Module 1: Foundations of Bayesian Thinkir for Trading

Course: Bayesian Regression and Time Series Forecasting for Commodities Trading

---

Learning Objectives

By the end of this module, you will be able to:

1. **Understand** the difference between Bayesian and Frequentist approaches in a financial cor
2. **Apply** Bayes' theorem to calculate updated beliefs about market conditions
3. **Interpret** probability as a degree of belief about uncertain market outcomes
4. **Implement** basic Bayesian calculations using Python
5. **Recognize** when Bayesian methods are preferable for trading applications

---

Why This Matters for Trading

Markets are fundamentally uncertain. Traditional statistical methods often assume we have unlin data and that parameters are fixed constants waiting to be discovered. But in commodities tradir

- **Regimes change**: The dynamics of crude oil in 2020 are different from 201
- **Data is limited**: We may only have 50 years of wheat prices, but wheat has been tra millennia
- **Uncertainty matters**: A forecast of "$80/barrel" is useless without knowing how confiden
- **Prior knowledge exists**: Expert traders have beliefs about seasonality, mean reversion, an demand

Bayesian methods let us:

- **Quantify uncertainty** in every prediction
- **Incorporate domain expertise** through prior distributions
- **Update beliefs systematically** as new data arrives
- **Make decisions** that account for model uncertainty

---

1. The Two Philosophies of Probability

1.1 Frequentist View

In the **frequentist** view, probability represents the long-run frequency of events:

- "There's a 60% chance of rain" means "In similar atmospheric conditions, it rains 6 the time"
- Parameters (like true average return) are **fixed but unknown constants**

- Data are random samples from a population
- We use data to estimate the fixed parameters

The problem for trading: We can't repeat the 2008 financial crisis. Each market ever unique. What does "probability of a recession" mean in frequentist terms?

1.2 Bayesian View

In the **Bayesian** view, probability represents our **degree of belief**:

- "There's a 60% chance of rain" means "Given my current information, I believe there's 60% chance of rain"
- Parameters are **random variables** with probability distributions
- We start with **prior beliefs** and update them with data
- The result is a **posterior distribution** reflecting updated beliefs

The advantage for trading: We can talk about "the probability that crude oil will exceed $10 though this specific future hasn't happened yet.

2. Bayes' Theorem: The Foundation

At the heart of Bayesian inference is **Bayes' theorem**:

$$P(\theta | \text{Data}) = \frac{P(\text{Data} | \theta) \cdot P(\theta)}{P(\text{Data})}$$

Or in words:

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$$

Let's break this down:

| Term | Symbol | Trading Interpretation | |------|--------|-----------------------| | **Prior** | $P(\theta)$ | Ou about parameter θ before seeing data (e.g., "Gold tends to rise during uncertainty") | | **Lik | $P(\text{Data} | \theta)$ | How likely is the observed data given our hypothesis? | | **Evidence** | $P(\text{Data})$ | Total probability of the data (normalizing constant) | | **Posterior** | $P(\theta | \te $ | Updated belief after seeing the data |

The key insight: **The posterior combines prior knowledge with observed data.**

In [ ]:
```
# Setup: Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
import warnings
warnings.filterwarnings('ignore')

# Set random seed for reproducibility
np.random.seed(42)

# Plotting style
plt.style.use('seaborn-v0_8-whitegrid')
plt.rcParams['figure.figsize'] = (12, 6)
```

```
plt.rcParams['font.size'] = 12

print("Libraries loaded successfully!")
```

3 . Trading Example: Is This Strategy Profitable?

Let's apply Bayes' theorem to a practical trading question.

Scenario

You've developed a trading strategy and backtested it for 20 days. It was profitable on out of 20 days (70 % win rate).

Question: What's the probability that this strategy has a true win rate above 50 %?

Frequentist Approach

A frequentist would calculate a confidence interval:

- Point estimate: 70 %
- 95 % CI: approximately [46 %, 88 %] (using normal approximation)

But the frequentist **cannot say** "there's an X% probability the true win rate is above 50 %". T only say "if we repeated this experiment many times, 95 % of intervals would contain the tru parameter."

