**Doing business, the Bayesian way**

**Part 1: from a business problem to a Bayesian solution**

Bayesian modeling is a lesser known yet very promising area of Statistics and Data Science. It thrives where traditional approaches fail and comes with a few nice perks. In this series of blog posts we consider an illustrative business case about price optimization showing why Bayesian modeling and the related tech (such as the PyMC3 library for Python) is a useful set of tools to have in your data arsenal. We will learn how Bayesian modeling (unlike traditional approaches) allows us to skip determining arbitrary assumptions and thresholds, while transparently getting back not just point estimates but whole distributions of possible values. The first part of this series will introduce you to an illustrative business case and the core Bayesian modeling concepts, translating the former into the language of the latter. The second part will focus on actually constructing Bayesian models (with PyMC3), making inferences and unleashing the full potential of the field. All data as well as all further analysis, graphs and models are available in [this GitHub repo](https://github.com/VadimNelidov/bayesian_modeling_blog).

In principle, no prior knowledge of (Bayesian) Statistics is necessary, but having some would naturally simplify the read (see for example [this series](https://www.researchgate.net/publication/236907708_Back_to_basics_An_introduction_to_statistics) for some helpful prep material).

**The business case**. Consider a (fictional) B2B service provider company “Virtuoso” which operates in France, Germany, and Italy. Virtuoso can charge different prices in each country but does not know how to do so optimally. Their data consists of prices that have been offered historically in each country and led to a sale in some cases. Our goal will be to optimize their pricing strategy the Bayesian way. This will help Virtuoso not only to improve sales but to determine how distinct, price-sensitive, and uncertain each country’s market is.

Table

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

**Understanding the problem**. Before we continue, it is important to identify some potential issues in Virtuoso’s past operations and pricing strategy. This will aid us in formulating our data-driven solutions. From the data, we quickly identify that successful sales are not equally likely across all countries. Besides, for purchases made in each country sales prices also differ. Figure 1 illustrates these two observations for each country (using 95% confidence intervals to acknowledge data variability and potential data scarcity):

Chart, box and whisker chart

Description automatically generated

Figure : Two major observations from the data

Why are more offers rejected in Germany than in Italy? The average purchasing price in Germany is lower, so customers might have tighter budgets and be more price-sensitive. Therefore, using the same pricing strategy in all countries might not be a good idea. Currently, proportional to the number of offers, revenues in Italy are more than twice as high as in Germany proportional to the number of offers. This clearly leaves some room for improvement. If we properly quantify price-sensitivity in each country, we could make such advances.

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 between-country differences better and to develop more personalized sales strategies. Using the Bayesian approach we can not only make usual forecasts (that will help with suggesting better prices in each country), but also incorporate any insider perceptions about each country into the model (e.g. German budgets are more stringent). If these turn out somewhat wrong, the data will correct us. So, let’s get to it.

**Theory in a nutshell**. In essence, Bayesian modeling uses a form of [Bayes Theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem) to determine how the problem at hand really looks like given the observed data and our insider knowledge/believes. We may only have a rough idea about how fair a given coin is or how stringent German companies are, but with a bit of relevant data we can understand these problems much better. And Bayesian modeling tells us how exactly this new data should be considered in combination with our rough beliefs.

This approach presents several advantages: unlike traditional (frequentist) modeling approaches, there is less need to make questionable assumptions beforehand or to set and misuse arbitrary thresholds ([p-values](https://en.wikipedia.org/wiki/Misuse_of_p-values) and such). Instead, we explicitly incorporate what we know about the problem beforehand (our priors). This is more intuitive, transparent and leaves less room for misinterpretation.

Diagram

Description automatically generated

Figure 2: Frequentist vs Bayesian approach

The rest is taken care of by Bayes Theorem. As a result, we acquire not just point estimates but 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 (see Figure 2). This can in its turn translate into valuable business insights such as not just average predicted values, but a range of such values coupled with associated levels of certainty.

