

Bayesian analysis of Spotify song data

Mckay Jensen,¹ April 2020

Introduction

In this project, I use a Bayesian linear regression to determine which features of songs are associated with popularity on Spotify. I also fit a Bayesian logistic regression to determine which song features are associated with me liking a song. Although this analysis does not answer any pressing economic question, I chose it as an interesting demonstration of using Bayesian statistical analysis to find trends in a reasonably large data set with many descriptive features.

Spotify uses a sophisticated machine learning algorithm to generate song recommendations to its users (Jacobson et al. 2016); while the details of this algorithm are not available to users, it's likely that the underlying model is some sort of deep neural network as is common in user recommendation systems for online content (see, for example, Covington et al. 2016 for YouTube's approach to content recommendation). In any case, Spotify makes publicly available a large amount of data on the musical features of its songs, which presumably are important inputs to its recommendation system; I use these features to get an idea of which features are associated with more popular songs and with my own music preferences. The models I use in this project are not intended to replicate the functionality or predictive power of the system that Spotify uses -- although deep neural networks are good for classification or prediction since they can flexibly approximate almost any function (Liang & Srikant 2017), I am more interested in creating simpler, more easily interpretable models from whose posterior parameter distributions I can feasibly sample.

As far as I know, no music streaming service uses Bayesian inference to recommend content, so this is a novel approach to the problem. I have found evidence that Microsoft's discontinued Groove radio service used variational Bayesian methods to construct playlists for its users (Ben-Elazar et al. 2017), although that is not quite the same as predicting popularity or personal preference, and in any case the computational method I use here is sampling from the posterior distribution via Markov chain Monte Carlo, not variational inference.

¹ The Python code I used for this project, including data collection and cleaning, and model fitting is available at <https://github.com/quevivasbien/bayesian-spotify>.

Data collection

I used Spotify's developer API² to collect data on over 50.000 songs. The API interface makes it difficult to collect a truly random sample of music, so the sample I used was instead every song in Spotify's library from artists who had at least one song on the weekly Spotify top 200 global charts³ between 30 August 2019 and 28 February 2020. This should be taken into account when analyzing the results I find in this project: the results I draw about popularity and personal affinity should be understood as applying to artists who have had at least some success recently in their musical career.

I collected the data on 11 March 2020. For that reason, the data set does not contain any songs released after that date, and the popularity measures are based on popularity as of that date. I collected 18 features for each song in the data set; a description of each feature is found in Appendix 1. The features used comprise every characteristic of songs that is publicly available via the Spotify developer API and should therefore provide a reasonably precise characterization of each song in the data set. Any data with missing or invalid entries for any feature were removed prior to analysis.

The code used to extract and clean these data is available on [my GitHub repository for this project](#).

² See <https://developer.spotify.com> I used the Python package "spotipy" (<https://pypi.org/project/spotipy>) to make HTTP requests easier.

³ <https://spotifycharts.com/regional>

I. Linear regression for popularity analysis

Model specification and inference

The first analysis I carried out was designed to answer the question, “What song features are associated with popularity on Spotify?” To do this, I estimated the model

$$\text{popularity}_i = \alpha + \sum_{k=1}^{17} \beta_k [\text{feature}]_{ki} + \epsilon_i$$

with the [feature] variables being the features beside popularity listed in Appendix 1. I assumed that the ϵ_i were i.i.d. distributed as scaled student's t random variables with degrees of freedom ν and scale parameter σ . Because the degrees of freedom of a student's t distribution is related to its variance, I was able to make my MCMC sampler faster and more stable by defining an auxiliary parameter σ^* (on which I set a prior) and letting

$$\sigma = \sigma^* \sqrt{\frac{\nu - 2}{\nu}}.$$

I was also able to greatly improve the convergence of my sampler by normalizing each feature to have zero sample mean and sample standard deviation 1. With this change, the model I estimated can be rewritten as

$$y = \alpha_{std} + X_{std} \beta_{std} + \epsilon$$

where y is the vector of popularity values, and X_{std} is the matrix of normalized data. Note that the original β can be retrieved from the β_{std} simply by dividing by the original sample standard deviation of each feature. Using the standardized data and coefficients is also useful because the scale and location of many of these variables is rather arbitrary anyway, and standardization allows me to more directly compare effect sizes.

Near-multicollinearity in the data can also introduce numerical problems and slow convergence. Because of this, I computed the QR decomposition of X_{std} and estimated the θ vector in the model

$$y = \alpha_{std} + Q\theta + \epsilon$$

at which point β_{std} can simply be computed as $\beta_{std} = R^{-1}\theta$. This works better because Q is an orthogonal matrix, meaning its columns are orthogonal, so there shouldn't be problems with correlated variables.

Because I had so much data, the estimated model wasn't too sensitive to the choice of priors. The priors I used were chosen to be nearly uninformative while still allowing for reasonably fast convergence:

$$\begin{aligned}\alpha &\sim N(\bar{y}, 10) \\ \theta_k &\sim N(0, 1000), \quad k = 1, \dots, 17 \\ \sigma^* &\sim \text{HalfCauchy}(0, 5) \\ \nu &\sim \text{Gamma}(2, 0.1)\end{aligned}$$

I fit the model in Stan using 2 chains of 10.000 iterations, with a 1.000 iteration warm-up period.

Model checks⁴

Trace plots for the post-warm-up period are shown in Appendix 2; the trace plots included are for the θ vector rather than the β , since the Markov chain of θ is what is actually directly computed. Visual inspection of the trace plots does not indicate any problems (they look sufficiently “fuzzy”). Geweke scores computed for every chain comparing the sample means of the first 10% and last 50% of each chain are all very small (magnitudes less than 0.05 across the board), suggesting that the chains have converged. Sample means of the two chains are also very close (as have their variances, as indicated by the potential scale reduction statistic $\hat{R} = 1.0$). To assess the model’s predictive power, I also fit the model on a subset of the full data set (40.000 songs, approximately 80% of the data) and then computed goodness-of-fit statistics for the posterior predictions on both the data used to fit the model and the excluded data. These statistics indicate that the model fits the data reasonably well and that its predictive power is nearly as good on out-of-sample data as on data used for fitting. A full presentation of the goodness-of-fit tests I did is in Appendix 8.

Results

Summary statistics for the standardized and unstandardized β coefficients are shown in Appendix 3a (summary statistics for other parameters are in Appendix 3b). The standardized coefficients give an idea of the relative importance of each variable, while the unstandardized coefficients indicate the expected change in popularity due to a change in each variable. Unsurprisingly, the largest standardized coefficient is for the availability variable (songs available in more countries are more likely to be more popular globally). More interestingly, the next strongest association seems to be with loudness (louder songs are substantially more popular). The next largest standardized coefficients are track number (popular songs tend to

⁴ Detailed diagnostic statistics, as well as the code for generating plots and tables, are available in this Jupyter notebook: https://github.com/quevivasbien/bayesian-spotify/blob/master/make_graphs.ipynb

appear near the beginning of an album), explicit (popular songs tend to be explicit), and valence (popular songs tend to be less upbeat), in that order. The only variable with an ambiguous relationship to popularity (95% confidence interval containing zero) seems to be speechiness. Histograms for α , β , v , and σ are shown in Appendix 4.

II. Logistic regression for prediction of my liked songs

Model specification and inference

The second analysis I did was designed to answer the question, “What song features make me more likely to like a song?” To answer this, I pulled the feature data for each of the songs in my “liked songs” playlist (180 songs total), in addition to the feature data I had already collected. I then fit the following logistic regression model:

$$P(\text{song } i \text{ is one of my liked songs}) = \frac{1}{1 + e^{-\eta_i}}$$

where

$$\eta_i = \alpha + \sum_{k=1}^{18} [\text{feature}]_k \beta_k.$$

The [feature] variables here are each of the variables in Appendix 1 (*including* popularity). The likelihood for this model is a Bernoulli likelihood with parameter $1/(1 + e^{-\eta_i})$. I did not do an intermediate QR decomposition in this case, but I did again find that standardizing the data (in the same way as described before) helped the MCMC chains to converge much faster. That is, I computed standardized versions of each feature and used

$$\eta_i = \alpha_{std} + \sum_{k=1}^{18} [\text{feature}]_{std,k} \beta_{std,k}$$

retrieving the unstandardized coefficients after the fact. I used uninformative priors for all the parameters.

Because I had only 180 songs in the “liked songs” category, it wasn’t necessary to include the entire data set of more than 50,000 songs when fitting the model; past a certain point, the only measurable effect of including more data would be to decrease the α parameter representing the base rate for the proportion of liked songs in the data (I tested this), and I was only interested in the β parameters. Because of this, I included only a random sample of 2,000 songs in the data along with the liked songs. I again fit the model in Stan using 2 chains of 10,000 iterations, with a 1,000 iteration warm-up period.

Model checks

Trace plots for the parameters are in Appendix 5. The Geweke scores (calculated again using the first 10% and last 50% of each chain) are all very small (< 0.03 for all chains), and the \hat{R} statistics are all essentially 1, indicating that the chains have all most likely converged. To get an idea of

the model's predictive power, I re-fit the model on 5 subsets of the full data set, with each excluding fifteen samples from my liked songs and fifteen from the rest of the data (a sort of k-fold cross-validation). I then computed accuracy scores on the excluded data by drawing repeatedly from the posterior distributions for each fitted model and averaging the number of correct Bernoulli draws across each sample in the excluded data. The average accuracy score was about 0.72, indicating that the model performs significantly better than random chance (which would be 50/50) although there is certainly room for improvement. It would probably require a larger sample size with more features and a more complicated model to make highly accurate predictions; of course, my intent here was to explore the factors that play into my tastes, not design a model that can predict my favorite songs very accurately (at the cost of being more difficult to interpret). A full description of the predictive tests I did is in Appendix 8.

Results

Summary statistics for the β parameters are shown in Appendix 6. The uncertainty in the estimates is much higher than in the linear model, which is not surprising: the model is fundamentally different, and there is much less data to inform the posterior distribution. The variable that seems to have the greatest impact on my probability of liking a song is `track_number` (I'm more likely to like songs near the beginning of an album), which is not surprising. The next-strongest association is with the `explicit` variable (I'm less likely to like songs marked as explicit). The next most clear associations are with `loudness` (I'm less likely to like loud songs), `liveness` (I'm less likely to like songs performed live), and `popularity` (I'm more likely to like songs that are popular). Availability doesn't seem to be nearly as important as it was in the linear popularity model -- I display only a weak propensity to like songs that are more widely available. Histograms for each parameter are shown in Appendix 7.

Conclusion

The linear model for analysis of popular songs was able to provide precise estimates of the degree to which various song features are associated with popularity among Spotify users. The logistic model for analysis of my own musical preferences was also able to provide reasonably precise inferences about the song features common in my liked songs relative to other songs.

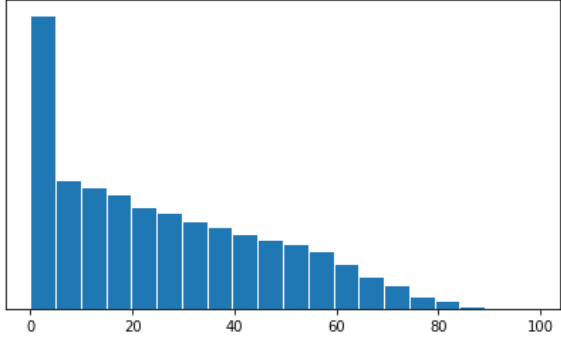
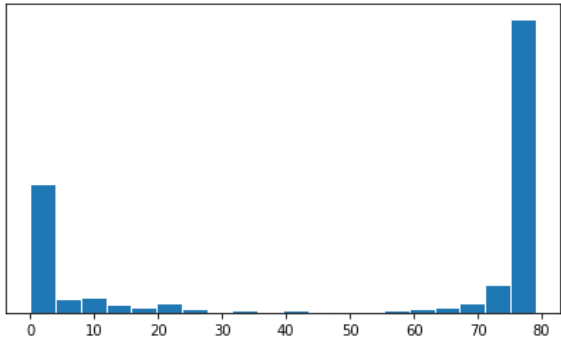
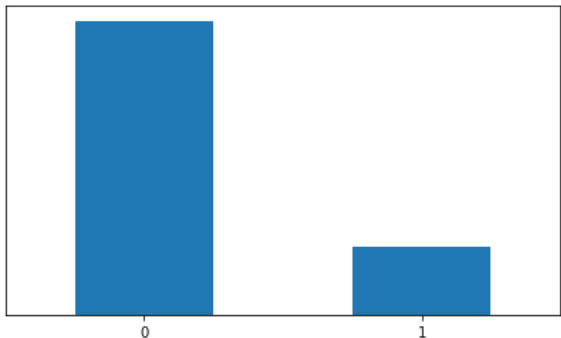
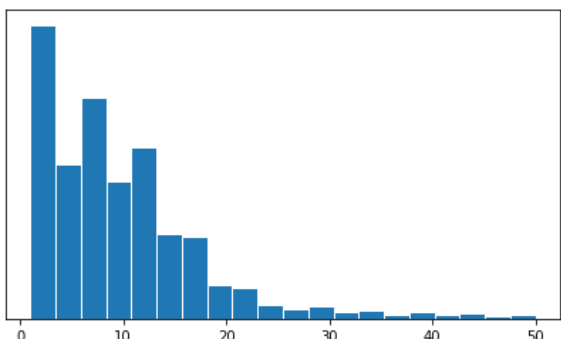
Both of these fitted models could be used to make out-of-sample predictions, which, although they may not be as accurate as the sophisticated models used by Spotify and other companies in the business of content recommendation, have the advantage of having fully defined posterior parameter distributions that can be used to give degrees of confidence and confidence intervals for the predictions they make. Although the logistic model I fitted was based on my personal preferences, the same model could be fitted using any Spotify user's data and could provide an interesting alternative to Spotify's song recommendation system. For that purpose, it would probably be necessary to collect more data on users than what I used here and employ a more complicated statistical model than just a logistic regression. I have experimented [elsewhere](#) with using Bayesian neural networks for binary and categorical classification, and it seems like a very promising approach for this kind of task. Although the computational constraints of the MCMC approach taken for the models used in this project make it difficult to fit such models using the same procedure, it is possible to use variational Bayesian inference (see Kingma et al. 2015) or probabilistic backpropagation (see Hernández-Lobato & Adams 2015) as computationally tractable alternatives.⁵

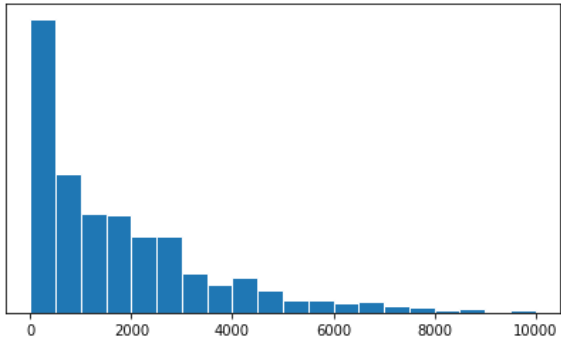
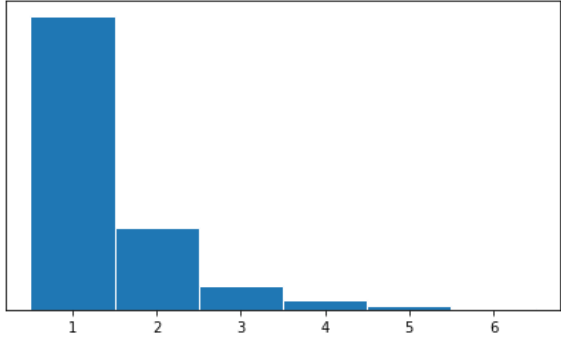
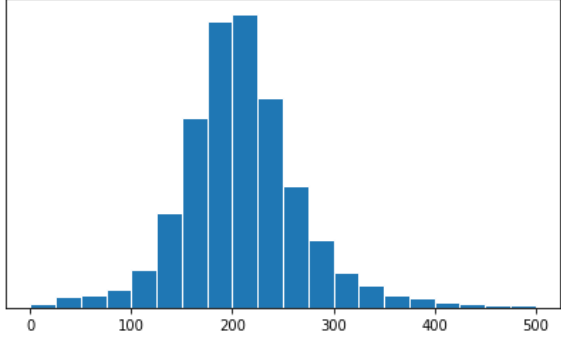
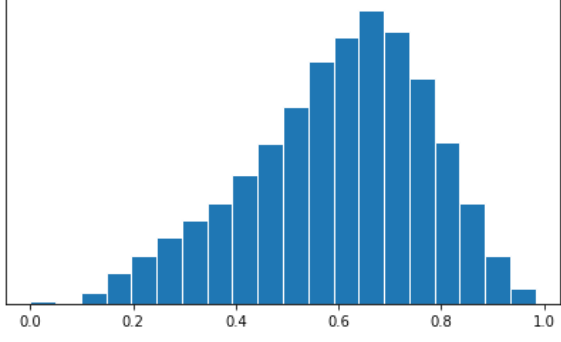
⁵ Variational methods generally seem to provide a good approximation of the posterior distribution that is useful when the ability to fit a model quickly is helpful or when the number of latent variables and/or model complexity are high. For example, I was able to fit the same linear model I used in this project with a multivariate normal variational distribution in 1/10 the time and with basically identical results -- see the charts in [this notebook](#).

