Intro to Bayes

Daniel Anderson

Week 7

Agenda

[We're not going to get through it all]

- Review Homework 2
- Some equation practice
- Introduce Bayes theorem
    - Go through an example with estimating a mean
- Discuss Bayes in the context of regression modeling

Homework 2 Review

Equation practice

The data

From an example in the Mplus manual. I made up the column names.

```r
mplus_d <- read_csv(here::here("data", "mplus920.csv"))
mplus_d
```

```
## # A tibble: 7,500 x 6
##        score  baseline sch_treatment  dist_ses schid distid
##        <dbl>     <dbl>         <dbl>     <dbl> <dbl>  <dbl>
##  1  5.559216  1.383101             1 -0.642262     1      1
##  2 -0.107394 -0.789654             1 -0.642262     1      1
##  3  0.049476 -0.760867             1 -0.642262     1      1
##  4 -2.387703 -0.798527             1 -0.642262     1      1
##  5  1.180393 -0.411377             1 -0.642262     1      1
##  6  3.959005 -0.987154             1 -0.642262     2      1
##  7 -0.895792 -1.966773             1 -0.642262     2      1
##  8  2.879087  0.42117              1 -0.642262     2      1
##  9  5.611088  1.67047              1 -0.642262     2      1
## 10  2.828119  0.001154             1 -0.642262     2      1
## # … with 7,490 more rows
```

01:30

Model 1

Fit the following model

$$
\begin{aligned}
\text{score}_i &\sim N\left(\alpha_{j[i]}, \sigma^2\right) \\
\alpha_j &\sim N\left(\mu_{\alpha_j}, \sigma^2_{\alpha_j}\right), \text{for distid j} = 1, \dots, \text{J}
\end{aligned}
$$

```r
lmer(score ~ 1 + (1|distid), data = mplus_d)
```

01:30

Model 2

Fit the following model

$$\begin{aligned}\text{score}_i &\sim N\left(\alpha_{j[i],k[i]} + \beta_1(\text{baseline}), \sigma^2\right)\\ \alpha_j &\sim N\left(\mu_{\alpha_j}, \sigma^2_{\alpha_j}\right), \text{for schid j} = 1, \ldots, \text{J}\\ \alpha_k &\sim N\left(\mu_{\alpha_k}, \sigma^2_{\alpha_k}\right), \text{for distid k} = 1, \ldots, \text{K}\end{aligned}$$

```
lmer(score ~ baseline + (1|schid) + (1|distid),
     data = mplus_d)
```

01:00

Model 3

Fit the following model

$$
\begin{aligned}
\text{score}_i &\sim N\left(\alpha_{j[i],k[i]} + \beta_{1j[i]}(\text{baseline}), \sigma^2\right) \\
\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} &\sim N\left(\begin{pmatrix} \mu_{\alpha_j} \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right), \text{ for schid j} = 1, \dots, \text{J} \\
\alpha_k &\sim N\left(\gamma_0^\alpha + \gamma_1^\alpha(\text{dist\_ses}) + \gamma_2^\alpha(\text{baseline} \times \text{dist\_ses}), \sigma^2_{\alpha_k}\right), \text{ for distid k} = 1,
\end{aligned}
$$

```
lmer(score ~ baseline * dist_ses +
        (baseline|schid) + (1|distid),
      data = mplus_d)
```

01:30

Model 4

Fit the following model

$$
\begin{aligned}
\text{score}_i &\sim N\left(\alpha_{j[i],k[i]} + \beta_{1j[i]}(\text{baseline}), \sigma^2\right) \\
\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} &\sim N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{sch\_treatment}) \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_j}^2 & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma_{\beta_{1j}}^2 \end{pmatrix}\right), \text{ for schid j} = 1 \\
\alpha_k &\sim N\left(\gamma_0^\alpha + \gamma_1^\alpha(\text{dist\_ses}), \sigma_{\alpha_k}^2\right), \text{ for distid k} = 1, \dots, \text{K}
\end{aligned}
$$

```
lmer(score ~ baseline + sch_treatment + dist_ses +
        (baseline|schid) + (1|distid),
     data = mplus_d)
```

01:30

Model 5

Fit the following model

$$\text{score}_i \sim N\left(\alpha_{j[i],k[i]} + \beta_{1j[i]}(\text{baseline}), \sigma^2\right)$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_{1k[i]}^\alpha(\text{sch\_treatment}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{sch\_treatment}) \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_j}^2 & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma_{\beta_{1j}}^2 \end{pmatrix}\right), \text{ for schid j} =$$

$$\begin{pmatrix} \alpha_k \\ \gamma_{1k} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{dist\_ses}) \\ \mu_{\gamma_{1k}} \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_k}^2 & \rho_{\alpha_k\gamma_{1k}} \\ \rho_{\gamma_{1k}\alpha_k} & \sigma_{\gamma_{1k}}^2 \end{pmatrix}\right), \text{ for distid k} = 1, \ldots,$$

```
lmer(score ~ baseline * sch_treatment + dist_ses +
         (baseline|schid) + (sch_treatment|distid),
      data = mplus_d)
```

01:30

Model 6

Fit the following model

$$\text{score}_i \sim N\left(\alpha_{j[i],k[i]} + \beta_{1j[i],k[i]}(\text{baseline}), \sigma^2\right)$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_{1k[i]}^{\alpha}(\text{sch\_treatment}) \\ \gamma_{1k[i0}^{\beta_1} + \gamma_{1k[i]}^{\beta_1}(\text{sch\_treatment}) \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right)\right), \text{ for schid j} = 1, \dots, \text{J}$$

$$\begin{pmatrix} \alpha_k \\ \beta_{1k} \\ \gamma_{1k} \\ \gamma_{1k}^{\beta_1} \end{pmatrix} \sim N\left(\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_1^{\alpha}(\text{dist\_ses}) \\ \mu_{\beta_{1k}} \\ \mu_{\gamma_{1k}} \\ \mu_{\gamma_{1k}^{\beta_1}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_k} & 0 & 0 & 0 \\ 0 & \sigma^2_{\beta_{1k}} & 0 & 0 \\ 0 & 0 & \sigma^2_{\gamma_{1k}} & 0 \\ 0 & 0 & 0 & \sigma^2_{\gamma_{1k}^{\beta_1}} \end{pmatrix}\right)\right), \text{ for distid k} = 1, \dots, \text{K}$$

```
lmer(score ~ baseline * sch_treatment + dist_ses +
       (baseline|schid) +
       (baseline * sch_treatment||distid),
          data = mplus_d)
```

01:30

Final one

Fit the following model

$$
\begin{aligned}
&\text{score}_i \sim N\left(\alpha_{j[i],k[i]} + \beta_{1j[i],k[i]}(\text{baseline}), \sigma^2\right) \\
&\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_1^{\alpha}(\text{sch\_treatment}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{sch\_treatment}) \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right) \text{, for schid j} = 1, \dots, \text{J} \\
&\begin{pmatrix} \alpha_k \\ \beta_{1k} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_1^{\alpha}(\text{dist\_ses}) + \gamma_2^{\alpha}(\text{dist\_ses} \times \text{sch\_treatment}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{dist\_ses}) + \gamma_1^{\beta_1}(\text{dist\_ses} \times \text{sch\_treatment}) \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_k} & \rho_{\alpha_k\beta_{1k}} \\ \rho_{\beta_{1k}\alpha_k} & \sigma^2_{\beta_{1k}} \end{pmatrix}\right) \text{, for distid k} = 1, \dots, \text{K}
\end{aligned}
$$

```
lmer(score ~ baseline * sch_treatment * dist_ses +
               (baseline|schid) + (baseline |distid),
             data = mplus_d)
```