A picture containing diagram

Description automatically generated

Figure 3: Bayes formula for posteriors in the language of modeling

**Engineering a Bayesian model**. While helping Virtuoso, our goal is to apply Bayesian modeling such that we learn from the data about the relationship between prices and sales in each country. In Bayesian terms, we are after the posterior (post-learning) distributions of price sensitivity parameters (left part of Figure 3). As the Bayes' formula for modeling in Figure 3 suggests, we need to determine three important parts.

1. is the “likelihood distribution” – the statistical relationship that indicates how *likely* the data is, given our model. This component usually follows from the problem itself: many real life processes are binary events like coin flips, which is commonly modelled using a Bernoulli likelihood.

2. is the “prior distribution” – another statistical relationship that acknowledges all initial information and believes that we have about the problem. If we have no such information, we would deal with “uninformed priors”, indicating that everything is possible. If we have some domain knowledge, expectations or just pure intuition, this framework allows us to directly incorporate it here.

3. is the most technical and least intuitive parameter. The good news is that it is basically a data-dependent constant value that can be ignored during practical modeling.

Bayes theorem tells us how to combine these three factors in a single equation that results in the posterior distribution. As a result, the observed data reshapes our prior distribution into a more *informed* one. In simpler terms, our initial understanding of what the problem entails mathematically evolves into a more informed one. Understanding this means understanding Bayesian modeling:

Diagram

Description automatically generated

Figure 4: Bayesian approach in essence

**Translating** **the business case**

To conclude the first part of this series, let us translate the Virtuoso’s business case in the language of Bayes modeling. This will be essential to later construct and use actual models.

First, what are we actually after in this business case? We want to be better informed about purchasing behaviors across countries and differences in how price-sensitive clients are. Therefore, we are primarily after price sensitivity distributions (which may be different in each country). In Bayesian terms, we are attempting to acquire posteriors for price-sensitivity parameters – whole distributions, not just point estimates such as means. Once we have those, we will be able to compare international clients not just *on average*, but also in terms of risks and variation.

How will we acquire these posteriors? We are going to use Bayes formula together with the three core components introduced above. *Priors* allows us to put any already available knowledge directly into the problem. For instance, as prices grow, customers are usually less willing to purchase. Consequently, our prior can capture that by restricting price sensitivities to only be of negative sign. Other than that, we may have little knowledge of what these parameters can possibly be. In such cases, an *uninformed* prior is commonly chosen to allow a broad range of possibilities.

*Likelihood distribution* concerns the process that turns our inputs (different prices) into some result (purchase or no purchase). This is a binary event, which in statistics is commonly modeled using [*Bernoulli* distribution](https://en.wikipedia.org/wiki/Bernoulli_distribution). There are [many resources](https://en.wikipedia.org/wiki/List_of_probability_distributions) that help matching various real-life processes with appropriate statistical distributions.

Finally, using *data* in this framework comes naturally once everything else is defined. We already have data that matches price suggestions with the 0/1 result of purchase or no purchase. Such data is supplied to a Bayesian model that combines all its components to produce posterior distribution(s). However, actually doing this will have to wait until the next part of this series.

**Short retrospective**

So far in this first part of our series about *doing business the Bayesian way*, we identified a core analytical question from a realistic business case. Then we saw how it can be translated into the language of Bayesian modeling, while also getting introduced to some of the core concepts in this field. We can summarize our translation with the following illustration:

Diagram

Description automatically generated

Figure 5: Summary of problem translation

While we will hold on with constructing a model and analyzing the results until the next part, we can already appreciate some features of Bayesian modeling. This includes being able to include our prior knowledge directly into the problem, while allowing a possibility for it to be corrected if we are wrong. Every step is transparently modeled – which leaves less room for misinterpretation and hidden issues. Furthermore, thinking about the business problem in such structural terms helps to understand its essence. And in the end, we can anticipate not just a point estimate but broad information about the question at hand (involved risks, amount of uncertainty, variability etc.), which will be of higher value to the business.

