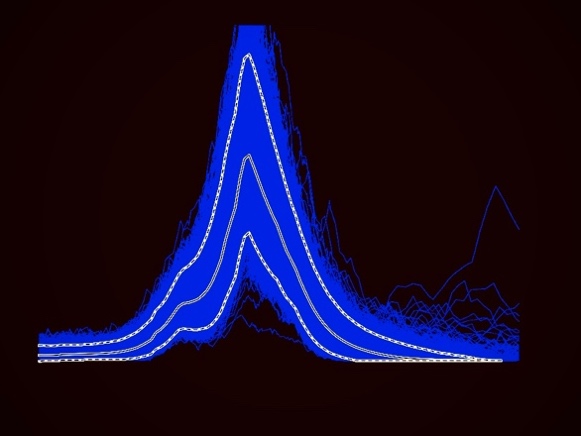
**Doing business, the Bayesian way**

Bayesian modeling is a less known yet a very promising area of Statistics and Data Science. It may come strong where traditional approaches fail and even bring a few additional perks on top of it. In this blog we will consider an illustrative business case showing why Bayesian modeling and the related tech (such as PyMC3 library for Python) may be an impactful set of tools to keep in your data arsenal.



**Theory in a nutshell**. In essence, Bayesian modeling implies using a form of Bayes Theorem to acquire information about what the posterior distributions of model parameters could be given the observed data. In other words, we aim to make the most of the available evidence to update our structural understanding of how a certain process / problem works. This approach presents several advantages: unlike traditional (frequentist) modeling approaches, there is less needs to make questionable assumptions before making any inference or to set arbitrary p-value thresholds. Instead, we transparently indicate what we might know about the problem and the model beforehand (the priors) and how the data can in general be produced by different versions of the model (the likelihood). The rest is taken care of by Bayes Theorem: our prior believes about the model parameters get updated given the observed data. As the result, we acquire not just point estimates but whole distributions of possible model parameters with an indication of how much uncertainty there is about their values – a valuable quantification of how informative the observed data was. This can in its turn translate into irreplaceable business insights such as not just average predicted values, but a range of such values coupled with associated levels of certainty.

**The business case**. To better understand the value and ideas behind Bayesian modeling, we will further consider an illustrative practical use case that mimics a range of common real life business problems. Let us consider ourselves working for a (fictional) B2B service provider company “Virtuoso B.V.” which operates in France, Germany, and Italy. It offers services for varying prices to clients in each country and is primarily interested to learn from the available data how likely each regional client is to purchase services as the price increases. Their data consists of accepted and refused offers in each country for various prices (in millions of fictional digital currency *bigcoins*):

Table

Description automatically generated **Table 1 Sample of the B2B sales data**

**Understanding the problem**. Before we continue any further, it is important to identify some potential problems in Virtuoso’s operations, based on which we can suggest our data-driven solutions. Upon a closer look at the data, we can quickly identify that the sales are not equally successful across all countries. Furthermore, there seem to be cross-country differences in clients’ preferences and budgets. Figure 1 illustrates these two observations for each country (with 95% confidence intervals helping to acknowledge variability in the data and potential data scarcity). Please note that all data analysis in Python as well as graph design and following Bayesian modeling are openly available in [this GitHub repo](https://github.com/VadimNelidov/bayesian_modeling_blog).

Chart, box and whisker chart

Description automatically generated

Figure 1: Two major observations from the data

Clearly, if Virtuoso B.V. is using the same sales and pricing strategies across these three countries, this may explain the relative difference in sales. In fact, proportional to the number of offers in each country, revenues in Italy are more than twice as high as in Germany, which most certainly leaves some room for improvement. On the positive side, having data with both successful and unsuccessful offers in each country for various prices provides a perfect ground for using Bayesian analysis to understand cross-country differences better and develop more personalized sales strategies that would take all available information into account. A valuable distinction of the Bayesian approach is that it would allow us to not only construct a forecasting model (that will help with suggesting better prices in each country), but that it practically allows us to use our initial believes about each country and the available data to confirm or refute these believes and acquire the true picture. So, let’s get to it.

