Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

# Basic Bayesian 🎲

## [Dr. Hanjo Odendaal]

# The 5 steps of Bayesian data analysis

In general, Bayesian analysis of data follows these steps:

1) **Identify the relevant data**

- Figure out which data matters for your research questions.
- What type of measurements do you have?
- Which variables are you trying to predict, and which ones help make those predictions?

2) **Choose a model that describes the data**

- Pick a mathematical model that makes sense for your data.
- The model and its settings should align with the goals of your analysis
- Bayesian estimation allows for a lot of flexibility in specification e.g., binomial, gamma etc.

# The 5 steps of Bayesian data analysis

3) **Set a prior belief about the parameters**

- Before looking at the data, define your initial assumptions.
- These assumptions should be reasonable and convincing to your audience (e.g., other scientists).
- Usually derived from other research on theory. (More on this later)

4) **Update beliefs based on the data**

- Use Bayesian methods to refine your understanding by combining your prior beliefs with new evidence.
- Interpret the updated results (posterior distribution) in a way that makes sense for your research.

5) **Check if the model makes good predictions**

- Compare the model's predictions to the actual data ("posterior predictive check") aka `pp_check()`.
- If it doesn't match well, consider tweaking the model or using a different one.
    - We will do this analytically!

[1] *See A Solomon Kurz extensive work on this matter.*

Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

# Frequentist  vs Bayesian 

# Why and when Bayesian?

📌 Why Use Bayesian Models?

- **Frequentist vs. Bayesian:** All frequentist models can be approximated by a Bayesian model, but not all Bayesian models can be modeled in a frequentist way.
- **Handle Complex Models:** Fit more advanced models with temporal and spatial components or differential equations (ODEs).
- **Easier Interpretation:** Sometimes more intuitive to understand and explain.
- **Full Probability Distributions:** Obtain a complete probability distribution for each parameter, making simulations and calculations smoother.
- **Use Prior Knowledge:** Incorporate additional data, conclusions, or expectations with priors.
- **Better Outlier Handling:** Bayesian models can be more robust to extreme values by incorporating prior knowledge and using heavy-tailed distributions.

*Otherwise...*

🎯 When you have a lot of data, frequentist methods tend to perform well, as the law of large numbers ensures stable parameter estimates without the need for priors.

# Kamala or Trump

# Kamala or Trump

You have been hired as a statistical consultant to determine whether Trump is leading in an election. The true proportion of voters who support Trump is either 45% or 55%. Your tool to determine is polling...

You need to make a decision, and for a given choice, there is a payoff/loss you must consider. If you win, you get a big office in the new candidates cabinet, otherwise you are out. You get a quote and the price is $200 per person and they only deal 5 respondents at a time ($1000). Your total budget is $4000. So your options is to poll 5, 10, ..., 20 people.

💡 Making a wrong decision is high, so you need to be quite confident, BUT polling is also costly, so you don't want to spend more than you need to, to get your answer.

Lets see how a frequentist vs bayesian consultant would tackle this problem.

# Frequentist Approach

We start off by saying we need our confidence to be at the $\alpha = 0.05$ level and we are going to poll 5 people, so $n = 5$.

- Null Hypothesis ( $H_0$ = 45%): Trump has 45% support.
- Alternative Hypothesis ( $H_1$ > 45%): Trump has more than 45% support.

Suppose you survey 5 random voters and find that 2 of them support trump.

The probability of getting $k = 2$ or more supporters under the null hypothesis ( $p = 0.45$ ) is calculated as P-value:

$$P(k \geq 2 \mid n = 5, p = 0.45) = 1 - P(k = 0 \mid n = 5, p = 0.45) - P(k = 1 \mid n = 5, p = 0.45)$$

$$P(k \geq 2 \mid n = 5, p = 0.45) = 1 - 0.0503 - 0.206 = 0.7437$$

This means 74.37% probability of observing at least 2 Trump supporters in a sample of 5 voters if his actual support is 45%. Since this probability (p-value) is greater than the typical significance level (0.05), we fail to reject the null hypothesis.

```
1- dbinom(0, 5, 0.45) - dbinom(1, 5, 0.45)
```

```
## [1] 0.7437825
```

# Bayesian Approach

The Bayesian inference approach works differently from the frequentist approach.

- $H_1$ = 45%, and $H_2$ = 55%, so Trump has 45% or 55% support.

We assume equal priors, meaning we initially believe both hypotheses are equally likely: $P(H_1) = P(H_2) = 0.5$

Using the binomial probability mass function (PMF)[1] (`dbinom`), we calculate the **Likelihood**:

$$P(k = 2 \mid H_1) = \binom{5}{2}(0.45)^2(1 - 0.45)^3$$

$$= 10 \times (0.45)^2 \times (0.55)^3 \approx 0.337$$

$$P(k = 2 \mid H_2) = \binom{5}{2}(0.55)^2(1 - 0.55)^3$$

$$= 10 \times (0.55)^2 \times (0.45)^3 \approx 0.276$$

[1] *Conventionally interpreted as the number of 'successes' in size trials.*

# Bayesian Approach

Once we have the **Likelihood**, we can calculate the **Posterior Probabilities** using Bayes' Theorem:

$$P(H_1 \mid k = 2) = \frac{P(H_1)P(k = 2 \mid H_1)}{P(k = 2)}$$

$$= \frac{0.5 \times 0.337}{(0.5 \times 0.337) + (0.5 \times 0.276)}$$

$$\approx \frac{0.145}{0.145 + 0.165} = \frac{0.17}{0.31} \approx 0.55$$

$$P(H_2 \mid k = 2) = 1 - P(H_1 \mid k = 2) = 1 - 0.55 = 0.45$$

🚨 Since these values are close, the Bayesian approach does not strongly favor either hypothesis with such a small sample size! With equal priors and a low sample size, it is difficult to make a decision with a strong confidence, given the observed data. That said, $H_1$ has a higher posterior probability than $H_2$, so if we had to make a decision at this point, we should pick $H_1$.

- Note that this decision agrees with the decision based on the frequentist approach, but with much less confidence.

# Conclusion

- As sample size increases ( $N$ ), the Bayesian posterior probabilities, $P(H \mid D)$, shift more strongly toward the true proportion.
- The Bayesian method updates beliefs dynamically, while frequentist inference only considers the probability under the null hypothesis ( $\alpha$ ). This shows that the frequentist method is highly sensitive to the null hypothesis.

| **Observed Data** | **Frequentist $P(k$ or more $\mid$ 45% Trump$)$** | **Bayesian $P(45\%$ Trump $\mid n, k)$** | **Bayesian $P(55\%$ Trump $\mid n, k)$** |
|---|---|---|---|
| $n = 5, k = 2$ | 0.7438 | 0.55 | 0.45 |
| $n = 10, k = 4$ | 0.7340 | 0.60 | 0.40 |
| $n = 15, k = 6$ | 0.7392 | 0.64 | 0.36 |
| $n = 20, k = 8$ | 0.7480 | 0.69 | 0.31 |

# Importance to estimation of regression

In the final example we will be estimating the relationship between inflation and unemployment (also known as the Philips Curve) using the `brms` package. Why is the previous example important for the upcoming example?

