(\sum_{k=1}^{\infty}\pi_k\right) &= \left(\frac{1}{1+\alpha}\right)\sum_{k=1}^{\infty}\left(\prod_{i=1}^{k-1}\left(\frac{\alpha}{1+\alpha}\right)\right)\\
\mathbb{E}\left(\sum_{k=1}^{\infty}\pi_k\right) &= \left(\frac{1}{1+\alpha}\right)\sum_{k=1}^{\infty}\left(\frac{\alpha}{1+\alpha}\right)^{k-1}, \text{ by independence wrt } i\\
\mathbb{E}\left(\sum_{k=1}^{\infty}\pi_k\right) &= \left(\frac{1}{1+\alpha}\right)\sum_{k=0}^{\infty}\left(\frac{\alpha}{1+\alpha}\right)^{k}, \text{ by index shifting}\\
\mathbb{E}\left(\sum_{k=1}^{\infty}\pi_k\right) &= \left(\frac{1}{1+\alpha}\right)\left(\frac{1}{1-\frac{\alpha}{1+\alpha}}\right), \text{ since } \left|\frac{\alpha}{1+\alpha}\right| < 1\\
\mathbb{E}\left(\sum_{k=1}^{\infty}\pi_k\right) &= \left(\frac{1}{1+\alpha}\right)\left(\frac{1}{\frac{1}{1+\alpha}}\right)\\
\mathbb{E}\left(\sum_{k=1}^{\infty}\pi_k\right) &= \left(\frac{1}{1+\alpha}\right)\left(\frac{1+\alpha}{1}\right) = 1, \text{ as required.} \quad \square
\end{aligned}$$

Using the process outlined above, it is direct to show that $\mathbb{E}(\pi_k^2) \to 1$, implying the asymptotic consistency $\text{var}(\pi_k) \overset{n\to\infty}{\longrightarrow} 0$ and thus the absolute convergence to 1.

Proof of Gamma-Poisson Conjugacy : Posterior

We noted the posterior in the Methods section, and left the full derivation of the conjugate pair for the Appendix here. As mentioned in the Methods section, the conjugate pair is helpful for Gibbs samplers where the nonparametric Dirichlet Process is centered about a Gamma base measure.

Let $\lambda \sim \text{Gamma}(\alpha, \beta)$ be the prior on the Poisson rate parameter λ. Let $x_i \mid \lambda \sim \text{Poisson}(\lambda)$ for $i = 1, 2, \dots, n$. We derive the expression for the conjugate prior, beginning with the proportionality:

$$\begin{aligned}
&\text{Posterior} \propto \text{Prior} \ \times \ \text{Likelihood} \\
&\text{Posterior} \propto p_{\text{gam}}(\lambda; \alpha, \beta) \times \prod_{i=1}^{n} p_{\text{pois}}(x_i\,; \lambda) \\
&\text{Posterior} \propto p_{\text{gam}}(\lambda; \alpha, \beta) \times \prod_{i=1}^{n} \Big(\frac{e^{-\lambda}\lambda^{x_i}}{x_i!}\Big) \\
&\text{Posterior} \propto p_{\text{gam}}(\lambda; \alpha, \beta) \times \prod_{i=1}^{n}(e^{-\lambda}) \cdot \prod_{i=1}^{n}(\lambda^{x_i}) \cdot \prod (x_i!)^{-1} \\
&\text{Posterior} \propto \Big(\frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}\Big) \times (e^{-n\lambda}) \cdot (\lambda^{\sum x_i}) \cdot \prod_{i=1}^{n}(x_i!)^{-1} \\
&\text{Posterior} \propto \underbrace{\Big(\frac{\beta^\alpha}{\Gamma(\alpha)\prod_{i=1}^{n} x_i!}\Big)}_{\text{constant wrt } \lambda} \cdot (e^{-n\lambda}e^{-\beta\lambda}) \cdot (\lambda^{\sum x_i}\lambda^{\alpha-1}) \\
&\text{Posterior} \propto (e^{-n\lambda-\beta\lambda}) \cdot (\lambda^{\sum x_i + \alpha - 1}) \\
&\text{Posterior} \propto (e^{-\lambda(n+\beta)}) \cdot (\lambda^{\sum x_i + \alpha - 1}) \\
&\text{Posterior} \propto \text{Gamma}(\alpha + \sum_{i=1}^{n} x_i, \beta + n)
\end{aligned}$$

The above gives the Posterior Distribution used in Part 3 of the DPMM Mixing Distribution definition, as required.

Proof of Gamma-Poisson Conjugacy : Posterior Predictive

As was discussed in the main work, the posterior predictive is also needed to use the sampler. Since we need the full expression (not a proportionality), we utilize line 5 from the previous proof for a single observation.

For a single observation, we have the following from the gamma distribution using x_n and 1 rather than n and the observation sum.

$$p(\lambda \mid x_n) = \Big(\frac{(\beta+1)^{\alpha+x_n}}{\Gamma(\alpha+x_n)}\lambda^{\alpha+x_n-1}e^{-(\beta+1)\lambda}\Big)$$

Then, we compute $P(x_{n+1})$ by marginalization to arrive at the predictive distribution.

$$\begin{aligned}
p(x_{n+1} \mid x_n) &= \int_0^\infty p(x_{n+1} \mid \lambda)p(\lambda \mid x_n)\mathrm{d}\lambda \\
p(x_{n+1} \mid x_n) &= \int_0^\infty p(x_{n+1} \mid \lambda)\Big(\frac{(\beta+1)^{\alpha+x_n}}{\Gamma(\alpha+x_n)}\lambda^{\alpha+x_n-1}e^{-(\beta+1)\lambda}\Big)\mathrm{d}\lambda \\
p(x_{n+1} \mid x_n) &= \int_0^\infty \Big(\frac{e^{-\lambda}\lambda^{x_{n+1}}}{x_{n+1}!}\Big)\Big(\frac{(\beta+1)^{\alpha+x_n}}{\Gamma(\alpha+x_n)}\lambda^{\alpha+x_n-1}e^{-(\beta+1)\lambda}\Big)\mathrm{d}\lambda \\
p(x_{n+1} \mid x_n) &= \frac{(\beta+1)^{\alpha+x_n}}{\Gamma(\alpha+x_n)x_{n+1}!}\int_0^\infty e^{-\lambda}\lambda^{x_{n+1}}\Big(\lambda^{\alpha+x_n-1}e^{-(\beta+1)\lambda}\Big)\mathrm{d}\lambda \\
p(x_{n+1} \mid x_n) &= \frac{(\beta+1)^{\alpha+x_n}}{\Gamma(\alpha+x_n)x_{n+1}!}\int_0^\infty \Big(\lambda^{\alpha+x_n+x_{n+1}-1}e^{-(\beta+2)\lambda}\Big)\mathrm{d}\lambda
\end{aligned}$$

At this point, we recall the definition of the complete gamma function.

