German Punctuality and the Deutsche Bahn

By: Maxim Rebguns

Figure 1: A white round German train exiting a tunnel surrounded by greenery with overhead electric wires and poles visible

Image by Sebastian Terfloth via Wikimedia Commons

I stumbled upon a Kaggle dataset titled Trains and Delays Deutsche Bahn and thought it would be interesting, considering the stereotype of Germans being punctual and having an efficient

railroad system. Plus, I like trains. Given the context of the data, my goal of this project is to use the Beta-Binomial bayesian model fed with my own assumptions and a likelihood based on the data set to analyze the current state of train delays on German railways.

Context

The German train system is controlled by the state-owned enterprise Deutsche Bahn. Trains are an integral part of the German public transport system, and it is known as one of the most modern systems in the world. Trains have clear schedules, and are considered on time if they arrive within the scheduled time. Otherwise, they're considered late. A lot of data about train paths is available online, allowing us to conduct analyses.

Assumptions

Ideally, I would like to understand the prevalence of delays in the Deutsche Bahn. My project is based around a yes-or-no observation: either a train arrives on time or it doesn't. Additionally, I make the assumption that train delays aren't correlated with each other (this is a simplification, since the train network is complex and interconnected, but this relationship is not easily quantifiable). At any time, we can collect data and see how many trains had delays (the Duetsche Bahn makes this information public).

My project satisfies the definition of a probability. Whether a train has a delay or doesn't is mutually exclusive. A train cannot have been on time and delayed at the same time. Thus, the probability will be somewhere between 0 and 1.

I will ignore all outcomes but delays. In this analysis, I will choose to ignore train station, train model, route, and other factors. This will limit the scope of my analysis.

The Prior Probability

Because I don't have a prior dataset, I will need to use my expertise to create a probability distribution for $p(\theta)$. Based on what I know about German culture and efficiency, I would estimate roughly 80% would arrive without delays. I found that $\alpha = 9$ and $\beta = 4$ gave me a good distribution and mean (0.82), but multiplied both by 2 to make the distribution more narrow to emphasize my confidence. Essentially, the prior probability represents my prior assumptions about the prevalence of train delays.

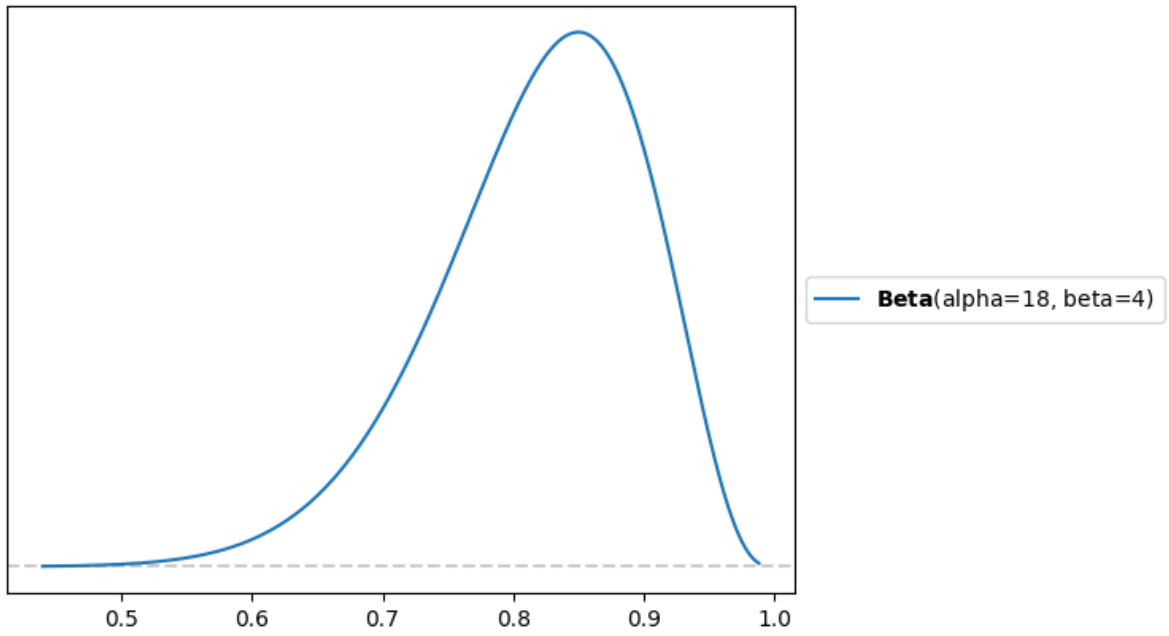
```
import preliz as pz
```

```
WARNING (pytensor.tensor.blas): Using NumPy C-API based implementation for BLAS functions.
```

```
prior_  = 18
prior_  = 4

p_  = pz.Beta(alpha=prior_ , beta=prior_ )
p_ .mean()
```

```
0.8181818181818182
```

```
p_ .plot_pdf()
```



