Analysis of Market Volatility

Exploring Bayesian Non-Parametric and Bootstrap Methods

Raul Unnithan

January 14, 2025

Contents

Chapter 1 Introduction

1.1 Research Objective and Aims

This report's primary objective is to analyse the volatility index (VIX), a measure of market volatility, and quantify uncertainty in VIX returns with Classical and Bayesian Bootstrap methods to help traders when making their investment decisions.[1] Classical Bootstrap estimates frequentist confidence intervals for the mean and variance of the VIX log returns, and Bayesian Bootstrap gives posterior distributions for key statistics, such as the mean and variance. This report also aims to use the Dirichlet Process Mixture Model (DPMM), a Bayesian non-parametric method, to identify VIX return clustering patterns. DPMMs do this by allowing the number of clusters to grow with the data, using the Dirichlet Process Prior to govern the distribution over cluster assignments.[2] Clustering VIX returns helps traders assess market volatility to facilitate asset position adjustment. This study also aims to compare the results obtained from Classical and Bayesian Bootstrap methods to compare them and evaluate their strengths and limitations. Finally, plots will be generated to understand Bootstrap distributions, posterior densities, and cluster-specific behaviours over time. The final aim is to use cluster assignments as a feature in XGBoost, a machine-learning (ML) regression model, to predict VIX returns using cluster-specific volatility patterns. A.1 in the appendix shows a summary of the key figures in this report.

1.2 Dataset Description and Pre-processing

The VIX data was obtained from Kaggle, and it spans 2012 to 2021 and includes the dates and different prices of the VIX, but only open and close prices were used. It is derived from the prices of S&P 500 index options.[1] This dataset came with two main issues: missing data and anomalies. Missing values were omitted for accuracy, and no anomalies were removed because these methods explore all the features of the VIX data. The date column was standardised for time series analysis, see B.1, which shows a spike in 2020 due to COVID-19 and its market uncertainty. Percentage log returns were then computed through this calculation: $r_t = \log\left(\frac{P_t}{P_{t-1}}\right) \times 100$, where r_t is the log return at time t, and P_t is price. This transformation normalises and prepares the data for analysis.

Chapter 2 Methodology

2.1 Bootstrap Methods

Bootstrap methods give a non-parametric approach for estimating the distribution of a statistic, which is useful for financial data like VIX returns that predominantly deviate from normality, as proven by the QQ-Plot in B.2. Classical Bootstrap (CB) approximates the sampling distribution of a statistic by resampling with replacement from the empirical distribution of the data.[3] The empirical distribution function (EDF) assigns equal probability to each observation, allowing CB to estimate bias, variance, and confidence intervals without relying on asymptotic normality assumptions.[2]

Bayesian Bootstrap (BB) is an extension of the CB where instead of resampling the observed data as in CB, BB assigns random weights to the data points. These weights, π, for the data points are drawn from the conjugate Dirichlet distribution: $\pi \sim \text{Dir}(\alpha_1, \dots, \alpha_K)$, where the sum of the weights is one.[2] This approach is very adaptable as the distribution of π can reflect prior uncertainty or remain flat, which is important for capturing uncertainty in volatile financial data. When the distribution is flat, the BB closely approximates the CB, making the CB a special case of the Bayesian.[2]

2.2 Dirichlet Processes

A Dirichlet Process (DP) is a distribution over distributions.[2] It is defined as $G \sim \text{DP}(\alpha, G_0)$, where: G_0 is the centring, representing prior knowledge about the distribution and α is the precision or total mass parameter, controlling the number of clusters.[2] Given observations $X_1, \dots, X_n$, the posterior distribution of G is itself a DP:

$$G \mid X_1, \dots, X_n \sim \text{DP}\left(\alpha + n, \frac{\alpha G_0 + \sum_{i=1}^{n} \delta_{X_i}}{\alpha + n}\right),$$

where δ_{X_i} is the Dirac delta function centred at X_i.[4] The predictive distribution for X_{n+1} is obtained by marginalising over the posterior G:

$$P(X_{n+1} \mid X_1, \dots, X_n) = \int P(X_{n+1} \mid G) \, dP(G \mid X_1, \dots, X_n).$$

By the definition of the DP, the posterior predictive distribution is a weighted mixture of the base measure G_0 and the empirical distribution of the observed data:

$$P(X_{n+1} = \theta \mid X_1, \ldots, X_n) = \begin{cases} \frac{m_k}{\alpha+n}, & \text{if } \theta = \theta_k \text{ (existing cluster)}, \\ \frac{\alpha}{\alpha+n}, & \text{if } \theta \text{ is a new cluster}, \end{cases}$$

where m_k is the number of observations in cluster k, n is the total number of observations and α is the precision parameter.[4] $\frac{m_k}{\alpha+n}$ reflects the likelihood of X_{n+1} joining an existing cluster k, proportional to the size of the cluster m_k. $\frac{\alpha}{\alpha+n}$ represents the probability of forming a new cluster, which depends on the concentration parameter α. The predictive distribution exhibits the "rich-get-richer" property, where larger clusters are more likely to attract new data points.[4] This consideration of using existing clusters and creating new clusters makes the DP useful for non-parametric (NP) clustering tasks.

