Module 2: Prior Selection and Market Knowledge Encoding

Course: Bayesian Regression and Time Series Forecasting for Commodities Trading

---

Learning Objectives

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

1. **Translate** domain expertise into mathematically rigorous prior distributions
2. **Select** appropriate conjugate priors for computational efficiency
3. **Elicit** priors from historical commodity volatility and seasonality patterns
4. **Distinguish** between weakly informative, informative, and strongly informative priors
5. **Validate** prior distributions using prior predictive checks
6. **Assess** prior sensitivity to ensure robust conclusions
7. **Implement** prior selection for real commodity forecasting problems

---

Why This Matters for Trading

Prior selection is where **Bayesian methods transform from mathematical theory into trading** In commodity markets:

- **Seasonality is real**: Corn prices peak before harvest, natural gas spikes in winter
- **Mean reversion exists**: Crude oil historically reverts to marginal production cost ($40-8 barrel)
- **Volatility clusters**: High volatility periods follow market shocks (2008, 2020)
- **Regimes shift**: OPEC decisions, climate events, and policy changes alter market dynamics

Without priors, your model treats a 300% oil price spike as equally likely as a 2% mo intelligent priors:

- You **regularize** estimates when data is noisy
- You **encode** decades of market knowledge into models
- You **prevent** overfitting to recent anomalies
- You **quantify** how strongly beliefs should influence decisions

Bad priors can bias your models. **Good priors** are the difference between a model that crashes trading and one that systematically makes money.

---

1. The Prior Spectrum: From Ignorance to Certainty

Priors exist on a continuum from complete ignorance to near-certainty:

Prior Type	When to Use	Example
Flat/Uniform	Truly no information	Beta(1 , 1) for unknown probability
Weakly Informative	Regularization, prevent extremes	Normal(0 , 1 0) for regression coefficie
Informative	Domain knowledge available	Normal(5 0 , 5) for corn seasonal peak
Strongly Informative	Physical/economic constraints	Gamma(1 0 0 , 2) for volatility (must b positive)

Mathematical Formulation

Recall Bayes' theorem:

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

The prior $P(\theta)$ can be:

- **Uninformative**: $P(\theta) \propto$ constant (equal weight to all values)
- **Weakly informative**: $P(\theta)$ gently favors reasonable values
- **Informative**: $P(\theta)$ concentrates probability mass around expert beliefs

Key insight: The influence of the prior decreases as n (sample size) increases, but with limite commodity data, priors matter significantly.

In [ ]:
```python
# 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'] = 11

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

In [ ]:
```python
# Visualize the prior spectrum for a regression coefficient
fig, axes = plt.subplots(2, 2, figsize=(14, 10))
x = np.linspace(-30, 30, 1000)

priors = [
    ("Flat Prior", stats.uniform(-30, 60), "No prior knowledge"),
    ("Weakly Informative", stats.norm(0, 10), "Mild regularization"),
    ("Informative", stats.norm(0, 3), "Some domain knowledge"),
    ("Strongly Informative", stats.norm(2, 0.5), "Strong expert belief")
]

for ax, (title, prior, description) in zip(axes.flatten(), priors):
    if isinstance(prior, stats._continuous_distns.uniform_gen):
```

```
        pdf_vals = prior.pdf(x, -30, 60)
    else:
        pdf_vals = prior.pdf(x)

    ax.plot(x, pdf_vals, 'blue', linewidth=2.5)
    ax.fill_between(x, pdf_vals, alpha=0.3, color='blue')
    ax.axvline(0, color='red', linestyle='--', alpha=0.5, label='Zero eff
    ax.set_title(f"{title}\n{description}", fontsize=12, fontweight='bold
    ax.set_xlabel('Coefficient Value (β)', fontsize=11)
    ax.set_ylabel('Probability Density', fontsize=11)
    ax.set_xlim(-30, 30)
    ax.legend()

plt.tight_layout()
plt.show()

