Fundamental definitions

Frequentist and Bayesian statistics:

The **Frequentist** approach conceptualizes probability as the long-run relative frequency of events in repeated trials. Parameters are considered fixed but unknown quantities. Inference methods, such as estimation and hypothesis testing, rely entirely on observed data without integrating prior beliefs.

Example: Estimating a coin's fairness based solely on the proportion of heads observed over numerous flips, without assuming any initial belief about the coin.

Bayesian statistics redefines probability as a degree of belief about uncertain events. Here, parameters are random variables, and the learning process involves updating prior beliefs with new empirical evidence using Bayes' theorem.

Example: An initial belief that a coin is fair can be revised after a sequence of flips strongly favoring heads, leading to a probabilistic reassessment of fairness.

What is Bayesian Statistics?

Bayesian statistics is a particular approach to applying probability to statistical problems. It provides us with mathematical tools to update our beliefs about random events in light of seeing new data or evidence about those events.
In particular Bayesian inference interprets probability as a measure of believability or confidence that an individual may possess about the occurance of a particular event.

We may have a prior belief about an event, but our beliefs are likely to change when new evidence is brought to light. Bayesian statistics gives us a solid mathematical means of incorporating our prior beliefs, and evidence, to produce new posterior beliefs.

Bayes theorem

gives a mathematical rule for inverting conditional probabilities, allowing one to find the probability of a cause given its effect

$$P(\theta \mid D) = \frac{P(D \mid \theta) \cdot P(\theta)}{P(D)}$$

$P(\theta)$ is the prior distribution of our model parameters, which represents our opinions about the relationship between our outcome and predictor variables before we've seen any data.

$P(D|\theta)$ is the likelihood term, indicating how well the prior fits the observed data.

$P(D)$ is the marginal distribution of the predictor variables. In other words, it represents the probability of having observed the data given all the possible values for θ. When dealing with discrete distributions, P(D) can be obtained by summing over all values of θ; in the continuous case it is obtained by integrating over θ.

The Posterior Distribution

What is the posterior distribution?

It is a reflection of our beliefs about our parameter of interest, having incorporated all of the information we have access to. It is the product of two components: our prior beliefs about theta, and the likelihood, or the evidence, which reflects the information we gained from the observed data. In combining these two pieces, we obtain a probability distribution for our parameter of interest. This is a key piece separating the Bayesian and Frequentist approach. In the frequentist methodology, our answer is expressed in the form of a point estimate (with a confidence interval). In the Bayesian methodology however, our answer is expressed in the form of a probability distribution, allowing us to place a value on the probability of each potential value of p being correct.

Let's start off by analyzing the situation in which we've flipped the coin a single time and observed heads. In the frequentist approach, we saw that, in this scenario, our estimate of p = 1, an obviously incorrect conclusion, and yet we had no other way to answer the question. In the Bayesian world, our answer is very different. Because we defined our prior beliefs to say that the most likely value of theta is 0.5, we are only slightly swayed by that first flip. Think about this intuitively, if you really believed that a coin was fair, and you flipped it once and it came up heads, would that be enough evidence to convince you that the coin isn't fair? Probably not!

The **Bayesian approach** performs better than the Frequentist approach when there is relatively **little data** to work with. The viability of the Frequentist answer relies on the law of large numbers, and thus in the absence of large amounts of data, the results aren't always reliable.

Short Story

The three friends Frequentist Frank, Stubborn Stu, and Bayesian Betty go to a funfair where a mysterious-looking tent catches their eyes. Inside, they meet Claire Voyant, who claims to be a... fortune teller. The friends don't believe her, of course they need proof. So they conduct a little experiment:
The friends take an ordinary deck of cards. Frank shuffles it and draws a card randomly, not showing it to Claire. He asks her to name the color of the card he has just drawn; red or black. The chance of succeeding for a fortune teller should be 100%; for an average person, it is only 50%.
they conduct this experiment for 2 more times and in all cases Claire gives the right answers.

Stu starts with a radical point of view: he has a prior belief that he thinks is the absolute truth in his head. In mathematical terms, this is

$$p(\theta)$$

Stu's prior belief. Note that there is no 'data' term involved. where θ is either ‘fortune teller’ with a probability of 0.1% and ‘not a fortune teller’ with a probability of 99.9%, i.e.

Frequentist:

Data is everything. She got 3 out of 3 right, so I say she clearly has psychic powers.

He has come to this conclusion via a Maximum Likelihood approach, where he maximized the following likelihood formula with respect to