A DPMM is an advanced extension of the standard DP model.[2] It allows for greater flexibility as it removes the "awkwardness" of using the DP model for continuous data.[2] A DPMM can be written as:

$$y_i \sim F = \int p(y_i|\theta)\, dG(\theta),$$

where y_i is an observed data point, F is the mixture distribution, G is the random distribution drawn from a DP, θ are the component distribution parameters and $p(y_i|\theta)$ is the likelihood for the data y_i. Mixture distributions are combinations of many PDFs, which can be similar or different types.[5] DPMMs are mainly applied in density estimation and clustering tasks, which are useful when modelling complex data distributions. DP generates discrete distributions, making its use not entirely appropriate for continuous density estimation; however, this can be fixed by convolving its trajectories with a proper continuous kernel function, $p(y_i|\theta)$.[2]

DPMMs also allow the number of clusters to grow with new data. This lack of need for pre-specifying cluster numbers is why it was used over other cluster methods, especially for VIX returns whose true cluster number is unknown. DPMMs induce clustering because the DP generates discrete random measures. The discreteness of the measure means there are ties among the latent variables, resulting in the clustering that is central to DPMMs.[6] This property ensures that multiple observations can share the same parameter values, resulting in natural groupings or clusters.

The Chinese Restaurant Process gives the intuitive logic behind clustering: each customer (data point) either join an existing table (cluster) or starts a new one.[6] The probability of joining an existing cluster is proportional to its size, while the likelihood of starting a new cluster is proportional to the precision parameter α.[6] The precision parameter of the DP controls the number of clusters formed. A higher α increases the likelihood of creating more clusters, while a smaller α favours fewer, larger clusters.

DPMMs are analysed here using the Pitman-Yor Process (PYP), which generalises the DP.[2] The PYP uses a discount parameter, $\beta \in [0,1]$, a strength parameter $\alpha > -\beta$, and the centring measure G_0. The PYP is: $G \sim \text{PY}(\beta, \alpha, G_0)$. When $\beta = 0$, the PYP reduces to a standard DP, $G \sim \text{DP}(\alpha, G_0)$.[2] For univariate DPMMs, the model allows infinite mixture fitting of normal densities through stick-breaking representation.[2]

Stick-breaking representation represents the random probability measure as a discrete random sum whose weights are formed by independent and identically distributed (i.i.d) sequences of beta variates and draws from the normalised DP parameter base measure: $\pi_k = \beta_k \prod_{j=1}^{k-1}(1-\beta_j), \quad \beta_k \sim \text{Beta}(1,\alpha)$, where β_k is sampled from a Beta distribution, and π_k are the resulting weights for the mixture components.[7,8] The stick-breaking process provides a way to represent the DP as a weighted sum of distributions, where each 'break' represents the proportion of data allocated to each cluster. Imagine breaking a stick at random points, with each piece representing the weight of a cluster.

Lee and MacEachern (2020) gave proof of the stick-breaking property of DP. They started with the posterior distribution of F under a DP prior. When conditioned on a single observation $X = x$, the posterior distribution is given by: $F \mid X = x \sim \text{DP}(\alpha + \delta_x)$.

They then use the self-similarity property of the DP to further express the posterior as: $F \sim \beta\delta_x + (1-\beta)G$, where $\beta \sim \text{Beta}(1,\alpha)$, $G \sim \text{DP}(\alpha)$, and β and G are independent, which indicates that the updated distribution F is a mixture of a degenerate distribution at x and another DP G, scaled by β and $1-\beta$, respectively. Self-similarity is key for the recursive construction of F. First, sample $\theta_1 \sim F$, where $F \sim \text{DP}(\alpha)$. Self-similarity implies that $F \sim \pi_1\delta_{\theta_1} + (1-\pi_1)G_1$, where $\pi_1 = \beta_1$, $\beta_1 \sim \text{Beta}(1,\alpha)$, and $G_1 \sim \text{DP}(\alpha)$. Next, given G_1, sample $\theta_2 \sim G_1$. The same self-similarity property of G_1 implies: $G_1 \sim \pi_2\delta_{\theta_2} + (1-\pi_2)G_2$, where $\pi_2 = \beta_2(1-\beta_1)$, $\beta_2 \sim \text{Beta}(1,\alpha)$, and $G_2 \sim \text{DP}(\alpha)$. Repeating this process, the weights π_k are constructed iteratively as:

$$\pi_k = \beta_k \prod_{j=1}^{k-1}(1-\beta_j), \quad \beta_k \sim \text{Beta}(1,\alpha).$$

The recursive process constructs F as a sum of degenerate measures, scaled by the weights:

$$F \sim \sum_{k=1}^{K} \pi_k \delta_{\theta_k} + \prod_{j=1}^{K}(1-\beta_j)G_K,$$

where $G_K \sim \text{DP}(\alpha)$. As $K \to \infty$, the remainder term $\prod_{j=1}^{K}(1-\beta_j)G_K$ vanishes with probability one, leading to the infinite sum: $F = \sum_{k=1}^{\infty}\pi_k\delta_{\theta_k}$. Finally, Lee and MacEacher verify the properties of the components. The sequence $\{\theta_k\}_{k=1}^{\infty}$ is marginally i.i.d. from the base measure F_0, and the weights π_k are derived from the stick-breaking process using the Beta-distributed random variables β_k. Thus ensuring F satisfies DP properties. $\blacksquare$

2.3 MCMC Sampling

Another key aspect of DPMMs is Markov Chain Monte Carlo (MCMC) Sampling. MCMC methods, such as Gibbs sampling, perform DPMM posterior inference. This is done by iteratively updating each component, which is conditioned on the most recent values of the other parameters with a Markov chain that has the posterior as its equilibrium distribution.[9] Clusters are assigned via Gibbs sampling, and assignments for each data point are resampled based on the current state of all other observations. Gibbs sampling updates cluster assignments like DPMMs, with new clusters formed based on α.

In cases where the model is non-conjugate, direct sampling from the posterior is not feasible. In non-conjugate cases, Metropolis-Hastings is used to refine cluster assignments. Metropolis-Hastings updates reassign observations by proposing candidate clusters and accepting or rejecting the proposal based on the ratio of likelihoods:

$$a(c^*, c) = \min\left(1, \frac{f(y_i|\phi_{c^*})}{f(y_i|\phi_c)}\right),$$

where c^* denotes the proposed cluster and c is the current assignment.[9] This allows for posterior exploration without needing explicit integration over cluster parameters.