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\mathrm{d}t, \ \text{where} \ \Re(z) > 0$$

Note by the strictly real-valued supports and parameters of both the Gamma and Poisson distributions that the $\Re(z) > 0$ clause holds trivially in this case.

For our expression, we will proceed with substitution to get it into this form.

First, we let $t = (\beta + 2)\lambda$. To solve via substitution, we note that:

$$\mathrm{d}t = (\beta + 2)\mathrm{d}\lambda, \text{ hence } \mathrm{d}\lambda = \frac{\mathrm{d}t}{(\beta + 2)}$$

Let us substitute this value into our expression and simplify.

$$\begin{aligned}
p(x_{n+1} \mid x_n) &= \frac{(\beta+1)^{\alpha+x_n}}{\Gamma(\alpha+x_n)x_{n+1}!}\int_0^\infty \left(\frac{t}{(\beta+2)}\right)^{\alpha+x_n+x_{n+1}-1}(e^{-t})\frac{\mathrm{d}t}{(\beta+2)} \\
p(x_{n+1} \mid x_n) &= \frac{(\beta+1)^{\alpha+x_n}}{\Gamma(\alpha+x_n)x_{n+1}!(\beta+2)^{x_n+x_{n+1}-1}(\beta+2)}\int_0^\infty t^{\alpha+x_n+x_{n+1}-1}(e^{-t})\mathrm{d}t \\
p(x_{n+1} \mid x_n) &= \frac{(\beta+1)^{\alpha+x_n}}{\Gamma(\alpha+x_n)x_{n+1}!(\beta+2)^{\alpha+x_n+x_{n+1}-1}(\beta+2)^1}\Gamma(\alpha+x_n+x_{n+1}) \\
p(x_{n+1} \mid x_n) &= \frac{(\beta+1)^{\alpha+x_n}\Gamma(\alpha+x_n+x_{n+1})}{\Gamma(\alpha+x_n)x_{n+1}!(\beta+2)^{\alpha+x_n+x_{n+1}}} \quad \square
\end{aligned}$$

The above gives the Posterior Predictive used in Part 4 of the DPMM Mixing Distribution definition, as required.

Example of a σ-Algebra

For additional clarity, we provide an example of a σ-algebra $\mathcal{F}$ to demonstrate the properties mentioned in the Literature Review.

Let's consider the finitely countable and simple set $\mathbb{X} = \{a, b, c\}$.

Directly, $P(\mathbb{X}) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$ is the power set of $\mathbb{X}$, where we note $\{a, b, c\} = \mathbb{X}$. Consider $\mathcal{F} = \{\emptyset, \{a\}, \{b, c\}, \{a, b, c\}\} \subseteq P(\mathbb{X})$.

We will apply the properties discussed in the Literature Review section to prove that $\mathcal{F}$ is a σ-algebra.

To verify Property 1 (universality and non-emptiness), we note that we can also write $\mathcal{F}$ as $\{\emptyset, \{a\}, \{b, c\}, \mathbb{X}\}$. From this definition, it is direct to see that $\mathbb{X} \in \mathcal{F}$ and $\emptyset \in \mathcal{F}$, verifying universality and non-emptiness.

To verify Property 2 (closure under complementation), we note that $A^{c^c} = A$. Hence, if we show $A \in \mathcal{F} \implies A^c \in \mathcal{F}$ this is equivalent to showing that $A \in \mathcal{F} \iff A^c \in \mathcal{F}$. Since $\mathcal{F}$ is finitely countable, we can consider a case-wise basis for verification. Firstly, we have $A = \emptyset$. By definition, $A^c = \mathbb{X}$. We see that $\mathbb{X} \in \mathcal{F}$. As mentioned before, this implies that the case where $A = \mathbb{X}$ also holds. Now, we can proceed to verify the case where $A = \{a\}$. We note that $\{a\}^c = \{b, c\}$, and that $\{b, c\} \in \mathcal{F}$. Therefore, this case holds and thus so does the case where $A = \{b, c\}$ by biconditionality. Hence, we can conclude that $\mathcal{F}$ is closed under complementation.

To verify Property 3 (closure under countable unions), we first consider the concrete case where $\{A_i\}_{i=1}^2 = \{\{a\}, \{b, c\}\}$ where $A_1 = \{a\}$ and $A_2 = \{b, c\}$. We note that $A_1, A_2 \in \mathcal{F}$, so we would expect that $A_1 \cup A_2 \in \mathcal{F}$. Directly, $A_1 \cup A_2 = \{a\} \cup \{b, c\} = \{a, b, c\} = \mathbb{X}$, and we see that $\mathbb{X} \in \mathcal{F}$. For the remaining cases, $\forall A \in \mathcal{F}, A \cup \emptyset = A$ by fundamental set properties, and directly $A \in \mathcal{F}$ by construction. Similarly, $\forall A \in \mathcal{F}, A \cup \mathbb{X} = \mathbb{X}$ and we know that $\mathbb{X} \in \mathcal{F}$. Hence, for all cases, $\mathcal{F}$ is closed under countable unions.

From all of these properties, we can conclude that $\mathcal{F}$ is a σ-algebra, as required $\square$.

Example of a Probability Measure

Using the σ-algebra $\mathcal{F} = \{\emptyset, \{a\}, \{b, c\}, \{a, b, c\}\}$ we will define $\mu : \mathcal{F} \mapsto [0, 1]$ and prove that μ is a probability measure using the properties discussed in the Literature Review.