print("\nKey Observations:")
print("1. Flat prior: All values equally likely (rarely appropriate)")
print("2. Weakly informative: Gently discourages extreme values")
print("3. Informative: Clear preference for values near zero")
print("4. Strongly informative: Almost certain the value is around 2")
```

2. Conjugate Priors: Mathematical Elegance Meets Computational Efficiency

A **conjugate prior** is a prior distribution that, when combined with a specific likelihood, produces posterior in the same distributional family. This allows for closed-form Bayesian updates without

Three Essential Conjugate Pairs for Commodities

2.1 Beta-Binomial: Win Rates and Directional Forecasts

Use case: Probability of price increases, directional forecast accuracy

$$\begin{align} \text{Prior: } & p \sim \text{Beta}(\alpha, \beta) \\ \text{Likelihood: } & X \sim \text{(n, p) \\ \text{Posterior: } & p | X \sim \text{Beta}(\alpha + x, \beta + n - x) \end{align}$$

Hyperparameter interpretation:

- α = prior "successes" (e.g., days corn price rose)
- β = prior "failures" (e.g., days corn price fell)
- Prior mean: $\mu = \frac{\alpha}{\alpha + \beta}$
- Prior strength: $\alpha + \beta$ (larger = stronger prior)

2.2 Normal-Normal: Price Levels and Returns

Use case: Estimating mean return, average price level

$$\begin{align} \text{Prior: } & \mu \sim N(\mu_0, \sigma_0^2) \\ \text{Likelihood: } & X_i \sim \sigma^2) \text{ (known variance)} \\ \text{Posterior: } & \mu | X \sim N(\mu_n, \sigma_n^2) \end{align}$$

where: $$\mu_n = \frac{\frac{\mu_ 0 }{\sigma_ 0 ^ 2 } + \frac{n\bar{x}}{\sigma^ 2 }}{\frac{ 1 } {\sigma_ 0 ^ 2 } + \frac{n}{\sigma^ 2 }}, \quad \sigma_n^ 2 = \frac{ 1 }{\frac{ 1 }{\sigma_ 0 ^ 2 } + \frac{n}{\sigma^ 2 }}$$

Interpretation: Posterior mean is a **precision-weighted average** of prior mean and sample me

2.3 Gamma-Poisson: Event Counts

Use case: Number of price spikes per month, supply disruption events

$$\begin{align} \text{Prior: } & \lambda \sim \text{Gamma}(\alpha, \beta) \\ \text{Likelihood: } & X \text{Poisson}(\lambda) \\ \text{Posterior: } & \lambda | X \sim \text{Gamma}(\alpha + \sum x_i, \b \end{align}$$

In [ ]:
```
# Example: Beta-Binomial for directional forecasting
# Scenario: Estimating probability that corn price rises in June (pre-har

def beta_binomial_update(prior_alpha, prior_beta, successes, trials):
    """
    Perform Beta-Binomial conjugate update.

    Returns:
    --------
    dict with prior, posterior, and statistics
    """
    # Prior
    prior = stats.beta(prior_alpha, prior_beta)

    # Posterior (conjugate update)
    post_alpha = prior_alpha + successes
    post_beta = prior_beta + (trials - successes)
    posterior = stats.beta(post_alpha, post_beta)

    return {
        'prior': prior,
        'posterior': posterior,
        'prior_mean': prior.mean(),
        'posterior_mean': posterior.mean(),
        'prior_std': prior.std(),
        'posterior_std': posterior.std(),
        'credible_interval': (posterior.ppf(0.025), posterior.ppf(0.975))
    }

# Prior belief: Corn rises in June about 60% of the time (historical know
# We're moderately confident: equivalent to seeing 12 rises in 20 Junes
prior_alpha = 12
prior_beta = 8

# New data: Last 10 years, corn rose in 7 Junes
successes = 7
trials = 10

result = beta_binomial_update(prior_alpha, prior_beta, successes, trials)

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

```
ax.plot(x, result['prior'].pdf(x), 'orange', linewidth=2.5,
        label=f"Prior: Beta({prior_alpha}, {prior_beta}), mean={result['p
ax.plot(x, result['posterior'].pdf(x), 'blue', linewidth=2.5,
        label=f"Posterior: mean={result['posterior_mean']:.2f}")
ax.axvline(0.5, color='red', linestyle='--', alpha=0.5, label='50% (rando
ax.fill_between(x, result['posterior'].pdf(x), alpha=0.2, color='blue')

ax.set_xlabel('Probability Corn Rises in June', fontsize=12)
ax.set_ylabel('Probability Density', fontsize=12)
ax.set_title('Beta-Binomial Conjugate Update: Corn June Seasonality', fon
ax.legend(fontsize=11)
ax.set_xlim(0.3, 0.9)

plt.tight_layout()
plt.show()

print("="*70)
print("BETA-BINOMIAL CONJUGATE UPDATE")
print("="*70)
print(f"Prior belief: Corn rises in June {result['prior_mean']:.1%} of th
print(f"Prior uncertainty: ± {result['prior_std']:.1%}")
print(f"\nObserved data: {successes} rises in {trials} Junes")
print(f"\nPosterior belief: {result['posterior_mean']:.1%}")
print(f"Posterior uncertainty: ± {result['posterior_std']:.1%}")
print(f"95% Credible Interval: [{result['credible_interval'][0]:.1%}, {re
print(f"\nInterpretation: Updated belief is weighted average of prior and
```