To improve sampling efficiency, burn-in periods are used to discard initial samples generated before the Markov chain reaches stationarity, ensuring that only samples from the posterior distribution are kept.[9] Thinning reduces autocorrelation by selecting every nth sample from the chain, keeping retained samples independent.[9]

Gibbs sampling generates an ergodic Markov chain that converges to the posterior.[9] It decomposes the joint posterior into conditionally independent distributions. Then, repeated cluster assignments and parameter updates result in the Markov chain converging to the actual posterior distribution of cluster configurations and mixture components.

Simulating this Markov chain for a sufficient number of iterations ensures that samples from the chain approximate the posterior expectation, which can be expressed as:

$$E[f(x)] \approx \frac{1}{T}\sum_{t=1}^{T} f(x^{(t)}),$$

where T is the post-burn-in samples.[9] This scheme is essential for DPMM clustering.

A trace plot shows DPMM counts across MCMC iterations to assess convergence and mixing. Stable, random fluctuations indicate convergence, such as in B.9. While frequently used to assess convergence, the Gelman-Rubin statistic was not used due to the trace plot stability.

XGBoost is a new ML model which can use DPMM clusters assigned by Gibbs Sampling to improve its predictions.

2.4 XGBoost

Extreme gradient boosting or XGBoost is an extension of gradient tree boosting, where trees are iteratively added to minimise a predefined loss function.[10] Gradient tree boosting is defined as follows:

$$\hat{y}_i = \sum_{t=1}^{T} f_t(x_i), \quad f_t \in \mathcal{F},$$

where $\mathcal{F}$ represents the space of regression trees.[10] At each step, a new tree f_t is added to correct the current model residuals, aiming for ensemble loss minimisation.

XGBoost uses an additional regularisation term in its objective function, which helps smooth the learned weights and mitigates over-fitting by selecting simpler, more predictive models.[10] Its objective function is the following:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{t=1}^{T} \Omega(f_t),$$

where l is the loss function, and $\Omega(f_t)$ is a regularisation term penalising model complexity.[10] XGBoost optimises the objective function using a second-order Taylor approximation. The loss function is approximated as follows:

$$\tilde{\mathcal{L}}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t),$$

where $g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ and $h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ are the first and second-order gradients of the loss function, respectively.[10] Regularisation in XGBoost also optimises the objective function, as stated previously. It is defined as:

$$\Omega(f_t) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2,$$

where T is the number of leaves in the tree, w_j are the leaf weights, γ penalises additional leaves, and λ controls the regularisation of leaf weights.[10] To construct decision trees efficiently, XGBoost evaluates potential splits by maximising the gain function:

$$\text{Gain} = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma,$$

where G_L and H_L are the sums of first and second-order gradients for the left child and G_R and H_R for the right child, respectively. XGBoost handles sparse data with its split-finding algorithm, reinforcing its proficiency as a regression model.[10] XGBoost was used because it handles non-linear relationships and noise well, both essential properties.

Chapter 3 Results and Conclusion

3.1 Results

All analysis here was done on close price VIX returns. The distribution of VIX returns demonstrates significant volatility, starting in the summary statistics, which display a `-29.98` minimum return and a `76.82` maximum return. A Classical Bootstrap with 2000 resamples was performed to estimate mean, and variance VIX return confidence intervals. See B.3 and B.4 for their plots. The 95% bias-corrected and accelerated confidence intervals from resampling were: `(-0.31, 0.28)` for the mean and `(57.56, 73.67)` for the variance, suggesting near-zero average returns with high uncertainty, reflecting volatility.

BB was conducted to estimate posterior distributions for the mean and median of VIX returns. B.5 shows both statistics' posterior distributions. The median distribution (in blue) is more concentrated, meaning its estimations are more confident, while the broader mean distribution (in red) suggests less confidence in estimations. This plot demonstrates the median's advantage compared to the mean in outliers' presence. Then, a CB and BB comparison is displayed in B.6, where they are overlaid. The similarity between the two methods demonstrates their consistency despite their theoretical differences.

A DPMM was applied for density estimation to capture the non-normal VIX return characteristics. B.7 is a DPMM plot of VIX returns using the `BNPMix package`'s `PYdensity` function, with a singular peak and heavy tails, which is expected of VIX return volatility clustering. The unimodal distribution demonstrates one volatility type dominated 2012-2021. Cluster 3 dominated with 694 observations, returns averaging -0.105, moderate variance for high density, and some extremes due to heavy tails and positive skewness. Despite small DPMM credible intervals elsewhere, there are larger peak intervals, demonstrating lower estimated density certainty. The long tails demonstrate that there are rare extreme market swings, which highlights how well the DPMM can pick up common and uncommon events that shape the data. The DPMM identified a mean of 10 clusters, with the number of clusters ranging from 4 to 16. This range shows how much market conditions can change. The clusters with the most observations were matched to VIX return time series, based on MCMC iterations for stabilisation, see B.8. Each cluster represents a distinct market regime, which are a set of market conditions that describe market behaviour, and moving between clusters represents market sentiment change.

Finally, XGBoost was applied, using DPMM cluster assignments to help predict VIX returns. Its performance was evaluated using the Relative Standardised Residual (RSR), calculated as RSR = RMSE/σ, which was 0.134, indicating excellent performance (since RSR < 0.5). Grid search and cross-validation were tested, but they did not return a good RSR due to over-fitting, so fixed hyper-parameters were used. The learning rate, η, was 0.01, so the model learned gradually, reducing over-fitting risk. The maximum tree depth was 6, and fine-tuned regularisation was used, both to balance capturing complexity and model simplicity. The feature sampling ratio of 0.8 introduced randomness by selecting 80% of the data for each of the 500 boosting rounds, diversifying features in each tree.

