Week 3: Intro to Bayesian Inference

PLSC 40502 - Statistical Models

Review

Previously

- **Generalized Linear Models**
  - Extending our **linear predictor** $X_i'\beta$ to non-linear transformations of the CEF
  - **Three** elements: A distribution for Y_i, a linear predictor, and a link function $g()$: $g(E[Y_i|X_i]) = X_i'\beta$
  - Estimation via Maximum Likelihood - nicely-behaved likelihoods when Y_i is in the exponential family.
- **Robust** inference:
  - Can we still do inference on β in a GLM when the model is mis-specified?
  - In many cases $\hat{\beta}$ is consistent for β but the MLE is not efficient and its variance does not approach the C-R lower bound.
  - "Sandwich" estimator using the estimated gradient.

This week

- **Bayesian inference**
  - What happens when we define a distribution for our "beliefs" about θ
  - How do we obtain the **posterior** distribution $\theta|\mathbf{Y}$?
- **Defining a Bayesian Model**
  - Writing down the data-generating process
  - "Plate notation" for concise model summaries
- **Estimation** via computation
  - Most posterior distributions can't be expressed in closed form
  - But we can construct an algorithm to **sample** from that distribution
  - Metropolis-Hastings algorithm - "Gibbs sampling" as a special case

Intro to Bayesian Inference

Bayesian inference

- In the **frequentist** paradigm, we took a particular approach to thinking about *randomness* when trying to learn about an unknown parameter θ using observed data $\mathbf{Y}$
  - θ is an unknown **constant**
  - $\mathbf{Y}$ are random variables
  - $\hat{\theta}$ is our estimator of θ - it's a function of random random variables
  - Probabilities reflect behavior in *repeated samples*
- **Bayesian** inference takes a different approach to the problem. Rather than treating θ as a constant, we consider it to be **random** as well!
  - Probabilities reflect **beliefs** about a particular quantity
  - $f(\theta)$ denotes our *prior* beliefs about the value of θ
  - $f(\theta|\mathbf{Y})$ is the *posterior* distribution given the observed data. This is our target of inference
  - $\mathbf{Y}$ are still random variables...but the posterior distribution *conditions* on them
- We derive the posterior using **Bayes rule**

$$f(\theta|\mathbf{Y}) = \frac{f(\mathbf{Y}|\theta)}{f(\mathbf{Y})} f(\theta)$$

Bayesian inference

- The denominator can be written as an integral over all of the possible values of θ (marginalizing over θ)

$$f(\mathbf{Y}) = \int f(\mathbf{Y}, \theta) d\theta = \int f(\mathbf{Y}|\theta) f(\theta) d\theta$$

- So often we'll write the posterior distribution *up to a proportionality constant* as

$$\underbrace{f(\theta|\mathbf{Y})}_{\text{posterior}} \propto \underbrace{f(\mathbf{Y}|\theta)}_{\text{likelihood}} \times \underbrace{f(\theta)}_{\text{prior}}$$

- In addition to the posterior distribution of the parameters θ, we'll often want to generate "forecasts" of an out-of-sample $\tilde{Y}$ conditional on what we have observed
  - This involves integrating over the posterior values of the parameter θ

$$\begin{aligned} f(\tilde{Y}|\mathbf{Y}) &= \int f(\tilde{Y}, \theta|\mathbf{Y}) d\theta \\ &= \int f(\tilde{Y}|\theta, \mathbf{Y}) f(\theta|\mathbf{Y}) d\theta \\ &= \int f(\tilde{Y}|\theta) f(\theta|\mathbf{Y}) d\theta \end{aligned}$$

Bayesian inference

- We also might reason about two different values of θ: $\{\theta_1, \theta_2\}$ and their relative odds. This has a nice expression in terms of the *prior* times a *likelihood ratio*

$$\frac{f(\theta_1|\mathbf{Y})}{f(\theta_2|\mathbf{Y})} = \frac{f(\theta_1)}{f(\theta_2)} \times \frac{f(\mathbf{Y}|\theta_1)}{f(\mathbf{Y}|\theta_2)}$$

- Inference on the posterior is often referred to as **updating** the prior and often can be conceptualized in terms of *sequential* changes to our beliefs about the parameter.
- Suppose we have two samples $\mathbf{Y}_1$ and $\mathbf{Y}_2$. Our posterior can be written as:

$$\begin{aligned} f(\theta|\mathbf{Y}_1, \mathbf{Y}_2) &\propto f(\mathbf{Y}_2, \mathbf{Y}_1|\theta) \times f(\theta) \\ &\propto f(\mathbf{Y}_2|\mathbf{Y}_1, \theta) \times f(\mathbf{Y}_1|\theta) \times f(\theta) \\ &\propto f(\mathbf{Y}_2|\mathbf{Y}_1, \theta) \times f(\theta|\mathbf{Y}_1) \end{aligned}$$

- The posterior becomes the new "prior"!

Application: Predicting Elections

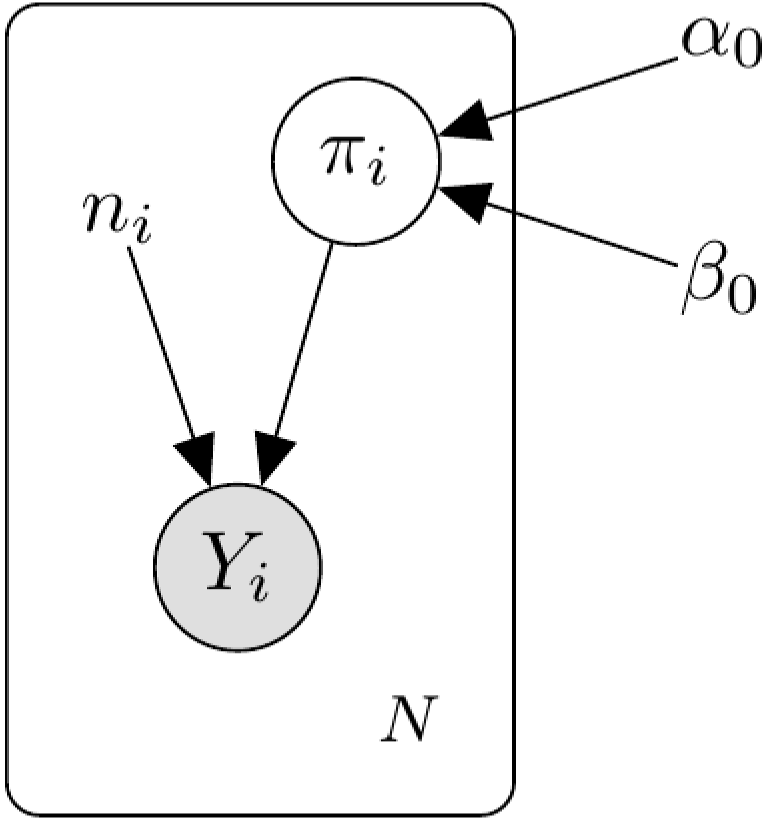
- We'd like to predict U.S. House of Representatives elections using data from the prior Presidential election at the county level
  - *Pettigrew (2018)* assembled the 2018 US House returns at the county level
  - Available as `us-house-wide.csv` on the Course Github
- For each county i, $i \in \{1, 2, \ldots, N\}$, we observe:
  - Y_i: The number of votes for the Democratic candidate county
  - n_i: The number of votes cast in total in that county
- Let's start by building a simple model and going from there.
  - Assume Y_i is a draw from a binomial distribution with a total number of of trials K_i and "success" probability π.
  - π_i comes from a **Beta** distribution with *hyperparameters* α_0 and β_0

$$Y_i \sim \text{Binomial}(n_i, \pi_i)$$

$$\pi_i \sim \text{Beta}(\alpha_0, \beta_0)$$

Plate notation

- A common method of writing statistical models is via **plate notation**
  - These concisely encode independence and dependence assumptions across parameters and data.



Features of a Bayesian model

- There are four types of variables in a Bayesian model
  - **Observed Data**: Variables that have a probability distribution but on whose observed values we condition - Y_i
  - **Known Constants**: Fixed quantities that do not have probability distributions (e.g. regressors or other features of the nodes) - n_i, N, α_0, β_0
  - **Deterministic quantities**: Transformations of other variables
  - **Latent parameters**: Variables that have a probability distribution that we do not observe - π_i
- Sometimes the known constants are *actually* known by us (e.g. N or n_i) and in other cases they are *assumed to be known*
  - These are typically parameters of the prior distribution which are called **hyperparameters** (here: α_0 and β_0)
  - The hyperparameters govern the distribution of the prior -- crucially, its mean and variance.
- In Bayesian inference, we are interested in obtaining either the posterior of the latent parameters *or* integrating them out (in the latter case, they are sometimes referred to as "nuisance" parameters)
  - Here, we want to obtain $f(\pi_i|\mathbf{Y})$
  - When writing the posterior, we'll often omit the implicit conditioning on the observed constants.

