CGS698C, Module 2: Model building in the Bayesian framework

Himanshu Yadav

2024-05-30

Contents

1 *Using Bayes' theorem for statistical inference*

Suppose that an outcome x observed in an experiment is assumed to come from a normal distribution, such that

$f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

where $f(x)$ is the probability density function; $f(x)$ assigns the probability density value to the outcome x conditional on the parameters mean μ and variance σ^2 of the normal distribution. The probability density of x conditional on μ and σ^2 can be written as,

$p(x|\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

The goal of statistical inference is figure out what value(s) of μ and σ^2 have generated the observed outcome x.

We know the probability density of obtaining x given μ and σ^2, can we calculate the probability density of (a range of) values μ and σ^2 conditional on the observed outcome x?

$p(\mu,\sigma^2|x) =$?

Using Bayes' theorem,

$p(\mu,\sigma^2|x) = \frac{p(x|\mu,\sigma^2)\cdot p(\mu,\sigma^2)}{\int\int p(x|\mu,\sigma^2)\cdot p(\mu,\sigma^2)\, d\mu\, d\sigma^2}$

More generally, suppose the observed outcome x is assumed to be a value of the random variable X whose probability density function is $f(x;\theta)$; $f(x;\theta)$ assigns a probability density value to x conditional on a parameter θ. The probability density of x given the parameter θ is given by $p(x|\theta)$.

Our goal is to infer what value(s) of the parameter θ has generated the given (observed) datapoint x.

$$p(\theta|x) = \frac{p(x|\theta)\cdot p(\theta)}{\int p(x|\theta)\cdot p(\theta)\, d\theta} \tag{1}$$

The term $p(x|\theta)$ is called the **likelihood function**, $p(\theta)$ is called the **prior distribution** of θ, and $p(\theta|x)$ is called the **posterior distribution** of θ.

Note: When $f(x;\theta)$ is seen as a function of x, it is called a probability density function; and when $f(x;\theta)$ is seen as a function of θ, it is called a likelihood function, also denoted by $\mathcal{L}(\theta|x)$.

Imagine you collect reading times data in an experiment. Based on your sample of reading times, you try to infer the underlying reality that has generated this observed sample. How do we make this inference using Bayesian modeling?

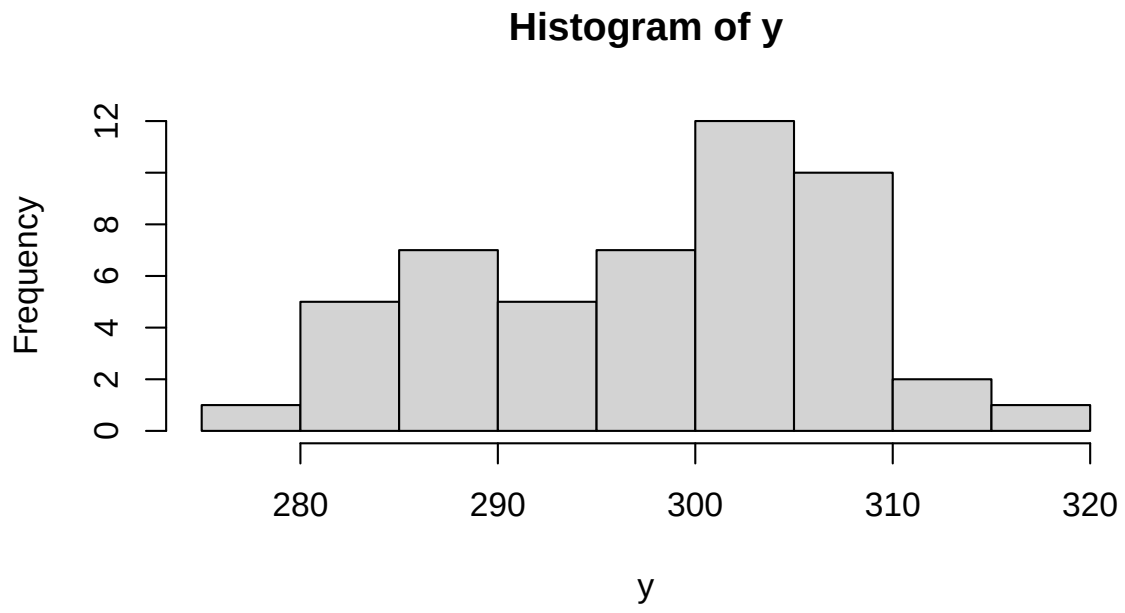
2 *Unknown reality*

Suppose the reality (the true generative process) that underlies the observed reading times is described as follows.

Each individual reading time is (independently) generated by a probability density function $f(x) = \frac{1}{10\sqrt{2\pi}} e^{-\frac{(x-300)^2}{200}}$

3 *Observations*

```
y <- rnorm(50,300,10)
hist(y)
```



4 *Assumptions about the generative process*

(1) **The likelihood assumption:** The observed reading times are normally distributed.

$$\mathcal{L}(\mu, \sigma | y) = \frac{1}{(\sigma\sqrt{2\pi})^n} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu)^2}$$

where $\sigma = 10$

(2) **The prior assumptions:** Based on your prior knowledge or beliefs, suppose you can assume the following about the two parameters of interest.

$\mu \sim Normal(350, 50)$

$\sigma = 10$

5 The likelihood function and the prior distributions

5.1 The likelihood

Each datapoint y_i has a normal distribution:

$$f(y_i; \mu, \sigma) = \frac{1}{(\sigma\sqrt{2\pi})^n} e^{-\frac{1}{2\sigma^2}(y_i-\mu)^2}$$

We can write the above probability density function as a likelihood function. A likelihood function is a function of μ and σ when the data y_i is fixed. The likelihood function is given by

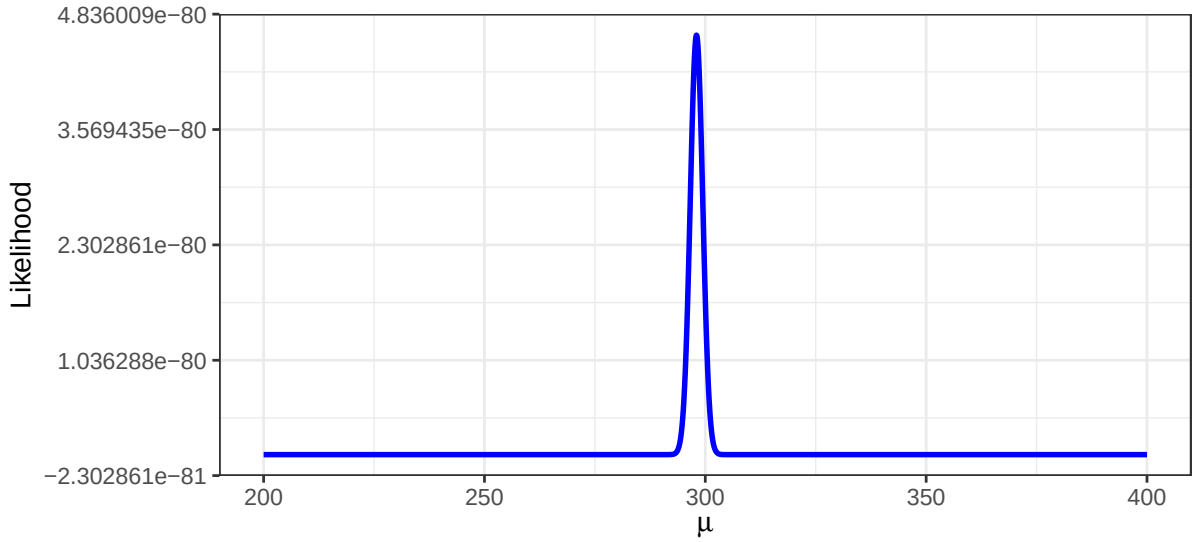
$$\mathcal{L}(\mu, \sigma|y_i) = \frac{1}{(\sigma\sqrt{2\pi})^n} e^{-\frac{1}{2\sigma^2}(y_i-\mu)^2}$$

