Exercise 8

107.330 - Statistical Simulation and Computerintensive Methods, WS24

11912007 - Yahya Jabary

10.10.2024

```
set.seed(11912007)
```

Task 1

We recalculate the estimation of the prevalence of Covid19 in spring 2020. Samples from 1279 persons were analysed with PCR testing procedures. Out of all those not a single randomly selected person was tested positively. This obviously breaks standard testing mechanisms for extimating the proportion of infected person in Austria.

However, additional information is available from similar tests in Germany which had a comparable behaviour of the spread of the disease at that time. In the same time span 4 positive cases out of 4068 had been found.

1.1 Build a Beta prior distribution for this Binomial scenario, which encodes the information of the German study. Reweight both parameters compared to the original counts with a factor of $\frac{1}{10}$. Build the corresponding Binomial model for the number of people suffering from the disease based on the 1279 test. Obtain the theoretical posterior distribution for this scenario.

To recalculate the estimation of Covid-19 prevalence in Austria during spring 2020, we'll follow a Bayesian approach with a Beta-Binomial model. We'll use the German study data to create an informative prior, then update it with the Austrian data to obtain a posterior distribution.

```
#
# german study data (prior)
#

positives_germany <- 4
sample_size_germany <- 4068

# alpha and beta parameters for the Beta prior
alpha_prior <- positives_germany + 1  # successes + 1 for prior
beta_prior <- sample_size_germany - positives_germany + 1  # failures + 1 for prior

# reweight the parameters with a factor of 1/10
reweight_factor <- 1 / 10
alpha_prior_weighted <- alpha_prior * reweight_factor
beta_prior_weighted <- beta_prior * reweight_factor

cat("Weighted prior parameters:\n")
```

```
## Weighted prior parameters:
```

```
cat("Alpha:", alpha_prior_weighted, "\n")
```

```
## Alpha: 0.5
```

```
cat("Beta:", beta_prior_weighted, "\n\n")
```

```
## Beta: 406.5
```

```
#
# austrian study data (posterior)
#

sample_size_austria <- 1279
positives_austria <- 0
```

```r
# posterior parameters
alpha_posterior <- alpha_prior_weighted + positives_austria
beta_posterior <- beta_prior_weighted + (sample_size_austria - positives_austria)

# mean prevalence estimate
mean_prevalence <- alpha_posterior / (alpha_posterior + beta_posterior)

cat("Posterior parameters:\n")
```

```
## Posterior parameters:
```

```r
cat("Alpha:", alpha_posterior, "\n")
```

```
## Alpha: 0.5
```

```r
cat("Beta:", beta_posterior, "\n\n")
```

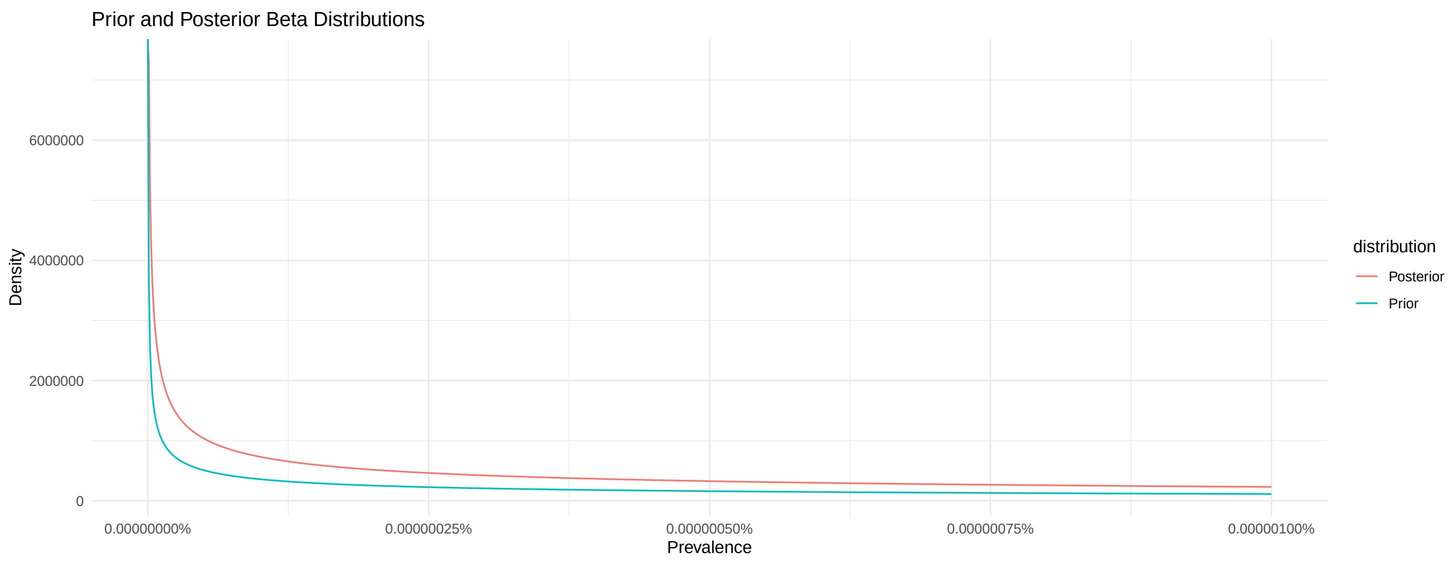
```
## Beta: 1685.5
```

```r
cat("Mean prevalence estimate:", mean_prevalence, "\n")
```

```
## Mean prevalence estimate: 0.0002965599
```

