Bayesian Modeling and Inference: An Introduction to STAN for the Social Sciences

Bayesian Linear Regression with Stan

May 23, 2025

Contents

Objective

In this tutorial, we will:

- Implement a simple Bayesian linear regression model in Stan.
- Fit the model to simulated data using R.
- Inspect model convergence and posterior distributions.
- Perform posterior predictive checks.
- Interpret the results.
- Explore how changes in priors, data, and model specification affect the outcomes.

Prerequisites

- Basic understanding of R.
- R and RStudio installed.
- `rstan`, `bayesplot`, `ggplot2`, `dplyr` (or `tidyverse`) R packages installed.
- Conceptual understanding of Bayesian linear regression, priors, likelihood, posterior, and MCMC.

You can install the necessary packages using:

```r
install.packages(c("rstan", "bayesplot", "ggplot2", "dplyr"))
```

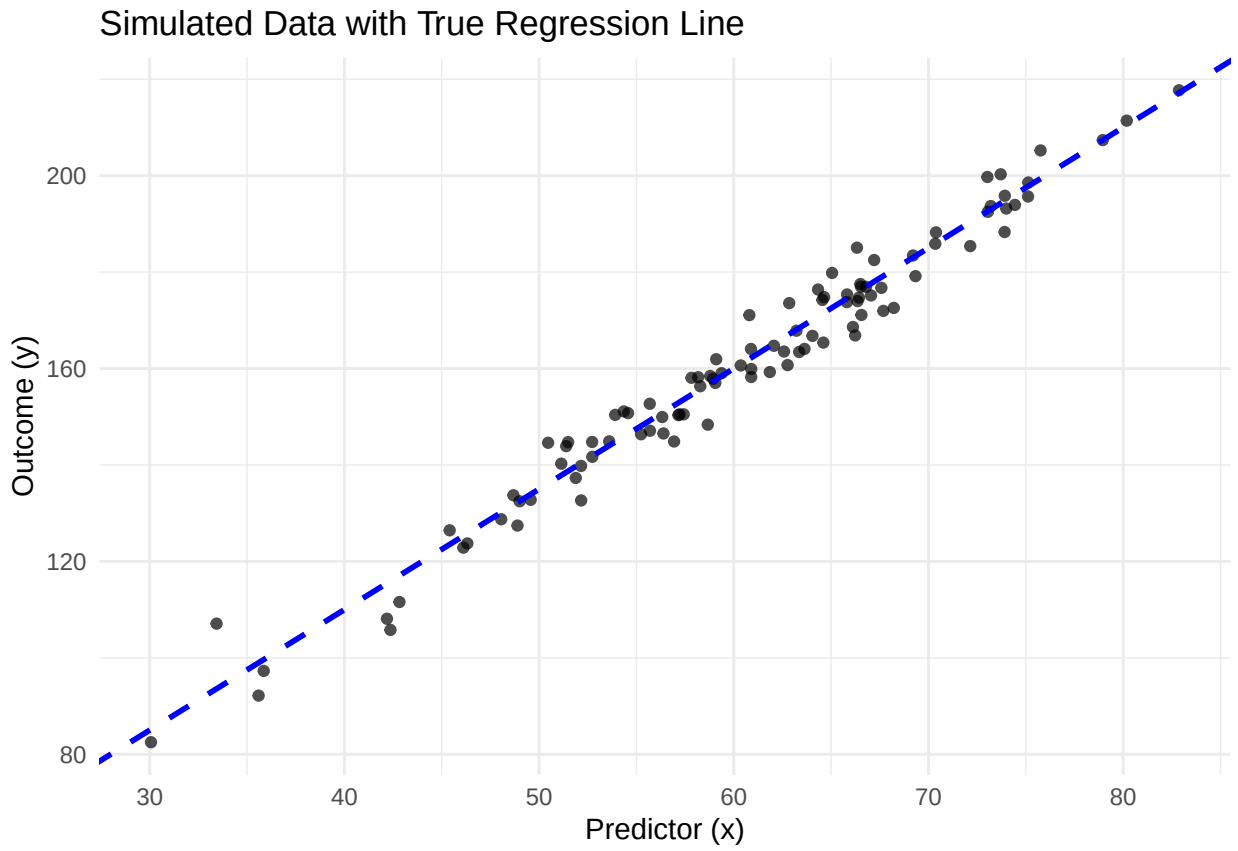
Part 1: Setting Up

```r
library(rstan)
library(bayesplot)
library(ggplot2)
library(dplyr)

rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
```

