CGS698C, Lectures 11-14: Bayesian regression modeling

Himanshu Yadav

2024-03-07

Contents

See chapters 3 and 4 of the book "An Introduction to Bayesian Data Analysis for Cognitive Science (`https://vasishth.github.io/bayescogsci/book/`)" for reference.

1 *Parameter estimation using brms/stan*

- There are packages in R and Python that can estimate model parameters using one of the posterior simulation algorithms introduced in the previous lectures.
- You only need to define your likelihood and priors in a given syntax; the algorithm in the background will start drawing samples from the posterior.
- A popular one is Rstan/pystan/brms package, which uses a Hamiltonian Monte Carlo algorithm for sampling.

1.1 *Implementation*

Example. A normal model with unknown mean and unknown variance

Suppose you are given 100 independent and identically distributed data points that are assumed to come from a Normal distribution with mean μ and standard deviation σ. Let y_i be the i^{th} data point,

Likelihood:

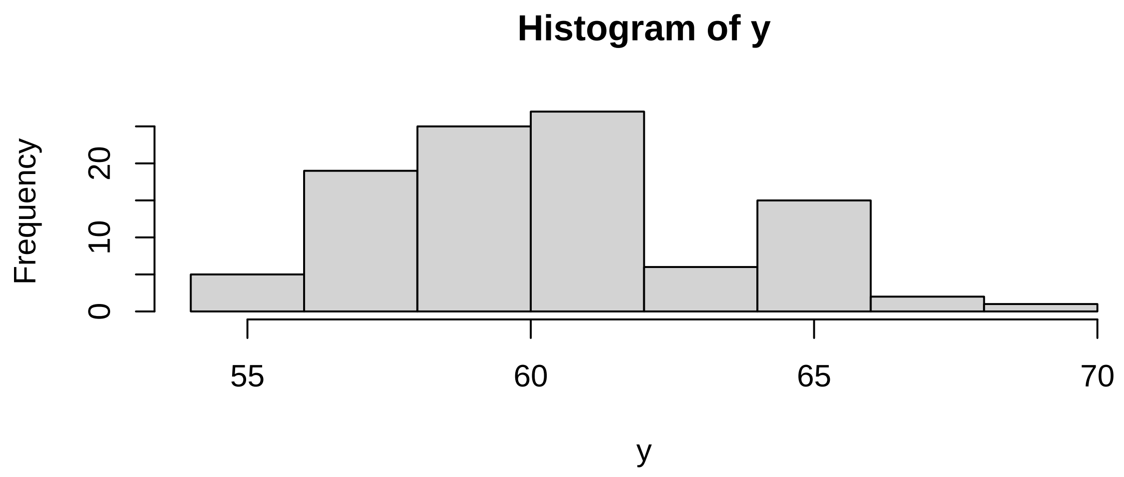
$y_i \sim Normal(\mu, \sigma)$

Priors:

$\mu \sim Normal(50, 10)$

$\sigma \sim Normal_{+}(0, 5)$

```r
# Assuming, true parameter values, mu=60, sigma=3
# Observed (fake) data
y <- rnorm(100,60,3)
hist(y)
```



```r
# Estimate the parameters mu and sigma

# Prepare the data for brms; it needs a dataframe
```

```
dat <- data.frame(y=y)
head(dat)
```

```
##          y
## 1 59.62189
## 2 57.69213
## 3 56.86780
## 4 64.54228
## 5 58.42823
## 6 60.25551
```

```
# Define priors
priors <- c(prior(normal(50, 10), class = Intercept),
            prior(normal(0, 5), class = sigma))

# Fit the model (estimate parameters)
mfit <-
  brm(formula = y ~ 1,
      data=dat,
      family = gaussian(),
      prior = priors,
      chains = 4,cores = 4,
      iter = 2000,warmup = 1000)

save(mfit,file="FittedModels/Simple-gaussian-model.Rda")

# You can use pickle in python for saving large model fits
```