In [ ]:

```
# Example: Normal-Normal for crude oil mean price
# Scenario: Estimate mean WTI crude price with prior from expert knowledg

def normal_normal_update(prior_mean, prior_std, data, data_std):
    """
    Normal-Normal conjugate update (known variance case).

    Parameters:
    -----------
    prior_mean : float
        Prior belief about mean
    prior_std : float
        Prior uncertainty
    data : array
        Observed data
    data_std : float
        Known standard deviation of data
    """
    n = len(data)
    data_mean = np.mean(data)

    # Prior precision (inverse variance)
    prior_precision = 1 / prior_std**2
    data_precision = n / data_std**2

    # Posterior parameters
    post_precision = prior_precision + data_precision
    post_mean = (prior_precision * prior_mean + data_precision * data_mea
    post_std = np.sqrt(1 / post_precision)

    return {
        'prior': stats.norm(prior_mean, prior_std),
```

```
        'posterior': stats.norm(post_mean, post_std),
        'prior_mean': prior_mean,
        'posterior_mean': post_mean,
        'prior_std': prior_std,
        'posterior_std': post_std,
        'data_mean': data_mean,
        'credible_interval': (post_mean - 1.96*post_std, post_mean + 1.96
    }

# Prior: Expert believes WTI crude should be around $65, but uncertain (±
prior_mean = 65
prior_std = 15

# Data: Last 30 days average price
np.random.seed(42)
true_price = 72
data_std = 8  # Known daily volatility
observed_prices = np.random.normal(true_price, data_std, 30)

result = normal_normal_update(prior_mean, prior_std, observed_prices, dat

# Visualize
fig, ax = plt.subplots(figsize=(12, 6))
x = np.linspace(30, 100, 1000)

ax.plot(x, result['prior'].pdf(x), 'orange', linewidth=2.5,
        label=f"Prior: N({prior_mean}, {prior_std}²)")
ax.axvline(result['data_mean'], color='green', linestyle=':', linewidth=2
          label=f"Sample Mean: ${result['data_mean']:.2f}")
ax.plot(x, result['posterior'].pdf(x), 'blue', linewidth=2.5,
        label=f"Posterior: N({result['posterior_mean']:.1f}, {result['pos
ax.fill_between(x, result['posterior'].pdf(x), alpha=0.2, color='blue')

ax.set_xlabel('WTI Crude Price ($/barrel)', fontsize=12)
ax.set_ylabel('Probability Density', fontsize=12)
ax.set_title('Normal-Normal Conjugate Update: WTI Crude Mean Price', font
ax.legend(fontsize=11)

plt.tight_layout()
plt.show()

print("="*70)
print("NORMAL-NORMAL CONJUGATE UPDATE")
print("="*70)
print(f"Prior belief: Mean price = ${prior_mean:.2f} ± ${prior_std:.2f}")
print(f"Sample mean: ${result['data_mean']:.2f} (n={len(observed_prices)}
print(f"\nPosterior: Mean price = ${result['posterior_mean']:.2f} ± ${res
print(f"95% Credible Interval: [${result['credible_interval'][0]:.2f}, ${
print(f"\nNotice: Posterior is between prior and sample mean (precision-w
```

Key Insight: Conjugacy = Speed

With conjugate priors:

- **No MCMC needed**: Instant analytical updates
- **Interpretable**: Clear relationship between prior and posterior
- **Scalable**: Can update sequentially as new data arrives

When to use conjugate priors:

- Real-time trading systems (low latency)
- Simple models where conjugacy applies
- Teaching/prototyping before complex models

When NOT to use:

- Complex hierarchical models
- Non-standard likelihoods
- When you want maximum modeling flexibility

3. Prior Elicitation from Historical Data

The challenge: How do you convert "corn is volatile in August" into a prior distribution?

Strategy 1: Empirical Bayes (Use the Data Twice)

Use historical data to set hyperparameters, then use recent data for likelihood:

1. Calculate historical volatility: $\sigma_{\text{hist}} = \text{std}(\text{returns}_{2000-2
   $
2. Set prior: $\sigma \sim \text{Half-Normal}(\sigma_{\text{hist}})$
3. Update with recent data: returns$_{2020-2024}$

Pros: Data-driven, objective
Cons: "Uses data twice," not fully Bayesian

Strategy 2: Expert Elicitation

Ask domain experts to specify:

- "What's your best guess for average June corn price?" → prior mean
- "What range would you be 90% confident includes the true value?" → prior variance

Strategy 3: Maximum Entropy Priors

Choose the prior with maximum entropy subject to constraints (e.g., mean, variance). This repre
"least informative" prior given constraints.