We will define a concrete example as follows, and show it is a probability measure on $\mathcal{F}$.

$$\mu(A) = \begin{cases} 2/3, & |A| = 1 \\ 1/3, & |A| = 2 \\ 1, & |A| = 3 \\ 0, & \text{otherwise} \end{cases}$$

Let's evaluate the properties discussed in the literature review to verify that μ is in fact a probability measure.

First, we evaluate Property 1 (non-negativity.) In effect, we wish to verify that $\forall A \in \mathcal{F}, \mu(A) \geq 0$. We can easily evaluate the universal in a case-wise basis.

a. $\emptyset \in \mathcal{F}$ and $\mu(\emptyset) = 0 \geq 0$.
b. $\{a\} \in \mathcal{F}$ and $\mu(\{a\}) = 2/3 \geq 0$.
c. $\{b, c\} \in \mathcal{F}$ and $\mu(\{b, c\}) = 1/3 \geq 0$.
d. $\mathbb{X} \in \mathcal{F}$ and $\mu(\mathbb{X}) = 1 \geq 0$.

Hence, $\forall A \in \mathcal{F}, \mu(A) \geq 0$, verifying the non-negativity clause. Further, we see that $\forall A \in \mathcal{F}, 0 \leq \mu(A) \leq 1$.

In addition, we can utilize the evaluations above to verify Property 2. Directly, we see that $\mu(\emptyset) = 0$ and $\mu(\mathbb{X}) = 1$, verifying Property 2 that a value of 1 is assigned to the sample space $\mathbb{X}$.

Finally, we verify Property 3 (countable additivity.) Again, because the σ-algebra is finitely countable, we verify all pairwise disjoint intersections on a case-wise basis.

a. First, we consider the set of pairwise disjoint sets $\{\emptyset, A\}$ for $A \in \mathcal{F}$. We see that $\forall A \in \mathcal{F}, \emptyset \cap A = \emptyset$ and $\emptyset \cup A = A$. Using, $\mu(\emptyset) = 0$, we observe that $\mu\left(\bigcup_i A_i\right) = \mu(A) = \mu(A) + \mu(\emptyset) = \sum_{i=1} \mu(A_i)$, as required.
b. Similarly, we consider $\mathbb{X} \in \mathcal{F}$, noting that $\forall A \in \mathcal{F}, \mathbb{X} \cap A = A$ is non-disjoint, so the case holds vacuously by falsity of the antecedent. A similar argument can be applied for $\{A, A\}, A \in \mathcal{F}$ which is evidently a non-disjoint pair.
c. The other pairwise disjoint set in $\mathcal{F}$ is $\{A_i\} = \{\{a\}, \{b, c\}\}$, since $\{a\} \cap \{b, c\} = \emptyset$. Hence, this pair should be countably additive. We can see directly that $\bigcup_i A_i = \{a\} \cup \{b, c\} = \mathbb{X}$, Hence, $\mu\left(\bigcup_i A_i\right) = \mu(\mathbb{X}) = 1$, so we anticipate $\sum_i \mu(A_i) = 1$. To verify, we see that $\sum_i \mu(A_i) = \mu(\{a\}) + \mu(\{b, c\}) = 2/3 + 1/3 = 1$, so countable additivity holds.

From all of these properties, we can conclude that μ is a probability measure on σ-algebra $\mathcal{F}$, as required $\square$.

Appendix 2: R Code

Click to Access GitHub Repository. The code to follow is separated across multiple different files. The Results Code is named `poisson_DPMM.R`, the DP Finite Approximation is named `finite_appx.R`, the plotting files are `plots_posterior_and_data.R` and `sampling_plot.R`. Finally, the data cleaning is from `data_processing_final.R`. The original Excel spreadsheet was too large to upload remotely. It can be accessed from Kaggle link in the Sources, and works directly with the data processing code.

Results Code

This code features F1 through F4 as discussed in the Methods section, and an implementation of the custom mixing distribution using the source data.

```r
library(dirichletprocess)
# set seed for reproducibility
set.seed(447)
# start the clock
start_time = proc.time()
```

```r
M = 10000
RUN = FALSE # TRUE if running sampler
TESTING = FALSE # TRUE if running synthetic validation

###########################
### Mixing Distribution ###
###########################

# define the framework conjugate mixture model
poisMd = MixingDistribution(
  distribution = "poisson",
  priorParameters = c(1, 2), # G_0 params
  conjugate = "conjugate"
)
# F 1: Poisson Likelihood
Likelihood.poisson = function(mdobj, x, theta){
  return(as.numeric(dpois(x, theta[[1]])))
}
# F 2: Gamma Prior : Base Measure
PriorDraw.poisson = function(mdobj, n){
  draws = rgamma(n, mdobj$priorParameters[1], mdobj$priorParameters[2])
  theta = list(array(draws, dim=c(1,1,n)))
  return(theta)
}
# F 3: Posterior Draw (defined by conjugacy)
PosteriorDraw.poisson = function(mdobj, x, n=1){
  priorParameters = mdobj$priorParameters
  theta = rgamma(n,  priorParameters[1] + sum(x),
                     priorParameters[2] + nrow(x))
  return(list(array(theta, dim=c(1,1,n))))
}

# F 4: Predictive Distribution by Marginalization
Predictive.poisson = function(mdobj, x){
  priorParameters = mdobj$priorParameters
  alpha = priorParameters[1]
  beta = priorParameters[2]
  pred = numeric(length(x))
  for(i in seq_along(x)){
    alphaP = alpha + x[i]
    betaP =  beta  + 1
    pred[i] = (beta ^ alpha) * gamma(alphaP)
    pred[i] = pred[i] / ( (betaP^alphaP) * gamma(alpha) )
    pred[i] = pred[i] / prod(factorial(x[i]))
  }
  return(pred)
}

###########################
### D.P. Gibbs Sampling ###
###########################

# read in cleaned data frame
df = read.csv("final_project/cleaned_crash_data.csv")
# monthly crash count, in 100s of crashes
y = ( round((df$crash_count)/100) )
```

```r
# create DP Poisson Mixture Model from mix dist. defined earlier
dirp = DirichletProcessCreate(y, poisMd)
# if running main code
if(RUN){
  # initialize and fit DPMM via Gibbs
  dirp = Initialise(dirp)
  dirp = Fit(dirp, M)
  dirp = Burn(dirp, 100)
  # compute, posterior frame: sampling from the posterior
  cat("Generating Posterior Frame...")
  # include 95% and 99% Credible Intervals
  postf = PosteriorFrame(dirp, 0:22, 10000, ci_size = c(0.1, 0.01))
  # save to avoid repeat simulation
  saveRDS(postf, file = "final_project/posterior_sampleframe.RDS")
  saveRDS(dirp, file =  "final_project/posterior_results.RDS")
}

###########################
## Synthetic Validation ###
###########################

if(TESTING){
  # generate three synthetic rates (total size N = 1,076)
  synth_data = c(rpois(269, lambda = 3),
                 rpois(538, lambda = 12),
                 rpois(269, lambda = 0.8))
  # initiate the dirichlett process poisson
  synth = DirichletProcessCreate(synth_data, poisMd)
  # initialize and fit DPMM via Gibbs
  synth = Initialise(synth)
  synth = Fit(synth, M)
  synth = Burn(synth, 100)
  # compute, posterior frame: sampling from the posterior
  cat("Generating Posterior Frame for Synthetic Data...")
  # include 95% and 99% Credible Intervals
  post_synth = PosteriorFrame(synth, 0:22, 10000, ci_size = c(0.1, 0.01))
  # save locally
  saveRDS(postf, file = "final_project/synthetic_sampleframe.RDS")
  saveRDS(dirp, file =  "final_project/synthetic_results.RDS")
}

# report runtime
total_time = proc.time() - start_time
cat("Total Runtime of Script: ", total_time['elapsed'], "seconds\n")
```