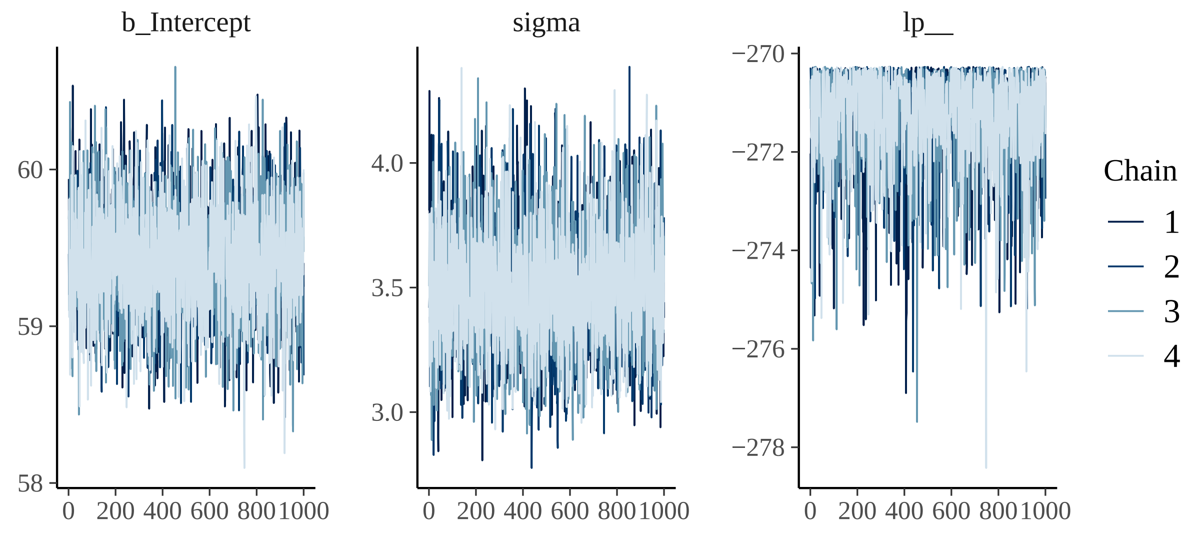
```
summary(mfit)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: y ~ 1
##    Data: dat (Number of observations: 100)
##   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    59.43      0.35    58.75    60.12 1.00     3490     2713
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     3.49      0.25     3.05     4.03 1.00     3281     2790
##
## 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).
```

A summary statistic of parameter estimates:

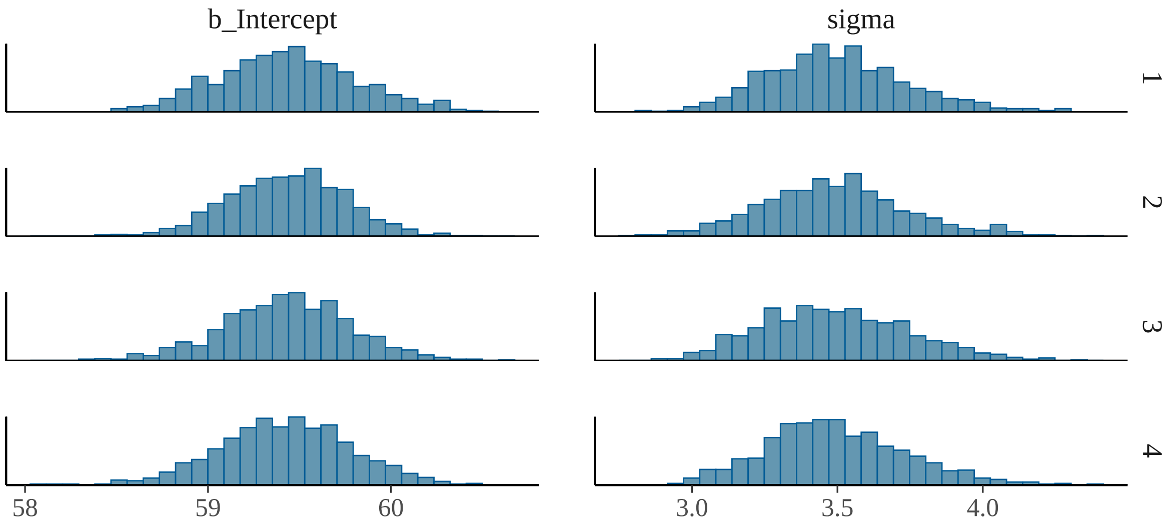
1. There is 95% certainty (0.95 probability) that the true value of parameter mu lies between 58.75 and 60.12
2. The true value of sigma lies between 3.05 and 4.03 with 95% certainty

```
mcmc_trace(mfit)
```