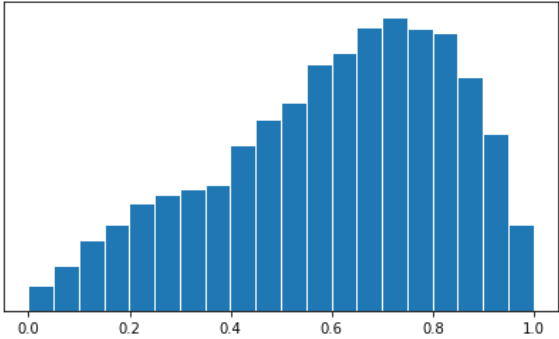
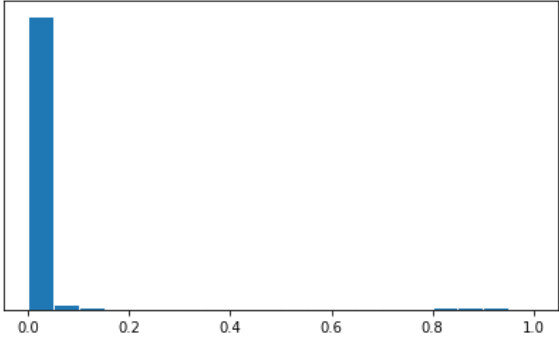
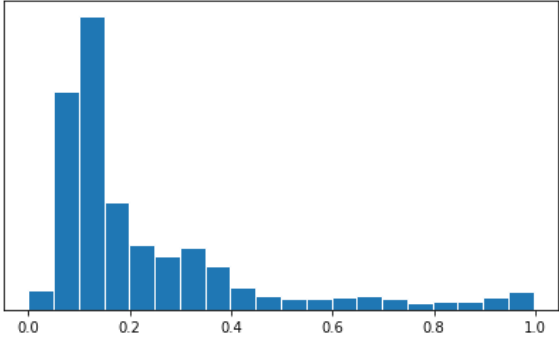
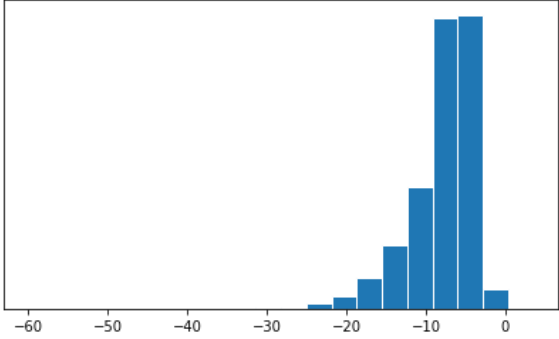
Works cited

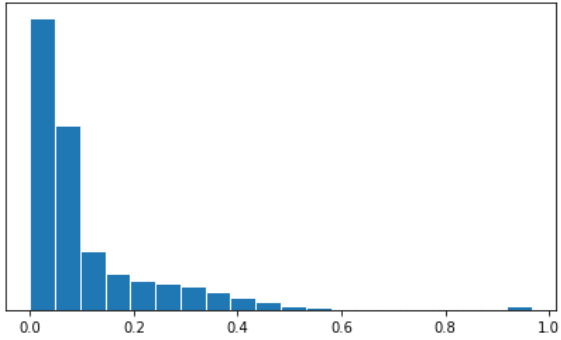
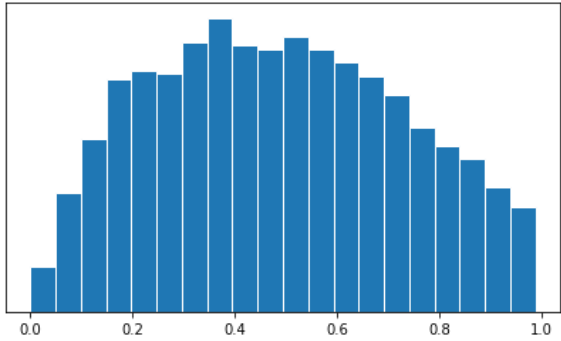
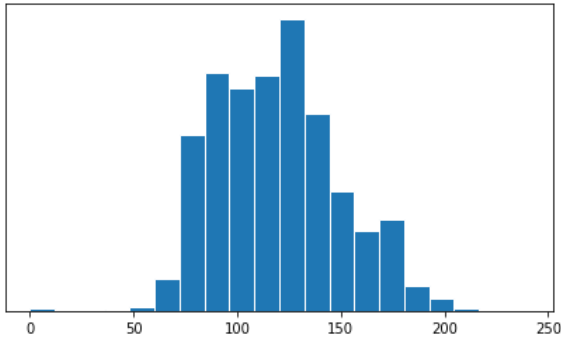
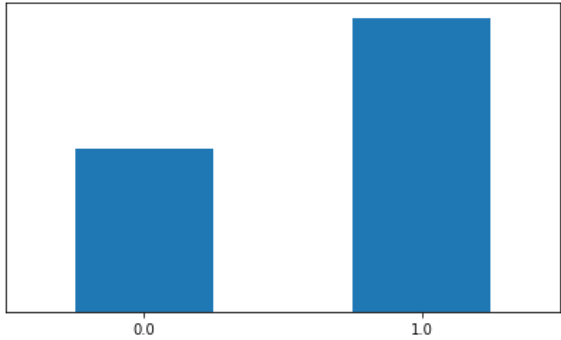
- Ben-Elazar, Shay; Lavee, Gal; Koenigstein, Noam; Barkan, Oren; Berezin, Hilik; Paquet, Ulrich; and Zaccai, Tal. "Groove Radio: A Bayesian Hierarchical Model for Personalized Playlist Generation." *WSDM '17: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 445-453. February 2017.
- Covington, Paul; Adams, Jay; and Sargin, Emre. "Deep Neural Networks for YouTube Recommendations." *RecSys '16: Proceedings of the 10th ACM Conference on Recommender Systems*, 191-198. September 2016.
- Hernández-Lobato, José Miguel and Adams, Ryan. "Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks." *International Conference on Machine Learning*, 1861-1869. June 2015.
- Jacobson, Kurt; Murali, Vidhya; Newett, Edward; Whitman, Brian; and Yon, Romain. "Music Personalization at Spotify." *RecSys '16: Proceedings of the 10th ACM Conference on Recommender Systems*, 373. September 2016.
- Kingma, Diederik; Salimans, Tim; and Welling, Max. "Variational Dropout and the Local Parameterization Trick." 2015. <https://arxiv.org/abs/1506.02557>
- Liang, Shiyu and Srikant, R. "Why Deep Neural Networks for Function Approximation?" Presented at 5th International Conference on Learning Representations (ICLR). 2017.

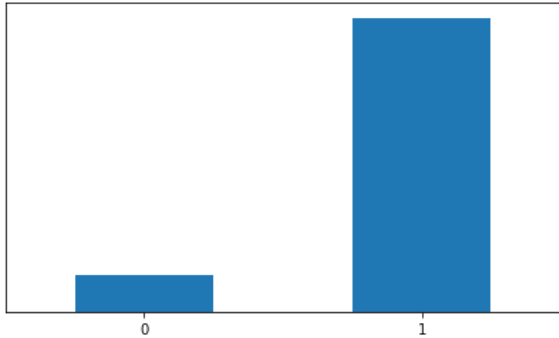
Appendix 1: Song features in the data set

Feature	Description	Distribution
popularity	measure of popularity, combining number and recency of listens, relative to other songs; from 0 to 100	
availability	number of countries in which the song is available	
explicit	1 if the song is marked as explicit, 0 otherwise	
track_number	the song's track number on the album it appears in; 1 if the song is a single	

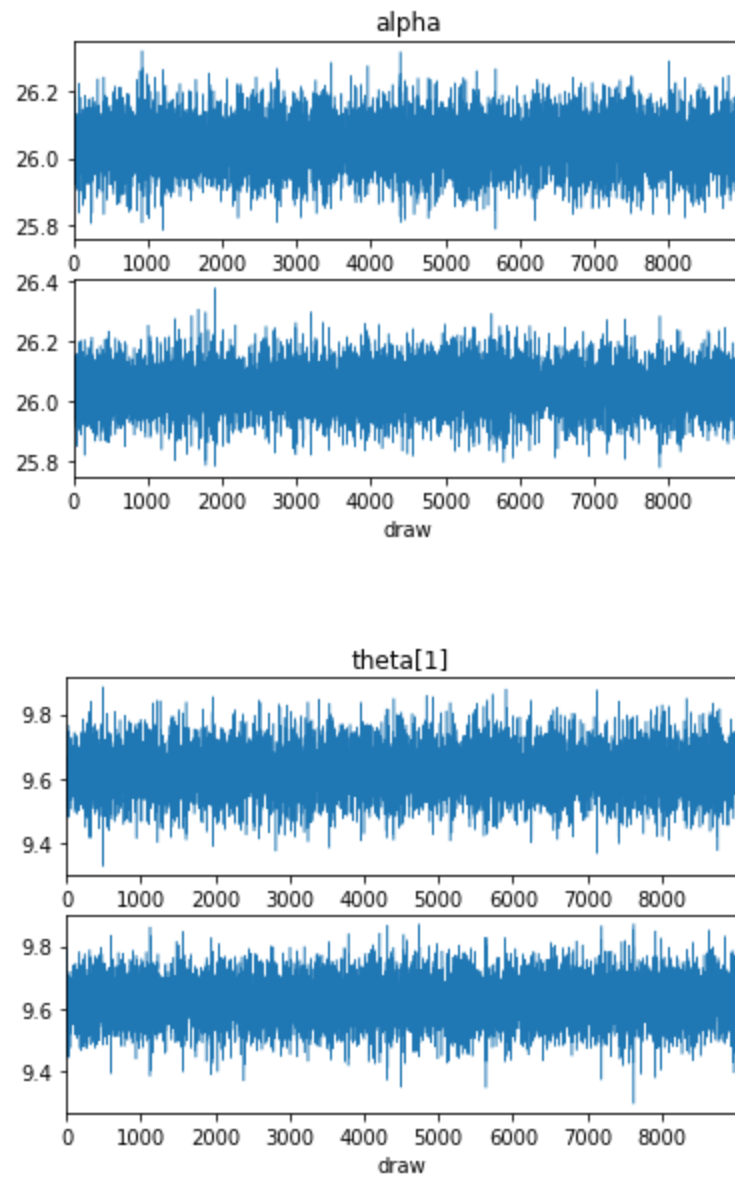
days_since_release	number of days between the song's release date and 11 March 2020	 <p>A histogram showing the distribution of days since release. The x-axis ranges from 0 to 10,000 with major ticks every 2,000 units. The y-axis represents frequency. The distribution is highly right-skewed, with the highest frequency occurring in the first bin (0-500 days), followed by a gradual decline as the number of days increases.</p>
num_artists	the number of artists who collaborated on the song	 <p>A histogram showing the distribution of the number of artists who collaborated on the song. The x-axis ranges from 1 to 6 with major ticks at each integer. The y-axis represents frequency. The distribution is highly right-skewed, with the highest frequency occurring for 1 artist, followed by a sharp drop for 2 artists, and very low frequencies for 3 or more artists.</p>
duration_s	the song's length in seconds	 <p>A histogram showing the distribution of song duration in seconds. The x-axis ranges from 0 to 500 with major ticks every 100 units. The y-axis represents frequency. The distribution is roughly bell-shaped and centered around 200-220 seconds, with a slight right skew. Most songs fall between 100 and 400 seconds.</p>
danceability	value between 0 and 1 meant to measure how good the song is for dancing	 <p>A histogram showing the distribution of danceability scores. The x-axis ranges from 0.0 to 1.0 with major ticks every 0.2 units. The y-axis represents frequency. The distribution is roughly bell-shaped and centered around 0.7, with a slight right skew. Most songs have danceability scores between 0.4 and 0.9.</p>

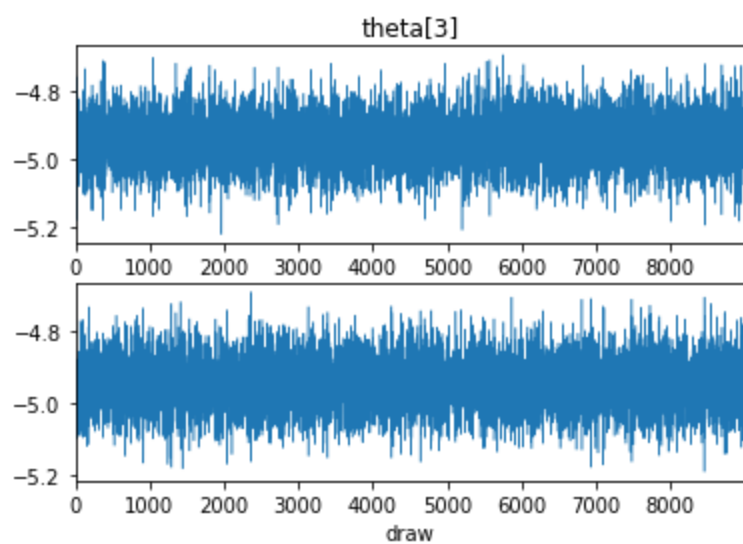
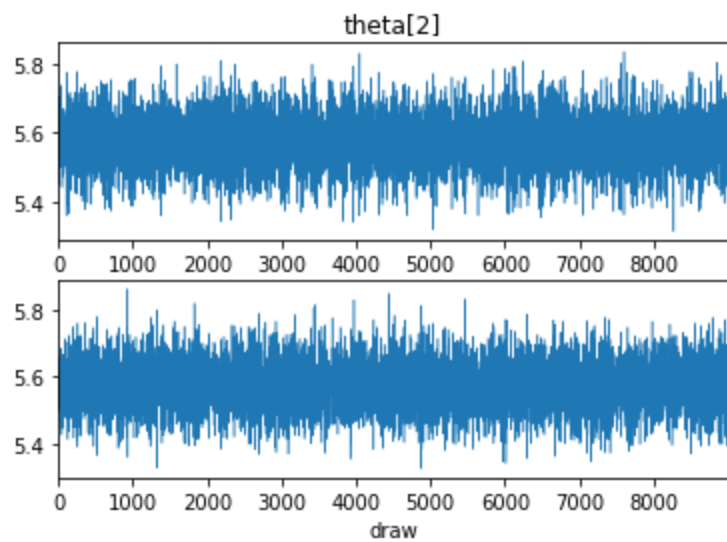
energy	value between 0 and 1 meant to measure how intense and “active” the song is	 <p>A histogram showing the distribution of energy values. The x-axis ranges from 0.0 to 1.0 with major ticks every 0.2. The y-axis represents frequency. The distribution is unimodal and slightly right-skewed, peaking around 0.75 with a frequency of approximately 15.</p>
instrumentalness	value between 0 and 1, representing Spotify’s probability prediction that the song contains no vocals	 <p>A histogram showing the distribution of instrumentalness values. The x-axis ranges from 0.0 to 1.0 with major ticks every 0.2. The y-axis represents frequency. The distribution is highly concentrated near 0.0, with a single bar at 0.0 having a frequency of approximately 15, and very few values elsewhere.</p>
liveness	value between 0 and 1, representing Spotify’s probability prediction that the song was performed live	 <p>A histogram showing the distribution of liveness values. The x-axis ranges from 0.0 to 1.0 with major ticks every 0.2. The y-axis represents frequency. The distribution is unimodal and left-skewed, peaking around 0.15 with a frequency of approximately 15, and tapering off towards 1.0.</p>
loudness	the song’s overall loudness in decibels	 <p>A histogram showing the distribution of loudness values in decibels. The x-axis ranges from -60 to 0 with major ticks every 10 units. The y-axis represents frequency. The distribution is unimodal and left-skewed, peaking around -5 dB with a frequency of approximately 15, and tapering off towards -60 dB.</p>

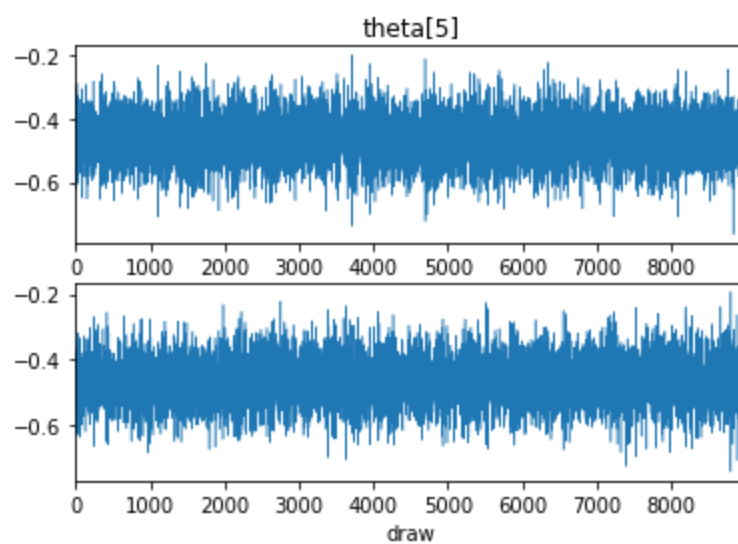
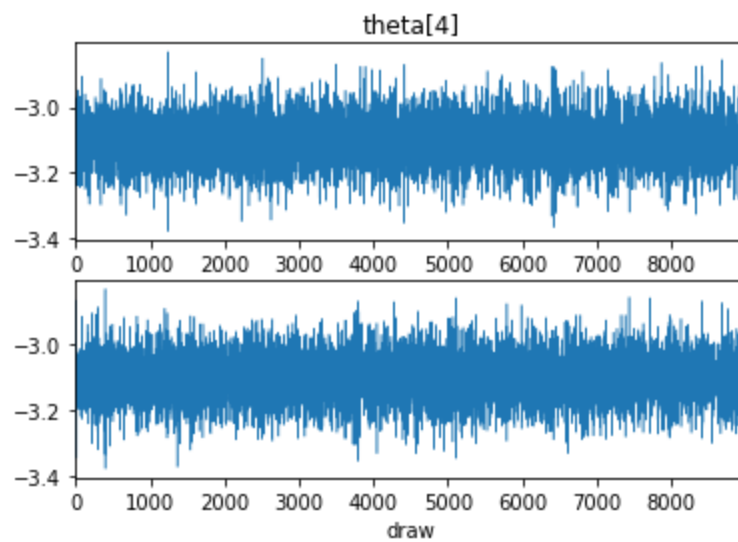
speechiness	value between 0 and 1, measures how much speech there is in the song	
valence	value between 0 and 1, meant to measure how emotionally positive the song sounds	
tempo	the song's tempo in beats-per-minute	
mode	1 if the song is in a major key, 0 if it is in a minor key	