A picture containing diagram

Description automatically generated

Figure 2: Bayes formula in the language of modeling

**Engineering a Bayesian model.** Our goal here will be to apply Bayesian modeling such that we will learn from the data about the relationship between prices and offer acceptance in each country. In Bayesian terms, we are after the posterior (post-learning) distributions of price sensitivity parameters (left part of Figure 2). As the Bayes' formula for modeling shown in Figure 2 suggests, to do so we will need to determine 3 important pieces.

1. is the “likelihood distribution” – the statistical relationship that indicates how *likely* the data is, given our model. With some experience, this component follows from the problem itself: service purchases can be considered binary events similar to coin flips, which is commonly modelled using Bernoulli likelihood.

2. is the “prior distribution” – another statistical relationship that acknowledges all initial information and believes that we have about the problem. In case of its absence, we would deal with “uninformed priors” essentially indicating that everything is possible. In our problem we may actually have some useful prior information that we may want to use. For instance, there may be a price above which nobody would ever buy our services. As the price *decreases* below this point, the probability of a purchase would *increase* but possibly to a different extent in each country. We will later incorporate this logic into the prior parameters of our model.

3. is the most technical and least intuitive parameter. The good news is that it is basically a data-dependent constant value that can mostly be ignored during practical modeling as we will deal with Bayesian optimization algorithms that optimize the problem up to a constant. This constant would affect the scale of the posterior distribution but not the shape, which is what we are after.

Bayes theorem essentially tells us how to combine these factors in a single equation that reconstructs the posterior. As a result, the observed data reshapes our prior distribution into a more *informed* one. Understanding this means understanding most of Bayesian modeling!

Now let’s see how all this combines in code into an actual usable model. To achieve that, we will use Python’s PyMC3 package that is excellent for probabilistic modeling. It may appear harder to learn than the usual “fit-predict” libraries like Scikit-Learn, but it allows construction of highly customizable probabilistic models. The code below translates our above intuition into a ready-to-optimize Bayesian framework:

Text

Description automatically generated

Figure 3: First (pooled) model in PyMC3

**Understanding PyMC3**. Here we deal with a model definition inside a context manager. We define a generic PyMC3 model and named it “pooled\_model” (since to begin with, we will consider all countries combined together as one). Then we start to define model parameters one by one. A Bernoulli likelihood requires a set of real observation (y\_train) and a definition of event probabilities. For the latter we chose a common logistic probability function, that essentially allows us to translate our previous intuition into a simple equation: as the price increases, the probability will fall proportional to *b1* coefficient, while *b0* will affect the maximum prices which can possibly be accepted. Then their combination is translated into a 0 to 1 probability inside a logistic function.

The interesting part here is that we openly admit that we do not know exactly what *b0* and *b1* should actually be. But we do incorporate some broad intuition for them: *b0* could probably be a value drawn from a generic normal distribution with mean 0 and std of 100, while *b1* has to be positive (otherwise higher prices would increase offer acceptance chances) and otherwise also somewhere close to [0; 100). The good news is that if we are somewhat wrong about this, the bayes formula will correct us towards the more realistic parameter values. This makes Bayesian modeling not only very flexible, but also very transparent and a lower-risk approach. Besides, if we have some important domain knowledge and believes, we can also incorporate it here. For example, if we knew that there is a legal maximum for what price can be set – we could use it to further restrict our parameter priors.