The Data

The dataset I found on Kaggle is titled Trains and Delays Deutsche Bahn. It looks at trains covering Germany's 20 largest cities. Each record in the data is a train arriving at a certain station and all associated charactersitics. These include expected time, train model, route, platform, and delay. First, I have to read in the data and turn it into a list of yeses and nos. Luckily, there is a column called `has_delay` which indicates whether a delay occured.

```
import pandas as pd
```

```
df = pd.read_csv("trains.csv")
df
```

	date	Hbf	scheduled_time	expected_time	train_model	route
0	2024-09-01	Köln Hbf	13:00	13:00	RE 8(10819)	Rommerskirch
1	2024-09-01	Hannover Hbf	13:00	13:00	S 2	Nienburg(Wes
2	2024-09-01	Münster(Westf)Hbf	13:00	13:00	RB 89(90018)	Paderborn Hbf
3	2024-09-01	Hamburg Hbf	13:01	13:01	S 5	Hamburg Elbg
4	2024-09-01	Frankfurt(Main)Hbf	13:01	13:01	S 4	Kronberg(Tau
...	...	...	...	...	...	...
77141	2024-07-23	Bremen Hbf	09:39	09:40	RE 8(4408)	Hannover Hbf,
77142	2024-07-25	Bremen Hbf	09:39	09:41	RE 8(4408)	Hannover Hbf,
77143	2024-07-21	Bremen Hbf	09:39	09:42	RE 8(4408)	Hannover Hbf,
77144	2024-07-24	Bremen Hbf	09:39	09:39	RE 8(4408)	Hannover Hbf,
77145	2024-07-24	Bremen Hbf	09:39	09:39	RE 8(4408)	Hannover Hbf,

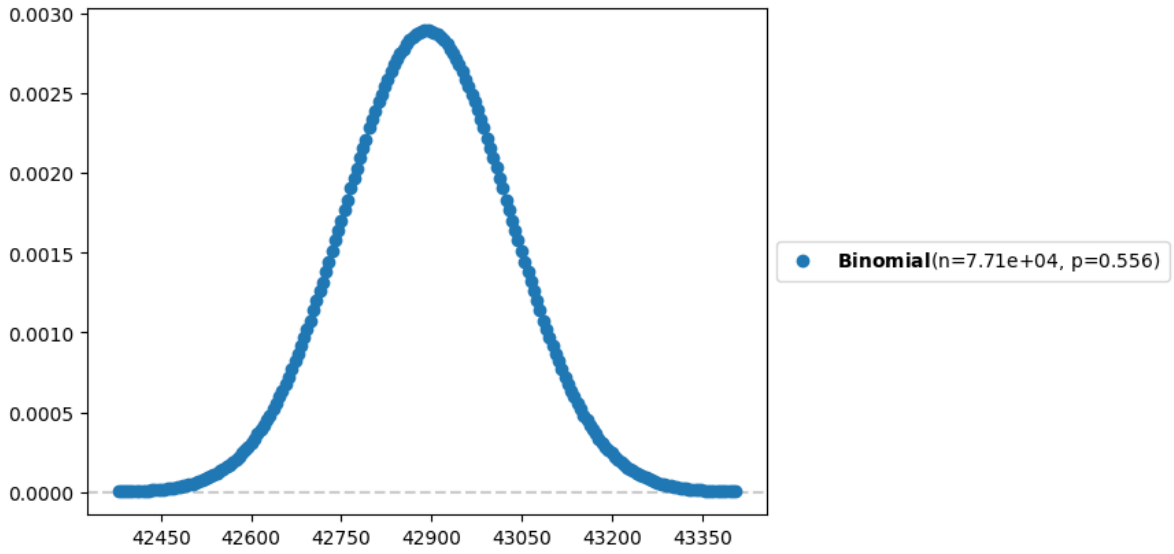
```
num_on_time = len(df[df.has_delay == 0])
num_delayed = len(df[df.has_delay == 1])
total = num_on_time + num_delayed

print(f"On time: {num_on_time}, Delayed: {num_delayed}")
```

```
On time: 42892, Delayed: 34254
```

The Likelihood

So we know that, 42,892 trains came on time and 34,254 were delayed. We can now proceeed to create a likelihood ($p(y|\theta)$) based on the data using a binomial distribution.