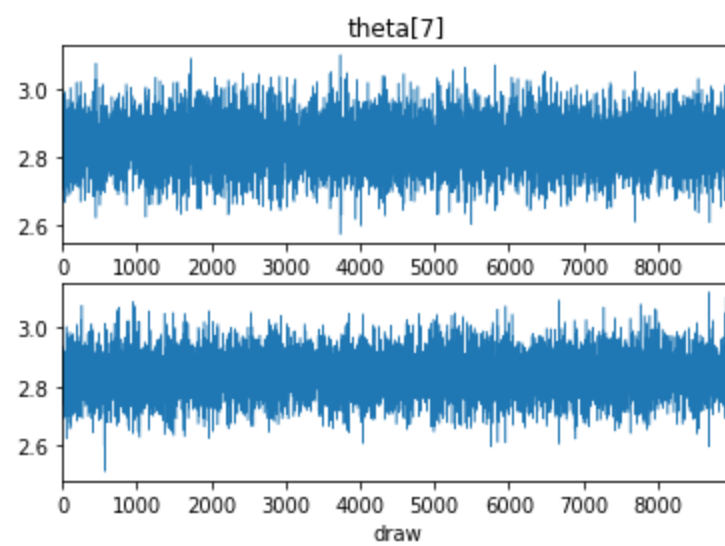
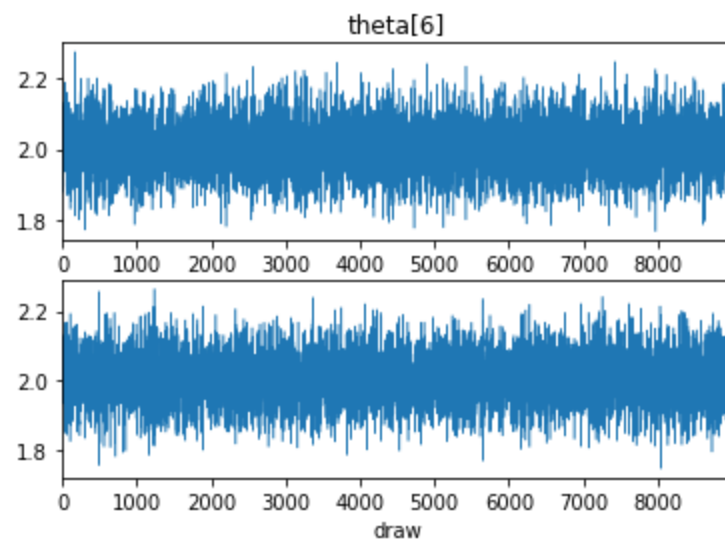
time_signature_4	1 if the song has four beats per measure, 0 otherwise	 <p>A bar chart illustrating the frequency of the variable 'time_signature_4'. The x-axis represents the values 0 and 1. The bar for 0 is significantly shorter than the bar for 1, indicating a much higher frequency for the value 1.</p> <table><tr><th>Value</th><th>Frequency</th></tr><tr><td>0</td><td>Low</td></tr><tr><td>1</td><td>High</td></tr></table>	Value	Frequency	0	Low	1	High
Value	Frequency							
0	Low							
1	High							

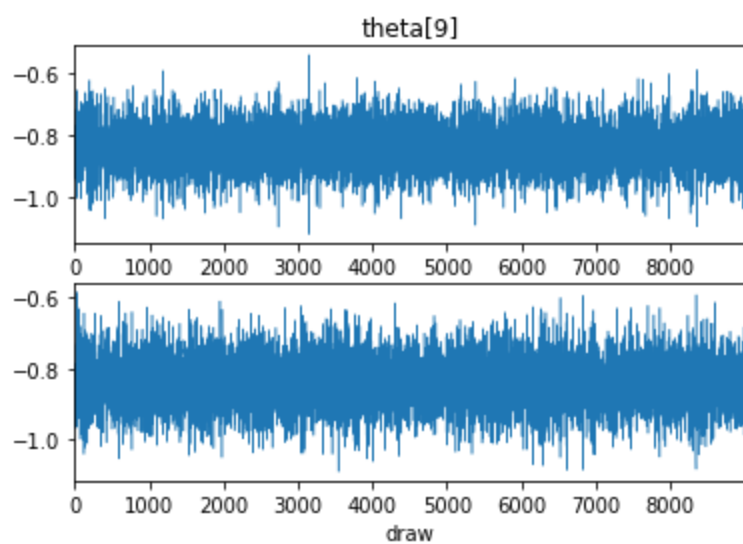
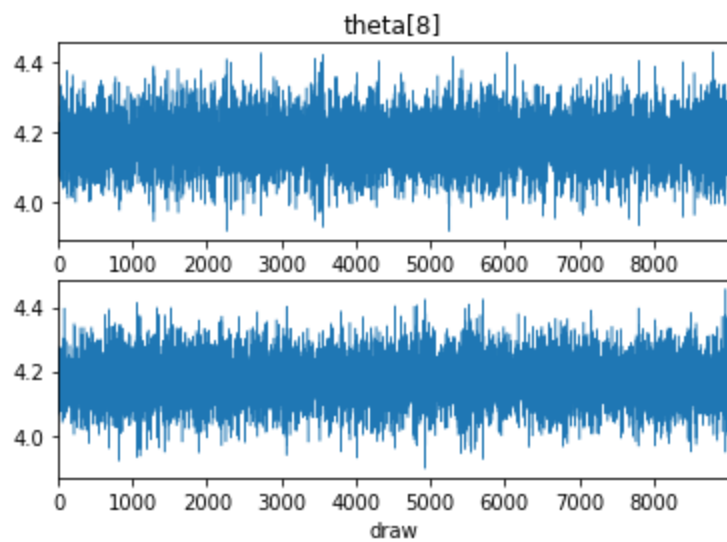
Appendix 2: Trace plots for linear model MCMC

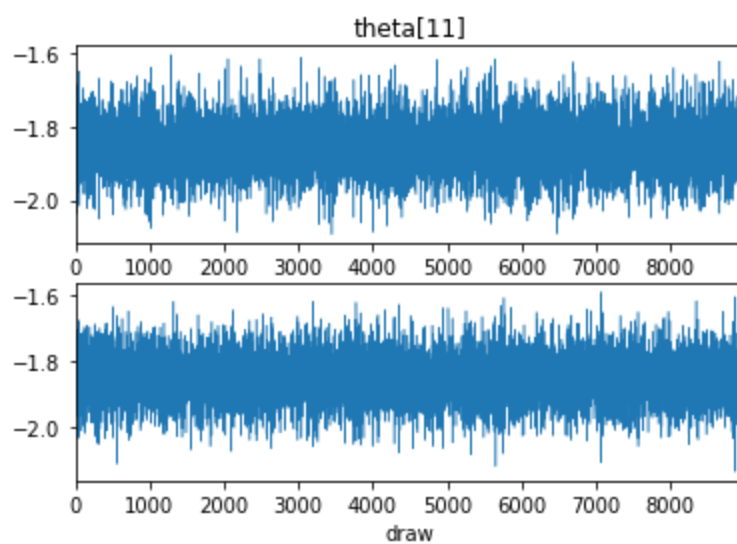
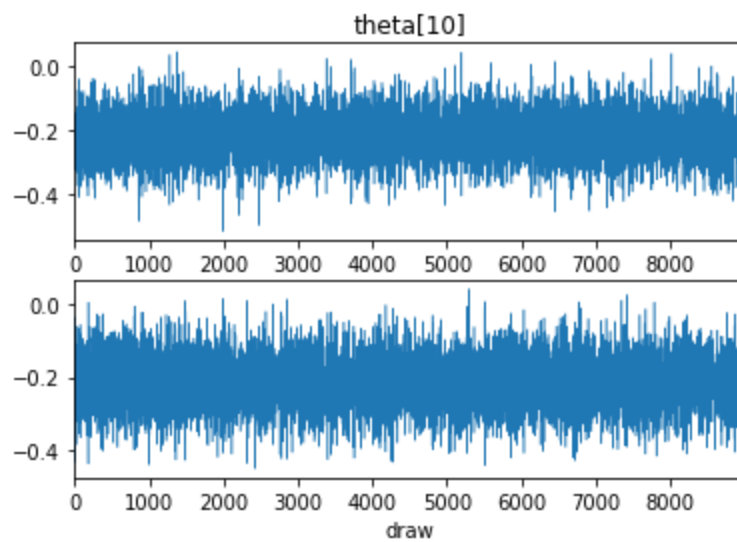


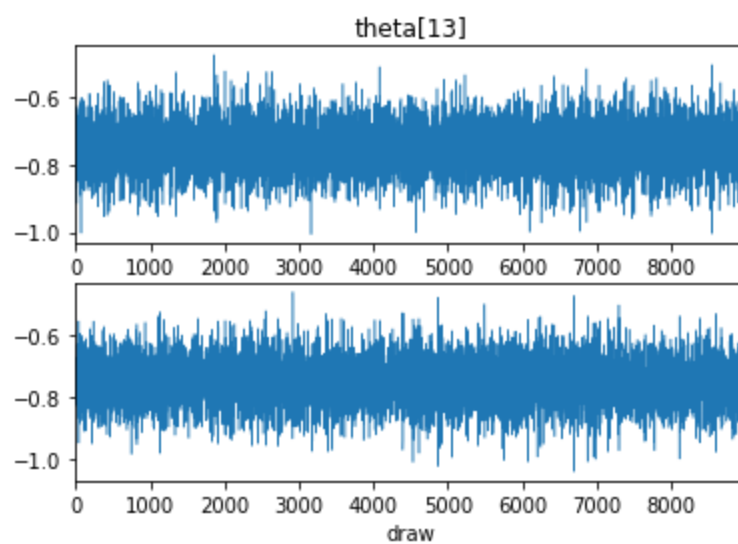
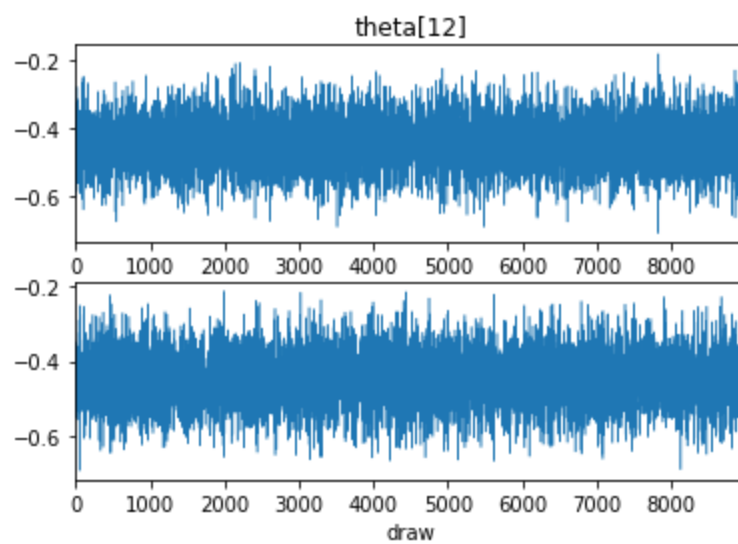


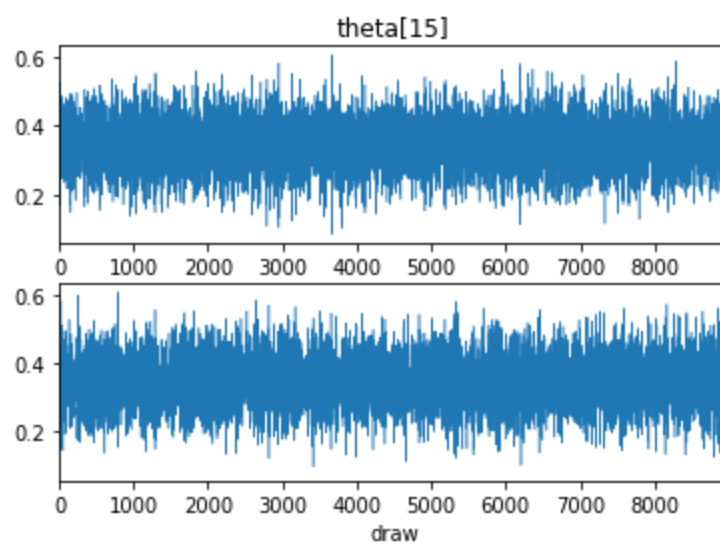
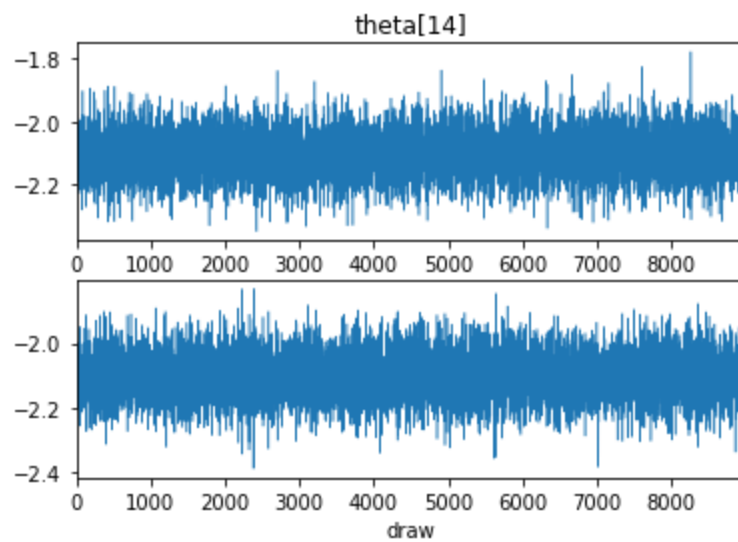


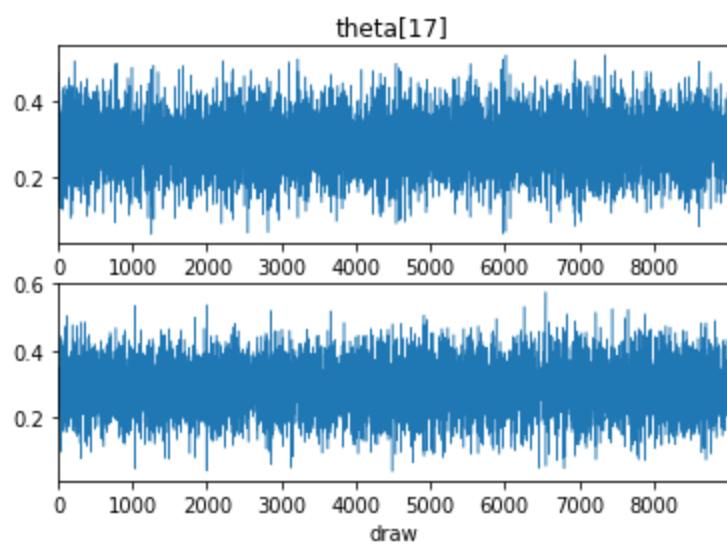
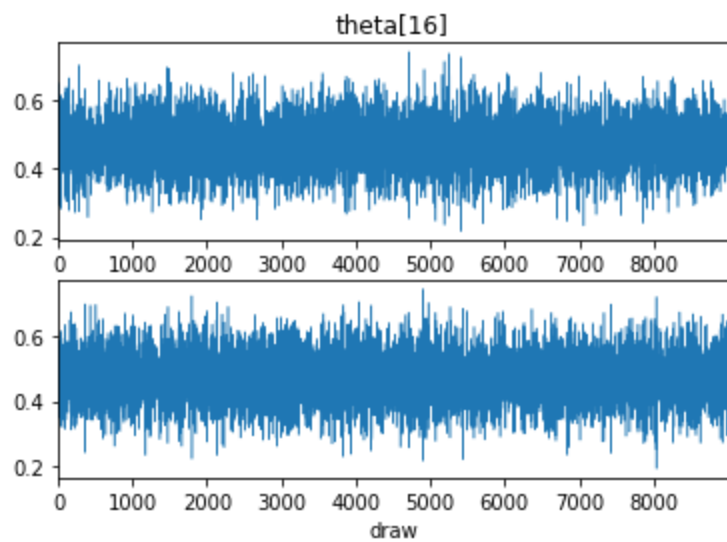


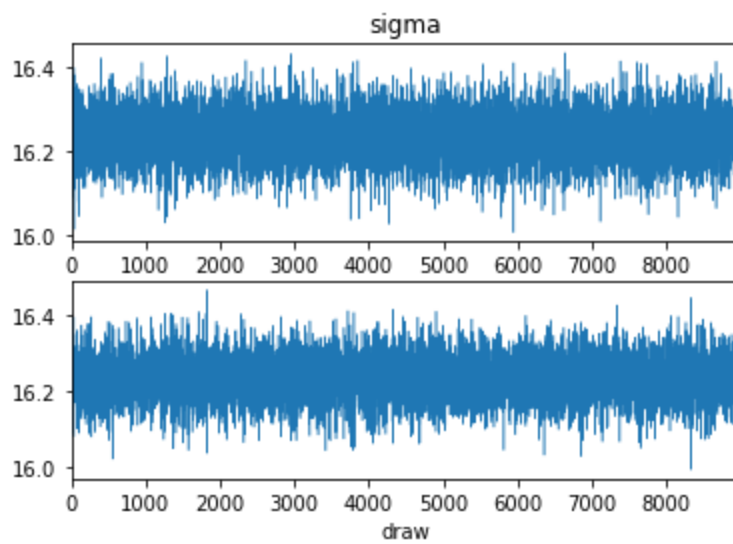
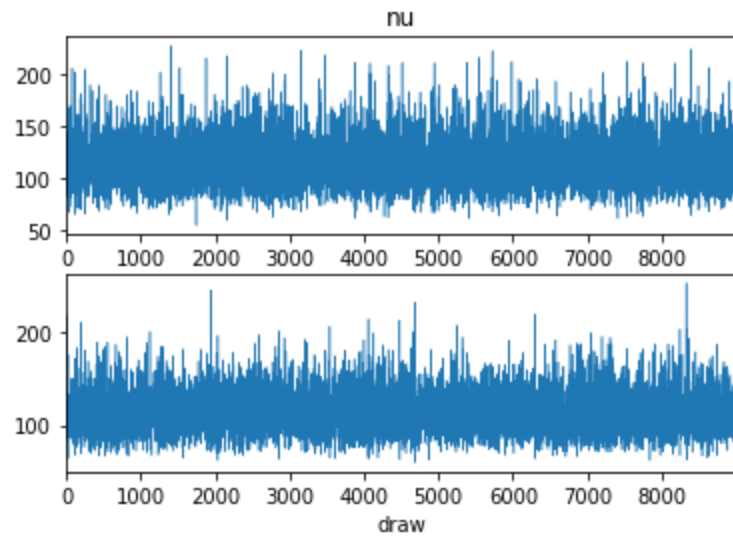












