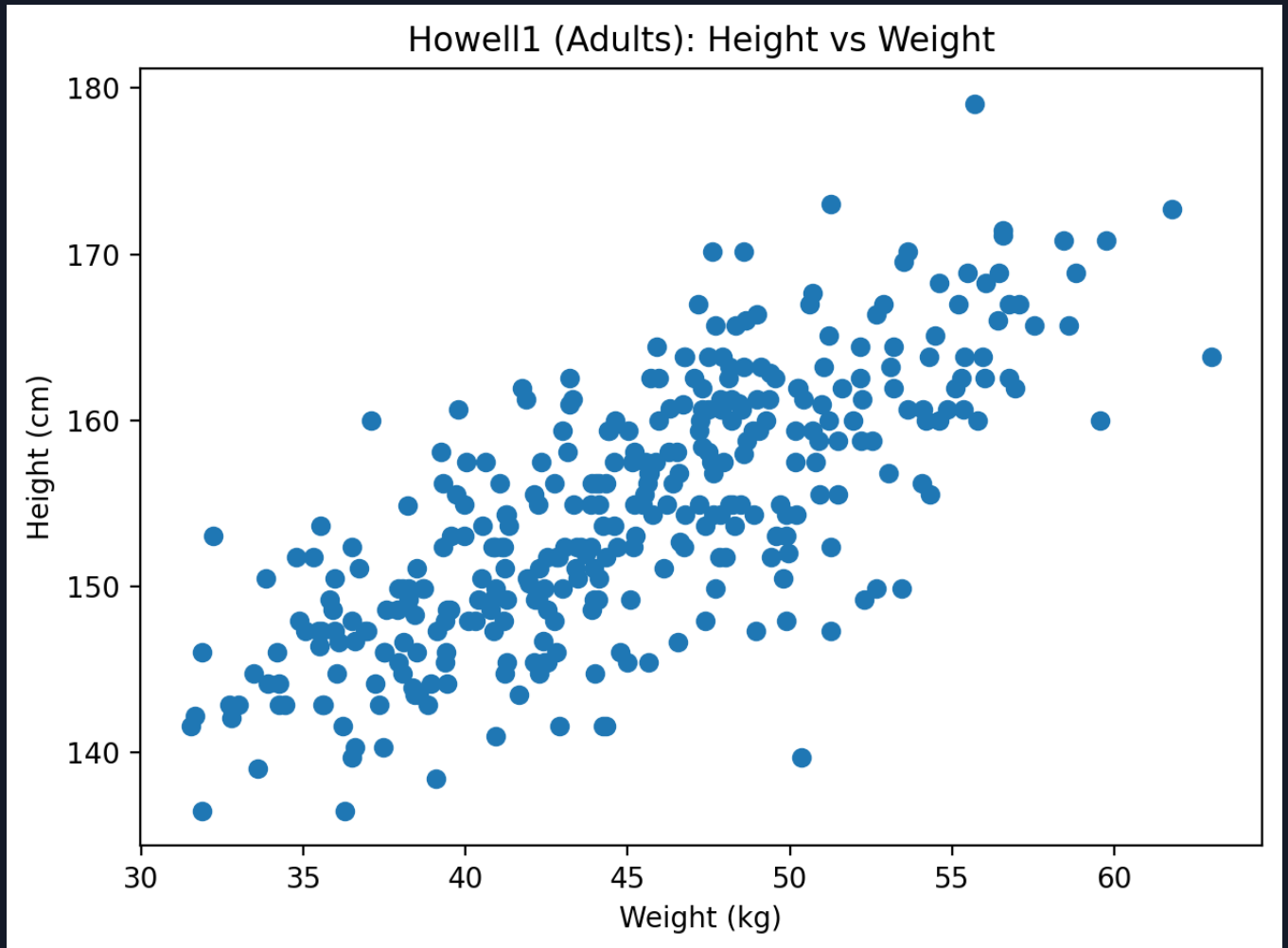


Howell1 (Adults) — Plots + Code (Aligned)