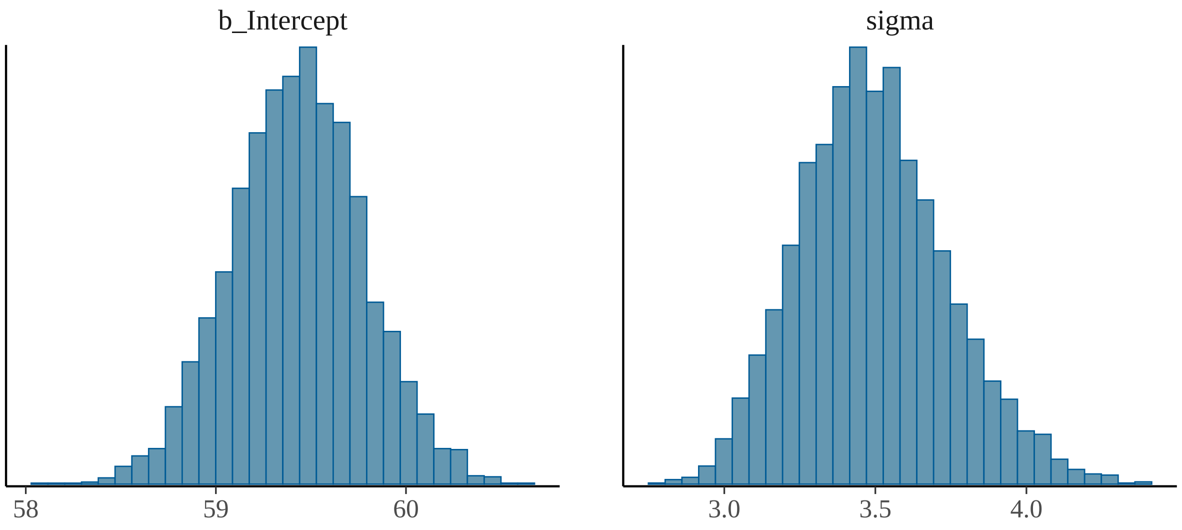
```
mcmc_hist_by_chain(mfit,pars = c("b_Intercept","sigma"))
```

```
## Warning: The ‘facets‘ argument of ‘facet_grid()‘ is deprecated as of ggplot2 2.2.0.
## i Please use the ‘rows‘ argument instead.
## i The deprecated feature was likely used in the bayesplot package.
##   Please report the issue at <https://github.com/stan-dev/bayesplot/issues/>.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.

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



```
mcmc_hist(mfit,pars = c("b_Intercept","sigma"))
```

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



1.2 *Generating posterior predictions*