**Frequentist:**

- Frequentist methods do not incorporate prior beliefs about $\beta$ (e.g., economic theory suggesting $\beta < 0$).
- Confidence intervals rely on large samples; small samples may lead to unstable estimates.
- Estimates do not update as new data arrive and outliers can skew estimation through leverage effects.

**Bayesian:**

- The posterior mean of $\beta$ balances the information from the prior and the data.
- As more data are observed, the posterior variance shrinks, meaning the estimate becomes more precise and we can trust the estimation more. So uses *credible* instead of *confidence* intervals

If we have large amounts of historical inflation-unemployment data, Frequentist OLS regression works well. But if we have limited data or want to incorporate prior economic knowledge, Bayesian estimation is a better choice. Bayesian methods are more flexible and provides credibility intervals, while frequentist approach is simple.

# Terminology

Applied to parameters and data:

$$\underbrace{p(\theta \mid D)}_{\text{posterior}} = \frac{\overbrace{p(\theta)}^{\text{prior}}\overbrace{p(D \mid \theta)}^{\text{likelihood}}}{\underbrace{p(D)}_{\text{evidence}}}$$

As per Kruschke, 2015, pp. 106–107:

- The "prior", $p(\theta)$, is the credibility of the $\theta$ values without the data $D$.
- The "posterior" $p(\theta|D)$, is the credibility of $\theta$ values with the data $D$ taken into account.
- The "likelihood" $p(D|\theta)$, is the probability that the data could be generated by the model with parameter value $\theta$.
- The "evidence" (or "Marginal Probability") for the model, $p(D)$, is the overall probability of the data according to the model, determined by averaging across all possible parameter values weighted by the strength of belief in those parameter values.

Lets see how these apply when we think about the relationship between inflation and unemployment. NB - I leave out the intercept for simplicity.

# Terminology: Prior

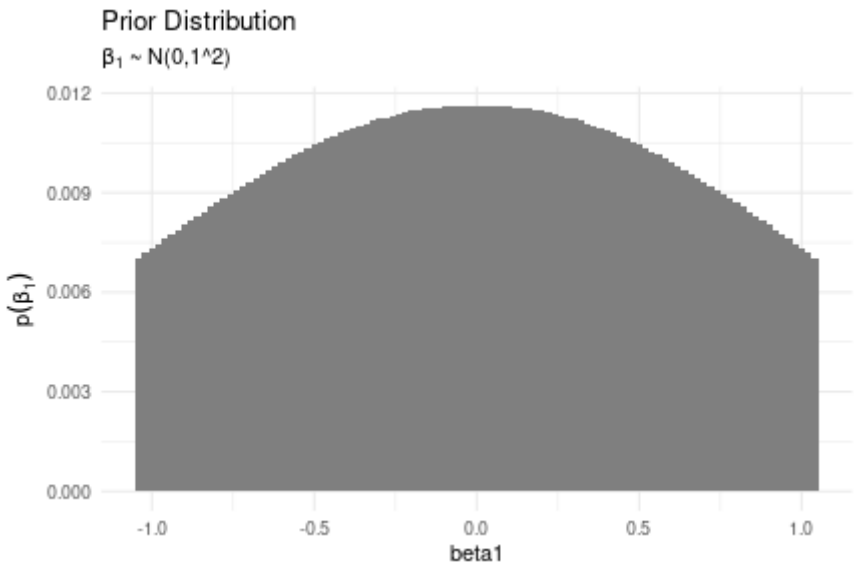
**Prior:** I believe that the relationship between inflation and employment is: $\pi = \beta_0 + 0 \times unemployment$.

We start with a prior belief that $\beta_1 = 0$ with some uncertainty. We assume a normal prior $\beta_1 \sim N(0, 0.1^2)$. Lets create a discrete grid of candidate $\beta_1$ values for illustration. We compute the corresponding density from the normal distribution, and then normalize these values.

```
beta_grid ← seq(-1, 1, length.out = 101)

prior ← dnorm(beta_grid, mean = 0, sd = 1)
prior ← prior / sum(prior)
df ← tibble(beta1 = beta_grid, prior = prior)
```

Normalise in the last step to ensure sum = 1:

$P(\beta_1 \mid \text{data}) \propto P(\text{data} \mid \beta_1)P(\beta_1)$



# Terminology: Likelihood

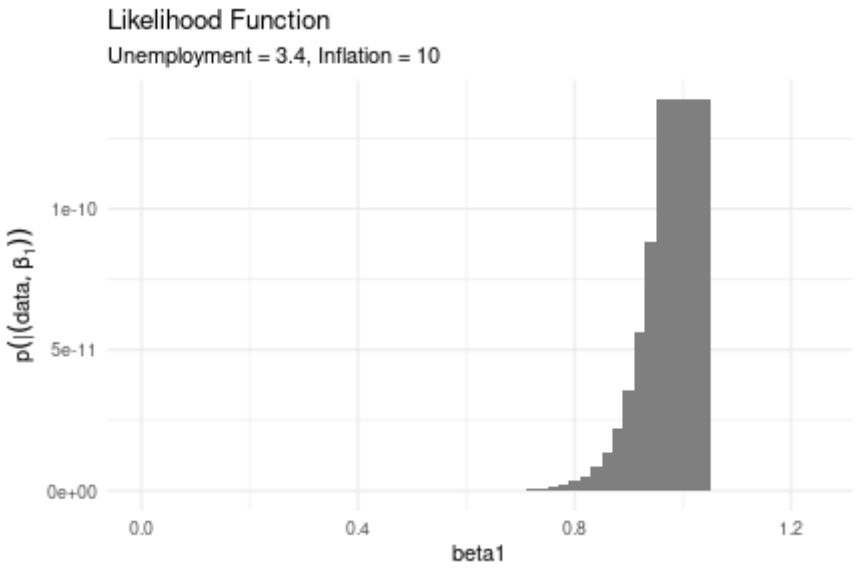
**Likelihood:** Now the first data point comes in and its unemployment = 3.4, inflation = 10.

We now ask, what is the probability of inflation being 10, given $\beta_1$, modeled as a normal distribution with mean $3.4 \times \beta_1$ and $\sigma = 1$:

$$P(\text{inflation} = 10 \mid \beta_1) = \text{dnorm}(10, \text{mean} = 3.4 \times \beta_1, \text{sd} = 1)$$

```
likelihood ← dnorm(10, mean = 3.4 * beta_grid, sd
df ← df %>% mutate(likelihood = likelihood)
```

- Probability of observing inflation = 10 for each candidate $\beta_1$.



# Terminology: Marginal Probability

**Marginal Probability:** The marginal probability is calculated by summing the product of the prior and the likelihood over all candidate values of $\beta_1$.