Bayesian Approach

We can directly answer: **What's the probability the true win rate > 50 %?**

In [ ]:
```
def bayesian_win_rate_analysis(wins, total, prior_alpha=1, prior_beta=1):
    """
    Analyze win rate using Bayesian inference with Beta-Binomial model.

    Parameters:
    -----------
    wins : int
        Number of winning trades
    total : int
        Total number of trades
    prior_alpha, prior_beta : float
        Beta distribution parameters for prior
        alpha=1, beta=1 gives uniform prior (no prior belief)

    Returns:
    --------
    dict with posterior statistics
    """
    # Posterior parameters (Beta-Binomial conjugate update)
    posterior_alpha = prior_alpha + wins
    posterior_beta = prior_beta + (total - wins)

    # Create distributions
    prior = stats.beta(prior_alpha, prior_beta)
```

```
    posterior = stats.beta(posterior_alpha, posterior_beta)

    # Calculate key quantities
    results = {
        'posterior_mean': posterior.mean(),
        'posterior_std': posterior.std(),
        'prob_above_50': 1 - posterior.cdf(0.5),  # P(win_rate > 0.5)
        'prob_above_60': 1 - posterior.cdf(0.6),  # P(win_rate > 0.6)
        'hdi_95': (posterior.ppf(0.025), posterior.ppf(0.975)),
        'prior': prior,
        'posterior': posterior
    }

    return results

# Our trading strategy: 14 wins out of 20 trades
wins = 14
total = 20

# Analyze with uniform prior (no prior belief)
results = bayesian_win_rate_analysis(wins, total)

print("=" * 60)
print("BAYESIAN ANALYSIS: Trading Strategy Win Rate")
print("=" * 60)
print(f"\nObserved: {wins} wins out of {total} trades ({100*wins/total:.0
print(f"\nPosterior Analysis:")
print(f"  Expected win rate: {results['posterior_mean']:.1%}")
print(f"  Standard deviation: {results['posterior_std']:.1%}")
print(f"  95% Credible Interval: [{results['hdi_95'][0]:.1%}, {results['h
print(f"\nKey Probabilities:")
print(f"  P(true win rate > 50%) = {results['prob_above_50']:.1%}")
print(f"  P(true win rate > 60%) = {results['prob_above_60']:.1%}")
```

In [ ]:

```
# Visualize Prior and Posterior
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Plot range
x = np.linspace(0, 1, 1000)

# Prior vs Posterior
ax = axes[0]
ax.plot(x, results['prior'].pdf(x), 'orange', linewidth=2, label='Prior (
ax.plot(x, results['posterior'].pdf(x), 'blue', linewidth=2, label='Poste
ax.axvline(0.5, color='red', linestyle='--', label='Break-even (50%)')
ax.fill_between(x[x > 0.5], results['posterior'].pdf(x[x > 0.5]),
                alpha=0.3, color='green', label=f'P(>50%) = {results["pro
ax.set_xlabel('Win Rate', fontsize=12)
ax.set_ylabel('Probability Density', fontsize=12)
ax.set_title('Bayesian Update: Prior → Posterior', fontsize=14, fontweigh
ax.legend()
ax.set_xlim(0, 1)

# How data changes belief
ax = axes[1]
# Show posteriors with different amounts of data
for n_trades, n_wins in [(5, 4), (20, 14), (100, 70), (500, 350)]:
    posterior = stats.beta(1 + n_wins, 1 + n_trades - n_wins)
    ax.plot(x, posterior.pdf(x), linewidth=2,
            label=f'{n_trades} trades ({100*n_wins/n_trades:.0f}% win)')
```

```
ax.axvline(0.7, color='red', linestyle='--', alpha=0.5, label='True rate
ax.set_xlabel('Win Rate', fontsize=12)
ax.set_ylabel('Probability Density', fontsize=12)
ax.set_title('More Data → More Certainty', fontsize=14, fontweight='bold'
ax.legend()
ax.set_xlim(0.3, 1)

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

Key Insight: Uncertainty Decreases with More Data

Notice how the posterior distribution becomes **narrower** (more certain) as we collect more data, peak stays around the true 70% win rate. This is Bayesian learning in action:

- 5 **trades**: Very uncertain - win rate could be anywhere from 30% to 95%
- 20 **trades**: Moderate certainty - likely between 50% and 85%
- 100 **trades**: Pretty confident - likely between 60% and 78%
- 500 **trades**: Highly confident - tightly concentrated around 70%

Trading implication: Don't bet big on a strategy with only 20 trades of history. The uncer still high!

4. The Power of Prior Information

One of Bayesian inference's greatest strengths is incorporating prior knowledge. Let's see how d priors affect our conclusions.

Scenario: Experienced vs Naive Trader

Naive trader: "I have no prior beliefs about my strategy" (Uniform prior)

Experienced trader: "Most trading strategies underperform. I expect a 45% win rate on av and I'm fairly confident about this" (Informative prior centered at 0.45)

In [ ]:
```
# Compare different priors
wins, total = 14, 20  # Same observed data