Part 2: Simulating Data

```r
alpha_true <- 10
beta_true  <- 2.5
sigma_true <- 5
N <- 100

set.seed(42)
x <- rnorm(N, mean = 60, sd = 10)
y_deterministic <- alpha_true + beta_true * x
y <- rnorm(N, mean = y_deterministic, sd = sigma_true)
sim_data <- data.frame(x = x, y = y)

ggplot(sim_data, aes(x = x, y = y)) +
```

```
  geom_point(alpha = 0.7) +
  geom_abline(intercept = alpha_true, slope = beta_true, color = "blue", linetype = "dashed", size=1) +
  labs(title = "Simulated Data with True Regression Line",
       x = "Predictor (x)", y = "Outcome (y)") +
  theme_minimal()
```



Discussion

- `alpha_true`: The true intercept.
- `beta_true`: The true slope.
- `sigma_true`: The standard deviation of errors.

Part 3: Defining the Stan Model

Save the following code in a file named `linear_regression.stan`:

```
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;
}
parameters {
```

```
  real alpha;
  real beta;
  real<lower=0> sigma;
}
model {
  alpha ~ normal(0, 100);
  beta ~ normal(0, 10);
  sigma ~ cauchy(0, 20);

  y ~ normal(alpha + beta * x, sigma);
}
generated quantities {
  vector[N] y_rep;
  vector[N] log_lik;

  for (n in 1:N) {
    y_rep[n] = normal_rng(alpha + beta * x[n], sigma);
    log_lik[n] = normal_lpdf(y[n] | alpha + beta * x[n], sigma);
  }
}
```

Part 4: Fitting the Model in R

```
setwd("/Users/user/Desktop/Lectures 2024/Bayesian Course - UoM/Bayesian Linear Regression")

stan_data <- list(
  N = N,
  x = sim_data$x,
  y = sim_data$y
)

model_compiled <- stan_model(file = "linear_regression.stan")

fit <- sampling(
  object = model_compiled,
  data = stan_data,
  iter = 2000,
  warmup = 1000,
  chains = 4,
  seed = 123,
  refresh = 0
)
```