3.2 Conclusion and Future Research

MCMC sampling issues arose during the analysis, including early convergence. The burn-in period and iterations solved these by allowing Markov chain stabilisation and a proper parameter space search. In the XGBoost model, hyper-parameter tuning was another challenge because minor changes sometimes significantly increased RSR. `PYRegression` plots were also modelled to understand the open and close returns' relationship, see B.10. There were large, credible intervals at extreme open values, showing lower estimated density certainty. However, the plots were not as interpretable as expected for this report's purpose, which is trader insights, so they were not a main part of this analysis.

This report ultimately demonstrates how effective NP methods are. Still, there were some unexpected results, including significant BB posterior distribution variability for the mean, highlighting the sensitivity of returns and emphasising the point of using the median in volatile markets. The DPMM plotted a unimodal distribution of returns, proving the existence of a dominant market regime volatility. It also returned a lot of clusters, indicating significant market sentiment variability from 2012 to 2021, which is expected because there are so many dynamic factors which play into how well the market is performing. Using these clusters as features for the XGBoost model helped it perform, demonstrating the superiority in combining NP and ML methods for market analysis.

This report can offer trader insights. Market regime clustering analysis can influence traders' strategies by helping them better anticipate and manage market turbulence. Identifying clusters allows them to detect market sentiment shifts, while Bootstrap helps traders estimate the range of likely outcomes in a specific cluster. They can then use XGBoost to predict future market regimes. Traders may use XGBoost to go long in low-volatility clusters and short in high-volatility phases identified by the DPMM.

To take this further, testing other NP clustering methods, such as Hierarchical Bayesian Clustering or Gaussian Mixture Models, may improve clustering. Also, using different economic factors, such as interest rates, as additional XGBoost variables would strengthen current findings and better the understanding of market volatility and its drivers.

Bibliography

1. Justin Kuepper. Cboe volatility index (vix): What does it measure in investing?, August 2024. Reviewed by Samantha Silberstein. Accessed December 18, 2024.
2. Konstantinos Perrakis. Nonparametric statistics: Lecture notes, michaelmas term, 2024. Last updated December 5, 2024. Accessed December 13, 2024.
3. J Martin Bland and Douglas G Altman. Statistics notes: Bootstrap resampling methods. *BMJ*, 350:h2622, 2015.
4. Eric P. Xing, Carl Malings, and Jingkun Gao. Bayesian nonparametrics: Dirichlet processes. *10-708: Probabilistic Graphical Models*, 2014. Lecture notes from 10-708, Spring 2014.
5. Amr Khaled Khamees, Almoataz Y. Abdelaziz, Ziad M. Ali, Mosleh M. Alharthi, Sherif S.M. Ghoneim, Makram R. Eskaros, and Mahmoud A. Attia. Mixture probability distribution functions. *Ain Shams Engineering Journal*, 13(3):101613, 2022.
6. Dilan Görür and Carl Edward Rasmussen. Dirichlet process gaussian mixture models: Choice of the base distribution. *Journal of Computer Science and Technology*, 25(4):653–664, 2010.
7. Jaeyong Lee and Steven N MacEachern. A proof of the stick-breaking representation of dirichlet processes. *Journal of the Korean Statistical Society*, 49:389–394, 2020.
8. Tamara Broderick, Michael I Jordan, and Jim Pitman. Beta processes, stick-breaking and power laws. *Journal of the Korean Statistical Society*, 2012.
9. Radford M Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.
10. Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM sigkdd International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

Chapter A Appendix: Key Values

Table A.1: Key Figures from the Results Section

Method	**Value**
Exploratory Data Analysis	
Minimum Return	-29.98
Maximum Return	76.82
Interquartile Range	[-4.44, 3.60]
Bootstrap Methods	
Bootstrap Mean (95% CI)	(-0.31, 0.28)
Bootstrap Variance (95% CI)	(57.56, 73.67)
Non-Parametric Analysis	
Range of DPMM Clusters	4-16
Average DPMM Clusters	10.07
XGBoost	
Learning Rate (η)	0.01
Max Depth	6
Subsample Ratio	0.8
Rounds	500
Relative Standardised Residual (RSR)	0.134