This step is important in normalizing the posterior:

$$P(\text{D}) = \sum_{\beta_1} \text{Prior}(\beta_1) \times \text{Likelihood}(\beta_1)$$

Think of $P(D)$ as a way of ensuring that after updating our beliefs with the likelihood ("Bayesian Thinking"), the resulting posterior still follows the rules of probability. Without this normalization step, the posterior might not sum to 1 - which would make it meaningless as a probability distribution.

```
marginal_probability ← sum(df$prior * df$likelihood)
```

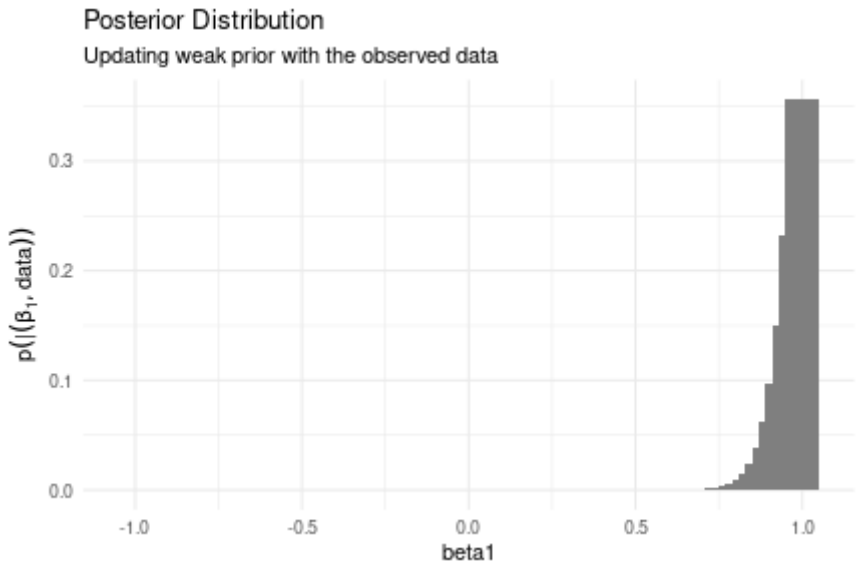
# Terminology: Posterior

And now for the final step:

$$\underbrace{p(\theta \mid D)}_{\text{posterior}} = \frac{\overbrace{p(\theta)}^{\text{prior}}\overbrace{p(D \mid \theta)}^{\text{likelihood}}}{\underbrace{p(D)}_{\text{evidence}}}$$

We compute this for each candidate $\beta_1$ on our grid:

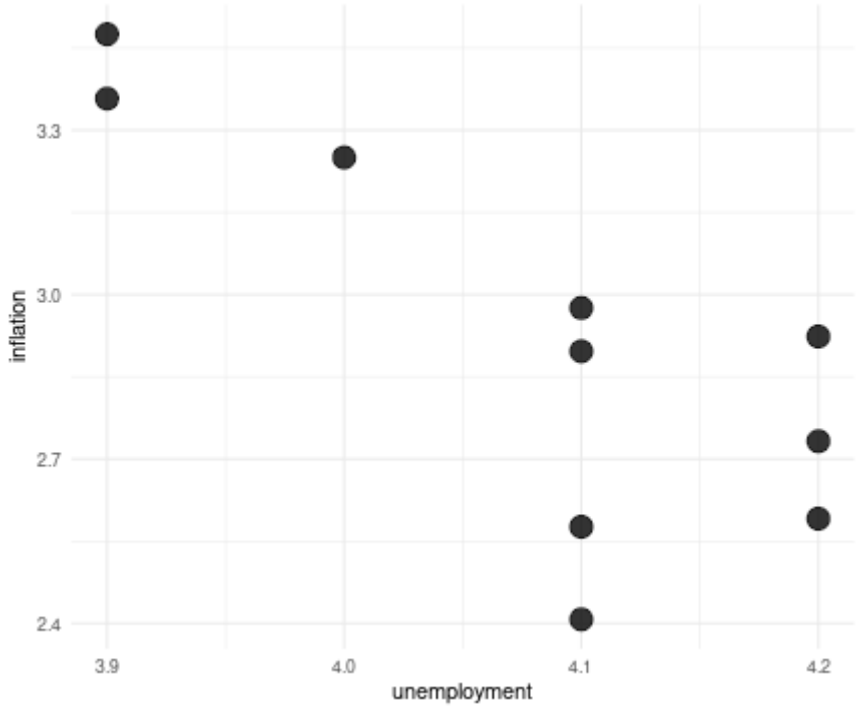
```
# Compute the posterior distribution using Bayes' r
posterior ← (df$prior * df$likelihood) /
  marginal_probability

df ← df %>% mutate(posterior = posterior)
```



# Putting it to use: OLS vs Bayesian

How close are these models with 10 data points?

| date | unemployment | inflation |
|---|---|---|
| 2024-03-01 | 3.9 | 3.475131 |
| 2024-04-01 | 3.9 | 3.357731 |
| 2024-05-01 | 4.0 | 3.250210 |
| 2024-06-01 | 4.1 | 2.975629 |
| 2024-07-01 | 4.2 | 2.923566 |
| 2024-08-01 | 4.2 | 2.591227 |
| 2024-09-01 | 4.1 | 2.407513 |
| 2024-10-01 | 4.1 | 2.576326 |
| 2024-11-01 | 4.2 | 2.732579 |
| 2024-12-01 | 4.1 | 2.896593 |



# Putting it to use: OLS vs Bayesian

How close are these models with 10 data points?

```
lm(inflation ~ unemployment -1, data = tail(economic_data, 10))
```

```
##
## Call:
## lm(formula = inflation ~ unemployment - 1, data = tail(economic_data,
##     10))
##
## Coefficients:
## unemployment
##       0.7132
```

# Putting it to use: OLS vs Bayesian

How close are these models with 10 data points?

We compute the likelihood for each observation and, assuming independent observations, multiply these together to get the combined likelihood:

$$p(\beta_1 \mid \text{data}) \propto p(\beta_1) \times \prod_i p(\text{inflation}_i \mid \beta_1, \text{unemployment}_i)$$

Where $p(\beta_1 \mid \text{data})$ is the posterior, $p(\beta_1)$ is the prior and $p(\text{inflation}_i \mid \beta_1, \text{unemployment}_i)$ is the likelihood. The product notation $\prod_i$ indicates that we multiply the likelihoods across all data points $i$.

- Did you spot the $\propto$ ? In reality, the denominator $p(D)$ is hard to compute as its the integral over all values. So instead of calculating this directly, we work with the "proportionality" and later normalize if needed.
- We actually approximate the "Posterior" using Markov Chain Monte Carlo (MCMC). More on that later.

Can we get 0.7132 using just basic R and our new fancy method?

# Putting it to use: OLS vs Bayesian

First we calculate all the components:

```
df ← tail(economic_data, 10)
beta_grid ← seq(-1, 1, length.out = 101)
prior ← dnorm(beta_grid, mean = 0, sd = 1)
prior ← prior / sum(prior)

likelihood_function ← function(b1, df){prod(dnorm(df$inflation, mean = b1 * df$unemployment, sd = 1))}
likelihood ← map_dbl(beta_grid, likelihood_function, df = df)
posterior ← (prior * likelihood) / sum(prior * likelihood)
```

The posterior mean: $\sum \beta_1 \times p(\beta_1 \mid \text{data})$, is computed as the Bayesian estimate of $\beta_1$.

```
bayes_estimate ← sum(beta_grid * posterior)
cat("Bayesian estimate (posterior mean) for beta1:", bayes_estimate, "\n")
```

```
## Bayesian estimate (posterior mean) for beta1: 0.7088872
```

Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

# Markov Chain Monte Carlo

# Understanding Markov Chain Monte Carlo

Metropolis and Ulam developed the Metropolis algorithm in the 1940s while working in chemistry and physics. Their clever algorithm was designed to approximate integrals for problems in thermodynamics, specifically in statistical mechanics. The primary goal was to solve problems involving marginal probabilities (a central concept in Bayesian statistics).