**Pooled Bayesian model**. All things combined, this *pooled* model will tell us how offer acceptance likelihood is related to various price levels in all countries *combined*. All we need to do is apply one of the (sampling) algorithms that take care of applying the Bayes’ rule to our problem and producing the posterior distribution. The algorithmic approach that we use here is the famous Markov chain Monte Carlo (MCMC) algorithm with a NUTs sampler behind the hood. While the exact math behind it will be out of scope of this post, what we need to know is that it is a random-walk-type algorithm that sequentially samples values to construct the posterior distribution given the data and the prior believes. Provided a reasonable model, and enough time for tuning & sampling, it is proven that MCMC converges to the real posterior distribution without any biases. In terms of code, we acquire these MCMC samples using one single line in Figure 3, and this is the result:

Icon

Description automatically generated

Figure 4: Posterior distributions of parameters from the pooled model

Based on the data that we supplied to the model, our prior distributions of *b0* and *b1* have been substantially updated. It can now be said with high confidence that *on average* in all countries combined *b1* is likely between 1 and 2, while *b0* is between 6 and 11. This implies a valuable result: if we take these lower and higher confidence bounds for both parameters, we can learn that our clients are very unlikely to accept prices above 13. At the same time, prices under 10 would have at least a 50% chance of being accepted. Unlike other modeling approaches, Bayesian modeling allows us here to get not just point estimates, but whole distributions of acceptance likelihoods for various prices that would allow us to directly take uncertainty into account when making difficult decisions.

On top of such posterior analyses, we can also directly use this model for traditional forecasting and estimate its accuracy (and other metrics). Before optimizing this model, we have split the full dataset into a 2-to-1 train-test split, and only training data has so far been supplied to PyMC3. Now we can directly access the posterior samples from this model (using ‘posterior’ variable) to further make predictions for various prices. While PyMC3 has built-in methods to do so, it would be more illustrative to reconstruct probabilities ourselves. If we do want a point-estimate, we can infer a single property of each posterior distribution such as the mean. As seen in Figure 4, the mean of *b0* is 8.5 and of *b1* is 1.3. Consequently, for a price of 5 this would translate into *logit(8.5 – 1.3 \* 5) = 0.85* where logit is the standard [logit function](https://en.wikipedia.org/wiki/Logit). In other words, a price of 5 would have a 85% chance to be accepted in each country on average. Similarly, we can calculate probabilities for each of the *X\_test* prices, convert them to 1 or 0 (e.g. based on a stand 0.5 threshold) and compare them to the real *y\_test* values. As a result, this “pooled Bayesian model” has an accuracy of 85.4% as well as a weighted f1-score of 85%, which can be further improved if we change aspects such as the chosen probability threshold or the posterior statistics (e.g. some quantile value like the median rather than the mean).

**A hierarchal framework**. But we can do even better than this. So far, we have pooled all country data together, constraining ourselves to only learn the ‘average truth’. But Bayesian modeling particularly shines when we let it handle several distinct yet related problems together. Such frameworks are usually referred to as ‘hierarchal’ – we assume that there is a certain hierarchy in how people react to our prices in different countries. These reactions are probably similar yet somewhat different. The way we can incorporate this idea into our Bayesian model is by assuming that our three countries have different *b0* and *b1*, that are yet related to each other by being drawn from the same common distributions. This way we do not have to assume that Italy, Germany and France are practically the same, but we also do not want to assume that they have completely nothing in common. As a result, such hierarchal models will learn the differences, while acknowledging the similarities present in the data. This presents a valuable middle ground that other modeling approaches rarely can provide. In terms of code, our extended model would take this form:

**Text

Description automatically generated**

Figure 5: Second (hierarchal) model in PyMC3

This model may look more complicated at first glance, but in fact it is built around a very similar idea. Again, Bernoulli likelihood combines the observed *y\_train* data with the logistic probabilities that combine *b0* and *b1*(this time different yet related for each country). Each *b0* and *b1* are still drawn from the same type of prior distributions as before but now not fixed ones but ‘flexible’ distributions with unknown mu and std (which all have priors of their own above). Those new additional priors are very generic (uninformed) as we might not have a good idea how mu and std typically should be distributed. A normal distribution around zero with a large std is a common choice in this case. Finally, there is an additional technicality of assigning an index to each country and inserting it into probability definitions to let PyMC3 use the correct betas for each country. This “*ids\_train*” variable must have the same order of countries as prices in *X\_train*.