Observed data: $y_1, y_2, y_3, ..., y_n$

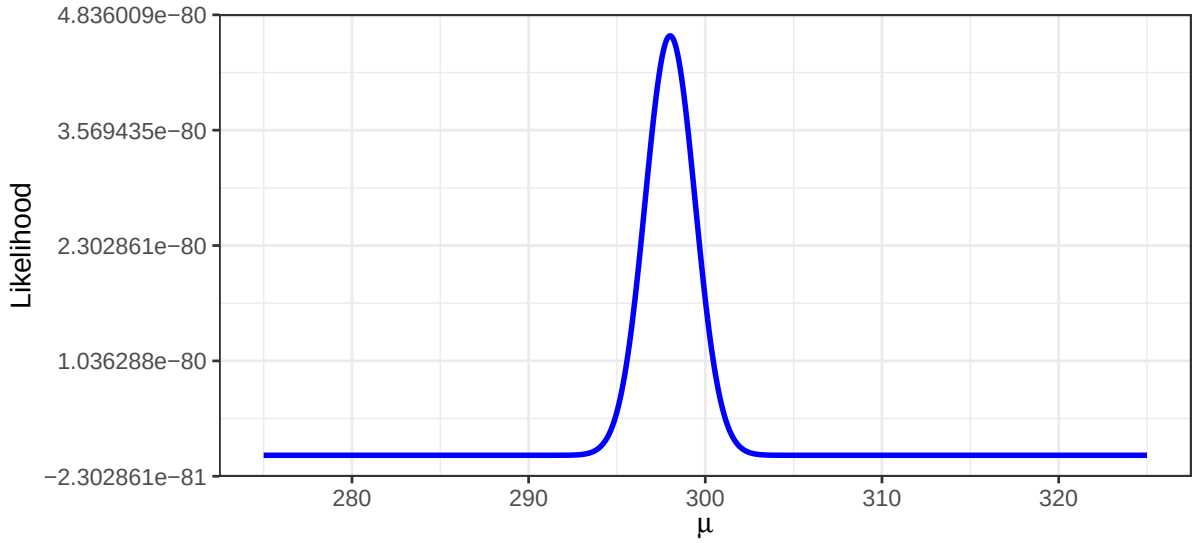
$$\mathcal{L}(\mu, \sigma|y) = \frac{1}{(\sigma\sqrt{2\pi})^n} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu)^2}$$

where $\sigma = 10$

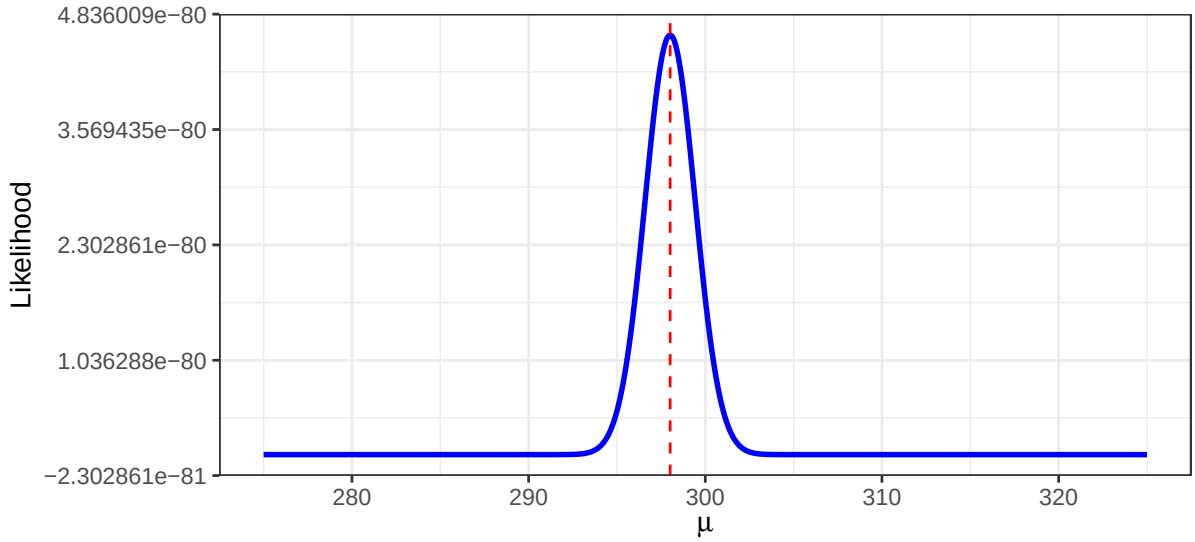
```
sigma <- 10
mu <- seq(from=200,to=400,by=0.05)
likelihoods <- data.frame(mu=mu)
likelihoods$lkl <- NA
for(i in 1:length(mu)){
  likelihoods$lkl[i] <- prod(dnorm(y,mu[i],sd=10))
}

ggplot(likelihoods,aes(x=mu,y=lkl))+geom_line(size=1,color="blue")+
  theme_bw()+xlab(expression(mu))+ylab("Likelihood")
```




```
ggplot(likelihoods,aes(x=mu,y=lkl))+geom_line(size=1,color="blue")+
  theme_bw()+xlab(expression(mu))+ylab("Likelihood")+
  scale_x_continuous(limits = c(275,325))
```




```
ggplot(likelihoods,aes(x=mu,y=lkl))+geom_line(size=1,color="blue")+
  theme_bw()+xlab(expression(mu))+ylab("Likelihood")+
  scale_x_continuous(limits = c(275,325))+
  geom_vline(xintercept = mean(y),color="red",linetype="dashed")
```



5.2 *The priors*

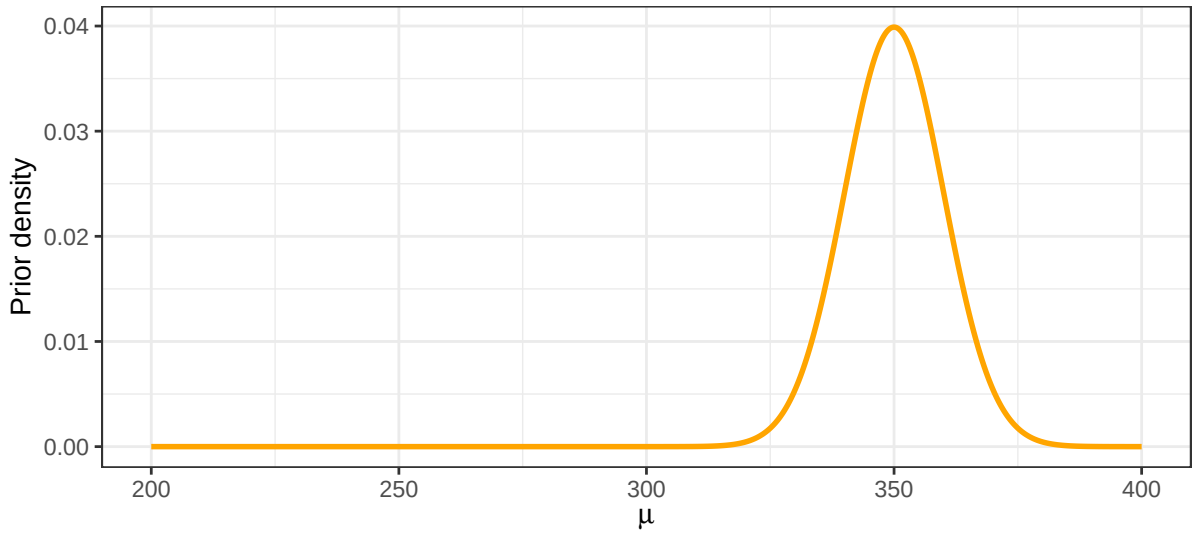
The prior density of σ and μ is given by

$$p(\sigma) = \begin{Bmatrix} 1 & \text{when} \;\; \sigma = 10 \\ 0 & \text{when} \;\; \sigma \neq 10 \end{Bmatrix}$$

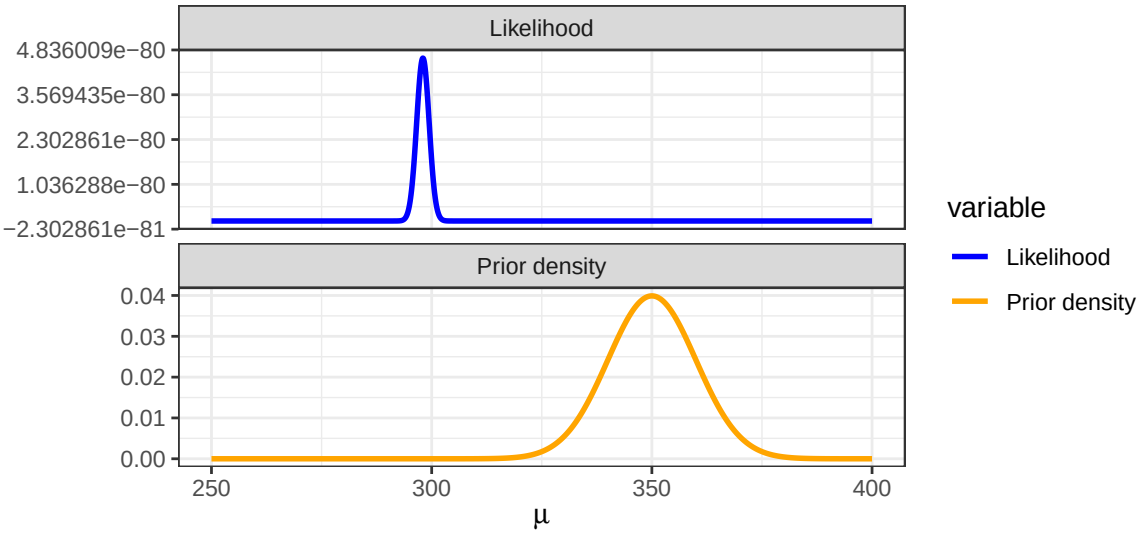
$$p(\mu) = \frac{1}{\sigma_0\sqrt{2\pi}} e^{-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}}$$

where $\mu_0 = 350$ and $\sigma_0 = 50$.

```
likelihoods$prior_density <- NA
for(i in 1:length(mu)){
  likelihoods$prior_density[i] <- dnorm(mu[i],mean=350,sd=10)
}

ggplot(likelihoods,aes(x=mu,y=prior_density))+geom_line(size=1,color="orange")+
  theme_bw()+xlab(expression(mu))+ylab("Prior density")
```




```
df.lkl_prior <- melt(likelihoods,id=c("mu"))
df.lkl_prior$variable <- ifelse(df.lkl_prior$variable=="lkl","Likelihood","Prior density")

ggplot(df.lkl_prior,aes(x=mu,y=value,color=variable))+geom_line(size=1)+
  theme_bw()+xlab(expression(mu))+ylab("")+
  scale_x_continuous(limits = c(250,400))+
  facet_wrap(~variable,scales = "free_y",ncol = 1)+
  scale_color_manual(values = c("blue","orange"))
```