It would take about 40 years before statisticians discovered the Metropolis paper (under a somewhat obscure name) and started realizing its potential applications in statistical sampling.

- **1940s:** Metropolis and Ulam develop the Metropolis algorithm in physics to approximate integrals in thermodynamics.
- **40 years later:** Statisticians discover the algorithm, but early computers lack the power to fully utilize it.
- **Late 1980s-1990s:** Advances in computational power and statistical theory lead to the creation of BUGS, a key software for implementing MCMC methods.
- **Result:** The right combination of factors — theory, technology, and software — make MCMC a practical tool for statistics.

https://www.youtube.com/watch?v=072Q18nX91I

# Understanding (Markov Chain) Monte Carlo

We define a two-state Markov chain with states "Sunny" and "Rainy" (imagine its not Cape Town). The key takeaway here is that the probability of tomorrow's weather is dependent on today's weather (not on the sequence of events that preceded it):

**Step 1**: Define the States and Transition Matrix

```
states ← c("Sunny", "Rainy")
# Define the transition matrix (rows = current stat
trans_matrix ← matrix(c(0.8, 0.2,  # From Sunny: 0
                        0.3, 0.7), # From Rainy: 0
             nrow = 2, byrow = TRUE)

rownames(trans_matrix) ← states
colnames(trans_matrix) ← states
trans_matrix
```

*Cool little visualisation: https://willhipson.netlify.app/post/markov-sim/markov_chain/*

**Step 2**: Simulate the chain

```
simulate_markov_chain ← function(trans_matrix, start_sta
  chain ← character(n_steps)
  chain[1] ← start_state

  # Sample the next state based on the current state
  for (i in 2:n_steps) {
    current_state ← chain[i - 1]
    chain[i] ← sample(states, size = 1, prob = trans_mat
  }

  return(chain)
}

set.seed(123)
simulate_markov_chain(trans_matrix, start_state = "Sunny"
```

# Understanding Markov Chain (Monte Carlo)

Monte Carlo is a method where we approximate an outcome1 and hopefully also the distribution thereof.

> Imagine a portfolio of 100 loans (or debts) where each loan has a fixed probability of default (5%, p = 0.05).

**Goal:** Simulate many 10,000 scenarios to see, on average, how many defaults occur in the portfolio.

For each simulation:

- For each of the 100 loans, decide whether it defaults.
- Count the total defaults in that simulation.
- Repeat the process 10,000 times.
- Estimate the expected number of defaults by taking the average of the simulated default counts.

# Understanding Markov Chain (Monte Carlo)

In order to conduct this simulation we use our friend the Bernoulli trial (aka coin flip):

```
set.seed(123)
n_simulations ← 10000

f ← function(x) {
  n_loans ← 100
  sum(rbinom(n_loans, size = 1, prob = 0.05))
}

# Run the Monte Carlo simulation
defaults ← map_dbl(1:n_simulations, f)

expected_defaults ← mean(defaults)
cat("Expected number of defaults:", expected_defaults, "\n")
head(defaults)
```

# Understanding Markov Chain Monte Carlo!

We are finally there 😮💨. Suppose you have a portfolio of 100 loans and you observe 5 defaults.

Under a simple model with a uniform prior for $p$ (i.e. a Beta(1,1) prior), the likelihood is given by the Binomial probability. With a uniform prior, the (unnormalized) posterior for pp is proportional to:

$$p^5(1-p)^{95}$$

This is actually very convenient, setup because the Beta distribution is **conjugate** to the Binomial likelihood.

> A prior and likelihood are conjugate if the resulting posterior distribution is of the same family as the prior.

Instead of integrating to find the posterior, we can just update the parameters directly:

$$\underbrace{\text{Beta}(\alpha, \beta)}_{\text{Prior}} + \underbrace{\text{Binomial}(x, n)}_{\text{Likelihood}} \Rightarrow \underbrace{\text{Beta}(\alpha + x, \beta + (n - x))}_{\text{Posterior}}$$

Conjugacy is great when available, but for real-world Bayesian modeling, we often must use MCMC or other numerical methods to estimate posterior distributions.

# For my own sanity (background math)

$P(\beta_1)$: A uniform prior on $p$, meaning $P(p)$ is constant. $P(D|\beta_1)$: A Bernoulli likelihood based on observed defaults.

**The posterior is:** $P(p \mid D) \propto P(D \mid p)P(p)$

If we observe $k$ defaults out of $n$ loans, the likelihood follows a Binomial distribution:

$$P(D \mid p) = p^k(1-p)^{n-k}$$

A uniform prior on $p$ means:

$$P(p) = \text{constant}$$

By Bayes' theorem:

$$P(p \mid D) \propto P(D \mid p)P(p)$$

Since $P(p)$ is constant:

$$P(p \mid D) \propto p^k(1-p)^{n-k}$$

# Markov Chain Monte Carlo: Metropolis algorithm

Since $P(D)$ does not depend on $\beta_1$, MCMC methods sample from the unnormalized posterior:

$$\underbrace{P(\beta_1 \mid D)}_{\text{Posterior}} \propto \underbrace{P(D \mid \beta_1)}_{\text{Likelihood}} \times \underbrace{P(\beta_1)}_{\text{Prior}}$$

So lets build a little MCMC sampler it for our default example, where we have 100 loans and observed 5 defaults where:

```
posterior_density ← function(p) {
  n_loans ← 100
  observed_defaults ← 5

  if (p < 0 || p > 1) return(0)
  return(p^observed_defaults * (1 - p)^(n_loans - o
}
```

```
metropolis_sampler_default ← function(initial, iterations, proposal_sd) {
  chain ← numeric(iterations)
  chain[1] ← initial

  for (i in 2:iterations) {
    current ← chain[i - 1]
    # Propose a new candidate by adding a small random jump
    candidate ← current + rnorm(1, mean = 0, sd = proposal_sd)

    # If candidate is outside [0,1], reject it immediately
    if (candidate < 0 || candidate > 1) {
      chain[i] ← current
      next
    }

    # Compute the acceptance ratio using the unnormalized posterior densitie
    ratio ← posterior_density(candidate) / posterior_density(current)

    # Accept the candidate with probability min(1, ratio)
    if (runif(1) < ratio) {
      chain[i] ← candidate
    } else {
      chain[i] ← current
    }
  }

  return(chain)
}
```

# Markov Chain Monte Carlo: Metropolis algorithm

Starting with $X^{(0)} := (X_1^{(0)}, \ldots, X_p^{(0)})$ iterate for $t = 1, 2, \ldots, u \sim U(0, 1)$.