In [ ]:
```
# Example: Eliciting volatility prior from historical corn data

# Simulate historical corn price returns (2000-2015)
np.random.seed(42)
historical_vol = 0.25  # True historical volatility
n_hist = 250 * 15  # 15 years of daily data
historical_returns = np.random.normal(0, historical_vol, n_hist)

# Calculate empirical statistics
empirical_mean = np.mean(historical_returns)
empirical_std = np.std(historical_returns, ddof=1)
```

```
print("="*70)
print("PRIOR ELICITATION: Corn Volatility")
print("="*70)
print(f"Historical data: {n_hist:,} daily returns (2000-2015)")
print(f"\nEmpirical mean return: {empirical_mean:.4f}")
print(f"Empirical volatility (std): {empirical_std:.4f}")
print(f"\nPrior specification:")
print(f"  Mean return: μ ~ Normal(0, 0.01)  [weakly informative, expect ~
print(f"  Volatility: σ ~ Half-Normal({empirical_std:.3f})  [from histori

# Visualize the elicited volatility prior
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Historical returns distribution
ax = axes[0]
ax.hist(historical_returns, bins=50, density=True, alpha=0.6, color='blue
x_range = np.linspace(-1, 1, 1000)
ax.plot(x_range, stats.norm(0, empirical_std).pdf(x_range), 'red', linewi
        label=f'Fitted Normal(0, {empirical_std:.3f})')
ax.set_xlabel('Daily Return', fontsize=12)
ax.set_ylabel('Density', fontsize=12)
ax.set_title('Historical Returns Distribution', fontsize=12, fontweight='
ax.legend()
ax.set_xlim(-1, 1)

# Elicited volatility prior
ax = axes[1]
vol_range = np.linspace(0, 0.6, 1000)
# Half-Normal prior for volatility
prior_vol = stats.halfnorm(scale=empirical_std)
ax.plot(vol_range, prior_vol.pdf(vol_range), 'orange', linewidth=2.5,
        label=f'Prior: Half-Normal(σ={empirical_std:.3f})')
ax.axvline(empirical_std, color='red', linestyle='--', linewidth=2, label
ax.fill_between(vol_range, prior_vol.pdf(vol_range), alpha=0.3, color='or
ax.set_xlabel('Volatility (σ)', fontsize=12)
ax.set_ylabel('Probability Density', fontsize=12)
ax.set_title('Elicited Volatility Prior', fontsize=12, fontweight='bold')
ax.legend()

plt.tight_layout()
plt.show()

print(f"\nPrior predictive check: Generate returns from prior")
# Sample from prior
n_prior_samples = 1000
prior_vol_samples = prior_vol.rvs(n_prior_samples)
print(f"Prior volatility samples: mean={np.mean(prior_vol_samples):.3f},
      f"90% interval=[{np.percentile(prior_vol_samples, 5):.3f}, {np.perc
print(f"This captures our uncertainty about true volatility before seeing
```

4 . Weakly Informative vs Informative Priors

Weakly Informative Priors: The Goldilocks Zone

Goal: Regularize the model without imposing strong beliefs

Characteristics:

- Wide enough to not bias estimates
- Narrow enough to rule out nonsense values
- Often used for nuisance parameters

Examples:

- Regression coefficient: $\beta \sim N(0, 10)$
  - Allows large effects if data supports it
  - But coefficients of 100 are implausible
- Volatility: $\sigma \sim \text{Half-Cauchy}(0, 2.5)$
  - Heavy tails allow high volatility if needed
  - But infinite volatility ruled out

Informative Priors: Encoding Real Knowledge

When to use:

- Strong domain expertise exists
- Physical/economic constraints apply
- Regularizing against overfitting to noise

Commodity examples:

- **Seasonal amplitude**: Historical seasonality rarely exceeds ±15%
  - Prior: $A \sim N(0, 0.10)$ (10% seasonal swing)
- **Mean reversion speed**: Economics suggests $\lambda \in [0.1, 1]$ per year
  - Prior: $\lambda \sim \text{Beta}(2, 2)$ rescaled to $[0, 1]$
- **Oil price floor**: Can't go negative (pre-2020!), marginal cost ~$40
  - Prior: $P_{\text{min}} \sim \text{Truncated-Normal}(40, 10, \text{lower}=0)$

In [ ]:
```
# Compare weakly informative vs informative priors with limited data

# Scenario: Estimating seasonal effect in natural gas prices (winter spik
# True effect: +20% in winter

np.random.seed(42)
true_seasonal_effect = 0.20
n_observations = 15  # Only 15 winters of data

# Generate noisy observations
observed_effects = np.random.normal(true_seasonal_effect, 0.15, n_observa

# Define priors
priors = {
    'Weakly Informative': stats.norm(0, 0.50),  # Wide, centered at zero
    'Informative': stats.norm(0.15, 0.10),      # Based on historical kno
}

# Calculate posteriors (assuming known std of 0.15)
data_mean = np.mean(observed_effects)
data_std = 0.15
```

```
fig, axes = plt.subplots(1, 2, figsize=(14, 5))
x = np.linspace(-0.3, 0.7, 1000)

for ax, (name, prior) in zip(axes, priors.items()):
    # Update using Normal-Normal conjugacy
    prior_mean = prior.mean()
    prior_std = prior.std()

    prior_prec = 1 / prior_std**2
    data_prec = n_observations / data_std**2

    post_prec = prior_prec + data_prec
    post_mean = (prior_prec * prior_mean + data_prec * data_mean) / post_
    post_std = np.sqrt(1 / post_prec)

    posterior = stats.norm(post_mean, post_std)

    # Plot
    ax.plot(x, prior.pdf(x), 'orange', linewidth=2, label='Prior')
    ax.axvline(data_mean, color='green', linestyle=':', linewidth=2, labe
    ax.plot(x, posterior.pdf(x), 'blue', linewidth=2, label='Posterior')
    ax.axvline(true_seasonal_effect, color='red', linestyle='--', linewid
    ax.fill_between(x, posterior.pdf(x), alpha=0.2, color='blue')

    ax.set_xlabel('Seasonal Effect', fontsize=12)
    ax.set_ylabel('Density', fontsize=12)
    ax.set_title(f'{name} Prior\nPost. Mean: {post_mean:.2f} ± {post_std:
                 fontsize=12, fontweight='bold')
    ax.legend(fontsize=10)
    ax.set_xlim(-0.3, 0.7)

plt.tight_layout()
plt.show()

print("="*70)
print("PRIOR STRENGTH COMPARISON")
print("="*70)
print(f"True seasonal effect: {true_seasonal_effect:.0%}")
print(f"Observed data mean: {data_mean:.2f} (n={n_observations})")
print(f"\nWith limited data, the informative prior pulls estimate closer
print(f"This is regularization in action.")
```