Part 5: Inspecting Model Fit and Convergence

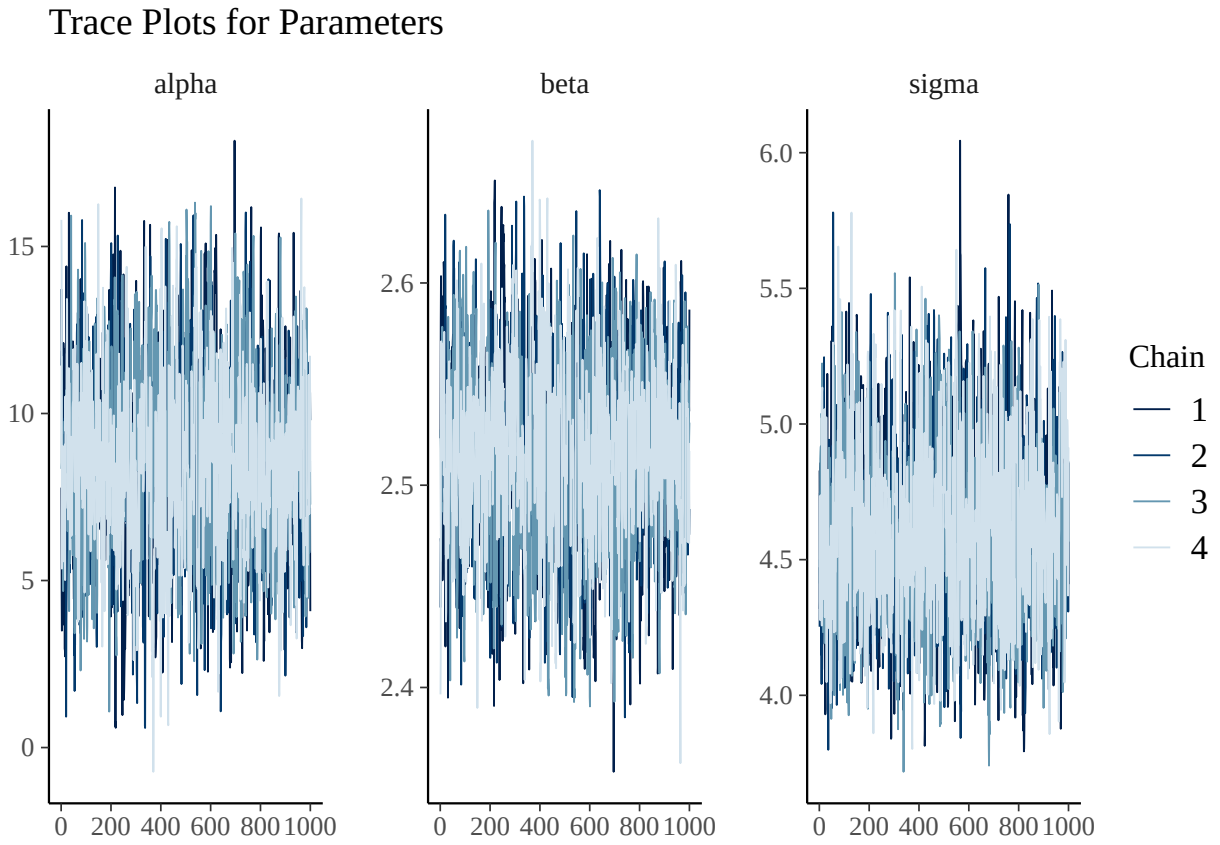
```
print(fit, pars = c("alpha", "beta", "sigma"), probs = c(0.025, 0.5, 0.975))
```

```
## Inference for Stan model: anon_model.
```

```
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##       mean se_mean   sd 2.5%  50% 97.5% n_eff Rhat
## alpha 8.62    0.08 2.77 3.25 8.62 14.16  1247    1
## beta  2.52    0.00 0.05 2.42 