Now we are again ready to generate posterior distribution samples using MCMC. Note that for more complex models like this one, it may be necessary to experiment with the number of tuning steps, draws and a few other parameters. Fortunately, if anything goes wrong, PyMC3 provides direct feedback and recommendations in the form of warnings after sampling. After the MCMC sampling succeeds, we can inspect the unique parameter distributions generated for each country:

Chart

Description automatically generatedA picture containing shape

Description automatically generated

While all *b0* seem very similar, there is some more noticeable distinction in *b1* parameters. This actually makes practical sense – *b1* indicated how sensitive clients in each country are to price increases, and the posterior with the highest mean is the one for Germany – where Virtuoso B.V. had by far lowest sales. By contrast, Italy, where sales were the highest, appears to be least sensitive to price increases. This is a rather unique advantage of the hierarchal Bayesian framework – as together with the forecasting model we gain diverse analytical information about the problem we are dealing with.

Now that each country has its own parameter distributions inferred, we can also see how similar prices would be received in different countries. Let’s say Virtuoso B.V. wants to set a fixed price of 7 (millions of Bigcoins) for all of its clients and offer a (temporary) discount of 2 for (more price-sensitive) clients in Germany. We can calculate not just point estimates but whole probability distributions for how likely each country’s residents would be to accept such prices:

Chart, line chart

Description automatically generated

Interestingly, while German clients have their price reduced to 5, they still appear least likely to actually purchase our services (with a mean of ~20%). Perhaps a higher discount was necessary? Italian clients are at the same time more than 80% likely to accept their price of 7. Not only distributions’ means are valuable here though: we can also infer that the probability distribution of France at these prices is by far the most uncertain one. Knowing uncertainty around our predictions that Bayesian modeling allows, provides us with additional valuable information: one may prefer a seemingly less profitable regional client if it yields less uncertainty. Or perhaps we may want to experiment more with the French clients to gather more data and reduce this uncertainty. There are many more valuable analytical questions that this framework can help us answer. For example, we may learn about multiple informative peaks of the posterior distribution or reject an intuitive, yet incorrect assumption incorporated into the priors.

Finally, we can see by how much our model’s predictive strength has improved now that we added this hierarchal structure. A few calculations similar to those we did before (but using different betas for each country) show an accuracy of 93.8% and a weighted f1-score of 94%. Therefore, allowing our model to capture such hierarchal relationships most certainly improved its overall forecasting performance.

**To sum it up**

As we’ve seen so far, Bayesian modeling is not just some fancy confusing concept that statisticians use to scare freshmen. By contrast, it is a flexible and powerful framework that can provide substantial advantages over more ‘traditional’ approaches. With it, we can add more transparency, direct domain knowledge and/or educated assumptions right into the model definition. And even if we were somewhat wrong, real-world data will adjust all this via Bayes’ theorem, leaving us with a more realistic understanding of the problem at hand, and a usable forecasting framework.

The use case, that we have looked at, has taught us several important practical lessons in its turn. There were several factors that made constructing a Bayesian model (using PyMC3) successful. First, we had to make several decisions regarding the amount of complexity that we want our model to address (such as combining cases/countries together, treating sales as binary events etc). Second, a careful selection of prior parameter values was necessary (that incorporates our believes and domain knowledge and does not unnecessarily restrict the model). Then, we had to combine it all in an appropriate likelihood function with the real data supplied to it. Finally, we acquired posterior samples using MCMC with a cautious selection of settings (that can if needed iterate over a few rounds depending on PyMC3’s feedback). And voila – an accurate and informative Bayesian model is at our service.

Carefully following these steps, as well as gaining more practical experience with the Bayesian framework, will make it a powerful and convenient tool, that would often beat competing approaches or help us when no alternatives are even available.