The Likelihood

- Remember the posterior distribution is proportional to the **likelihood** times the **prior**

$$\underbrace{f(\pi_i|\mathbf{Y})}_{\text{posterior}} \propto \underbrace{f(\mathbf{Y}|\pi_i)}_{\text{likelihood}} \times \underbrace{f(\pi_i)}_{\text{prior}}$$

- First, let's derive the **likelihood** $f(\mathbf{Y}|\pi_i)$
  - Since the Y_i are independent conditional on π_i, we can write:

$$f(\mathbf{Y}|\pi_i) = \prod_{j=1}^{N} f(Y_j|\pi_i)$$

- Next, for $j \neq i$, we have $f(Y_j|\pi_i) = f(Y_j)$ since Y_j only depends on π_j. That's just a constant term and drops out of the posterior distribution, leaving

$$f(\pi_i|\mathbf{Y}) = f(\pi_i|Y_i) \propto f(Y_i|\pi_i)f(\pi_i)$$

The Likelihood and Prior

- What's $f(Y_i|\pi_i)$? We've defined it as the binomial PMF

$$f(Y_i|\pi_i) = \binom{n_i}{Y_i}\pi_i^{Y_i}(1-\pi_i)^{n_i-Y_i}$$

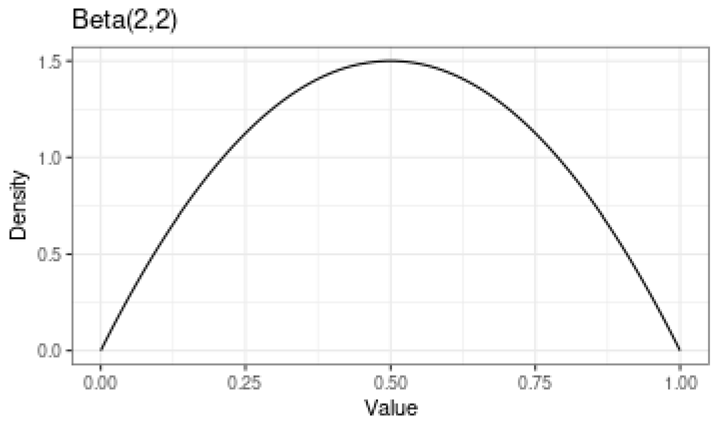
- What about the prior? $f(\pi_i)$? We've chosen the **beta** distribution

$$f(\pi_i) = \frac{\Gamma(\alpha_0)\Gamma(\beta_0)}{\Gamma(\alpha_0+\beta_0)}\pi_i^{\alpha_0-1}(1-\pi_i)^{\beta_0-1}$$

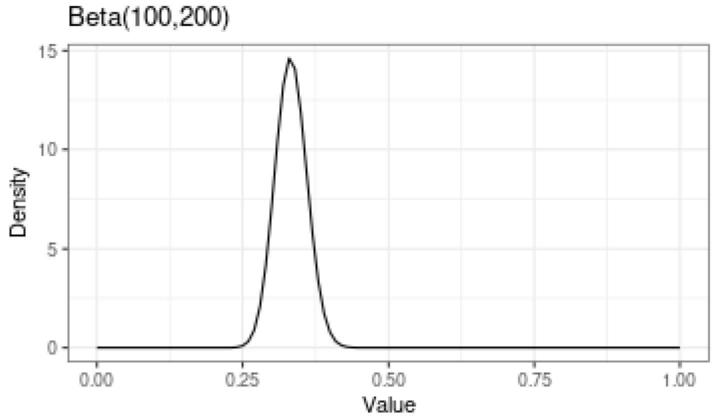
- The Beta distribution has some notable features.
  - It's mean: $E[\pi_i] = \frac{\alpha_0}{\alpha_0+\beta_0}$
  - It's variance $Var(\pi_i) = \frac{\alpha_0\beta_0}{(\alpha_0+\beta_0)^2(\alpha+\beta+1)}$

The Beta Distribution

- Let's see how the parameters influence the shape of the beta. $\mathrm{Beta}(2, 2)$:



- $\mathrm{Beta}(100, 200)$:



Choosing a prior

- There are two ways to think about choosing a prior distribution
  - **Informative** prior - Use the prior to encode our existing beliefs about the parameter
  - **Uninformative/Diffuse** prior - Pick a prior that will have the least impact on the posterior distribution
- We also need to consider the *shape* of the prior distribution $f(\pi_i)$
  - Why did we pick the Beta distribution?
  - Because it has a special property when combined with a binomial likelihood. It is a **conjugate** prior to the binomial likelihood.
- **Conjugate prior**: A prior distribution is *conjugate* to a particular likelihood if the posterior distribution is of the same form as the prior
  - $f(\theta)$ is beta; $f(\mathbf{Y}|\theta)$ is binomial; $f(\theta|\mathbf{Y})$ is beta

Choosing a prior

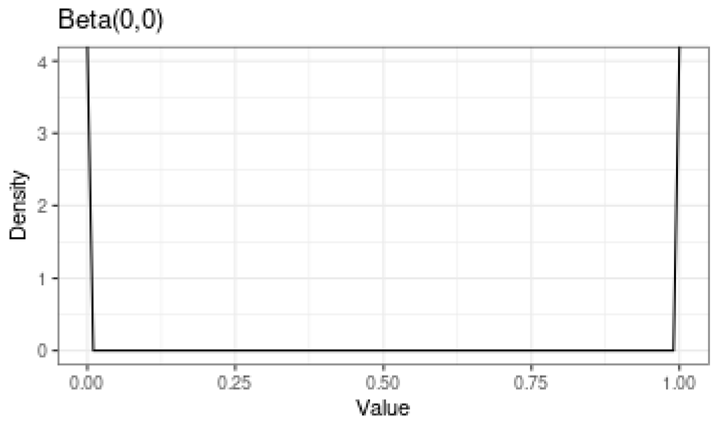
- In this case, π_i must be between 0 and 1, so we already have some information.
- One possible choice for an *uninformative* prior is the uniform distribution - each parameter value is equally likely:

$$f(\pi_i) \propto 1$$

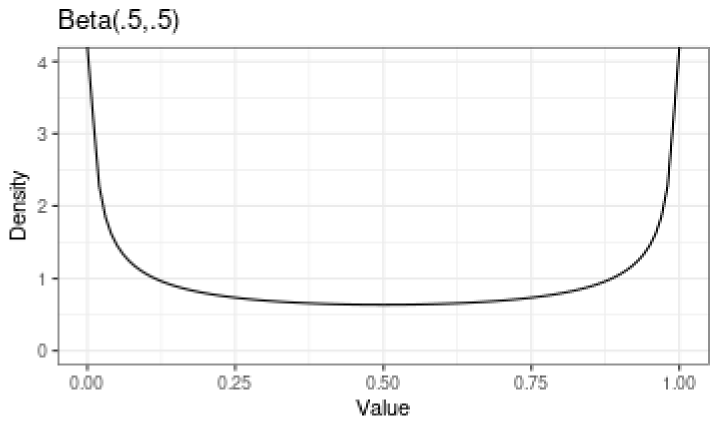
  - In settings where the parameter takes on values $(-\infty, \infty)$, we could still consider a "uniform" prior but it will be **improper** as it's not a density that integrates to 1
- For $\pi_i \in (0, 1)$, the uniform prior corresponds to the $\text{Beta}(1, 1)$ distribution
  - Originally proposed by Bayes.

Choosing a prior

- Haldane proposed instead $Beta(0, 0)$ - notably this is *improper*



- And Jeffreys proposed $\mathrm{Beta}(.5, .5)$



The posterior density

- We've mentioned that $f(\pi_i|Y_i)$ is a beta distribution -- let's show that!

$$f(\pi_i|\mathbf{Y}) \propto f(Y_i|\pi_i)f(\pi_i)$$

- Plug in the densities (we'll drop any multiplicative constants that don't depend on π_i)

$$f(\pi_i|\mathbf{Y}) \propto \left[\pi_i^{Y_i}(1-\pi_i)^{n_i-Y_i}\right] \times \left[\pi_i^{\alpha_0-1}(1-\pi_i)^{\beta_0-1}\right]$$

- Adding the exponents

$$f(\pi_i|\mathbf{Y}) \propto \pi_i^{Y_i+\alpha_0-1}(1-\pi_i)^{n_i-Y_i+\beta_0-1}$$

- And we can recognize this as the **kernel** of the beta distribution with parameters $\alpha = Y_i + \alpha_0$ and $\beta = n_i - Y_i + \beta_0$