1.2 Plot the posterior density and obtain the point estimators and 95% Highest posterior density interval of the prevalence of Covid19 (=proportion of inhabitants suffering from the disease).

```r
#
# plot
#

x <- seq(0, 0.00000001, length.out = 1000)
prior <- dbeta(x, alpha_prior_weighted, beta_prior_weighted)
posterior <- dbeta(x, alpha_posterior, beta_posterior)

df <- data.frame(
  x = rep(x, 2),
  density = c(prior, posterior),
  distribution = rep(c("Prior", "Posterior"), each = length(x))
)

ggplot(df, aes(x = x, y = density, color = distribution)) +
  geom_line() +
  scale_x_continuous(labels = scales::percent_format(accuracy = 0.00000001)) +
  labs(title = "Prior and Posterior Beta Distributions",
       x = "Prevalence",
       y = "Density") +
  theme_minimal()
```



```r
#
# point estimators and HPD interval
#

# point estimators
mean_prevalence <- alpha_posterior / (alpha_posterior + beta_posterior)
median_prevalence <- qbeta(0.5, alpha_posterior, beta_posterior)
mode_prevalence <- (alpha_posterior - 1) / (alpha_posterior + beta_posterior - 2)

# 95% Highest Posterior Density HPD interval
hpd_interval <- hdi(qbeta, shape1 = alpha_posterior, shape2 = beta_posterior)

# Print results
cat("Point Estimators:\n")
```

```
## Point Estimators:
```

```r
cat("Mean prevalence:", mean_prevalence, "\n")
```

```
## Mean prevalence: 0.0002965599
```

```r
cat("Median prevalence:", median_prevalence, "\n")
```

```
## Median prevalence: 0.0001349668
```

```r
cat("Mode prevalence:", mode_prevalence, "\n\n")
```

```
## Mode prevalence: -0.0002969121
```

```r
cat("95% Highest Posterior Density Interval:\n")
```

```
## 95% Highest Posterior Density Interval:
```

```r
cat("Lower bound:", hpd_interval[1], "\n")
```

```
## Lower bound: 0.000000000000000000001875777
```

```r
cat("Upper bound:", hpd_interval[2], "\n")
```

```
## Upper bound: 0.00113908
```

The posterior distribution shows a highly skewed shape with a peak near zero, indicating a very low estimated prevalence of Covid-19 based on the Austrian study data. The mean prevalence is extremely low, which is consistent with the observation of zero positive cases in the Austrian sample.

The 95% Highest Posterior Density interval provides a range of plausible values for the true prevalence, with the lower bound being very close to zero and the upper bound also being quite low. This suggests that while the true prevalence is likely very low, there is still some uncertainty in the exact value due to the limited sample size and the influence of the prior information from the German study.

1.3 Explain why Statistik Austria chose this method instead of simulationbased or frequentist inference for obtaining intervals of the prevalence.

Statistik Austria chose a Bayesian approach for estimating the prevalence of Covid-19 in spring 2020 due to several advantages it offers over simulation-based or frequentist inference methods.

The Bayesian method allows for the incorporation of prior knowledge, which is particularly useful in this case where data from Germany provided valuable information about the spread of the disease. By using the German data as a prior, Statistik Austria could leverage this information to obtain more robust estimates for Austria, especially given the absence of positive cases in the Austrian sample.