1) Draw $X \sim q(\cdot \mid X^{(t-1)})$

2) Compute

$$\alpha(X \mid X^{(t-1)}) = \min \left\{ 1, \frac{f(X) \cdot q(X^{(t-1)} \mid X)}{f(X^{(t-1)}) \cdot q(X \mid X^{(t-1)})} \right\}$$

3) If $\alpha(X > u$, set $X^{(t)} = X$, otherwise set $X^{(t)} = X^{(t-1)}$.

| Symbol | Meaning |
|---|---|
| $X$ | Candidate state (proposed sample) |
| $X^{(t-1)}$ | Current state in the Markov chain |
| $f(X)$ | **Target distribution** (the desired posterior or density) |
| $q(X \mid X^{(t-1)})$ | **Proposal distribution** (used to generate candidate samples) |
| $\alpha(X \mid X^{(t-1)})$ | **Acceptance probability** |

# Markov Chain Monte Carlo: Metropolis algorithm

```
chain_defaults ← metropolis_sampler_default(initial = 0.15,
                                            iterations = 10000,
                                            proposal_sd = 0.02)

posterior_mean ← mean(chain_defaults)
posterior_sd   ← sd(chain_defaults)

cat("Posterior mean estimate for p:", posterior_mean, "\n")
```

```
## Posterior mean estimate for p: 0.05947809
```

```
cat("Posterior standard deviation for p:", posterior_sd, "\n")
```

```
## Posterior standard deviation for p: 0.0236256
```

# Setting Reasonable Priors Without Prior Knowledge

1. **Use a Non-Informative (or Weakly Informative) Prior**

   - If you genuinely have no prior knowledge, choose a **non-informative prior** that spreads probability evenly across all possible values.
   - Example: A **uniform prior** (e.g., $\beta \sim Uniform(-10, 10)$) assumes all values are equally likely.
   - Alternatively, use a **weakly informative prior** (e.g., $\beta \sim Normal(0, 5)$) that assumes small effects are more likely than extreme ones.

2. **Leverage Historical Data or Expert Opinions**

   - If past studies suggest inflation and GDP have a small but negative relationship, you could use $\beta \sim Normal(-0.5, 1)$.
   - If expert opinions exist, incorporate them into your prior.

3. **Use Empirical Priors (Data-Driven Approach)**

   - Collect past inflation and GDP data and estimate a simple regression model.
   - Use the estimates from this model as a **starting point** for your Bayesian priors.
   - Example: If historical regression gives $\beta \approx -0.3$, set $\beta \sim Normal(-0.3, 0.2)$.

# Setting Reasonable Priors Without Prior Knowledge

1. **Perform Sensitivity Analysis**
   - Try different priors and check how much they affect the results.
   - If results change drastically with different priors, that means your prior matters a lot, and you may need more data or expert input.

2. **Default to a Broad, Centered Prior**
   - If in doubt, a **normal prior centered around zero** is often reasonable (e.g., $\beta \sim Normal(0, 10)$).
   - This allows for both positive and negative relationships without being too restrictive.

Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

# Practical

# Bayesian Estimation of Philips Curve

> The Phillips Curve describes the inverse relationship between inflation and unemployment

We assume a simple linear relationship between **inflation** ( $\pi_t$ ) and **unemployment** ( $u_t$ ):

$$\pi_t = \alpha + \beta u_t + \epsilon_t$$

where:

- $\pi_t$ = Inflation rate at time $t$
- $u_t$ = Unemployment rate at time $t$
- $\alpha$ = Intercept (inflation when unemployment is zero)
- $\beta$ = Slope (effect of unemployment on inflation, in most cases negative relationship)
- $\epsilon_t \sim N(0, \sigma^2)$ = Random error term, assumed to be normally distributed with mean 0 and variance $\sigma^2$

This model suggests that inflation is influenced by unemployment, where $\beta$ represents the **Phillips Curve effect**. In most situations $\beta < 0$ suggests an inverse relationship.

# Time for estimation: OLS

Lets start with the OLS estimation:

```
lm_model ← lm(inflation ~ unemployment, data = economic_data)
summary(lm_model)
```

```
##
## Call:
## lm(formula = inflation ~ unemployment, data = economic_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6151 -1.8978 -0.6297  0.9773 11.0083
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.04248    0.32981   9.225   <2e-16 ***
## unemployment  0.08595    0.05557   1.547    0.122
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.885 on 922 degrees of freedom
## Multiple R-squared:  0.002588,    Adjusted R-squared:  0.001506
## F-statistic: 2.392 on 1 and 922 DF,  p-value: 0.1223
```

```
lm_model_2000 ← lm(inflation ~ unemployment, data = economic_data %>% filte
summary(lm_model)
```

```
##
## Call:
## lm(formula = inflation ~ unemployment, data = economic_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6151 -1.8978 -0.6297  0.9773 11.0083
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.04248    0.32981   9.225   <2e-16 ***
## unemployment  0.08595    0.05557   1.547    0.122
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.885 on 922 degrees of freedom
## Multiple R-squared:  0.002588,    Adjusted R-squared:  0.001506
## F-statistic: 2.392 on 1 and 922 DF,  p-value: 0.1223
```

# Time for estimation: OLS

# Time for estimation: Bayesian

```
library(tidyverse)
library(bertheme) # Used for BER style Plots
library(brms)
library(bayesplot)
```

Our model specification utilises weak priors:

$$\text{Intercept} \sim \text{N}(3, 2) \text{Slope (b)} \sim \text{N}(-0.5, 0.5) \text{Sigma} \sim \text{Cauchy}(0, 2)$$

In R the we do this by setting:

```
# Weakly Informative Priors
prior_weak ← c(
  prior(normal(3, 2), class = "Intercept"),
  prior(normal(-0.5, 0.5), class = "b"),
  prior(cauchy(0, 2), class = "sigma")
)
```

# Time for estimation: Bayesian

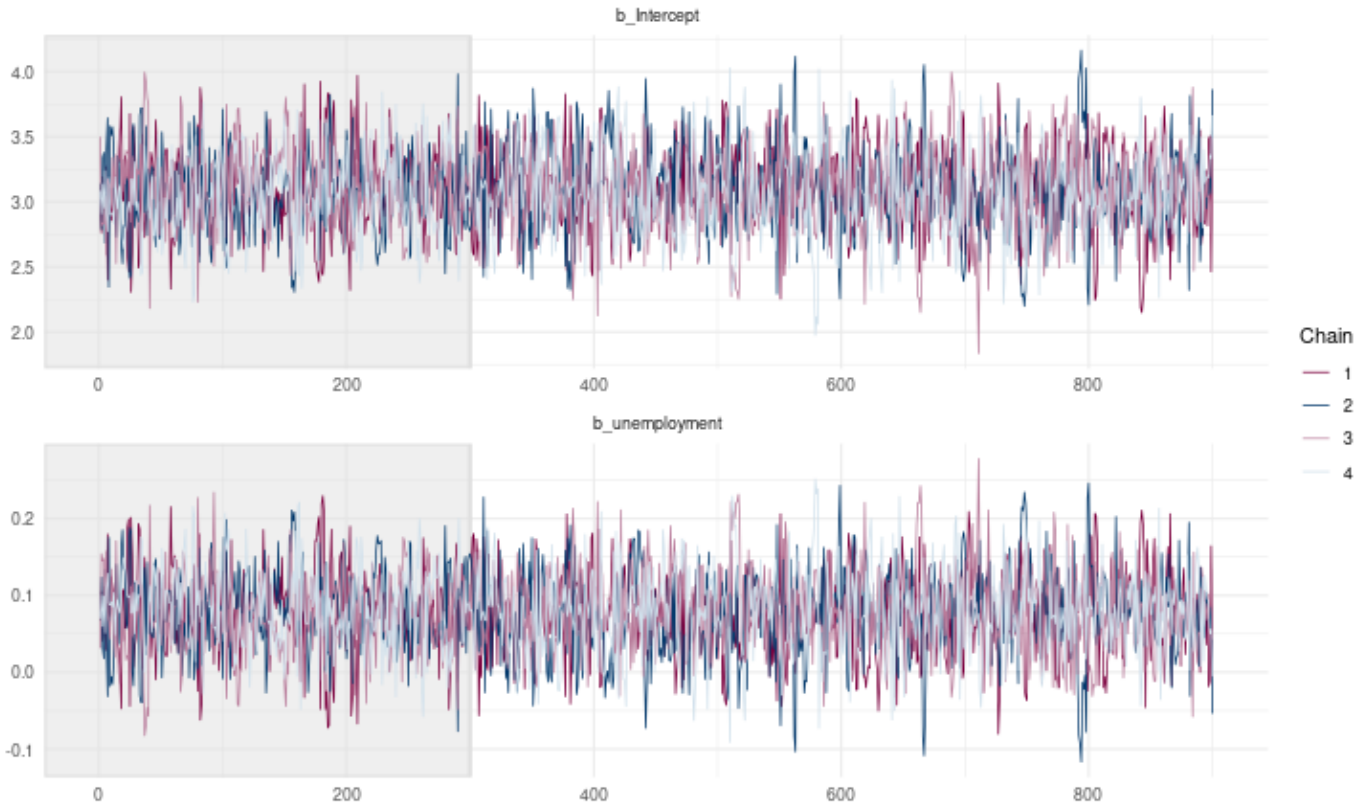
Next, lets estimate the model!

```
prior_weak ← c(
  prior(normal(3, 2), class = "Intercept"),
  prior(normal(-0.5, 0.5), class = "b"),
  prior(cauchy(0, 2), class = "sigma")
)

model_weak ← brm(inflation ~ unemployment, data = economic_data,
                 family = gaussian(),
                 prior = prior_weak,
                 iter = 100, # change in real-application
                 warmup = 50, # change in real-application
                 chains = 4,
                 cores = 4)
```