5 . Prior Predictive Checks: Validating Your Priors

The question: Before seeing any data, does my prior generate realistic data?

Prior Predictive Distribution

$$P(\tilde{y}) = \int P(\tilde{y} | \theta) P(\theta) d\theta$$

In words: Average the likelihood over all possible parameter values weighted by the prior.

Workflow:

1 . **Sample** parameter values from prior: $\theta^{(1)}, \ldots, \theta^{(N)} \sim P(\theta)$
2 . For each $\theta^{(i)}$, **generate** fake data: $\tilde{y}^{(i)} \sim P(y | \theta^{(i)})$

3 . **Inspect** simulated datasets:

- Do they look like real commodity data?
- Are the ranges reasonable?
- Do they exhibit expected features (volatility clustering, seasonality)?

Red Flags:

- Simulated prices go negative (need positivity constraint)
- Volatility is 500% (unrealistic for most commodities)
- No seasonal patterns when you expect them

In [ ]:

```
# Prior predictive check for corn price model
# Model: log(Price_t) = μ + β*sin(2π*t/365) + ε, where ε ~ N(0, σ)

# Define priors
prior_mu = stats.norm(np.log(400), 0.3)       # Mean log-price around $40
prior_beta = stats.norm(0, 0.15)              # Seasonal amplitude (15% s
prior_sigma = stats.halfnorm(scale=0.10)      # Daily volatility

# Prior predictive sampling
n_prior_samples = 100
n_days = 365 * 2  # 2 years
t = np.arange(n_days)

fig, axes = plt.subplots(2, 1, figsize=(14, 10))

# Generate prior predictive samples
for i in range(n_prior_samples):
    # Sample parameters from priors
    mu_sample = prior_mu.rvs()
    beta_sample = prior_beta.rvs()
    sigma_sample = prior_sigma.rvs()

    # Generate data from model
    seasonal_component = beta_sample * np.sin(2 * np.pi * t / 365)
    noise = np.random.normal(0, sigma_sample, n_days)
    log_price = mu_sample + seasonal_component + noise
    price = np.exp(log_price)

    # Plot
    axes[0].plot(t, price, alpha=0.3, color='blue', linewidth=0.5)

axes[0].set_xlabel('Days', fontsize=12)
axes[0].set_ylabel('Corn Price (¢/bushel)', fontsize=12)
axes[0].set_title('Prior Predictive Check: Simulated Corn Prices from Pri
                  fontsize=13, fontweight='bold')
axes[0].set_ylim(200, 800)
axes[0].axhline(400, color='red', linestyle='--', alpha=0.5, label='Expec
axes[0].legend()

# Distribution of simulated prices at a single time point
simulated_prices_t0 = []
for i in range(5000):
    mu_sample = prior_mu.rvs()
    beta_sample = prior_beta.rvs()
    sigma_sample = prior_sigma.rvs()
    log_price = mu_sample + beta_sample * np.sin(0) + np.random.normal(0,
```

```
    simulated_prices_t0.append(np.exp(log_price))

axes[1].hist(simulated_prices_t0, bins=50, density=True, alpha=0.6, color
            label='Prior predictive distribution')
axes[1].axvline(400, color='red', linestyle='--', linewidth=2, label='Exp
axes[1].set_xlabel('Corn Price (¢/bushel)', fontsize=12)
axes[1].set_ylabel('Density', fontsize=12)
axes[1].set_title('Prior Predictive Distribution at t=0', fontsize=13, fo
axes[1].legend()
axes[1].set_xlim(150, 800)

plt.tight_layout()
plt.show()

print("="*70)
print("PRIOR PREDICTIVE CHECK INTERPRETATION")
print("="*70)
print(f"\nSimulated price range: ${np.percentile(simulated_prices_t0, 1):
print(f"Median: ${np.median(simulated_prices_t0):.0f}")
print(f"\nQuestions to ask:")
print(f"  1. Do these prices look realistic? ✓ (corn trades $300-600 typi
print(f"  2. Is seasonality visible? ✓ (smooth annual cycles)")
print(f"  3. Are there negative prices? ✗ (log-scale prevents this)")
print(f"  4. Is volatility reasonable? ✓ (not too wild)")
print(f"\nConclusion: Prior seems reasonable! Proceed to fit model.")
```

6. Prior Sensitivity Analysis

The concern: What if my prior is wrong? Will it ruin my conclusions?