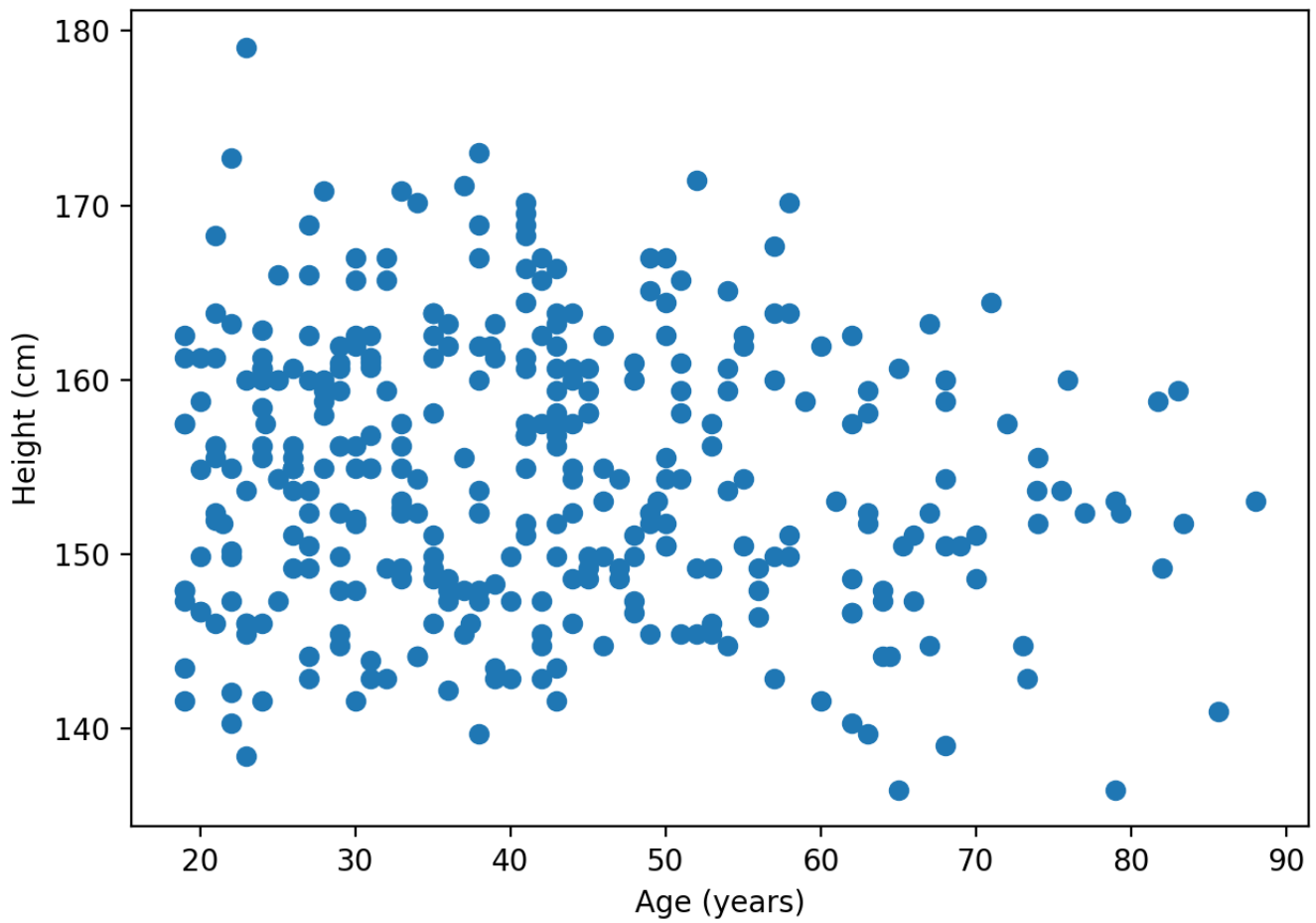
Each figure is followed by the exact code section that generates it. Full script included at the end.

EDA — Height vs Weight (Adults) + EDA — Height vs Age (Adults)

EDA scatter plots.