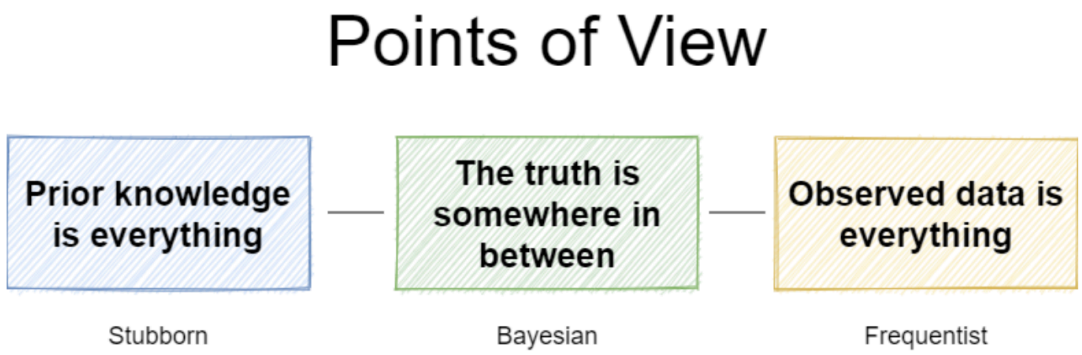
$$P(\text{data} \mid \theta)$$

$$P(3 \text{ right} \mid \text{fortune teller}) = 1$$

$$P(3 \text{ right} \mid \text{not a fortune teller}) = \left(\frac{1}{2}\right)^3 = \frac{1}{8}$$

Frank votes for psychic powers because the probability of observing this kind of outcome (getting 3 colors right) is higher for a fortune teller than an ordinary person. Note that he can’t give a probability that she is not a fortune teller. The maximum likelihood approach does not allow for that.

Bayesian point of view



I start off with a prior belief. Then, I look at the observed data and let each data point change my mind slightly. The more data I observe, the further I can drift off my initial belief. This procedure results in my posterior belief.

The key to expressing this kind of reasoning is – you guessed it – the Bayes Formula, which contains some old friends:

$$P(\theta \mid x) = \frac{P(x \mid \theta) \cdot P(\theta)}{P(x)}$$

Here, you can see that the Bayes approach combines the frequentist's likelihood and the simple prior by multiplication. Because the resulting quantity $p(data|\theta)p(\theta)$ is not a probability (or density) anymore in general, we have to scale it. That is why the $p(data)$ is around in the denominator.

But now we have another problem: What is *p(data)* anyway? How to compute it? The law of total probability says the following:

$$p(\text{data}) = \sum_{\theta} p(\text{data} \mid \theta) \cdot p(\theta)$$

$$\begin{aligned} p(\text{fortune teller} \mid 3 \text{ right}) &= \frac{p(3 \text{ right} \mid \text{fortune teller}) \cdot p(\text{fortu}}{p(3 \text{ right} \mid \text{fortune teller}) \cdot p(\text{fortune teller}) + p(3 \text{ right} \mid \text{not a}} \\ &= \frac{1 \cdot 0.001}{1 \cdot 0.001 + \frac{1}{8} \cdot 0.999} \\ &\approx 0.008 \end{aligned}$$

In the same way, or even simpler, she computes:

$$p(\text{not a fortune teller} \mid 3 \text{ right}) = 1 - p(\text{fortune teller} \mid 3 \text{ right}) = 0.992$$

After a more extended break, she says:

Considering Stu's prior and Frank's likelihoods, I think Claire merely a normal person. If I have to quantify my data-backed belief, I would say there is a 0.8% chance that she has psychic powers.

Note how she has come to the same conclusion as Stu. The probability of Claire being a fortune teller is still low. However, Betty uses existing data in addition to a simple prior belief to boost the confidence in Claire being a fortune teller by eight times compared to Stu's 0.1%. And it makes sense: Even if they have repeated the experiment only 3 times, Claire has undeniably gotten everything right. Intuitively, this has to increase the chance of her having psychic powers.

The friends discuss it again. Is Claire a fortune teller? They all agree with Betty's arguments and conclude:

We don't know, but probably not.

Discrete and Continuous distributions

When working with statistics, we often have to deal with two types of distributions: discrete and continuous. A discrete distribution is used when a variable can only take on certain separate values — for example, the number of heads you get when flipping a coin a few times. A continuous distribution, on the other hand, is used when the variable can take on any value within a range, like someone's exact height or the time it takes them to finish a race.

This difference between discrete and continuous variables shows up in both Frequentist and Bayesian statistics, but the way it matters is a bit different depending on which approach you are using.

Frequentist Perspective

In Frequentist statistics, parameters are assumed to be fixed but unknown — randomness only comes from the data we observe. If the data are discrete, like counts or categories, then naturally a discrete distribution like the Binomial or Poisson would be used to model it. If the data are continuous, we use distributions like the Normal or Exponential.

However, in Frequentist thinking, whether the model is discrete or continuous doesn't change how we think about the parameter itself. It's still fixed; the distribution only changes the methods we use — like whether we apply a binomial test or a t-test.

Bayesian Perspective

In Bayesian statistics, things are different because parameters are treated as random variables. We describe them with probability distributions: we start with a prior belief about a parameter, update it with data, and end up with a posterior distribution.

Bayes' theorem formally states:

$$\text{Posterior} \propto \text{Prior} \times \text{Likelihood}$$

or more precisely:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

When the parameter θ can take only a few values (***discrete case***), calculating the posterior is straightforward. You simply compute the unnormalized posterior for each possible θ, and then normalize:

$$p(\theta_i|x) = \frac{p(x|\theta_i)p(\theta_i)}{\sum_j p(x|\theta_j)p(\theta_j)}$$

where the sum runs over all possible discrete values of θ.