6 *Model checking*

The model can be described using the following statements.

$$y_i \sim Normal(\mu, \sigma)$$

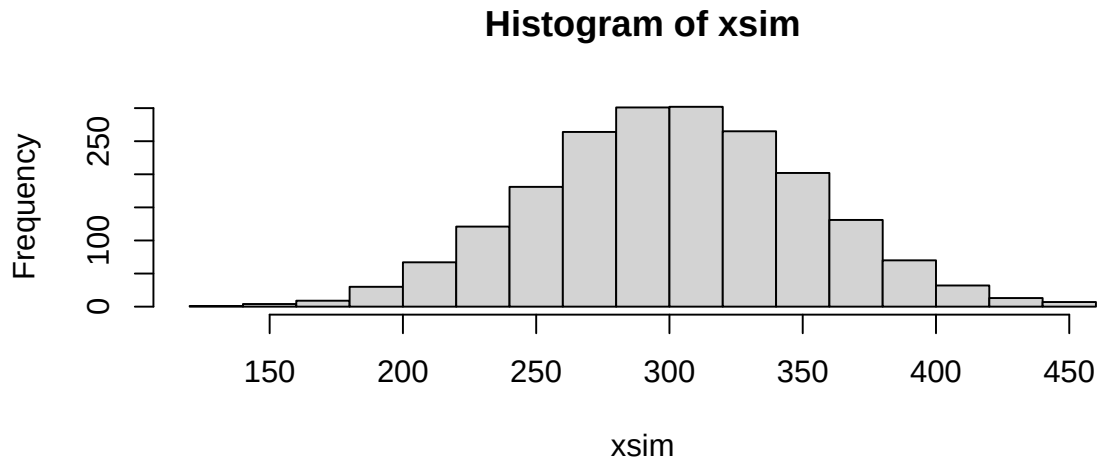
$$\mu \sim Normal(300, 50)$$

$$\sigma = 10$$

6.1 Simulating data from the model

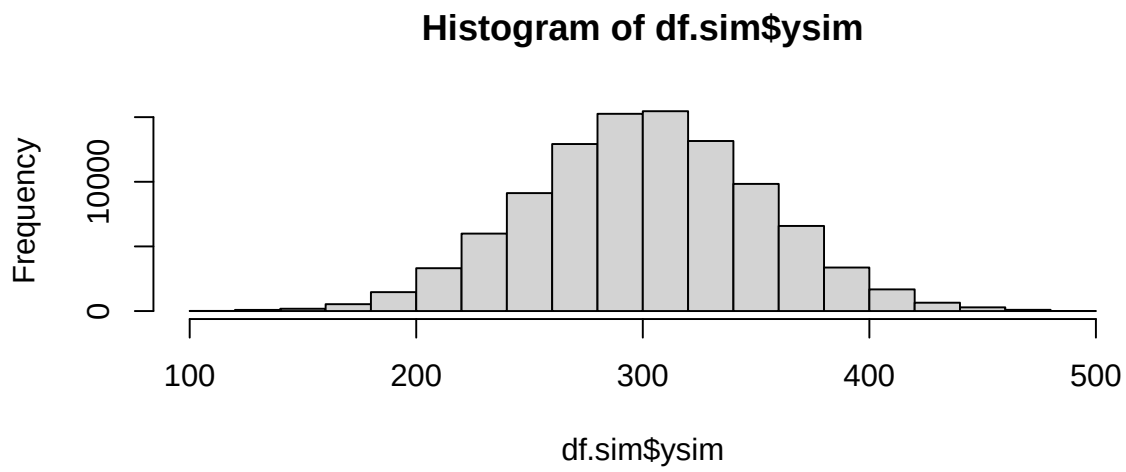
1. Sample a lot of values for the parameters μ and σ from their priors
2. Generate data from the model for each set of parameter values

```
# Sample from the priors
mu <- rnorm(2000,300,50)
sigma <- rep(10,2000)

# Create a dataframe to store the simulated data

# One way is to simply generate a datapoint
# corresponding to each value of mu
xsim <- rep(NA,length(mu))
for(i in 1:length(mu)){
  xsim[i]<- rnorm(1,mu[i],sd=sigma[i])
}
hist(xsim)
```

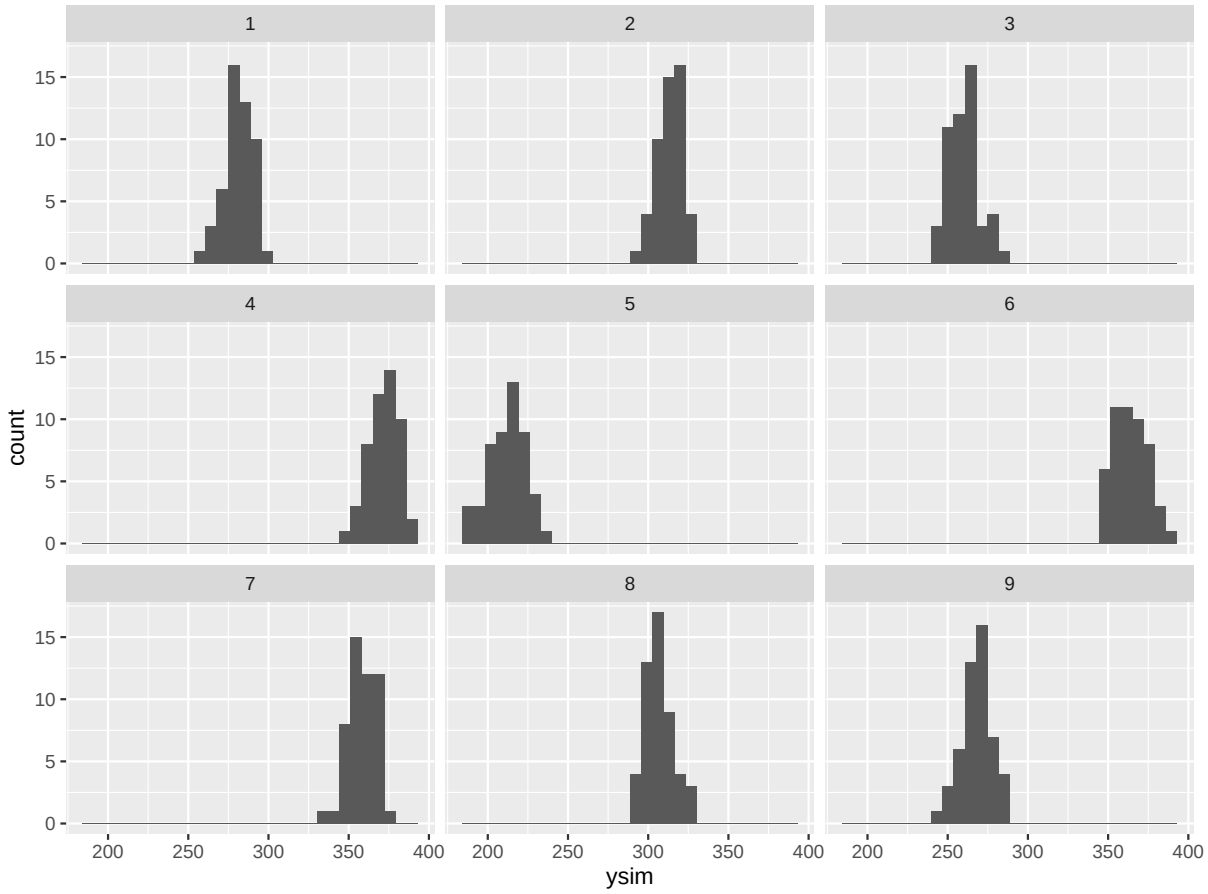



```
# But we want to make the simulated data
# as comparable as we can to the observed data
```

```r
# Our observed data y contained 50 observations
# So, we should simulate samples containing 50 observations
# for each value of mu

# Thus, we need to generate 2000 samples
# each containing N observations (datapoints)

N <- 50 # I will keep it same as the number of observations in our data y

df.sim <- data.frame(sample = rep(1:2000,each=N),
                     mu=rep(mu,each=N),sigma=rep(sigma,each=N),
                     observation = rep(1:N,2000))

df.sim$ysim <- NA
for(i in 1:length(mu)){
  df.sim[df.sim$sample==i,]$ysim <- rnorm(N,mean=mu[i],sd=sigma[i])
}

hist(df.sim$ysim)
```