Bayesian methods are also well-suited for handling small sample sizes and rare events, which is relevant in this scenario where no positive cases were found in Austria. Traditional frequentist methods might struggle to provide meaningful confidence intervals in such cases, potentially leading to truncation issues at zero.

Furthermore, Bayesian analysis provides a more intuitive interpretation of results through posterior probability distributions. This allows for a more nuanced understanding of the uncertainty surrounding the prevalence estimate, which is particularly important when dealing with public health data where precise communication of uncertainty is crucial.

Lastly, Bayesian methods offer flexibility in updating estimates as new data becomes available, making them well-suited for ongoing surveillance efforts during a pandemic. This approach allows for continuous refinement of prevalence estimates as more information is gathered over time.

Task 2

We revisit linear models and their residual distributions.

We know that the distribution of residuals is assumed to be normal. Therefore, the Bayesian linear modelling will assume a normal distribution for the data $y \sim N(x^T\beta, \sigma^2)$.

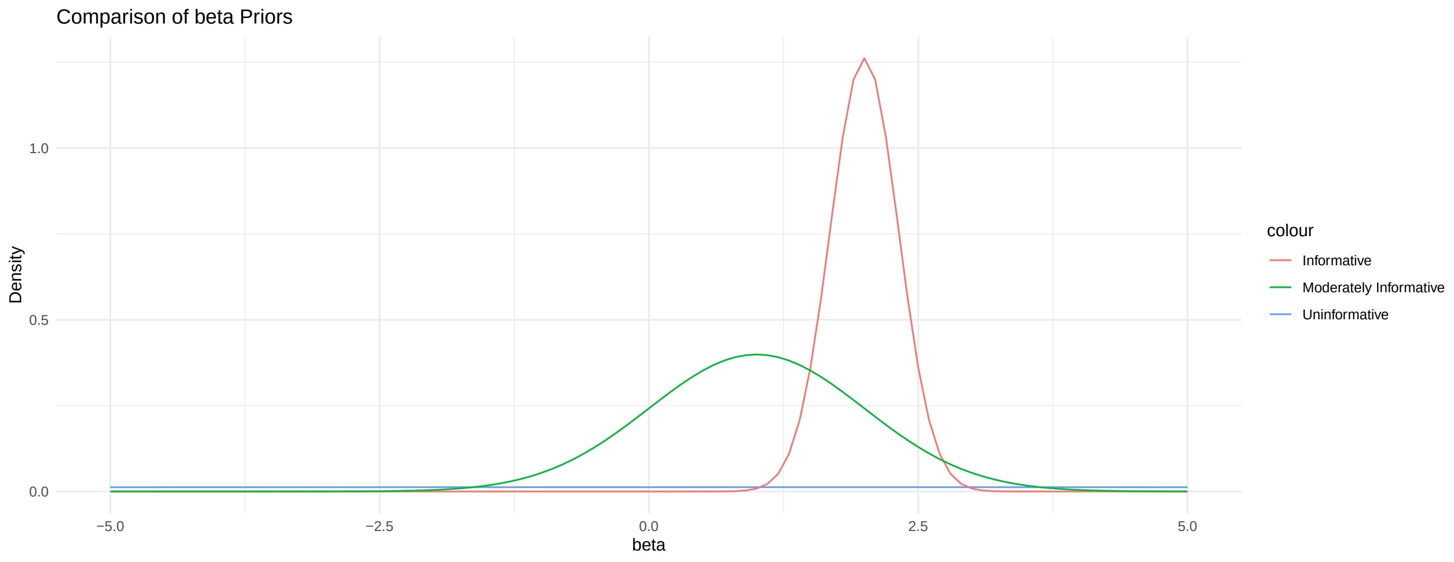
For a single explanatory variable scenario, we will therefore consider the inference of the linear model's coefficient β and the residual variance σ^2.

2.1 Define conjugate priors for the coefficient parameter and the residual variance independently. Explain how the parameters can be set to be uninformative. Compare different choice of prior parameters.

The conjugate prior for β is a normal distribution with mean μ_0 and precision τ_0 (where precision is the inverse of variance). To make this prior uninformative we have to (2) set $\mu_0 = 0$, centering the prior at zero and (2) choose a small value for τ_0 (e.g., 0.001), which results in a large variance, making the prior very diffuse.