01:30

Bayes

A disclaimer

- There is **no** chance we'll really be able to do Bayes justice in this class
- The hope for today is that you'll get an introduction
- By the end you should be able to fit the models you already can, but in a Bayes framework
- Hopefully you also recognize the tradeoffs, and potential extensions

Bayes theorem

In equation form

You'll see this presented many different ways, perhaps mostly commonly as

$$p(B \mid A) = \frac{p(A \mid B) \times p(B)}{p(A)}$$

where $\mid$ is read as "given" and p is the probability

I prefer to give A and B more meaningful names

$$p(\text{prior} \mid \text{data}) = \frac{p(\text{data} \mid \text{prior}) \times p(\text{prior})}{\text{data}}$$

A real example

Classifying a student with a learning disability. We want to know

$$p(\mathbf{LD} \mid \mathbf{Test}_p) = \frac{p(\mathbf{Test}_p \mid \mathbf{LD}) \times p(\mathbf{LD})}{\mathbf{Test}_p}$$

Notice, this means we need to know:

- True positive rate of the test, $p(\mathbf{Test}_p \mid \mathbf{LD})$
- Base rate for learning disabilities, $p(\mathbf{LD})$
- Base rate for testing positive, $\mathbf{Test}_p$

Estimating

If we have these things, we can estimate the probability that a student has a learning disability, given a positive test.

Let's assume:

- $p(\mathbf{Test}_p \mid \mathbf{LD}) = \mathbf{0.90}$
- $p(\mathbf{LD}) = \mathbf{0.10}$
- $\mathbf{Test}_p = \mathbf{0.20}$

$$p(\mathbf{LD} \mid \mathbf{Test}_p) = \frac{.90 \times .10}{.20}$$

$$p(\mathbf{LD} \mid \mathbf{Test}_p) = 0.45$$

A bit less than you might have expected? Probability is hard...

When we see things like "90% true positive rate" we want to interpret it as $p(\mathbf{LD} \mid \mathbf{Test}_p)$, when it's actually $p(\mathbf{Test}_p \mid \mathbf{LD})$

Pieces

If we know all the pieces, we can estimate Bayes theorem directly.

Unfortunately this is almost never the case...

Alternative view

$$\textbf{updated beliefs} = \frac{\textbf{likelihood of data} \times \textbf{prior information}}{\textbf{average likelihood}}$$

How do we calculate the likelihood of the data? We have to assume some distribution.

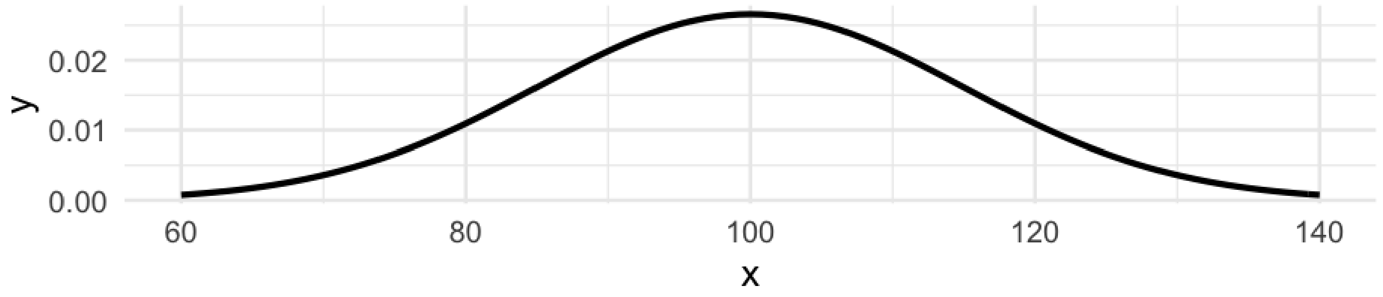
Example with IQ

```r
#install.packages("carData")
iqs <- carData::Burt$IQbio
iqs
```

```
##  [1]  82  80  88 108 116 117 132  71  75  93  95  88 111  63  77  86  83
## [19]  97  87  94  96 112 113 106 107  98
```

IQ scores are generally assumed to be generated from a distribution that looks like this:

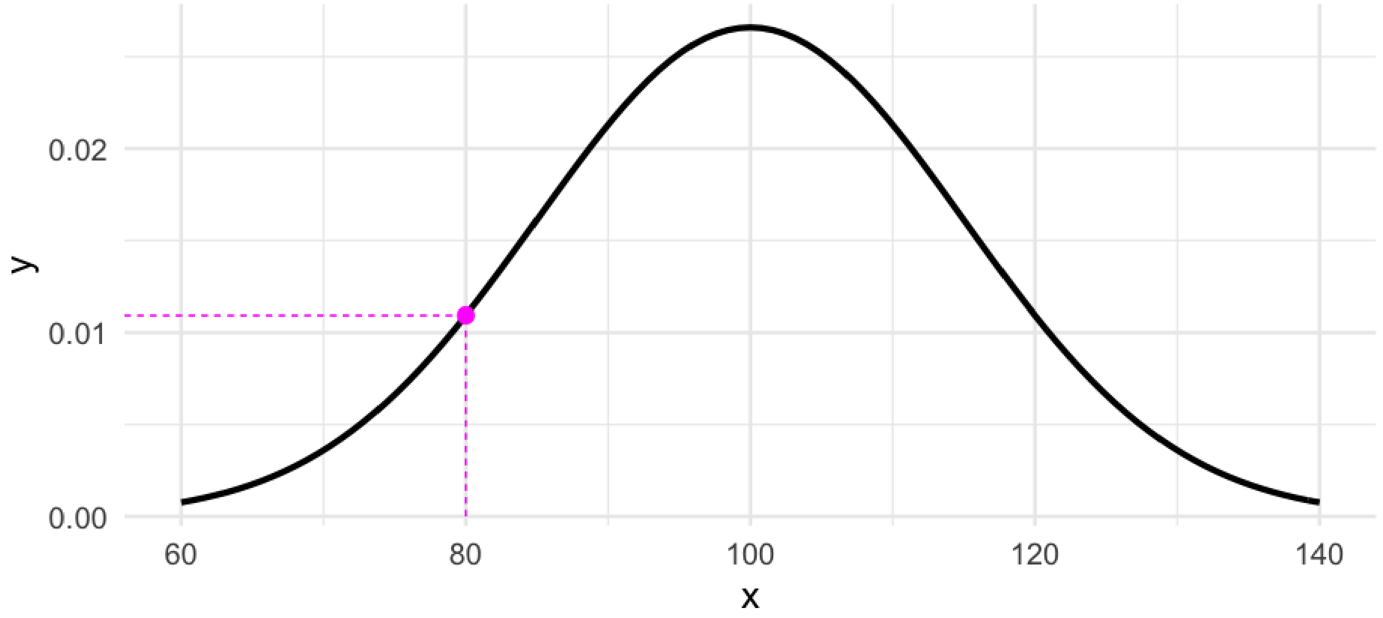
$$IQ_i \sim N(100, 15)$$



This example borrowed from TJ Mahr

Likelihood

What's the likelihood of a score of 80, assuming this distribution?



```r
dnorm(80, mean = 100, sd = 15)
```

```
## [1] 0.010934
```

Likelihood of the data

We sum the likelihood to get the overall likelihood of the data. However, this leads to very small numbers. Computationally, it's easier to sum the *log* of these likelihoods.

```
dnorm(iqs, mean = 100, sd = 15, log = TRUE)
```

```
##  [1] -4.346989 -4.515878 -3.946989 -3.769211 -4.195878 -4.269211 -5.9025
##  [8] -5.495878 -5.015878 -3.735878 -3.682544 -3.946989 -3.895878 -6.6692
## [15] -4.802544 -4.062544 -4.269211 -3.735878 -3.646989 -4.002544 -3.7069
## [22] -3.662544 -3.946989 -4.002544 -3.706989 -3.735878 -3.635878
```

```
sum(dnorm(iqs, mean = 100, sd = 15, log = TRUE))
```

```
## [1] -114.3065
```

Alternative distributions

What if we assumed the data were generated from an alternative distribution, say $IQ_i \sim N(115, 5)$?