2.52  2.60  1240    1
## sigma 4.61    0.01 0.33 4.03 4.59  5.30  1717    1
##
## Samples were drawn using NUTS(diag_e) at Mon May 19 11:26:24 2025.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Trace Plots

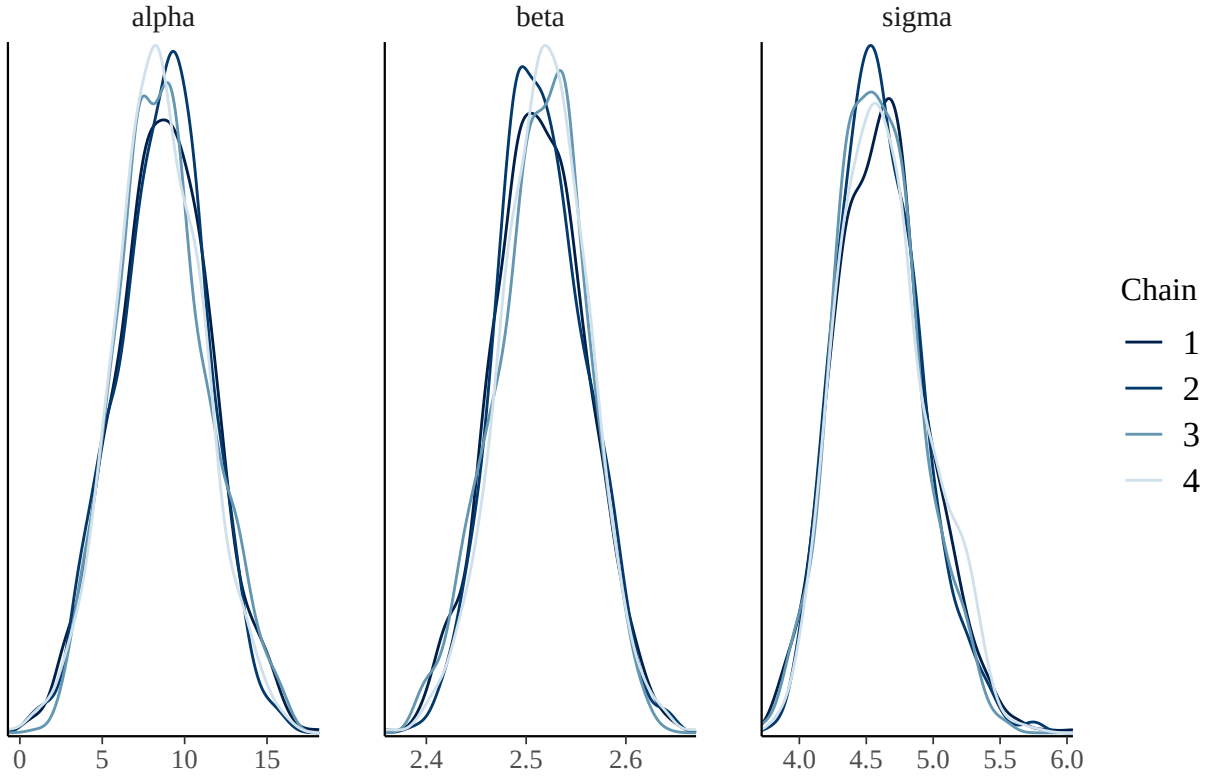
```
mcmc_trace(fit, pars = c("alpha", "beta", "sigma")) +
  ggtitle("Trace Plots for Parameters")
```



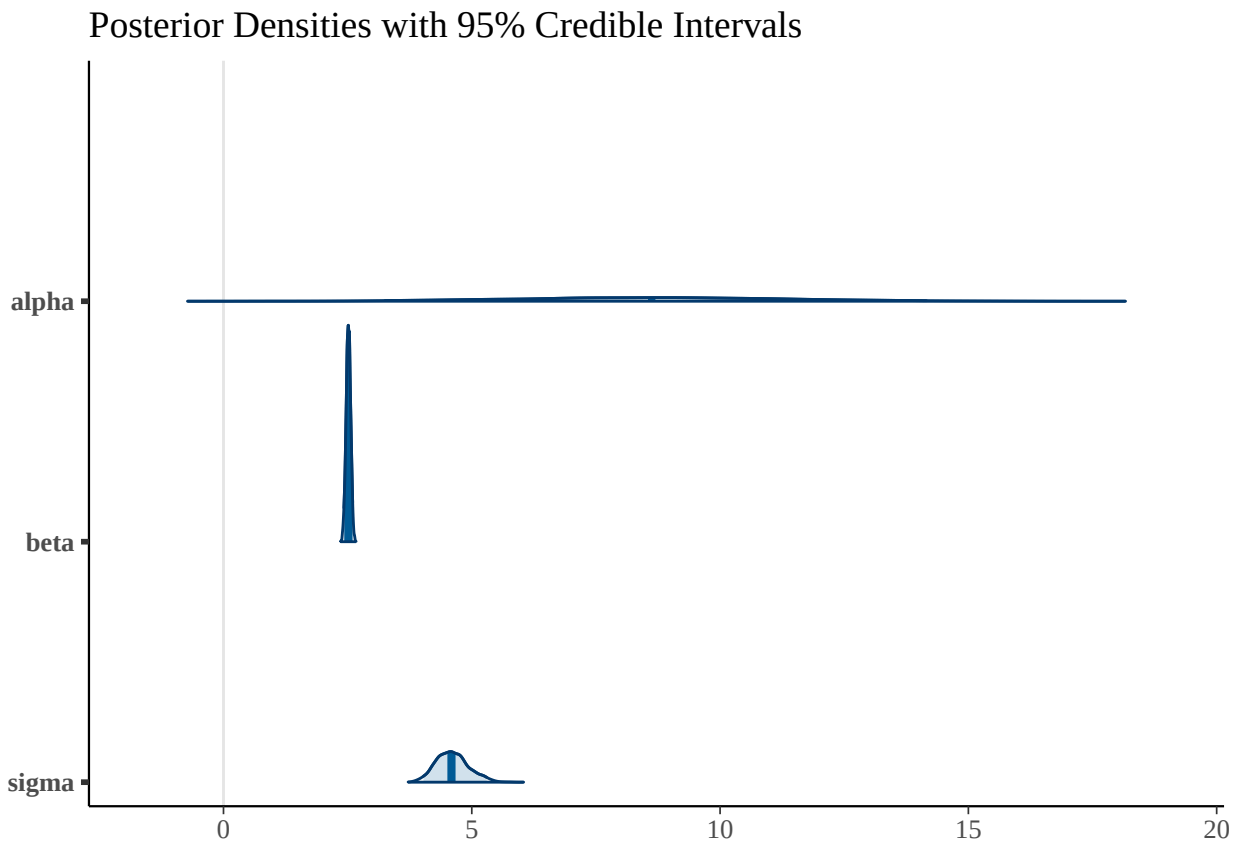
Posterior Density Plots