# Time for estimation: Bayesian

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: inflation ~ unemployment
##    Data: economic_data (Number of observations: 924)
##   Draws: 4 chains, each with iter = 1000; warmup = 100; thin = 1;
##          total post-warmup draws = 3600
##
## Population-Level Effects:
##              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept        3.08      0.32     2.45     3.71 1.00     2036     1602
## unemployment     0.08      0.06    -0.03     0.19 1.00     2007     1612
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     2.89      0.07     2.76     3.02 1.00     4633     2891
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

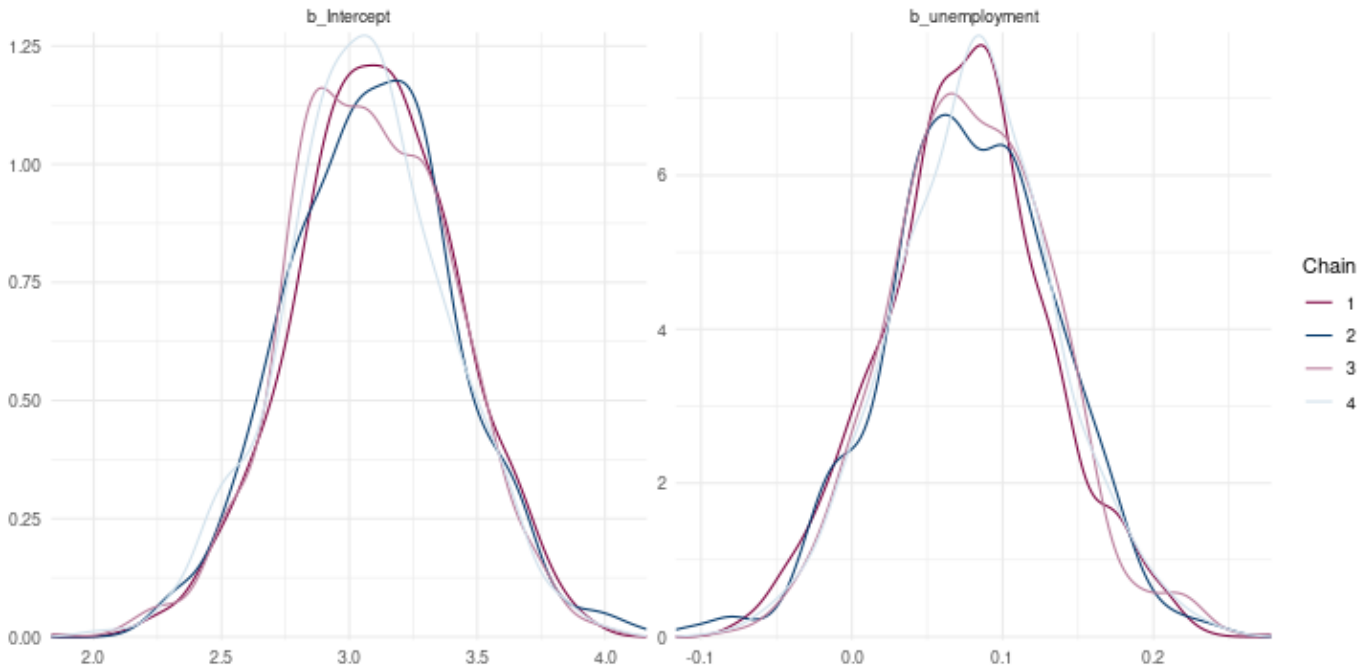
# Time for estimation: Bayesian




```
color_scheme_set("mix-blue-pink") #bayesplot
parameters ← c("b_Intercept", "b_unemploymen
mcmc_trace(
  model_weak,
  pars = parameters,
  n_warmup = 300,
  facet_args = list(nrow = 2,
                    labeller = label_parsed)
) +
  facet_text(size = 15) +
  theme_minimal()
```

# Posterior density plot

The shape of the posterior distributions provides valuable insight into whether the MCMC chains have properly converged. If the distributions align well, it suggests stability. However, if we observe a bimodal distribution (resembling a camel's ridge with two peaks), this may indicate issues with model specification, such as poor priors, identifiability problems, or inadequate mixing of the chains.




```
mcmc_dens_overlay(model_weak, pars = paramete
  facet_text(size = 15) +
  theme_minimal()
```

# Gelman-Rubin diagnostics

The Gelman-Rubin diagnostic, also known as $\hat{R}$ (R-hat) or the potential scale reduction factor, is a key metric for assessing MCMC chain convergence. It evaluates whether independently initialized chains have converged to the same posterior distribution.

The $\hat{R}$ statistic is computed as the ratio of the average variance within each chain to the variance of the pooled samples across all chains.

$$\hat{R} = \sqrt{\frac{\frac{1}{M}\sum_{m=1}^{M} s_m^2}{s_{\text{pooled}}^2}}$$

$M$ is the number of chains, $s_m^2$ is the variance within chain $m$, $s_{\text{pooled}}^2$ is the variance of the pooled samples across all chains.