However, when θ is ***continuous***, you can't sum over values — you have to integrate. The marginal likelihood becomes:

$$p(x) = \int p(x|\theta)p(\theta)\, d\theta$$

This integral is often very hard or impossible to compute exactly. Therefore, in practice, Bayesian inference with continuous parameters usually relies on sampling methods like MCMC, which generate a large number of approximate samples $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(N)}$ from the posterior $p(\theta|x)$.

From these samples, you can estimate expectations such as:

$$\mathbb{E}[\theta|x] \approx \frac{1}{N}\sum_{i=1}^{N}\theta^{(i)}$$

and other properties of the posterior distribution.

Why the Difference Matters

Understanding whether a model is discrete or continuous is crucial for both philosophical and practical reasons. In Frequentist statistics, the distinction affects how data is modeled and analyzed but does not change the nature of the parameters themselves. In Bayesian statistics, however, it fundamentally changes how posterior computations are performed: discrete models allow direct and exact solutions, while continuous models typically require sophisticated numerical methods such as MCMC to approximate the posterior.

Subjective and Objective Definitions

In Bayesian statistics, probability can be interpreted differently:

- **Subjective Bayesianism**:
  - Probability represents a personal belief about an event.
  - Different people might assign different probabilities to the same event based on their knowledge or intuition.
  - Example: How likely you think it will rain tomorrow depends on your personal judgment.
- **Objective Bayesianism**:
  - Tries to minimize personal subjectivity by using "non-informative" or "reference" priors.
  - These priors aim to reflect ignorance or neutrality before seeing any data.
  - Example: Using a uniform prior when you have no prior information.

Conjugates

Conjugate priors are special types of priors that make the math of Bayesian updating easier.

- **Definition**: A prior is *conjugate* to a likelihood function if the posterior is in the same family as the prior.
- **Why important?**
  - Easier to compute posterior distributions.
  - Often gives closed-form solutions (no complicated integration).
- **Example**:
  - For a Binomial likelihood (coin toss), a **Beta distribution** prior is conjugate.
  - Posterior will also be a Beta distribution.

Mathematically: $\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$

Methods and Procedures for Both Discrete and Continuous Models

- **Discrete Models**:
  - Models with countable outcomes (e.g., number of heads in coin tosses).
  - Methods:
    - Compute exact posterior distribution using Bayes' rule.
    - Often involve distributions like Binomial, Poisson, and Beta priors.
    - Analytical methods are often possible.
- **Continuous Models**:
  - Models with uncountable outcomes (e.g., measuring someone's height).
  - Methods:
    - Use densities instead of probabilities.
    - Compute posterior using continuous Bayes' theorem.
    - Conjugate priors help; otherwise, numerical methods like MCMC (Markov Chain Monte Carlo) might be needed.

In both cases, the core idea is the same: $\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$

But what if the space is huge, and god forbid additional parameters are involved, like in any real-life modeling scenario?

Now we have to test not only the possible parameter values but also all their possible combinations i.e. the solution space expands exponentially, rendering a grid search computationally infeasible. Luckily, physicists have worked on the problem of efficient sampling, and advanced algorithms exist today (eg. Metropolis-Hastings MCMC, Variational Inference) that are able to quickly explore high dimensional spaces of parameters and find convex points. You don’t have to code these complex algorithms yourself either, probabilistic computing languages like PyMC or STAN make the process highly streamlined and intuitive.