Appendix 3a: Summary statistics for β parameters in linear model

<i>parameter</i>	<i>mean ($\bar{\beta}$) / standardized mean ($\bar{\beta}_{std}$)</i>	<i>95% conf. intvl.* (unstd. / standardized)</i>	<i>effective n</i>	<i>\hat{R}</i>
β_1 availability	0.27 / 9.36	[0.27, 0.27] / [9.22, 9.51]	41538	1.0
β_2 explicit	8.81 / 3.44	[8.39, 9.22] / [3.28, 3.60]	49921	1.0
β_3 track_number	-0.52 / -4.24	[-0.54, -0.50] / [-4.38, -4.09]	42326	1.0
β_4 days_since_release	-6.9x10 ⁻⁴ / -2.01	[-7.4x10 ⁻⁴ , -6.4x10 ⁻⁴] / [-2.16, -1.86]	44418	1.0
β_5 num_artists	-0.28 / -0.25	[-0.43, -0.12] / [-0.40, -0.11]	42791	1.0
β_6 danceability	6.37 / 1.11	[0.56, 7.45] / [0.93, 1.30]	41832	1.0
β_7 energy	-7.41 / -1.71	[-8.7, -6.12] / [-2.01, -1.41]	42986	1.0
β_8 loudness	1.37 / 5.65	[1.31, 1.43] / [5.39, 5.90]	44825	1.0
β_9 mode	-1.36 / -0.65	[-1.65, -1.06] / [-0.79, -0.51]	40778	1.0
β_{10} speechiness	-0.44 / -0.06	[-1.56, 0.68] / [-0.08, 0.10]	40372	1.0
β_{11} acousticness	-5.52 / -1.79	[-6.19, -4.84] / [-2.01, -1.57]	40572	1.0
β_{12} instrumentalness	-4.44 / -0.74	[-5.33, -3.55] / [-0.89, -0.59]	41789	1.0
β_{13} liveness	-4.06 / -0.87	[-4.78, -3.34] / [-1.02, -0.71]	47673	1.0
β_{14} valence	-10.59 / -2.56	[-11.34, -9.85] / [-2.74, -2.39]	47037	1.0
β_{15}	0.01 / 0.36	[0.01, 0.02] /	43465	1.0

tempo		[0.22, 0.51]		
β_{16} duration_s	6.1×10^{-3} / 0.48	$[4.2 \times 10^{-3}, 8.1 \times 10^{-3}]$ / [0.43, 0.63]	45123	1.0
β_{17} time_signature_4	1.02 / 0.32	[0.52, 1.52] / [0.16, 0.47]	39418	1.0

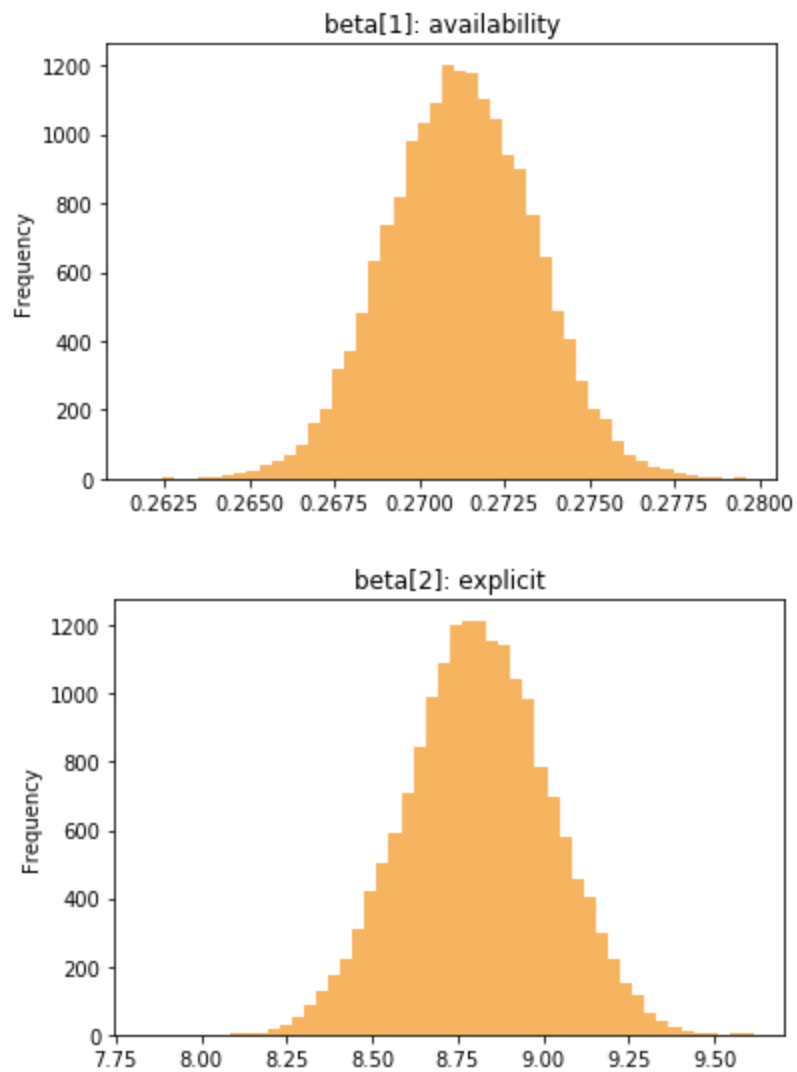
*The distributions for the β variables are approximately symmetric, so the confidence intervals above can also be reasonably treated as highest posterior density intervals.

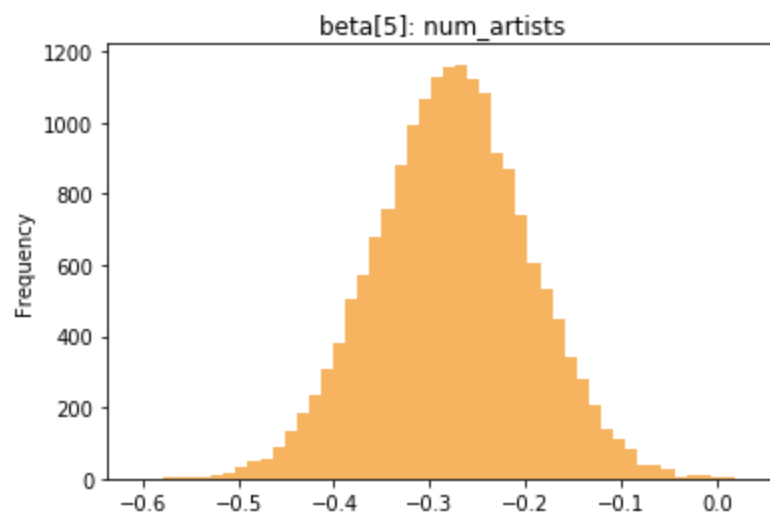
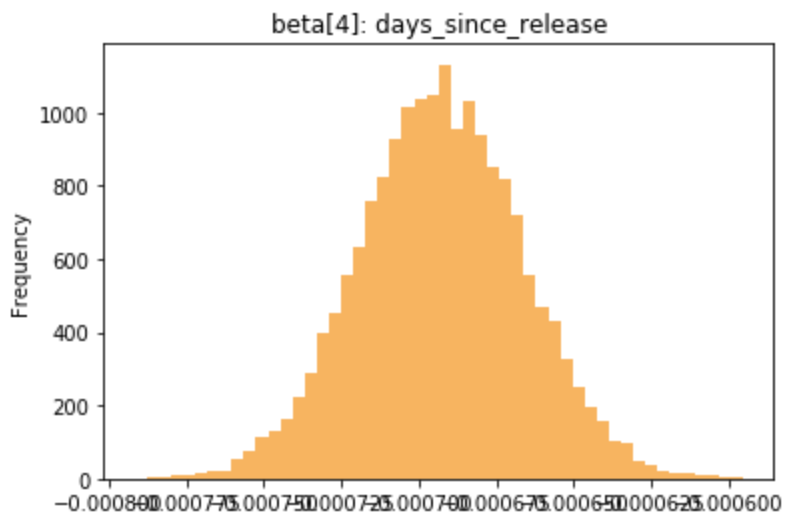
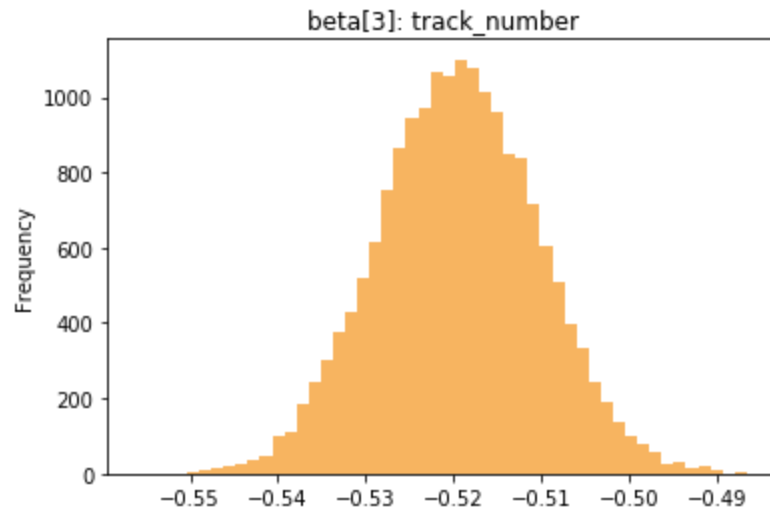
Appendix 3b: Summary statistics for other parameters in linear model

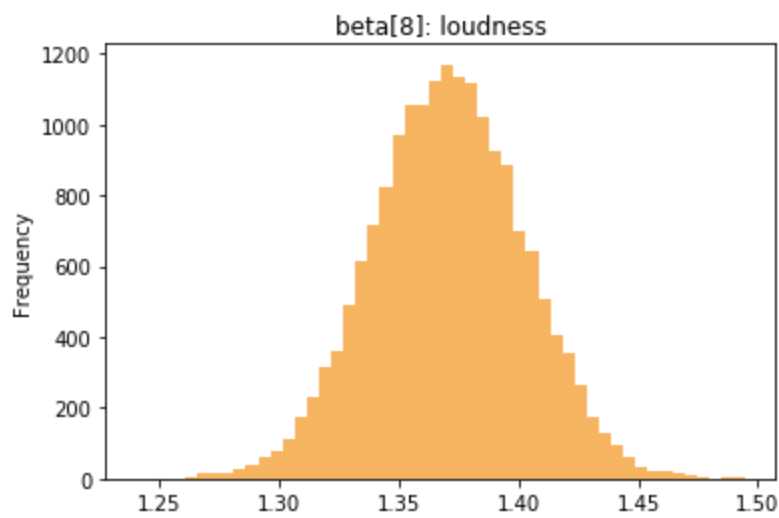
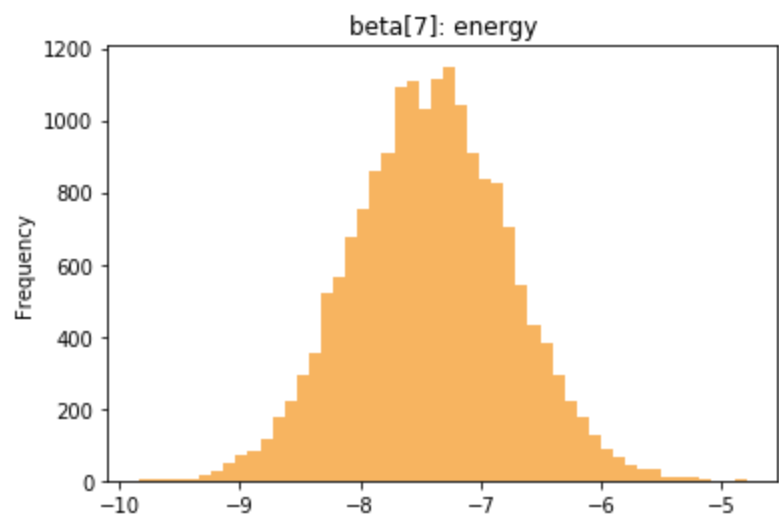
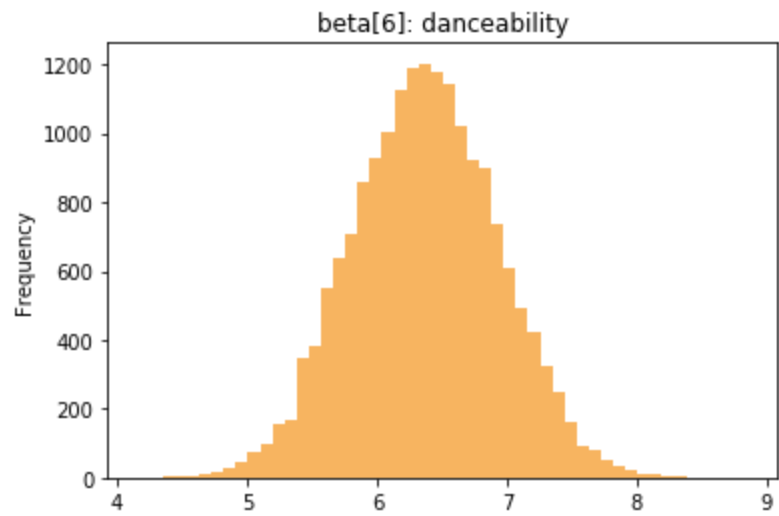
<i>parameter</i>	<i>mean</i>	<i>95% h.p.d. interval</i>	<i>effective n</i>	\hat{R}
α	26.04	[25.89, 26.18]	43297	1.0
ν	114.13	[73.32, 160.75]	35570	1.0
σ	16.24	[16.12, 16.35]	29426	1.0

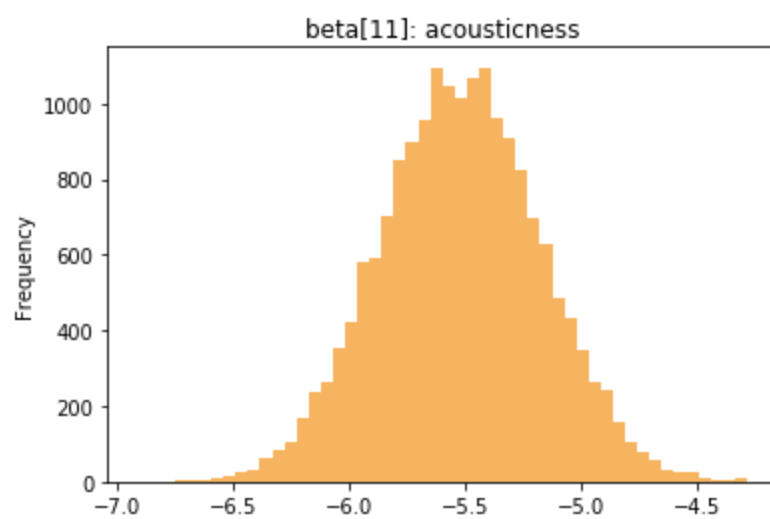
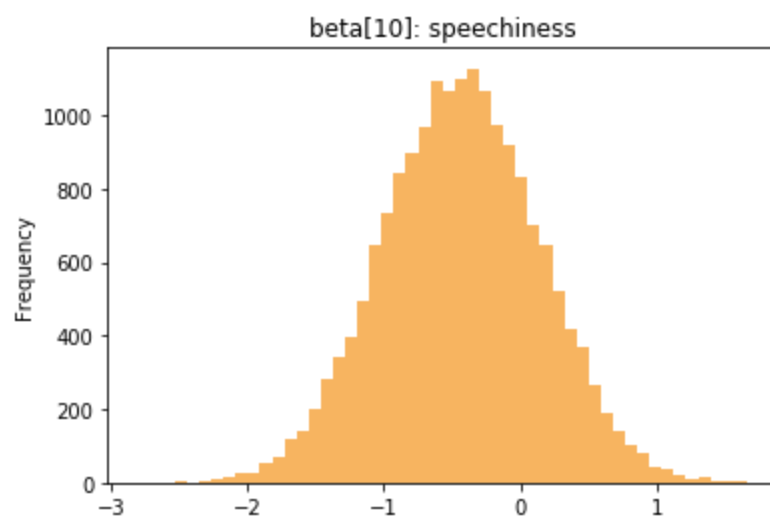
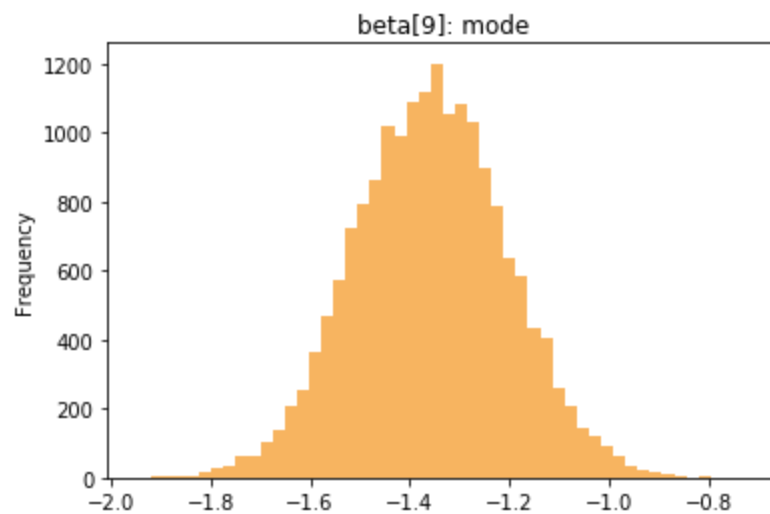
Appendix 4: Histograms for linear model (both chains are combined)

