Paper replication: Bayesian Inference of Arrival Rate and Substitution Behavior from Sales Transaction Data with Stockouts

Marija

Evgeniya Zakharova

Maria Jose

July 29, 2022

Abstract

As a group, we embarked on a journey to replicate the paper from Letham et al. (2016) dealing with the prediction of real demand while taking into consideration stockout. During our process, we encountered several challenges and overcame them. We coded from scratch the Bayesian hierarchical model proposed by the paper for inferring the underlying customer arrival rate and choice model from sales transaction data and the corresponding stock levels. The Bayesian model uses a nonhomogeneous Poisson process to allow the arrival rate to vary throughout the day and a nonparametric choice model. The model parameters are inferred using the Markov Chain Monte Carlo (MCMC) tool PyMC3. For the MCMC, we generated data, coded the log-likelihood function and included the prior distributions. Also, applied the empirical data from a bakery to the Bayesian Model. We ultimately got some results from the MCMC.
Keywords: *Bayesian Model; MCMC; Empirical Data; Arrival Rate; Choice Model.*

1 Introduction

Our report documents the attempt of replicating a paper related to modelling a real problem, using Markov Chain Monte Carlo (MCMC) for sampling and testing the model with empirical data. After a thorough search in Google Scholar, we found a paper presented at the International Conference on Knowledge Discovery and Data Mining. The title of the paper is "Bayesian inference of arrival rate and substitution behaviour from sales transaction data with stockouts" authored by Letham, B., Letham, L. M., & Rudin, C. (2016). It provides a Bayesian hierarchical model for inferring the underlying customer arrival rate and choice model from sales transaction data and the corresponding stock levels (Letham et al. 2016).

The underlying problem that the paper aims to address is very common for retailers. When a customer arrives at a store, they are looking for a preferred item to purchase, however, if the item is not available, they would either purchase another substitute item or leave without a purchase. This situation can be prevented by having a stocking strategy aligned to the real demand for the items. Traditionally, stocking strategies depend on the level of sales of an item and its available stock level. This paper tries to predict what customer segments come to the store and how the arrival times are distributed during the day. The results from the simulations would estimate the parameters for arrival rate and customer segments' distribution.

During our process, we did not have the original coding from the paper which meant that we must replicate the code for the model described in the paper by ourselves. The empirical dataset used in this paper was available on the author´s GitHub repository[1]. Overall, we were able to understand and interpret the mathematical model described on the paper and able to translate it into code, however, in the implementation of the MCMC via PyMC3 with our customized likelihood, we encountered some limitations. The main limitation was related to MCMC, as the paper used the stochastic gradient Riemannian Langevin dynamics (SGRLD) algorithm of Patterson and Teh (2013). This MCMC algorithm does not require the calculation of the full likelihood function at every MCMC iteration which reduces the computational time and the risk of recursion error.

In the following sections, we will provide a summary of Letham´s paper with a focus on their proposed hierarchical Bayesian model. Next, we will describe the model implementation, the dataset generated during the simulation and the empirical data used, followed by the model estimation results and finally a discussion section with some reflections and insights on the paper and our process.

2 Summary of the paper

As mentioned before, a common challenge faced by retailers is to understand the customer preferences in the presence of stockouts. The paper proposes a hierarchical Bayesian model for inferring the underlying customer arrival rate and the customer´s choice model. It uses the sales transaction data and the corresponding stock levels to infer the arrival rate and choice model. These data are usually easily available for retailers. The paper describes two options for the Poisson process of the customer arrival rate, however, in our experiment, we only focused on the nonhomogeneous Poisson process. Furthermore, the paper also describes three different choice models for customer substitution behaviour, however, in our experiment we focused only on the non-parametric choice model.

2.1 Customer arrival rate

The first component of the Bayesian model is the customer arrival rate, which includes the customers that arrive at the retailer and leave with or without a purchase. The arrival rate of customers is nonhomogeneous in time, meaning the customers arrive at the store at different times during the day with peak times, which is very common for many retailers. To model this arrival rate a nonhomogeneous Poisson process (NHPP) was used. An NHPP is a generalization of the Poisson process that allows for the intensity to be described by a function $\lambda(t) \geq 0$ as opposed to being constant (Pinsky and Karlin 2011). The function of the arrival rate shows several peaks at the busiest time and decays at the end of the day. The intensity function is parametrized with η^{θ}, and uses the following derivative of the

[1]https://github.com/bletham/bakery

Hill equation (Goutelle et al. 2008):

$$\lambda\left(t \mid \eta^{\sigma}\right)=\eta_{1}^{\sigma} \frac{\left(\frac{\eta_{2}^{\sigma}}{\eta_{3}^{\sigma}}\right)\left(\frac{t}{\eta_{3}^{\sigma}}\right)^{\eta_{2}^{\sigma}-1}}{\left(1+\left(\frac{t}{\eta_{3}^{\sigma}}\right)^{\eta_{2}^{\sigma}}\right)^{2}} \tag{1}$$

This first component aims to infer the posterior of each η^{θ}. To accomplish this the paper uses the conditional density function for the NHPP arrivals, which for a log-likelihood function for arrival times $t_1, ..., t_m$ over interval $[0, T]$ is:

$$\log p(t_1, ..., t_m) = \sum_{j=1}^{m} log(\lambda(t_j)) - \Lambda(0, T) \tag{2}$$

where $\Lambda(0, T) = \int_0^T \lambda(t_j)dt$.