8. Coin Toss Example with the Plots and Python Code

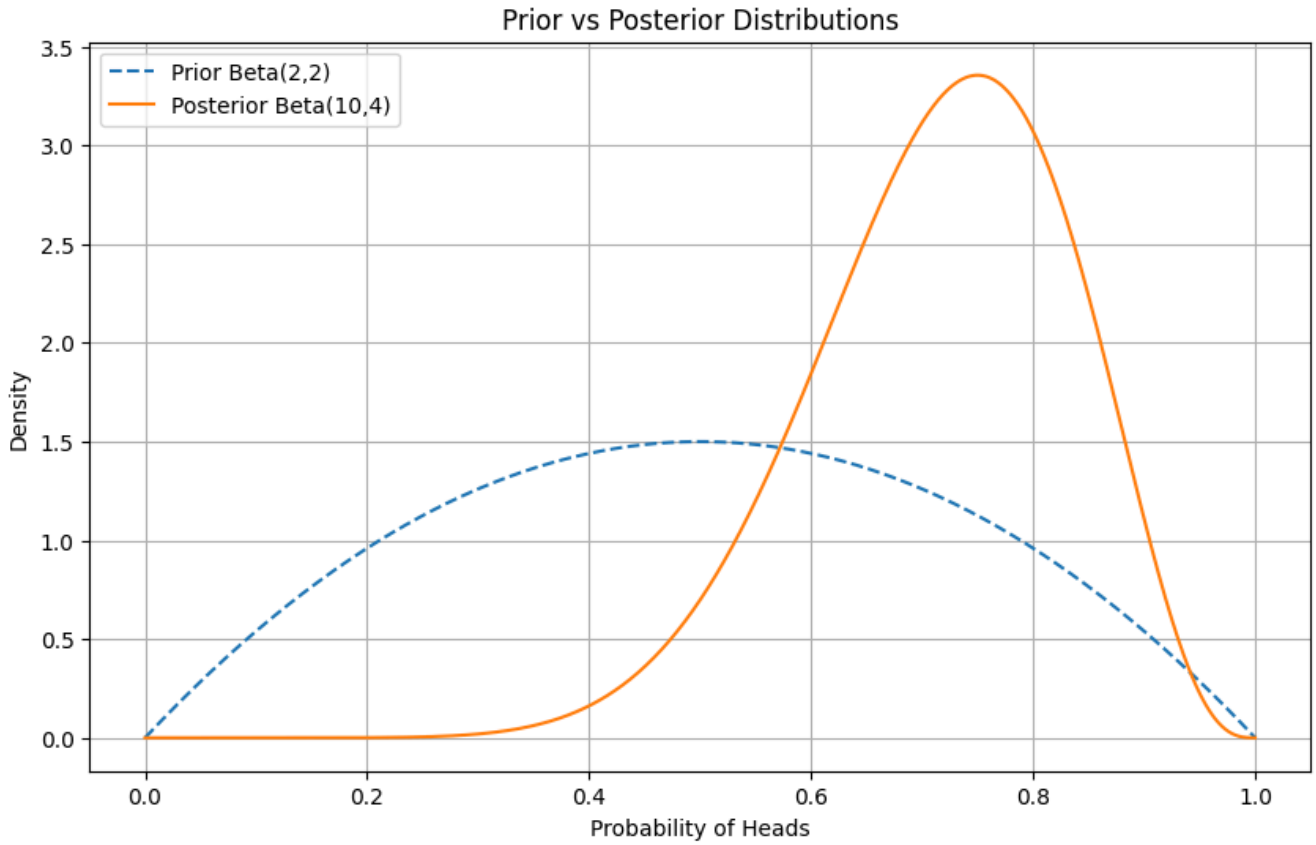
Let's do a simple Bayesian update for a coin toss using a Beta prior.

```
!jupyter nbconvert --ClearMetadataPreprocessor.enabled=True --ClearOutputPrepr
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import yfinance as yf
import pymc as pm
import seaborn as sns
import arviz as az
from scipy import stats
from sklearn.model_selection import train_test_split
from statsmodels.stats.proportion import proportion_confint
from sklearn.linear_model import LinearRegression
```

```
## Parameters
# Prior: Beta(2,2) - a common "neutral" prior
a_prior = 2
b_prior = 2

# Observations: 8 heads, 2 tails
heads = 8
tails = 2

# Posterior: Beta(2 + heads, 2 + tails)
a_post = a_prior + heads
b_post = b_prior + tails

# Plot prior and posterior
x = np.linspace(0, 1, 1000)

plt.figure(figsize=(10,6))
plt.plot(x, stats.beta.pdf(x, a_prior, b_prior), label='Prior Beta(2,2)', line
plt.plot(x, stats.beta.pdf(x, a_post, b_post), label=f'Posterior Beta({a_post}
plt.title('Prior vs Posterior Distributions')
plt.xlabel('Probability of Heads')
plt.ylabel('Density')
plt.legend()
plt.grid()
plt.show()
```



Bayesian and frequentist in practice:

```
#fetch data
data = yf.download('SPY GLD USO SLV',period='1y')
data.head()
data = data.loc[:,'Close']
returns = np.log(data / data.shift(1))
returns = returns.dropna()
```

```
/tmp/ipython-input-3-3902428861.py:2: FutureWarning: YF.download() has changed
  data = yf.download('SPY GLD USO SLV',period='1y')
[*********************100%***********************]  4 of 4 completed
```

```
returns.corr()['GLD'].sort_values()
```