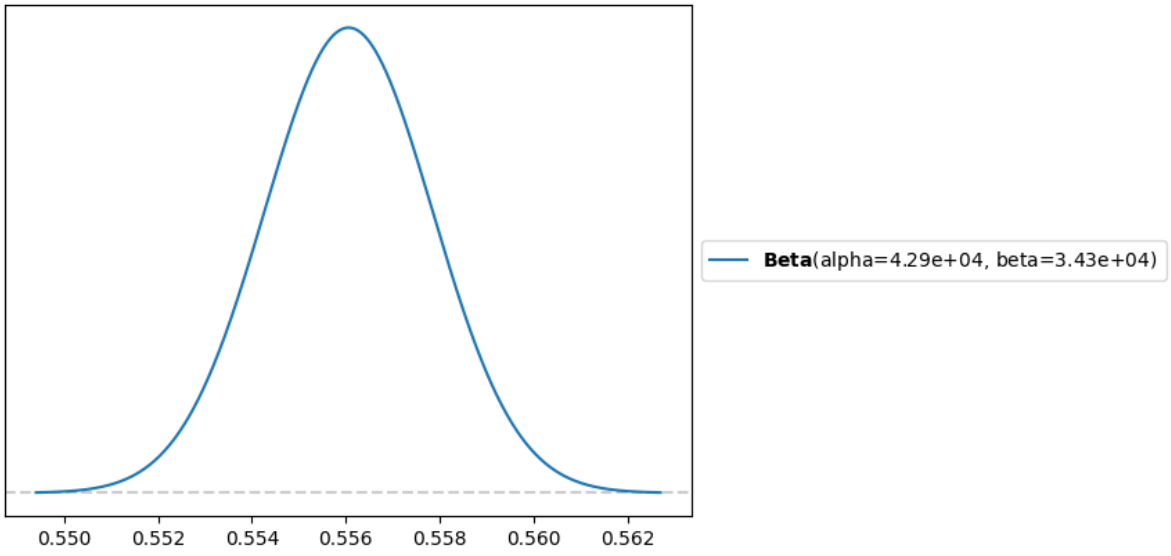
```
p_y_  = pz.Binomial(n = total, p = num_on_time / total)
p_y_ .plot_pdf()
```



The Posterior

Given the prior and likelihood, the posterior can now be created analytically using a Beta distribution.

```
analytic_posterior = pz.Beta(alpha = prior_  + num_on_time, beta = prior_  + total - num_on_t
analytic_posterior.plot_pdf()
```



Compared to the size of the data, my prior is extremely weak. The dataset is magnitudes larger than my prior, so the resulting posterior was basically unaffected by my prior. I decided to leave my prior as-is because my level of confidence is pretty low.

```
analytic_posterior.mean()
```

```
0.5560595065312046
```

This is my best guess for the percentage of trains that arrive on time in Germany using the analytical model. I will my thoughts on this result at the end of this Jupyter notebook when I compare this to the probabalistic result.

Probabilistic Analysis

Another way to come to a result is to use Markov Chain Monte Carlo (MCMC). I am essentially doing the same analysis, but probabalistically (the results change slightly each time this code is run). It is a powerful technique for analyzing data.

```
import pymc as pm
import numpy as np
```

```
data_on_time = np.repeat(1, num_on_time)
data_delayed = np.repeat(0, num_delayed)
data = np.concatenate([data_on_time, data_delayed])
```

```
with pm.Model() as train_delay_model:
    # Prior
      = pm.Beta(" ", alpha=prior_ , beta=prior_ )

    # Likelihood
    y = pm.Bernoulli("y", p= , observed=data)

    sample_data = pm.sample(1000)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: []
```