```
mcmc_dens_overlay(fit, pars = c("alpha", "beta", "sigma")) +
  ggtitle("Posterior Density Overlays")
```

Posterior Density Overlays

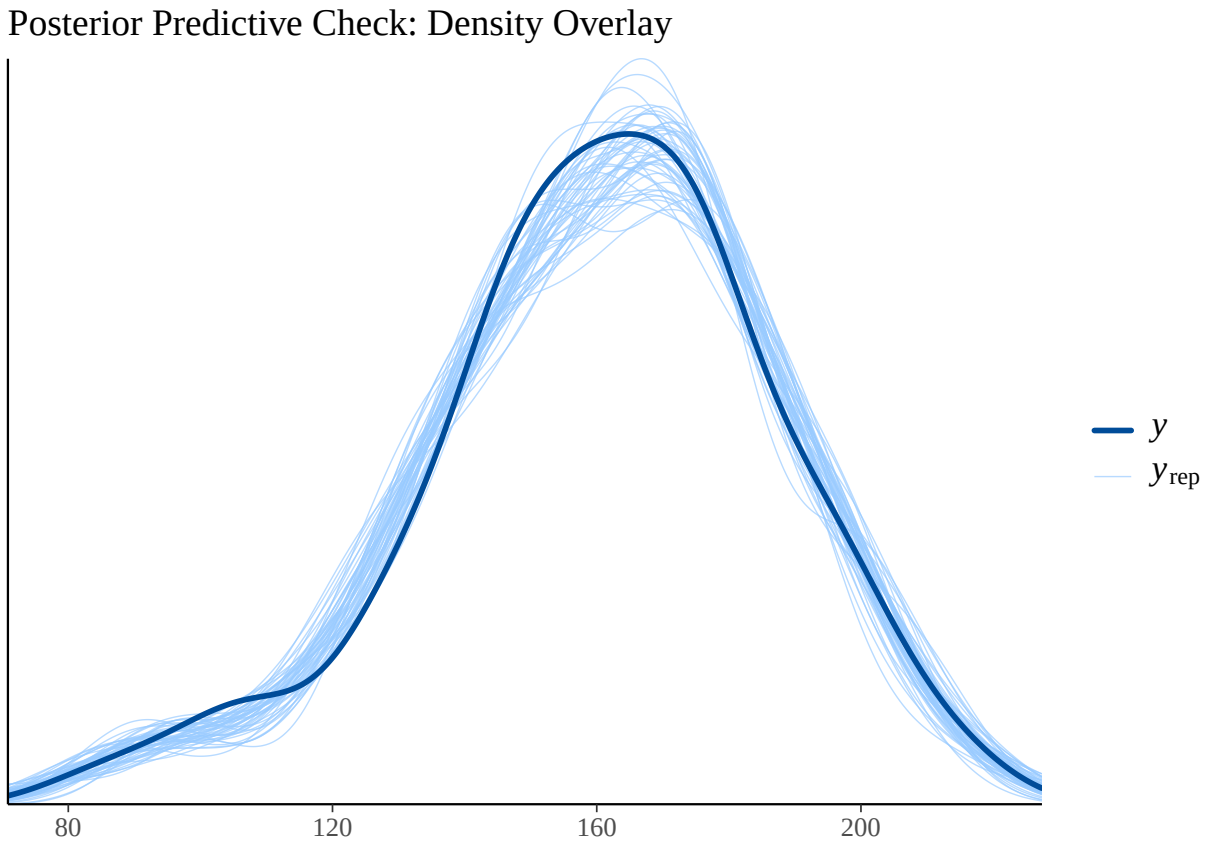