Quantities of interest

- Once we have a posterior distribution, we typically will report **summaries** in the style of our typical frequentist point and interval estimates.
  - Note, however, that these have a different interpretation in the Bayesian framework.
- **Point** summaries
  - *Posterior Mean*: $\hat{\theta} = E[\theta|\mathbf{Y}]$
  - *Posterior Mode*: $\hat{\theta} = \arg\max_{\theta} p(\theta|\mathbf{Y})$
- **Credible interval**: A 95% credible interval (l_{95}, h_{95}) is a range of values that contains 95% of the posterior density

$$\int_{l_{95}}^{h_{95}} f(\theta|\mathbf{Y})d\theta = .95$$

- There is no one unique credible interval! As a result, there are a few common choices for how to construct a credible interval
  - **Highest Density** interval (HDI) - no values outside of the interval have higher density than values *inside* the interval
  - **Equal-tailed** interval (ETI) - the probability of being below the lower limit is equal to the probability above the upper limit

Application

- Let's take a look at the elections data. Start by loading it in

```r
elections <- read_csv("data/us-house-wide.csv")
```

- One feature of this county-level House data is that because House districts don't perfectly overlap counties, we have some county-district combinations with very few voters!
  - Consider my home county: **Hennepin County, MN**

```r
hennepin <- elections %>% filter(county == "Hennepin")
print(hennepin)
```

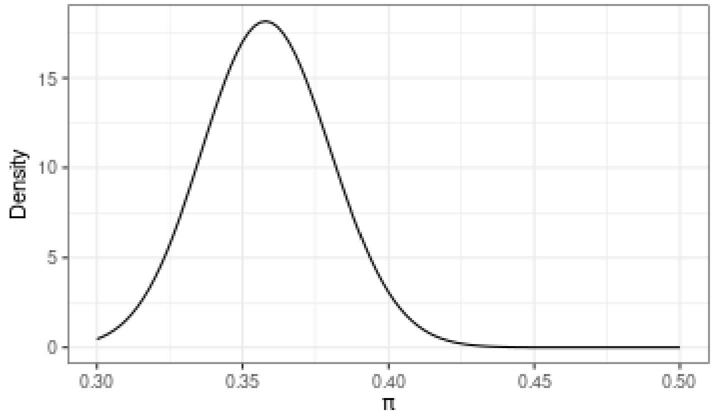
```
## # A tibble: 3 × 10
##   state county   fipscode fipscode2  office  distr…¹ total…²    dem    rep other
##   <chr> <chr>    <chr>    <chr>      <chr>   <chr>     <dbl>  <dbl>  <dbl> <dbl>
## 1 MN    Hennepin 27053    2705300000 US Hou… 3        304621 172435 131604   582
## 2 MN    Hennepin 27053    2705300000 US Hou… 5        318328 251739  65471  1118
## 3 MN    Hennepin 27053    2705300000 US Hou… 6           475    170    305     0
## # … with abbreviated variable names ¹district, ²total.votes
```

- MN-6 only has 475 votes from Hennepin County.

Application

- Let's get the posterior distribution for π_i for MN-6 in Hennepin County under an uninformative uniform prior

```
mn6_hennepin <- elections %>% filter(state=="MN"&county == "Hennepin"&district == 6)
posterior_alpha <- mn6_hennepin$dem + 1
posterior_beta <- mn6_hennepin$total.votes - mn6_hennepin$dem + 1
# Plot the posterior
mn6_posterior <- ggplot() + xlim(.3, .5)  + geom_function(fun=dbeta, args=list(shape1=posterior
  ylab("Density")
mn6_posterior
```



Application

- We know the form of the beta mean, so our posterior mean estimate is

```
posterior_mean <- posterior_alpha/(posterior_alpha + posterior_beta)
posterior_mean
```

```
## [1] 0.358
```

- And we can obtain an equal-tailed 95% credible interval simply via the quantile function

```
posterior_ci <- c(qbeta(.025, shape1=posterior_alpha, shape2=posterior_beta), qbeta(.975, shape
posterior_ci
```

```
## [1] 0.316 0.402
```

Application

- But we have some more information from the other counties in the district

```
mn6 <- elections %>% filter(state=="MN"&district == 6)
mn6
```

```
## # A tibble: 8 × 10
##   state county     fipscode fipscode2  office   distr…¹ total…²   dem   rep other
##   <chr> <chr>      <chr>    <chr>      <chr>    <chr>     <dbl> <dbl> <dbl> <dbl>
## 1 MN    Anoka      27003    2700300000 US Hou… 6        107459 45178 62062   219
## 2 MN    Benton     27009    2700900000 US Hou… 6         15922  5800 10081    41
## 3 MN    Carver     27019    2701900000 US Hou… 6         18457  6511 11938     8
## 4 MN    Hennepin   27053    2705300000 US Hou… 6           475   170   305     0
## 5 MN    Sherburne  27141    2714100000 US Hou… 6         38996 13271 25665    60
## 6 MN    Stearns    27145    2714500000 US Hou… 6         53937 22204 31666    67
## 7 MN    Washington 27163    2716300000 US Hou… 6         21545  9085 12445    15
## 8 MN    Wright     27171    2717100000 US Hou… 6         58935 20113 38769    53
## # … with abbreviated variable names ¹district, ²total.votes
```

- What if we instead constructed our prior such that its mean was centered on the average of all the other counties?
    - We can control the strength of the prior via the prior variance

Application

```
prior_mean <- sum(mn6 %>% filter(county != "Hennepin") %>% pull(dem))/sum(mn6 %>% filter(county
prior_variance <- (prior_mean*(1-prior_mean))/1000
# Convert these to alpha/beta
prior_alpha <- (((1 - prior_mean)/prior_variance) - (1/prior_mean))*prior_mean^2
prior_beta <- prior_alpha*(1/prior_mean - 1)
```

Application

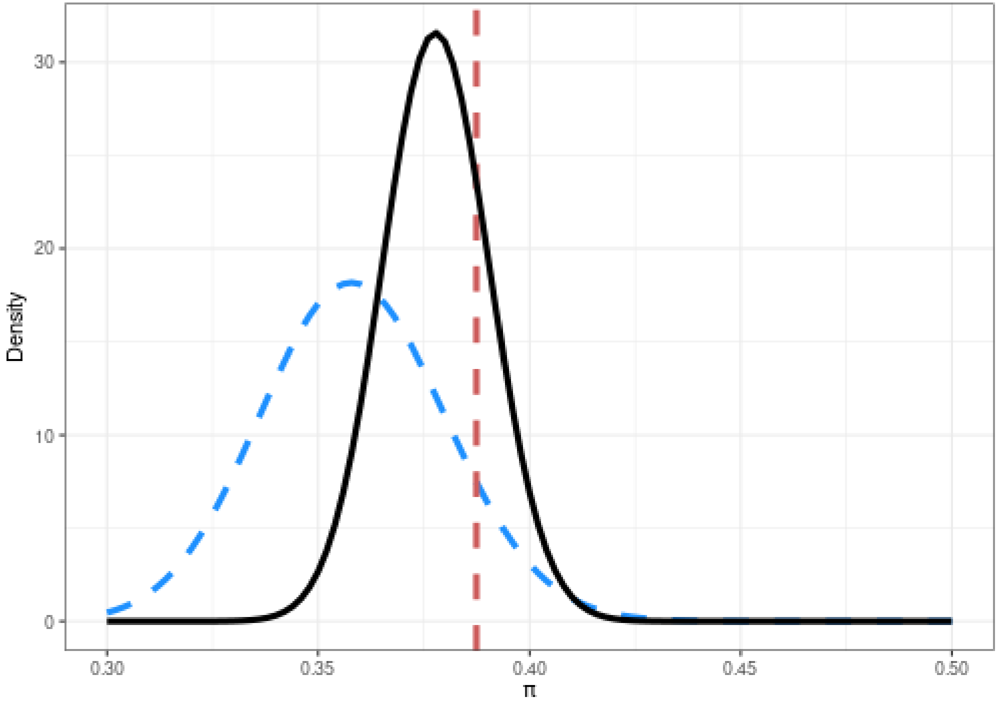
- Let's plug-in the new prior parameters

```
posterior2_alpha <- mn6_hennepin$dem + prior_alpha
posterior2_beta <- mn6_hennepin$total.votes - mn6_hennepin$dem + prior_beta
# Plot the posterior
mn6_posterior2 <- ggplot() + xlim(.3, .5)  +  geom_function(fun=dbeta, args=list(shape1=posteri
  theme_bw() + xlab(expression(pi)) +
  ylab("Density") + geom_vline(xintercept=prior_mean, lty=2, lwd=1.5, col="indianred") +
  geom_function(fun=dbeta, args=list(shape1=posterior2_alpha, shape2=posterior2_beta),lwd=1.5)
```

Application

- Let's plug-in the new prior parameters

```
mn6_posterior2
```



Application

- Our new posterior mean

```
posterior2_mean <- posterior2_alpha/(posterior2_alpha + posterior2_beta)
posterior2_mean
```

```
## [1] 0.378
```

- And 95% credible interval

```
posterior2_ci <- c(qbeta(.025, shape1=posterior2_alpha, shape2=posterior2_beta), qbeta(.975, sh
posterior2_ci
```

```
## [1] 0.353 0.403
```

Application

- The posterior mean can be thought of as a "weighted average" of the prior mean and the MLE
  - The weights are controlled by the variance of the prior.
  - Can interpret the hyper-parameters for this case - α_0 and β_0 - as the number of "previously observed" counts.

