Lab 8

PSTAT 115, Summer 2023

September, 2023

Simple linear regression model

The simple linear regression model is:

$$y_i = \beta x_i + \alpha + \epsilon_i \quad i = 1, \ldots, n$$

$\epsilon_i \sim \text{normal}(0, \sigma^2)$ and ϵ_i's are independent.

This model can be written as

$$y_i - \beta x_i - \alpha \sim \text{normal}(0, \sigma^2)$$

which means

$$y_i \sim \text{normal}(\beta x_i + \alpha, \sigma^2)$$

. And we will use this last expression of the model to code simple linear regression with stan.

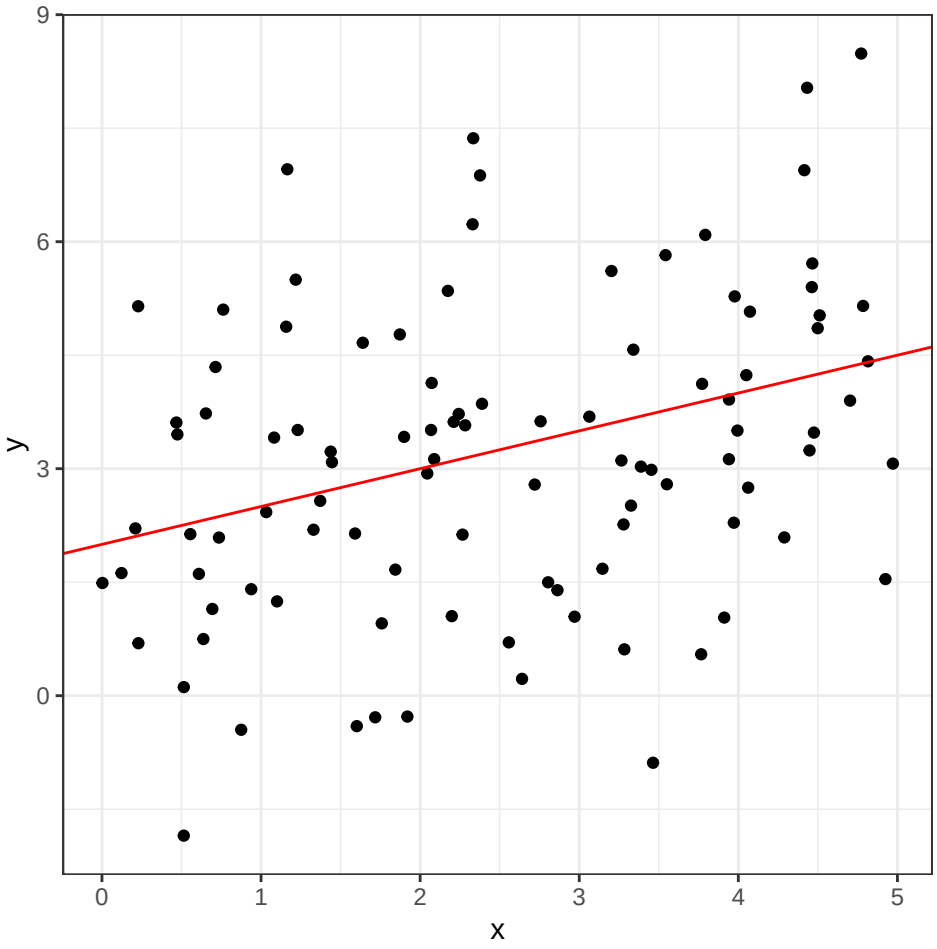
Now the parameters that we have are β, α and σ. In frequentist view, we find the likelihood and compute the MLE then mission's complete. But in Bayesian context, we can specify some priors, find the posterior distribution and estimate those parameters with posterior means/medians/modes etc. Of course, the posterior distribution might be complicated and we need to turn to MCMC for help. That's when we would like to use stan to tackle problems.

We code the model in a stan file as follows. Note: in stan, if you don't specify the prior, it will use the improper priors for the unbounded parameters.

```
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;
}
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma;
}
model {
  y ~ normal(alpha + x*beta, sigma);
}
```

```
set.seed(123)
x <- runif(100, 0,5)
a <- 2
b <- 0.5
```

```r
y <- b*x+a+rnorm(100, mean = 0, sd = 2)
n <- length(y)
ggplot(data = data.frame(x=x,y=y))+geom_point(aes(x=x,y=y))+geom_abline(slope = b, intercept = a, col =
```




```r
stan_model2 = cmdstan_model("simple_linear_regression.stan")
stan_fit2 =   stan_model2$sample(
    data = list(N = n, y = y, x=x),
    refresh = 0, show_messages=FALSE)

samples <- stan_fit2$draws(format = "df")
samples
```

```
## # A draws_df: 1000 iterations, 4 chains, and 4 variables
##    lp__ alpha beta sigma
## 1  -117   2.6 0.24   1.8
## 2  -117   2.6 0.28   1.7
## 3  -120   3.2 0.17   2.3
## 4  -116   2.5 0.24   1.8
## 5  -115   2.2 0.43   1.9
## 6  -115   2.0 0.43   2.0
## 7  -118   2.4 0.38   2.3
```

```
## 8  -122   2.6 0.14   2.5
## 9  -121   2.5 0.18   2.5
## 10 -120   1.5 0.61   1.6
## # ... with 3990 more draws
## # ... hidden reserved variables {'.chain', '.iteration', '.draw'}
```

```
alpha_samples <- samples$alpha
beta_samples <- samples$beta
sigma_samples <- samples$sigma
```