Dirichlet Process Finite Approximation

Here, we provde an implementation of the GEM process, used in the Methods section.

```r
library(ggplot2)
library(latex2exp)
library(RColorBrewer)
library(scales)
# seed for reproducibility
set.seed(1924)
# number of clusters
```

```r
K = 20
# base measure/distribution
G_0 = function(n) {
  rgamma(n, 1, 2)
}
alpha = rgamma(1, 1, 1)

##############################
##### Finite D.P. Appx. #####
##############################

# compute beta pulls
b <- rbeta(K, 1, alpha)
# empty vector for population
p <- numeric(length = K)
# initial stick break
p[1] <- b[1]
# further breaks following GEM(a) definition
p[2:K] <- sapply(2:K, function(i)
  b[i] * prod(1 - b[1:(i - 1)]))
# then, sample from base distribution by weight probabilities
# this creates the finite approximation as discussed in the methods
theta <- sample(G_0(K), prob = p, replace = TRUE)

##############################
##### FINITE D.P. PLOT ######
##############################


plotDF = data.frame(DirB = theta,  DirP = log(p))
# plot heatmap of results
p1 = ggplot(plotDF, aes(x = DirB, y = DirP)) +
  geom_density_2d_filled() +
  labs(
    title =
      TeX(
        "Finite Approximation of Dirichlet Process : DP($\\alpha G_0$) Realization"
      ),
    subtitle = TeX(
      "Where $K = 20$, $G_0 \\sim$ gamma(1, 2) and $\\alpha \\sim$ gamma(1,1)"
    ),
    y = TeX("Log of Mixture Weights: $\\{\\pi_k\\}_{k = 1}^K$"),
    x = TeX("Cluster Parameters: $\\{\\lambda_k\\}_{k = 1}^K$")
  )  + theme_bw() + scale_fill_viridis_d(option = "magma")  + theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    plot.background = element_rect(fill = "white", colour = "white"),
    plot.title = element_text(margin = margin(b = -3.5, unit = "pt")),
    plot.subtitle = element_text(margin = margin(b = -5, unit = "pt")),
    legend.position = "none",
    legend.title = element_blank(),
    axis.ticks.length = unit(-2, "mm"),
    legend.text = element_text(size = 8),
    legend.margin = margin(t = 0, unit = "mm", l = -5)
  ) + scale_y_continuous(n.breaks = 10) +
  scale_x_continuous(n.breaks = 10)
```

```r
print(p1) # trbl
ggsave(
  "final_project/dirch_appx.png",
  plot = p1,
  width = 7,
  height = 5
)
```

Data Processing Code

```r
library(dplyr)
library(lubridate)
library(readxl)
library(tidyr)
library(pbapply)
#######################
### Data Processing ###
#######################

# NOTE: the aggregation is over a very large dataset, so runtime is slow

cat("Reading Data... \n")
# read Chicago Crash data from excel using `readxl`
df <- read_excel("final_project/crash_dates_and_damages.xlsx")
cat("Processing Dates...\n")
df <- df %>%
  # drop weirdly formatted excel dates
  filter(!grepl("^\\d+\\.\\d+$", CRASH_DATE)) %>%
  # standrdize remaining dates to same timezone
  mutate(CRASH_DATE = mdy_hms(CRASH_DATE, tz = "UTC", quiet = TRUE)) %>%
  # remove NAs
  drop_na(CRASH_DATE) %>%
  # standardize formatting to MDY
  mutate(CRASH_DATE = format(CRASH_DATE, "%m/%d/%Y"))

cat("Aggregating... \n")
# here, we aggregate severity counts weekly
aggregated_data <- lapply(damage_levels, function(damage_level) {
  # we filter through each damage level
  df_filtered <- df %>% filter(DAMAGE == damage_level)
  # and aggregate by week
  weekly_aggregation <- df_filtered %>%
    mutate(Week = floor_date(as.Date(CRASH_DATE, format = "%m/%d/%Y"), "week")) %>%
    group_by(Week) %>%
    summarise(Crash_Count = n(), .groups = 'drop') %>%
    arrange(Week)
  # use NNI for poorly-captured 2014, 2015 data
  cat("Imputing ", damage_level, "...\n")
  # get 2016 weeks
  weeks_2016 <- weekly_aggregation %>%
    filter(year(Week) == 2016) %>%
    pull(Week)
  # get exact week if present, else nearest week by euclidean distance
  weekly_aggregation <- weekly_aggregation %>%
    rowwise() %>%
    mutate(
      Nearest_2016_Week = if(year(Week) %in% c(2014, 2015)) {
        weeks_2016[which.min(abs(difftime(Week, weeks_2016, units = "weeks")))]
      } else {
        Week
      }
    ) %>%
    left_join(weekly_aggregation %>% filter(year(Week) == 2016) %>%
                select(Week, Crash_Count), by = c("Nearest_2016_Week" = "Week")) %>%
    mutate(
      # use 2016's Crash_Count for nearest week and add to scaled 2014 count if present
      Crash_Count_Adjusted = if_else(year(Week) == 2014, Crash_Count.y + Crash_Count.x / max(Crash_Count.x
    ) %>%
    select(Week, Crash_Count_Adjusted)

  return(weekly_aggregation)
})

# the data frame we use is then the aggregated weeks & counts
outDF = data.frame(
  crash_time = c(
    aggregated_data[[1]]$Week,
    aggregated_data[[2]]$Week,
    aggregated_data[[3]]$Week
  ),
  crash_count =  c(
    aggregated_data[[1]]$Crash_Count_Adjusted,
    aggregated_data[[2]]$Crash_Count_Adjusted,
    aggregated_data[[3]]$Crash_Count_Adjusted
  )
)

# which we write to a csv
cat("Writing to File... \n")
write.csv(outDF, "final_project/cleaned_crash_data.csv", row.names = FALSE)
```