Ticker	GLD
SPY	0.087452
USO	0.271954
SLV	0.673683
GLD	1.000000

dtype: float64

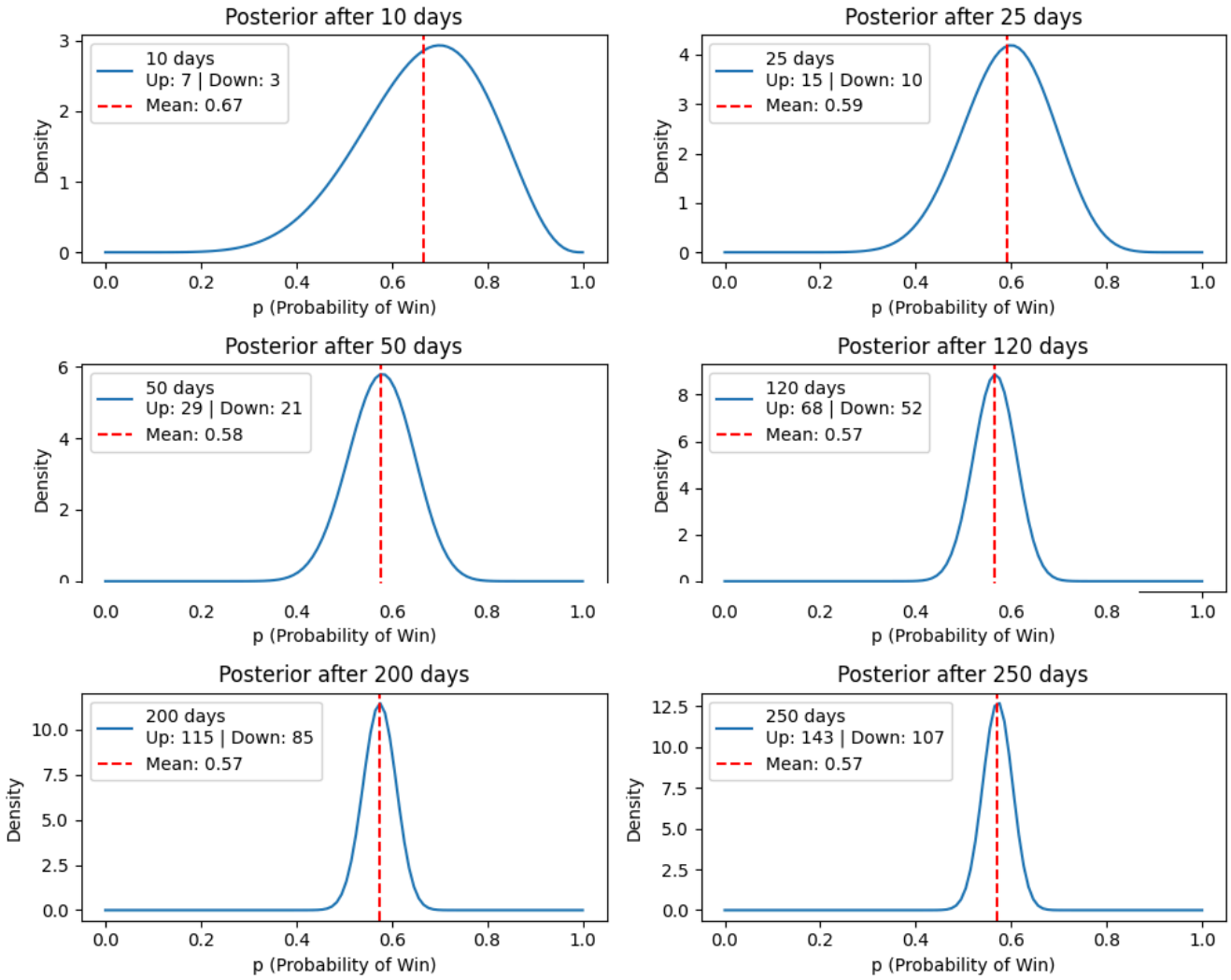
```
returns['Gold Binary'] = (returns['GLD'] > 0).astype(int)
returns
```

Ticker	GLD	SLV	SPY	USO	Gold Binary
Date					
2024-06-25	-0.004975	-0.022473	0.003844	-0.010088	0
2024-06-26	-0.009271	-0.003415	0.001247	-0.001903	0
2024-06-27	0.011273	0.005307	0.001575	0.014748	1
2024-06-28	0.000093	0.004527	-0.003943	-0.004138	1
2024-07-01	0.002601	0.010483	0.002056	0.024205	1
...	...	...	...	...	...
2025-06-17	0.000513	0.021268	-0.008582	0.046127	1
2025-06-18	-0.005400	-0.014028	-0.000151	-0.000365	0
2025-06-20	-0.000419	-0.016670	-0.002351	0.010279	0
2025-06-23	0.003058	0.007308	0.009829	-0.084172	1
2025-06-24	-0.018493	-0.016211	0.009403	-0.051287	0