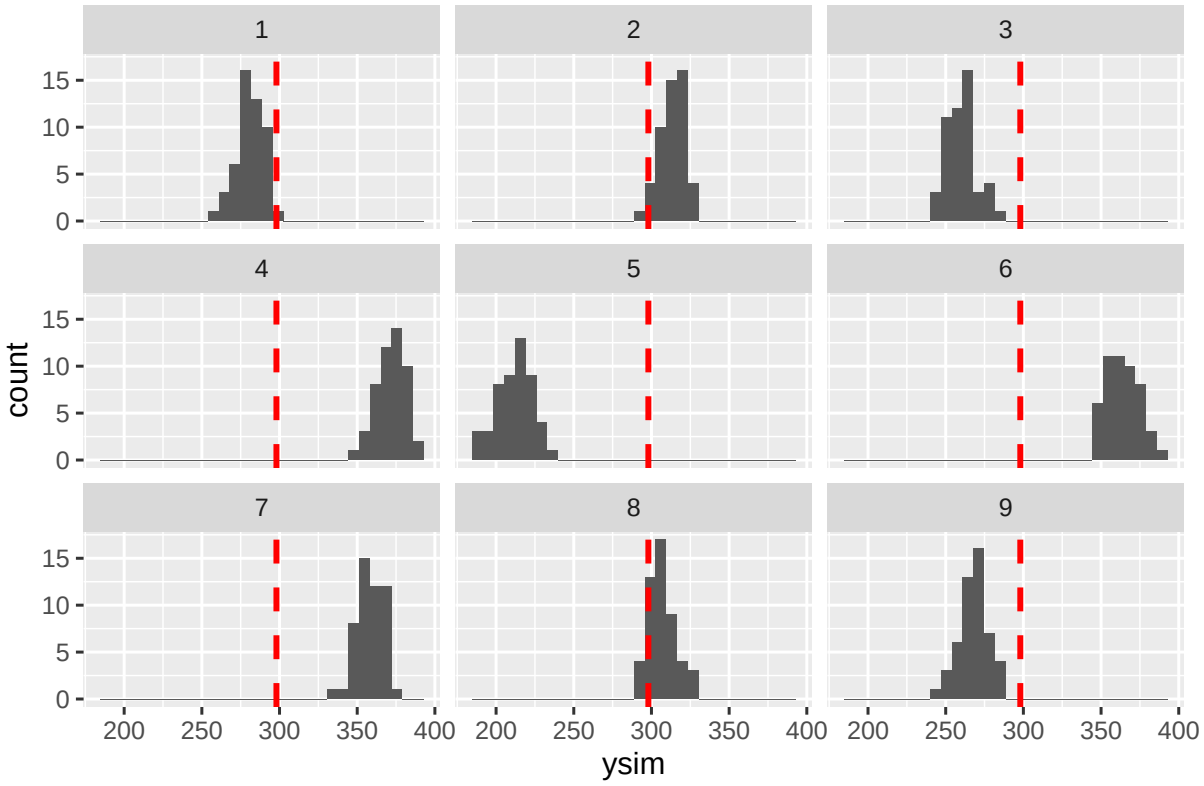

```r
ggplot(subset(df.sim,sample<10),aes(x=ysim))+geom_histogram()+facet_wrap(~sample)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



6.2 *Prior predictive checks*

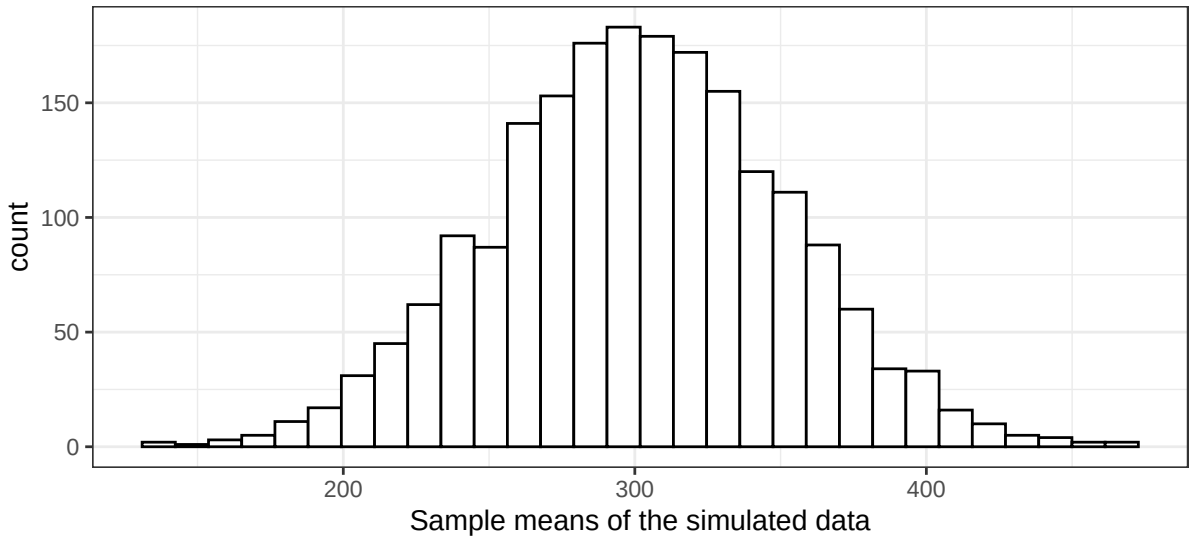
```
ggplot(subset(df.sim,sample<10),aes(x=ysim))+geom_histogram()+
  facet_wrap(~sample)+
  geom_vline(xintercept = mean(y),color="red",linetype="dashed",
             size=1)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



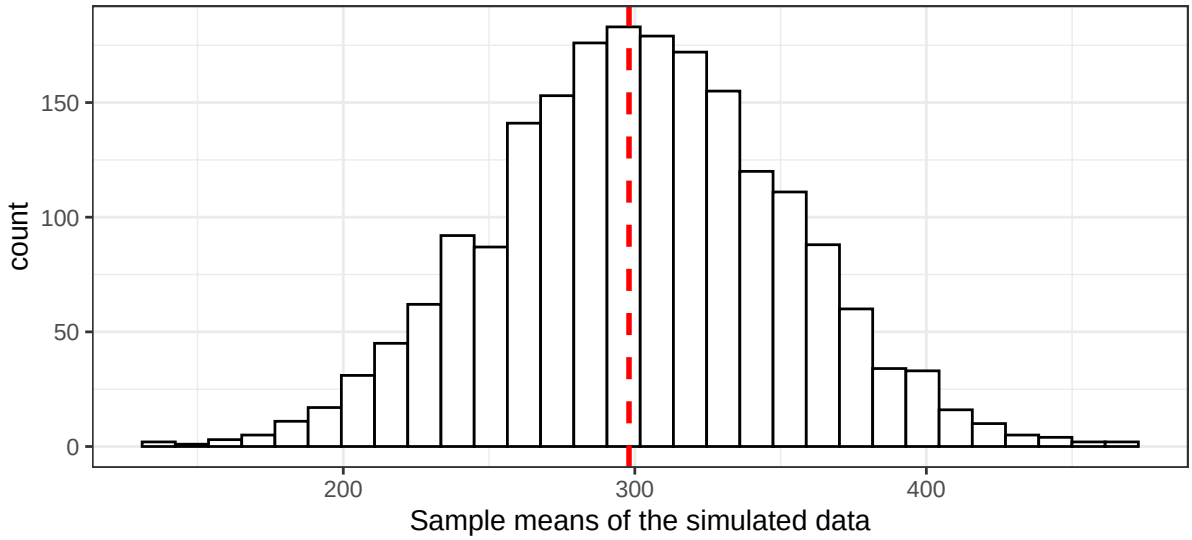

```
df.sim.summary <- df.sim %>% group_by(sample) %>%
  summarise(meanRT=mean(ysim),sdRT=sd(ysim))

ggplot(df.sim.summary,aes(x=meanRT))+
  geom_histogram(fill="white",color="black")+
  theme_bw()+xlab("Sample means of the simulated data")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(df.sim.summary,aes(x=meanRT))+
  geom_histogram(fill="white",color="black")+
  theme_bw()+xlab("Sample means of the simulated data")+
  geom_vline(xintercept = mean(y),color="red",
             linetype="dashed",size=1)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

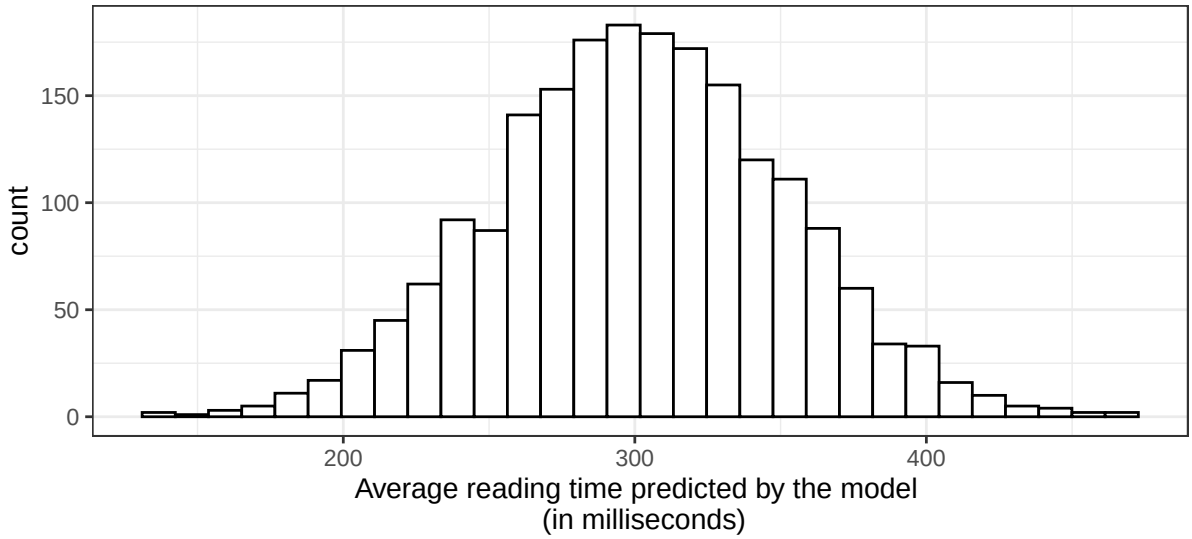


6.3 Prior predictions of the model