Chapter B Appendix: Images

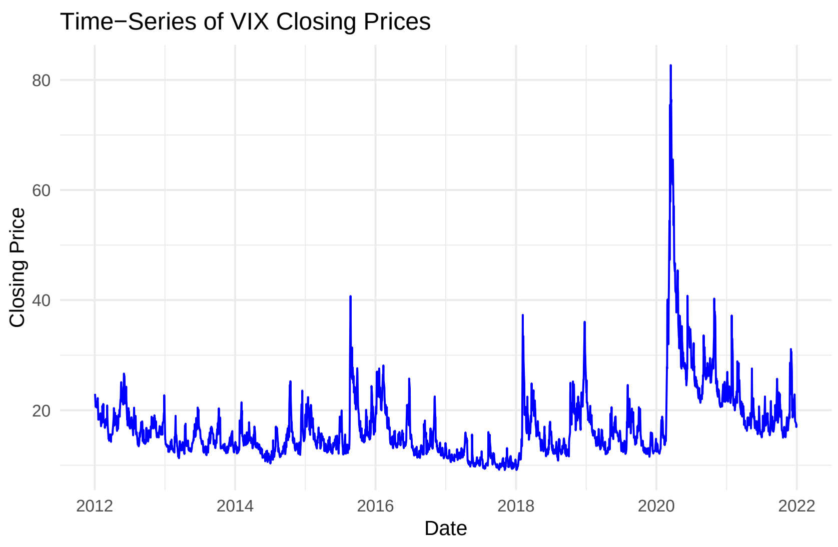

Figure B.1: Initial Time Series

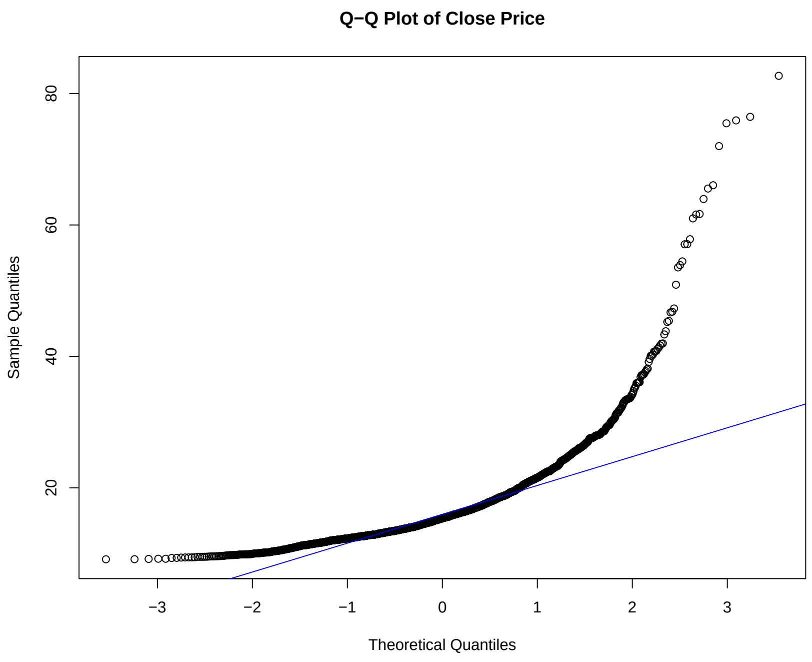

Figure B.2: Close Price QQPlot

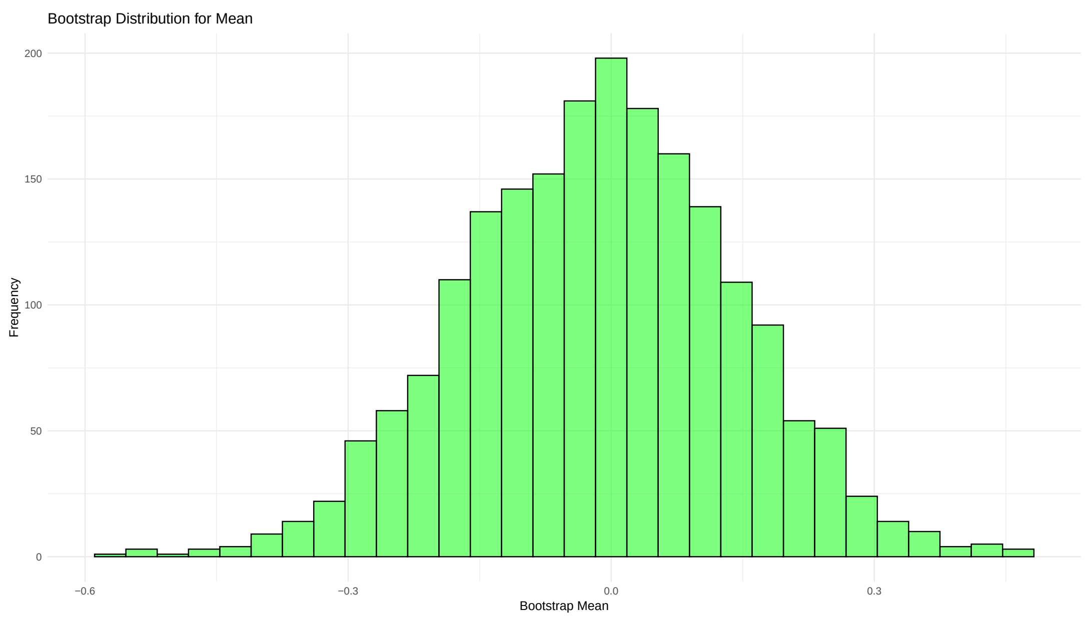

Figure B.3: Mean Bayesian Bootstrap Distribution