priors = {
    'Uniform (Naive)': (1, 1),           # No prior belief
    'Skeptical': (9, 11),                # Prior mean = 45%, skeptical of
    'Optimistic': (14, 6),               # Prior mean = 70%, believes in
    'Strong Skeptic': (45, 55)           # Prior mean = 45%, very confide
}

fig, axes = plt.subplots(2, 2, figsize=(14, 10))
x = np.linspace(0, 1, 1000)

for ax, (name, (alpha, beta)) in zip(axes.flatten(), priors.items()):
    # Prior
    prior = stats.beta(alpha, beta)

    # Posterior
    post_alpha = alpha + wins
    post_beta = beta + (total - wins)
```

```
    posterior = stats.beta(post_alpha, post_beta)

    # Plot
    ax.plot(x, prior.pdf(x), 'orange', linewidth=2, label='Prior')
    ax.plot(x, posterior.pdf(x), 'blue', linewidth=2, label='Posterior')
    ax.axvline(0.5, color='red', linestyle='--', alpha=0.5)
    ax.axvline(wins/total, color='green', linestyle=':', label='Observed

    # Annotations
    prob_above_50 = 1 - posterior.cdf(0.5)
    ax.set_title(f'{name}\nP(win rate > 50%) = {prob_above_50:.1%}', font
    ax.set_xlabel('Win Rate')
    ax.set_ylabel('Density')
    ax.legend(loc='upper left')
    ax.set_xlim(0, 1)

plt.tight_layout()
plt.show()

# Summary table
print("\n" + "="*70)
print("SUMMARY: How Priors Affect Conclusions")
print("="*70)
print(f"{'Prior':<20} {'Prior Mean':>12} {'Post. Mean':>12} {'P(>50%)':>1
print("-"*70)
for name, (alpha, beta) in priors.items():
    prior_mean = alpha / (alpha + beta)
    post_alpha = alpha + wins
    post_beta = beta + (total - wins)
    post_mean = post_alpha / (post_alpha + post_beta)
    posterior = stats.beta(post_alpha, post_beta)
    prob_above_50 = 1 - posterior.cdf(0.5)
    print(f"{name:<20} {prior_mean:>12.1%} {post_mean:>12.1%} {prob_above
```

Key Insights on Priors

1. **Priors matter** when data is limited (2 0 trades)
2. **Skeptical priors** protect against overfitting to noise
3. **Strong priors** require more data to overcome
4. **With enough data**, all reasonable priors converge to similar posteriors

Trading Wisdom: In the absence of strong evidence, skepticism is rational. Most trading strateg Bayesian approach naturally encodes this skepticism.

5. Bayesian vs Frequentist: Credible Intervals vs Confidence Intervals

This is one of the most important distinctions in statistics.

Frequentist 9 5 % Confidence Interval

"If we repeated this experiment many times and calculated a 9 5 % CI each time, 9 5 % intervals would contain the true parameter."

What it does NOT mean: "There's a 9 5 % probability the true parameter is in this interval."

Bayesian 95% Credible Interval

"Given our prior and the observed data, there's a 95% probability the true parameter is in th interval."

This IS what traders usually want to know!

In [ ]:
```
# Compare Frequentist CI vs Bayesian Credible Interval
wins, total = 14, 20
observed_rate = wins / total

# Frequentist: Normal approximation CI
se = np.sqrt(observed_rate * (1 - observed_rate) / total)
freq_ci = (observed_rate - 1.96 * se, observed_rate + 1.96 * se)

# Bayesian: Credible interval from posterior
posterior = stats.beta(1 + wins, 1 + total - wins)
bayes_ci = (posterior.ppf(0.025), posterior.ppf(0.975))

print("="*60)
print("CREDIBLE vs CONFIDENCE INTERVALS")
print("="*60)
print(f"\nObserved: {wins} wins out of {total} trades")
print(f"\nFrequentist 95% CI: [{freq_ci[0]:.1%}, {freq_ci[1]:.1%}]")
print(f"  Interpretation: If we repeated this 100 times, ~95 intervals")
print(f"                  would contain the true win rate.")
print(f"\nBayesian 95% Credible Interval: [{bayes_ci[0]:.1%}, {bayes_ci[1
print(f"  Interpretation: There's a 95% probability the true win rate")
print(f"                  is between {bayes_ci[0]:.1%} and {bayes_ci[1]:.