Now let's look at the estimates of posterior means.

```
mean(beta_samples)
```

```
## [1] 0.4663945
```

```
mean(alpha_samples)
```

```
## [1] 1.979645
```

```
mean(sigma_samples)
```

```
## [1] 1.963557
```

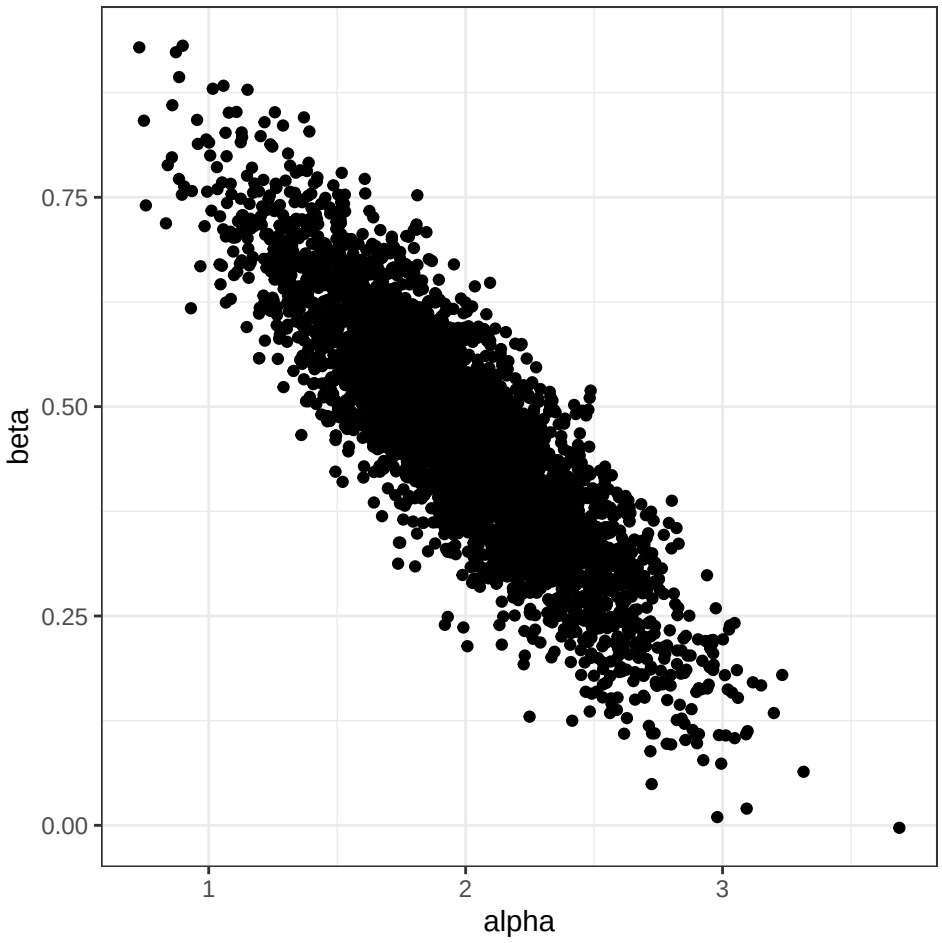
Let's say if we are interested in the predicted value when x=5, that is $\hat{y} = \alpha + 5\beta$. Then we can get samples of this predict value. The sample mean will give us a good estimate of the posterior mean of this predict value at $x = 5$.

```
y_hat <- alpha_samples + 5*beta_samples
mean(y_hat)
```

```
## [1] 4.311617
```

We can also explore the posterior relationship between α and β.

```
ggplot(tibble(alpha=alpha_samples, beta=beta_samples)) + geom_point(aes(x=alpha, y=beta)) + theme_bw()
```