- Stronger priors $\rightsquigarrow$ narrower credible intervals

  - But it takes a *lot* more data to move the posterior distribution from the prior.

- Often the prior serves to **regularize** our estimates

  - We want our estimates to be "pulled" towards a particular value if there's very little data.
  - A common type of "regularizing" prior is designed to attenuate our estimates to 0 - we'll talk about this when we get to Week 8!

Connections with Frequentism

- You'll notice that for the uninformative uniform prior, our posterior mode is equivalent to the MLE
  - If $f(\theta) \propto 1$, $f(\theta|\mathbf{Y}) = f(\mathbf{Y}|\theta)$
- More generally, for most well-behaved priors and likelihoods, the **Bernstein-von Mises** theorem states that as $n \to \infty$...
  - ...the posterior distribution $f(\theta|\mathbf{Y})$ will converge to a **normal** distribution
  - ...centered at the true parameter θ_0
  - ...with variance-covariance matrix equal to the inverse Fisher information
- Essentially: In large samples, posterior distributions converge to the sampling distribution of the MLEs

Markov-Chain Monte Carlo

Markov-Chain Monte Carlo

- In many settings, we have this problem in computing the posterior

$$f(\theta|\mathbf{Y}) = \frac{\overbrace{f(\mathbf{Y}|\theta)}^{\text{Easy!}} \times \overbrace{f(\theta)}^{\text{Easy!}}}{\underbrace{f(\mathbf{Y})}_{\textbf{HARD!}}}$$

- The **prior** $f(\theta)$ has a known distribution (sometimes up to a proportionality constant)
- The **likelihood** $f(\mathbf{Y}\ \theta)$ has a known distribution specified by our data-generating process
- In general, we write our model so that the joint density of the data and the parameters $f(\mathbf{Y}, \theta) = f(\mathbf{Y}|\theta)f(\theta)$ factors very neatly
- But $f(\mathbf{Y})$ is a tough to evaluate integral!

$$f(\mathbf{Y}) = \int f(\mathbf{Y}, \theta)d\theta$$

- We've used the trick of having a **conjugate prior** which lets us know the distribution of the posterior
  - More generally, we've often been able to inspect $f(\mathbf{Y}|\theta) \times f(\theta)$ to identify its distributional form.

Markov-Chain Monte Carlo

- Markov-Chain Monte Carlo methods allow us to generate a sample of observations from the target posterior **even when we don't know it** as long as we can evaluate a function that is **proportional** to the posterior
    - **Monte Carlo**: Use repeated samples to obtain numeric approximations of key quantities
    - **Markov-Chain**: The samples from the posterior are constructed via a "chain" that has the Markov property
- **Our Goal**: Generate a monte carlo sample $\{\theta^{(1)}, \theta^{(2)}, \theta^{(3)}, \ldots\}$ of arbitrary length such that the sequence of draws **converges** to a **stationary distribution** that is the target posterior
- This sample is a **markov chain** in that the distribution of the $(i+1)$th draw depends on the value of the ith, but only on that past value

$$f(\theta^{(i+1)}|\mathbf{Y}, \theta^{(i)}, \theta^{(i-1)}, \ldots, \theta^{(1)}) = f(\theta^{(i+1)}|\mathbf{Y}, \theta^{(i)})$$

- Important note: our samples from the target posterior will be **dependent**

Metropolis-Hastings Algorithm

- The foundational algorithm for generating MCMC samples is the **Metropolis-Hastings** algorithm.
  - **Crucially**: We can sample from almost *any* distribution (though some distributions are better than others)
- The algorithm relies on two key concepts to generate another sample $\theta^{(i+1)}$ given the past sample value $\theta^{(i)}$:
  - The **proposal distribution** - A probability distribution $f(\theta^{(i+1)}|\theta^{(i)})$ that generates our "proposal" value
  - The **acceptance probability** - A calculation for the probability of "accepting" the proposed value or rejecting it

Metropolis-Hastings Algorithm

- To generate the markov chain of M samples from the target posterior distribution $\{\theta^{(1)}, \theta^{(2)}, \theta^{(3)}, \ldots, \theta^{(M)}\}$, the **Metropolis-Hastings** algorithm iterates between two steps:

1. **Proposal**: Conditional on the current value θ, generate a draw θ^* from $Q(\theta^*|\theta)$ where Q is the proposal distribution.
2. **Accept/Reject**: With probability α, "accept" the proposal and set $\theta^{(i+1)} = \theta^*$ - otherwise "reject" and set $\theta^{(i+1)} = \theta$

$$\alpha = \min\left\{1, \frac{f(\mathbf{Y}|\theta^*)f(\theta^*)}{f(\mathbf{Y}|\theta)f(\theta)} \times \frac{Q(\theta|\theta^*)}{Q(\theta^*|\theta)}\right\}$$

- **Intuitively** - If the proposal distribution is symmetric...
  - ...if the unnormalized posterior density is higher at the proposed rather than the current location, **always accept**
  - ...if the unnormalized posterior density is lower at the proposed rather than current location, **maybe accept**

Proposal Distribution

- Our choice of Q governs how quickly the Markov Chain will "converge" to the true posterior
  - We want to choose a proposal distribution that could "eventually" propose each value in the domain of θ
  - We also want a proposal distribution that generates a high acceptance probability α
- It is common to choose a proposal distribution that is **symmetric** such that

$$\frac{Q(\theta|\theta^*)}{Q(\theta^*|\theta)} = 1$$

- For $\theta \in (-\infty, \infty)$, common to propose θ^* from a $\text{Normal}(\theta, \Omega)$
  - That is, from a normal distribution centered on the "current" parameter with an arbitrarily chosen variance Ω
  - This is **symmetric** since the chances of proposing θ^* from θ are the same as the chances of proposing θ from θ^*

Proposal Distribution

- Another alternative would be a $\text{Uniform}(\theta - \Delta, \theta + \Delta)$
  - Uniform distribution centered at θ where every value in the interval has density $\frac{1}{2\Delta}$

- For parameters that are constrained to $(0, 1)$, a simple choice would be to sample from the

$$\text{Uniform}(0, 1)$$

  - Trivially symmetric since it doesn't depend at all on θ
  - But likely to generate bad proposals!

- We're usually better off transforming our parameter to an unconstrained value and sampling there

Proposal Distribution

- To get some intuition, it's worth considering the case where somehow we could generate independent samples from the true posterior and

$$Q(\theta^*|\theta) = f(\theta^*|\mathbf{Y})$$

- What is our acceptance probability in this scenario?

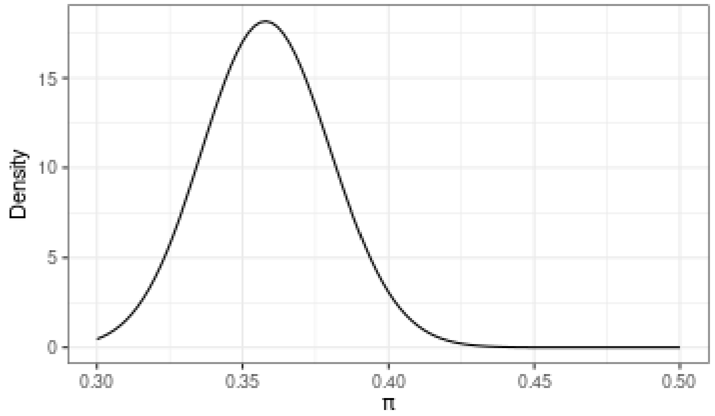
$$\frac{f(\mathbf{Y}|\theta^*)f(\theta^*)}{f(\mathbf{Y}|\theta)f(\theta)} \times \frac{f(\theta|\mathbf{Y})}{f(\theta^*|\mathbf{Y})} = \frac{f(\theta^*|\mathbf{Y})f(\mathbf{Y})}{f(\theta|\mathbf{Y})f(\mathbf{Y})} \times \frac{f(\theta|\mathbf{Y})}{f(\theta^*|\mathbf{Y})} = 1$$

- If we were somehow proposing from the true posterior distribution, Metropolis-Hastings would **always accept** each proposal!

Illustration

- Let's go back to our posterior distribution for the Democratic vote share in the part of MN-6 that is in Hennepin County

```
mn6_hennepin <- elections %>% filter(state=="MN"&county == "Hennepin"&district == 6)
posterior_alpha <- mn6_hennepin$dem + 1
posterior_beta <- mn6_hennepin$total.votes - mn6_hennepin$dem + 1
# Plot the posterior
mn6_posterior <- ggplot() + xlim(.3, .5)  + geom_function(fun=dbeta, args=list(shape1=posterior
  ylab("Density")
mn6_posterior
```



- In this case, we calculated the posterior directly since it was beta, but suppose we couldn't?