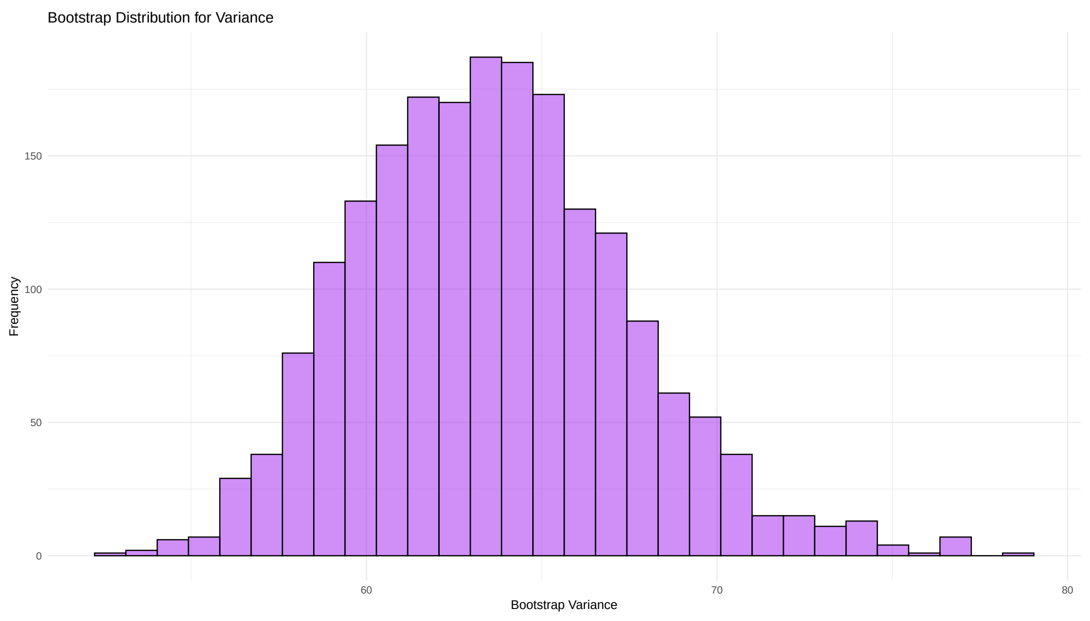

Figure B.4: Variance Bayesian Bootstrap Distribution

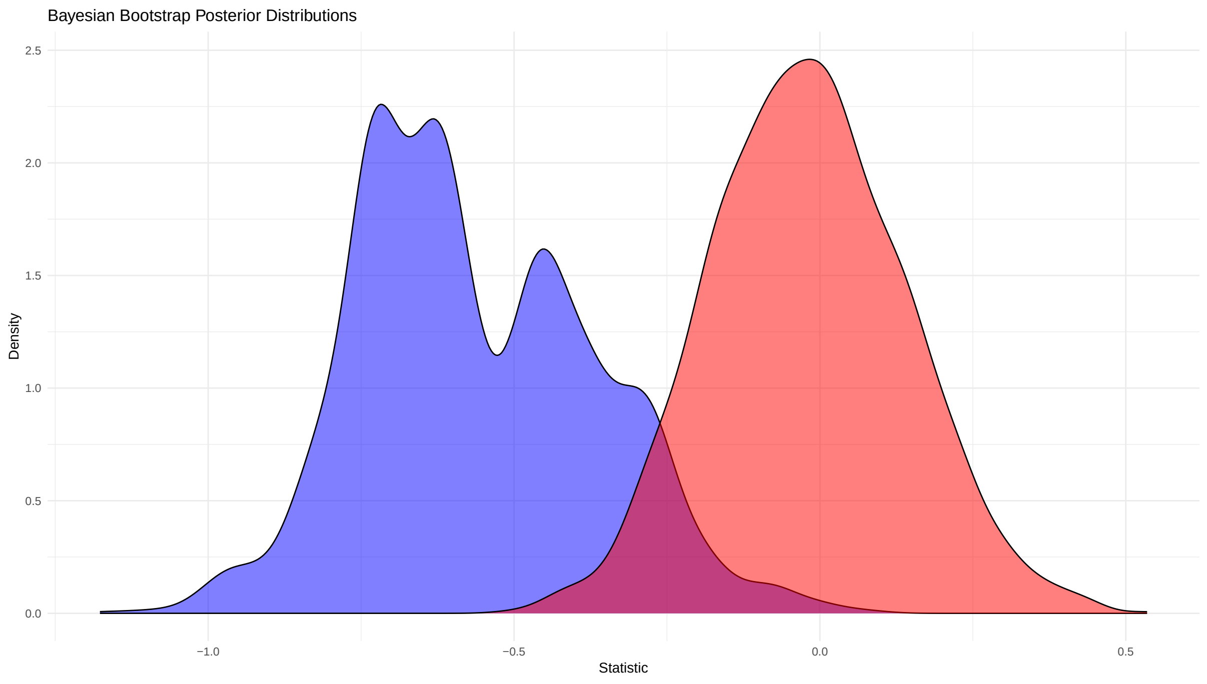

Figure B.5: Bayesian Bootstrap Mean and Median Posterior Distributions