2.2 Nonparametric Choice Model

The second component of the Bayesian Model is the choice model which describes how the customers substitute when their preferred item is out of stock. Choice models are econometric models describing a customer's choice between several alternatives (Verma and Plaschka 2003). In our experiment, we applied the nonparametric choice model which depends on the stock level of the items $s(t)$. This choice model is presented as an ordered subset of the items ϕ^k. The choice model function which defines whether the preferred item of the customer segment k is in the stock or not is:

$$f_i^{np}(s(t), \phi^k) = \begin{cases} 1 \text{ if } i = min\left\{j \in \{1, ..., |\phi^k|\} : s_{\phi_j^k}(t) = 1\right\} \\ 0 \quad otherwise \end{cases} \tag{3}$$

With this nonparametric choice model, there are no parameters to infer. However, as it deals with different customer segments k, the aim is to infer the distribution of the customer segment θ_k.

2.3 Log-Likelihood

Together with the customer arrival rate and the customer choice model, the authors explained the log-likelihood function used for the Bayesian model. Before diving into the log-likelihood function, first, we need transaction data. This data is the recorded time purchases for each item represented as t. For the simulation, this transaction data was generated following an algorithm described in this report in the section called **Dataset**. For the empirical data, the transaction data t was provided by a bakery shop for three types of cookies.

The log-likelihood function of t is:

$$\log p(t \mid \eta, \theta, N, T) = \sum_{\sigma=1}^{S} \sum_{l=1}^{L^\sigma} \sum_{i=1}^{n} \left(\sum_{j=1}^{m_i^{\sigma,l}} log(\tilde{\lambda}_i^{\sigma,l}(t_{i,j}^{\sigma,l})) - \tilde{\Lambda}_i^{\sigma,l}(0,T) \right) \tag{4}$$

This log-likelihood includes the transaction data, the NHPP for customer arrival rate and the nonparametric choice model for each customer segment. This function is slightly modified from the paper to exclude the nonrelevant parameters for our experiment. The important quantity in this log-likelihood is the observed purchase rate, which is the arrival rate multiplied by the purchase probability:

$$\tilde{\lambda}_i^{\sigma,l}(t) = \lambda\left(t \mid \eta^\sigma\right) \pi_i(s(t \mid t^{\sigma,l}, N^{\sigma,l}), \theta^\sigma) \tag{5}$$

And the purchase probability $\pi_i(s(t \mid t^{\sigma,l}, N^{\sigma,l}), \theta^\sigma)$ is the probability that a randomly chosen customer purchases a product or does not purchase it:

$$\sum_{k=1}^{K} \theta_k^\sigma \, f_i(s(t \mid t^{\sigma,l}, N^{\sigma,l}), \phi^k) \tag{6}$$

The second quantity of the log-likelihood function is the corresponding mean function:

$$\tilde{\Lambda}_i^{\sigma,l}(0,T) = \int_0^T \tilde{\lambda}_i^{\sigma,l}(t)dt \tag{7}$$

The list of all the notations can be found in Appendix B.

2.4 Prior

For the Bayesian model, it is also required the prior distribution of the parameters that are going to be inferred. The prior distributions for the two types of parameters are:

$$\eta_v^\sigma \sim \text{Uniform}(\delta^v), \;\; v = 1, ..., |\eta^\sigma|, \;\; \sigma = 1, ..., S \tag{8}$$

$$\theta \sim \text{Dirichlet}(\alpha) \tag{9}$$

The prior distribution of the η^σ is uniform and the interval δ large enough to not be restrictive. The α is a prior hyperparameter for θ. Given that no expert knowledge is available for this study and our experiment a uniform prior distribution was used, setting the hyperparameter to be a vector of ones.

The prior distribution can be computed as:

$$p\;(\;\eta, \theta \mid \alpha, \delta\;) = p\;(\theta \mid \alpha) \left(\prod_{\sigma=1}^{S} \prod_{v=1}^{|\eta^\sigma|} p\;(\eta_v^\sigma \mid \delta^v) \right) \tag{10}$$

Finally, with the log-likelihood function and the prior, now the posterior distributions of the parameters η^σ and θ can be inferred.

3 Model Implementation

In our experiment, we were able to implement the same Bayesian model as the one described in the paper. We chose Python as the programming language to build the code from scratch and Google Colab as the development environment which facilitated the collaboration among the members of the team. It also uses packages and libraries which cannot be installed on personal computers.

Our implementation process consisted of several trials and errors and researching online to find the best packages, best methods and best coding practices to use. Our final version of the code is in the additional attachment included with this report. This section will outline the different packages, and classes, as well as their attributes and methods implemented for the simulation.

There were several packages we used during this process including packages for the MCMC. The required packages imported for running were:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import scipy
from scipy.stats import norm
import seaborn as sns
from scipy.integrate import quad
import math
import random
#PYMC3
import pymc3 as pm
from pymc3.math import log, exp, where
import theano.tensor as T
```

Once the packages were installed and imported, the next step consisted of the implementation of an algorithm to generate transactional data following the steps described in the next section called **Dataset**. Then computing the log-likelihood function. The code we elaborated was written in an Object-Oriented way which consisted of two main Classes:

1. Model() comprises the data generation algorithm and the log-likelihood function.

2. Nhpp() which is used in the generation of the transactional data and included within the Model() class

For the first class, Model() class has the following global attributes as described in the paper:

```
self.etas = []
self.nhpp = Nhpp()  #object for non-homogeneous process
self.init_stock_number=[]
self.transactions = [0, 0, 0]  #transaction per item number
self.transactions_times = [[], [], []]  #transaction time per item
self.items_list = [1,2,3]
self.preferenced_lists = [[1],[2],[3],[1,2],[1,3],[2,1],[2,3],[3,1],[3,2]]
self.customer_segments_num = 9
# to get purchase times
self.theta_first = [0.33,0.0,0.0,0.33,0.0,0.0,0.0,0.0,0.33]
self.alpha = [1,0.001,0.001,1,0.001,0.001,0.001,0.001,1]
self.theta=list(np.random.dirichlet(self.alpha))
self.time = 1000
```

All these attributes are necessary and follow the different attributes described in the paper and also in the next section **Dataset**. Additionally, the Model() class includes the following methods where we added a brief explanation for each:

```
#sets the global etas attributes
 def get_etas(self)
#requires parameter t for time and it returns #lambda values needed for
#arrival Hill rate
 def get_lambda(self,t)
#returns hill_rate_lambda
 def get_list_of_lambdas(self)
#formats lambda variables in dictionaries
 def format_lambdas(self)
#requires 2 parameters (list of times and a specific time period for l)
#and it sets self.transactions_times (a global attribute)
 def define_time_purchases(self,times,l)
#returns arrival times by using lambdas from get_lambda() method
#and using Nhpp object and its method get_arrivals()
 def get_arrival_times(self)
#returns 1 if item in stock otherwise zero
 def stock_indicator(self,elem,l)
#returns 1 if preferred item was purchased otherwise 0
 def purchase(self,item,customer_segment,arrival_time,l)
#returns variable pi, later used in likelihood
 def get_pi(self,item,arrival_time)
#returns purchase rate for specific item
 def get_purchase_rate(self,item,arrival_time)
#used in get_pi_for_integral() method
 def purchase_for_pi_integral(self,item,customer_segment,arrival_time)
```

```
#used in get_purchase_rate() method, returns pi based on thetas
 def get_pi_for_integral(self,item,arrival_time)
#returns lambdas for non-homogenous process, used in get_mean_purchase_rate()
 def get_integral_mean_rate(self,t,item)
#returns mean purchase rate for the item
 def get_mean_purchase_rate(self,item)
#returns likelihood of purchasing an item
 def likelihood_model(self,store,time_period,items_num,times_per_item)
```

The different methods which are included in Model() class contribute as building blocks to achieving the goal of running the log-likelihood function with the generated transactional data. The second class applied is the Nhpp(). This class was taken from a GitHub repository from Campbell (2021) to obtain the get_arrivals() method which generates a sequence from a nonhomogeneous Poisson process specified by a piecewise linear function determined from the knots parameter. The knots should specify the (domain, range) of each segment's knot. Parameters for this method were extremely important for the arrival rate so we will use the opportunity to explain them here in more detail:

- knots: dictionary where keys and values should all be numeric.
- func: a function that takes a single float and returns a float.
- func_args: additional args to func
- fung_kwargs: additional kwargs to func

This get_arrrivals() method returns a list of floats. The methods of the class Nhpp() are:

```
#based on the knots specified for a piecewise linear function and a point
#in the domain 't', return the value of the piecewise linear function
 def _get_piecewise_val(self,knots,t)
#sorts out the key-value pairs
 def _get_sorted_pairs(self,dic)
#gets the slopes of each section of the rate function
 def _get_rate_slopes(self,knot_vals,knot_times)
#returns a list of integrated rate values
 def _get_integrated_rate_values(self,knot_vals,knot_times)
#checks the type of the value passed, the value must be a dictionary,
#otherwise, it will raise a type error
 def _check_is_dict(self,a)
#checks if the values are positive, otherwise it will raise a value error
 def _check_arrivals_positive(self,knot_vals)
```

With the two main classes completed, we can start testing this code in order to generate the results of the transactional data and the log-likelihood function. The code for testing was the following:

```
model1 = Model()
model1.get_etas()
arrivals_per_time_period = []
time_period = 5
for l in range(time_period):
times = model1.get_arrival_times()
arrivals_per_time_period.append(times)
transactions_times_per_time_period = []
for l in range(time_period):
transactions = model1.define_time_purchases(arrivals_per_time_period[l],l)
transactions_times_per_time_period.append(transactions)
transactions_times_per_time_period
x=model1.likelihood_model(1,time_period,3,transactions_times_per_time_period)
print(x)
```

We implemented the Bayesian model, and the next step is to use the MCMC with the model and the log-likelihood function. The MCMC and the results are detailed in the section on **Model estimation**.

4 Dataset

To implement the Bayesian model described in Letham et al. (2016) paper, it is required to have a dataset. We deal with two types of datasets, the one that was generated, and the empirical dataset provided by the bakery as mentioned in the paper. Both types of datasets contain one column where each row records an exact date and time. These data represent the recorded day and time of the purchase of an item. For the generated dataset, we generated one list with l time periods, each of the time periods contains 3 lists of transaction data per item. And for the empirical data, we obtained 3 different files each for a type of cookie, which we also represented as one list with l time periods and lists of transaction data for each item.

4.1 Generated dataset

The generated dataset was used for the simulation. The paper details the steps taken to generate this data as follows:

- For store $\sigma = 1, ..., S$:
  - For time period $l = 1, ..., L^\sigma$:
    - Sample customer arrival times $\tilde{t}_1^{\sigma,l}, ..., \tilde{t}_{\tilde{m}^{\sigma,l}}^{\sigma,l} \sim \mathrm{NHPP}(\lambda(t \mid \eta^\sigma), T)$.
    - For customer arrival $j = 1, ..., \tilde{m}^{\sigma,l}$:
      - Sample this customer´s segment as $k \sim \mathrm{Multinomial}(\theta^\sigma)$
      - Choose item i for this customer´s purchase with probability $f_i(s(\tilde{t}_j^{\sigma,l}|t^{\sigma,l}, N^{\sigma,l}), \phi^k)$, or the no-purchase option with probability $1-\sum_{i=1}^{n} f_i(s(\tilde{t}_j^{\sigma,l}|t^{\sigma,l}, N^{\sigma,l}), \phi^k)$.
      - If the item i purchased, add the time to $t_i^{\sigma,l}$.

In our experiment, we followed all these steps as described in the previous section named **Model implementation**. For every simulation we ran, a unique dataset was generated, meaning no dataset generated is the same as another. So, for one store and for 2 time periods, we sampled the times within each time period when customers arrive following a nonhomogeneous Poisson process within time ranges T from 0 to 1000, these arrivals include purchase and no purchase samples. Next, for each customer arrival, we sampled the customer´s segment following a multinomial distribution, meaning assigning one customer segment from the defined 3 different customer segments. We defined the three customer segments as $\phi^1 = 1, \phi^2 = 2, \phi^3 = 3$ which indicates the first preferred item by the customer segment, if the first item is not available they will leave without a purchase. In this case, we had 3 different items and we did not use a list of preferred items, meaning no substitution if the first item is not available. Each customer segment has an equal probability of arrival. In the next step, the choice model is involved, where any of the 3 items is chosen given their stock availability. The stock availability varies for each time period with a uniform distribution between 0 and 500.

As mentioned before, every simulation results in a different distribution for the number of purchases in the generated data. On the following graph we can observe the data generated for one time period and the number of purchases for all three items:

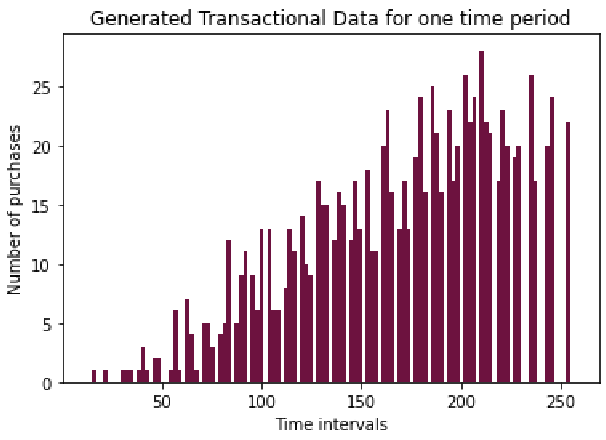

Figure 1: Generated transactional data for one time period

From the graph of the generated data, we can observe there is a nonhomogeneous distribution for the purchase time.

4.2 Empirical dataset

For the empirical dataset provided by the bakery and obtained from Letham´s GitHub repository, we obtained three different datasets each for a type of cookie. The names of the files were:

1. "chocolate_chip_cookie_transactions.csv"
2. "double_chocolate_cookie_transactions.csv"
3. "oatmeal_cookie_transactions.csv"

The empirical data had to be processed by us to comply with what the paper mentions regarding its characteristics. The paper indicates that the dataset included all purchase times for 151 days, treating each day as a time period between 11:00 a.m. to 7:00 p.m. with a total of 4084 purchases. After further analysis of the empirical data, there were recorded purchase times outside this time range of 11:00 a.m. to 7:00 p.m. which we had to filter out. Also, many time periods included very few purchases (less than 5 purchases during a day). For these reasons, in our experiment we had to make an assumption of combining several time periods in order to obtain a larger amount of purchase times, selecting those days with more than 5 purchase times and merging them as a one time period. Which resulted in one rich time period with multiple purchases throughout the day. This process was done for all three items.

Regarding the stock data, it was not available for the bakery. For this reason, in the paper and for our experiment we set the initial stock for each time period equal to the number of purchases for the time period. This assumption is reasonable given that the cookies are perishable baked goods and should be stocked out by the end of the day.

After processing the empirical data, this dataset replaces the generated dataset, and it is included in the log-likelihood function as t. The processed empirical dataset for the three cookie types had a total of 480 time intervals T between 11:00 am and 7:00 pm for one time period, as shown in the graph:

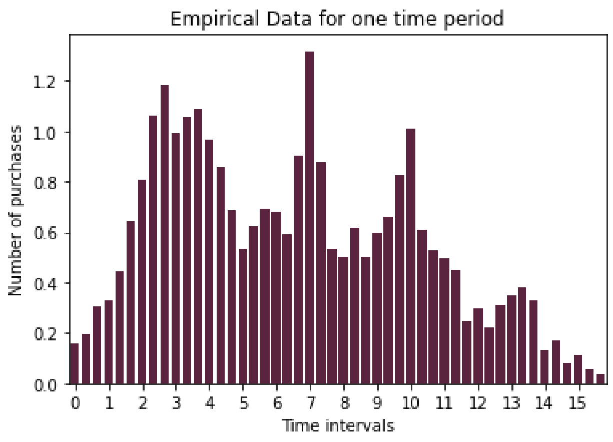

Figure 2: Empirical Data for one time period

We can observe the empirical data is also nonhomogeneous as there are three main peaks during a day which indicates more customers at these time intervals.

5 Model Estimation

Following the model implementation and the dataset generated and empirical dataset, we can continue the process to infer the posterior distribution of the parameters η and θ for the simulation and the empirical test. The next step of our process is the simulation which consists of inputting the model with the generated data and the log-likelihood function in an MCMC tool to get the posterior of the parameters.

5.1 Simulation: MCMC with the generated dataset

We had different software available to implement the MCMC, we decided on PyMC3 as we found good examples of customized likelihood implementation and easier to use in Google Colab. The first experiment involved simulating the parameters with the generated data. For generating data we used the Model() class as described in the previous section **Model implementation**.

First, we generate the dataset with the following code:

```
model1 = Model()
model1.get_etas()
times = model1.get_arrival_times()
transactions_generated = model1.define_time_purchases(times)
```

The transactional data generated had to be reformatted. The reformatting of transactions_generated was needed because the data that is handed to custom distribution for the likelihood expects data to be handed in a specific form such as series or columns of a DataFrame and not as a list of three lists as it is returned from the model1. After reformatting the generated data, the result is 3 series of transactional data ready to be input into PyMC3.

In the PyMC3, the first step consisted of creating an object using pm.Model(). Then, we defined the priors for each parameter that needed to be estimated, in this case for three different η´s and θ´s for customer segments:

```
eta1 = pm.Uniform("eta1", 2000, 4000)
eta2 = pm.Uniform("eta2", 2, 4)
eta3 = pm.Uniform("eta3", 200, 400)
alpha = pm.Dirichlet("Thetas", a=np.ones(k))
 #k=3 (one for each customer segment)
```

All the priors were defined based on the information given in the paper. As in the paper, we only performed one simulation. Also like the paper, in the inference, we used prior hyperparameters eta1 = [2000; 4000], eta2 = [2; 4], and eta3 = [200; 400]. In the case of the customer segment "Thetas", following the paper, they used 3 items with 9 customer segments, each segment with a preferred item list. The segment proportion was set to 1 for the preference lists. We decided to focus on 3 out of 9 customer segments, as the other 6 had a probability of occurring equally to zero as stated in the paper. Therefore, in total, we inferred two types of parameters etas and Thetas which in total resulted in 6 inferred parameters.

The next input in PyMC3 is the functions required for our customized log-likelihood. The list of functions is as follows:

```
def get_lambda(t)
def get_purchase_rate(item,arrival_time)
def get_integral_mean_rate(t,item)
def get_mean_purchase_rate(item)
def likelihood_model(cookie1_purchases,cookie2_purchases,cookie3_purchases)
```

These functions were implemented as methods of the Model() class, so more details on them can be seen in the section on **Model implementation**. As our model was based on our log-likelihood the likelihood_model() function is handed as the model as well as the data (purchase times per each item) described earlier. The object was then used to sample the previously defined distributions: 3 etas and 3 thetas.

Next, we defined trace in PyMC3 as follows:

```
trace = pm.sample(tune=1000, draws=10000, return_inferencedata=True)
```

The explanation for each of the parameters used in the sampling method is:

tune: Markov Chain Monte Carlo samplers are based on the concept of Markov Chains. Markov Chains start from a random distribution and slowly converge to the distribution of the model called stationary distribution (Sinclair 2020). Since we wanted to sample "real" (unbiased) samples from the model, we needed to "tune" (let it converge) the chain. So, by setting tune=1000, we told PyMC3 to let the chain converge to the distribution of our model for 1000 iterations. Once 1000 iterations are completed, it starts drawing from the distribution. This takes us to our next parameter draws.

draws: This parameter indicates PyMC3 how many samples we want to draw from the model's distribution (Markov chain) once the tuning step is completed. So, by setting draws=10000, we specify PyMC3 to draw 10000 samples.

return_inferencedata: It determines whether to return the trace as an arviz.InferenceData (True) object or a MultiTrace (False). The default is set to False, but we switched it to True as this was what was used in the samples we used as research material.

After running the PyMC3 we obtain the following results:

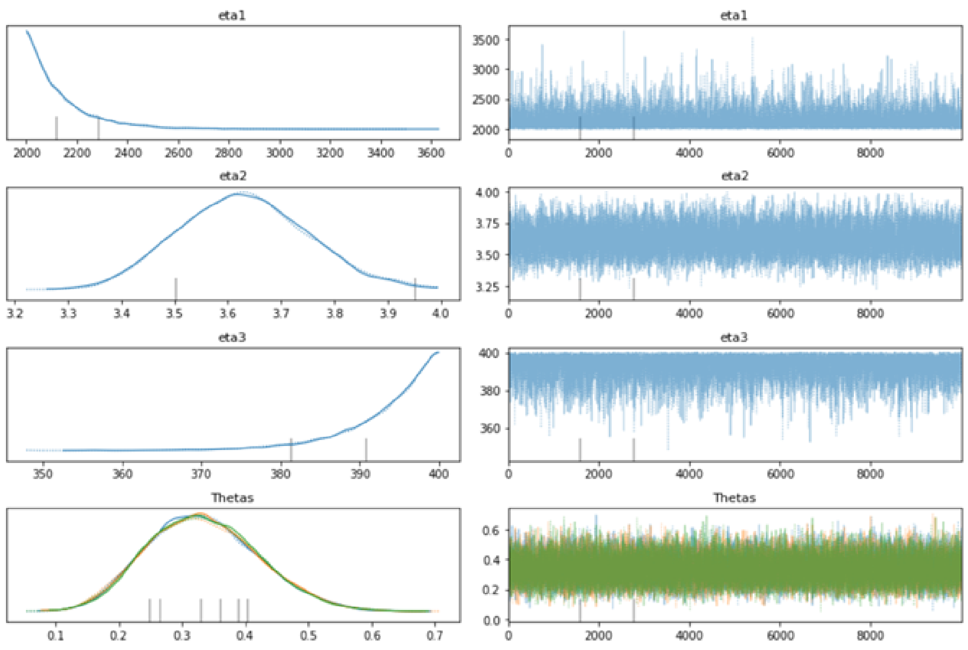

Figure 3: PyMC3 result for generated data with tune of 1000 and 10000 draws

The results obtained for one time period seem realistic to what was expected. Etas are within the range determined and the Thetas are around the proportion estimated of 0.33.

Additionally, we also ran the PyMC3 for more time periods. However, the occurring error *"maximum recursion depth exceeded"* did not allow us to have tune = 1000, draws = 10000. We tried several number combinations and the only one, which completed successfully was time period = 2, a tune of 10 and draws of 100. The results are comparable.

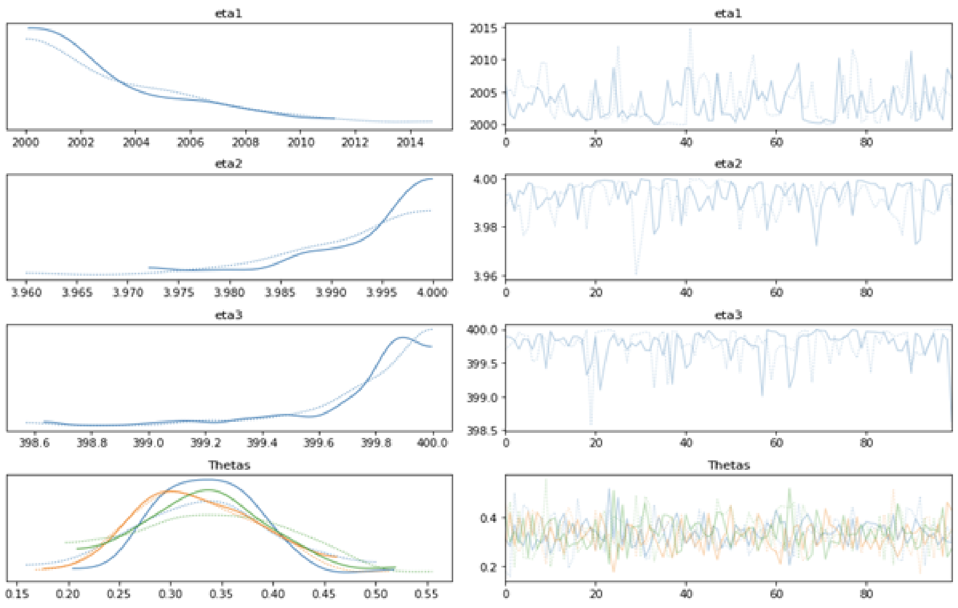

Figure 4: PyMC3 for generated data with a tune of 10 and 100 draws

5.2 Case Study: MCMC with the empirical dataset

For the case study of the bakery with its empirical dataset, we ran PyMC3 the same way as described before with the difference of the empirical data rather than the generated data. Also, we used the setting for the PyMC3 as tune = 1000 and draws = 10000 as they provided better results. We also used the same intervals for the etas. The posterior inference for etas and Thetas is as follows:

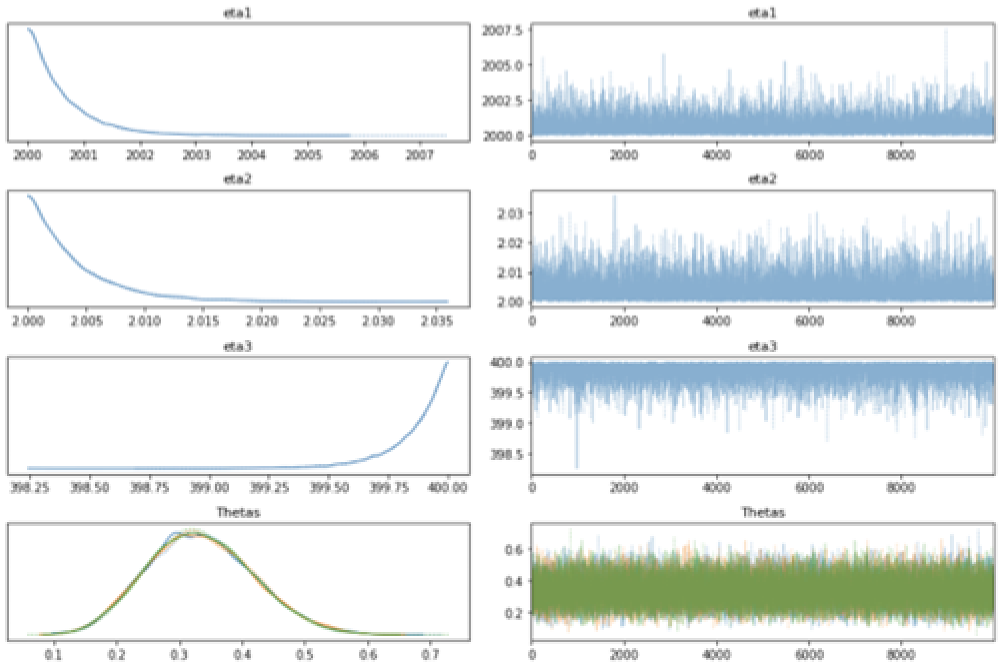

Figure 5: PyMC3 result for empirical data with a tune of 1000 and 10000 draws

6 Discussion

After a lot of time invested and perseverance, we achieved some results. Given the countless limitations we encountered throughout our journey, we are confident in our understanding of the model as described in the paper and all the functions coded to replicate the model, however, we do have some doubts about the quality and accuracy of the results presented. In this section, we outlined the different limitations we encountered and how we overcame them, as well as our contribution.

6.1 Limitations

During our process of estimating the parameters with MCMC, we encountered one main problem. PyMC3 did not allow the usage of sophisticated parameters, which was causing errors with tensors, and recursion, resulting in wrong values for the log-likelihood. Due to this problem, we decided to simplify the list of preferred items for the customer segments. The model itself is the same as the paper, except for the list of preferred items for the customer segments. Instead of the preferred items $\phi^1 = \{1\}, \phi^2 = \{1,2\}, \phi^3 = \{3,2\}$ we used $\phi^1 = \{1\}, \phi^2 = \{2\}, \phi^3 = \{3\}$ for the 3 customer segments. Which does not affect the observed purchase rate or the mean purchase rate. However, the probability that a randomly chosen arrival purchases product i (π_i) becomes 0.33 and now there is no possibility for substitution, which changes the purchase function. After these alterations, running the MCMC was viable, and no error came out. We also kept several time periods, which was also causing problems in the beginning. These were the case for the MCMC with the generated dataset and the empirical dataset.