```
mcmc_areas(fit, pars = c("alpha", "beta", "sigma"), prob = 0.95) +
  ggtitle("Posterior Densities with 95% Credible Intervals")
```

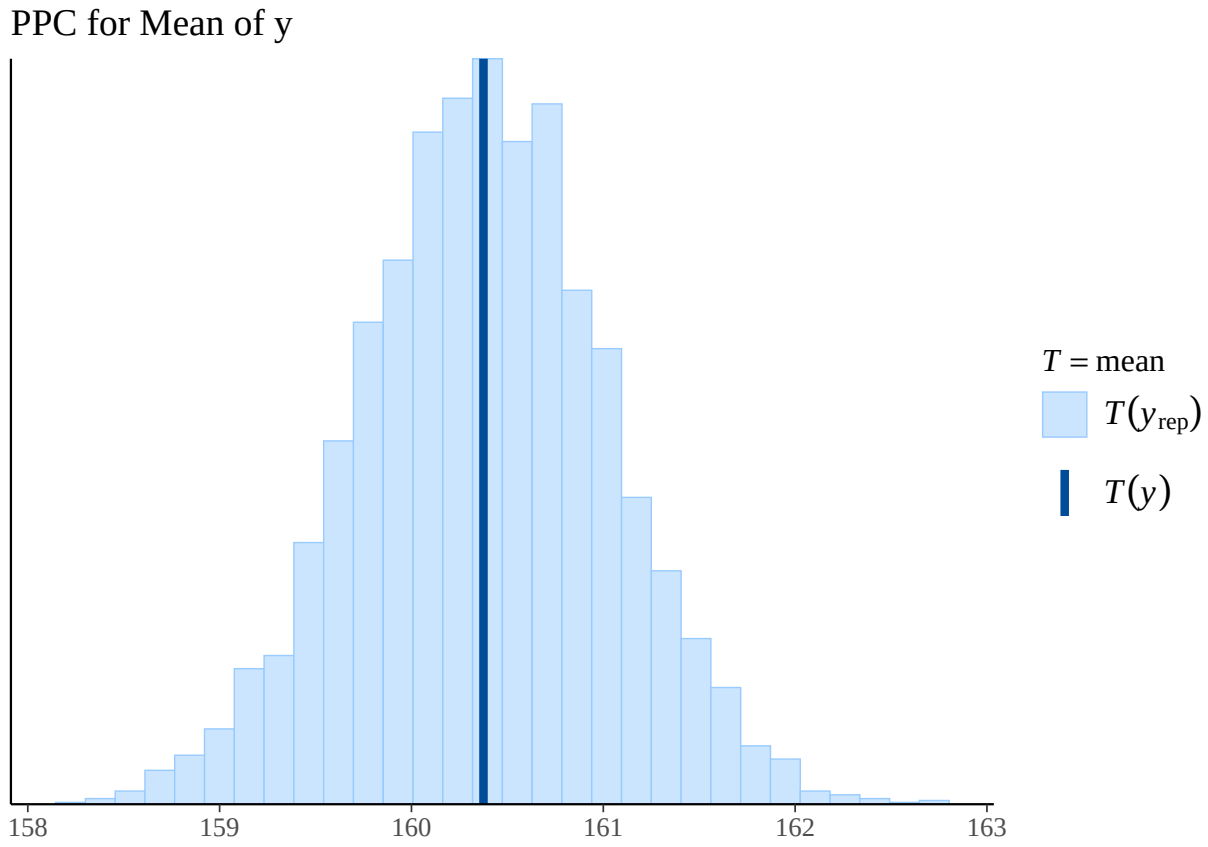


Part 6: Posterior Predictive Checks (PPCs)

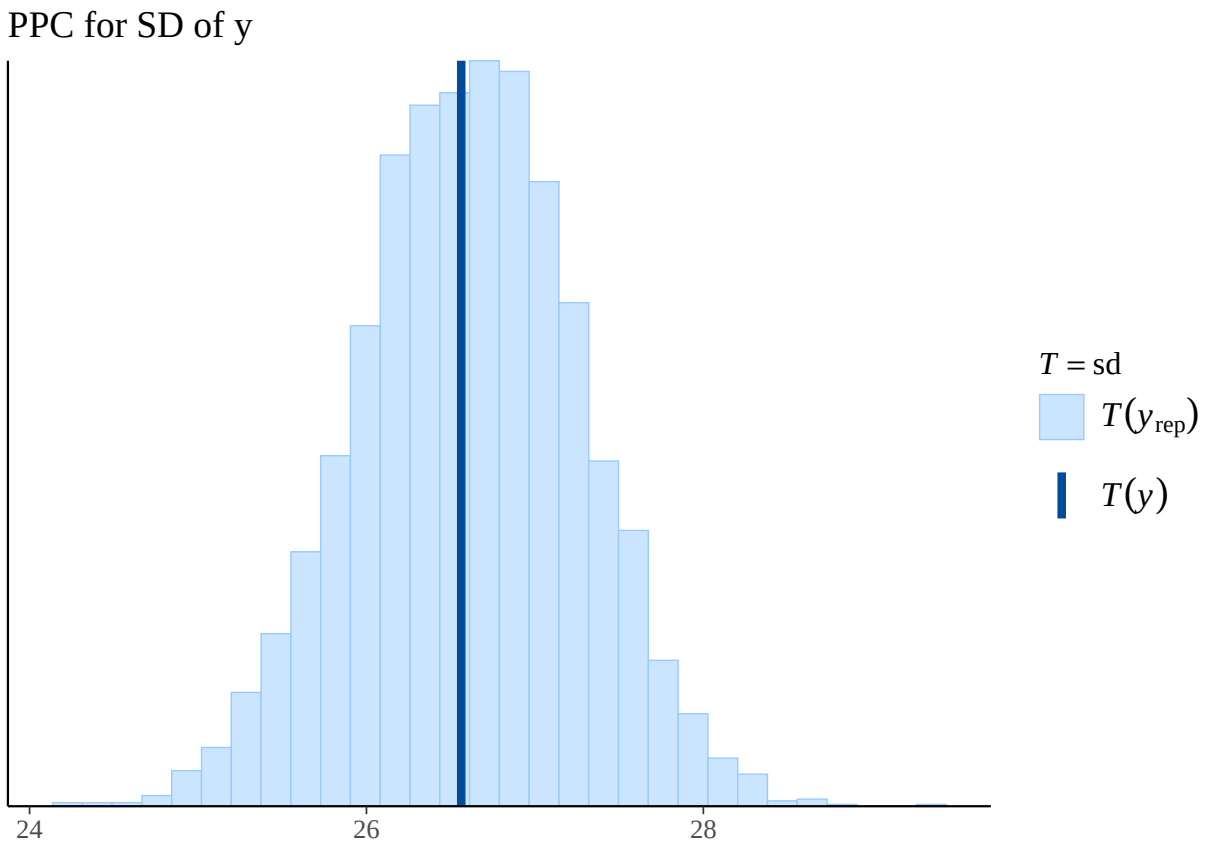
```
posterior_draws <- extract(fit)
y_rep_matrix <- posterior_draws$y_rep