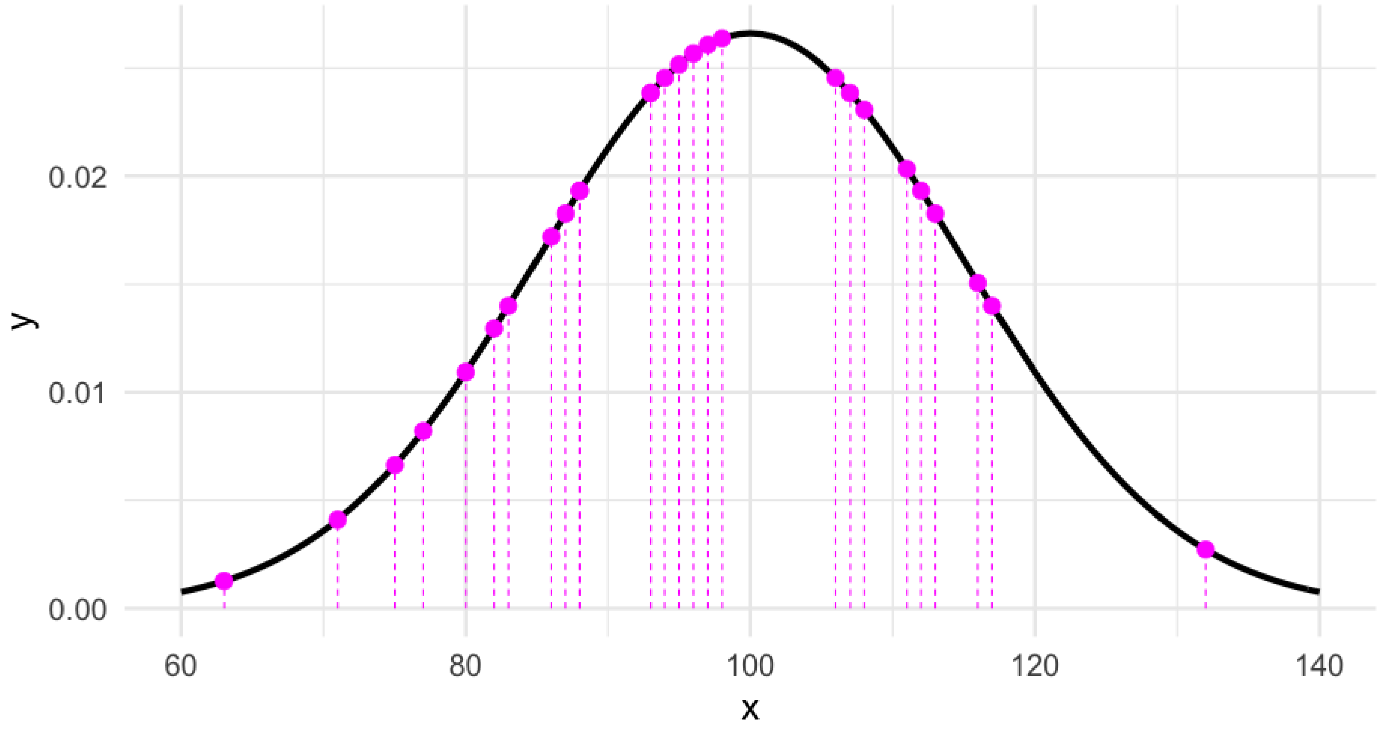
```r
sum(dnorm(iqs, mean = 115, sd = 5, log = TRUE))
```

```
## [1] -416.3662
```

The value is *much* lower. In most models, we are estimating μ and σ, and trying to find values that *maximize* the sum of the log likelihoods.

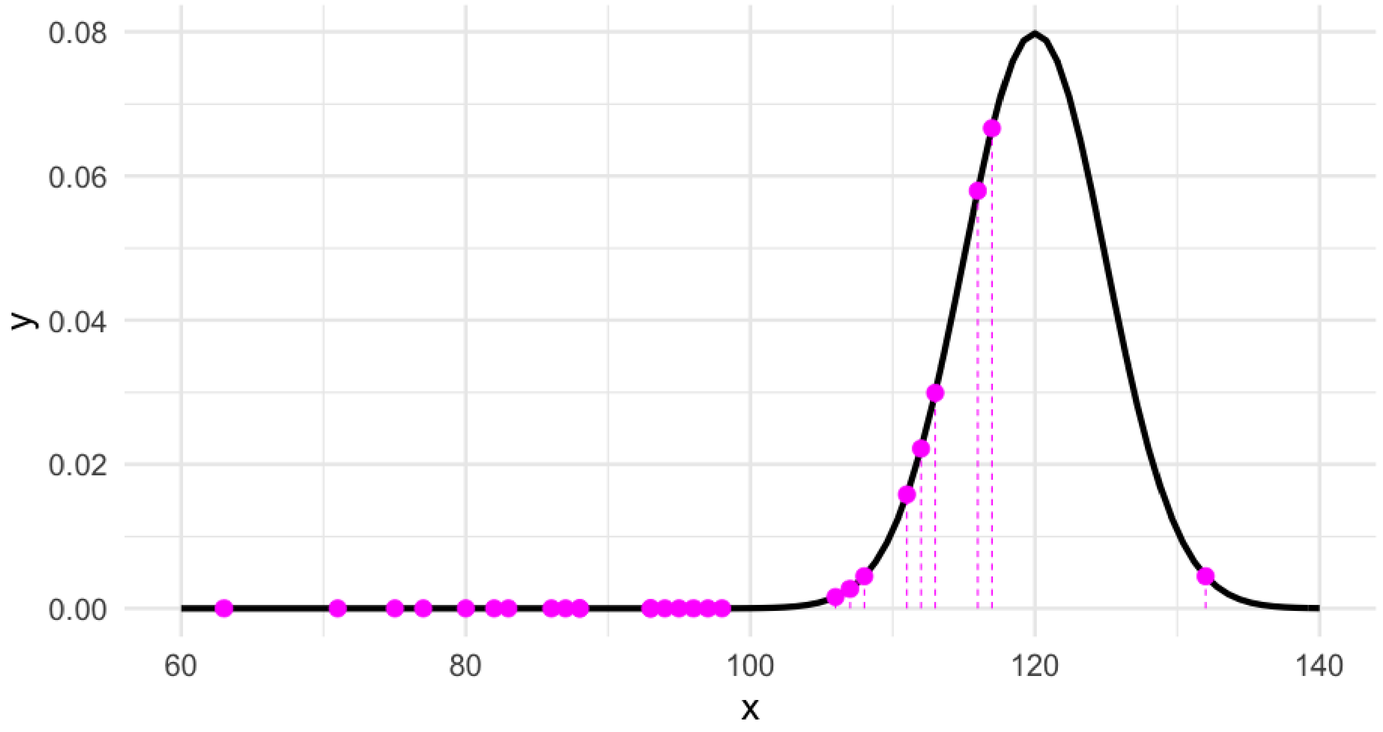
Visually

The real data generating distribution



Visually

The poorly fitting one



Non–Bayesian

In a frequentist regression model, we would find parameters that *maximize* the likelihood. Note – the distributional mean is often conditional.