Sensitivity Analysis Workflow

1. **Fit model** with your chosen prior
2. **Refit** with several alternative priors:
   - More skeptical (narrower)
   - More vague (wider)
   - Different center
3. **Compare** posterior inferences:
   - Do point estimates change substantially?
   - Do credible intervals overlap?
   - Do trading decisions change?

Interpretation:

- **Robust**: Conclusions similar across priors → data dominates
- **Sensitive**: Conclusions vary widely → need more data or justify prior choice

Trading rule: If your P&L depends critically on prior choice, you don't have enough data to trade strategy yet.

In [ ]:
```
# Prior sensitivity analysis: Does conclusion depend on prior?
# Scenario: Is natural gas seasonal spike > 10%?

np.random.seed(42)
```

```python
true_effect = 0.18
n_obs = 20
observed_data = np.random.normal(true_effect, 0.08, n_obs)
data_mean = np.mean(observed_data)
data_std = 0.08

# Test multiple priors
prior_scenarios = {
    'Skeptical': stats.norm(0.05, 0.05),
    'Neutral': stats.norm(0.10, 0.10),
    'Optimistic': stats.norm(0.20, 0.08),
    'Vague': stats.norm(0, 0.50),
}

fig, axes = plt.subplots(2, 2, figsize=(14, 10))
x = np.linspace(-0.1, 0.5, 1000)

results = {}
for ax, (name, prior) in zip(axes.flatten(), prior_scenarios.items()):
    # Conjugate update
    prior_mean = prior.mean()
    prior_std = prior.std()

    prior_prec = 1 / prior_std**2
    data_prec = n_obs / data_std**2

    post_prec = prior_prec + data_prec
    post_mean = (prior_prec * prior_mean + data_prec * data_mean) / post_
    post_std = np.sqrt(1 / post_prec)

    posterior = stats.norm(post_mean, post_std)

    # Calculate P(effect > 10%)
    prob_above_10 = 1 - posterior.cdf(0.10)

    results[name] = {
        'posterior_mean': post_mean,
        'posterior_std': post_std,
        'prob_above_10': prob_above_10
    }

    # 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(0.10, color='red', linestyle='--', linewidth=2, label='10%
    ax.fill_betweenx([0, ax.get_ylim()[1]], 0.10, 0.5, alpha=0.1, color='
    ax.axvline(data_mean, color='green', linestyle=':', linewidth=1.5, al

    ax.set_xlabel('Seasonal Effect', fontsize=11)
    ax.set_ylabel('Density', fontsize=11)
    ax.set_title(f'{name} Prior\nP(effect > 10%) = {prob_above_10:.1%}',
                 fontsize=12, fontweight='bold')
    ax.legend(fontsize=9)
    ax.set_xlim(-0.1, 0.5)

plt.tight_layout()
plt.show()

# Summary table
print("="*70)
```

```
print("PRIOR SENSITIVITY ANALYSIS")
print("="*70)
print(f"\nObserved data: mean = {data_mean:.2f}, n = {n_obs}")
print(f"True effect: {true_effect:.0%}\n")
print(f"{'Prior':<15} {'Post. Mean':>12} {'Post. Std':>12} {'P(>10%)':>12
print("-"*70)
for name, res in results.items():
    print(f"{name:<15} {res['posterior_mean']:>12.2f} {res['posterior_std

print(f"\nConclusion: All priors lead to P(effect > 10%) > 90%")
print(f"Result is ROBUST to prior choice - data dominates!")
```

7 . Practical Application: Encoding Corn Seasonality

Let's apply everything we've learned to a realistic commodity trading problem.

Problem Statement

You're building a model to forecast corn prices. You know:

- Corn has strong seasonality (planting in spring, harvest in fall)
- Prices typically peak in June-July (pre-harvest fear of supply shortage)
- Prices typically bottom in October-November (post-harvest glut)

Model

$$\log(P_t) = \mu + \beta_ 1 \sin\left(\frac{ 2 \pi t}{ 3 6 5 }\right) + \beta_ 2 \cos\left(\frac{ { 3 6 5 }\right) + \epsilon_t$$

where:

- μ = baseline log-price
- $\beta_ 1 , \beta_ 2 $ = seasonal coefficients
- Seasonal amplitude = $\sqrt{\beta_ 1 ^ 2 + \beta_ 2 ^ 2 }$
- Peak timing = $\arctan(\beta_ 2 / \beta_ 1)$

Prior Selection

From historical knowledge:

- Average price: ~$ 4 . 0 0 /bushel → $\mu \sim N(\log(4), 0 . 2)$
- Seasonal swing: 1 0 - 1 5 % → $\beta_ 1 , \beta_ 2 \sim N(0 , 0 . 1 0)$
- Volatility: 2 0 - 3 0 % annualized → $\sigma \sim \text{Half-Normal}(0 . 2 5 /\sqrt{ 2