color_scheme_set("brightblue")

ppc_dens_overlay(y = sim_data$y, yrep = y_rep_matrix[1:50, ]) +
  ggtitle("Posterior Predictive Check: Density Overlay")
```




```
ppc_stat(y = sim_data$y, yrep = y_rep_matrix, stat = "mean") +
  ggtitle("PPC for Mean of y")
```

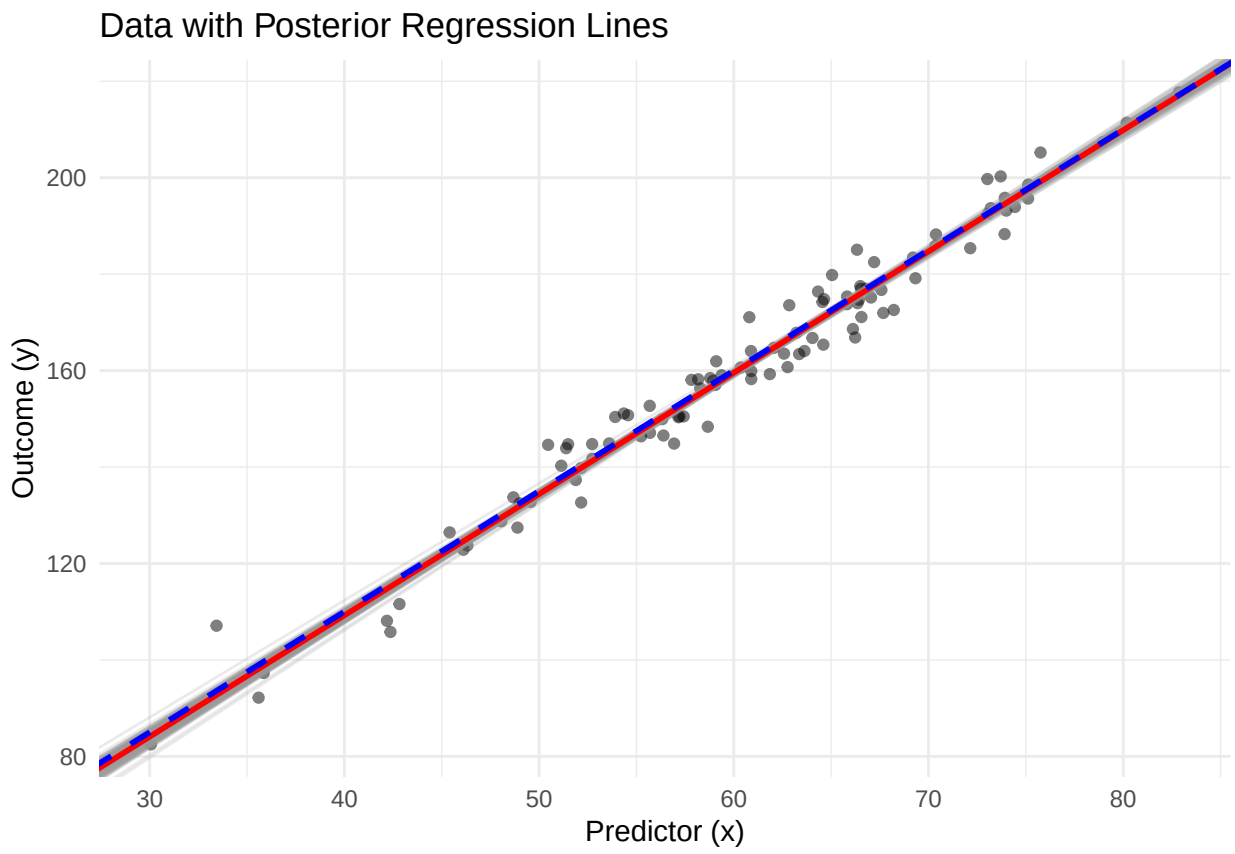