Another limitation was in the empirical dataset that we had to process. The empirical dataset had 151 time periods. Each time period had 3 lists of items transaction times. Many time periods did not even have transactions with an average of 23, 7, and 3 for each item respectively. The model did not work with this small amount as Google Colab did not allow it to run it because it exceeded the RAM, and it would stop the process and output a *Recursion Error: maximum recursion depth exceeded.*

6.2 Contribution

As a contribution, we focused on what happens when a customer buys more than one of their preferred items, meaning buying 3 chocolate chip cookies instead of one. This number of items purchased is discrete. We used a Poisson distribution where the size was the number of arrivals per day, with a mean of 3. In this case, some customers can leave without a purchase or buy 5 of the same preferred item. In the transactional data list, it would reflect the same purchase time five times for that preferred item.

We followed similar steps as described in this report to generate the data. The time of purchase has been simulated from the time of arrival by selecting one customer segment and checking for the availability of the stock. The stock level is different for each time period. In order to have more purchases at diverse times, the stock level is created by a uniform distribution between 1000 and 2000.

We set this scenario for 5 time periods, with time ranges T from 0 to 1000, 3 different customer segments with their preferred item as $\phi^1 = \{1\}, \phi^2 = \{2\}, \phi^3 = \{3\}$. Each customer segment's arrival had an equal probability. With the generated transactional data, we input the same log-likelihood and prior distribution into PyMC3 and estimated the posterior distribution for etas and Thetas using a tune of 1000 and 10000 draws. The results are presented in the following graph:

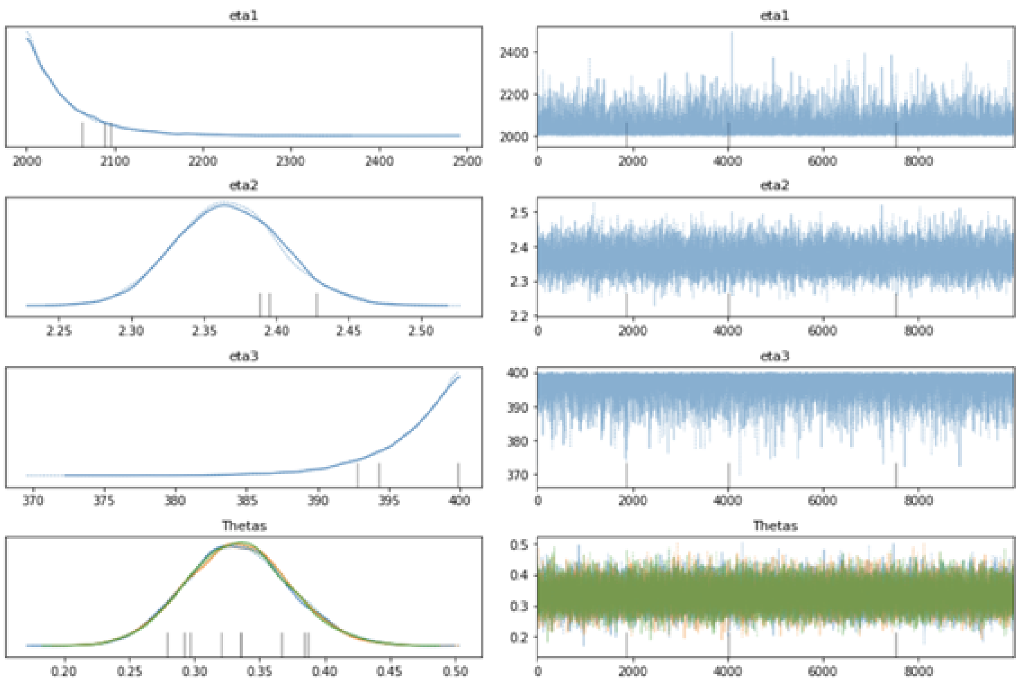

Figure 6: PyMC3 result for contribution with a tune of 1000 and 10000 draws

7 Conclusions

We can say with no doubt that the whole project was a learning process. We had a solid beginning by choosing a paper that we all enjoyed and understood quickly. Then transferring from the paper into code was a challenging process not because of coding itself but because there were a lot of questions along the way that we could not find someone or some website to answer which led us to a lot of trials and errors and taking our best guess. Overall, a process no one wants to repeat any time soon but luckily, we had good teamwork and organized tasks.

Regarding the results we obtained and following from the discussion, we are not very confident about the quality of the results, but we are extremely happy we got some results. There were a lot more comparisons and tests we wanted to do but due to time constraints, it was not possible. One of the main differences between our experiment and the paper was the different MCMC tools used. Adding to it is a large amount of time needed to run the multiple samples.

In addition, we got to see the powerful use of Bayesian models and MCMC in simple applications as in the sale of cookies in a small Bakery. MCMC is preferred over maximum estimate and understanding distributions and probabilities is necessary for future implementations in other fields related to data science.

The main purpose of the paper was to help accurately estimate the demand of retailers. Not considering the stock level can lead the retailer to set the stock for the substitutable items too high while leaving the stock of the stocked-out item too low which could cause a loss of customers and decrease revenue (Letham et al. 2016). Because of the limitation of not being able to include substitution, we cannot conclude this statement.

References

Campbell, Matthew. 2021. “Nhpp.” GitHub. May 16, 2021. https://github.com/matthewmcampbell/nhpp.

Goutelle, Sylvain, Michel Maurin, Florent Rougier, Xavier Barbaut, Laurent Bourguignon, Michel Ducher, and Pascal Maire. 2008. “The Hill Equation: A Review of Its Capabilities in Pharmacological Modelling.” Fundamental and Clinical Pharmacology 22 (6): 633–48. https://doi.org/10.1111/j.1472-8206.2008.00633.x.