In [ ]:
```
# Full seasonal model for corn prices

# Generate synthetic corn price data with known seasonality
np.random.seed(42)
n_days = 365 * 3  # 3 years
t = np.arange(n_days)

# True parameters
true_mu = np.log(4.0)
```

```
true_beta1 = 0.12  # Sin coefficient
true_beta2 = 0.08  # Cos coefficient
true_sigma = 0.015  # Daily volatility

# Generate data
seasonal = true_beta1 * np.sin(2*np.pi*t/365) + true_beta2 * np.cos(2*np.
noise = np.random.normal(0, true_sigma, n_days)
log_prices = true_mu + seasonal + noise
prices = np.exp(log_prices)

# Design matrix for regression
X = np.column_stack([
    np.ones(n_days),                    # Intercept
    np.sin(2*np.pi*t/365),              # Sin component
    np.cos(2*np.pi*t/365)               # Cos component
])
y = log_prices

# Bayesian linear regression with informative priors
# Prior parameters
prior_mean = np.array([np.log(4.0), 0, 0])  # μ, β1, β2
prior_cov = np.diag([0.2**2, 0.10**2, 0.10**2])  # Diagonal covariance

# Likelihood: y | β, σ² ~ N(Xβ, σ²I)
# We'll use known σ² for conjugacy (in practice, estimate this too)
sigma_sq = true_sigma**2

# Posterior (Normal-Normal conjugate update for regression)
# Posterior precision = Prior precision + Data precision
prior_precision = np.linalg.inv(prior_cov)
data_precision = X.T @ X / sigma_sq

post_precision = prior_precision + data_precision
post_cov = np.linalg.inv(post_precision)

# Posterior mean = post_cov @ (prior_precision @ prior_mean + data_precis
post_mean = post_cov @ (prior_precision @ prior_mean + (X.T @ y) / sigma_

print("="*70)
print("BAYESIAN SEASONAL MODEL: Corn Prices")
print("="*70)
print(f"\nTrue parameters:")
print(f"  μ (log-price): {true_mu:.3f} → ${np.exp(true_mu):.2f}/bushel")
print(f"  β₁ (sin): {true_beta1:.3f}")
print(f"  β₂ (cos): {true_beta2:.3f}")
print(f"  Amplitude: {np.sqrt(true_beta1**2 + true_beta2**2):.3f} ({100*n

print(f"\nPosterior estimates:")
print(f"  μ: {post_mean[0]:.3f} ± {np.sqrt(post_cov[0,0]):.3f} → ${np.exp
print(f"  β₁: {post_mean[1]:.3f} ± {np.sqrt(post_cov[1,1]):.3f}")
print(f"  β₂: {post_mean[2]:.3f} ± {np.sqrt(post_cov[2,2]):.3f}")
amplitude_est = np.sqrt(post_mean[1]**2 + post_mean[2]**2)
print(f"  Amplitude: {amplitude_est:.3f} ({100*amplitude_est:.1f}%)")

# Fitted values
fitted_log_prices = X @ post_mean
fitted_prices = np.exp(fitted_log_prices)

# Visualize
fig, axes = plt.subplots(2, 1, figsize=(14, 10))
```

```
# Time series
ax = axes[0]
dates = pd.date_range('2021-01-01', periods=n_days, freq='D')
ax.plot(dates, prices, 'o', alpha=0.3, markersize=2, label='Observed pric
ax.plot(dates, fitted_prices, 'red', linewidth=2, label='Bayesian fit (po
ax.set_xlabel('Date', fontsize=12)
ax.set_ylabel('Corn Price ($/bushel)', fontsize=12)
ax.set_title('Corn Price Seasonality: Bayesian Regression', fontsize=14,
ax.legend()
ax.grid(alpha=0.3)

# Seasonal component only
ax = axes[1]
seasonal_component = post_mean[1] * np.sin(2*np.pi*t/365) + post_mean[2]
ax.plot(t % 365, seasonal_component, 'o', alpha=0.5, markersize=3, color=
ax.axhline(0, color='black', linestyle='--', alpha=0.5)
ax.set_xlabel('Day of Year', fontsize=12)
ax.set_ylabel('Seasonal Effect (log-price)', fontsize=12)
ax.set_title('Extracted Seasonal Pattern', fontsize=14, fontweight='bold'
ax.set_xlim(0, 365)
ax.grid(alpha=0.3)

# Mark key agricultural dates
key_dates = [
    (120, 'Planting'),
    (180, 'Peak (pre-harvest)'),
    (280, 'Harvest'),
]
for day, label in key_dates:
    ax.axvline(day, color='red', linestyle=':', alpha=0.7)
    ax.text(day, ax.get_ylim()[1]*0.9, label, rotation=90, fontsize=9)

plt.tight_layout()
plt.show()

print(f"\nTrading insight: Seasonal pattern shows {amplitude_est:.1%} swi
print(f"Peak occurs around day {np.argmax(seasonal_component):.0f} (late
print(f"Trough occurs around day {np.argmin(seasonal_component):.0f} (lat
```