```
# You can directly use the function from bayesplot package
pp_check(mfit,ndraws = 100, type = "dens_overlay")
```

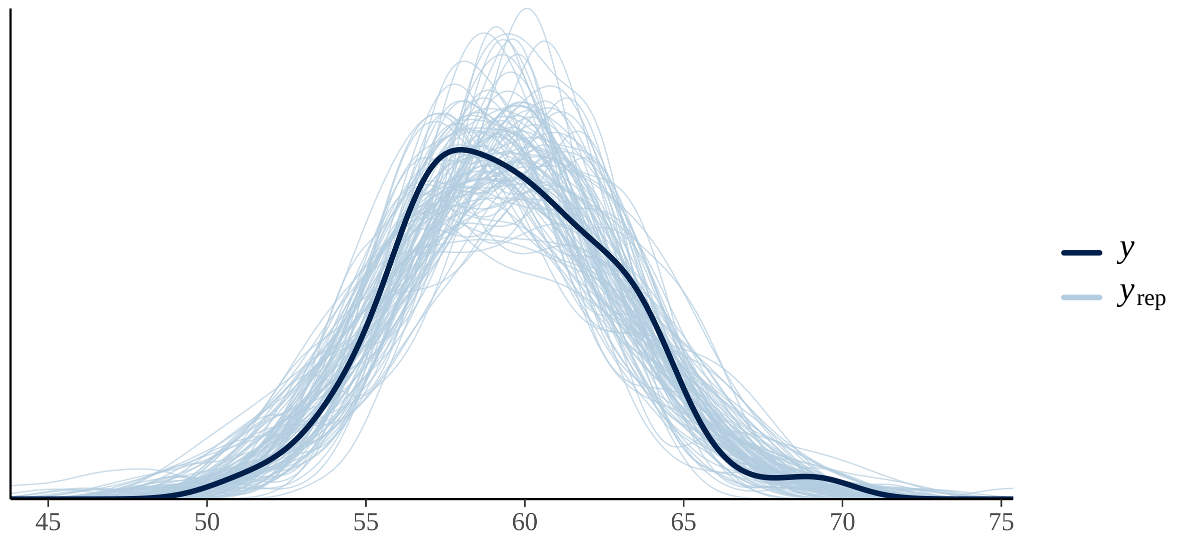
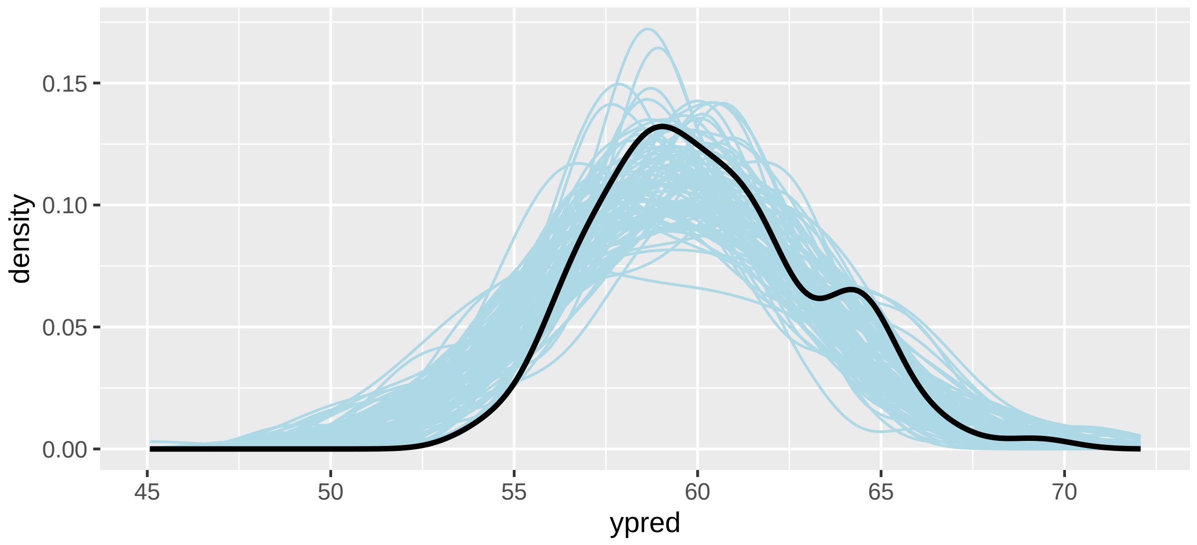


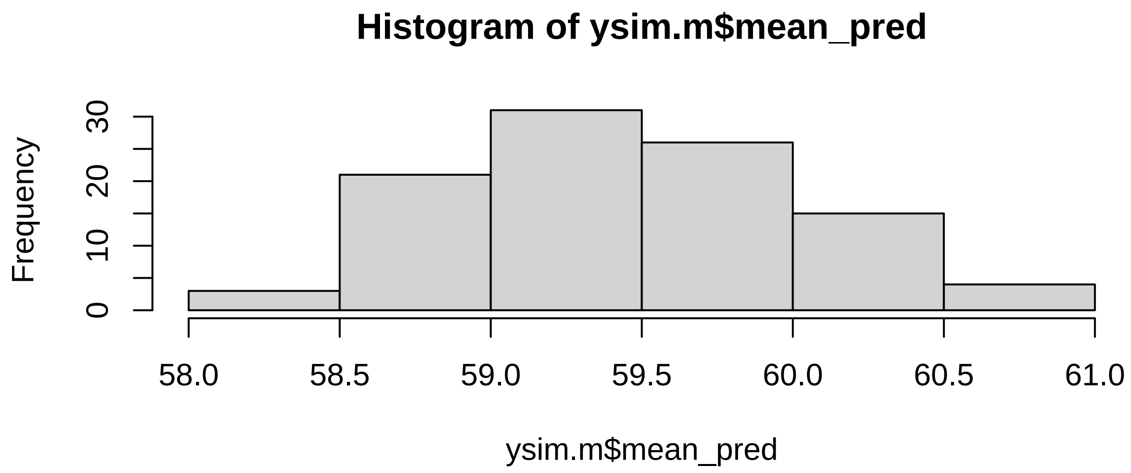
Figure description: Here, each density curve represents the distribution of data: the "dark blue" curve represents the observed data used to fit the model, and the "light blue" curves represent the posterior predictive data, which is predicted by the model after the model was fitted to observed data.

```
# You can generate posterior predictions manually

# First, extract posterior samples from the model fit
post_samples <- posterior_samples(mfit)

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

# Second, create a dataframe containing posterior samples
ysim <- data.frame(matrix(nrow=4000*length(y),ncol=4))
colnames(ysim) <- c("sample_id","mu","sigma","ypred")
ysim$sample_id <- rep(1:4000,each=length(y))
ysim$mu <- rep(post_samples$b_Intercept,each=length(y))
ysim$sigma <- rep(post_samples$sigma,each=length(y))