Kök, A. Gürhan, and Marshall L. Fisher. 2007. “Demand Estimation and Assortment Optimization under Substitution: Methodology and Application.” Operations Research 55 (6): 1001–21. https://doi.org/10.1287/opre.1070.0409.

Letham, Benjamin, Lydia M. Letham, and Cynthia Rudin. 2016. “Bayesian Inference of Arrival Rate and Substitution Behavior from Sales Transaction Data with Stockouts.” In KDD ’16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Pages 1695–1704. New York, NY, United States: Association for Computing Machinery. https://doi.org/10.1145/2939672.2939810.

Patterson, Sam, and Yee Whye Teh. 2013. “Stochastic Gradient Riemannian Langevin Dynamics on the Probability Simplex.” Neural Information Processing Systems. Curran Associates, Inc. 2013. https://proceedings.neurips.cc/paper/2013/hash/309928d4b100a5d75adff48a9bfc1ddb-Abstract.html.

Pinsky, Mark A., and Samuel Karlin, eds. 2011. “An Introduction to Stochastic Modeling (Fourth Edition).” ScienceDirect. Boston: Academic Press. January 1, 2011. https://www.sciencedirect.com/science/article/pii/B9780123814166000150?via3Dihub.

Ryzin, Garrett van, and Gustavo Vulcano. 2014. “A Market Discovery Algorithm to Estimate a General Class of Nonparametric Choice Models.” Management Science 61 (2): 281–300. https://doi.org/10.1287/mnsc.2014.2040.

Sinclair, Alistair. 2020. “CS294 Partition Functions.” University of California, Berkeley. https://people.eecs.berkeley.edu/ sinclair/cs294/n6.pdf.

Verma, Rohit, and Gerhard Plaschka. 2003. “The Art and Science of Customer-Choice Modeling.” Cornell Hotel and Restaurant Administration Quarterly 44 (5-6): 156–65. https://doi.org/10.1177/0010880403044 00521.

Vulcano, Gustavo, Garrett van Ryzin, and Richard Ratliff. 2012. “Estimating Primary Demand for Substitutable Products from Sales Transaction Data.” Operations Research 60 (2): 313–34. https://doi.org/10.1287/opre.1110.1012.

Appendix B - Table of Notations

Table 2: Table of Notations

$\sigma = 1, ..., S$	Each of S stores
$l = 1, ..., L^\sigma$	Each of L^σ time periods for store σ
T	Time ranges from 0 to T in each time period
$i = 1, ..., n$	Each of n items considered
$m_i^{\sigma,l}$	Number of purchases of item i in time period l at store σ
$j = 1, ..., m_i^{\sigma,l}$	Each of the $m_i^{\sigma,l}$ purchases
$t_i^{\sigma,l}$	Purchase times of item i during time period l at store σ
$t^{\sigma,l}$	All observed purchases (items $i = 1, ..., n$) during time period l at store σ
t	The complete set of purchase time data
$N_i^{\sigma,l}$	Initial stock for item i in time period l at store σ
$N^{\sigma,l}$	Initial stocks of all items in time period l at store σ
N	The complete set of initial stock data
$s_i(t \mid t^{\sigma,l}, N^{\sigma,l})$	Indicator function of the stock of item i at time t, given purchase times $t^{\sigma,l}$ and initial stocks $N^{\sigma,l}$
η^σ	Rate function parameters for store σ
η	Rate function parameters for all stores
$\lambda(t \mid \eta^\sigma)$	Arrival rate at time t, given parameters η^σ
$\Lambda(t_1, t_2 \mid \eta^\sigma)$	Integral of arrival rate function from t_1 to t_2
$k = 1, ..., K$	Each of K customer segments
ϕ^k	Choice model parameter relating to customer preference across items, for customer segment k. For the nonparametric model, this is an ordered set of items
ϕ	Choice model parameters ϕ^k for all customer segments
$f_i(s(t), \phi^k)$	Choice model - the probability a customer purchases item i given stock $s(t)$ and choice model parameters ϕ^k
θ^σ	Customer segment distribution for store σ
θ	Customer segment distributions for all stores
$\tilde{m}^{\sigma,l}$	Total number of arrivals in time period l at store σ
$\tilde{t}_1^{\sigma,l}, ..., \tilde{t}_{\tilde{m}^{\sigma,l}}^{\sigma,l}$	The arrival times in time period l at store σ
$\tilde{\lambda}_i^{\sigma,l}(t)$	The purchase rate for item i at time t in time period l at store σ
$\tilde{\Lambda}_i^{\sigma,l}(t_1, t_2)$	Integral of the purchase rate from t_1 to t_2, for item i in time period l at store σ
$\pi_i(s(t \mid t^{\sigma,l}, N^{\sigma,l}), \phi, \theta^\sigma)$	Probability that an arrival purchases item i, or leaves as no-purchase for $i = 0$
α	Prior hyperparameter for θ
δ^v	Prior hyperparameter for η_v^σ
$p(t \mid \eta, \theta, \phi, N, T)$	The likelihood
$p(\eta, \theta, \phi \mid \alpha, \delta)$	The prior
$p(\eta, \theta, \phi \mid t, \alpha, \delta, N, T)$	The posterior