Illustration

- Let's construct our MCMC sampler. First, let's choose a proposal distribution
  - For $\pi_i \in (0, 1)$, we could use a Beta, but the Beta isn't symmetric
  - $\text{Uniform}(0, 1)$ would generate a lot of bad proposals
- Instead, let's transform π_i to be unbounded. Define $\theta_i = \log\left(\frac{\pi_i}{1-\pi_i}\right)$
  - We'll sample θs and convert them back to π when we evaluate the likelihood and prior
  - An intuitive distribution $Q(\theta^*|\theta)$ would be the standard logistic distribution centered on θ. It's **symmetric**!

Illustration

```
set.seed(60637)
logistic <- function(x) 1/(1 + exp(-x)) # Helper Function
logit <- function(x) log(x/(1-x))
M <- 100000  # Number of MCMC samples
pi_mcmc <- rep(NA, M) # Vector to store our samples
pi_mcmc[1] <- .5 # Pick a starting value
```

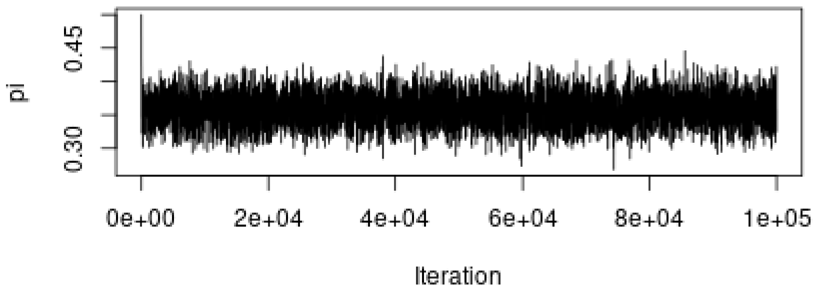
Illustration

```r
for (i in 1:(M-1)){ # For i in 1:(M-1)
  ## Step 1 - Proposal
  theta_i <- logit(pi_mcmc[i])
  theta_star <- rlogis(1, location = theta_i)
  ## Step 2 - Accept/Reject
  lik_star <- dbinom(mn6_hennepin$dem, mn6_hennepin$total.votes, prob = logistic(theta_star))
  prior_star <- dbeta(logistic(theta_star), shape1 = 1, shape2=1)
  lik_current <-  dbinom(mn6_hennepin$dem, mn6_hennepin$total.votes, prob = logistic(theta_i))
  prior_current <- dbeta(logistic(theta_i), shape1 = 1, shape2=1)
  Q_star_current <- dlogis(theta_star, location=theta_i) # This is all optional b/c of symmetry
  Q_current_star <- dlogis(theta_i, location=theta_star)
  #Assemble the acceptance ratio
  ar <- ((lik_star*prior_star)/(lik_current*prior_current))*(Q_current_star/Q_star_current)
  # Choose to accept or reject
  accept <- rbinom(1,1,min(1,ar))
  # Store the next value
  pi_mcmc[i+1] <- logistic(theta_star)*accept + logistic(theta_i)*(1-accept)
}
```

Illustration

- Let's first see how the chain progresses by generating the **trace plot**

```
plot(y=pi_mcmc, x=1:length(pi_mcmc), xlab="Iteration", ylab="pi", type="l")
```

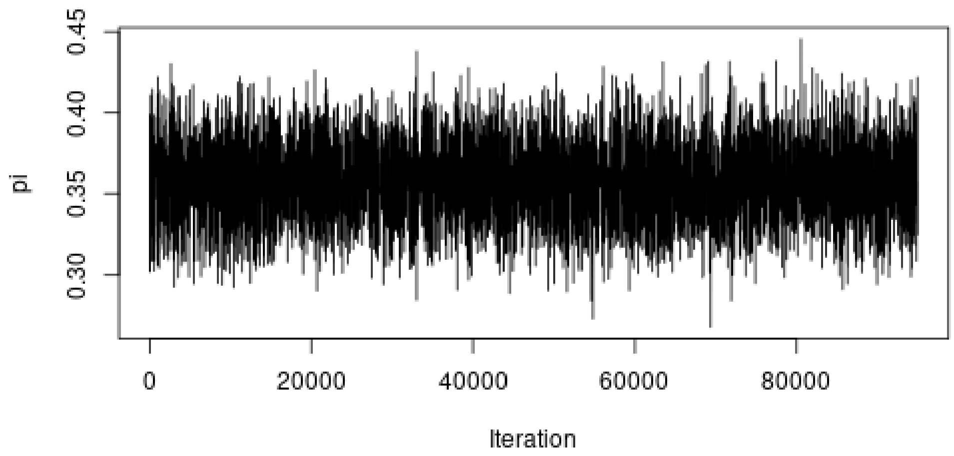


- You'll see that it takes a bit for the chain to reach the area of the posterior with the most mass
  - If we started closer to the "mass" of the distribution, we would have converged faster
- In practice, we will throw away some number of our initial samples as a **burn-in** period, since we know that the chain needs some time to reach the stationary distribution.
  - Here we'll drop the first 5000 samples and keep the rest

Illustration

- Our ideal trace plots have no clear trends - in such a case, we consider the chain to have reached its stationary distribution

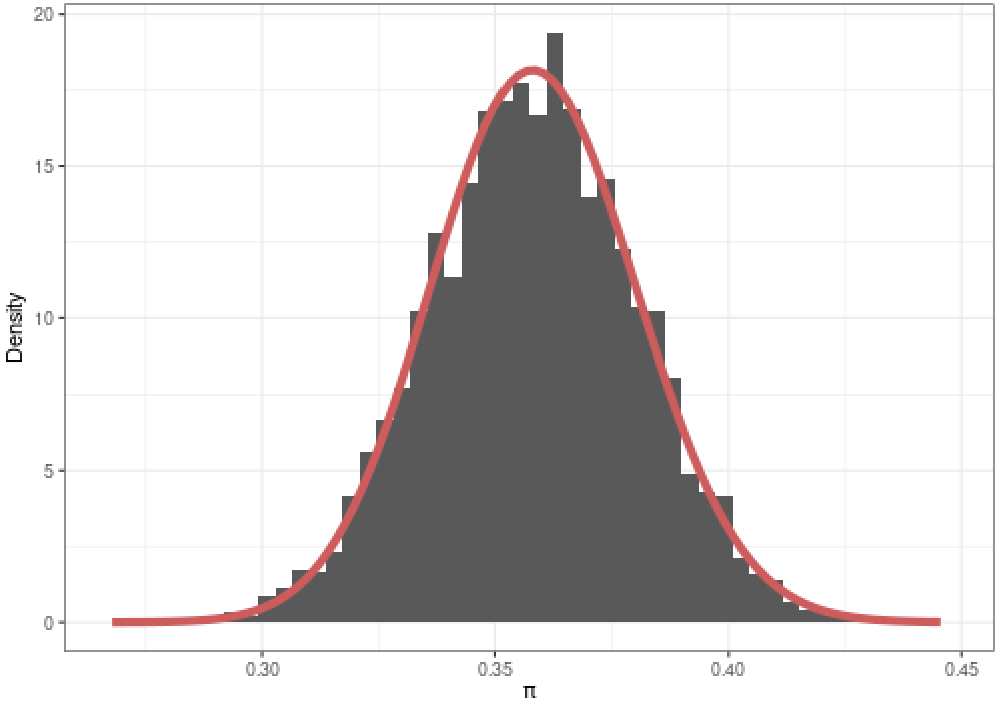
```
pi_mcmc_use <- pi_mcmc[5000:M] # Toss our burn-in period
plot(y=pi_mcmc_use, x=1:length(pi_mcmc_use), xlab="Iteration", ylab="pi", type="l")
```



- Note that trace plots are **diagnostics** -- they don't prove that the chain has "mixed," but a lot of jumps or stagnant periods would suggest that our chain needs to run longer (or we need to choose a better proposal distribution!)