# Third, generate data from the model
# conditional on each set of parameter values
for(k in 1:100){
  iter_range <- (length(y)*(k-1) + 1):(length(y)*k)
  ysim$ypred[iter_range] <-
    rnorm(length(y),post_samples$b_Intercept[k],post_samples$sigma[k])
}

dat$sample_id <- 0
ggplot(subset(ysim,sample_id<100),aes(x=ypred,group=sample_id))+
  geom_density(alpha=0.05,color="lightblue")+
  geom_density(data=dat,aes(x=y),color="black",size=1)
```

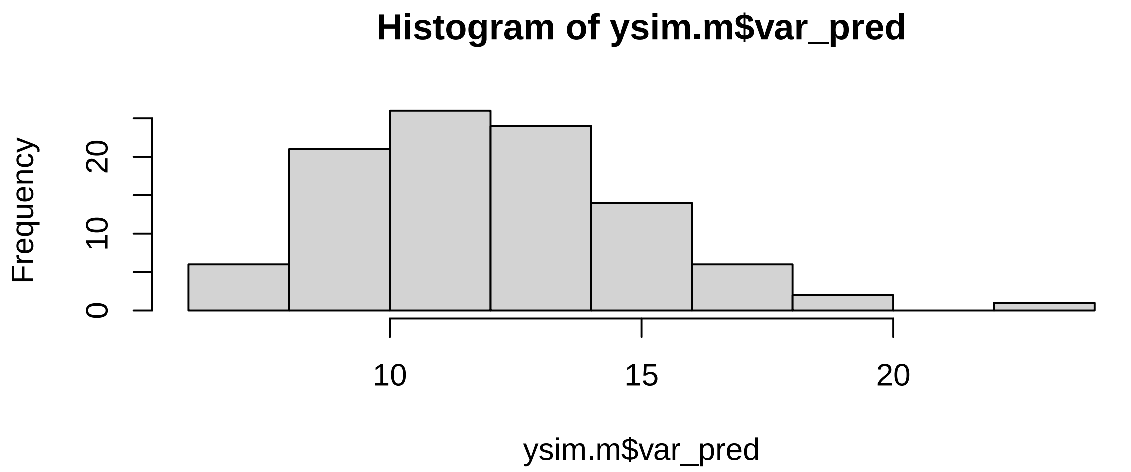
```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



```
# Summary statistics of your posterior predictions
ysim.m <- ysim %>% group_by(sample_id) %>%
  summarise(mean_pred=mean(ypred),
            var_pred=var(ypred))

# Distribution of predicted means
hist(ysim.m$mean_pred)
```



```r
# Distribution of predicted variance
hist(ysim.m$var_pred)
```



2 *Bayesian regression models*

Suppose, in an experiment, you are asked to identify a triangle among many other shapes shown on a screen. Your response time is being recorded.

The experimenter hypothesizes that the background color of the screen ("black" or "blue") affects your response time.

We do not have any further information about how exactly the background color affects the response time.

We can assume a linear relationship between the background color and the response time. Such that, **the mean response time changes as a linear function of the background color**

$$\mu_{rt} = \alpha + \beta X \tag{1}$$

where mu_{rt} is the mean response time, X is the background color and can take values 0 (for "black") or 1 (for "blue"); $alpha$ is the intercept of the straight line, and β is the slope of the straight line.

The slope β represents the extent to which the background color affects the mean response time, and the intercept α represents the mean response time when $X = 0$ i.e., when the background is black.

Now, suppose you collect n repeated observations in your experiment such that around half of them have black background and other half have blue background. The mean response time in a trial i would depend on the background color of the stimuli.

$$\mu_i = \alpha + \beta X_i \tag{2}$$

If you assume that the response times are normally distributed, you can write

$$rt_i \sim Normal(mu_i, \sigma) \tag{3}$$

In other words,

$$rt_i \sim Normal(\alpha + \beta X_i, \sigma) \tag{4}$$

The above equation represents a linear regression model, where rt_i is the dependent variable, X_i is an independent (or predictor) variable, and α, β, σ are the parameters of the model.

You can set priors on α, β, and σ.

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

$$\beta \sim Normal(0, 20)$$

$$\sigma \sim Normal_{+}(0, 10)$$

You can estimate the parameters α, β, and σ using **brms**; we are primarily interested in the estimates of β because we want to test the experimenter's hypothesis that said $\beta \neq 0$.