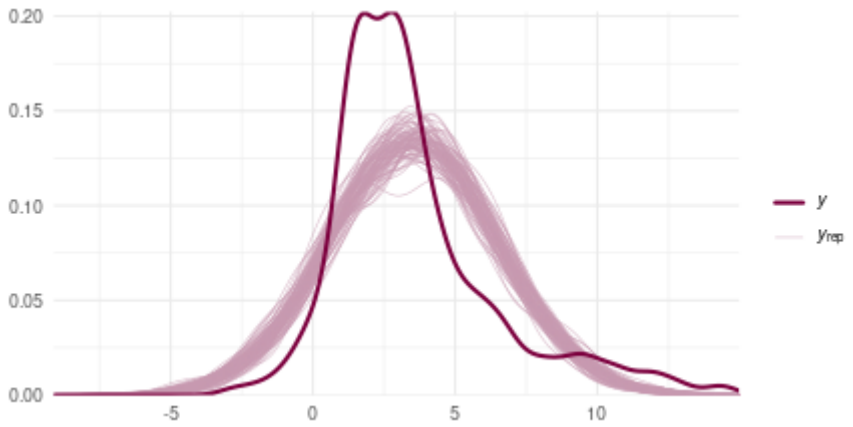
When the chains have fully converged, this ratio approaches 1. However, if $\hat{R} > 1$, it suggests that the chains have not yet stabilized and further sampling may be needed.

# Model fit

The posterior predictive check (PPC) is a valuable tool for assessing how well a model fits the observed data. The core idea behind PPCs is that if a model provides a good fit, the predicted values generated from the model should closely resemble the actual data used for fitting.

To perform a PPC, we simulate multiple draws from the posterior predictive distribution—the distribution of the response variable given the posterior estimates of the model parameters. By comparing these simulated values to the observed data, we can visually and statistically evaluate potential discrepancies, helping to identify issues such as model misspecification or unaccounted

```
pp_check(model_weak, ndraws = 100)
```

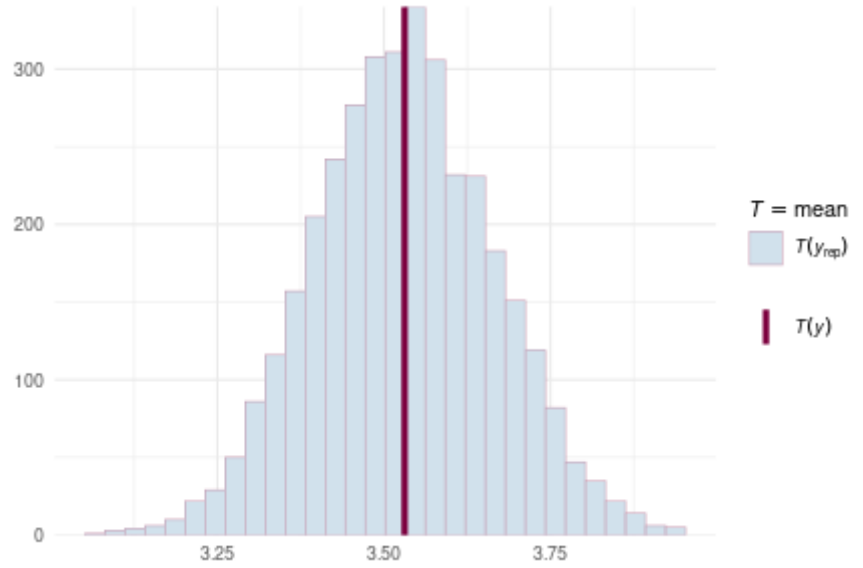


# Posterior prediction

In Bayesian statistics, the posterior prediction refers to the process of predicting new or future data, given the observed data and the model parameters:

$$p(y_{\text{new}} \mid \text{data}) = \int p(y_{\text{new}} \mid \theta) p(\theta \mid \text{data}) \, d\theta$$

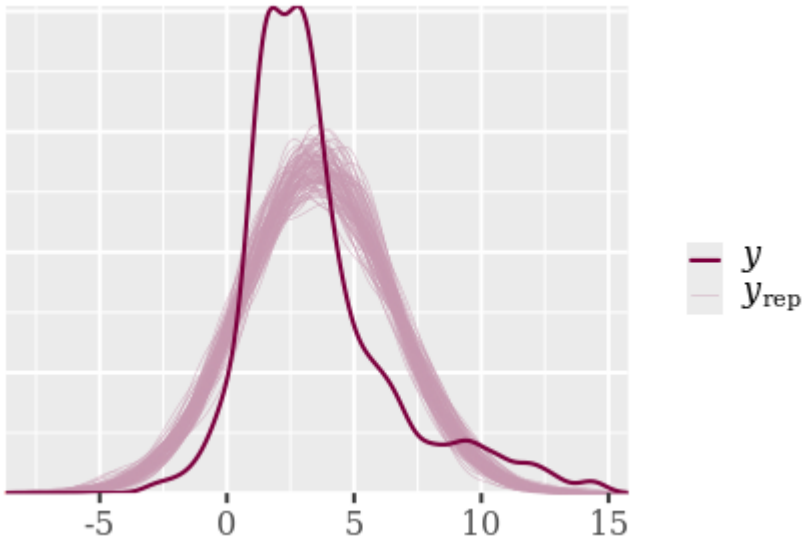
```
yrep ← model_weak %>%
  posterior_predict(draws = 500)

ppc_stat(y = model_weak$data$inflation,
         yrep = yrep) +
         facet_text(size = 15) +
         theme_minimal()
```



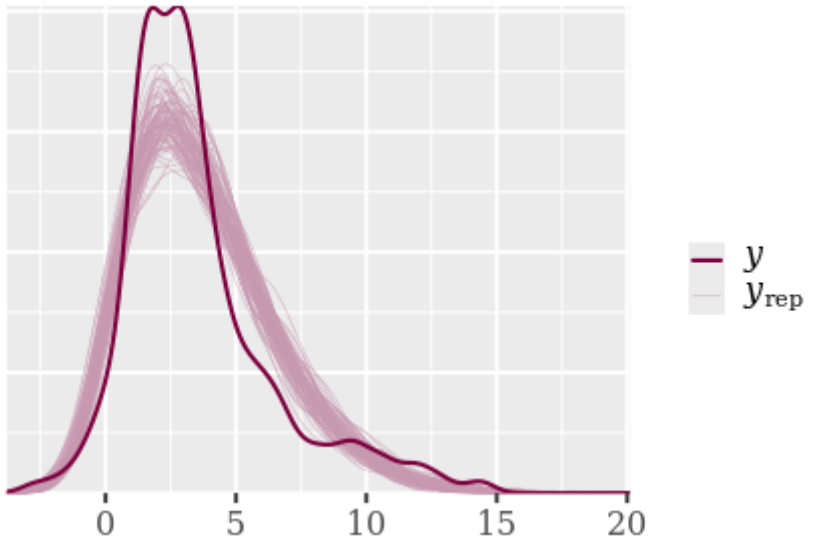
# Better model specification?

```
model_weak_skew ← brm(inflation ~ unemployment, data = economic_data,
                  family = skew_normal(),
                  prior = prior_weak, iter = 1000, warmup = 50, chains = 4, cores = 4)
```

```
pp_check(model_weak, ndraws = 100)
```




```
pp_check(model_weak_skew, ndraws = 100)
```



# Better model specification?

The posterior predictive checks (PPCs) already give us insight into which model fits the data better. In addition, brms provides several functions for comparing different models. One useful function is `add_criterion()`, which allows you to add model fit criteria to the model objects for a more detailed comparison.