# Visualize
fig, ax = plt.subplots(figsize=(12, 5))
x = np.linspace(0.3, 1, 1000)
ax.plot(x, posterior.pdf(x), 'blue', linewidth=2, label='Posterior Distri
ax.fill_between(x[(x >= bayes_ci[0]) & (x <= bayes_ci[1])],
                posterior.pdf(x[(x >= bayes_ci[0]) & (x <= bayes_ci[1])])
                alpha=0.3, color='blue', label='95% Credible Interval')
ax.axvline(observed_rate, color='red', linestyle='--', linewidth=2, label

# Mark frequentist CI
ax.axvline(freq_ci[0], color='orange', linestyle=':', linewidth=2)
ax.axvline(freq_ci[1], color='orange', linestyle=':', linewidth=2, label=

ax.set_xlabel('Win Rate', fontsize=12)
ax.set_ylabel('Probability Density', fontsize=12)
ax.set_title('Bayesian Credible Interval vs Frequentist Confidence Interv
ax.legend(loc='upper left')
plt.tight_layout()
plt.show()
```

6. Monte Carlo Simulation: Trading Under Uncertainty

One of the most powerful applications of Bayesian thinking is simulating future outcomes that ac parameter uncertainty.

Traditional Approach

"Our strategy has a 70% win rate. With 100 trades, we expect 70 wins."

Bayesian Approach

"Our strategy's win rate is uncertain (posterior distribution). Let's simulate 100 trades th of times, each time drawing a different win rate from our posterior."

This gives us a **distribution of outcomes** that properly accounts for our uncertainty about the tr rate.

In [ ]:
```python
def simulate_trading_outcomes(posterior, n_future_trades, n_simulations,
                              avg_win=100, avg_loss=-80):
    """
    Simulate future trading outcomes accounting for parameter uncertainty

    Parameters:
    -----------
    posterior : scipy.stats distribution
        Posterior distribution of win rate
    n_future_trades : int
        Number of future trades to simulate
    n_simulations : int
        Number of Monte Carlo simulations
    avg_win : float
        Average profit on winning trade
    avg_loss : float
        Average loss on losing trade

    Returns:
    --------
    final_pnl : array of final P&L for each simulation
    """
    final_pnl = np.zeros(n_simulations)

    for i in range(n_simulations):
        # Step 1: Draw a win rate from posterior
        true_win_rate = posterior.rvs()

        # Step 2: Simulate trades with this win rate
        wins = np.random.binomial(n_future_trades, true_win_rate)
        losses = n_future_trades - wins

        # Step 3: Calculate P&L
        pnl = wins * avg_win + losses * avg_loss
        final_pnl[i] = pnl

    return final_pnl

# Our posterior from earlier
wins, total = 14, 20
posterior = stats.beta(1 + wins, 1 + total - wins)

# Simulate next 100 trades
n_future_trades = 100
```

```
n_simulations = 10000

# Bayesian simulation (accounts for uncertainty)
bayesian_pnl = simulate_trading_outcomes(posterior, n_future_trades, n_si

# Naive simulation (assumes 70% is the true rate)
naive_pnl = np.zeros(n_simulations)
for i in range(n_simulations):
    wins = np.random.binomial(n_future_trades, 0.70)
    losses = n_future_trades - wins
    naive_pnl[i] = wins * 100 + losses * (-80)

# Compare results
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Bayesian simulation
ax = axes[0]
ax.hist(bayesian_pnl, bins=50, density=True, alpha=0.7, color='blue', lab
ax.axvline(0, color='red', linestyle='--', linewidth=2, label='Break-even
ax.axvline(np.percentile(bayesian_pnl, 5), color='orange', linestyle=':',
ax.axvline(np.percentile(bayesian_pnl, 95), color='orange', linestyle=':'
           label='90% Range')
ax.set_xlabel('Total P&L ($)', fontsize=12)
ax.set_ylabel('Density', fontsize=12)
ax.set_title('Bayesian: Accounts for Win Rate Uncertainty', fontsize=12,
ax.legend()