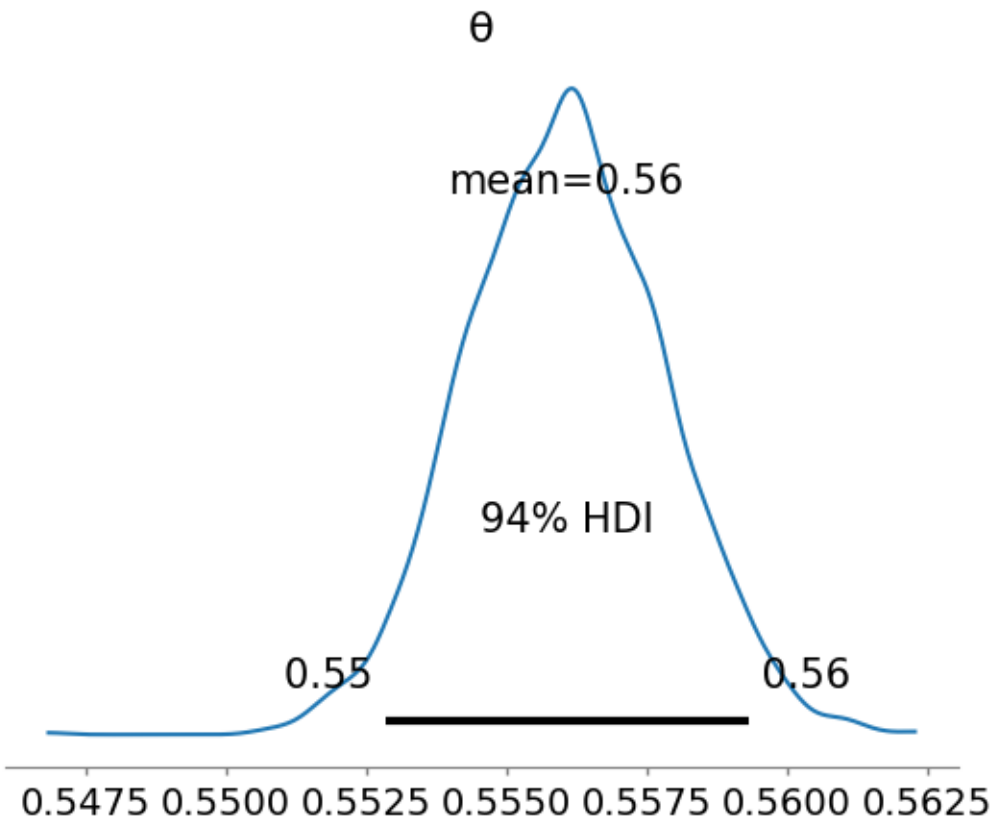
```
Output()
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws total) took 3
```

We can visualize the results:

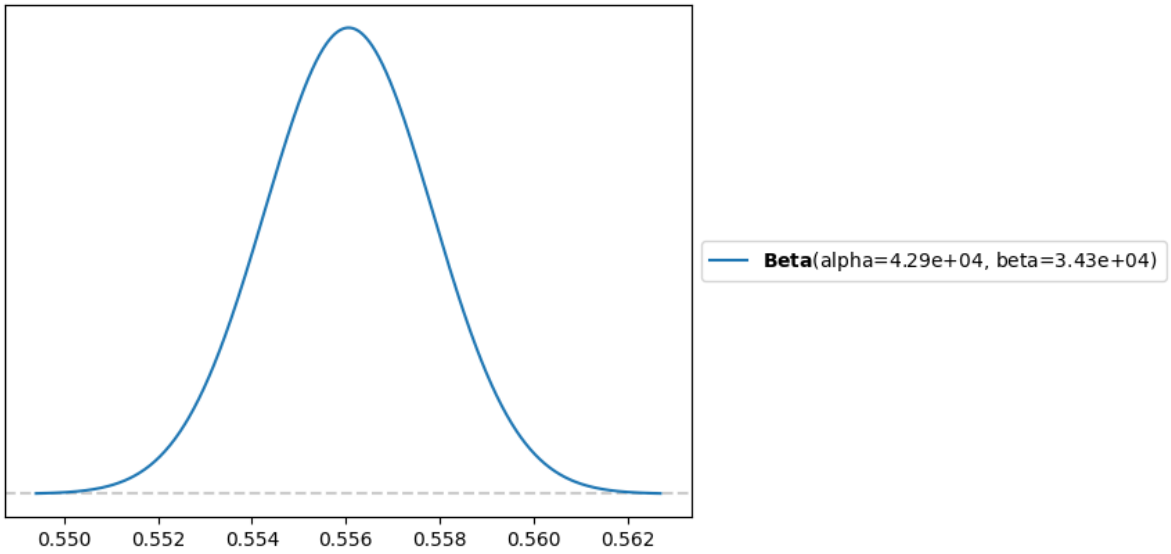
```
import arviz as az
```

```
az.plot_posterior(sample_data)
```