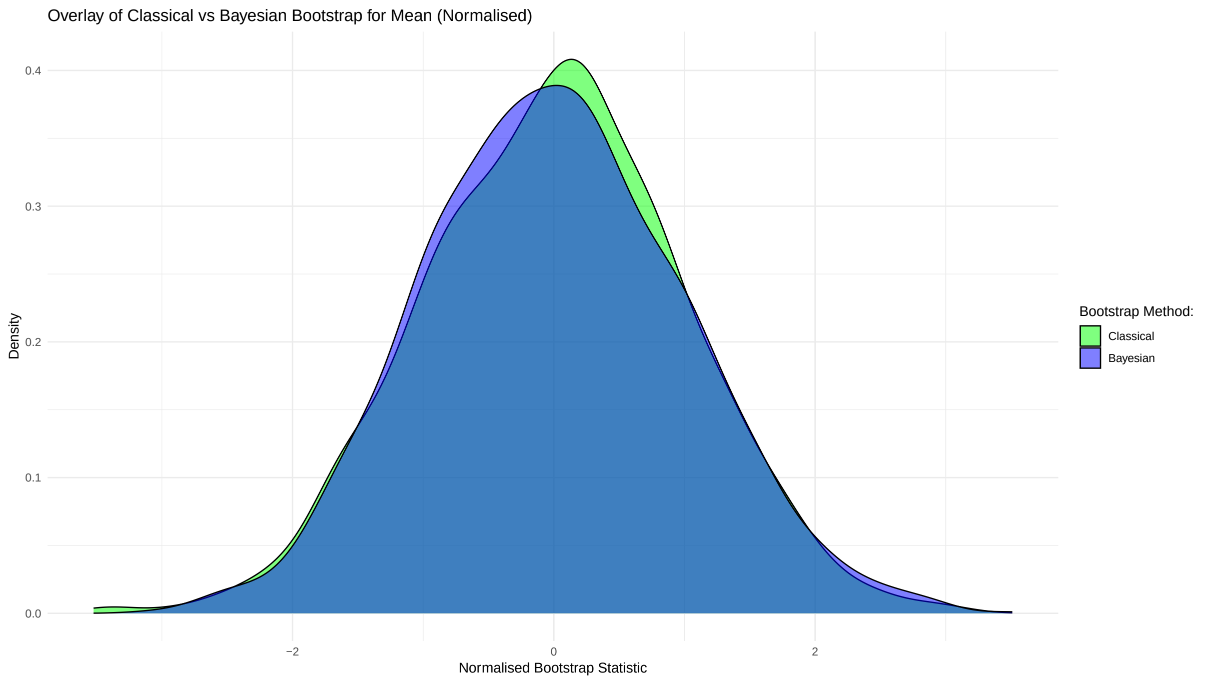

Figure B.6: Classical and Bayesian Bootstrap Overlay

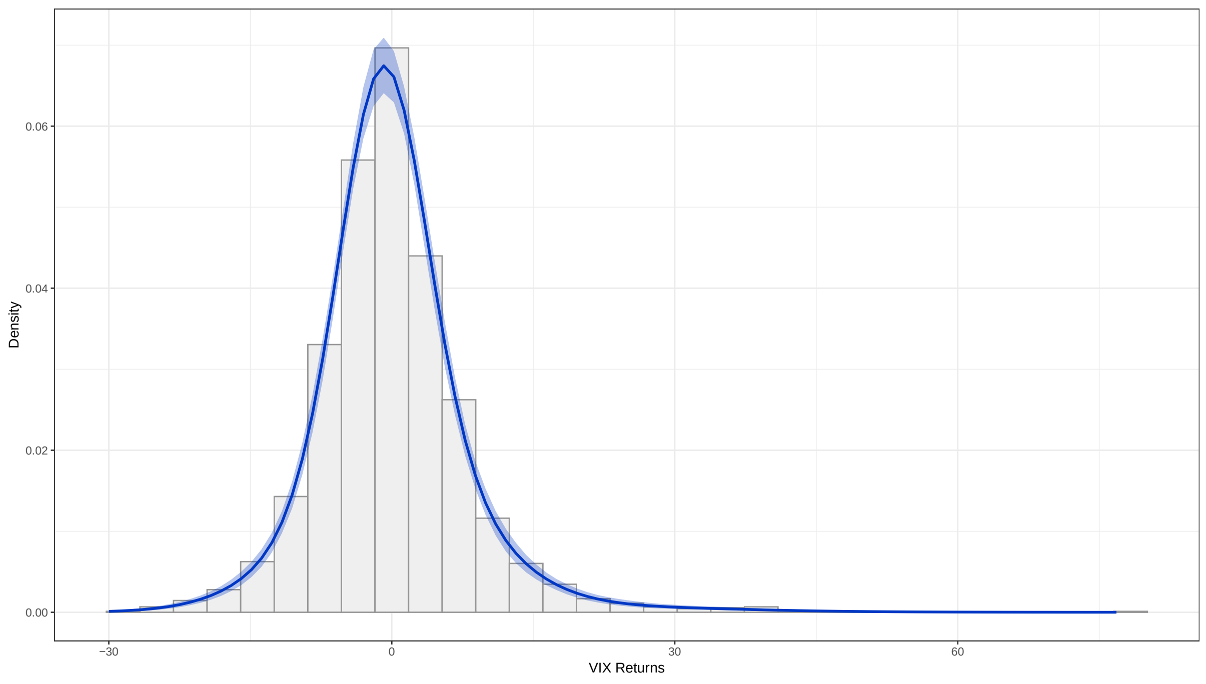

Figure B.7: DPMM Posterior Density

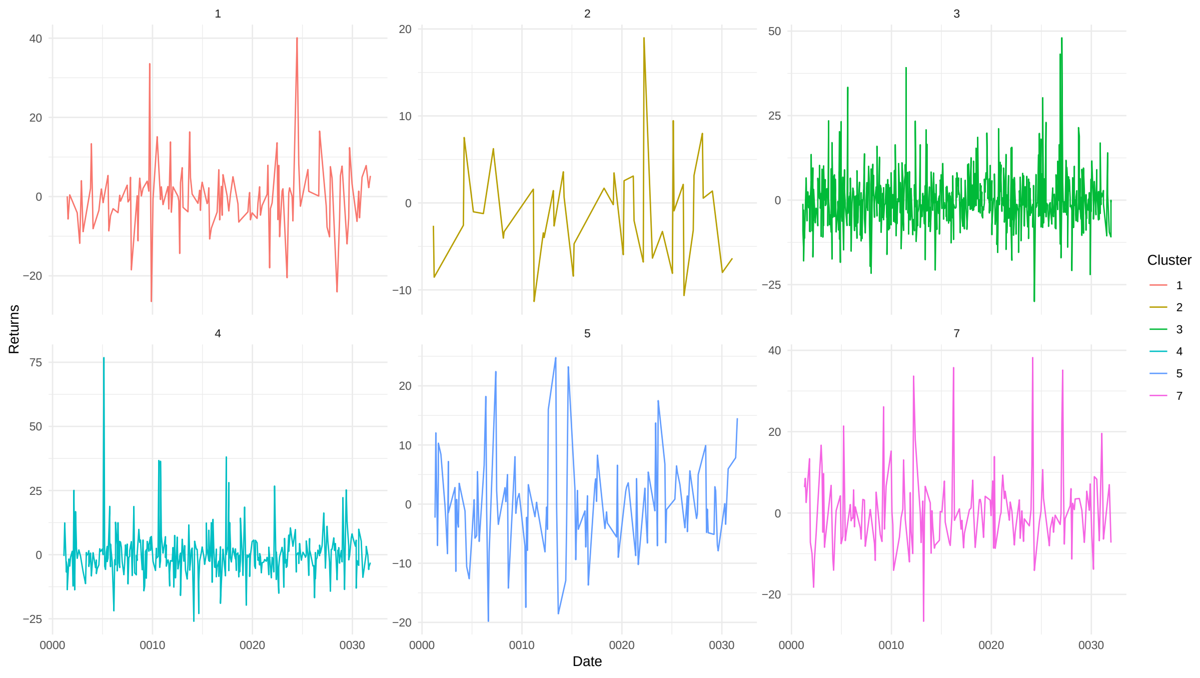

Figure B.8: Time Series for common DPMM Clusters