8 . Summary: The Art and Science of Prior Selection

Key Principles

Principle	Guidance
Honesty	Priors should reflect genuine beliefs, not desired conclusions
Transparency	Document and justify prior choices
Calibration	Use prior predictive checks to validate
Sensitivity	Test robustness to prior specification
Parsimony	Use weakly informative priors when in doubt

The Prior Selection Checklist

Before finalizing priors:

1. ✅ **Physical constraints**: Are negative prices/volatilities possible?
2. ✅ **Domain knowledge**: What do experts believe?
3. ✅ **Historical data**: What have we seen before?
4. ✅ **Prior predictive**: Do simulations look realistic?
5. ✅ **Sensitivity**: Do conclusions depend critically on prior?
6. ✅ **Documentation**: Can someone else understand your choices?

Common Mistakes to Avoid

- ❌ Using flat priors for unbounded parameters (they're not actually "uninformative")
- ❌ Picking priors to get desired answers (confirmation bias)
- ❌ Ignoring domain expertise when it exists
- ❌ Using overly confident priors with limited justification
- ❌ Failing to check prior predictive distributions

When Priors Matter Most

- **Limited data**: < 100 observations
- **High-dimensional models**: Many parameters relative to data
- **Hierarchical models**: Priors on hyperparameters strongly influence results
- **Risk management**: Tail probabilities sensitive to prior specification

---

Knowledge Check Quiz

Q 1 : A conjugate prior is valuable because:

- A) It always gives the most accurate results
- B) It allows closed-form posterior updates without MCMC
- C) It's always the correct prior to use
- D) It eliminates the need for data

Q 2 : In a Beta-Binomial model, increasing the prior hyperparameters (α + β) makes the prior:

- A) More influential (stronger)
- B) Less influential (weaker)
- C) Wider and more uncertain
- D) Has no effect

Q 3 : Prior predictive checks help you:

- A) Calculate the posterior distribution
- B) Determine if your prior generates realistic data

- C) Avoid using priors altogether
- D) Maximize the likelihood

Q 4 : A weakly informative prior is appropriate when:

- A) You have very strong domain knowledge
- B) You want regularization without imposing strong beliefs
- C) The data is highly informative
- D) You want results identical to frequentist methods

Q 5 : If your trading decision changes dramatically with different reasonable priors, you should:

- A) Pick the prior that gives the best backtest results
- B) Use a flat prior
- C) Recognize you need more data or a stronger justification for your prior
- D) Ignore the sensitivity and proceed

In [ ]:
```
# Quiz Answers
print("="*70)
print("QUIZ ANSWERS")
print("="*70)
print("""
Q1: B) It allows closed-form posterior updates without MCMC
    Conjugate priors provide computational efficiency through analytical
    solutions. They're not always "correct" but are very useful.

Q2: A) More influential (stronger)
    α + β represents "prior sample size." Larger values mean the prior
    counts more relative to the data, making it stronger/more influential

Q3: B) Determine if your prior generates realistic data
    Prior predictive checks sample from P(data|prior) to verify that
    your prior can generate datasets that look like real commodity data.

Q4: B) You want regularization without imposing strong beliefs
    Weakly informative priors are the "Goldilocks" choice: they prevent
    nonsense values without strongly biasing results.

Q5: C) Recognize you need more data or a stronger justification for your
    Prior sensitivity indicates conclusions aren't robust. Either collect
    more data or carefully justify your prior choice. Never pick priors
    to optimize backtests (overfitting)!
""")
```

Exercises

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

Exercise 1 : Conjugate Prior Derivation (Medium)

Derive the posterior parameters for the Normal-Normal conjugate pair. Verify your derivation mat
formula in the notes.

Exercise 2 : Agricultural Seasonality (Medium)

Using the corn seasonality model, encode priors for wheat (different growing season). Wheat is p in fall, harvested in summer. How should β_1 and β_2 change?

Exercise 3 : Prior Sensitivity in Trading (Hard)

You have a mean-reversion trading strategy. Build a model with a prior on the mean-reversion sp Test sensitivity to three priors: skeptical ($\lambda \sim 0$), moderate ($\lambda \sim 0.5$), strong ($\lambda \sim 1$). With only trades, how much do P&L distributions differ?

Exercise 4 : Prior Elicitation Interview (Hard)

Interview a hypothetical crude oil expert to elicit a prior for next year's average price. Convert the qualitative statements into a Normal distribution:

- "Most likely around $75"
- "Very unlikely to be below $50 or above $100"
- "90% confident it's between $60 and $90"

Next Module Preview

In **Module 3 : MCMC and Computational Inference**, we'll learn:

- Why conjugate priors aren't always possible
- How MCMC algorithms sample from complex posteriors
- Implementing Metropolis-Hastings from scratch
- Using PyMC for production-grade Bayesian inference
- Diagnosing convergence and sampling problems
- Applying MCMC to real commodity price forecasting

Module 2 Complete