**Part 2: from a Bayesian solution to business insights**

In Part 1 of this series, we familiarized ourselves with some of the core principals of Bayesian modeling. We also got introduced to an illustrative business problem: helping B2B service provider Virtuoso better understand its regional customers as well as their price-sensitivity and potentially find out better pricing strategies for each market. If you have not read Part 1, we highly encourage you to do so, as here we will continue with building a Bayesian framework that has been defined previously. This part will be more technical, so having familiarized yourself with (Bayesian) Statistics at least on the conceptual level would make it more easily accessible. All data as well as all further analysis, graphs and models are available in [this GitHub repo](https://github.com/VadimNelidov/bayesian_modeling_blog)sitory.

**Model building with PyMC3**. As discussed in Part 1, our goal is to acquire posterior distributions for price-sensitivity parameters in each regional market. While doing so, we will rely on mostly uninformed price-sensitivity priors. Using our domain knowledge, we will restrict those priors to always be negative, as purchase probability should decrease as price increases. Finally, we will use Bernoulli likelihood function to match input date (prices) to the outcome (purchase or no purchase).

But how to combine all this into a usable model and Python code? To achieve that, we use Python’s PyMC3 package. It appears 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 previous intuition into a ready-to-optimize Bayesian framework:

Text

Description automatically generated

**Understanding PyMC3**. Here we deal with a model definition inside a context manager. We define a generic PyMC3 model and name it “pooled\_model” (since to begin with, we consider all countries combined). 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 choose a common logistic probability function, that translates our previous intuition into a simple equation: as the price increases, the probability falls proportional to the *b1* coefficient (this is the price-sensitivity that we are most interested in), while *b0* affects the maximum prices which can possibly be accepted. Then their combination translates 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* are. No need to make any difficult assumptions! We only incorporate some broad intuition for the parameters: *b0* can be anything coming from a generic normal distribution with mean 0 and std of 100. So, it’s most likely around zero but can also be something further away from it. Secondly, *b1* is positive (higher prices decrease offer acceptance chances) and 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 (remember how Bayesian updating works from Part 1?). This makes Bayesian modeling not only very flexible, but also very transparent and a lower-risk approach. Besides, if we have domain knowledge, 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 definitions.

**Pooled Bayesian model**. All things combined; this *pooled* model tells us how offer acceptance likelihood is related to various price levels in all countries *combined*. All we need to do is apply one of PyMC3’s (sampling) algorithms that take care of applying the Bayes formula to our problem and producing the posterior distribution. To make this part more accessible to everyone, we will omit details on how these algorithms work exactly, but you are encouraged to separately read about the famous [Markov chain Monte Carlo](https://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo) (MCMC) algorithm. What we need to know is that PyMC3 will automatically generate a distribution that is proven to converge to the real posterior distribution without any biases provided enough sampling. Conveniently, if something goes wrong like if there was not enough sampling, PyMC3 gives feedback and suggestions on how to improve your script. In terms of code, we acquire the samples for this generated distribution using the last line in the previous code sequence, and here is the result:

Icon

Description automatically generated

Figure : Posterior distributions of parameters from the pooled model

Based on the supplied data, our prior distributions of *b0* and *b1* have been substantially updated. The two distributions now indicate what the parameters are most likely given both your priors and the observed real data. Now with high confidence in all countries combined *b1* (price-sensitivity) is likely between 1 and 2, while *b0* is between 6 and 11. This allows us to perform some inference and already implies a valuable result: Virtuoso’s clients seem very unlikely to accept prices above 13, while prices under 10 would have at least a 50% chance of being accepted. Unlike other modeling approaches, Bayesian modeling produced whole distributions of acceptance likelihoods for various prices that would allow us to directly take uncertainty into account when making difficult decisions!

We can also directly use this model for traditional forecasting and estimate its accuracy (and other metrics). We have previously split the full dataset into a 2-to-1 train-test split, and only training data has been supplied to PyMC3. Now we can directly access all available posterior values from this model (using ‘posterior’ variable) to further make predictions for various prices. While PyMC3 has built-in methods to do so, it is 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 6, 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 implies 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 standard 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. use some quantile value like the median rather than the mean).