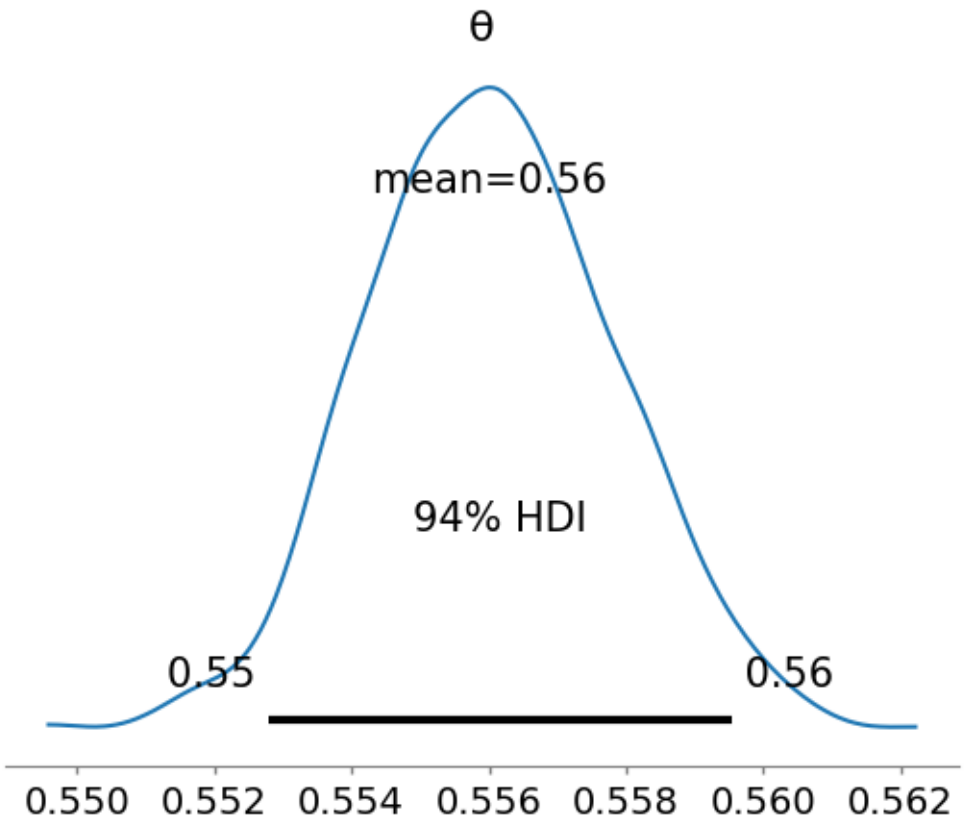
Here is the analytic solution, for reference:

```
analytic_posterior.plot_pdf()
```



We can also draw samples from the analytic model, for reference:

```
az.plot_posterior({" ": analytic_posterior.rvs(1000)})
```



PYMC Predictions

The PYMC model allows us to make predictions about future train delays.

```
pm.sample_posterior_predictive(sample_data, model=train_delay_model, extend_inferencedata=Tru
```

```
Sampling: [y]
```

```
Output()
```

```
Inference data with groups:
    > posterior
    > posterior_predictive
    > sample_stats
    > observed_data
```

```
stacked_samples = az.extract(sample_data, group="posterior_predictive")
stacked_samples.sel(sample=0).sel(draw=0).y.to_series().values
```

```
array([1, 0, 1, ..., 0, 0, 1])
```

Result

Of course, we can also find the mean of the samples:

```
stacked_samples.mean()
```

```
<xarray.Dataset> Size: 8B
Dimensions:  ()
Data variables:
    y        float64 8B 0.556
```

The means I got from both methods gave almost the same result. My prior did little to affect the end result due to the sheer size of the dataset. Overall, the 55% chance of not arriving late suprised me. I believed it would be much higher, but this assumption was refuted by the dataset – the resulting graph is quite narrow.

Sources

- The Beta-Binomial Bayesian Model
- Beta Distribution
- Deutsche Bahn