2.1 *Inferences based on posterior estimates*

```
# Data
alpha <- 250
beta <- 20
sigma <- 10
X <- rep(0:1,50)
rt <- rep(NA,100)
dat <- data.frame(X=X,rt=rt)
for(i in 1:nrow(dat)){
  dat$rt[i] <- rnorm(1,alpha + beta*X[i],sigma)
}
head(dat)
```

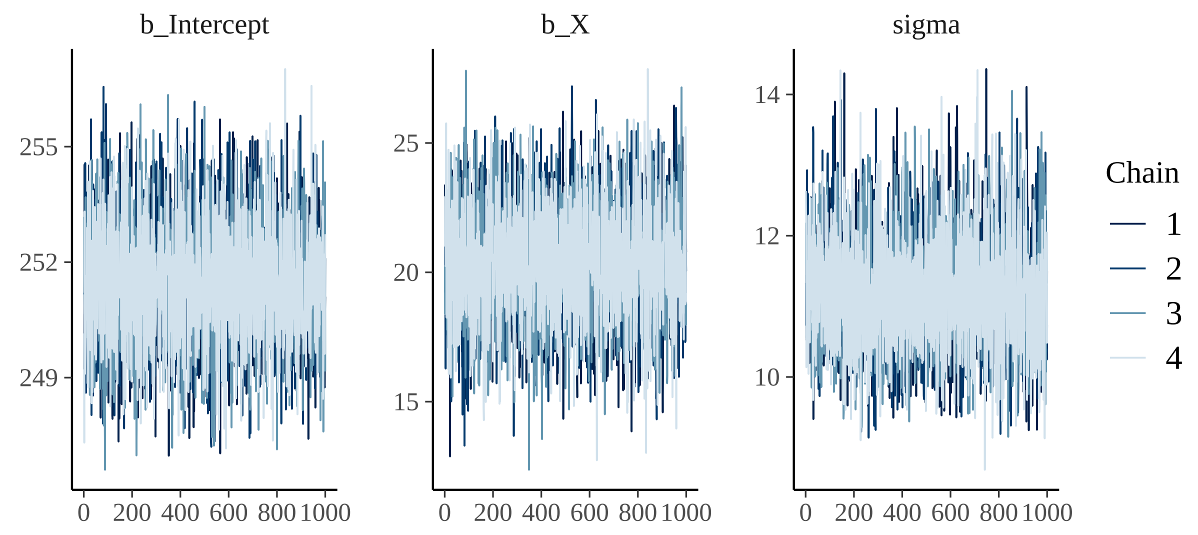
```
##   X       rt
## 1 0 247.5418
## 2 1 264.3281
## 3 0 261.5266
## 4 1 279.1902
## 5 0 271.1362
## 6 1 263.2989
```

```
# Define priors
priors <- c(prior(normal(300, 50), class = Intercept),
            prior(normal(0, 20), class = b, coef=X),
            prior(normal(0, 10), class = sigma))

# Fit the model (estimate parameters)
mfit <-
  brm(formula = rt ~ 1+X,
      data=dat,
      family = gaussian(),
      prior = priors,
      chains = 4,cores = 4,
      iter = 2000,warmup = 1000)

save(mfit,file="FittedModels/Simple-linear-regression.Rda")
```

```
summary(mfit)
```

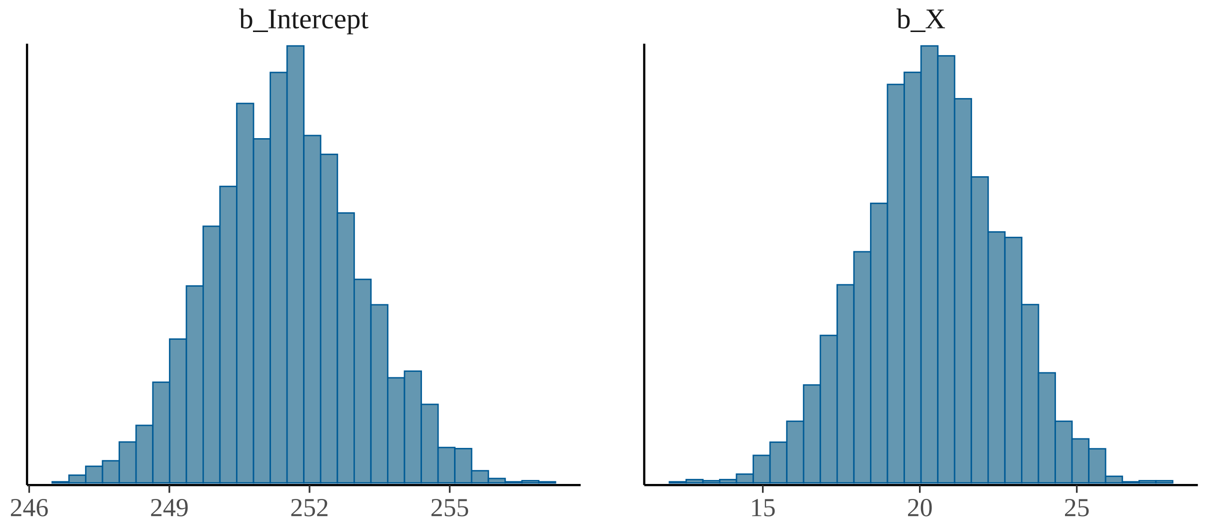
```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: rt ~ 1 + X
##    Data: dat (Number of observations: 100)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept   251.48      1.58   248.46   254.64 1.00     4074     3082
## X            20.39      2.21    16.00    24.64 1.00     3941     3125
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    11.18      0.81     9.74    12.89 1.00     3564     2680
##
## 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).
```

```
mcmc_trace(mfit,pars = c("b_Intercept","b_X","sigma"))
```



```
mcmc_hist(mfit,pars = c("b_Intercept","b_X"))
```