This is part of why I've come to prefer notation that emphasizes the data generating process.

Example

I know we've talked about this before, but a simple linear regression model like this

```r
m <- lm(IQbio ~ class, data = carData::Burt)
```

is generally displayed like this

$$\text{IQbio} = \alpha + \beta_1(\text{class}_{\text{low}}) + \beta_2(\text{class}_{\text{medium}}) + \epsilon$$

But we could display the same thing like this

$$\begin{aligned}&\text{IQbio} \sim N(\widehat{\mu}, \widehat{\sigma}) \\ &\widehat{\mu} = \alpha + \beta_1(\text{class}_{\text{low}}) + \beta_2(\text{class}_{\text{medium}})\end{aligned}$$

Priors

Bayesian posterior

$$\textbf{posterior} = \frac{\textbf{likelihood} \times \textbf{prior}}{\textbf{average likelihood}}$$

The above is how we estimate with Bayes.

In words, it states that our updated beliefs (posterior) depend on the evidence from our data (likelihood) and our prior knowledge/conceptions/information (prior).

Our prior will shift in accordance with the evidence from the data

Basic example

Let's walk through a basic example where we're just estimating a mean. We'll assume we somehow magically know the variance. Please follow along.

First, generate some data

```r
set.seed(123)
true_data <- rnorm(50, 5, 1)
```

Grid search

We're now going to specify a grid of possible means for our data. Let's search anywhere from –3 to 12 in 0.1 intervals.

```r
grid <- tibble(possible_mean = seq(-3, 12, 0.1))
```

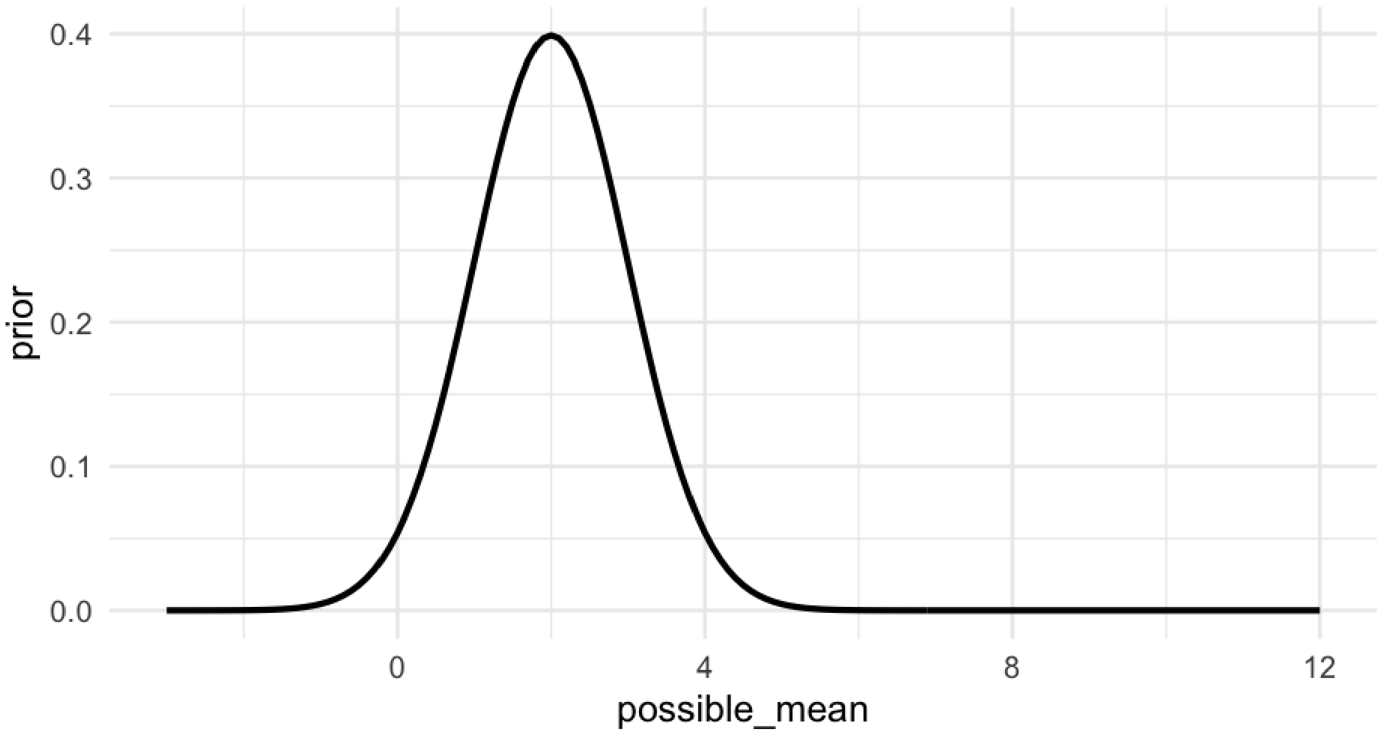
Next, we'll specify a *prior distribution*. That is – how likely do we *think* each of these possible means are?

Let's say our best guess is $\mu = 2$. Values on either side of 2 should be less likely.

```r
prior <- dnorm(grid$possible_mean, mean = 2, sd = 1)
```

Plot our prior

```
grid %>%
  mutate(prior = prior) %>%
  ggplot(aes(possible_mean, prior)) +
  geom_line()
```



Look at other priors

```r
grid %>%
  mutate(prior1 = dnorm(possible_mean, mean = 2, sd = 1),
         prior2 = dnorm(possible_mean, mean = 2, sd = 2),
         prior3 = dnorm(possible_mean, mean = 2, sd = 3)) %>%
  ggplot(aes(possible_mean)) +
  geom_line(aes(y = prior1)) +
  geom_line(aes(y = prior2), color = "cornflowerblue") +
  geom_line(aes(y = prior3), color = "firebrick")
```

Set prior

- Let's go with a fairly conservative prior, with $\mu = 2, \sigma = 3$.
- We also need to normalize it so the probability sums to 1.0

```
grid <- grid %>%
  mutate(prior = dnorm(possible_mean, mean = 2, sd = 3),
         prior = prior / sum(prior)) # normalize
```

Observe 1 data point

```r
grid <- grid %>%
  mutate(likelihood = dnorm(true_data[1], possible_mean, 2))
grid
```