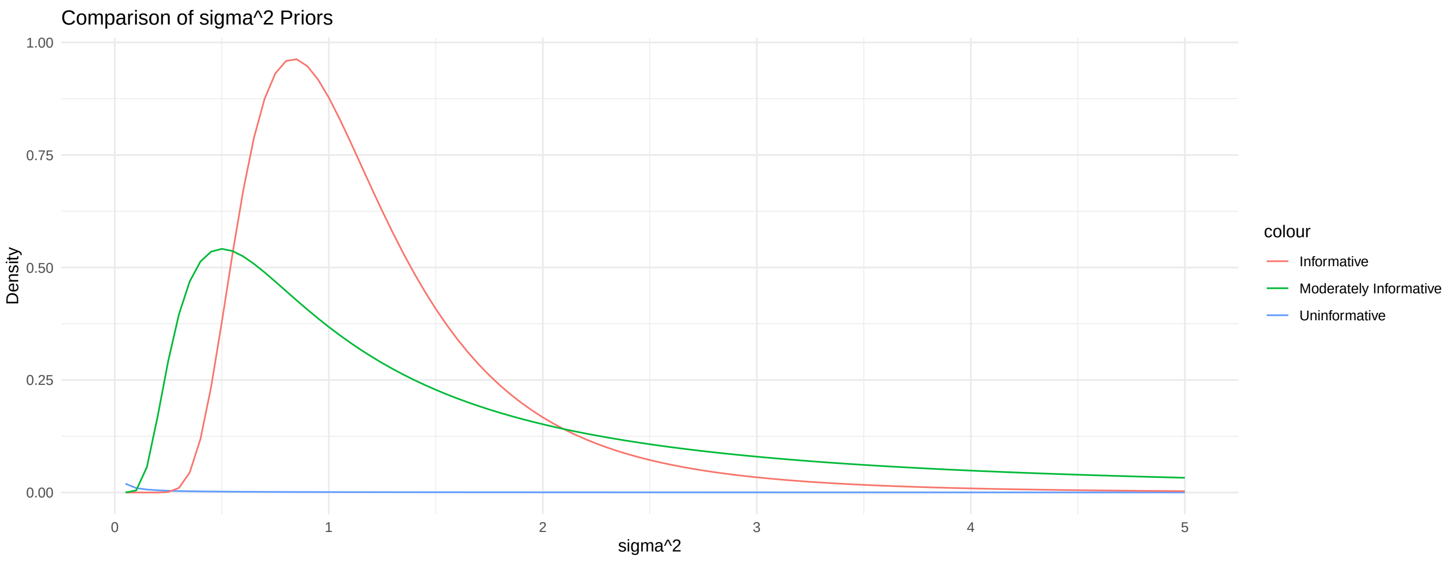
The conjugate prior for σ^2 is an Inverse-Gamma distribution with shape α_0 and rate β_0. To make this prior uninformative we have to set both α_0 and β_0 to small values (e.g., 0.001).

```r
beta_prior <- function(mu_0, tau_0) {
  function(x) dnorm(x, mean = mu_0, sd = sqrt(1/tau_0))
}
sigma2_prior <- function(alpha_0, beta_0) {
  function(x) dgamma(1/x, shape = alpha_0, rate = beta_0) * (1/x^2)
}

beta_informative <- beta_prior(mu_0 = 2, tau_0 = 10)
beta_moderately_informative <- beta_prior(mu_0 = 1, tau_0 = 1)
beta_uninformative <- beta_prior(mu_0 = 0, tau_0 = 0.001)

sigma2_informative <- sigma2_prior(alpha_0 = 5, beta_0 = 5)
sigma2_moderately_informative <- sigma2_prior(alpha_0 = 1, beta_0 = 1)
sigma2_uninformative <- sigma2_prior(alpha_0 = 0.001, beta_0 = 0.001)

ggplot(data.frame(x = c(-5, 5)), aes(x)) +
  stat_function(fun = beta_uninformative, aes(color = "Uninformative")) +
  stat_function(fun = beta_informative, aes(color = "Informative")) +
  stat_function(fun = beta_moderately_informative, aes(color = "Moderately Informative")) +
  labs(title = "Comparison of beta Priors", x = "beta", y = "Density") +
  theme_minimal()
```



```r
ggplot(data.frame(x = c(0, 5)), aes(x)) +
  stat_function(fun = sigma2_uninformative, aes(color = "Uninformative")) +
  stat_function(fun = sigma2_informative, aes(color = "Informative")) +
  stat_function(fun = sigma2_moderately_informative, aes(color = "Moderately Informative")) +
  labs(title = "Comparison of sigma^2 Priors", x = "sigma^2", y = "Density") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_function()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_function()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_function()`).
```