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



Based on the above posterior estimates, you can say the following:

1. The data are consistent with the experimenter's hypothesis that background color affects the response times because the 95% credible interval for *beta* is completely in the positive direction and does not cross zero.

2. The data suggest that $\beta > 0$, i.e., the mean response time is higher when the background color is blue.

However, you cannot say that there is **evidence** for the experimenter's hypothesis. Because the evidence for any model assumption is always computed with respect to a baseline model assumption. No model is absolutely correct; a model can be relatively better than the other.

All you can say given the above results is that the data are consistent with what the experimenter predicted.

2.2 *Prior sensitivity analysis*

What if another experimenter challenges your results saying that your prior assumptions are unreasonable for the β parameter?

You can illustrate how the posterior estimate of β changes under different prior assumptions.

Let us choose 7 different priors on β:

1. $\beta \sim Normal(0, 2)$

2. $\beta \sim Normal(0, 5)$

3. $\beta \sim Normal(0, 10)$

4. $\beta \sim Normal(0, 15)$

5. $\beta \sim Normal(0, 20)$

6. $\beta \sim Normal(0, 25)$

7. $\beta \sim Normal(0, 30)$

```
# Define priors
priors <- c(prior(normal(300, 50), class = Intercept),
            prior(normal(0, 20), class = b, coef=X),
            prior(normal(0, 10), class = sigma))

prior_sd <- c(2,5,10,15,20,25,30)
df.sensitivity <- data.frame(matrix(nrow=0,ncol=4))
colnames(df.sensitivity) <- c("prior","mean.beta","q.lower","q.upper")


# Fit the model (estimate parameters)
for(p in prior_sd){
  priors[2,] <- set_prior(paste("normal(0, ",p,")",sep=""), class = "b", coef="X")
  mfit <-
  brm(formula = rt ~ 1+X,
      data=dat,
      family = gaussian(),
      prior = priors,
      chains = 4,cores = 4,
      iter = 2000,warmup = 1000)

  save(mfit,file=paste("FittedModels/Simple-linear-regression-prior-sd-",as.character(p),".Rda",sep=""))
  post_samples <- posterior_samples(mfit)
  df.sensitivity[nrow(df.sensitivity)+1,] <-
    c(paste("normal(0, ",p,")",sep=""),
      mean(post_samples$b_X),
      unname(quantile(post_samples$b_X,probs=c(.025,.975))))
}

save(df.sensitivity,file="FittedModels/Prior-sensitivity-analysis.Rda")
```

```
load("FittedModels/Prior-sensitivity-analysis.Rda")
df.sensitivity
```

```
##            prior        mean.beta          q.lower          q.upper
## 8   normal(0, 2) 7.80440654632153 4.35792141388259 11.1519352077392
## 9   normal(0, 5) 17.1524443112993 12.9185487822437 21.2308021883041
## 10 normal(0, 10) 19.6630358513428 15.2200820790493 23.9755812727125
## 11 normal(0, 15) 20.2361781846795 15.8609839600651 24.4945166632383
## 12 normal(0, 20) 20.3728699565605 15.7195366832652 24.9504187637311
## 13 normal(0, 25) 20.4577454028757 16.0581165460414 24.8583639200569
## 14 normal(0, 30) 20.5156150048918 16.1419497687793 24.8608227487119
```

The posterior estimates are stable arcoss priors $Normal(0, 10)$, $Normal(0, 15)$, $Normal(0, 20)$,

$Normal(0, 25)$, and $Normal(0, 30)$.

2.3 *Another way to write the regression equation*

The regression model $rt_i \sim Normal(\alpha + \beta X_i, \sigma)$ can be rewritten as

$$rt_i = \alpha + \beta X_i + \epsilon_i \quad \text{where } \epsilon_i \sim Normal(0, \sigma)$$

2.4 *The lognormal likelihood*

We have observed previously that the typical response times are lognormally distributed.