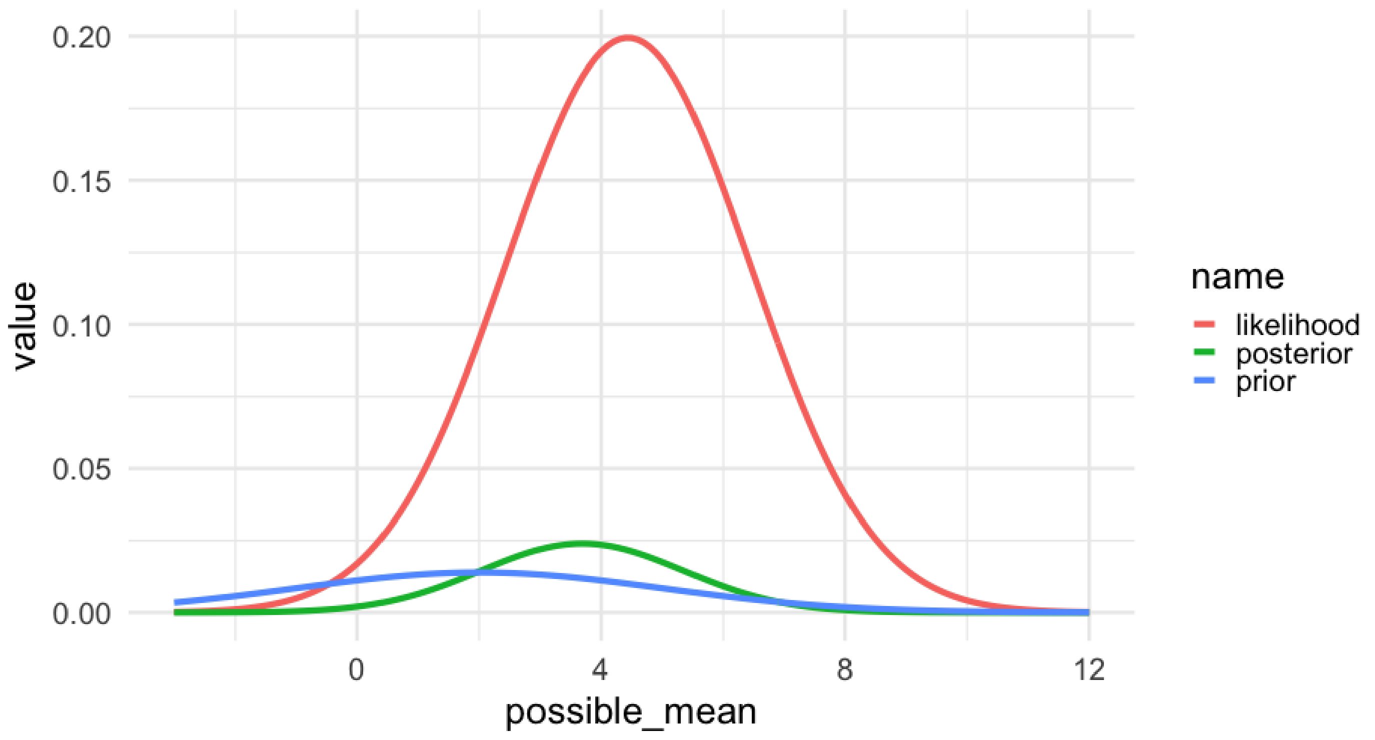
```
## # A tibble: 151 x 3
##    possible_mean       prior   likelihood
##            <dbl>       <dbl>        <dbl>
##  1          -3   0.003477802 0.0001973758
##  2          -2.9 0.003674439 0.0002374240
##  3          -2.8 0.003877883 0.0002848850
##  4          -2.7 0.004088046 0.0003409800
##  5          -2.6 0.004304813 0.0004071013
##  6          -2.5 0.004528041 0.0004848308
##  7          -2.4 0.004757554 0.0005759600
##  8          -2.3 0.004993151 0.0006825094
##  9          -2.2 0.005234594 0.0008067505
## 10          -2.1 0.005481619 0.0009512268
## # … with 141 more rows
```

Compute posterior

```r
grid <- grid %>%
  mutate(posterior = likelihood * prior,
         posterior = posterior / sum(posterior)) # normalize
```

Plot

```
grid %>%
  pivot_longer(-possible_mean) %>%
ggplot(aes(possible_mean, value)) +
  geom_line(aes(color = name))
```



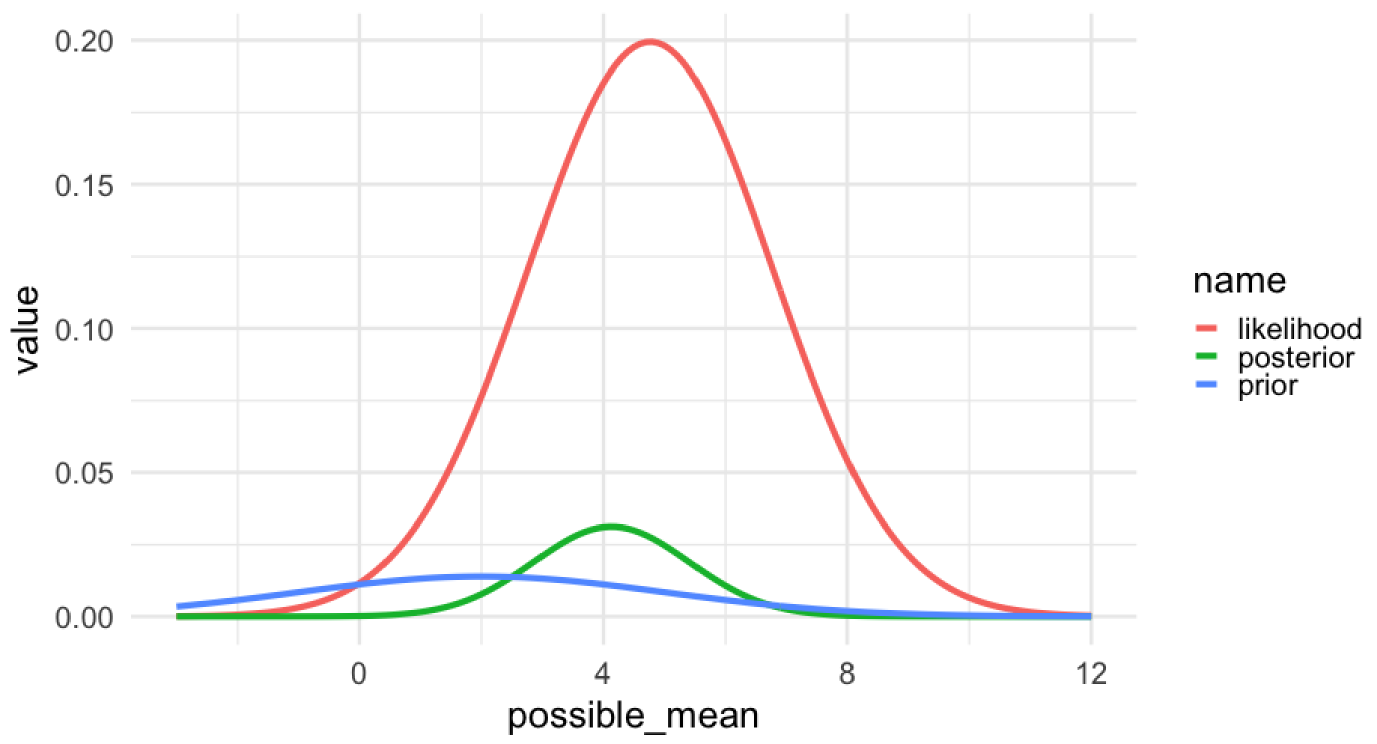
Observe a second data point

The old posterior becomes our new prior

```
grid <- grid %>%
  mutate(likelihood = dnorm(true_data[2], possible_mean, 2),
         posterior = likelihood * posterior,
         posterior = posterior / sum(posterior))
```