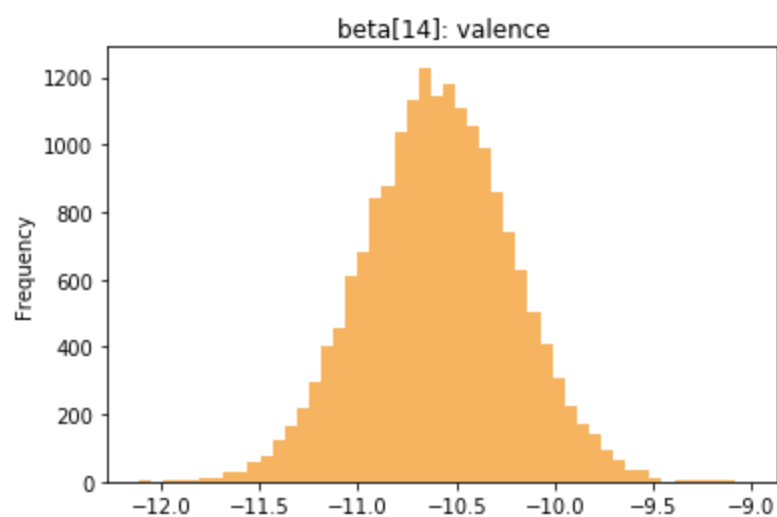
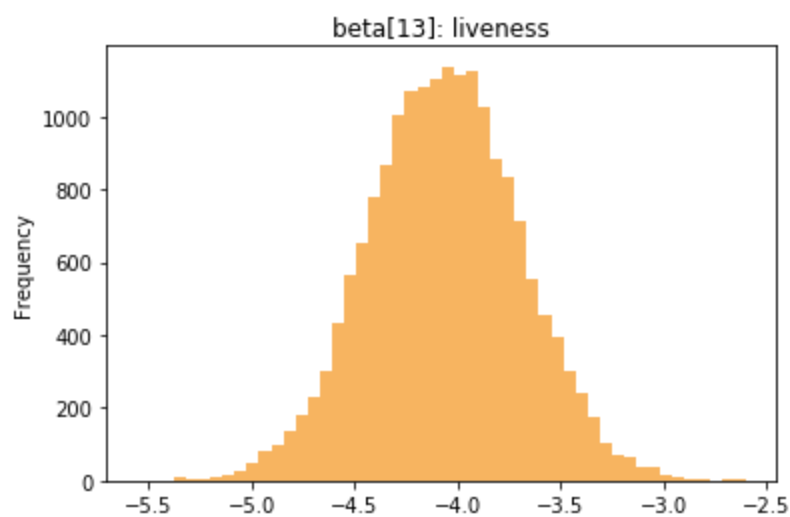
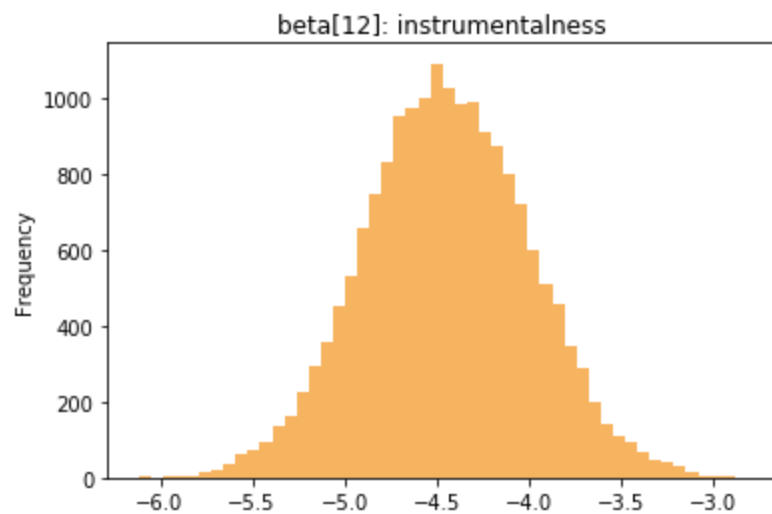
4a: Unstandardized β (histograms for β_{std} are the same but with a different scale)

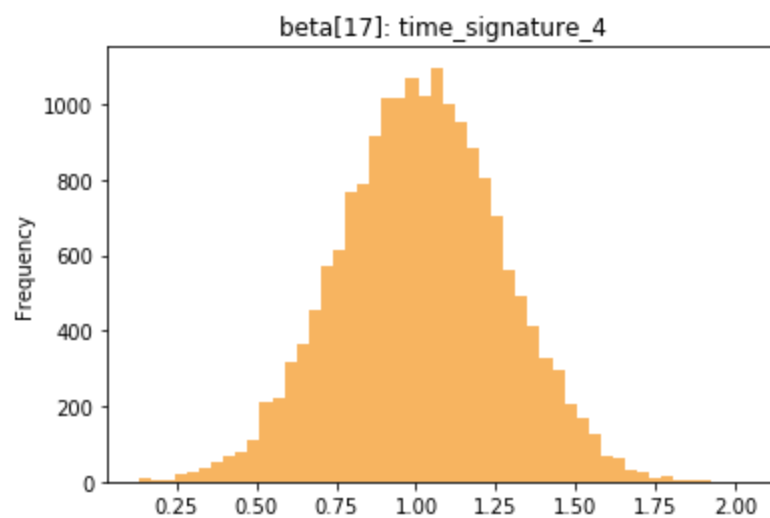
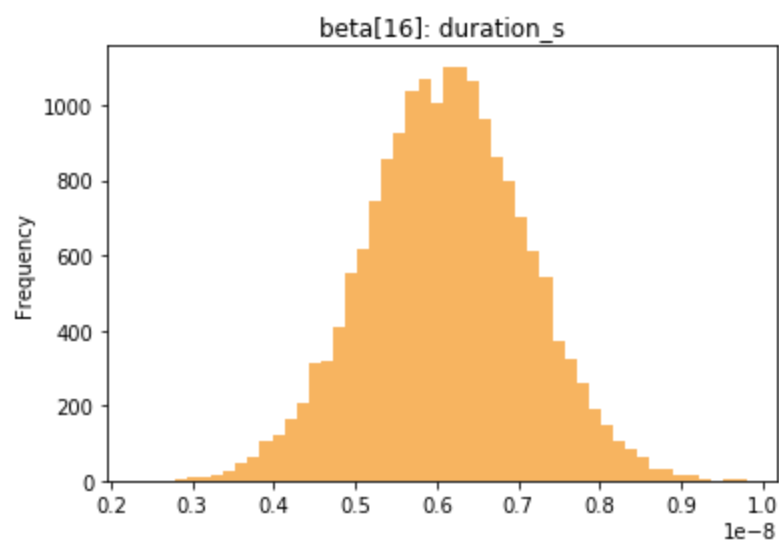
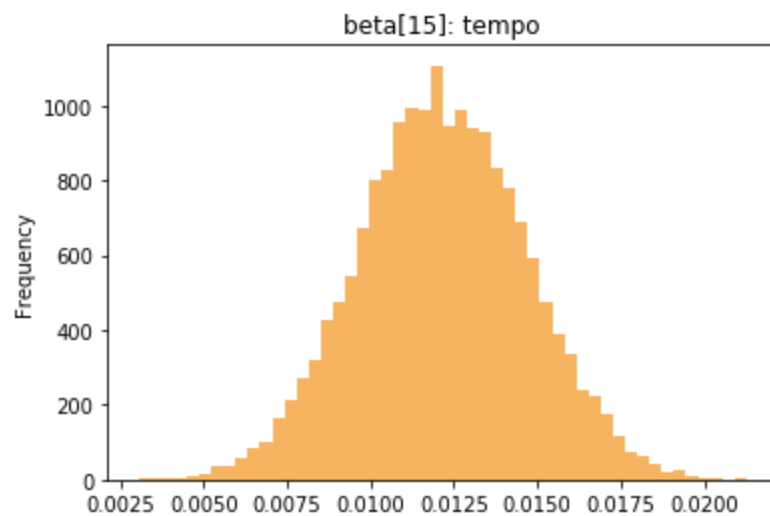




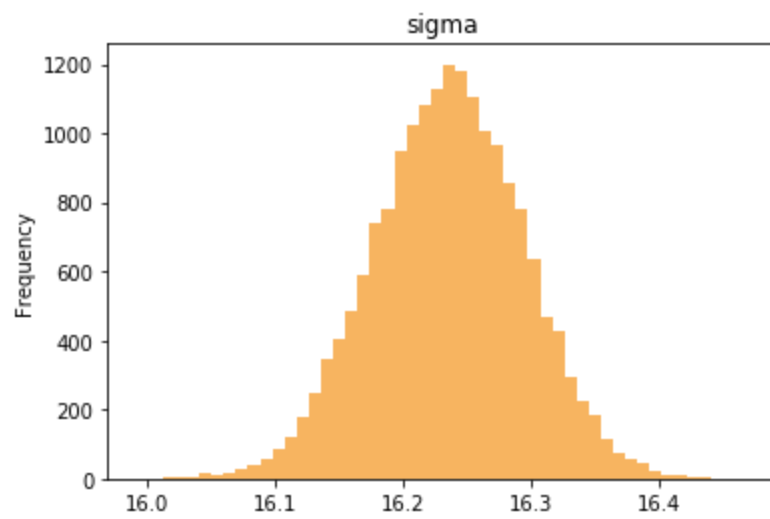
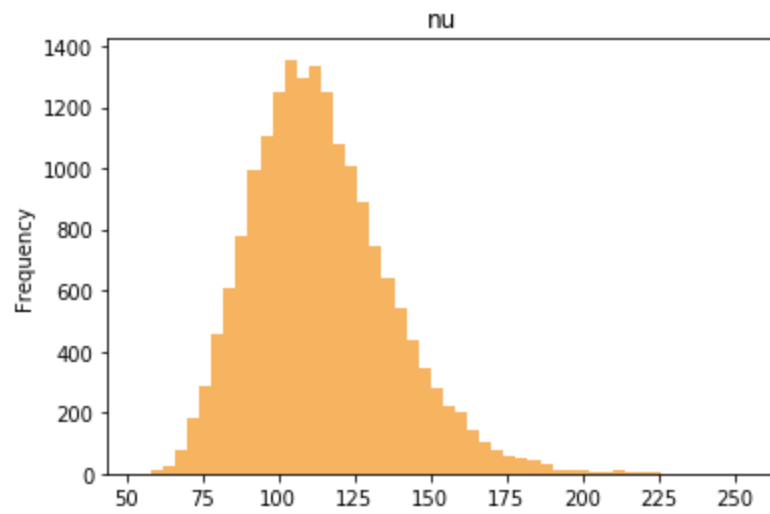
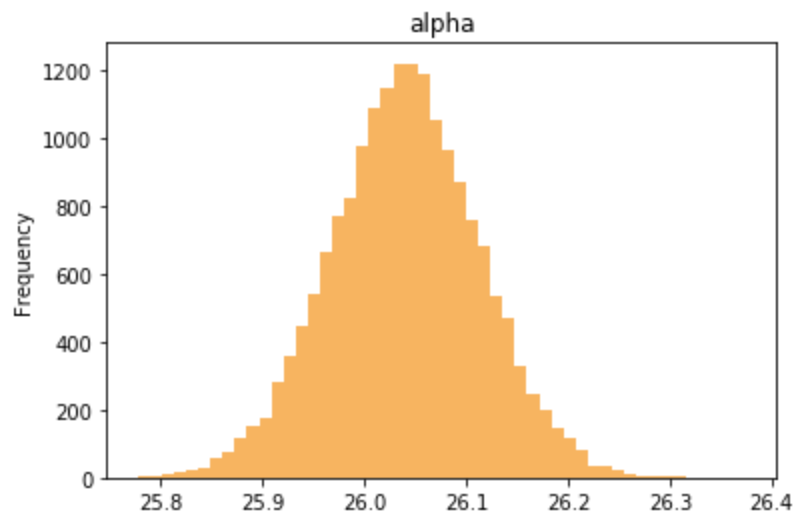




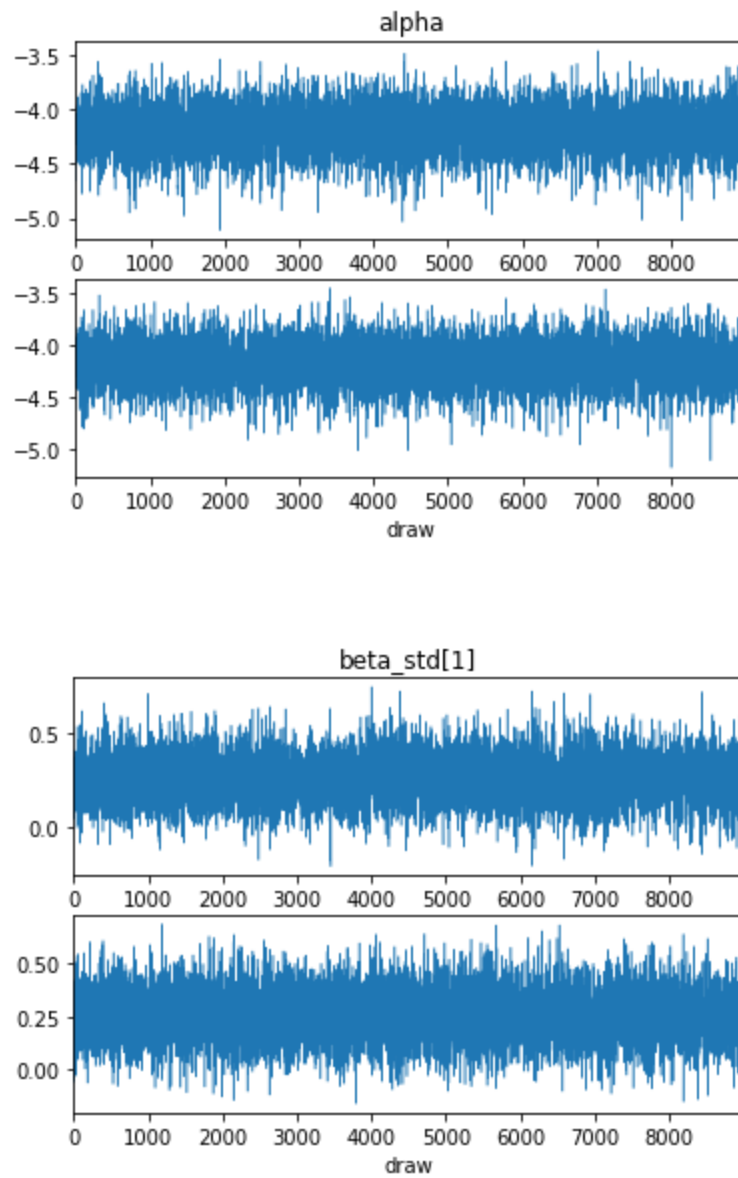


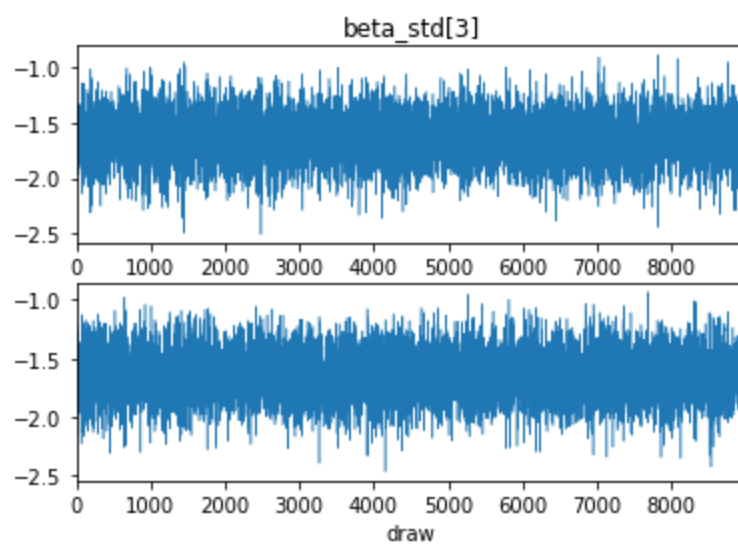
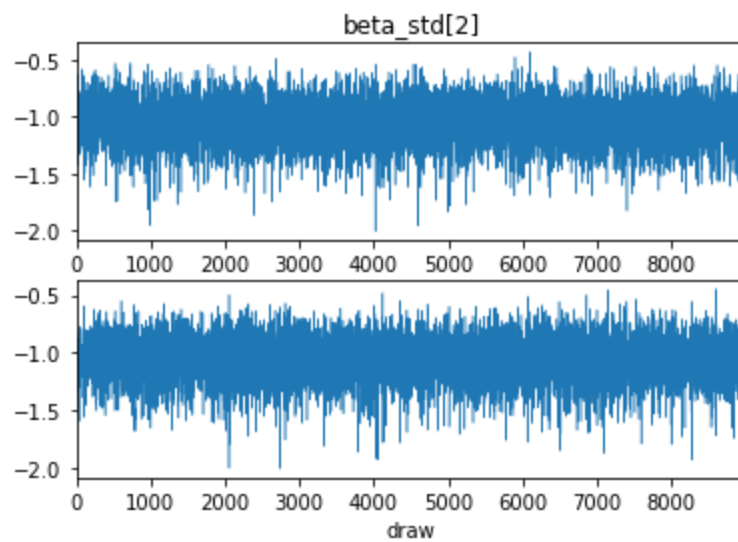