# Naive simulation
ax = axes[1]
ax.hist(naive_pnl, bins=50, density=True, alpha=0.7, color='green', label
ax.axvline(0, color='red', linestyle='--', linewidth=2, label='Break-even
ax.axvline(np.percentile(naive_pnl, 5), color='orange', linestyle=':', li
ax.axvline(np.percentile(naive_pnl, 95), color='orange', linestyle=':', l
           label='90% Range')
ax.set_xlabel('Total P&L ($)', fontsize=12)
ax.set_ylabel('Density', fontsize=12)
ax.set_title('Naive: Assumes 70% Win Rate is Certain', fontsize=12, fontw
ax.legend()

plt.tight_layout()
plt.show()

# Print summary
print("\n" + "="*60)
print("MONTE CARLO SIMULATION COMPARISON")
print("="*60)
print(f"\nSimulating {n_future_trades} future trades ({n_simulations:,} s
print(f"\n{'Metric':<25} {'Bayesian':>15} {'Naive':>15}")
print("-"*60)
print(f"{'Expected P&L'::<25} ${np.mean(bayesian_pnl):>14,.0f} ${np.mean(
print(f"{'Std Dev of P&L'::<25} ${np.std(bayesian_pnl):>14,.0f} ${np.std(
print(f"{'5th Percentile (VaR)'::<25} ${np.percentile(bayesian_pnl, 5):>1
print(f"{'P(Loss)'::<25} {np.mean(bayesian_pnl < 0):>14.1%} {np.mean(naiv
print(f"\nBayesian approach shows MORE uncertainty - this is realistic!")
```

Key Insight: Bayesian Simulations Are More Honest About Uncerta

Notice that the Bayesian simulation has:

- **Higher standard deviation**: Because we're uncertain about the true win rate
- **Fatter tails**: More extreme outcomes are possible
- **Higher probability of loss**: The naive approach underestimates risk

The naive approach falsely assumes we **know** the true win rate is 70%. The Bayesian appr acknowledges our uncertainty and produces more realistic risk estimates.

This is why Bayesian methods matter for risk management.

7. Real-World Example: Updating Beliefs About Gold

Let's apply Bayesian thinking to a real commodity trading scenario.

Scenario

You're analyzing whether gold prices tend to rise during periods of high uncertainty (measured b Your prior belief (from reading market research) is that there's a 65% chance of a positive correlation. You want to test this with data.

You collect data for 30 months:

- In 21 months with rising VIX, gold rose in 16 of them (76%)
- This suggests gold does tend to rise during uncertainty

How should you update your beliefs?

In [ ]:
```
# Gold-VIX relationship analysis

# Prior: 65% confident gold rises with VIX, moderate certainty
# This translates to Beta(13, 7) which has mean 0.65
prior_alpha = 13
prior_beta = 7

# Observed data
gold_rose_with_vix = 16
total_high_vix_months = 21

# Bayesian update
post_alpha = prior_alpha + gold_rose_with_vix
post_beta = prior_beta + (total_high_vix_months - gold_rose_with_vix)

prior = stats.beta(prior_alpha, prior_beta)
posterior = stats.beta(post_alpha, post_beta)

# Visualize
fig, ax = plt.subplots(figsize=(12, 6))
x = np.linspace(0, 1, 1000)

ax.plot(x, prior.pdf(x), 'orange', linewidth=2, label=f'Prior (mean={prio
ax.plot(x, posterior.pdf(x), 'blue', linewidth=2, label=f'Posterior (mean
```

```
ax.axvline(0.5, color='red', linestyle='--', alpha=0.5, label='50% (no re
ax.fill_between(x[x > 0.5], posterior.pdf(x[x > 0.5]), alpha=0.3, color='

ax.set_xlabel('P(Gold rises | VIX rises)', fontsize=12)
ax.set_ylabel('Probability Density', fontsize=12)
ax.set_title('Bayesian Analysis: Does Gold Rise When VIX Rises?', fontsiz
ax.legend()
ax.set_xlim(0.3, 1)

plt.tight_layout()
plt.show()