Raw Data Plotting Code

Includes LaTeX generating function for posterior results table.

```
library(ggplot2)
library(latex2exp)
library(hexbin)
library(scales)
library(knitr)
library(kableExtra)

##########################
##### Raw Data Plot ######
##########################

# read in data
data = read.csv("final_project/cleaned_crash_data.csv")
# dividing to account for aggregation
y = (round(data$crash_count / 100))
ind = seq_along(y)
```

```r
p0 = ggplot(data.frame(ind, y), aes(x = ind, y = y)) +
  geom_hex(alpha = 1) +
  scale_fill_gradient(low = "#e8e1df", high = "#5e1317",
                      name = "Point Density") +
  theme_bw() +
  labs(
    title = "Hex Plot of Weekly-Aggregated Car Accidents in Chicago by Logged Time",
    subtitle = "From 2014-2023, Missing Values Imputed by Nearest-Neighbours",
    x = "Time Logged in System",
    y = "Car Crashes (100s of Crashes)"
  ) +  theme(
    panel.grid.minor = element_line(
      color = "grey90",
      linetype = "dashed",
      linewidth = 0.5
    ),
    legend.margin = margin(0, 0, 0, 0),
    legend.justification = "top"
  )
print(p0)

##########################
###### Latex Table  ######
##########################

# read in results from other file
dirichlet_results = readRDS("final_project/posterior_results.RDS")
# format nicely
results_DF = data.frame(
  lambdas = unlist(dirichlet_results$clusterParameters),
  weights = dirichlet_results$weights )
# make a table - columns are clusters
results = kable(t(round(results_DF, 3)),
                format = "latex",
                booktabs = TRUE,
                caption = "DPMM Posterior Parameters and Weights") %>%
  kable_styling(latex_options = "striped", position = "center") %>%
  column_spec(1, bold = TRUE, border_left = TRUE)
# write to latex to render
# this gives the initial design, which I edited later
writeLines(results,  "final_project/results.tex")

ggsave("final_project/data_raw.PNG", plot = p0, width = 6.5, height = 5)
```

Posterior Simulations and Plotting

```r
library(dirichletprocess)
library(latex2exp)
library(logspline)
library(ggplot2)
library(scales)
library(pbapply)

##########################
#### Posterior Plots #####
```