250 rows × 5 columns

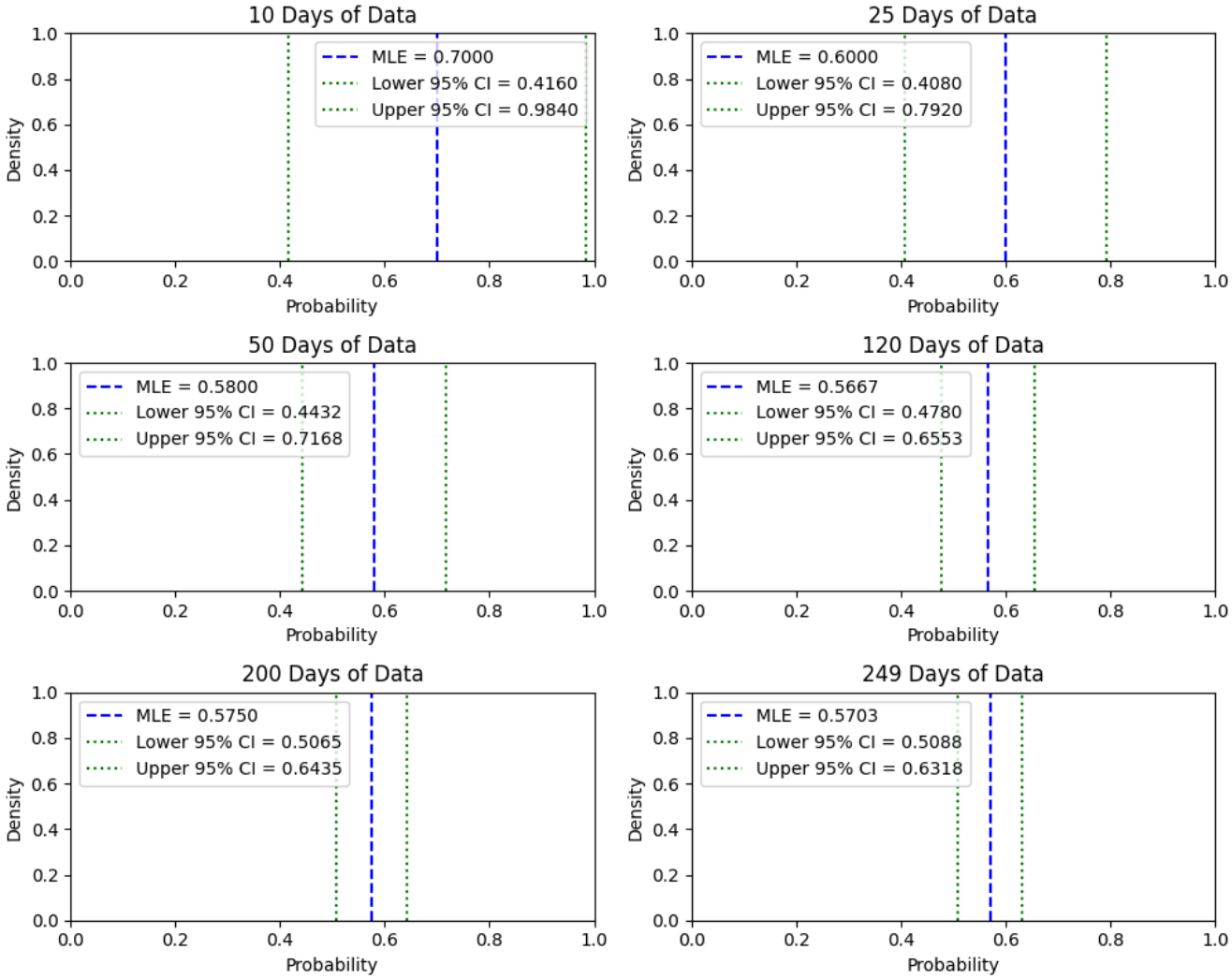
```
n_days = [10, 25, 50, 120,200, len(returns)]
outcomes = returns['Gold Binary'].sample(n_days[-1], random_state=42)
p = np.linspace(0, 1, 100)

# Uniform prior
a = b = 1

# Create 2 rows of subplots
fig, axes = plt.subplots(3, 2, figsize=(10, 8))
axes = axes.flatten()  # Flatten to use single loop

for i, days in enumerate(n_days):
    up = outcomes.iloc[:days].sum()
    down = days - up
    α_post = a + up
    β_post = b + down
    update = stats.beta.pdf(p, α_post, β_post)
```

```
    mean = α_post / (α_post + β_post)

    axes[i].plot(p, update, label=f"{days} days\nUp: {int(up)} | Down: {int(do
    axes[i].axvline(mean, color='red', linestyle='--', label=f"Mean: {mean:.2f
    axes[i].set_title(f"Posterior after {days} days")
    axes[i].set_xlabel("p (Probability of Win)")
    axes[i].set_ylabel("Density")
    axes[i].legend()

plt.tight_layout()
plt.show()
```




```
n_days = [10, 25, 50, 120,200, 249]
alpha =  0.05
total_days = len(returns)
```

```
outcomes = returns['Gold Binary'].sample(n_days[-1], random_state=42)

fig, axes = plt.subplots(3, 2, figsize=(10, 8))
axes = axes.flatten()

for i, days in enumerate(n_days):
    positive_days = outcomes.iloc[:days].sum()
    MLE = positive_days / days
    SD = np.sqrt(MLE * (1 - MLE))
    SE = SD / np.sqrt(days)
    lower, upper = proportion_confint(positive_days, days, alpha=alpha, method

    # Plot vertical lines on subplot
    axes[i].set_xlim(0, 1)
    axes[i].axvline(MLE, color='blue', linestyle='--', label=f'MLE = {MLE:.4f}
    axes[i].axvline(lower, color='green', linestyle=':', label=f'Lower 95% CI
    axes[i].axvline(upper, color='green', linestyle=':', label=f'Upper 95% CI

    # Subplot titles and labels
    axes[i].set_title(f"{days} Days of Data")
    axes[i].set_xlabel('Probability')
    axes[i].set_ylabel('Density')
    axes[i].legend()

plt.tight_layout()
plt.show()
```



implementing a simple machine learning model with bayesian and frequentist approaches

In practice, calculating the exact posterior distribution is computationally intractable for continuous values and so we turn to sampling methods such as Markov Chain Monte Carlo (MCMC) to draw samples from the posterior in order to approximate the posterior. Monte Carlo refers to the general technique of drawing random samples, and Markov Chain means the next sample drawn is based only on the previous sample value. The concept is that as we draw more

samples, the approximation of the posterior will eventually converge on the true posterior distribution for the model parameters.

The best library for probabilistic programming and Bayesian Inference in Python is currently PyMC. It includes numerous utilities for constructing Bayesian Models and using MCMC methods to infer the model parameters.