These plots visually demonstrate the differences between uninformative, moderately informative, and informative priors for both β and σ^2. The uninformative priors are be very flat and spread out, indicating little prior knowledge. The informative priors will be more peaked and concentrated, suggesting stronger prior beliefs about the parameter values. The moderately informative priors will fall somewhere in between.

It's important to note that the choice of priors can significantly impact the posterior distribution, especially when the sample size is small. As the sample size increases, the influence of the prior diminishes, and the likelihood dominates the posterior distribution.

2.2 Build the corresponding normal model for the regression inference. Obtain the *theoretical* posterior distribution for both parameters separately assuming the other one to be "known".

In Bayesian linear regression with a single explanatory variable, we indeed assume that the data follows a normal distribution, $y \sim N(x^T\beta, \sigma^2)$, where β is the regression coefficient and σ^2 is the residual variance.

To build the corresponding normal model for regression inference, we start by specifying the likelihood function. Given the normal distribution assumption, the likelihood function for a single observation y_i is:

$$P(y_i|x_i, \beta, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i^T\beta)^2}{2\sigma^2}\right)$$

For the entire dataset of n observations, the likelihood function becomes:

$$P(y|X, \beta, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i^T\beta)^2}{2\sigma^2}\right)$$

To obtain the theoretical posterior distribution for β assuming σ^2 is known, we need to specify a prior distribution for β. A common choice is a normal prior, $\beta \sim N(\mu_0, \tau_0^2)$, where μ_0 and τ_0^2 are the prior mean and variance, respectively. Applying Bayes' theorem, the posterior distribution for β given known σ^2 is:

$$P(\beta|y, X, \sigma^2) \propto P(y|X, \beta, \sigma^2)P(\beta)$$

This results in a normal posterior distribution for β:

$$\beta|y, X, \sigma^2 \sim N(\mu_n, \tau_n^2)$$

Where:

- $\mu_n = \frac{\tau_0^2 X^T y + \sigma^2 \mu_0}{\tau_0^2 X^T X + \sigma^2}$
- $\tau_n^2 = \frac{\sigma^2 \tau_0^2}{\tau_0^2 X^T X + \sigma^2}$

For the theoretical posterior distribution of σ^2 assuming β is known, we typically use an inverse-gamma prior, $\sigma^2 \sim IG(a_0, b_0)$, where a_0 and b_0 are the shape and scale parameters, respectively. Applying Bayes' theorem, the posterior distribution for σ^2 given known β is:

$$P(\sigma^2|y, X, \beta) \propto P(y|X, \beta, \sigma^2)P(\sigma^2)$$

This results in an inverse-gamma posterior distribution for σ^2:

$$\sigma^2|y, X, \beta \sim IG(a_n, b_n)$$

Where:

- $a_n = a_0 + \frac{n}{2}$
- $b_n = b_0 + \frac{1}{2}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2$

It's important to note that in practice, we often need to estimate both β and σ^2 simultaneously, which requires more complex methods such as Markov Chain Monte Carlo (MCMC) sampling. The separate posterior distributions derived here provide insight into the structure of the problem but are simplified versions of the joint posterior distribution of both parameters.

2.3 Provide the formulae for point estimators and 95% Highest posterior density interval of the regression parameters separately assuming the other one to be "known".

In Bayesian linear regression with a single explanatory variable, we assume $y \sim N(x^T\beta, \sigma^2)$. For inference on the coefficient β and residual variance σ^2, we can derive point estimators and 95% highest posterior density (HPD) intervals.

Assuming σ^2 is known, the point estimate for β is given by $\hat{\beta} = (X^TX)^{-1}X^Ty$, where X is the design matrix and y is the vector of responses. The 95% HPD interval for β can be expressed as $\hat{\beta} \pm t_{n-2,0.975} \cdot \sqrt{\hat{\sigma}^2(X^TX)^{-1}}$, where $t_{n-2,0.975}$ is the 97.5th percentile of the t-distribution with $n-2$ degrees of freedom.