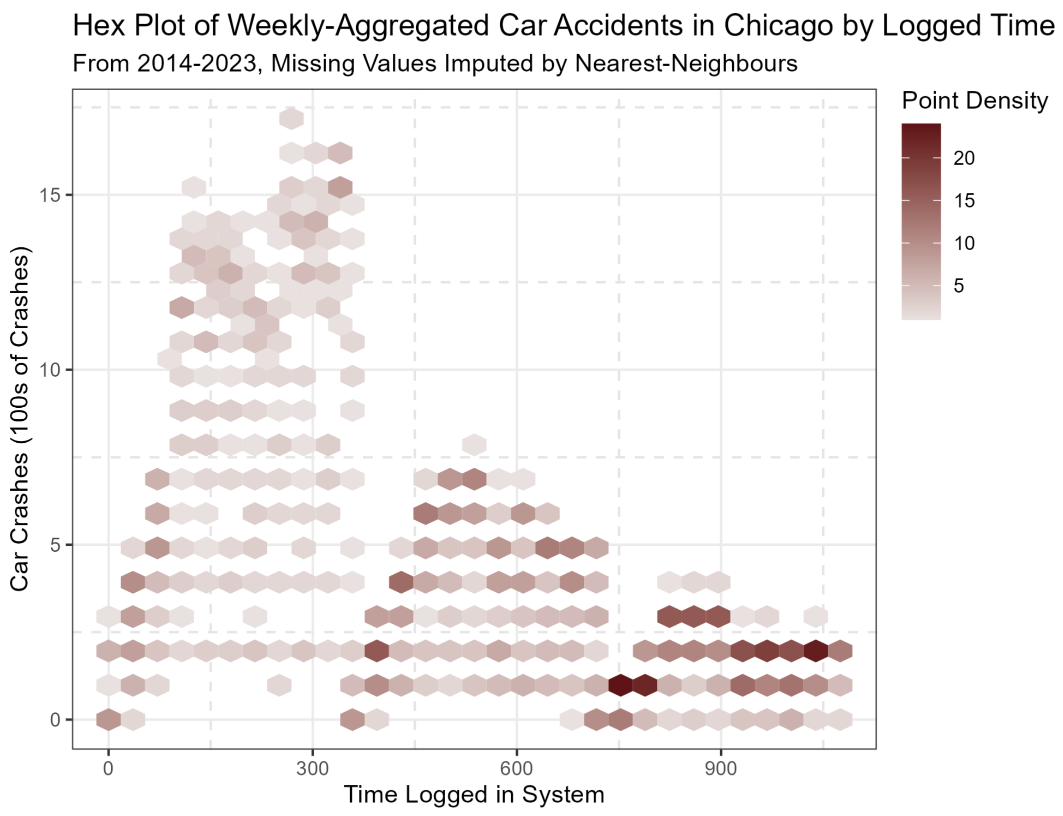
```r
###########################

set.seed(447)
# true if running simulations
RUN = FALSE
# read in DP Fit and Data
df = read.csv("final_project/cleaned_crash_data.csv")
dp = readRDS(file =  "final_project/posterior_results.RDS")
# get the rates in the clusters
rates = unlist(dp$clusterParameters)
# get the weights and cluster sizes
sizes = unlist(dp$pointsPerCluster)
weights = unlist (dp$weights)
# declare number of simulations
M = 10000
B = 100 # burn in
N = nrow(df)
# run M simulations of size N from the posterior mixture distribution
if (RUN) {
  cat("Generating posterior draws... \n")
  # then, simulate M draws from the posterior DPMM
  simulated_datasets =
    pbsapply(1:M,
             function(m) {
               # where each draw is of size n = 1076
               # mixed by rates and cluster sizes
               unlist(pbsapply(1:length(sizes), function(i) {
                 # generate z_i draws at rate r_i
                 rpois(sizes[i], rates[i])
               }))
             })
  cat("Done! \n")
  cat("Generating posterior densities... \n")
  # then compute the densities to be compared against the data histogram
  simulations = matrix(0, nrow = 24, ncol = M)
  # here, we get the proportion counts
  simulations = (pbsapply(1:M,
                          function(m) {
                            as.numeric(table(factor(simulated_datasets[, m], levels = 0:23)))
                          }))
  # save locally (reduces re-run time)
  saveRDS(simulations, file = "final_project/posterior_sims.RDS")
  cat("Done! \n")
} else {
  cat("Using saved simulations... \n")
  simulations = readRDS(file = "final_project/posterior_sims.RDS")
}
###########################
# Predictive vs. Actual ##
###########################

# density profile comparison
posterior_data <- data.frame(
  x = 0:23,
  avg_post = apply(simulations, MARGIN = 1, mean),
  low_post = apply(simulations, MARGIN = 1, quantile, probs = 0.025),
  hi_post  = apply(simulations, MARGIN = 1, quantile, probs = 0.975),
  vlow_post = apply(simulations, MARGIN = 1, quantile, probs = 0.005),
  vhi_post  = apply(simulations, MARGIN = 1, quantile, probs = 0.995)
)
p1 <- ggplot() +
  geom_bar(
    data = df,
    aes(x = round(crash_count / 100),
        y = after_stat(prop)),
    stat = "count",
    fill = "grey65",
    color = "white",
    alpha = 0.5
  ) +
  labs(
    title = TeX(
      "Poisson Predictive Densities from DPMM with Observed Data Histogram"
    ),
    subtitle = TeX(
      "Using Clusters $\\{z_i\\}_{i \\in [1,4]}$ and Rates $\\{\\lambda_i\\}_{i \\in [1,4]}$ including 95%
    ),
    y = "Probability Density",
    x = TeX("Count Values")
  ) +
  theme_bw() +
  xlim(-1, 20) +
  geom_ribbon(
    data = posterior_data,
    aes(x = x, ymin = low_post/N, ymax = hi_post/N),
    fill = "#5a189a", ##075957",
    alpha = 0.35
  ) +
  geom_line(
    data = posterior_data,
    aes(x = x, y = avg_post/N),
    color = "#240046",#"#075957",
    size = 0.8,
    alpha = 0.8
  ) +
  geom_ribbon(
    data = posterior_data,
    aes(x = x, ymin = hi_post/N, ymax = vhi_post/N),
    fill = "#c77dff",#"#10a19d",
    alpha = 0.5
  ) +
  geom_ribbon(
    data = posterior_data,
    aes(x = x, ymin = vlow_post/N, ymax = low_post/N),
    fill = "#c77dff",#"#10a19d",
    alpha = 0.5
  ) +
  scale_y_continuous(n.breaks = 10) +
  theme(
    panel.grid.minor = element_line(
      color = "grey90",
      linetype = "dashed",
      linewidth = 0.5
    ),
```

```r
    plot.title = element_text(size = 11.75, vjust = -0.1),
    plot.subtitle = element_text(size = 10)
  )
print(p1)
ggsave("final_project/post_comp.png", plot = p1, width = 7, height = 5)

##########################
###### Rate Boxplots #####
##########################

# get MCMC iterations post Burn-In for each rate param
L_data = data.frame(sapply(1:3, function(L) {
  sapply(1:(M - B), function(m) {
    (dp$clusterParametersChain)[[m]][[1]][L]
  })
}))

colnames(L_data) = c("Lambda 1", "Lambda 2", "Lambda 3")

L_data_long = L_data %>%
  mutate(id = row_number()) %>%
  pivot_longer(
    cols = -id,
    names_to = "Parameter",
    values_to = "Rate"
  )
p2 = ggplot(L_data_long,
       aes(
         x = Parameter,
         y = Rate,
         fill = Parameter,
         colour = Parameter
       )) +
  geom_boxplot(width = 0.2, alpha = 0.5,
               outlier.shape = 3, outlier.size = 0.5) +
  geom_violin(trim = TRUE, alpha = 0.3) +
  scale_fill_manual(values = c(
    "Lambda 1" = "#414833",
    "Lambda 2" = "#936639",
    "Lambda 3" = "#c2c5aa"
  )) +
  scale_colour_manual(values = c(
    "Lambda 1" = "#333d29",
    "Lambda 2" = "#7f4f24",
    "Lambda 3" = "#656d4a"
  )) +
  labs(
    title = TeX("Boxplots of Posterior Rate Parameters"),
    subtitle = TeX("For Dominant-Populated Clusters $i \\in \\{1, 2, 3\\}$"),
    x = "Parameter",
    y = "Posterior Rate"
  ) +
  scale_x_discrete(labels = c(
    TeX("$\\lambda_1$"),
    TeX("$\\lambda_2$"),
    TeX("$\\lambda_3$")
  )) +
  scale_y_continuous(n.breaks = 10)+
  theme_bw() +  theme(
    panel.grid.minor = element_line(
      color = "grey90",
      linetype = "dashed",
      linewidth = 0.5
    ),
    legend.position = "none",
    axis.text.x = element_text(size = 12)
  )

ggsave("final_project/post_box.png", plot = p2, width = 4, height = 6)


##########################
#### Cluster Reports #####
##########################

cluster_density = data.frame( table(sapply(dp$weightsChain, length) ))

probs = sapply(1:nrow(cluster_density),
       function(c){cluster_density$Freq[c]/(sum(cluster_density$Freq))})
# values used in paragraph
sum( seq(from = 3, to = 13)*probs )
final_density = data.frame(table(sapply(dp$weightsChain[9400:9900], length)))
final_probs = sapply(1:nrow(final_density),
               function(c){final_density$Freq[c]/(sum(final_density$Freq))})
final_probs
first_density = data.frame(table(sapply(dp$weightsChain[0:500], length)))
first_probs = sapply(1:nrow(first_density),
                     function(c){first_density$Freq[c]/(sum(first_density$Freq))})
```

