Bayesian vs Frequentist Regression Analysis

End-to-end comparison of Bayesian linear regression models with shrinkage priors (Normal, Laplace, Horseshoe) against a frequentist OLS baseline, implemented in R and Stan.

Bayesian vs Frequentist Regression Analysis

End-to-end comparison of Bayesian linear regression models with multiple shrinkage priors (Normal, Laplace, Horseshoe) against a frequentist OLS baseline, implemented in R and Stan.

Full R + Stan Implementation

The following script embeds all Stan models directly in R, performs MCMC sampling, and generates posterior summaries and interval comparison plots.

```
# ============================================================
# Bayesian Linear Regression with Alternative Shrinkage Priors
# Platform: R + Stan (rstan)
# ============================================================

# ---- Libraries ----
suppressPackageStartupMessages({
  library(rstan)
  library(dplyr)
  library(tidyr)
  library(tibble)
  library(ggplot2)
  library(medicaldata)
})

# ---- Reproducibility & performance ----
set.seed(2025)
n_cores <- max(1, parallel::detectCores() - 1)
options(mc.cores = n_cores)
rstan_options(auto_write = TRUE)
Sys.setenv(STAN_NUM_THREADS = n_cores)

# ============================================================
# 1) Data loading & preprocessing
#    Dataset: medicaldata::covid_testing
# ============================================================
data("covid_testing", package = "medicaldata")

covid_testing <- covid_testing %>%
  select(-subject_id, -fake_first_name, -fake_last_name) %>%
  tidyr::drop_na()

# Convert character columns to factors
covid_testing <- covid_testing %>%
  mutate(across(where(is.character), as.factor))

# Standardize selected numeric columns (including the target)
std_cols <- c("pan_day", "age", "drive_thru_ind", "ct_result", "orderset", "col_rec_tat", "rec_ver_ta
std_cols <- std_cols[std_cols %in% names(covid_testing)]
```

```
covid_testing[std_cols] <- scale(covid_testing[std_cols])

# Collapse high-cardinality categorical variables into top-3 + "Other"
collapse_top3 <- function(col) {
  freqs <- sort(table(col), decreasing = TRUE)
  top3 <- names(freqs)[1:min(3, length(freqs))]
  x <- as.character(col)
  x[!x %in% top3] <- "Other"
  factor(x, levels = c(top3, "Other"))
}

to_collapse <- names(which(sapply(covid_testing, function(col) is.factor(col) && nlevels(col) > 3)))
if (length(to_collapse) > 0) {
  covid_testing[to_collapse] <- lapply(covid_testing[to_collapse], collapse_top3)
}

# Build design matrix
# - Predict ct_result using all remaining fields (excluding identifiers and the original result label
covid_testing <- covid_testing %>% select(-test_id)

predictors <- covid_testing %>% select(-ct_result, -result)
X <- model.matrix(~ ., data = predictors)
Y <- as.vector(covid_testing$ct_result)

N <- nrow(X)
K <- ncol(X)

# ============================================================
# 2) Stan model definitions (embedded)
# ============================================================

stan_normal_prior <- "
data {
  int<lower=1> N;
  int<lower=1> K;
  matrix[N, K] X;
  vector[N] Y;


  real<lower=0> a0;
  real<lower=0> b0;
  vector[K] m0;
  vector[K] lambda0;

}

parameters {
  vector[K] beta;
  real<lower=0> sigma2;
}

model {

  sigma2 ~ inv_gamma(a0, b0);



  for (k in 1:K) {
    beta[k] ~ normal(m0[k], sqrt(sigma2 / lambda0[k]));
  }

  Y ~ normal(X * beta, sqrt(sigma2));
}
"

stan_laplace_prior <- "
data {
  int<lower=0> N;
  int<lower=0> K;
  matrix[N, K] X;
  vector[N] Y;
  vector[K] m0;
```

```
  vector[K] lambda0;
  real a0;
  real b0;
}

parameters {
  vector[K] beta;
  real<lower=0> sigma;
}

model {

  for (k in 1:K) {
    beta[k] ~ double_exponential(m0[k], lambda0[k]);
  }


  sigma ~ inv_gamma(a0, b0);


  Y ~ normal(X * beta, sigma);
}
"

stan_horseshoe_prior <- "
data {
  int<lower=0> N;
  int<lower=0> K;
  matrix[N, K] X;
  vector[N] y;
  real a0;
  real b0;
}

parameters {
  vector[K] beta;
  real<lower=0> sigma;
  real<lower=0> tau;
  vector<lower=0>[K] lambda;
}

model {

  sigma ~ inv_gamma(a0, b0);


  tau ~ cauchy(0, 1);


  lambda ~ cauchy(0, 1);


  beta ~ normal(0, sigma * tau * lambda);


  y ~ normal(X * beta, sigma);
}
"

# Compile (once)
mod_normal    <- stan_model(model_code = stan_normal_prior)
mod_laplace   <- stan_model(model_code = stan_laplace_prior)
mod_horseshoe <- stan_model(model_code = stan_horseshoe_prior)

# ============================================================
# 3) Fit models
#    - Normal prior (two variants of lambda0)
#    - Laplace prior (L1 shrinkage)
#    - Horseshoe prior (strong shrinkage for sparse signals)
# ============================================================

# Common hyperparameters (kept explicit for clarity)
```