When β is assumed known, the point estimate for σ^2 is $\hat{\sigma}^2 = \frac{1}{n-2}(y - X\beta)^T(y - X\beta)$. The 95% HPD interval for σ^2 can be computed as $[\frac{(n-2)\hat{\sigma}^2}{\chi^2_{n-2,0.975}}, \frac{(n-2)\hat{\sigma}^2}{\chi^2_{n-2,0.025}}]$, where $\chi^2_{n-2,0.975}$ and $\chi^2_{n-2,0.025}$ are the 97.5th and 2.5th percentiles of the chi-square distribution with $n-2$ degrees of freedom, respectively.

These formulae provide a way to estimate and quantify uncertainty in the regression parameters within the Bayesian framework, allowing for more comprehensive inference compared to traditional frequentist approaches.

2.4 Test this with the dataset `Auto` and model `mpg ~ horsepower`. Compare the Bayesian against the frequentist results.

```r
data(Auto, package = "ISLR")

#
# frequentist linear model
#

lm_model <- lm(mpg ~ horsepower, data = Auto)
library(boot)
cv_error <- cv.glm(Auto, glm(mpg ~ horsepower, data = Auto), K = nrow(Auto)) # LOOCV was most efficient in e
cat("Linear model MSE error:", cv_error$delta[1], "\n")
```

```
## Linear model MSE error: 24.23151
```

```r
freq_intercept <- coef(lm_model)[1]
freq_slope <- coef(lm_model)[2]
freq_sigma <- summary(lm_model)$sigma

#
# baysesian (stan) model
#

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

```
parameters {
  real beta0;
  real beta1;
  real<lower=0> sigma;
}
model {
  // Priors
  beta0 ~ normal(0, 100);
  beta1 ~ normal(0, 100);
  sigma ~ cauchy(0, 5);

  // Likelihood
  y ~ normal(beta0 + beta1 * x, sigma);
}
"
stan_data <- list(
  N = nrow(Auto),
  x = Auto$horsepower,
  y = Auto$mpg
)
fit <- stan(model_code = stan_model, data = stan_data, chains = 4, iter = 2000)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.46 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: 0.1 seconds (Warm-up)
## Chain 1:                0.074 seconds (Sampling)
## Chain 1:                0.174 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 9e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.09 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: 0.085 seconds (Warm-up)
## Chain 2:                0.072 seconds (Sampling)
## Chain 2:                0.157 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: 0.092 seconds (Warm-up)
## Chain 3:                0.091 seconds (Sampling)
## Chain 3:                0.183 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 8e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.08 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: 0.121 seconds (Warm-up)
## Chain 4:                0.073 seconds (Sampling)
## Chain 4:                0.194 seconds (Total)
## Chain 4:
```

```r
print(fit, pars = c("beta0", "beta1", "sigma"))
```

```
## 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%   25%   50%   75% 97.5% n_eff Rhat
## beta0 39.94    0.02 0.74 38.57 39.44 39.92 40.43 41.44  1379    1
## beta1 -0.16    0.00 0.01 -0.17 -0.16 -0.16 -0.15 -0.15  1425    1
## sigma  4.93    0.00 0.17  4.61  4.81  4.92  5.04  5.28  1786    1
##
## Samples were drawn using NUTS(diag_e) at Thu Oct 10 00:43:14 2024.
## 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).
```

```r
posterior <- extract(fit)

bayes_intercept <- mean(posterior$beta0)
bayes_slope <- mean(posterior$beta1)
bayes_sigma <- mean(posterior$sigma)