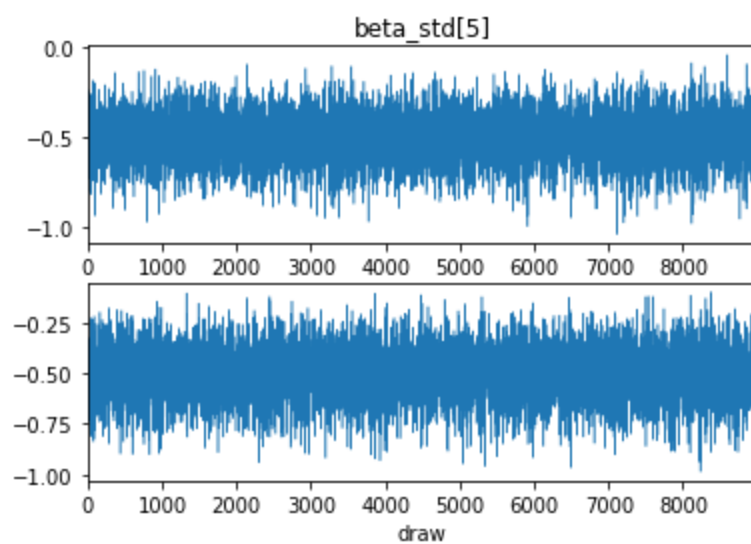
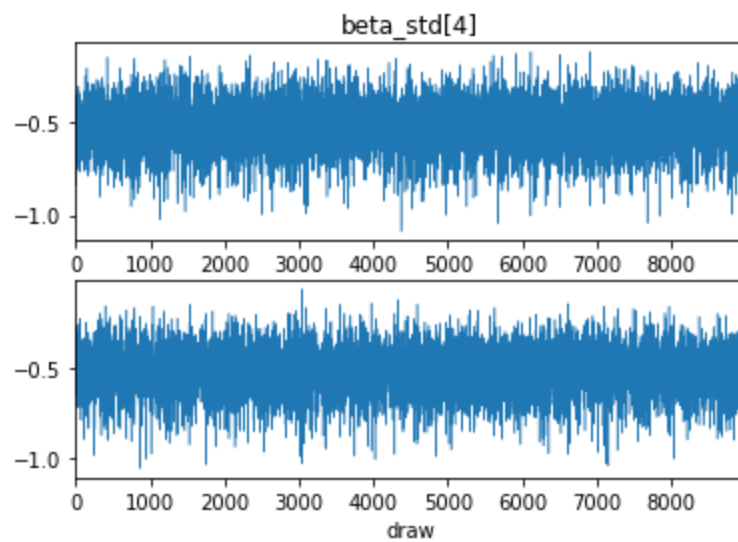
4b: Other parameters

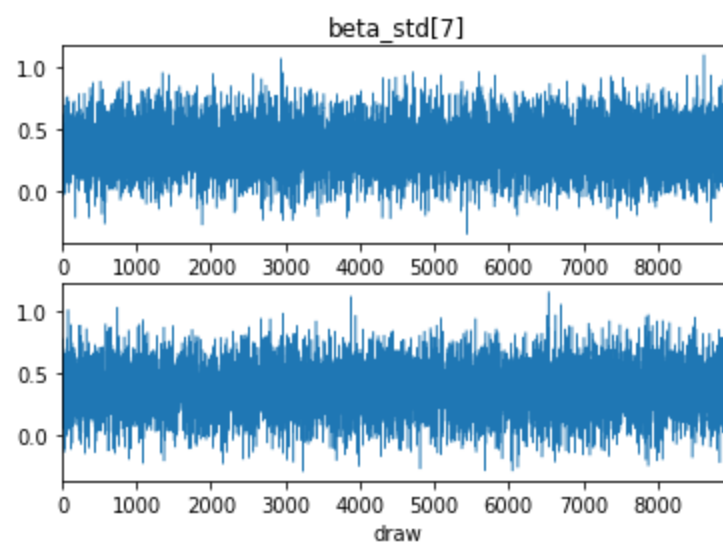
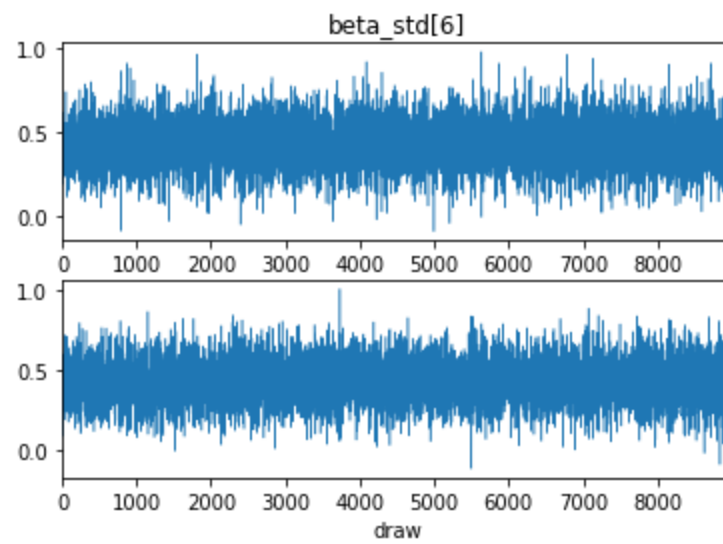


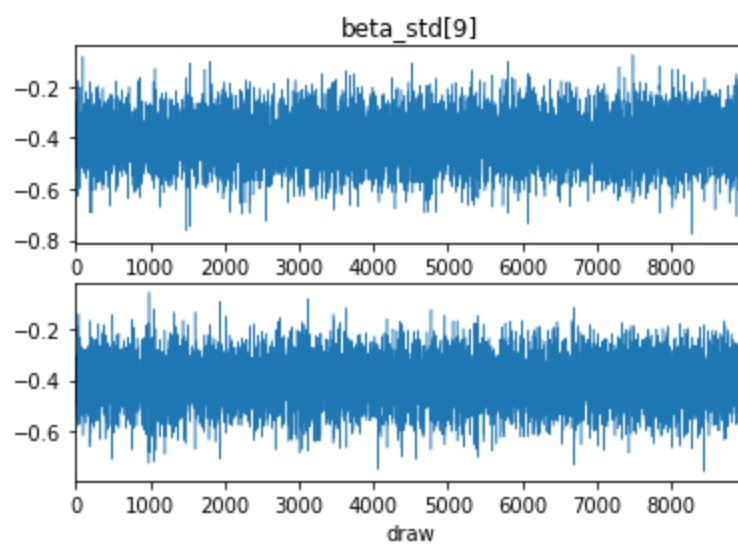
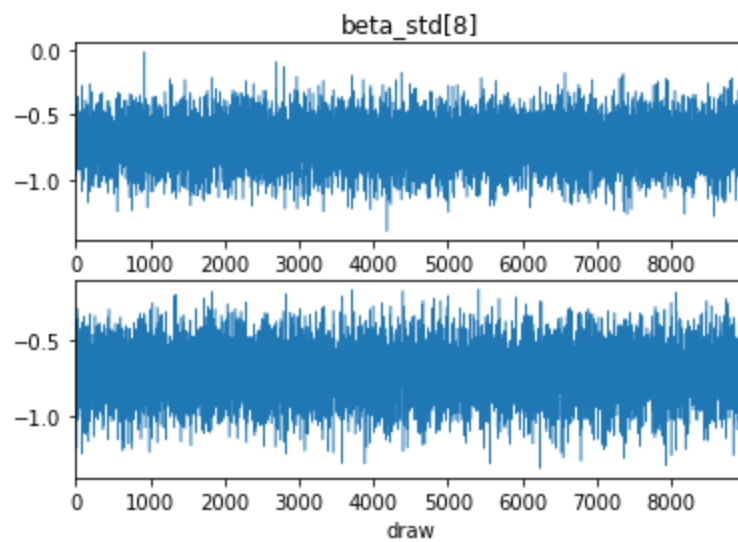
Appendix 5: Trace plots for logistic model MCMC

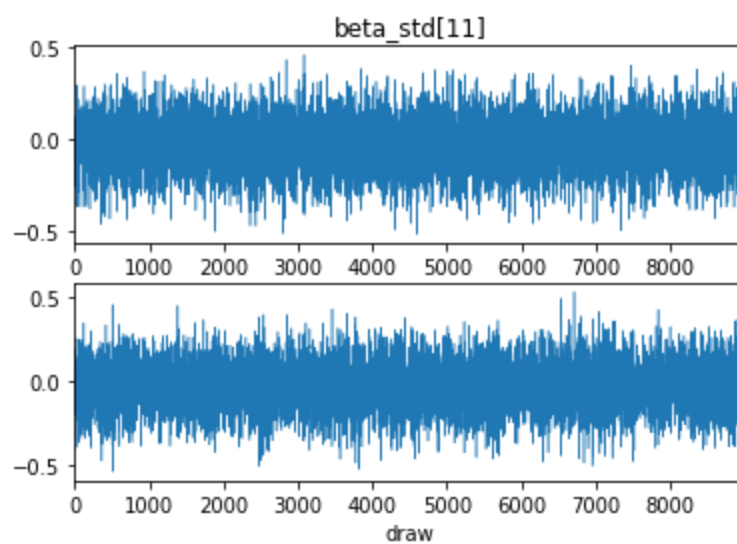
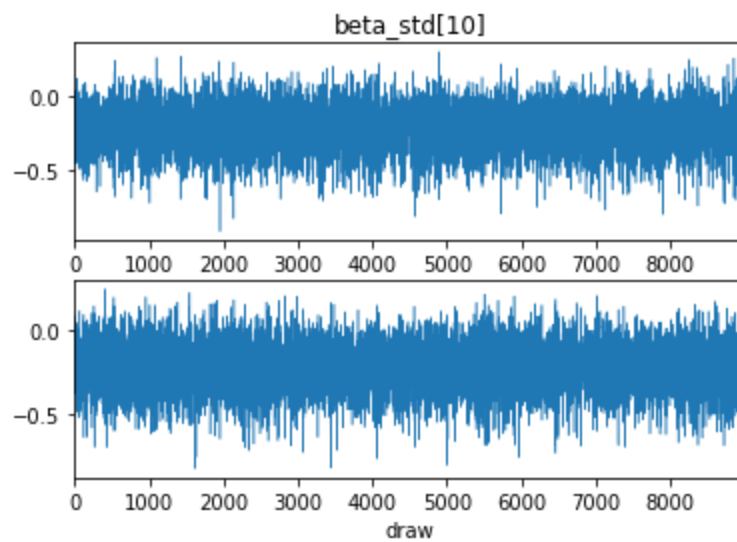


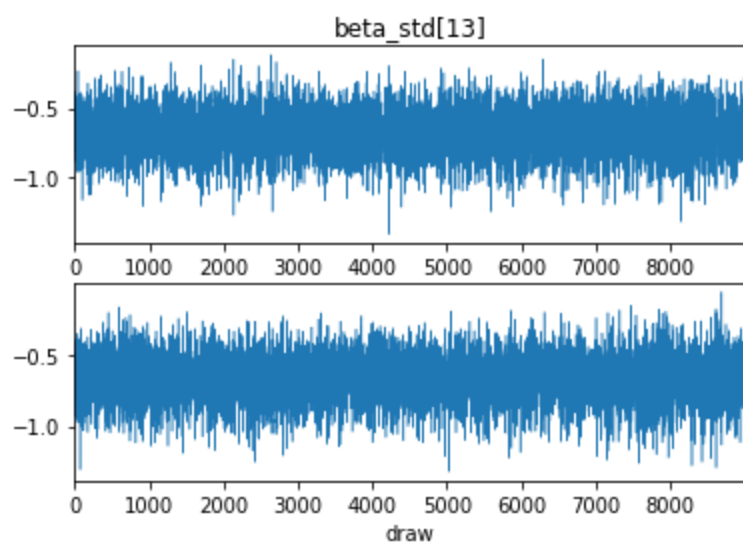
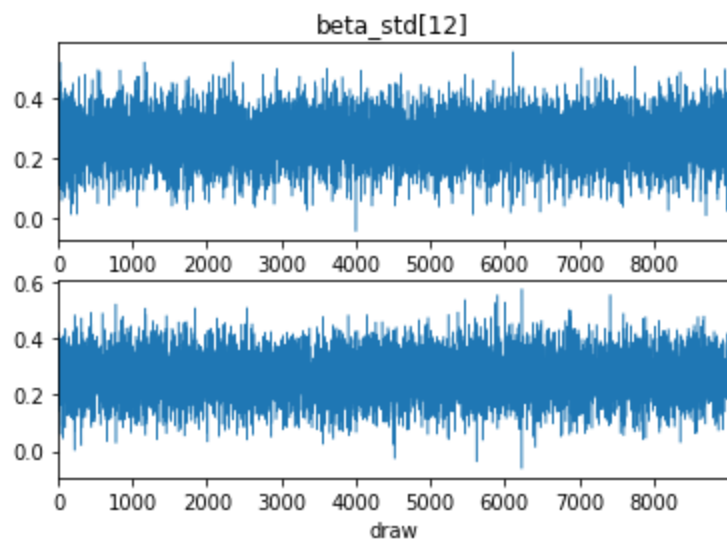


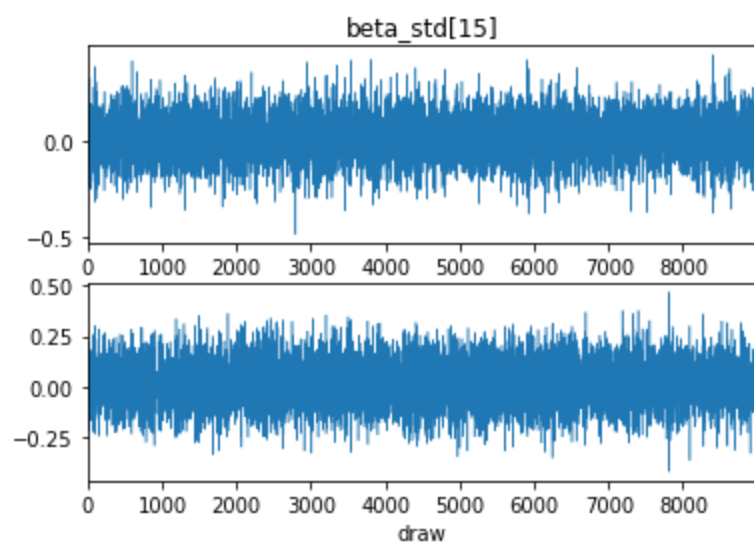
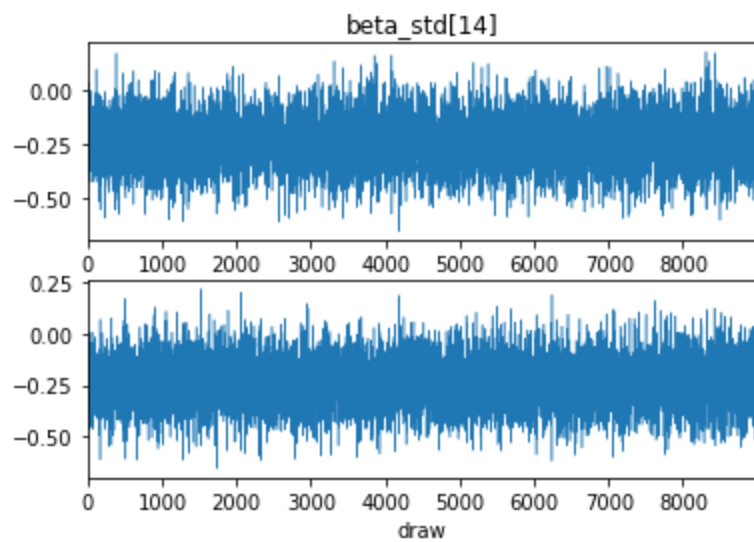


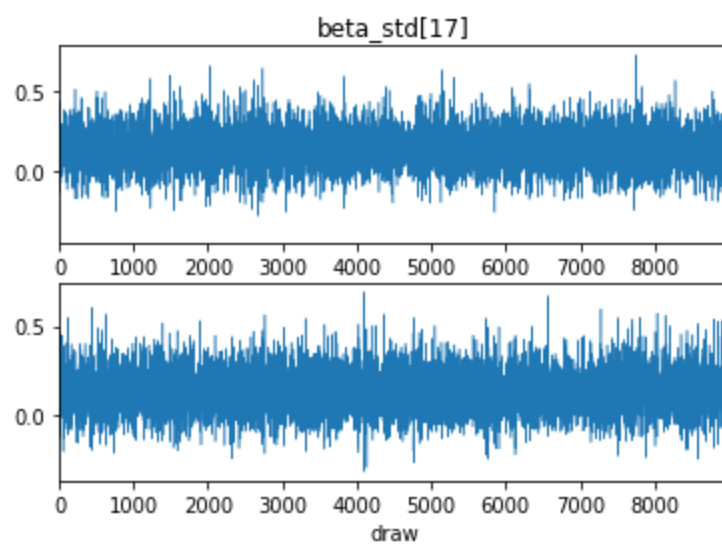
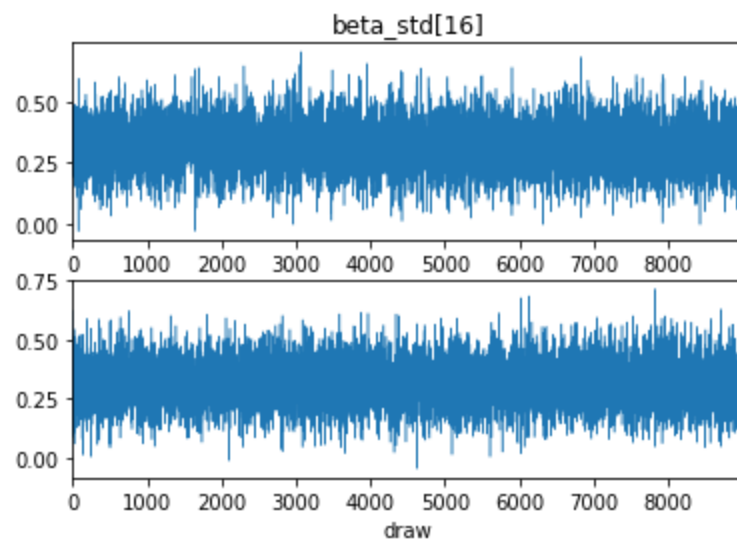