Plot

```r
grid %>%
  pivot_longer(-possible_mean) %>%
ggplot(aes(possible_mean, value)) +
  geom_line(aes(color = name))
```

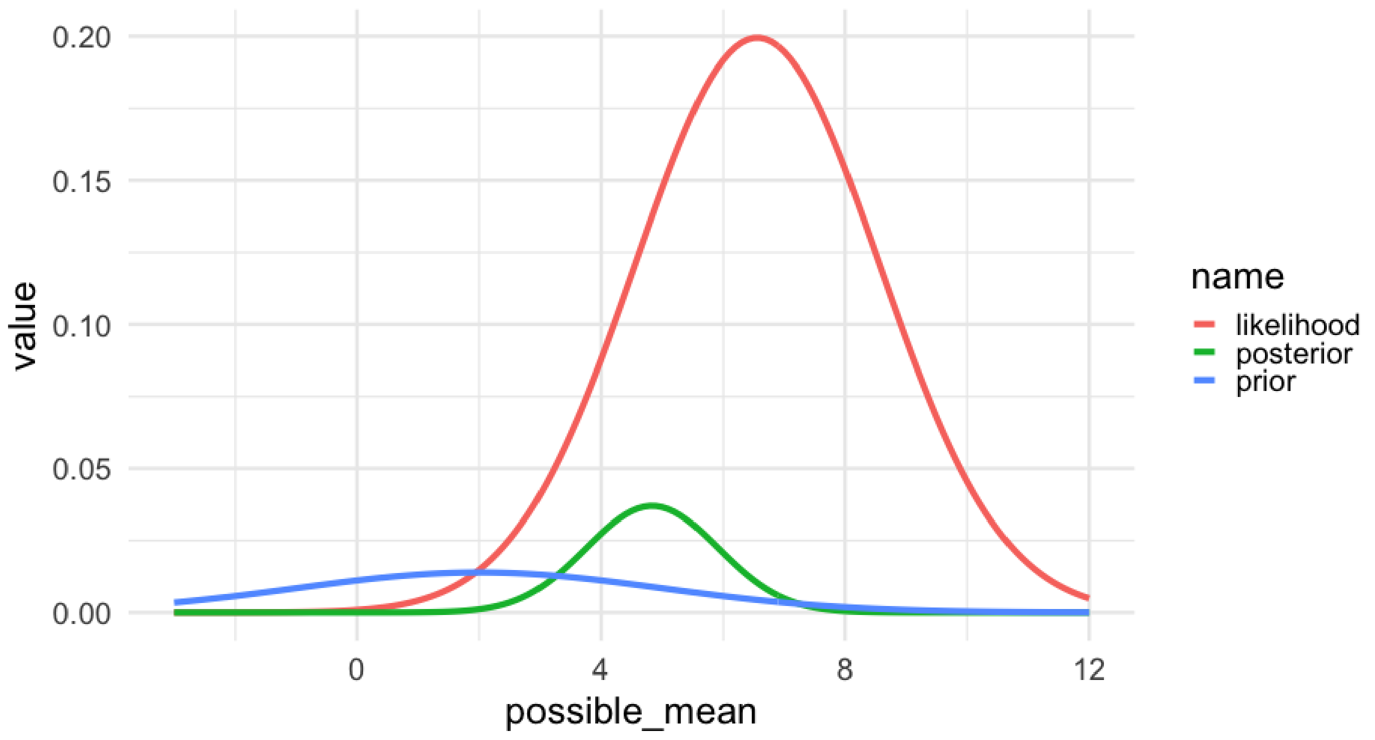


Observe a third data point

```r
grid <- grid %>%
  mutate(likelihood = dnorm(true_data[3], possible_mean, 2),
         posterior = likelihood * posterior,
         posterior = posterior / sum(posterior))
```

Plot

```r
grid %>%
  pivot_longer(-possible_mean) %>%
ggplot(aes(possible_mean, value)) +
  geom_line(aes(color = name))
```

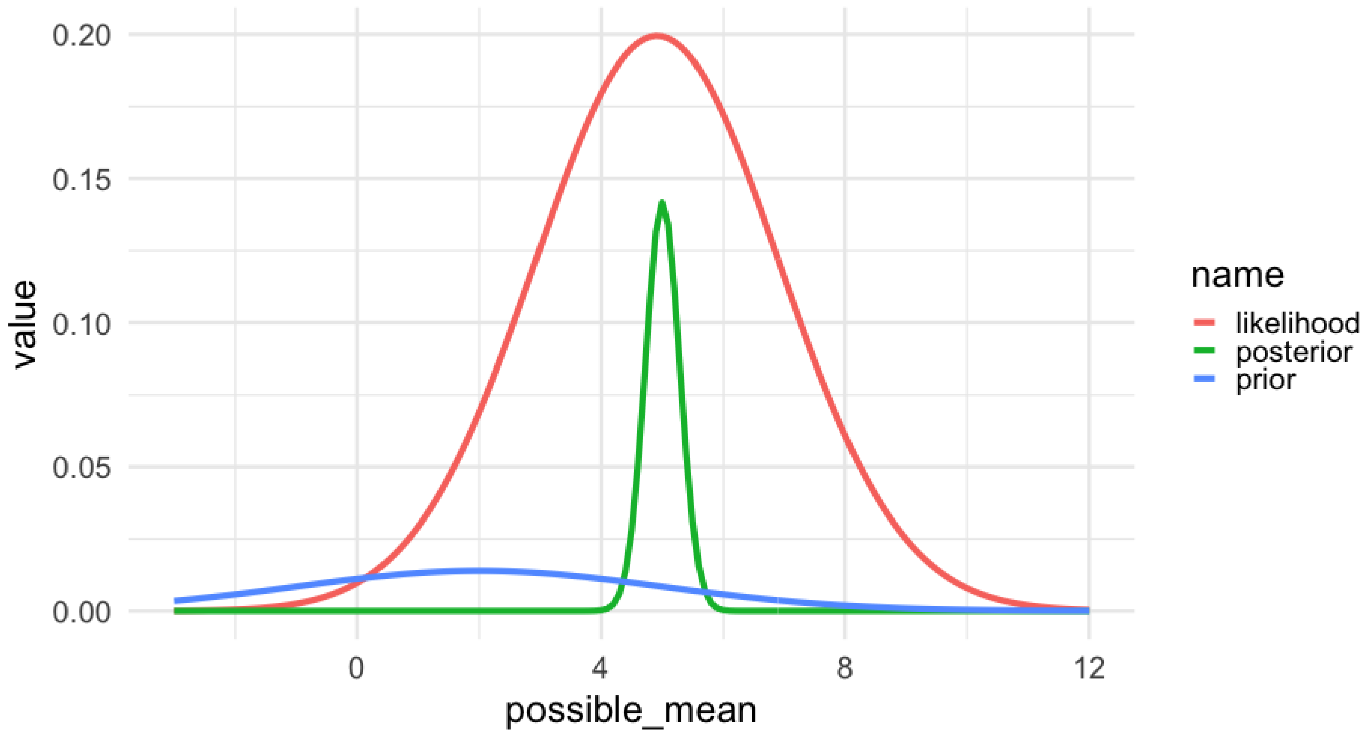


All the data

```r
grid <- grid %>%
  mutate(prior = dnorm(grid$possible_mean, mean = 2, sd = 3),
         prior = prior / sum(prior),
         posterior = prior) # best guess before seeing data

for(i in seq_along(true_data)) {
  grid <- grid %>%
    mutate(likelihood = dnorm(true_data[i], possible_mean, 2),
           posterior = likelihood * posterior,
           posterior = posterior / sum(posterior))
}
```

```r
grid %>%
  pivot_longer(-possible_mean) %>%
ggplot(aes(possible_mean, value)) +
  geom_line(aes(color = name))
```