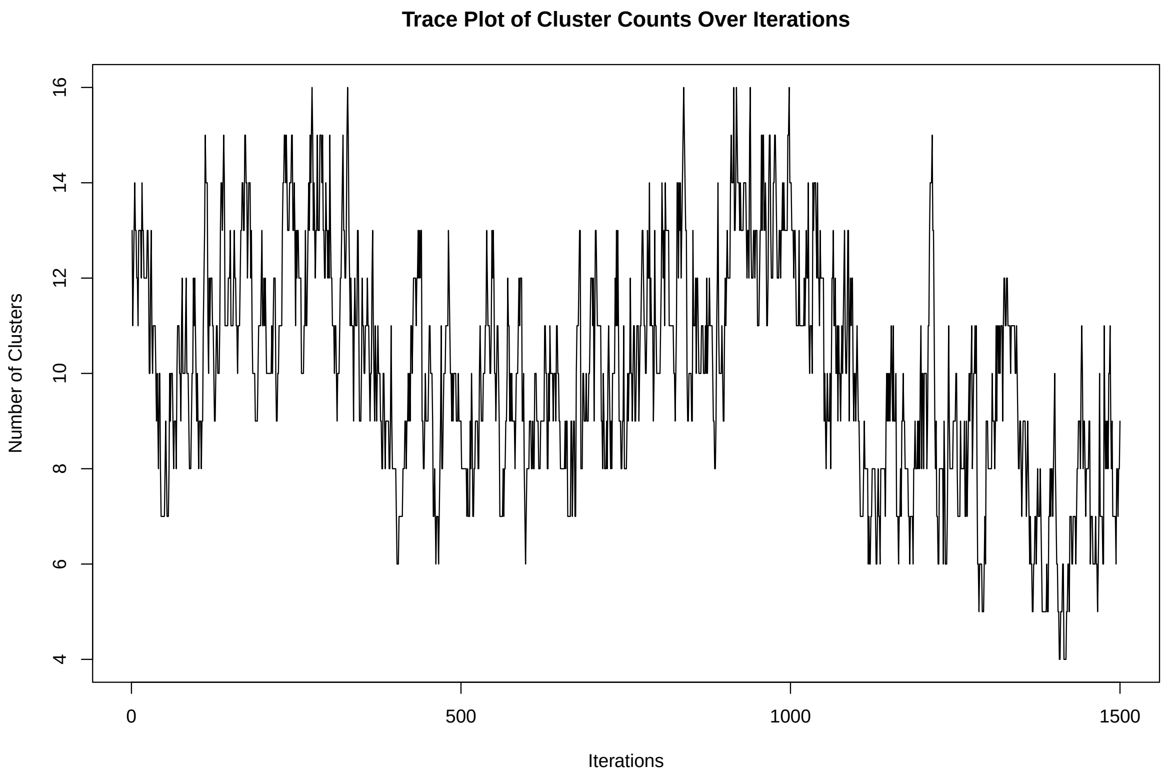

Figure B.9: Trace Plot of DPMM Cluster Counts Over Iterations

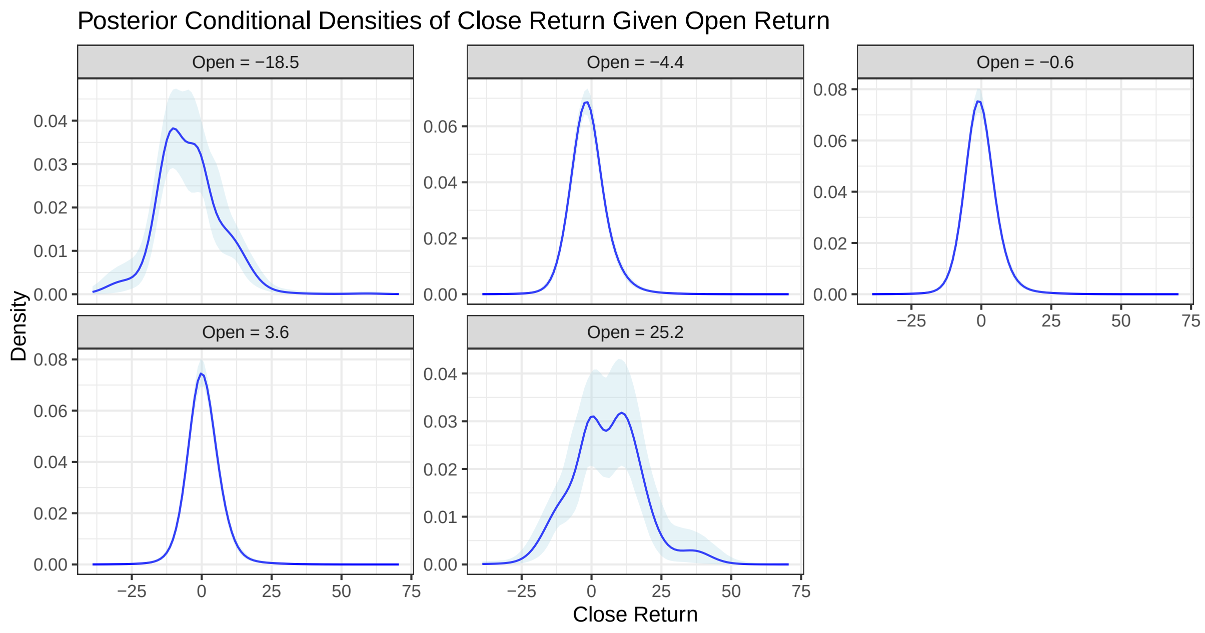

Figure B.10: Posterior Conditional Densities of Close VIX Returns Given Open VIX Returns