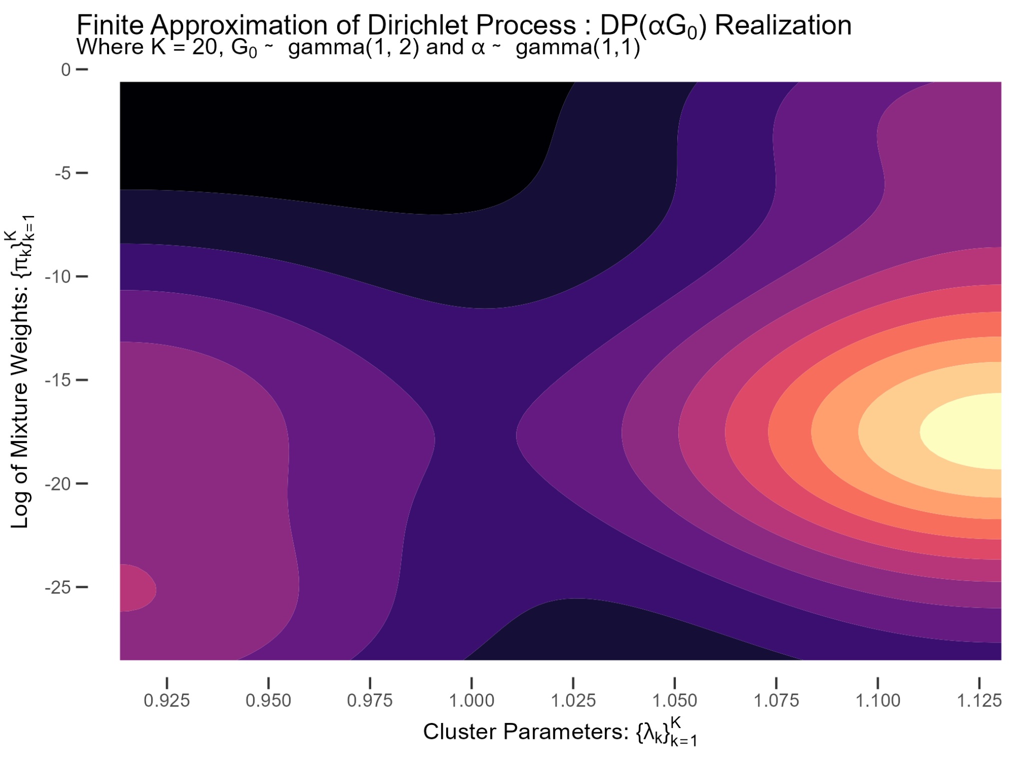
Appendix 3: Large Images

Larger images of the figures shown in the main document are included below, in order of appearance.