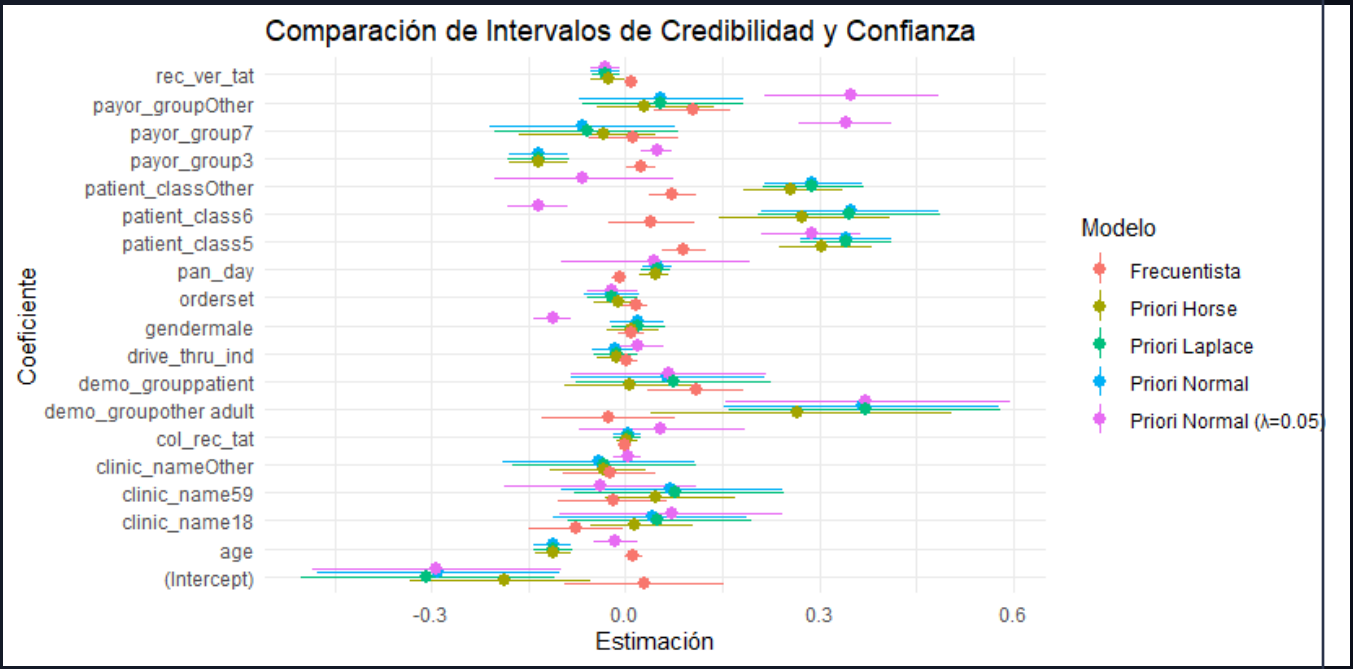
```r
a0 <- 1
b0 <- 1
m0 <- rep(0, K)

lambda0_default <- c(10, rep(1, K - 1))
lambda0_strong  <- c(10, rep(0.05, K - 1))

data_normal_default <- list(N = N, K = K, X = X, Y = Y, a0 = a0, b0 = b0, m0 = m0, lambda0 = lambda0_
data_normal_strong  <- list(N = N, K = K, X = X, Y = Y, a0 = a0, b0 = b0, m0 = m0, lambda0 = lambda0_

data_laplace <- list(N = N, K = K, X = X, y = Y, a0 = a0, b0 = b0, m0 = m0, lambda0 = lambda0_default
data_horseshoe <- list(N = N, K = K, X = X, y = Y, a0 = a0, b0 = b0)

fit_normal_default <- sampling(
  mod_normal,
  data = data_normal_default,
  iter = 2000,
  warmup = 1000,
  chains = 4,
  seed = 2025
)

fit_normal_strong <- sampling(
  mod_normal,
  data = data_normal_strong,
  iter = 2000,
  warmup = 1000,
  chains = 4,
  seed = 2025,
  control = list(adapt_delta = 0.95)
)

fit_laplace <- sampling(
  mod_laplace,
  data = data_laplace,
  iter = 2000,
  warmup = 1000,
  chains = 4,
  seed = 2025
)

fit_horseshoe <- sampling(
  mod_horseshoe,
  data = data_horseshoe,
  iter = 2000,
  warmup = 1000,
  chains = 4,
  seed = 2025
)

# ============================================================
# 4) Posterior summaries (credible intervals) + quick diagnostics
# ============================================================

summarize_beta <- function(fit, X_names, model_name) {
  draws <- rstan::extract(fit)
  beta <- as.data.frame(draws$beta)
  colnames(beta) <- X_names

  intervals <- beta %>%
    pivot_longer(cols = everything(), names_to = "term", values_to = "value") %>%
    group_by(term) %>%
    summarise(
      estimate = mean(value),
      lower = quantile(value, 0.025),
      upper = quantile(value, 0.975),
      .groups = "drop"
    ) %>%
    mutate(model = model_name, width = upper - lower)

  list(beta = beta, intervals = intervals)
}
```

```r
x_names <- colnames(X)

s_normal_default <- summarize_beta(fit_normal_default, x_names, "Normal prior (lambda0 = 1)")
s_normal_strong  <- summarize_beta(fit_normal_strong,  x_names, "Normal prior (lambda0 = 0.05)")
s_laplace        <- summarize_beta(fit_laplace,        x_names, "Laplace prior")
s_horseshoe      <- summarize_beta(fit_horseshoe,      x_names, "Horseshoe prior")