To determine which model is better, we need to look at the elpd_loo (expected log pointwise predictive density) and looic (leave-one-out cross-validation information criterion): (elpd_loo = *higher* values are better, looic = *lower* values are better)

```
model_weak ← add_criterion(model_weak, c("loo", "waic"))
model_weak_skew ← add_criterion(model_weak_skew, c("loo", "waic"))

comp_loo ← loo_compare(model_weak, model_weak_skew,
                       criterion = "loo")
print(comp_loo, simplify = FALSE, digits = 3)
```

```
##                 elpd_diff se_diff   elpd_loo  se_elpd_loo p_loo     se_p_loo  looic     se_looic
## model_weak_skew     0.000     0.000 -2188.830    28.953       5.244     0.762  4377.660    57.905
## model_weak       -104.004    11.572 -2292.834    30.229       4.119     0.397  4585.669    60.457
```

# Choosing Priors: Three approaches

❓ How does the prior impact the estimation?

Lets start off by saying: "I do not have strong prior knowledge, I will use **weakly informative priors**":

- **Intercept ( $\alpha$ ):** Inflation varies significantly, but we expect it to be around 2-5% on average.
  **Prior:** $\alpha \sim N(3, 2)$
- **Slope ($\beta$):** Economic theory suggests $\beta$ should be negative (higher unemployment → lower inflation).
  **Prior:** $\beta \sim N(-0.5, 0.5)$, allowing for uncertainty.
- **Standard deviation ( $\sigma$ ):** Inflation fluctuations should be positive but not extreme.
  **Prior:** $\sigma \sim \text{Half-Cauchy}(0, 2)$, ensuring positive values.

These priors reflect our expectations while allowing flexibility for the data to update our beliefs.

What other kind of priors are available to us?

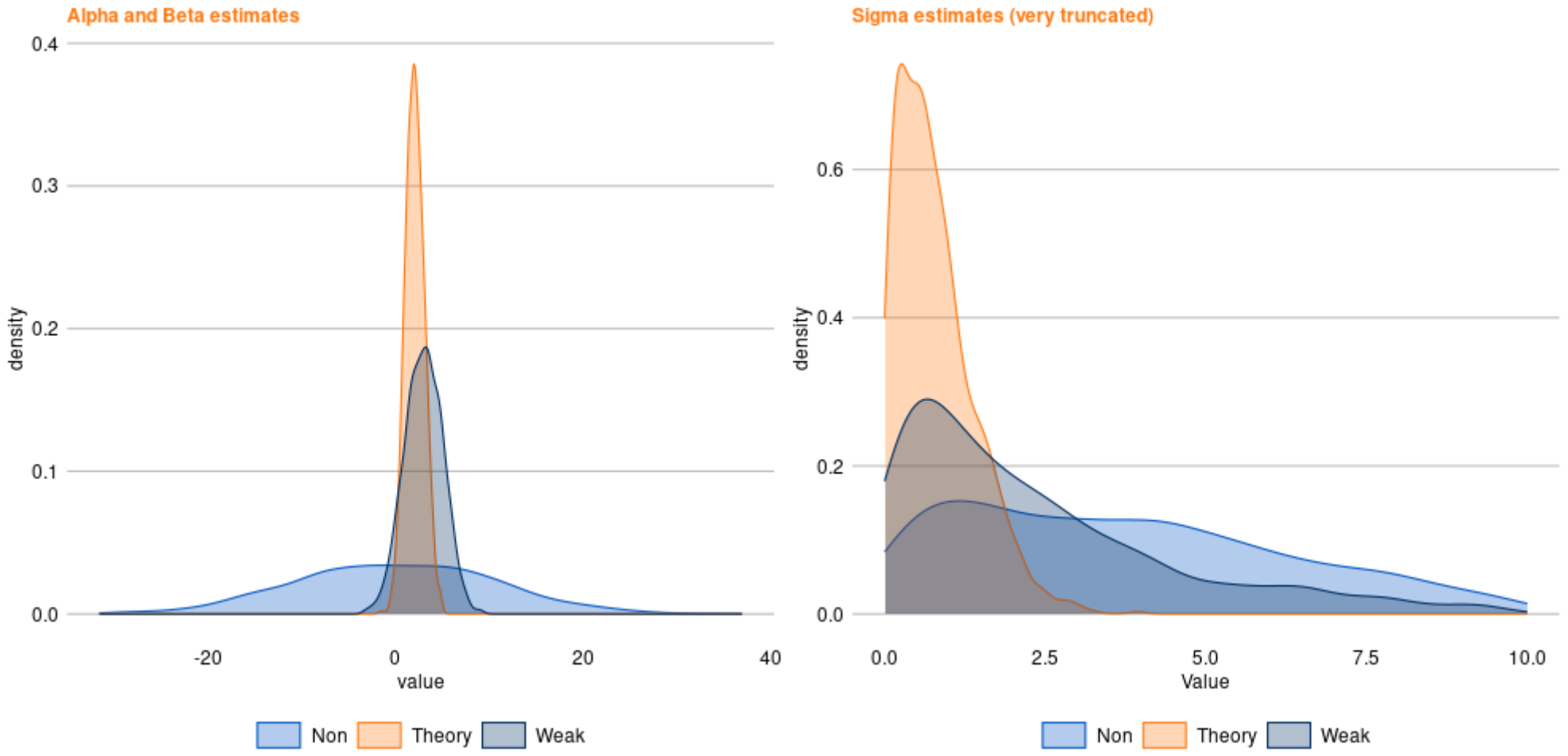
# Summary of priors

- **Weak**: Reflect our expectations while allowing flexibility for the data to update our beliefs.
- **Theory-Informed**: Impose stronger economic constraints, reducing extreme posterior estimates.
- **Non-Informative**: Let the data entirely dictate the relationship

Comparison of Prior Distributions for Bayesian Phillips Curve Model

| Parameter | WeaklyInformative | TheoryInformed | NonInformative |
|---|---|---|---|
| Intercept ( $\alpha$ ) | $N(3, 2)$ | $N(2, 1)$ | $N(0, 10)$ |
| Slope ( $\beta$ ) | $N(-0.5, 0.5)$ | $N(-0.3, 0.2)$ | $N(0, 5)$ |
| Standard Deviation ( $\sigma$) | Half-Cauchy(0, 2) | Half-Cauchy(0, 1) | Half-Cauchy(0, 5) |

# Visually, what have we done?



# Conclusion

- Frequentist vs. Bayesian: We explored the key differences, with Frequentists treating parameters as fixed and Bayesians updating beliefs using prior information.
- Bayesian Thinking: Parameters are random variables and probability represents our degree of belief rather than long-run frequencies.
- Markov Chain Monte Carlo (MCMC): We learned how MCMC methods, like the Metropolis algorithm, help sample from complex posterior distributions.
- Hands-on Intuition: Using simple analogies (e.g., the loan defaults), we saw how Bayesian updating and MCMC work in practice.
- BRMS for Practical Bayesian Modeling: We introduced BRMS as a powerful tool for implementing Bayesian regression models, making Bayesian inference more accessible and computationally efficient.

🌟 If you want to see the real power of `brms`, go to Andrew Heiss' website. The examples on hierachical modeling is especially of interest. Actually, just ALL of this blogs is AMAZING and a real life wizard in visualisation 🧙.