There are only two steps we need to do to perform Bayesian Linear Regression with this module:

1-Build a formula relating the features to the target and decide on a prior distribution for the data likelihood
2-Sample from the parameter posterior distribution using MCMC

```
    def test_model(trace, test_observation, actual):
        #print('Test Observation:')
        #print(test_observation)

        # See all variable names
        print("Available variables:", list(trace.posterior.data_vars))

        # Extract means
        var_means = {}
        for var in trace.posterior.data_vars:
            mean_val = trace.posterior[var].mean(dim=("chain", "draw")).values
            var_means[var] = mean_val

        intercept_name = [var for var in var_means if "β0" in var or "intercept" i
        theta_name = [var for var in var_means if "θ" in var or "theta" in var.low
        sigma_name = [var for var in var_means if "σ" in var or "sigma" in var.low

        intercept = var_means[intercept_name]
        coefs = var_means[theta_name]
        sigma = var_means[sigma_name]

        test_features = test_observation.values

        mean_loc = intercept + np.dot(test_features, coefs)

        # Generate posterior predictive samples
        estimates = np.random.logistic(loc=mean_loc, scale=sigma, size=1000)


        # Extract the actual value using integer-based indexing
        index = test_observation.name
        actual_value = actual.loc[index]  # Use integer-based indexing for the cor

        # Calculate the residuals (error terms)
        residuals = estimates - actual_value

        # Plotting the posterior predictive distribution
        plt.figure(figsize=(8, 6))
        sns.histplot(estimates, kde=True, bins=20, color="skyblue", edgecolor="k",

        # Mean of the estimates
        plt.axvline(x=mean_loc, color="orange", linestyle="-", label="Mean Estimat
```

```
    # Plot actual value (true grade)
    plt.axvline(x=actual_value, color="red", linestyle="--", label="True Value
    plt.legend()

    # Print summary statistics

    print(f"5% Estimate: {np.percentile(estimates, 5):.4f}")
    print(f"95% Estimate: {np.percentile(estimates, 95):.4f}")
    print(f"Mean Residual: {np.mean(residuals):.4f}")
    print(f"Residual Std Dev: {np.std(residuals):.4f}")


#fit distribution to the returns and select the one with highest likelihood
def fit_dist(selected_stocks, returns):
    marginals_df = pd.DataFrame(index=selected_stocks, columns=['Distribution'
    for stock in selected_stocks:
        data = returns[stock]
        dists = ['Normal', "t-Student", 'Logistic', 'Exponential']
        best_aic = np.inf
        for dist,name in zip([stats.norm, stats.t, stats.genlogistic, stats.ge
            params = dist.fit(data)
            dist_fit = dist(*params)
            log_like = np.log(dist_fit.pdf(data)).sum()
            aic = 2*len(params) - 2 * log_like
            if aic<best_aic:
                best_dist = name
                best_aic = aic
                best_bic = len(params) * np.log(len(data)) - 2 * log_like
                ks_pval = stats.kstest(data, dist_fit.cdf, N=100)[1]
        marginals_df.loc[stock] = [best_dist, best_aic, best_bic, ks_pval]
    return marginals_df
```

```
selected_stocks = returns.columns[:-1].tolist()

fit_dist(selected_stocks, returns)
```

	Distribution	AIC	BIC	KS_pvalue
GLD	Logistic	-1535.539349	-1524.974966	0.995042
SLV	Logistic	-1312.015416	-1301.451033	0.637469
SPY	t-Student	-1571.534507	-1560.970125	0.667593
USO	Logistic	-1230.468419	-1219.904036	0.990035

```
#splitting the data into train and test sets
labels = returns['GLD']
df = returns[['SPY', 'USO', 'SLV']]
X_train, X_test, y_train, y_test = train_test_split(df, labels,
                                                    test_size = 0.20,
                                                    random_state=42)
X_train.head()
```

Ticker	SPY	USO	SLV
Date			
2025-01-02	-0.002460	0.018101	0.022532
2025-05-19	0.001093	0.006003	0.004087
2025-06-06	0.010217	0.020523	0.007369
2024-12-12	-0.005166	-0.001638	-0.029352
2025-01-08	0.001460	-0.011192	0.002558