```
# You may need either complete simulated data for prior predictions
# Or just a summary statistic of the simulated samples
```

```
# Prior predictions of our model

ggplot(df.sim.summary,aes(x=meanRT))+
  geom_histogram(fill="white",color="black")+
  theme_bw()+
  xlab("Average reading time predicted by the model \n (in milliseconds)")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



7 Parameter estimation

7.1 Unnormalized posterior density

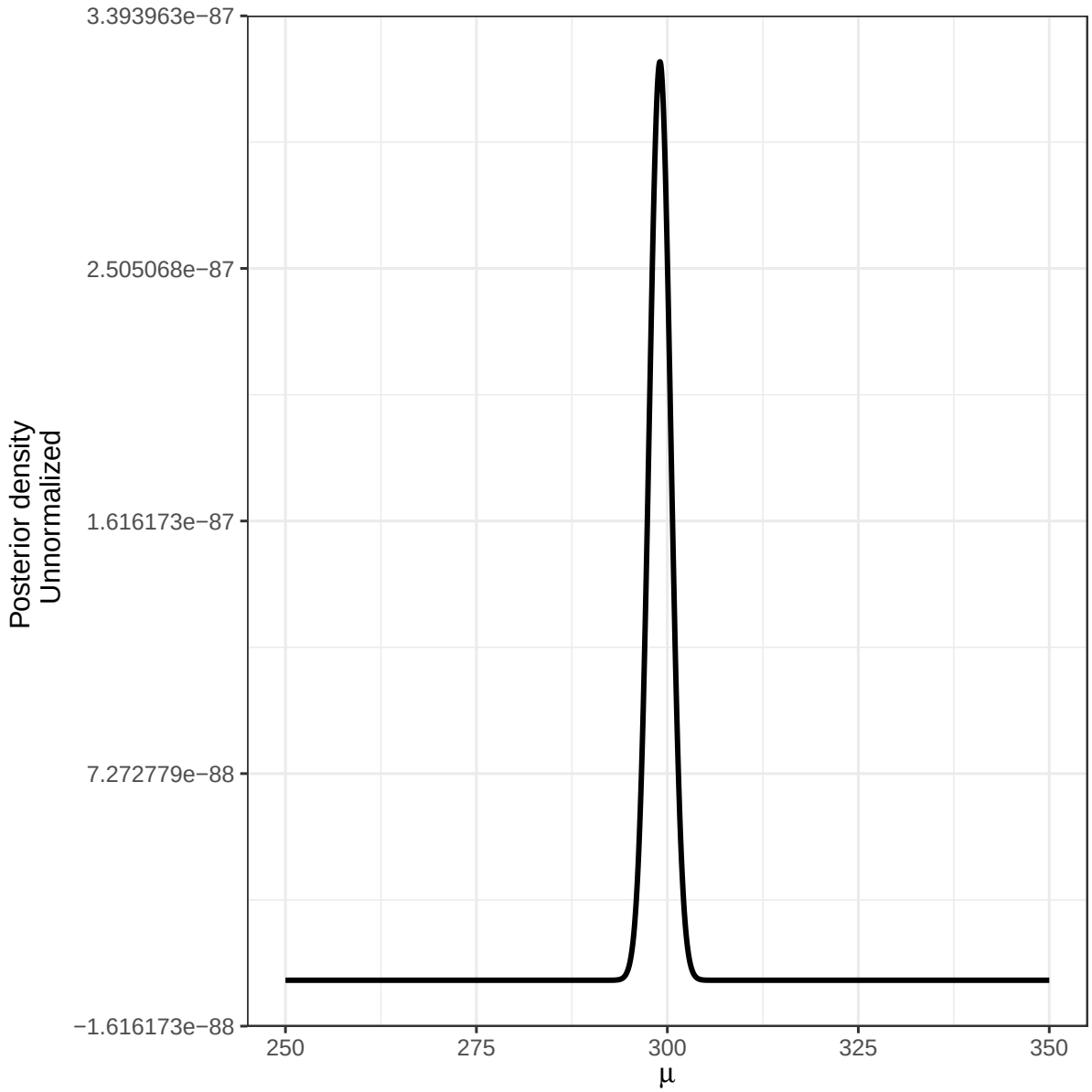
Given the likelihood and the prior density functions, we should be able to estimate the unnormalized posterior distribution using Bayes' rule:

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$$

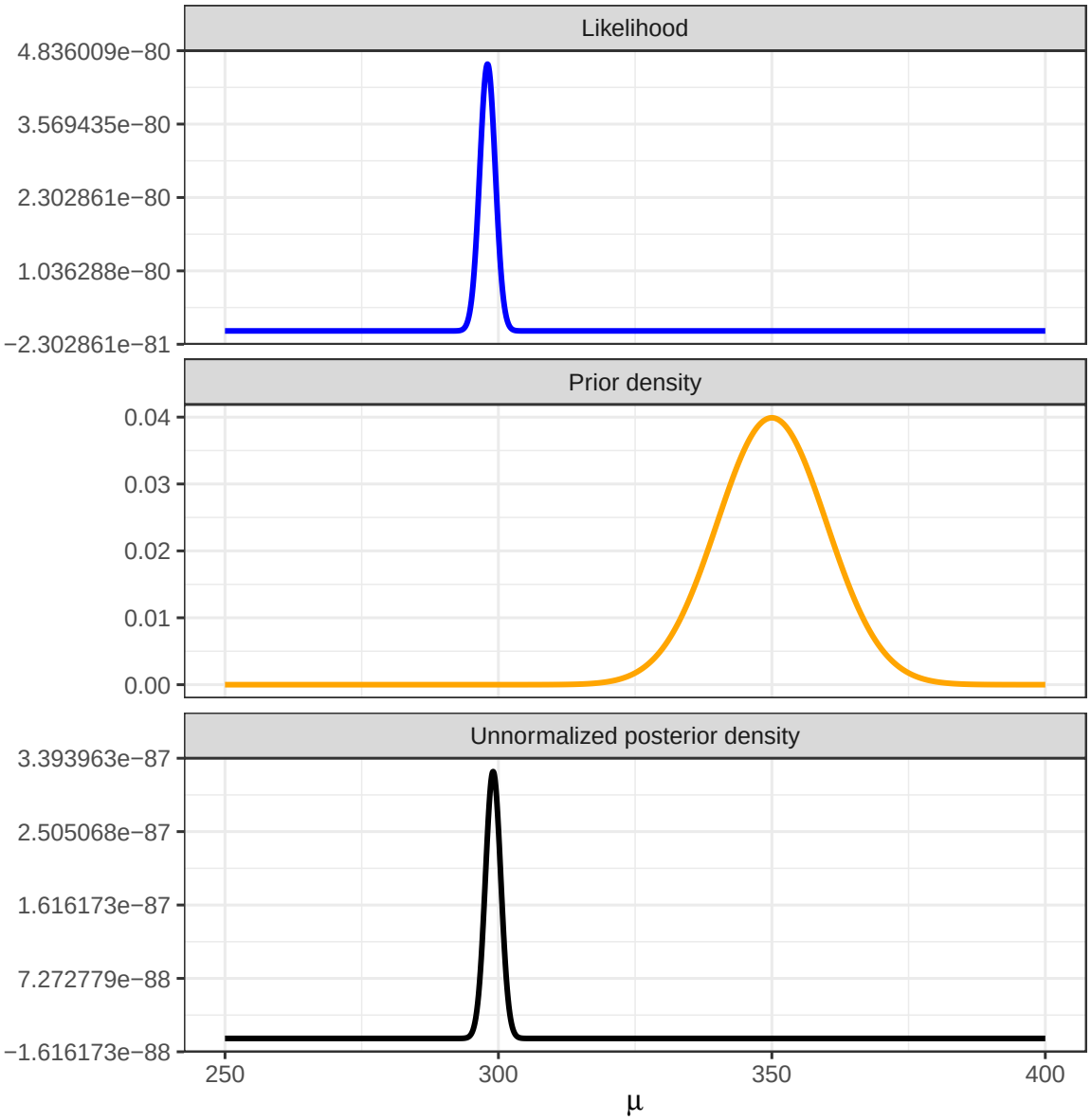
Unnormalized posterior distribution of μ:

$$p'(\mu|y) = \mathcal{L}(\mu|y)p(\mu)$$

```
likelihoods$posterior_unnorm <- likelihoods$lkl*likelihoods$prior_density
ggplot(likelihoods,aes(x=mu,y=posterior_unnorm))+geom_line(size=1,color="black")+
  theme_bw()+xlab(expression(mu))+ylab("Posterior density \n Unnormalized")+
  scale_x_continuous(limits = c(250,350))
```




```
df.lkl_prior <- melt(likelihoods,id=c("mu"))
df.lkl_prior$variable <-
  ifelse(df.lkl_prior$variable=="lkl","Likelihood",
         ifelse(df.lkl_prior$variable=="prior_density","Prior density","Unnormalized posterior density"))

ggplot(df.lkl_prior,aes(x=mu,y=value,color=variable))+geom_line(size=1)+
  theme_bw()+xlab(expression(mu))+ylab("")+
  scale_x_continuous(limits = c(250,400))+
  facet_wrap(~variable,scales = "free_y",ncol = 1)+
  scale_color_manual(values = c("blue","orange","black"))+
  theme(legend.position = "none")
```



Given the above estimates of the unnormalized posterior density, can you draw samples from the posterior distribution of μ?

We will need these samples for further modeling: (i) for estimating the 95% credible interval, (ii) for generating posterior predictions, and (ii) for model evaluation.

There are two ways in which you can draw samples from the posterior distribution of a parameter:

1. Analytically derive the actual, normalized posterior density function $p(\mu|y)$; once you have this function, you can directly sample from this density.

   $\mu* \sim p(\mu|y)$

   (Although, in practice, you can sample from $p(\mu|y)$ only if it is a well-known standard probability distribution, e.g., a normal distribution.)

2. Evaluate a lot of samples (values) of μ. Accept the samples which have high (unnormalized) posterior density; reject the samples which have low (unnormalized) posterior density. The histogram of the accepted samples should look like a posterior distribution of μ.

   (a) Write your own algorithms to draw samples from the posterior.

   (b) Use a pre-defined package (e.g., **brms/stan**) to draw samples from the posterior.

   These are called **parameter estimation methods**. We will learn about them in the next module.

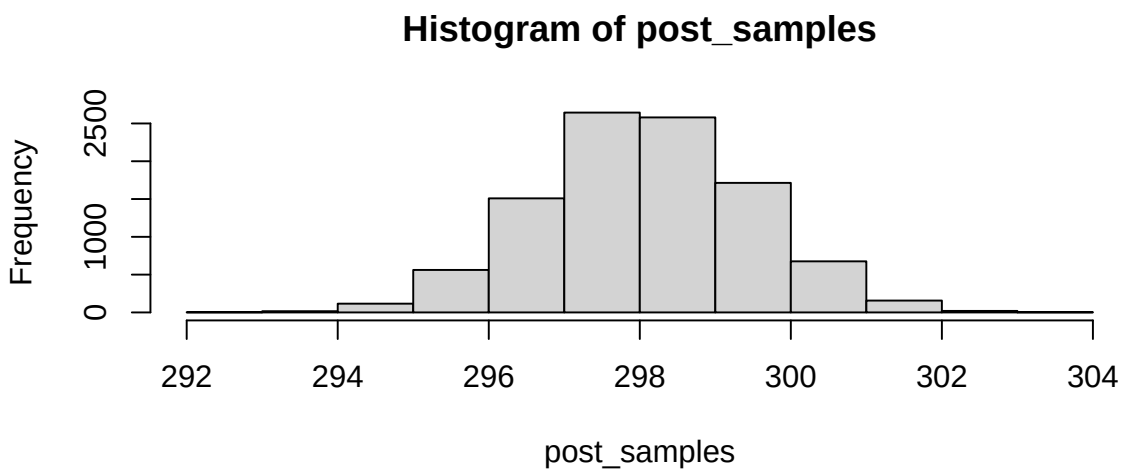
7.2 *Analytically-derived posterior distribution*

When
$y_i \sim Normal(\mu, \sigma)$
and,
$\mu \sim Normal(\mu_0, \sigma_0)$
and,
$\sigma = \sigma$
i.e., when the likelihood is normal and the prior on the mean is normal and the variance is known, the analytically-derived posterior distribution will be a normal distribution $Normal(\mu', \sigma')$ such that,

$$\mu|y \sim Normal(\mu', \sigma')$$

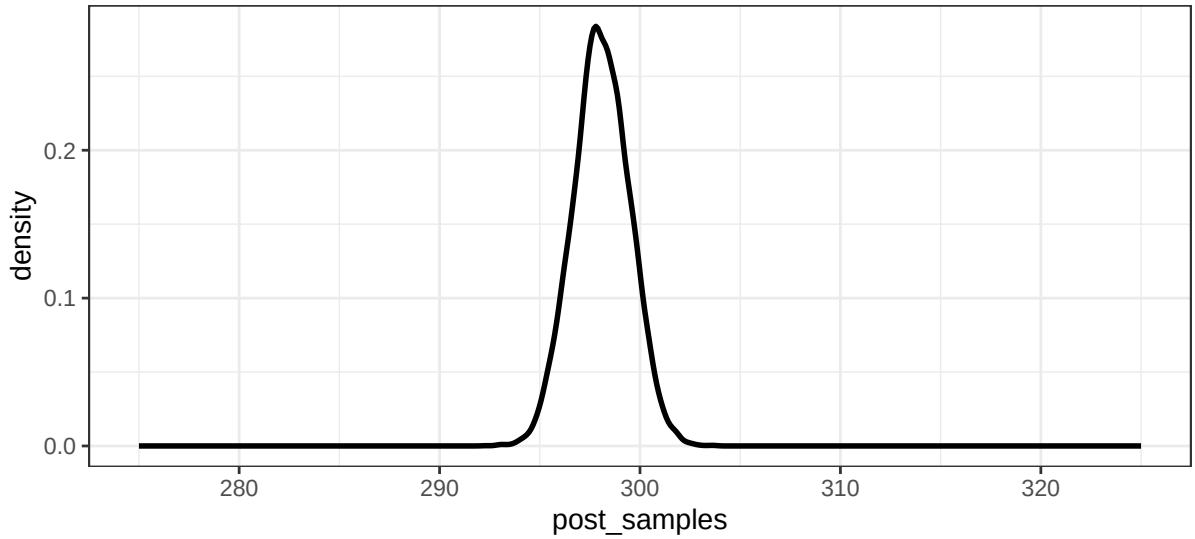
where $\mu|y$ represents a sample from the posterior distribution,
$\mu' = \frac{1}{\left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)} \left(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^{n} y_i}{\sigma^2}\right)$
and,
$\sigma' = \frac{1}{\sqrt{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}}}$

```
mu_0 <- 350
sigma_0 <- 50
sigma <- 10
n <- length(y)

# Parameters of the posterior distribution
sigma_post <- 1/sqrt((1/sigma_0^2)+(n/sigma^2))
mu_post <- (sigma_post^2)*((mu_0/sigma_0^2)+(sum(y)/sigma^2))

post_samples <- rnorm(10000,mu_post,sigma_post)
hist(post_samples)
```