Posterior

- We can summarize our posterior distribution
- This is a fundamental difference between Bayesian & frequentist approaches
  - In Bayes, our data is assumed fixed, our parameters random
  - In frequentist, our data is assumed random, our parameters fixed

Most likely?

```
grid %>%
  filter(posterior == max(posterior))
```

```
## # A tibble: 1 x 4
##   possible_mean       prior likelihood posterior
##           <dbl>       <dbl>      <dbl>     <dbl>
## 1             5 0.008459494  0.1992979 0.1416204
```

Sampling

- Now that we have a posterior distribution, we can sample from it to help us with inference

- Each possible mean should be sampled in accordance with its probability specified by the posterior.

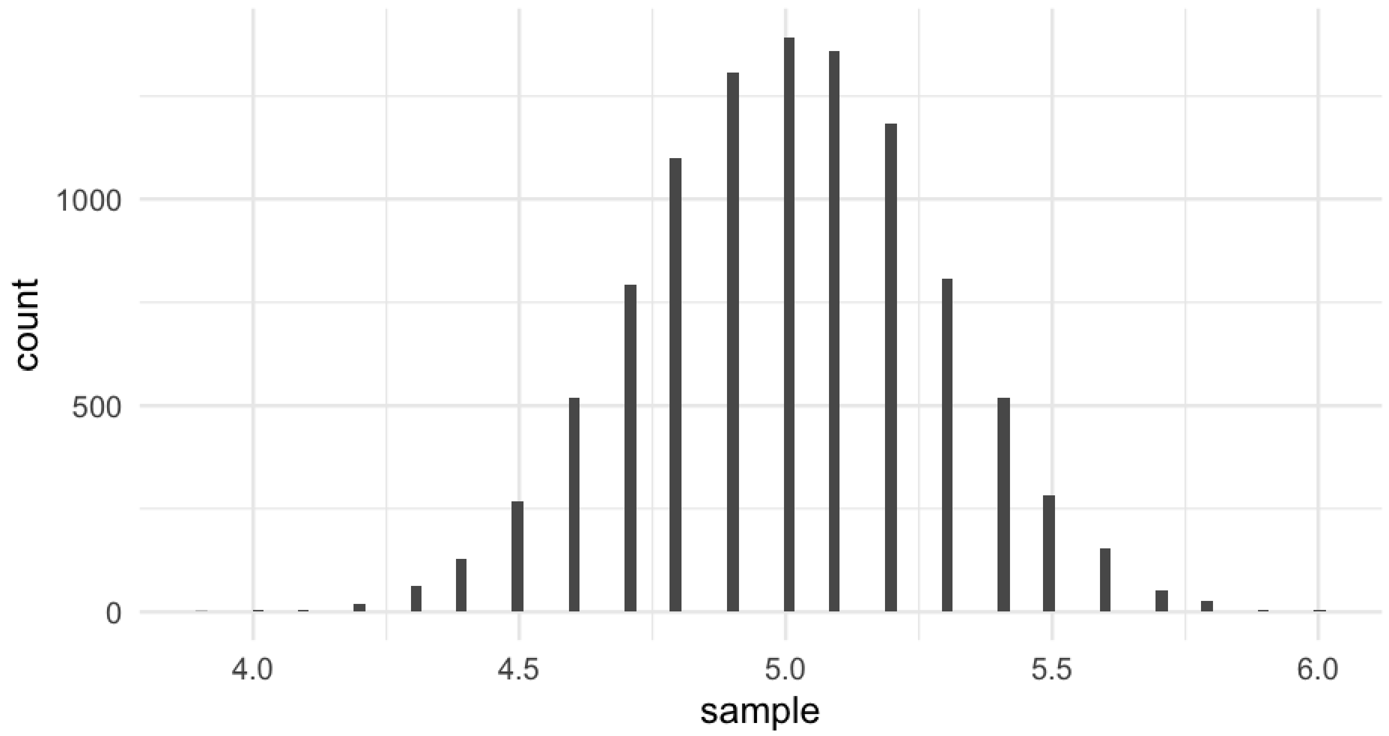
- Let's draw 10,000 samples

```
posterior_samples <- sample(grid$possible_mean,
                            size = 10000,
                            replace = TRUE,
                            prob = grid$posterior)
```

Inference

First, let's plot the samples

```r
ggplot(data.frame(sample = posterior_samples), aes(sample)) +
  geom_histogram(bins = 100)
```



Central tendency

```r
mean(posterior_samples)
```

```
## [1] 5.00441
```

```r
median(posterior_samples)
```

```
## [1] 5
```

Spread

```r
sd(posterior_samples)
```

```
## [1] 0.278466
```

Credible intervals

Let's compute an 80% credible interval

```
tibble(posterior_samples) %>%
  summarize(ci_80 = quantile(posterior_samples, c(0.1, 0.9)))
```

```
## # A tibble: 2 x 1
##   ci_80
##   <dbl>
## 1   4.6
## 2   5.4
```

What's the chance the "true" mean is less than 4.8?

```
sum(posterior_samples < 4.8) / length(posterior_samples) * 100
```

```
## [1] 18.05
```

Ranges

What's the probability the "true" mean is between 5.2 and 5.5?

```r
sum(posterior_samples >= 5.2 & posterior_samples <= 5.5) /
  length(posterior_samples) * 100
```

```
## [1] 27.94
```

Greater than 4.5?

```r
sum(posterior_samples > 4.5) / length(posterior_samples) * 100
```

```
## [1] 95.05
```