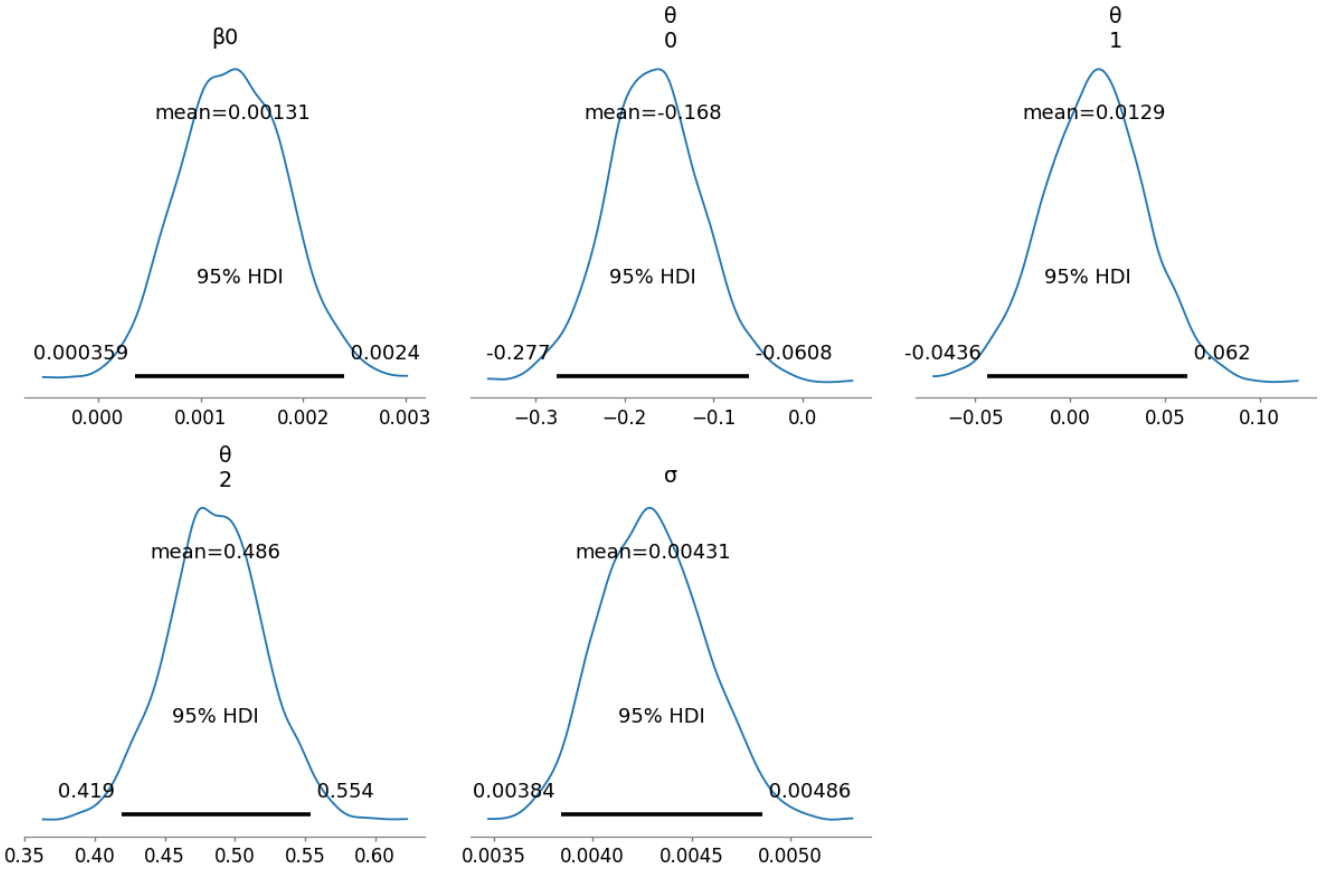
we assume that the relationship between the predictors and the response follows alinear model with a logistic likelihood. We first set priors for the model parameters: a Normal prior for the intercept (β_0), independent Normal priors for the regression coefficients (θ), and a Half-Normal prior for the scale parameter σ to ensure it remains positive. The expected value of the response (μ) is modeled as a linear combination of the predictors and coefficients plus the intercept. Given μ, the likelihood of the observed data (y_obs) is specified using a Logistic distribution, where σ controls the spread of the outcomes around μ. Finally, we perform posterior sampling using Markov Chain Monte Carlo (MCMC), generating 1000 samples after 1000 tuning steps, and store the results as an InferenceData object to facilitate posterior analysis.

```
with pm.Model() as model:
    # Priors for intercept and coefficients
    β0 = pm.Normal("β0", mu=0, sigma=10)
    θ = pm.Normal("θ", mu=0, sigma=10, shape=X_train.shape[1])
    σ = pm.HalfNormal("σ", sigma=1)

    # Expected value of outcome
    mu = β0 + pm.math.dot(X_train, θ)

    # Likelihood
    y_obs = pm.Logistic("y_obs", mu=mu, s=σ, observed=y_train)

    # Sampling
    trace = pm.sample(1000, tune=1000, return_inferencedata=True)
```

Progress	Draws	Divergences	Step size	Grad evals	Sa
————————————	2000	0	0.74	3	39
————————————	2000	0	0.72	3	24

```
# Plot posterior distributions
az.plot_posterior(
    trace,
    figsize=(12, 8),
    kind='kde',
    point_estimate='mean',
    hdi_prob=0.95,
```

```
    textsize=12,
    round_to=3,
)

plt.tight_layout()
plt.show()
```




```
az.summary(trace)
```

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
β0	0.001	0.001	0.000	0.002	0.000	0.000	2884.0	1546.0	1.0
θ[1]	0.013	0.027	-0.041	0.060	0.001	0.001	2333.0	1671.0	1.0
θ[2]	0.486	0.035	0.421	0.550	0.001	0.001	1816.0	1468.0	1.0

```
print(trace.posterior.dims)
```

```
FrozenMappingWarningOnValuesAccess({'chain': 2, 'draw': 1000, 'θ_dim_0': 3})
```

Start coding or generate with AI.

```
test_model(trace, X_test.iloc[34],y_test)
```

```
Available variables: ['β0', 'θ', 'σ']
5% Estimate: -0.0034
95% Estimate: 0.0214
Mean Residual: -0.0084
Residual Std Dev: 0.0078
```