Illustration

```
# Plot the histogram on top of the "true" distribution
mn6_mcmc <- data.frame(x=pi_mcmc_use) %>% ggplot(aes(x=x)) +  theme_bw() + xlab(expression(pi))
  ylab("Density")
mn6_mcmc
```



Illustration

- We can compute summaries like the posterior mean or a 95% credible interval by taking summaries of the MCMC sample

```
posterior_mcmc_mean <- mean(pi_mcmc_use)
posterior_mcmc_mean
```

```
## [1] 0.358
```

```
posterior_mean
```

```
## [1] 0.358
```

Illustration

- We get very close to the true posterior (and can get closer with more iterations)

```r
posterior_mcmc_ci <- quantile(pi_mcmc_use, c(.025, .975))
posterior_mcmc_ci
```

```
##  2.5% 97.5%
## 0.315 0.401
```

```r
posterior_ci
```

```
## [1] 0.316 0.402
```

Multiple parameters

- In this example, we've focused on estimating a single parameter. Suppose instead we have K parameters: $\theta = \{\theta_1, \theta_2, \theta_3, \dots, \theta_K\}$
- With multiple parameters, we'd need a multivariate proposal distribution (like a multivariate normal)
    - But this can be hard and mixing is often very slow with multivariate proposals.
- Instead, we can use **Gibbs sampling** -- generating each update $\theta_k^{(i+1)}$ conditional on the other current values of the other parameters $\theta_{-k}^{(i)}$
    - Importantly, if we **can** sample from the conditional distribution $\theta_k | \theta_{k-1}, \mathbf{Y}$ we can get proposals with acceptance probability 1
    - Even if we can't, we can use a M-H step for that parameter

Gibbs Sampling

- The **Gibbs sampling** algorithm generates updates $\theta^{(i+1)} = \{\theta_1^{(i+1)}, \theta_2^{(i+1)}, \theta_3^{(i+1)}, \ldots, \theta_K^{(i+1)}\}$ by sampling in sequence:

$$
\begin{aligned}
\theta_1^{(i+1)} &:= \theta_1^* \sim f(\theta_1 | \mathbf{Y}, \theta_2^{(i)}, \theta_3^{(i)}, \ldots \theta_K^{(i)}) \\
\theta_2^{(i+1)} &:= \theta_2^* \sim f(\theta_2 | \mathbf{Y}, \theta_1^{(i+1)}, \theta_3^{(i)}, \ldots \theta_K^{(i)}) \\
&\vdots \\
\theta_k^{(i+1)} &:= \theta_k^* \sim f(\theta_k | \mathbf{Y}, \theta_1^{(i+1)}, \theta_2^{(i+1)}, \ldots \theta_{k-1}^{(i+1)}, \theta_{k+1}^{(i)} \ldots, \theta_K^{(i)}) \\
&\vdots \\
\theta_K^{(i+1)} &:= \theta_K^* \sim f(\theta_K | \mathbf{Y}, \theta_1^{(i+1)}, \theta_2^{(i+1)}, \ldots \theta_{K-1}^{(i+1)})
\end{aligned}
$$

- **Important**: Sampling from the conditionals $f(\theta_k | \theta_{-k}, \mathbf{Y})$ is often easier to do than sampling from the marginal distribution $f(\theta_k | \mathbf{Y})$.
  - In fact, we'll often introduce latent variables to make a Gibbs sampling algorithm work where it otherwise wouldn't. This is called **data augmentation**

Gibbs Sampling

- To see why sampling from the conditionals works, consider the implied MH acceptance probability when we treat the conditional distribution $f(\theta_k|\theta_{-k}^{(i)}, \mathbf{Y})$ as the proposal distribution Q

$$\alpha = \frac{f(\mathbf{Y}|\theta_k^*, \theta_{-k}^{(i)})f(\theta_k^*, \theta_{-k}^{(i)})}{f(\mathbf{Y}|\theta_k^{(i)}, \theta_{-k}^{(i)})f(\theta_k^{(i)}, \theta_{-k}^{(i)})} \times \frac{f(\theta_k^{(i)}|\mathbf{Y}, \theta_{-k}^{(i)})}{f(\theta_k^*|\mathbf{Y}, \theta_{-k}^{(i)})}$$

- Rewrite the numerator/denominator in terms of the posterior and the data (and recall that $f(\mathbf{Y})$ cancels)

$$\alpha = \frac{f(\theta_k^*, \theta_{-k}^{(i)}|\mathbf{Y})}{f(\theta_k^{(i)}, \theta_{-k}^{(i)}|\mathbf{Y})} \times \frac{f(\theta_k^{(i)}|\mathbf{Y}, \theta_{-k}^{(i)})}{f(\theta_k^*|\mathbf{Y}, \theta_{-k}^{(i)})}$$

Gibbs Sampling

- Factor the posteriors

$$\alpha = \frac{f(\theta_k^*|\theta_{-k}^{(i)}, \mathbf{Y}) \times f(\theta_{-k}^{(i)}|\mathbf{Y})}{f(\theta_k^{(i)}|\theta_{-k}^{(i)}, \mathbf{Y}) \times f(\theta_{-k}^{(i)}|\mathbf{Y})} \times \frac{f(\theta_k^{(i)}|\mathbf{Y}, \theta_{-k}^{(i)})}{f(\theta_k^*|\mathbf{Y}, \theta_{-k}^{(i)})}$$

- Everything cancels so that $\alpha = 1$
  - So Gibbs sampling is a special case of Metropolis-Hastings where the proposal distribution choice **guarantees** acceptance.

Bayesian Regression

Application: Predicting Elections

- Now suppose that instead of observing counts, we look at the democratic party vote share in county i in the 2018 House elections (imagine we summed over all of the votes in all of the districts in that county).
    - Let Y_i denote the share of votes going to the democratic party candidate in 2018 in county i.
- We want to generate a predictive model based on X_i, (here, we'll just be using the democratic presidential vote share in 2016, but we'll be adding more as we go).

Bayesian Normal Model

- The conventional "normal" Bayesian regression model assumes

$$Y_i|X_i, \beta, \sigma^2 \sim \text{Normal}(X_i'\beta, \sigma^2)$$

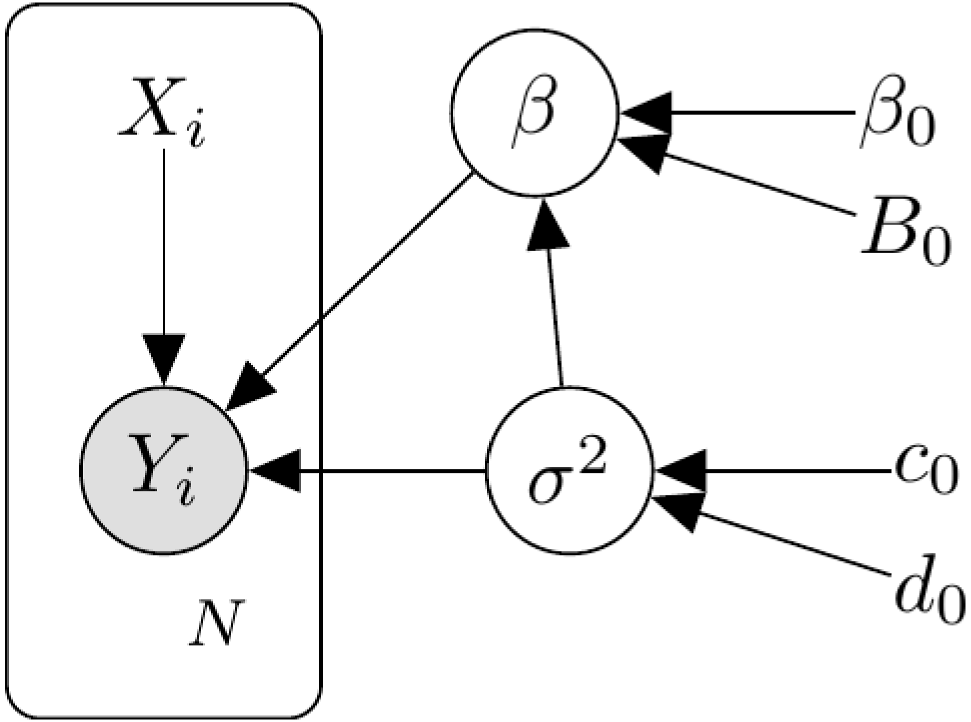
$$\beta|\sigma^2 \sim \text{Normal}(\beta_0, \sigma^2 B_0^{-1})$$

$$\sigma^2 \sim \text{Inverse-Gamma}(\frac{c_0}{2}, \frac{d_0}{2})$$

- In this case, β_0, B_0, c_0 and d_0 are all hyper-parameters
- This is referred to as the "Normal-Inverse-Gamma" model as the joint prior distribution on β, σ^2 is Normal-Inverse Gamma and it happens to also be the **conjugate prior** for the case with unknown β and σ^2
  - So the posterior is known in closed form to be **Normal-Inverse-Gamma**
  - And conditional on σ^2, $\beta|\mathbf{X}, \mathbf{Y}, \sigma^2$ is also normally distributed!
- A common simplification is to make β and σ^2 marginally independent

$$\beta \sim \text{Normal}(\beta_0, B_0^{-1})$$

Plate notation



Application: Predicting Elections

```
# Aggregate the house data to counties
elections_county <- elections %>% group_by(fipscode) %>% summarize(state=state[1], county=count
                                                                   total.votes = sum(total.vote
                                                                   dem = sum(dem))

# Merge in 2015 Presidential
pres_2016 <- read_csv("data/clinton_2016_vote.csv")
elections_county <- elections_county %>% left_join(pres_2016 %>% dplyr::select(county_fips, car
                                                   by=c(fipscode="county_fips"))

# Generate vote shares
elections_county$dem2018 <- elections_county$dem/elections_county$total.votes
elections_county$dem2016 <- elections_county$candidatevotes/elections_county$totalvotes

# Drop missing
elections_county <- elections_county %>% filter(!is.na(dem2018)&!is.na(dem2016))
```