# Frequentist baseline
fit_lm <- lm(ct_result ~ ., data = bind_cols(ct_result = covid_testing$ct_result, as.data.frame(predi
ci_lm <- confint(fit_lm) %>%
  as.data.frame() %>%
  tibble::rownames_to_column("term") %>%
  rename(lower = `2.5 %`, upper = `97.5 %`) %>%
  mutate(
    estimate = coef(fit_lm),
    model = "Frequentist OLS",
    width = upper - lower
  )

all_intervals <- bind_rows(
  s_normal_default$intervals,
  s_normal_strong$intervals,
  s_laplace$intervals,
  s_horseshoe$intervals,
  ci_lm
)

# Rank models by mean interval width (excluding the intercept if desired)
model_ranking <- all_intervals %>%
  filter(term != "(Intercept)") %>%
  group_by(model) %>%
  summarise(mean_width = mean(width, na.rm = TRUE), .groups = "drop") %>%
  arrange(mean_width)

print(model_ranking)

# ============================================================
# 5) Portfolio-ready visualizations
# ============================================================

# --- Credible interval comparison across models ---
plot_ci <- all_intervals %>%
  filter(term != "(Intercept)") %>%
  mutate(term = factor(term, levels = rev(unique(term)))) %>%
  ggplot(aes(x = estimate, y = term, xmin = lower, xmax = upper, color = model)) +
  geom_pointrange(position = position_dodge(width = 0.6), alpha = 0.9) +
  geom_vline(xintercept = 0, linetype = "dashed") +
  labs(
    title = "Coefficient Uncertainty: Bayesian Credible Intervals vs Frequentist Confidence Intervals
    x = "Coefficient estimate",
    y = "Predictor"
  ) +
  theme_minimal()

print(plot_ci)

# --- Traceplot for beta (pick one model for a clean report) ---
# Tip: For large K, consider selecting a subset of coefficients.
rstan::stan_trace(fit_horseshoe, pars = "beta") +
  ggtitle("Traceplot (Horseshoe prior): beta coefficients")

# --- Posterior density (example: horseshoe) ---
beta_long_hs <- s_horseshoe$beta %>%
  mutate(iteration = row_number()) %>%
  pivot_longer(cols = -iteration, names_to = "term", values_to = "value")

ggplot(beta_long_hs, aes(x = value)) +
  geom_density(alpha = 0.6) +
  facet_wrap(~ term, scales = "free", ncol = 4) +
  geom_vline(xintercept = 0, linetype = "dashed") +
  labs(
    title = "Posterior Distributions (Horseshoe prior)",
```

```
    x = "Value",
    y = "Density"
  ) +
  theme_minimal() +
  theme(strip.text = element_text(size = 8))
```