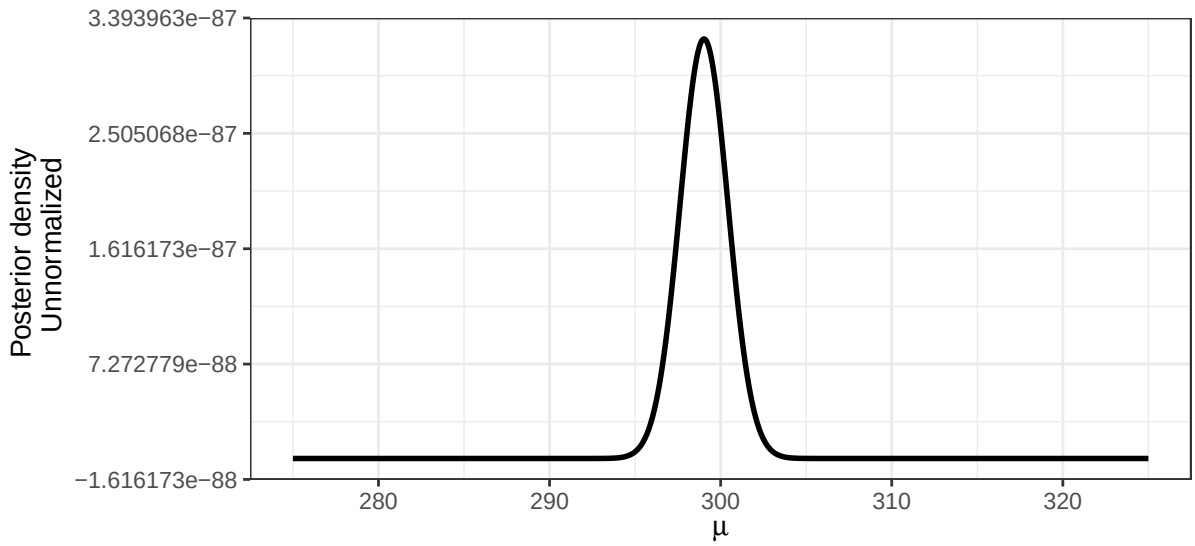

Histogram of post_samples
Frequency
0
1000
2500
292
294
296
298
300
302
304
post_samples

Does this look similar to the unnormalized posterior density graph?

```
df.post_samples <- data.frame(post_samples)
ggplot(df.post_samples,aes(x=post_samples))+
  geom_density(size=1)+theme_bw()+
  scale_x_continuous(limits = c(275,325))
```



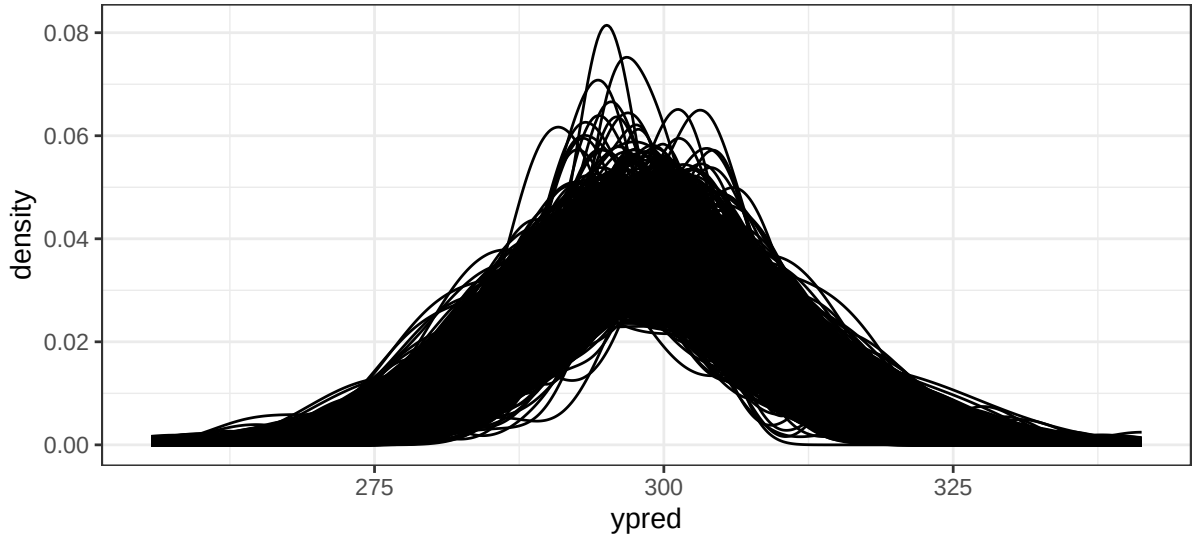

```
ggplot(likelihoods,aes(x=mu,y=posterior_unnorm))+geom_line(size=1,color="black")+
  theme_bw()+xlab(expression(mu))+ylab("Posterior density \n Unnormalized")+
  scale_x_continuous(limits = c(275,325))
```



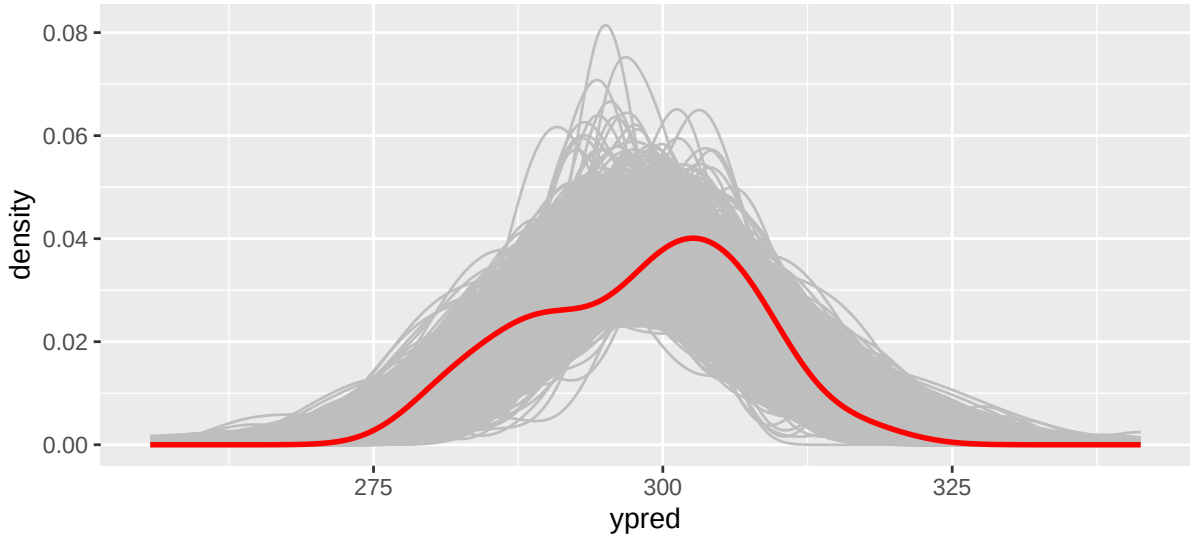
8 *Posterior predictions of the model*

1. Draw a lot of samples from the posterior distribution.
2. Simulate data (50 observations) from the model for each sample.

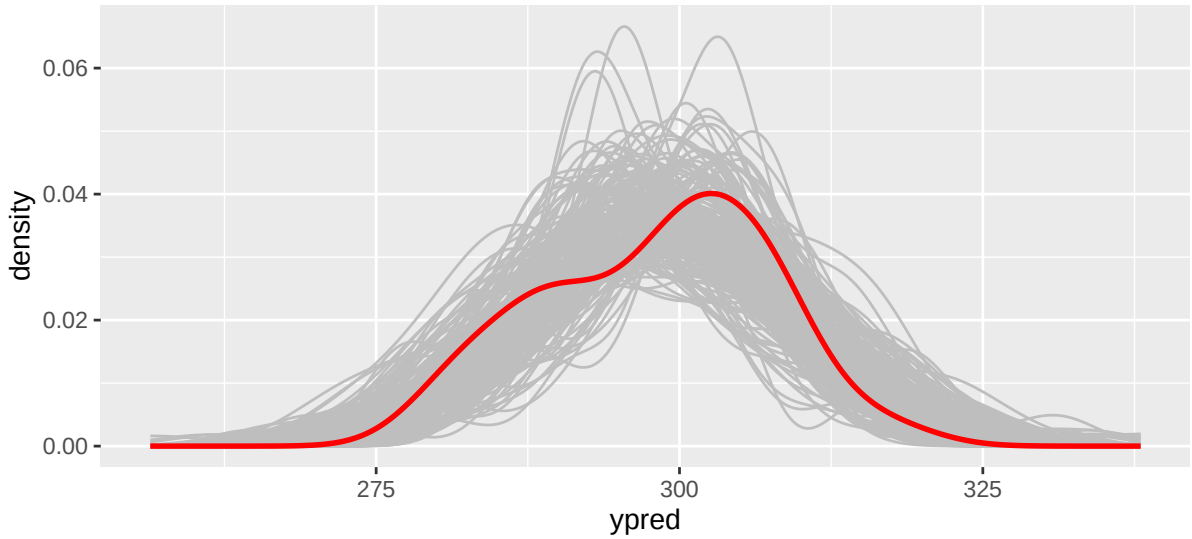
```
mu_samples <- rnorm(2000,mu_post,sigma_post)
sigma <- rep(10,2000)

N <- 50 # I will keep it same as the number of observations in our data y

df.pred <- data.frame(sample = rep(1:2000,each=N),
                      mu=rep(mu_samples,each=N),sigma=rep(sigma,each=N),
                      observation = rep(1:N,2000))

df.pred$ypred <- NA
for(i in 1:length(mu_samples)){
  df.pred[df.pred$sample==i,]$ypred <- rnorm(N,mean=mu_samples[i],sd=sigma[i])
}

ggplot(df.pred,aes(x=ypred,group=sample))+
  geom_density(alpha=0.0001)+theme_bw()
```




```
obs <- subset(df.pred,sample==1)
obs$ypred <- y

ggplot(df.pred,aes(x=ypred,group=sample))+
  geom_density(alpha=0.0001,color="gray")+
  geom_density(data=obs,aes(x=ypred),color="red",size=1)
```




```
ggplot(subset(df.pred,sample<200),aes(x=ypred,group=sample))+
  geom_density(alpha=0.0001,color="gray")+
  geom_density(data=obs,aes(x=ypred),color="red",size=1)
```