# Print analysis
print("\n" + "="*60)
print("GOLD-VIX RELATIONSHIP: Bayesian Analysis")
print("="*60)
print(f"\nPrior belief: Gold rises with VIX {prior.mean():.0%} of the tim
print(f"Observed: {gold_rose_with_vix}/{total_high_vix_months} months gol
print(f"\nPosterior:")
print(f"  Updated belief: Gold rises with VIX {posterior.mean():.1%} of t
print(f"  95% Credible Interval: [{posterior.ppf(0.025):.1%}, {posterior.
print(f"  P(positive relationship) = {1 - posterior.cdf(0.5):.1%}")
print(f"\nConclusion: Very strong evidence that gold tends to rise during
```

8. Summary: When to Use Bayesian Methods

Bayesian Methods Excel When:

Situation	Why Bayesian Helps
Limited data	Priors regularize estimates and prevent overfitting
Expert knowledge available	Priors encode domain expertise
Uncertainty quantification needed	Full posterior distribution, not just point estimate
Sequential decisions	Natural framework for updating beliefs
Risk management	Proper accounting of parameter uncertainty
Regime changes possible	Can incorporate beliefs about structural breaks

Key Concepts Learned

1. **Probability as belief**: Bayesian probability represents our degree of belief, not just long-run frequencies
2. **Prior → Data → Posterior**: We start with beliefs, update with evidence, get refined beliefs
3. **Credible intervals**: "95% probability the parameter is here" (what we actually want)
4. **Uncertainty propagation**: Monte Carlo simulation with parameter uncertainty gives realistic estimates
5. **Prior sensitivity**: With limited data, priors matter; with lots of data, they wash out

---

Knowledge Check Quiz

Test your understanding with these questions. Answers are in the next cell.

Q 1 : In Bayesian inference, what does the "prior" represent?

- A) The data we've collected
- B) Our beliefs before seeing data
- C) The probability of the data
- D) The final answer

Q 2 : A 95% Bayesian credible interval means:

- A) If we repeated the experiment, 95% of intervals would contain the true value
- B) There's a 95% probability the true value is in this interval
- C) The parameter is exactly in this range
- D) We are 95% confident in our methodology

Q 3 : With more data, Bayesian posteriors typically:

- A) Become wider (more uncertain)
- B) Become narrower (more certain)
- C) Stay the same width
- D) Become bimodal

Q 4 : Why might a skeptical prior be appropriate for evaluating trading strategies?

- A) It makes the math easier
- B) Most trading strategies fail, so skepticism is rational
- C) Skeptical priors always give lower risk estimates
- D) The SEC requires it

Q 5 : In the trading simulation, the Bayesian approach showed higher risk because:

- A) The calculations were wrong
- B) It accounts for uncertainty in the true win rate
- C) Bayesian methods always show more risk
- D) The prior was too pessimistic

In [ ]:
```
# Quiz Answers
print("="*60)
print("QUIZ ANSWERS")
print("="*60)
print("""
Q1: B) Our beliefs before seeing data
    The prior P(θ) represents our beliefs about the parameter before
    observing any data. It encodes domain knowledge.

Q2: B) There's a 95% probability the true value is in this interval
    This is the key difference from frequentist confidence intervals!
    Bayesian credible intervals have a direct probability interpretation.
```

```
Q3: B) Become narrower (more certain)
    As we collect more data, our uncertainty decreases and the posterior
    concentrates around the true value.

Q4: B) Most trading strategies fail, so skepticism is rational
    Given that most strategies underperform after costs, a skeptical prio
    encodes this base rate and protects against overfitting to noise.

Q5: B) It accounts for uncertainty in the true win rate
    The naive approach assumes we KNOW the win rate is 70%. The Bayesian
    approach acknowledges we're uncertain about the true rate, leading
    to wider outcome distributions and more realistic risk estimates.
""")
```

Exercises

Complete these exercises in the `exercises.ipynb` notebook.

Exercise 1: Bayes' Theorem Calculator (Easy)

Build a function that calculates posterior probabilities given prior, likelihood, and evidence.

Exercise 2: Strategy Evaluation (Medium)

You have a strategy with 45 wins out of 60 trades. Calculate:

- The posterior distribution of the win rate
- The probability the true win rate exceeds 60%
- How many more trades you'd need to be 95% confident the rate exceeds 50%

Exercise 3: Monte Carlo Risk Analysis (Hard)

Extend the Monte Carlo simulation to include:

- Variable win/loss sizes (also uncertain)
- Transaction costs
- Maximum drawdown analysis

Next Module Preview

In **Module 2: Prior Selection and Market Knowledge**, we'll learn:

- How to translate domain expertise into prior distributions
- Conjugate priors for computational efficiency
- Prior predictive checks to validate our assumptions
- Encoding commodity seasonality in priors