Metropolis-Hastings

- Set up the regression

```
X_mat <- model.matrix(dem2018 ~ dem2016, data=elections_county)
Y <- elections_county$dem2018
K <- ncol(X_mat) # Number of beta parameters
```

- Set up a diffuse prior

```
beta_0 <- rep(0, K)
B_inv_0 <- solve(diag(rep(1/9, K)))
c_0 = 0.001
d_0 = 0.001
```

- Set up the MCMC

```
M <- 40000  # Number of MCMC samples
burnin <- 5000
beta_mcmc <- matrix(nrow = M, ncol=K) # Vector to store our samples
beta_mcmc[1,] <- c(0,1) # Pick a starting value
sigma_mcmc <- rep(NA, M)
sigma_mcmc[1] <- 1
```

Metropolis-Hastings

- Write some functions to evaluate the likelihood and priors

```
log_lik_norm <- function(b, sigma, Y, X){
  linpred <- X%*%b
  sum(dnorm(Y, mean=linpred, sd=sigma, log=T))
}
```

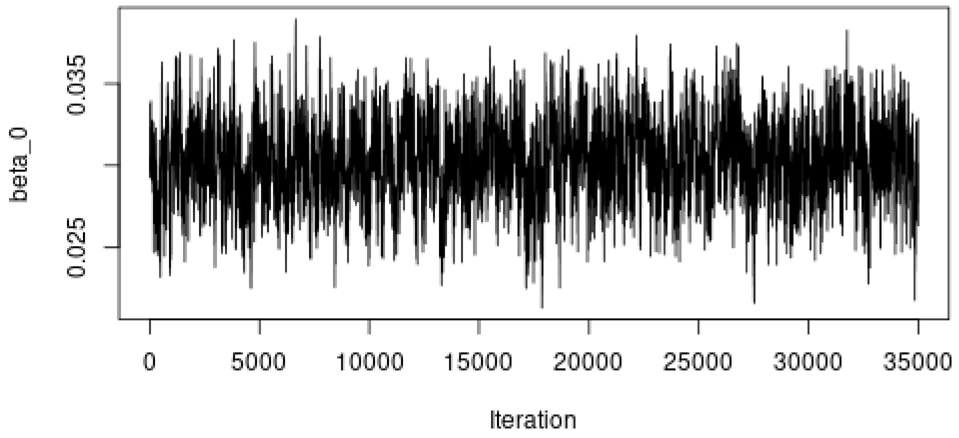
Metropolis-Hastings

```r
set.seed(60637)
for (i in 1:(M-1)){ # For i in 1:(M-1)
  ## Beta
    ## Step 1 - Proposal
    beta_star <- as.vector(mvtnorm::rmvnorm(1, mean  = beta_mcmc[i,], sigma=.00001*diag(K)))
    ## Step 2 - Accept/Reject
    lik_star_beta <- log_lik_norm(beta_star, sigma_mcmc[i], Y, X_mat)
    lik_current_beta <- log_lik_norm(beta_mcmc[i,], sigma_mcmc[i], Y, X_mat)

    prior_star_beta <- mvtnorm::dmvnorm(beta_star, mean = beta_0, sigma = B_inv_0, log=T)
    prior_current_beta <- mvtnorm::dmvnorm(beta_mcmc[i,], mean = beta_0, sigma = B_inv_0, log=T

    ## Accept/reject
    ar_beta <- exp( lik_star_beta + prior_star_beta - lik_current_beta - prior_current_beta)
    accept_beta <- rbinom(1,1,min(1,ar_beta))
    beta_mcmc[i+1,] <- beta_star*accept_beta + beta_mcmc[i,]*(1-accept_beta)

  ## Sigma
    ## Step 1 - Proposal
    sigma_log <- rnorm(1, mean = log(sigma_mcmc[i]), sd=.01)
    sigma_star <- exp(sigma_log)

    ## Step 2 - Accept/Reject
    lik_star_sigma <- log_lik_norm(beta_mcmc[i+1,], sigma_star, Y, X_mat)
    lik_current_sigma <- log_lik_norm(beta_mcmc[i+1,], sigma_mcmc[i], Y, X_mat)
    # The trick here is independence - otherwise if \sigma^2 appeared in the prior for \beta, w
    prior_star_sigma <- log(MCMCpack::dinvgamma(sigma_star^2, shape = c_0/2, scale = d_0/2))
```

Convergence

- β_0

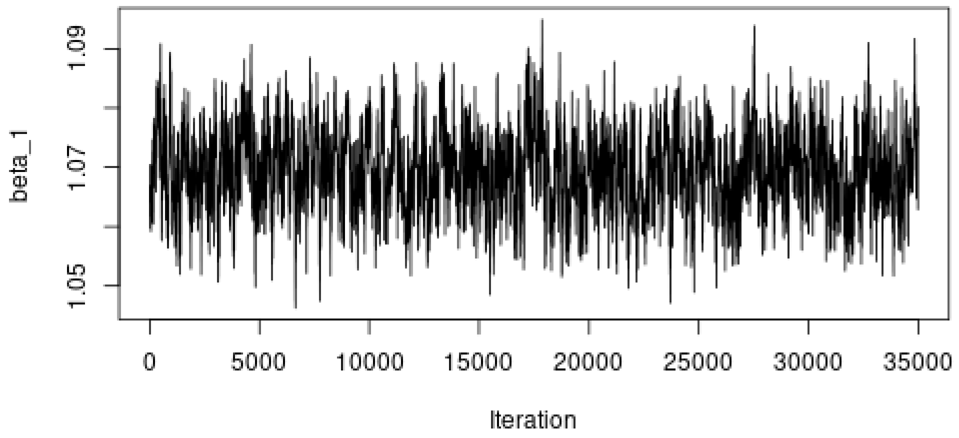
```
beta_mcmc_use <- beta_mcmc[burnin:M,] # Toss our burn-in period
plot(y=beta_mcmc_use[,1], x=1:length(beta_mcmc_use[,1]), xlab="Iteration", ylab="beta_0", type=
```



Convergence

- β_1

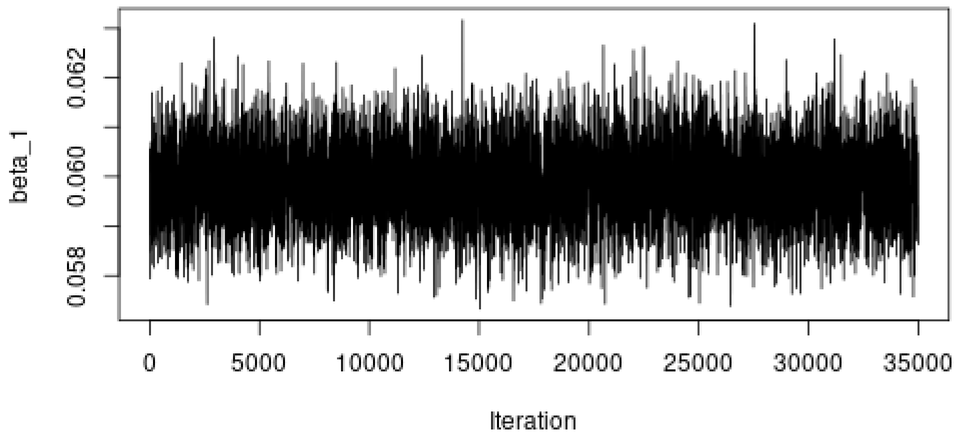
```
plot(y=beta_mcmc_use[,2], x=1:length(beta_mcmc_use[,2]), xlab="Iteration", ylab="beta_1", type=
```



Convergence

- σ

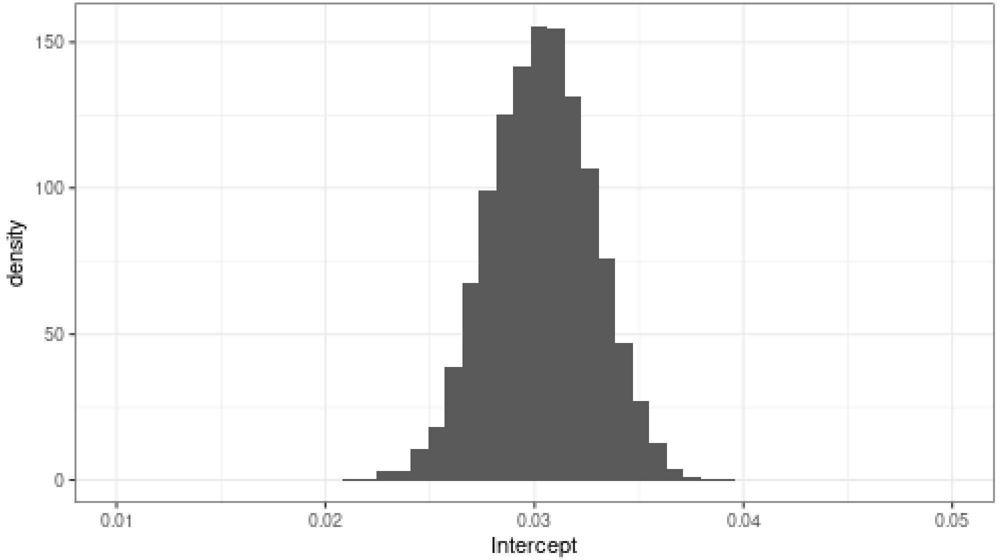
```
sigma_mcmc_use <- sigma_mcmc[burnin:M]
plot(y=sigma_mcmc_use, x=1:length(sigma_mcmc_use), xlab="Iteration", ylab="beta_1", type="l")
```