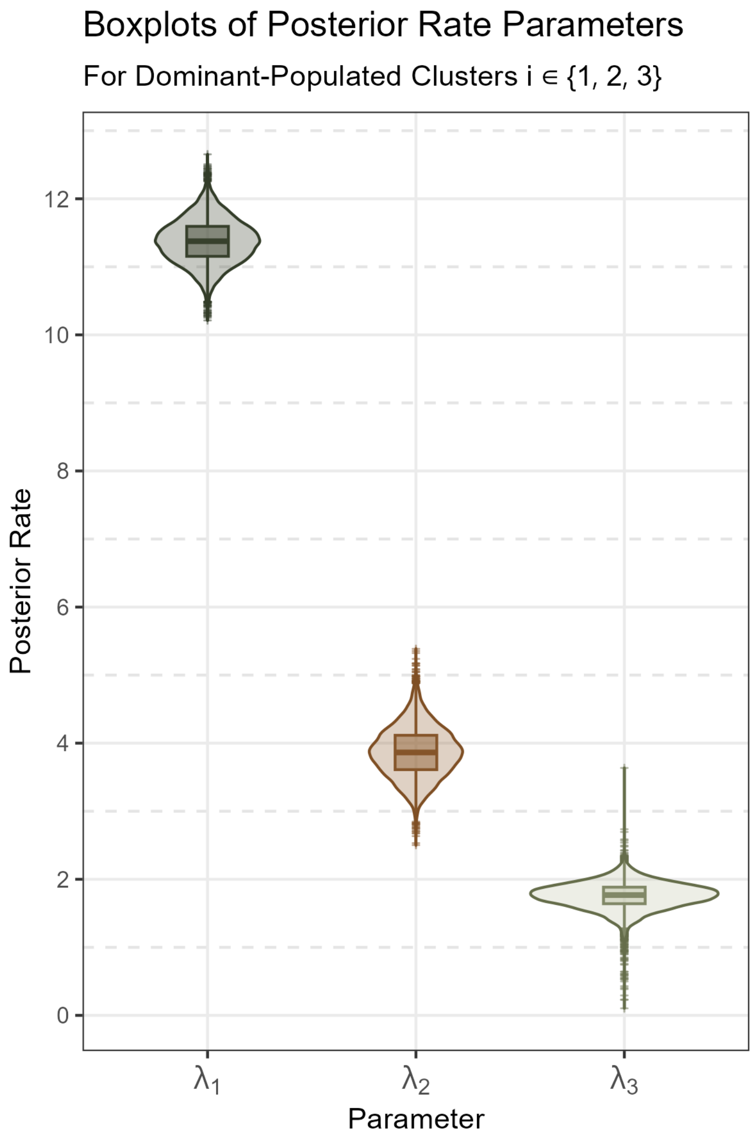
Plot of CPD Data



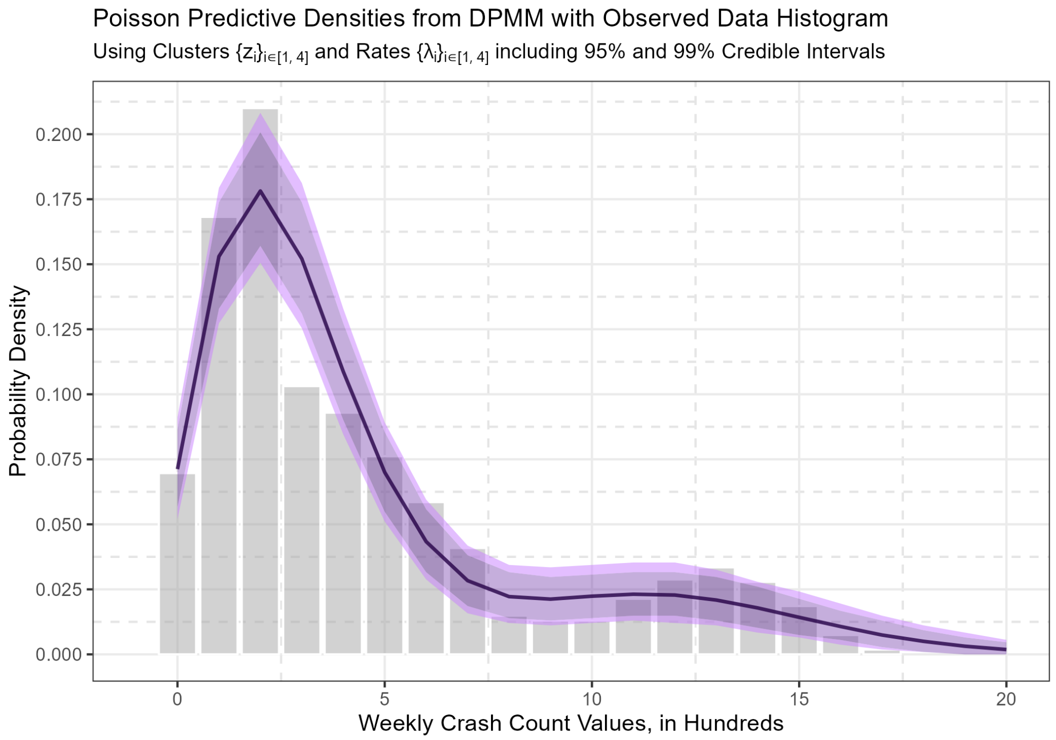
Plot of DP Finite Approximation



Boxplots of Posterior Rates



Posterior Predictive vs. Source



Acknowledgements: Special thanks to Prof. Lasantha Premarathna for sparking my interest in non-parametrics and to my friends and family who put up with me talking about statistics constantly.

Sources

(Zhang et al. 2019)

Billingsley, Patrick. 2012. *Probability and Measure, Anniversary Edition*. Wiley.

CPD. 2024. "Chicago Traffic Crashes - Chicago Police Dept." Kaggle. https://doi.org/10.34740/KAGGLE/DSV/7339559.

Demirbas, Seckin, and Andrew Rechnitzer. 2023. "An Introduction to Mathematical Proof : MATH 220." Free web and pdf textbook. https://personal.math.ubc.ca/~PLP/.

Ferguson, Thomas. 1973. "Bayesian Analysis of Some Nonparametric Problems." *Annals of Statistics* 1 (2): 209–30. https://doi.org/10.1214/aos/1176342360.

Hannah, Lauren A. 2011. "Dirichlet Process Mixtures of Generalized Linear Models." *Journal of Machine Learning Research* 12: 1923–53. https://www.jmlr.org/papers/volume12/hannah11a/hannah11a.pdf.

Ishwaran, Hemant, and Lancelot F. James. 2001. "Gibbs Sampling Methods for Stick-Breaking Priors." *Journal of the American Statistical Association* 96 (453): 161–73. https://doi.org/10.1198/016214501750332758.

Johny, Anoop. 2018. "Chicago Traffic Crashes - Chicago Police Dept." Kaggle kernel. https://www.kaggle.com/code/anoopjohny/chicago-traffic-crashes-chicago-police-dept.

Markwick, Dean. 2023. *Dirichletprocess: An 'r' Package for Dirichlet Process Mixture Models*. https://dm13450.github.io/dirichletprocess/.

Neal, Radford M. 2000. "Markov Chain Sampling Methods for Dirichlet Process Mixture Models." *Journal of Computational and Graphical Statistics* 9 (2): 249–65. https://doi.org/10.1080/10618600.2000.10474879.

Rudin, Walter. 1986. *Real and Complex Analysis*. 3rd ed. McGraw-Hill.

Sethuraman, Jayaram. 1994. "A Constructive Definition of Dirichlet Priors." *Statistica Sinica*.

StackOverflow. 2023. "How to Use Custom Functions in Mutate (Dplyr)." Stack Overflow. https://stackoverflow.com/questions/44730774/how-to-use-custom-functions-in-mutate-dplyr.

Xing, Eric P. 2014. "Hierarchical Dirichlet Processes." Carnegie Mellon University; Online. https://www.cs.cmu.edu/~epxing/Class/10708-14/scribe_notes/scribe_note_lecture20.pdf.

Zhang, Biyao, Kaisong Zhang, Luo Zhong, and Xuanya Zhang. 2019. "Research on Dirichlet Process Mixture Model for Clustering." *Ingénierie Des Systèmes d'Information* 24 (2): 183–89. https://doi.org/10.18280/isi.240209.