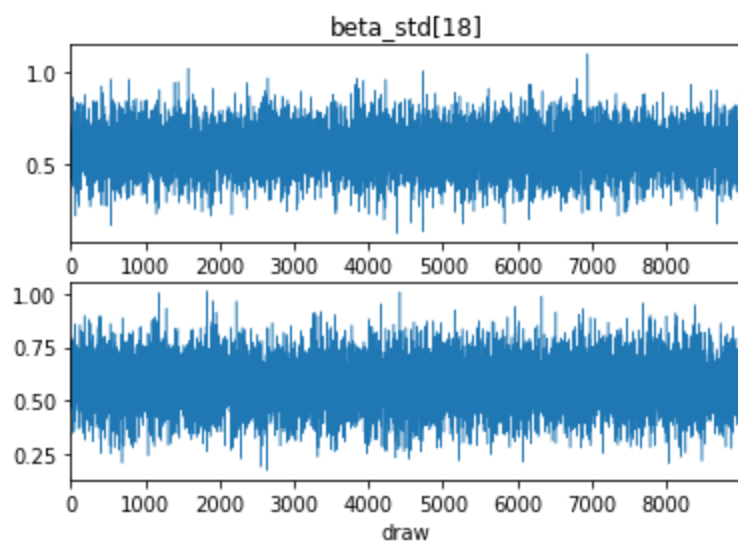












Appendix 6: Summary statistics for β parameters in logistic model

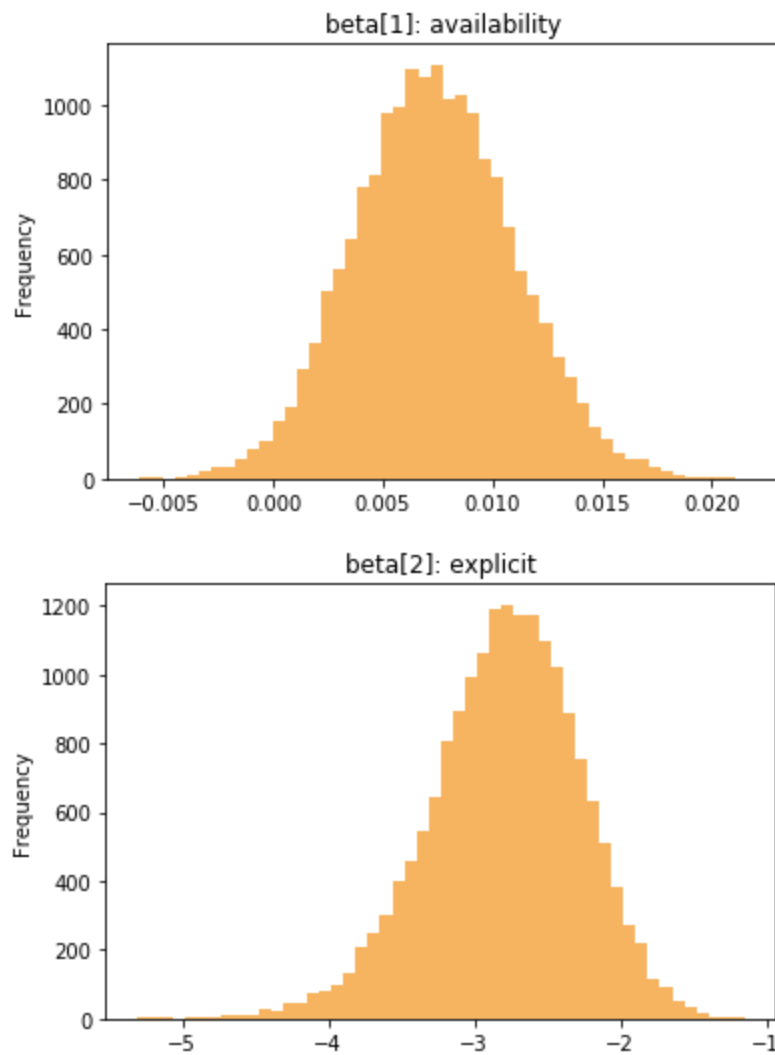
<i>parameter</i>	<i>mean ($\bar{\beta}$) / standardized mean ($\bar{\beta}_{std}$)</i>	<i>95% conf. intvl.* (unstd. / standardized)</i>	<i>effective n</i>	<i>\hat{R}</i>
β_1 availability	7.3×10^{-3} / 0.25	$[3.8 \times 10^{-4}, 0.01]$ / $[0.01, 0.50]$	29863	1.0
β_2 explicit	-2.8 / -1.06	$[-3.89, -1.86]$ / $[-1.47, -0.70]$	27318	1.0
β_3 track_number	-0.21 / -1.64	$[-0.26, -0.16]$ / $[-2.06, -1.24]$	25660	1.0
β_4 days_since_release	-1.9×10^{-4} / -0.52	$[-2.9 \times 10^{-4}, -1.0 \times 10^{-4}]$ / $[-0.8, -0.28]$	28408	1.0
β_5 num_artists	-0.59 / -0.50	$[-0.90, -0.29]$ / $[-0.77, -0.25]$	27972	1.0
β_6 danceability	2.5 / 0.43	$[0.95, 4.04]$ / $[0.16, 0.70]$	23662	1.0
β_7 energy	1.62 / 0.36	$[-0.09, 3.34]$ / $[-0.02, -0.75]$	18656	1.0
β_8 loudness	-0.18 / -0.72	$[-0.27, -0.09]$ / $[-1.07, -0.38]$	21981	1.0
β_9 mode	-0.85 / -0.41	$[-1.22, -0.47]$ / $[-0.59, -0.23]$	36208	1.0
β_{10} speechiness	-1.65 / -0.22	$[-4.02, 0.50]$ / $[-0.53, 0.07]$	28656	1.0
β_{11} acousticness	-0.10 / -0.03	$[-0.99, 0.78]$ / $[-0.31, 0.24]$	26220	1.0
β_{12} instrumentalness	1.33 / 0.26	$[0.55, 2.08]$ / $[0.11, 0.41]$	26634	1.0
β_{13} liveness	-3.23 / -0.67	$[-4.87, -1.75]$ / $[-1.01, -0.36]$	29753	1.0
β_{14} valence	-0.97 / -0.24	$[-1.92, -4.1 \times 10^{-3}]$ / $[-0.47, -0.00]$	24475	1.0
β_{15}	6.1×10^{-4} / 0.02	$[-6.9 \times 10^{-3}, 8.0 \times 10^{-3}]$	32406	1.0

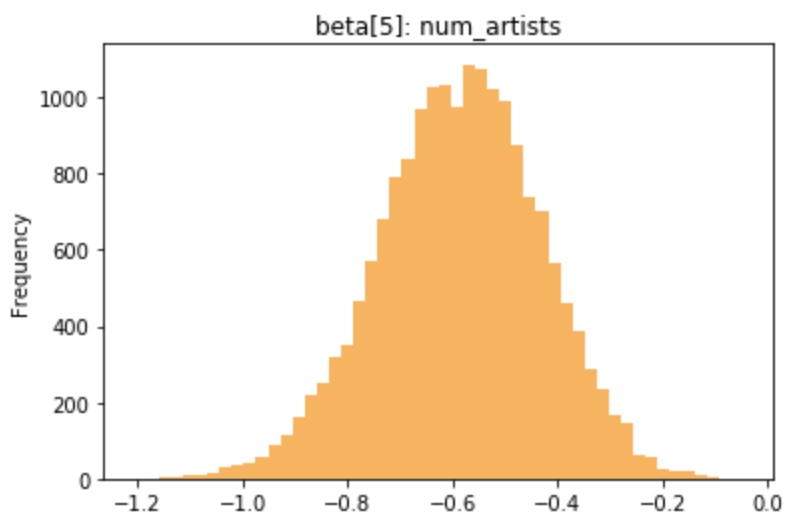
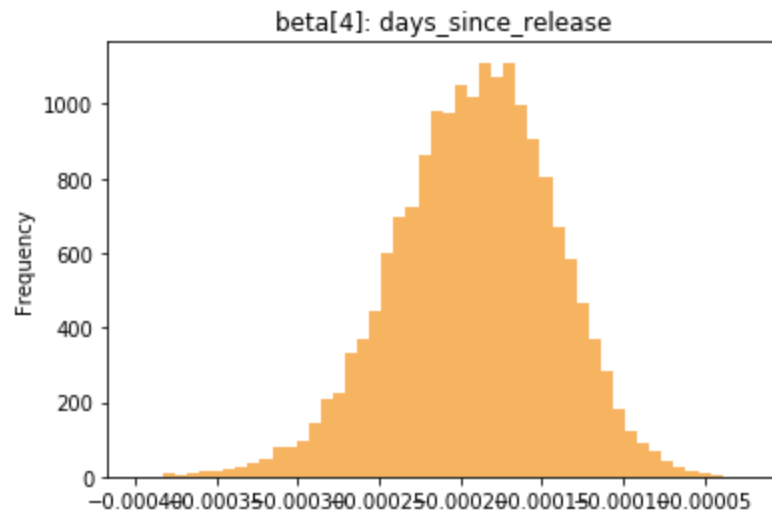
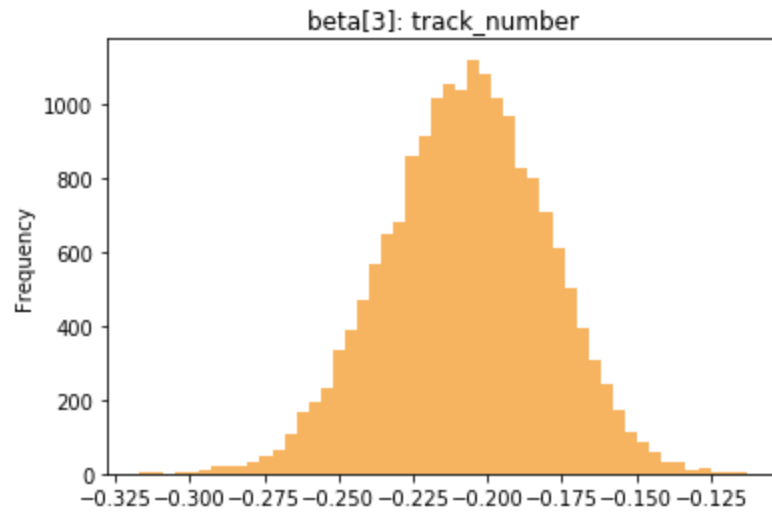
tempo		/ [-0.21, 0.24]		
β_{16} duration_s	4.4×10^{-3} / 0.32	$[1.7 \times 10^{-3}, 7.1 \times 10^{-3}]$ / [0.13, 0.51]	30265	1.0
β_{17} time_signature_4	0.44 / 0.14	[-0.29, 0.17] / [-0.09, 0.38]	32623	1.0
β_{18} popularity	0.03 / 0.57	[0.02, 0.04] / [0.34, 0.81]	28110	1.0

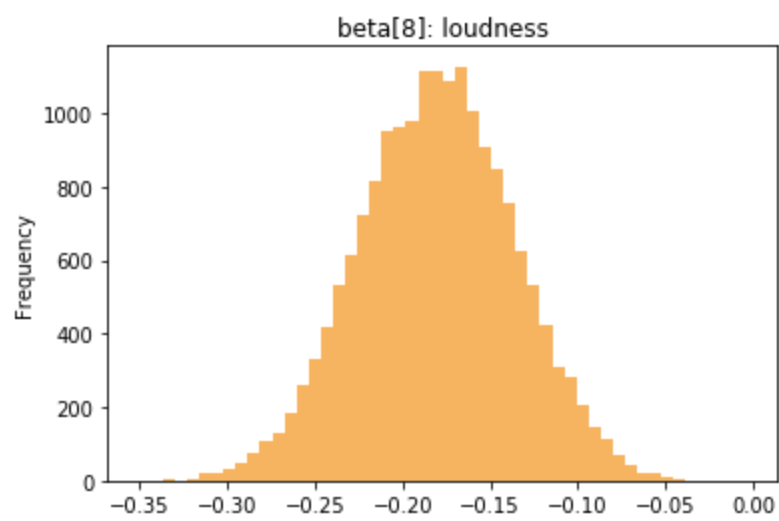
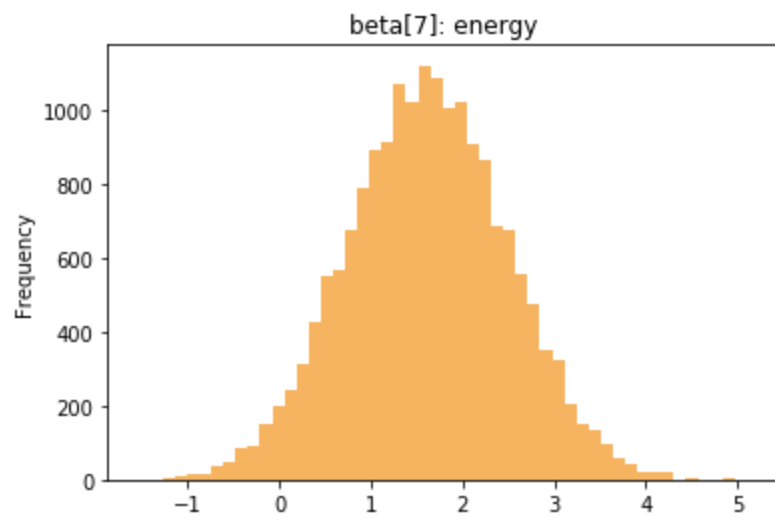
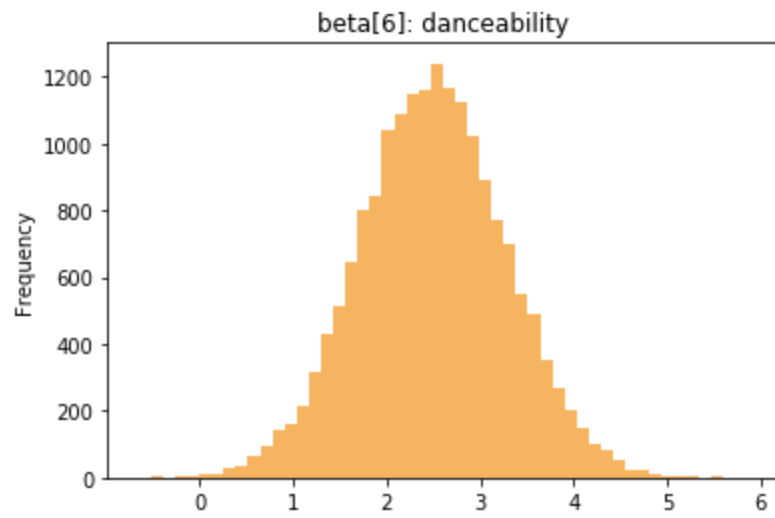
*The distributions for the β variables are approximately symmetric, so the confidence intervals above can also be reasonably treated as highest posterior density intervals.

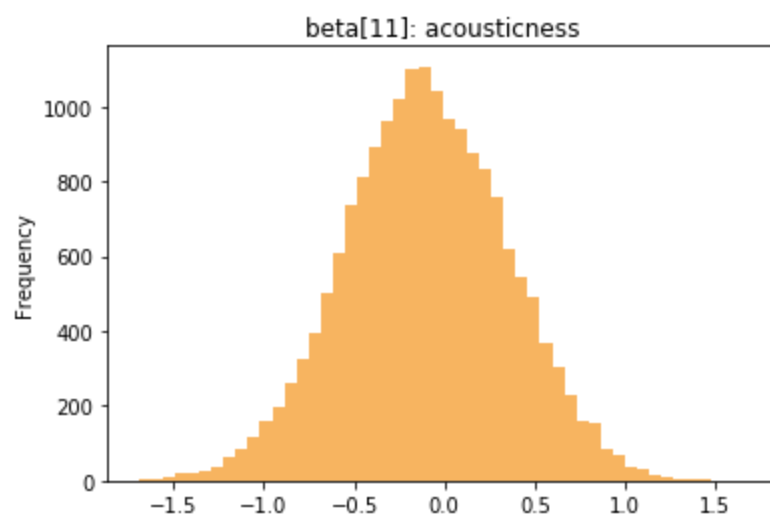
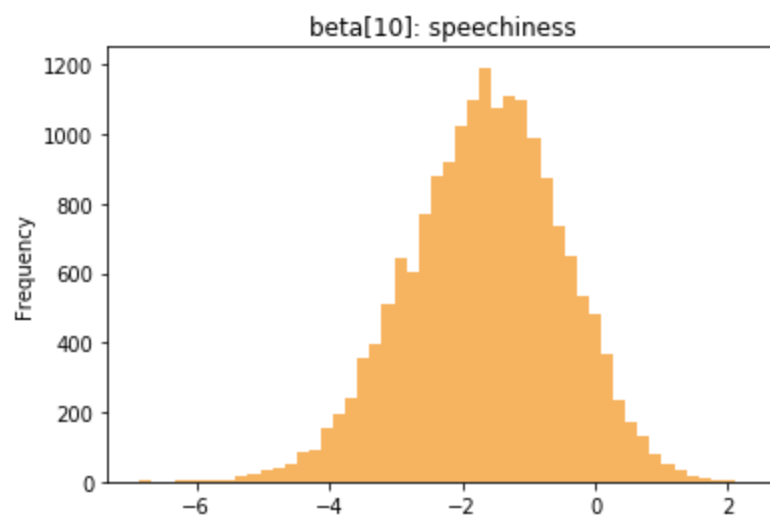
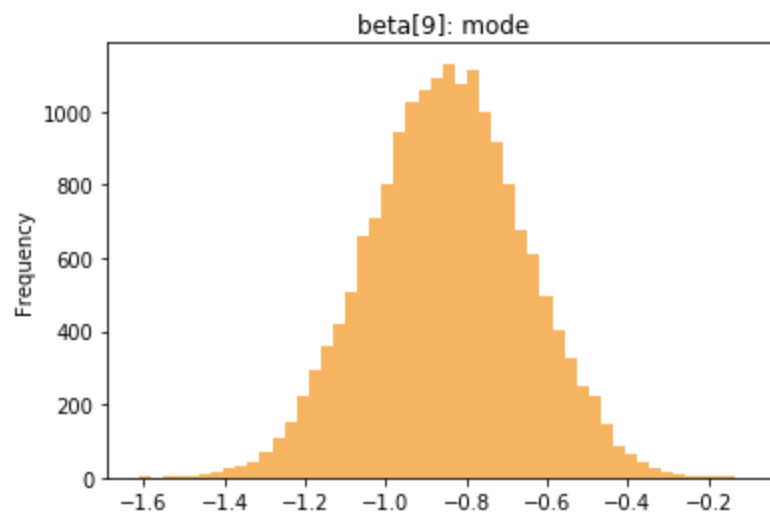
Appendix 7: Histograms for parameters in logistic model