Summaries

β_0

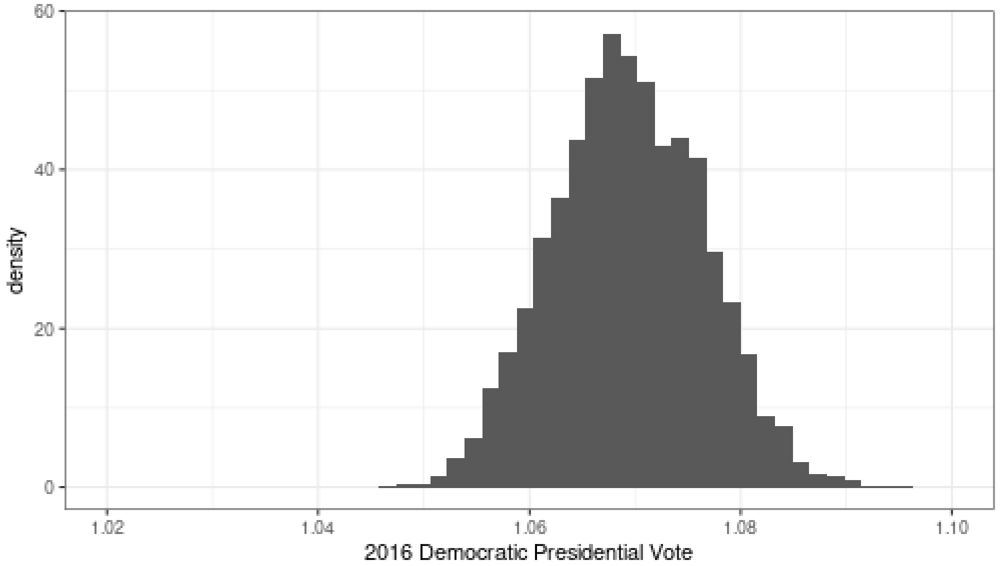
```
beta_mcmc_out <- as.data.frame(beta_mcmc_use)
colnames(beta_mcmc_out) <- c("Intercept", "dem2016")
beta_mcmc_out %>% ggplot(aes(x=Intercept)) + xlim(.01, .05) +  theme_bw() + xlab("Intercept") +
```



Summaries

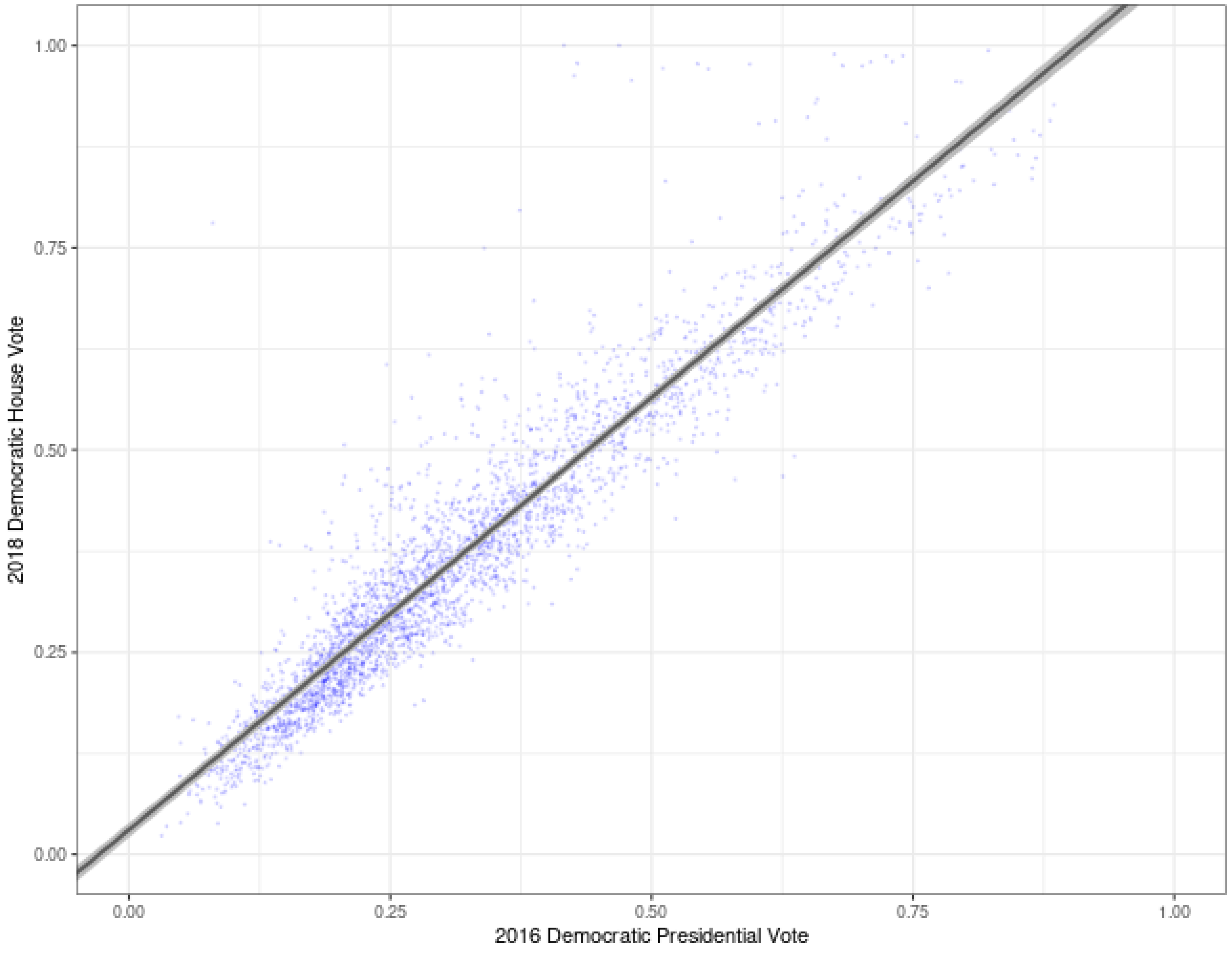
β_1

```
beta_mcmc_out %>% ggplot(aes(x=dem2016)) +  theme_bw() + xlim(1.02, 1.10) + xlab("2016 Democrat
```



Summaries

- Overlay the data and regression



Summaries

- Posterior means and 95% credible intervals

```
summary_results <- data.frame(variable = c("Intercept", "dem2016"), pm = colMeans(beta_mcmc_out
                              ci95_lower = apply(beta_mcmc_out, 2, function(x) quantile(x, .025
                              ci95_upper = apply(beta_mcmc_out, 2, function(x) quantile(x, .975

summary_results
```

```
##              variable     pm ci95_lower ci95_upper
## Intercept Intercept 0.0303     0.0255     0.0351
## dem2016     dem2016 1.0694     1.0561     1.0832
```

Gibbs sampler

- Let's check against an existing Gibbs sampling implementation from `MCMCpack`

```
set.seed(60637)
library(MCMCpack)
gibbs_results <- MCMCpack::MCMCregress(dem2018 ~ dem2016, burnin=burnin, mcmc=M-burnin,
                                       beta.start = c(0, 1), b0=beta_0, B_0 = diag(rep(1/9, K)),
                                       c0 = c_0, d0=d_0, data=elections_county)
```

Gibbs sampler

```
summary(gibbs_results)
```

```
##   (Intercept)         dem2016          sigma2
##  Min.   :0.0203   Min.   :1.03   Min.   :0.00326
##  1st Qu.:0.0288   1st Qu.:1.06   1st Qu.:0.00352
##  Median :0.0305   Median :1.07   Median :0.00359
##  Mean   :0.0305   Mean   :1.07   Mean   :0.00359
##  3rd Qu.:0.0322   3rd Qu.:1.07   3rd Qu.:0.00365
##  Max.   :0.0440   Max.   :1.10   Max.   :0.00399
```

Diagnostics

- The trace plots for the beta coefficients using Gibbs sampling have a lot less visible dependence!

```
coda::traceplot(gibbs_results)
```