Credible vs Confidence Intervals (All Priors)

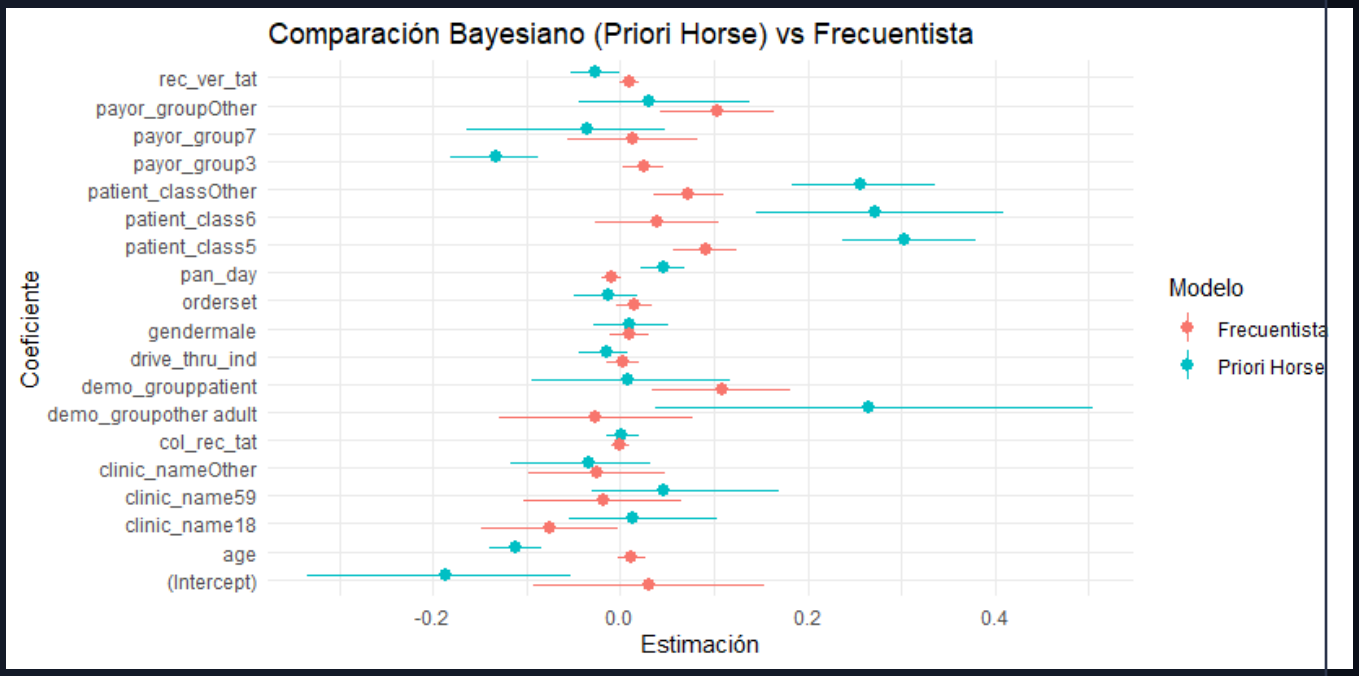
This visualization compares Bayesian credible intervals across multiple prior choices with frequentist confidence intervals, highlighting the effect of shrinkage.



Bayesian credible intervals (Normal, Laplace, Horseshoe) versus frequentist confidence intervals.

Horseshoe Prior vs Frequentist OLS

Focused comparison between the Horseshoe prior and the frequentist baseline, showing aggressive shrinkage of weak predictors while preserving strong signals.



Bayesian Horseshoe credible intervals compared with frequentist confidence intervals.

Conclusion