```
ppc_stat(y = sim_data$y, yrep = y_rep_matrix, stat = "sd") +
  ggtitle("PPC for SD of y")
```



Part 7: Interpretation and Visualization

```
alpha_samples <- posterior_draws$alpha
beta_samples <- posterior_draws$beta

p <- ggplot(sim_data, aes(x = x, y = y)) +
  geom_point(alpha = 0.5) +
  labs(title = "Data with Posterior Regression Lines", x = "Predictor (x)", y = "Outcome (y)") +
  theme_minimal()

num_lines_to_plot <- 100
num_available_samples <- length(alpha_samples)
sample_indices <- sample(num_available_samples, min(num_lines_to_plot, num_available_samples))

for (i in sample_indices) {
  p <- p + geom_abline(intercept = alpha_samples[i], slope = beta_samples[i], color = "grey60", alpha =
}

p <- p + geom_abline(intercept = mean(alpha_samples), slope = mean(beta_samples), color = "red", size =
p <- p + geom_abline(intercept = alpha_true, slope = beta_true, color = "blue", linetype = "dashed", si

print(p)
```



Part 8: Student Exploration Questions

Q1. Influence of Priors

- Q1.1: Change the prior for `sigma` to `sigma ~ student_t(7, 5, 1)` or `sigma ~ normal(5, 1) T[0,]`. Re-run the model. How does the posterior for `sigma`, `alpha`, and `beta` change?
- Q1.2: Change the prior for `beta` to a narrow prior `beta ~ normal(0, 0.1)`. Does the posterior move toward 0?
- Q1.3: Try a wrong but informative prior `beta ~ normal(-5, 0.5)`. Can the data override the prior?

Q2. Impact of Data

```
N_small <- 20
x_small <- rnorm(N_small, mean = 60, sd = 10)
y_deterministic_small <- alpha_true + beta_true * x_small
y_small <- rnorm(N_small, mean = y_deterministic_small, sd = sigma_true)
stan_data_small <- list(N = N_small, x = x_small, y = y_small)

fit_small_N <- sampling(model_compiled, data = stan_data_small, iter = 2000, warmup = 1000, chains = 4,
print(fit_small_N, pars = c("alpha", "beta", "sigma"))
```

- Q2.2: Increase the true error `sigma_true <- 15`. Re-simulate and re-fit. What changes?
- Q2.3: Standardize `x` and `y`. What happens to `alpha`, `beta`, and appropriate priors?

Q3. Model Diagnostics

```
fit_bad_mcmc <- sampling(model_compiled, data = stan_data, iter = 100, warmup = 50, chains = 4, seed =
print(fit_bad_mcmc, pars = c("alpha", "beta", "sigma"))
mcmc_trace(fit_bad_mcmc, pars = c("alpha", "beta", "sigma"))
```

- Examine Rhat and trace plots. Would you trust this model?

Q4. Interpreting Output

```
print(fit, pars = "beta", probs = c(0.05, 0.95))
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##      mean se_mean   sd   5%  95% n_eff Rhat
## beta 2.52       0 0.05 2.44 2.59  1240    1
##
## Samples were drawn using NUTS(diag_e) at Mon May 19 11:26:24 2025.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
beta_posterior_samples <- extract(fit)$beta
prob_beta_gt_2 <- mean(beta_posterior_samples > 2.0)
print(paste("Probability that beta > 2.0:", round(prob_beta_gt_2, 3)))
```

```
## [1] "Probability that beta > 2.0: 1"
```

Q5. Multiple Predictors (Advanced)

- Q5.1: Simulate a second predictor `x2 <- rnorm(N, mean = 30, sd = 5)` with `beta2_true <- -1.5`.
- Q5.2: Modify Stan code to include a matrix `X` of predictors and a vector `beta`.
- Q5.3: Update `stan_data` to include `K` and `X`. Re-fit and evaluate estimates of `beta1` and `beta2`.