**A hierarchical framework**. 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 ‘hierarchical’ – we assume that there is a certain hierarchy and common logic in how people react to our prices in different countries. These reactions are probably similar yet somewhat different. We can incorporate this idea into our Bayesian model by assuming that our three countries have different *b0* and *b1*, that are related by being generated by common distributions. It may sound tricky, but consider the following simplified example:

Diagram

Description automatically generated

Figure 7: Example of a hierarchical framework

In this example there are only Germany and Italy. Both have different likelihoods of purchasing at each price, as illustrated by the logistic curves in the rightmost plot. We already saw that we need two parameters for each of these curves: *b0* and *b1*. Here they are different for each country, but they come from common distributions (on the left). It is much like saying that there are different price-sensitivities in countries worldwide, distributed like shown in the bottom-left plot. Germany and Italy may have different parameters, but those parameters share common origin.

You may wonder how we could know these common worldwide distributions. Actually, we do not. But we can also determine what they could look like while solving the Bayesian problem as a whole! By having data from multiple countries, we can learn about the problem as a whole. This can later also inform us about the countries we have not even seen before or where we have very little data.

So, by adopting this hierarchical framework, we do not have to assume that Italy, Germany, and France are the same, but we also do not assume that they have nothing in common. As a result, such models 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 takes this form:

**Text

Description automatically generated**

Figure : Second (hierarchical) model in PyMC3

This model may look more complicated, but in fact it is built around an idea very similar to the previous *pooled* model. 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* still comes from the same type of prior distributions as before but now not fixed ones but ‘flexible’ distributions with unknown mu and std. These mu and std all have priors of their own as previously illustrated in Figure 7. Those new additional priors are very generic (uninformed) as we do not have a good idea how mu and std are distributed worldwide. A normal distribution around zero with a large std is a common choice in this case. Finally, we need to assign an index to each country and insert 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. When it succeeds, we can inspect the unique parameter distributions generated for each country:

Chart

Description automatically generatedA picture containing shape

Description automatically generated

Figure 8: Resulting parameter distributions in the hierarchical model

While all *b0* seem very similar, there is some more noticeable distinction in *b1* parameters. This makes practical sense – *b1* indicates how sensitive clients in each country are to price increases, and the posterior with the highest mean is the one for Germany – where Virtuoso had by far the lowest fraction of accepted offers. By contrast, Italy, where sales were the highest, appears to be least sensitive to price increases. Furthermore, the width of each distribution is also informative. Broader width indicates more uncertainty – we may want to experiment a bit more in Germany to get a better idea of a good price strategy there. This is a rather unique advantage of the hierarchical Bayesian framework – together with the forecasting model we gain diverse analytical information about the problem we are dealing with.

Now that we know price sensitivity in each country, we can compare various pricing strategies. Let’s say Virtuoso wants to set a fixed price of 7 for all 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 price acceptance of clients in each country:

Chart, line chart

Description automatically generated

Figure 9: Price acceptance in each country

Interestingly, while German clients have their price reduced to 5, they are still the least likely to purchase our services (with a mean of ~20%). Perhaps a higher discount was necessary? By contrast, Italian clients are more than 80% likely to accept their price of 7.

Not only mean values 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 (big advantage of Bayesian modeling), provides us with additional valuable information: one may prefer a less profitable regional client if it yields less uncertainty. Or one can experiment more with the French clients to gather more data and reduce this uncertainty. These are just a few valuable analytical insights that this framework can provide!

Finally, we can see by how much our model’s predictive strength has improved now that we added a hierarchical structure. A few calculations like those we did before show an accuracy of 93.8% and a weighted f1-score of 94%. Therefore, this new structure most certainly improved the model’s 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. It is a flexible and powerful framework that provides substantial advantages over more ‘traditional’ approaches. With it, we can add more transparency, direct domain knowledge and/or educated assumptions right into the model. And even if we were somewhat wrong, real-world data adjusts all this via Bayes’ theorem, leaving us with a more realistic understanding of the problem at hand, and a usable forecasting framework. Furthermore, it quantified uncertainty and provided valuable analytical insights on top of this all.

This use case 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 PyMC3’s automated MCMC sampling algorithm. 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.