Making 60% and 90% confidence bands plot for y

Important Code for HW

```
xgrid <- seq(0, 5, by=0.1)

xgrid
```

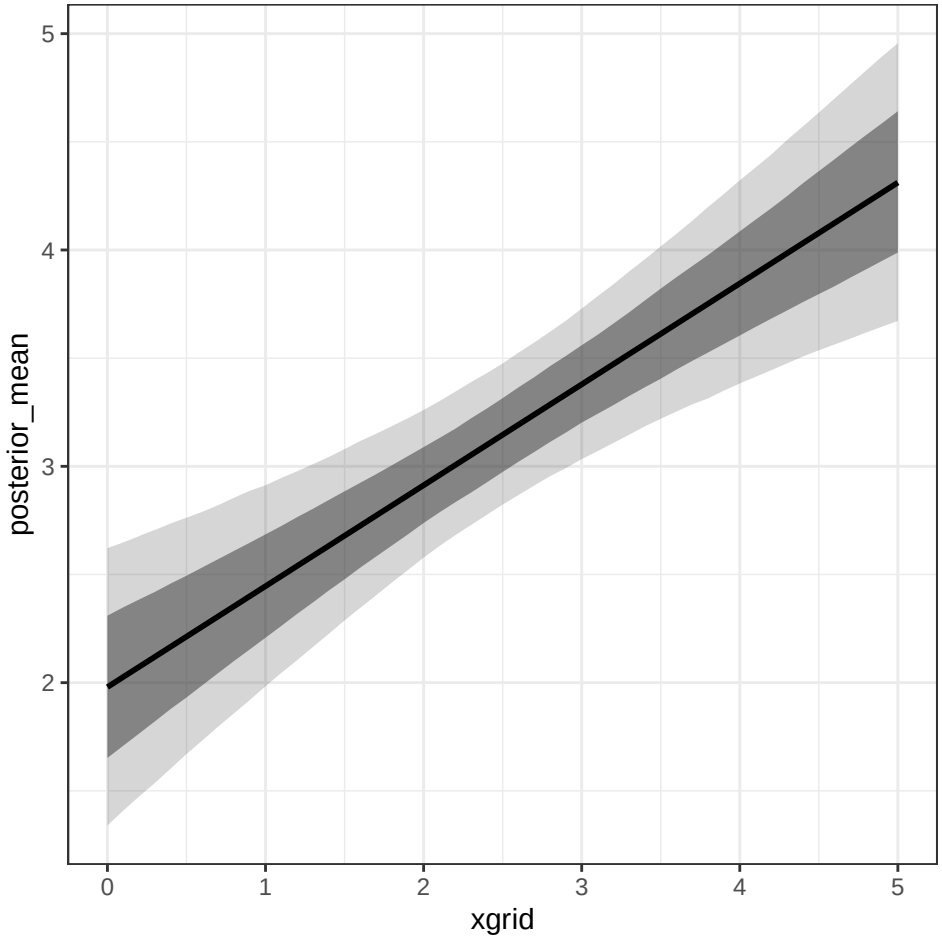
```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8
## [20] 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7
## [39] 3.8 3.9 4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.0
```

```
compute_curve <- function(sample) {
  alpha <- sample[1]
  beta <- sample[2]
  y_values <- alpha + beta*xgrid
}

res <- apply(cbind(alpha_samples, beta_samples), 1, compute_curve)
#each col of res is for a set of alpha,beta values
#each row of res is different y values computed from different sets of alpha beta for a fixed x values.
```

```r
quantiles <- apply(res, 1, function(x) quantile(x, c(0.05, 0.20, 0.80, 0.95)))


posterior_mean <- rowMeans(res)
#posterior_med <- apply(res, 1, median)
tibble(x=xgrid,
q05=quantiles[1, ],
q20=quantiles[2, ],
q80=quantiles[3, ],
q95=quantiles[4, ],
mean=posterior_mean) %>%
ggplot() +
geom_ribbon(aes(x=xgrid, ymin=q05, ymax=q95), alpha=0.2) +
geom_ribbon(aes(x=xgrid, ymin=q20, ymax=q80), alpha=0.5) +
geom_line(aes(x=xgrid, y=posterior_mean), size=1) +
theme_bw()
```



Metropolis Algorithm

To generate sample s + 1 of a Metropolis MCMC sampler given (possibly) unnormalized density $p(\theta)$:

1. Propose a new sample θ_* given the old sample θ_s from a symmetric distribution $J(\theta_*|\theta_s)$
2. If $p(\theta_*) > p(\theta_s)$, then set θ_{s+1} equal to θ_* and go to the next iteration
3. If $p(\theta_*) < p(\theta_s)$, then
    a. Generate a random number r from uniform(0, 1)
    b. If $r < \frac{p(\theta_*)}{p(\theta_s)}$, set θ_{s+1} equal to θ_* and go to the next iteration
    c. Otherwise set θ_{s+1} equal to θ_s

https://chi-feng.github.io/mcmc-demo/app.html