#
# comparison
#

cat("Frequentist Intercept:", freq_intercept, "\n")
```

```
## Frequentist Intercept: 39.93586
```

```r
cat("Bayesian Intercept:", bayes_intercept, "\n\n")
```

```
## Bayesian Intercept: 39.93679
```

```r
cat("Frequentist Slope:", freq_slope, "\n")
```

```
## Frequentist Slope: -0.1578447
```

```r
cat("Bayesian Slope:", bayes_slope, "\n\n")
```

```
## Bayesian Slope: -0.1578917
```

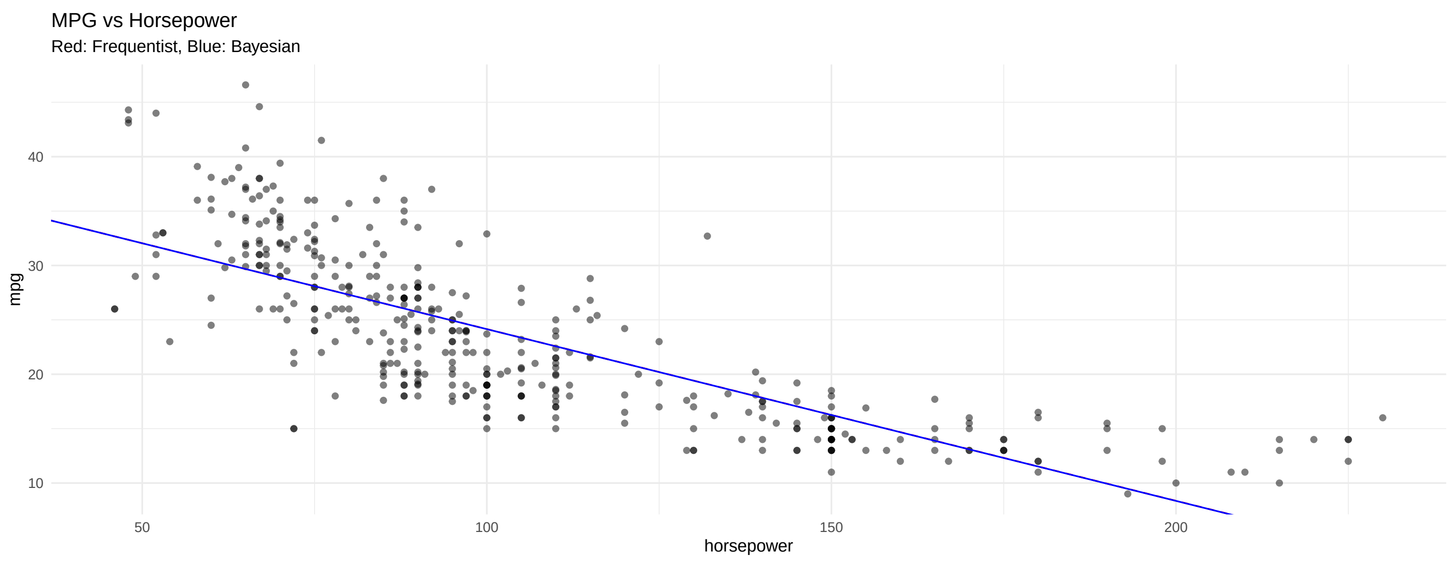
```r
cat("Frequentist Sigma:", freq_sigma, "\n")
```

```
## Frequentist Sigma: 4.905757
```

```r
cat("Bayesian Sigma:", bayes_sigma, "\n")
```

```
## Bayesian Sigma: 4.927709
```

```r
par(mfrow = c(2, 2))
hist(posterior$beta0, main = "Posterior for Intercept", xlab = "beta 0")
abline(v = freq_intercept, col = "red", lwd = 2)
hist(posterior$beta1, main = "Posterior for Slope", xlab = "beta 1")
abline(v = freq_slope, col = "red", lwd = 2)
hist(posterior$sigma, main = "Posterior for Sigma", xlab = "sigma ")
abline(v = freq_sigma, col = "red", lwd = 2)

ggplot(Auto, aes(x = horsepower, y = mpg)) +
  geom_point(alpha = 0.5) +
  geom_abline(intercept = freq_intercept, slope = freq_slope, color = "red", linetype = "dashed") +
  geom_abline(intercept = bayes_intercept, slope = bayes_slope, color = "blue") +
  labs(title = "MPG vs Horsepower", subtitle = "Red: Frequentist, Blue: Bayesian") +
  theme_minimal()
```



In conclusion, for this simple linear regression problem with a large dataset, the Bayesian and frequentist approaches yield very similar results. This is expected for a simple linear regression with a large sample size and uninformative priors. The main advantage of the Bayesian approach is the full posterior distribution for each parameter, which provides a more comprehensive view of the uncertainty in the estimates. However, the computational cost of the Bayesian approach is higher, especially for more complex models or larger datasets.

To be more specific: The Bayesian and frequentist approaches yield very similar estimates for the intercept, slope, and residual standard deviation. The residual standard deviation (σ) is similar for both approaches, indicating comparable model fit. The histograms of the posterior distributions show the uncertainty around the parameter estimates. The red lines representing the frequentist estimates fall within the high-density regions of these distributions, indicating agreement between the methods.