Note this is much more natural than frequentist statistics

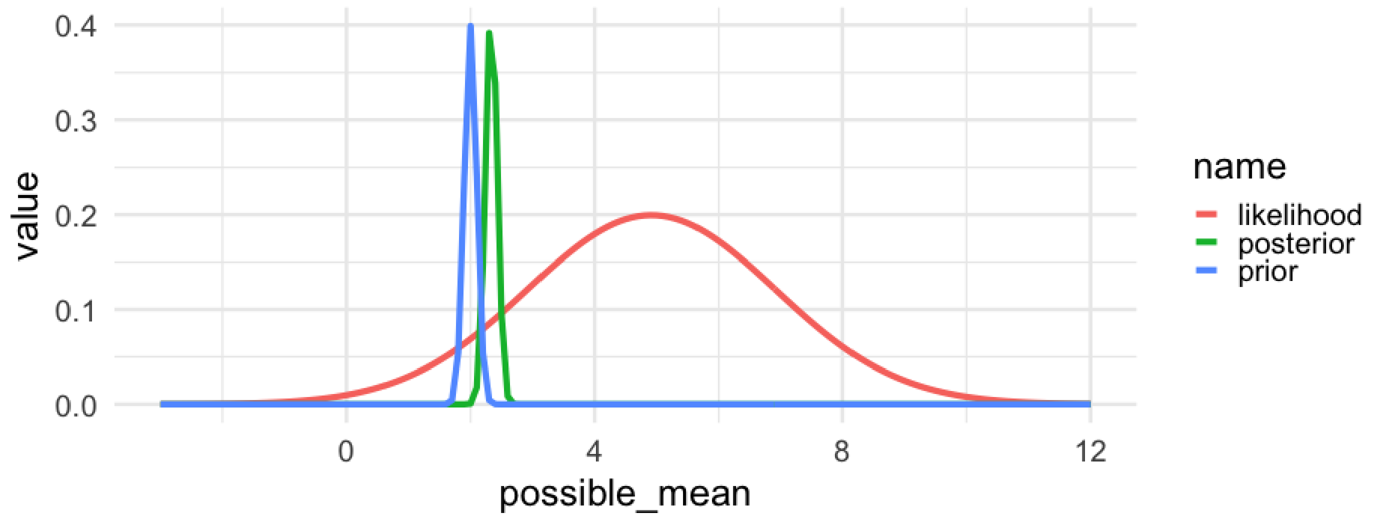
Change our prior

Let's try again with a tighter prior

```r
grid <- grid %>%
  mutate(prior = dnorm(grid$possible_mean, mean = 2, sd = 0.1),
         prior = prior / sum(prior),
         posterior = prior) # best guess before seeing data

for(i in seq_along(true_data)) {
  grid <- grid %>%
    mutate(likelihood = dnorm(true_data[i], possible_mean, 2),
           posterior = likelihood * posterior,
           posterior = posterior / sum(posterior))
}
```

```r
grid %>%
  pivot_longer(-possible_mean) %>%
ggplot(aes(possible_mean, value)) +
  geom_line(aes(color = name))
```



```r
grid %>%
  filter(posterior == max(posterior))
```

```
## # A tibble: 1 x 4
##   possible_mean       prior likelihood posterior
##           <dbl>       <dbl>      <dbl>     <dbl>
## 1           2.3 0.004431848 0.08476010 0.3915258
```

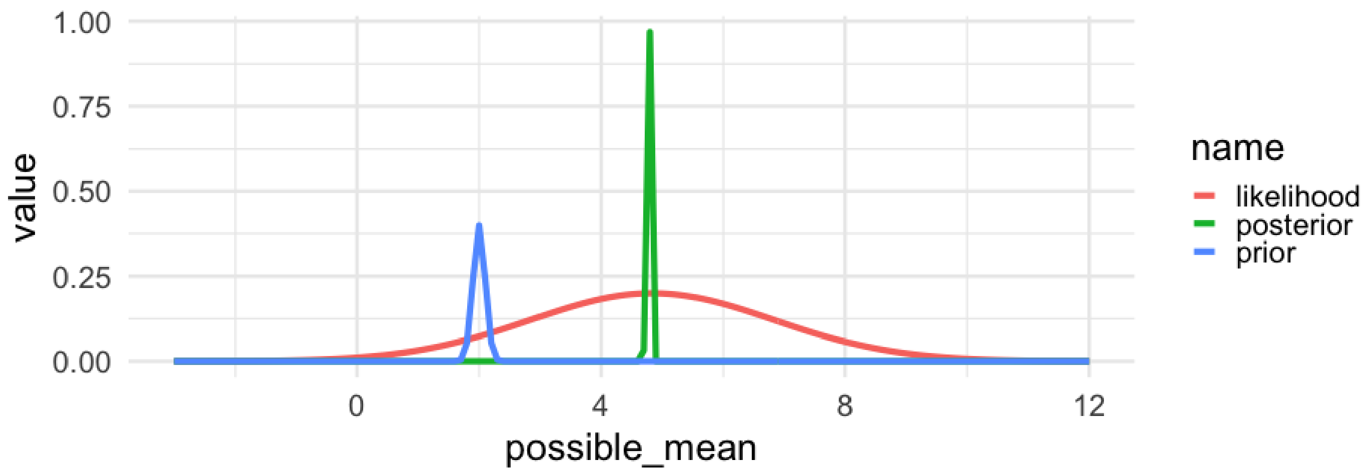
More data

Same thing, but this time with tons of data

```r
true_data <- rnorm(5000, 5, 1)
grid <- grid %>%
  mutate(prior = dnorm(grid$possible_mean, mean = 2, sd = 0.1),
         prior = prior / sum(prior),
         posterior = prior) # best guess before seeing data

for(i in seq_along(true_data)) {
  grid <- grid %>%
    mutate(likelihood = dnorm(true_data[i], possible_mean, 2),
           posterior = likelihood * posterior,
           posterior = posterior / sum(posterior))
}
```

```
grid %>%
  pivot_longer(-possible_mean) %>%
ggplot(aes(possible_mean, value)) +
  geom_line(aes(color = name))
```



```
grid %>%
  filter(posterior == max(posterior))
```

```
## # A tibble: 1 x 4
##   possible_mean         prior likelihood posterior
##           <dbl>         <dbl>      <dbl>     <dbl>
## 1           4.8 2.277577e-171  0.1994389 0.9678310
```

Taking a step back

- The purpose of the prior is to include *what you already know* into your analysis
- The strength of your prior should depend on your prior research
- Larger samples will overwhelm priors quicker, particularly if they are diffuse
- Think through the lens of updating your prior beliefs
- This whole framework is quite different, but also gives us a lot of advantages in terms of probability interpretation, as we'll see

Bayes for regression

More complicated

- Remember our posterior is defined by

$$\textbf{posterior} = \frac{\textbf{likelihood} \times \textbf{prior}}{\textbf{average likelihood}}$$

When we just had a single parameter to estimate, μ, this was tractable with grid search.

With even simple linear regression, however, we have three parameters: α, β, and σ

Our Bayesian model then becomes considerably more complicated:

$$P(\alpha, \beta, \sigma \mid x) = \frac{P(x \mid \alpha, \beta, \sigma)\, P(\alpha, \beta, \sigma)}{\iiint P(x \mid \alpha, \beta, \sigma)\, P(\alpha, \beta, \sigma)\, d\alpha\, d\beta\, d\sigma}$$

Estimation

Rather than trying to compute the integrals, we *simulate* observations from the joint posterior distribution.

This sounds a bit like magic – how do we do this?

Multiple different algorithms, but all use some form of Markov–Chain Monte–Carlo sampling

Conceptually

- Imagine the posterior as a hill
- We start with the parameters set to random numbers
    - Estimate the posterior with this values (our spot on the hill)
- Use information from the prior sample to determine whether and how to change the current parameter values
- Try to "walk" around in a way to a complete "picture" of the hill from the samples
- Use these samples as your posterior distribution

Metropolis–Hastings

We will use is called the Metropolis–Hastings algorithm:

- Compute a candidate "step" for the parameters
- Calculate an acceptance ratio $\alpha = f(x')/f(x_t)$. This will fall between 0 and 1.
- Generate number, u, from a random uniform distribution between 0 and 1
    - if $u \leq \alpha$, accept the candidate
    - if $u \geq \alpha$, reject the candidate

Complicated, but conceptually we're trying to sample the joint posterior distribution in a way that conforms with the

Script

- The `mh-alg.R` script works through the Metropolis–Hastings algorithm to get samples from the joint posterior of a simple linear regression model.
- The data are simulated, so we know the true values
- It's complicated, and not required at all, but it's there for you if you want more info

Next time

Continue discussing Bayes – more emphasis on model results & plotting

Fit and interpret multilevel logistic regression models