9 *Bayesian workflow using brms*

9.1 Model

Likelihood:

$$y_i \sim Normal(\mu, \sigma)$$

Priors:

$$\mu \sim Normal(350, 50)$$

$$\sigma = 10$$

9.2 *Check prior predictions*

9.3 *Prepare data*

```
dat <- data.frame(trial=1:length(y),y=y)
head(dat)
```

```
##   trial        y
## 1     1 279.4704
## 2     2 304.4840
## 3     3 314.6151
## 4     4 318.1779
## 5     5 289.0857
## 6     6 301.1756
```

9.4 *Specify the model in the 'brm' function*

```
m1 <- brm(y ~ 1, data = dat,
          family = gaussian(),
          prior = c(prior(normal(350,50),class=Intercept),
                    prior(constant(10),class=sigma))
          )
```

```
## Compiling Stan program...
```

```
## Start sampling
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 6.6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.66 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.02 seconds (Warm-up)
## Chain 1:                0.018 seconds (Sampling)
## Chain 1:                0.038 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 7e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.02 seconds (Warm-up)
## Chain 2:                0.019 seconds (Sampling)
## Chain 2:                0.039 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 8e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
```

```
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.02 seconds (Warm-up)
## Chain 3:                0.019 seconds (Sampling)
## Chain 3:                0.039 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 7e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.021 seconds (Warm-up)
## Chain 4:                0.021 seconds (Sampling)
## Chain 4:                0.042 seconds (Total)
## Chain 4:
```

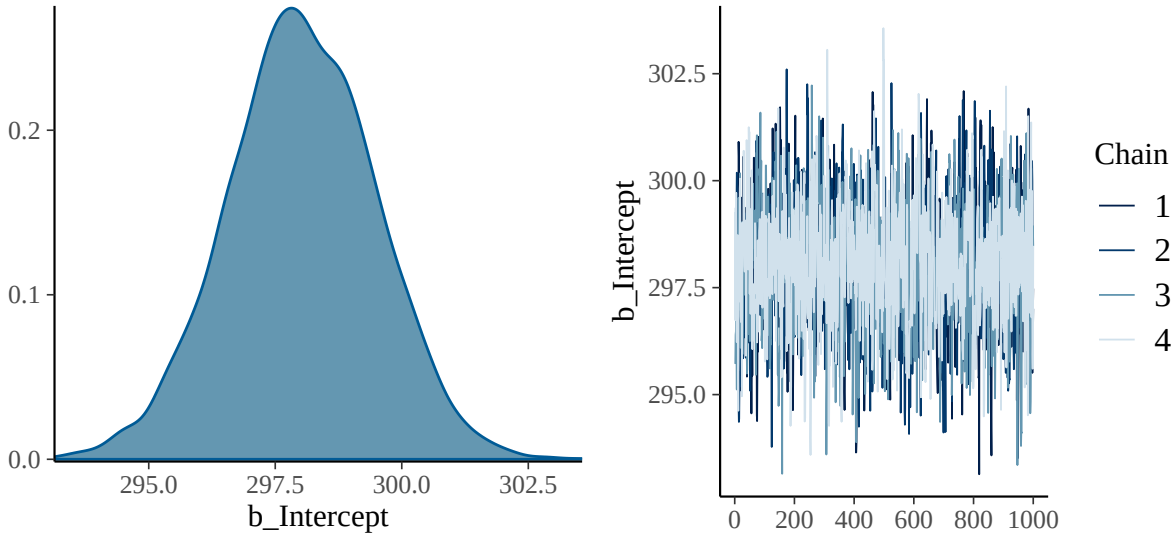
```
summary(m1)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: y ~ 1
##    Data: dat (Number of observations: 50)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept   298.02      1.46   295.15   300.81 1.00     1395     1717
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    10.00      0.00    10.00    10.00   NA       NA       NA
##
## 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).
```

9.5 Visualize the posterior samples

```
plot(m1,pars = c("b_Intercept"))
```

```
## Warning: Argument 'pars' is deprecated. Please use 'variable' instead.
```



9.6 Extract the posterior samples

```
post.samples <- posterior_samples(m1)
```

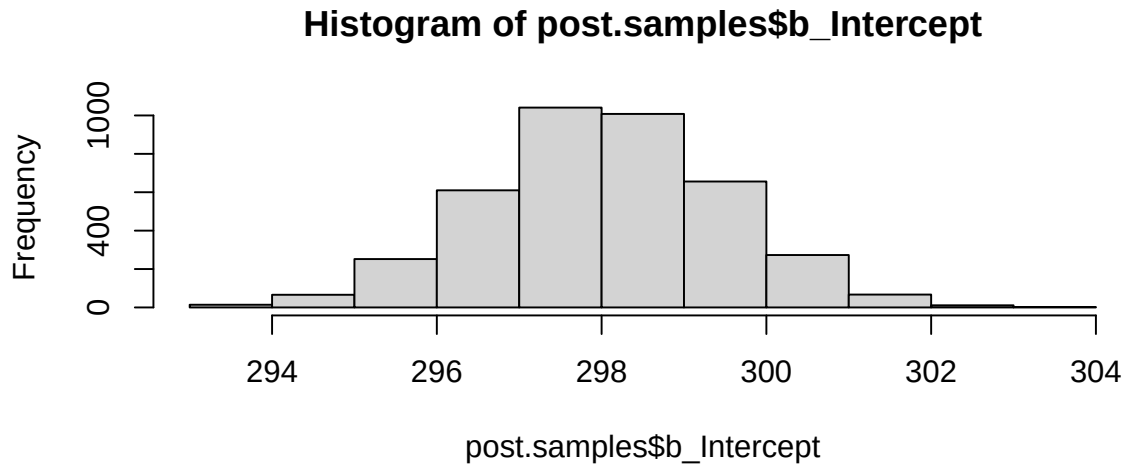
```
## Warning: Method 'posterior_samples' is deprecated. Please see ?as_draws for
```

```
## recommended alternatives.
```

```
head(post.samples)
```

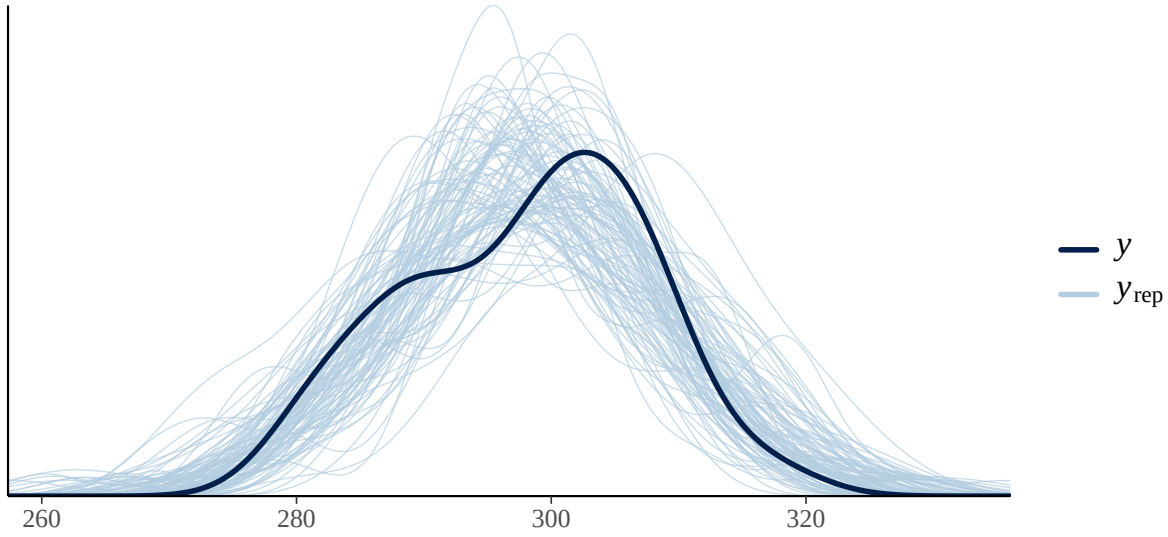
```
##   b_Intercept sigma    lprior      lp__
## 1    298.7763    10 -5.355734 -188.1826
## 2    297.9842    10 -5.372090 -188.0517
## 3    299.0723    10 -5.349688 -188.3121
## 4    296.7979    10 -5.397054 -188.4429
## 5    296.8983    10 -5.394920 -188.3826
## 6    296.8983    10 -5.394920 -188.3826
```

```
hist(post.samples$b_Intercept)
```



9.7 Check posterior predictions

```
pp_check(m1,ndraws = 100)
```


260
280
300
320
y
y rep