We can revise our regression model as

$$rt_i \sim Lognormal(\alpha + \beta X_i, \sigma)$$

or

$$\log rt_i = \alpha + \beta X_i + \epsilon_i \quad \text{where } \epsilon_i \sim Normal(0, \sigma)$$

An important thing to note is that the parameters in the above model are in the log space. Thus, we need to update our priors accordingly,

$$\alpha \sim Normal(6, 2)$$

$$\beta \sim Normal(0, 3)$$

$$\sigma \sim Normal_+(0, 2)$$

```r
# Define priors
priors <- c(prior(normal(6, 2), class = Intercept),
            prior(normal(0, 3), class = b, coef=X),
            prior(normal(0, 2), class = sigma))

# Fit the model (estimate parameters)
mfit <-
  brm(formula = rt ~ 1+X,
      data=dat,
      family = lognormal(),
      prior = priors,
      chains = 4,cores = 4,
      iter = 2000,warmup = 1000)

save(mfit,file="FittedModels/Lognormal-regression.Rda")

summary(mfit)

##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: rt ~ 1 + X
##    Data: dat (Number of observations: 100)
##   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     5.53      0.01     5.51     5.54 1.00     3600     3006
## X             0.08      0.01     0.06     0.10 1.00     3274     2570
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.04      0.00     0.04     0.05 1.00     2695     2306
##
## 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).
```

What is the effect size?

The above posterior estimates tell you effect sizes on log milliseconds scale, but we typically need them on raw milliseconds scale.

You can backtransform the effect of predictor X on the milliseconds scale.

The effect size on milliseconds scale is basically the difference in predicted response times when $X = 1$ compared to $X = 0$

Effect $= rt_{pred_{X=1}} - rt_{pred_{X=0}}$

We know,

$\log rt_{pred_i} = \alpha + \beta X_i$

when $X_i = 0$
$\log rt_{pred} = \alpha$
hence,
$rt_{pred_{X=0}} = e^{\alpha}$
when $X_i = 1$
$\log rt_{pred} = \alpha + \beta$
$rt_{pred_{X=1}} = e^{\alpha+\beta}$
The effect size on milliseconds scale: $rt_{pred_{X=1}} - rt_{pred_{X=0}} = e^{\alpha+\beta} - e^{\alpha}$.

```
post_samples <- posterior_samples(mfit)
```

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

```
beta_raw <- exp(post_samples$b_Intercept + post_samples$b_X)-exp(post_samples$b_Intercept)

c(mean=mean(beta_raw),quantile(beta_raw,probs = c(.025,.975)))
```

```
##     mean     2.5%    97.5%
## 20.71092 16.39950 25.16856
```

3 *Logistic regression models*

Consider another dependent variable in the same experiment: **the response accuracy**. If the participant was able to correctly identify the target shape, the response is recorded as 1 otherwise 0. The experimenter hypothesizes that the responses are on average more accurate when the background color is black.

Suppose, $correct_i$ is the vector of responses containing 1 for correct responses and 0 for incorrect ones.

The resoonses can be assumed to be generated by a Bernoulli distribution.

$$correct_i \sim Bernoulli(\theta_i)$$

where θ_i represents the average probability of producing a correct response in trial i. The linear regression is defined for the log-odds of parameter θ_i

$$\text{logit } \theta_i = \alpha + \beta X_i$$

or,

$$\log \frac{\theta_i}{1-\theta_i} = \alpha + \beta X_i$$

You can directly write,

$$correct_i \sim Bernoulli(\text{inverse-logit } (\alpha + \beta X_i))$$

The above model is called a logistic regression model.

The parameters α and β are in the log-odds (logit) space, we need to define the priors accordingly

$$\alpha \sim Normal(0, 1.5)$$

$$\beta \sim Normal(0, 0.5)$$

3.1 *Implementation*

```
dat$correct <- NA
for(i in 1:nrow(dat)){
  dat$correct[i] <- rbinom(1,size=1, prob = plogis(1.5 - 0.5 * dat$X[i]))
}
```

```
# Define priors
priors <- c(prior(normal(0, 1.5), class = Intercept),
            prior(normal(0, 0.5), class = b, coef=X))

# Fit the model (estimate parameters)
mfit <-
  brm(formula = correct ~ 1+X,
      data=dat,
      family = bernoulli(link = logit),
      prior = priors,
      chains = 4,cores = 4,
      iter = 2000,warmup = 1000)

save(mfit,file="FittedModels/Logistic-regression.Rda")
```

```
summary(mfit)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: correct ~ 1 + X
##    Data: dat (Number of observations: 100)
##   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     1.68      0.33     1.06     2.37 1.00     2961     2567
## X             0.08      0.36    -0.65     0.79 1.00     3254     2518
##
## 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).
```