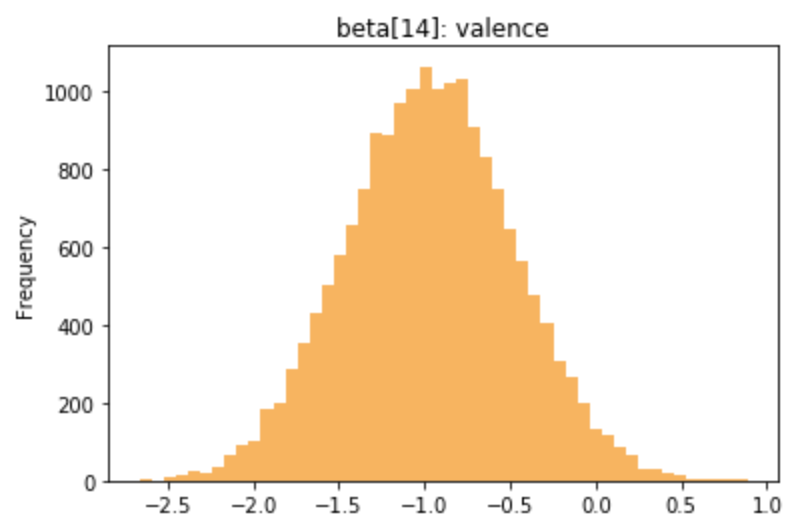
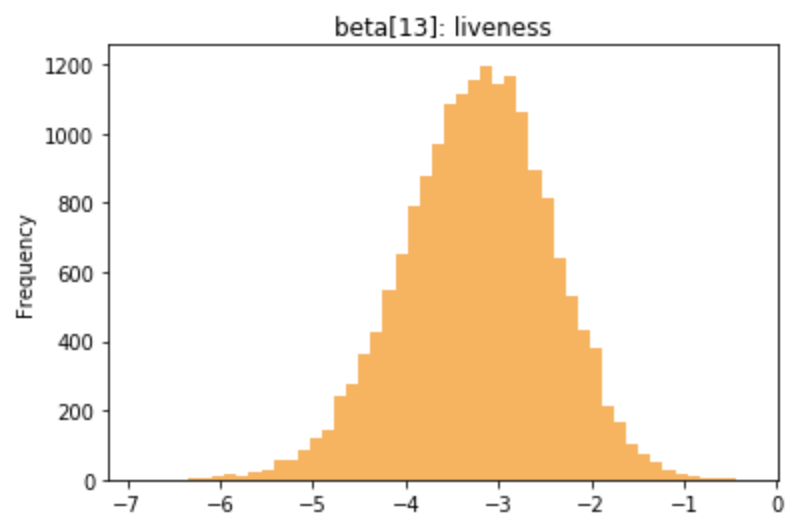
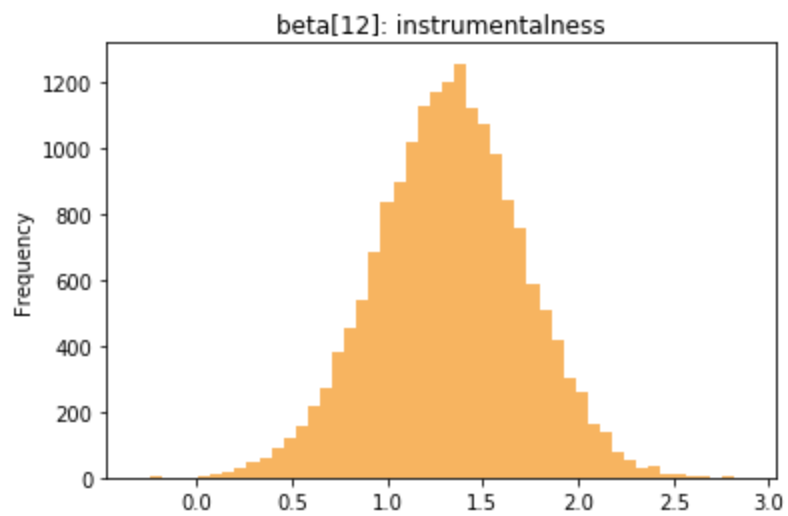
Histograms are for unstandardized β (histograms for β_{std} are the same but with a different scale).

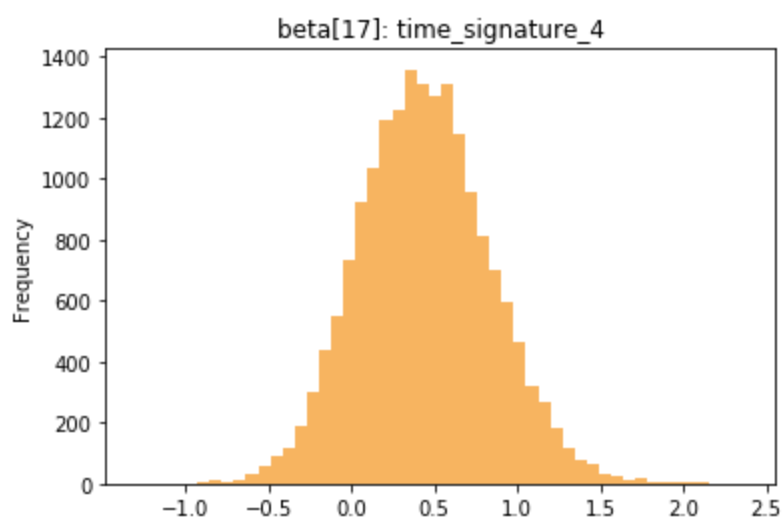
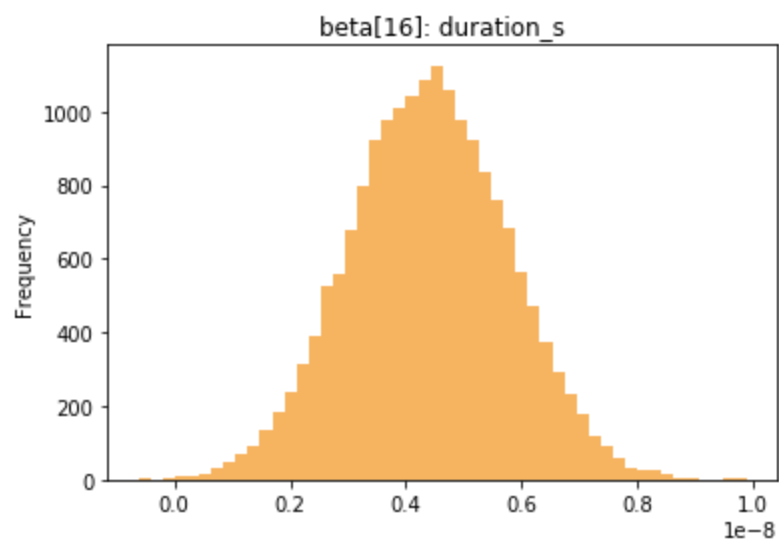
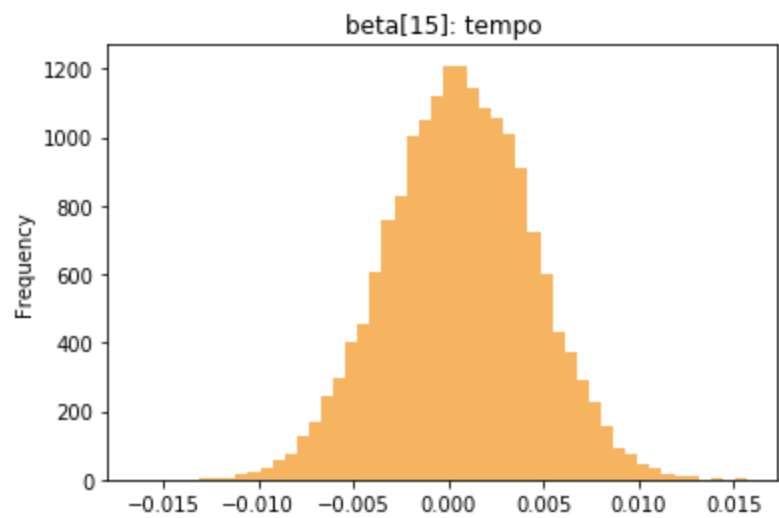


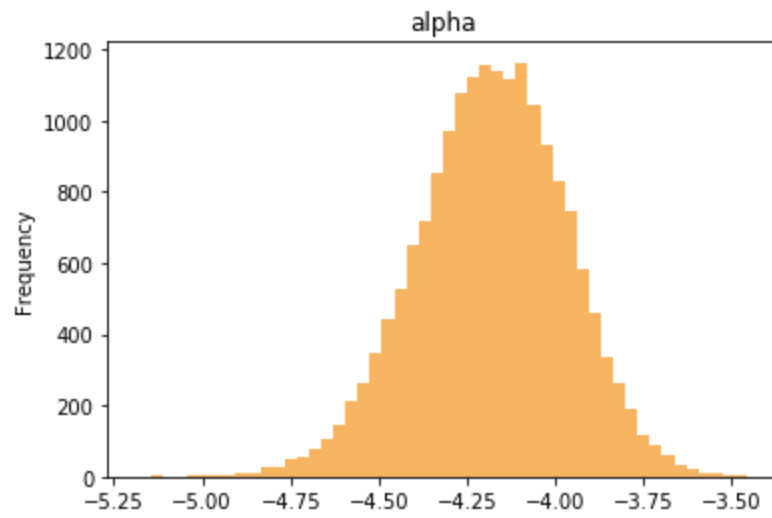
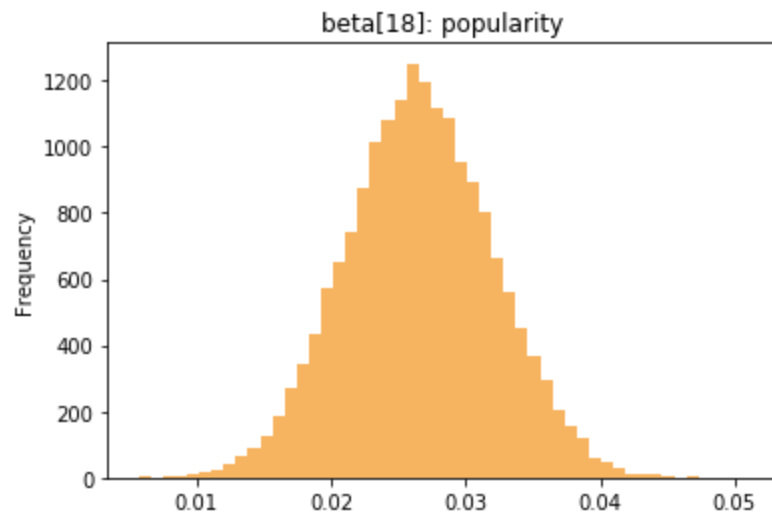












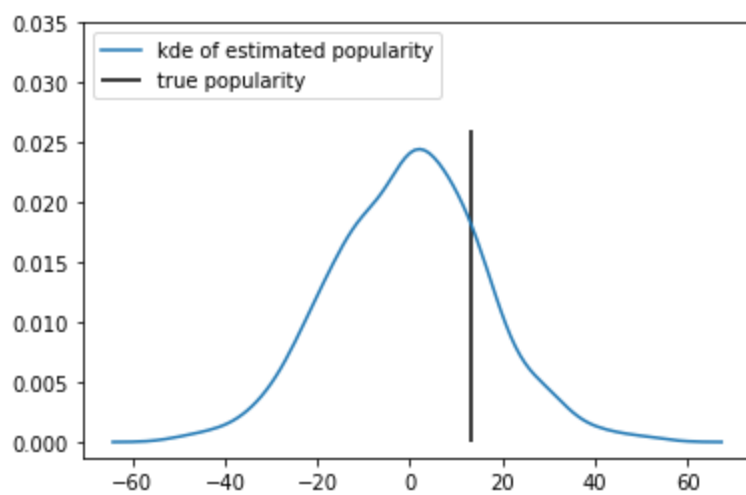
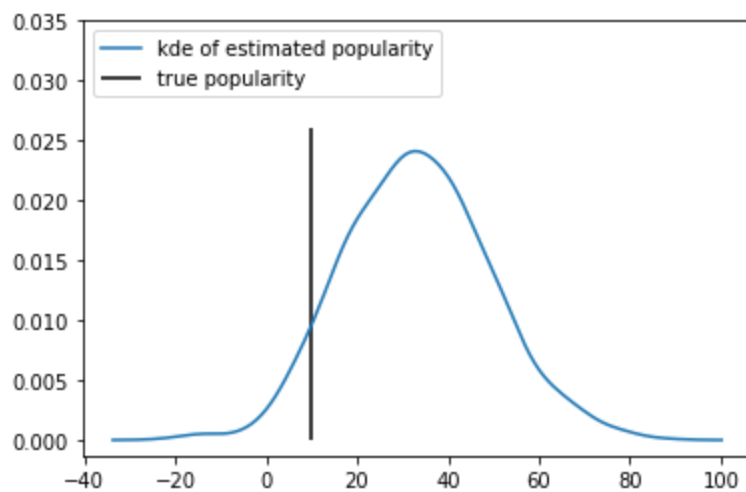
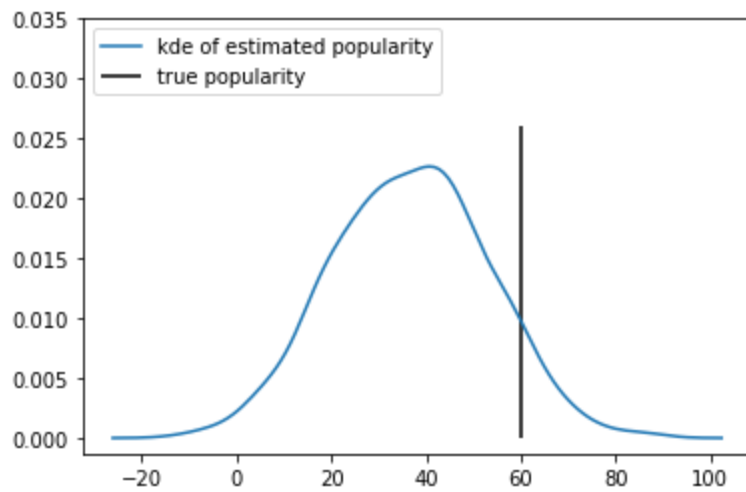
Appendix 8: Goodness-of-fit/posterior predictive tests

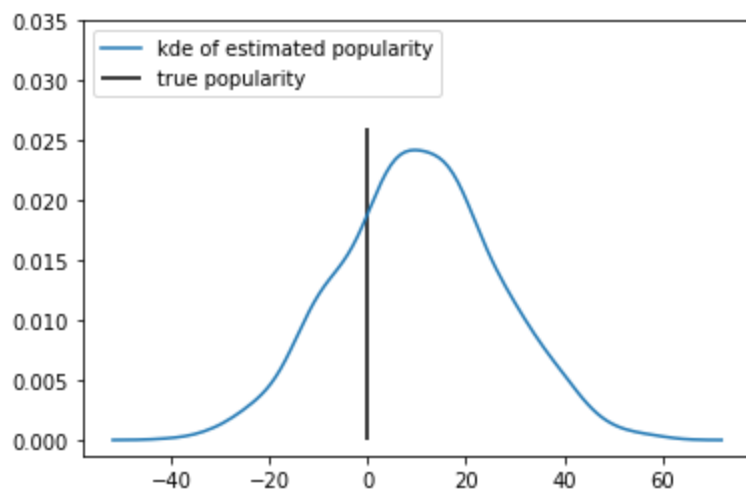
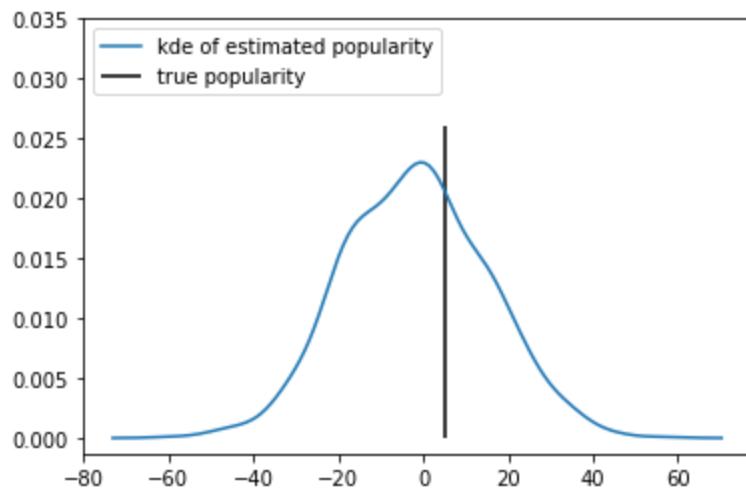
8a: Posterior predictions for linear model

To measure the model's predictive power, I computed the mean squared error (squared error of the true popularity value from a random draw from the posterior distribution for popularity -- notice this is not the same as the squared error of the true value from the expected value of the posterior distribution as it also penalizes high variance in the posterior distribution) across the training/testing data. I also computed average (log) posterior predictive density for both the training and testing sets. These values are shown in the table below:

	Mean squared error	Average p.p.d.
Training set	534.64	-4.21
Testing set	538.45	-4.21

Notice there is very little difference between the values for the training and testing set. This makes sense, since the size of the training set was large, so most characteristics of the testing set were probably already incorporated into the posterior. The mean squared error of 530 or so indicates that the average posterior prediction less than about $\sqrt{530} \approx 23$ away from the true value. The average distance between the *expected value* of the posterior and the true value should be even less. Posterior distributions for five randomly selected data from the testing set, along with their true popularity values, are shown below:





8a: Posterior predictions for logistic model

Average accuracies for draws from the posterior distribution of each of the five cross-validation models used (based on the test data excluded for each), along with log posterior predictive densities (basically the same thing, but averaged over the log values), are shown in the tables below:

Training data

	Accuracy	Average p.p.d.
Fold 1	0.90	-0.17
Fold 2	0.90	-0.18
Fold 3	0.90	-0.18
Fold 4	0.90	-0.17
Fold 5	0.90	-0.17
Average, all folds	0.90	-0.17

Test data

	Accuracy	Average p.p.d.
Fold 1	0.71	-0.45
Fold 2	0.74	-0.39
Fold 3	0.79	-0.29
Fold 4	0.61	-0.71
Fold 5	0.74	-0.38
Average, all folds	0.72	-0.45

The predictive accuracy is much higher on the training data relative to the test data. This is likely due to the rather small sample size (number of songs in the “liked” category). Incorporating just a bit more data can cause significant updates to the posterior.