Howell1 (Adults): Height vs Age



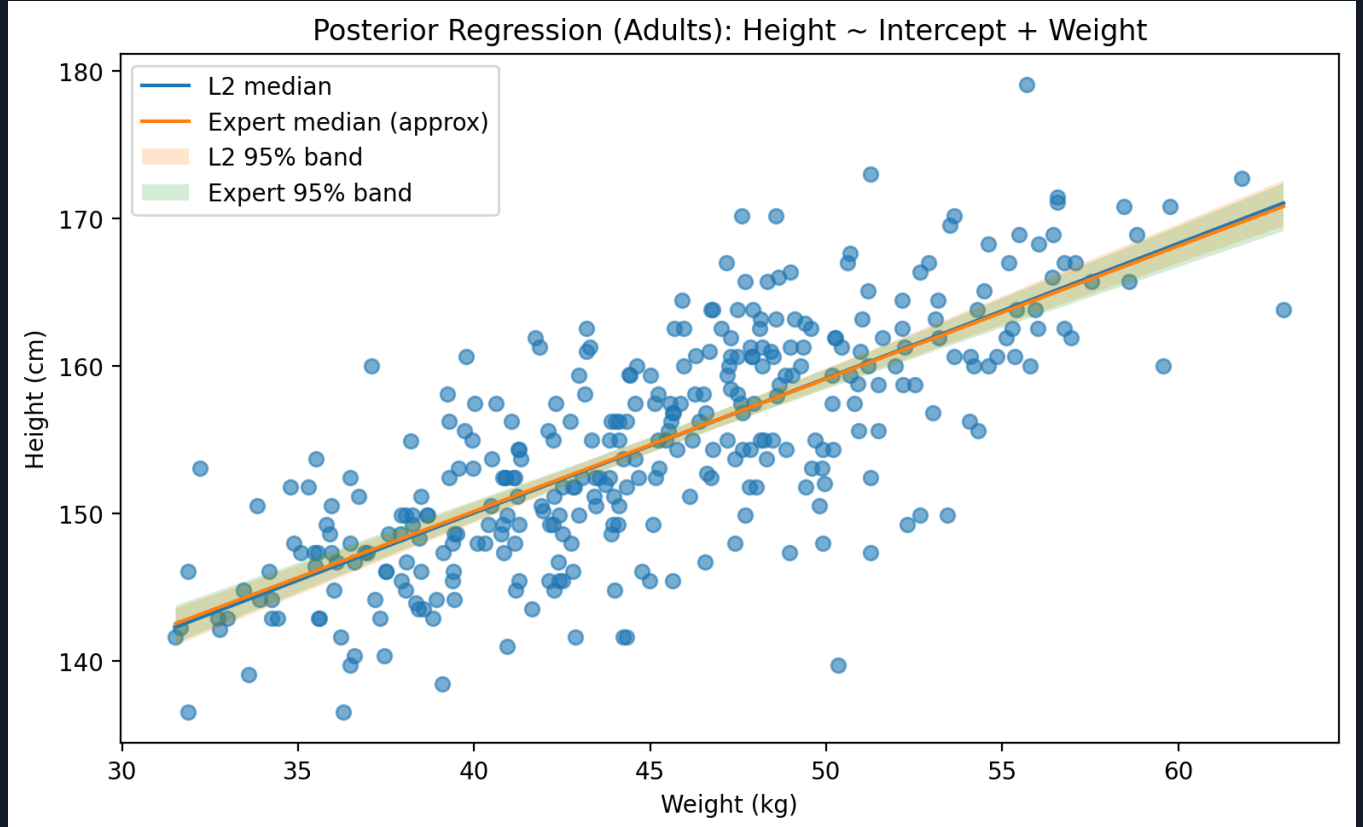
```
# -----
# 2) Part (a) - Exploratory Data Analysis
# -----
p1 <- ggplot(df, aes(x = weight, y = height)) +
  geom_point(size = 2) +
  labs(title = "Height vs Weight (Adults)",
       x = "Weight (kg)",
       y = "Height (cm)") +
  theme_minimal()

p2 <- ggplot(df, aes(x = age, y = height)) +
  geom_point(size = 2) +
  labs(title = "Height vs Age (Adults)",
       x = "Age (years)",
       y = "Height (cm)") +
  theme_minimal()

print(p1)
print(p2)
```

Posterior Regression (Adults): Median + 95% Bands (L2 vs Expert)

This plot uses posterior draws from the fitted models (L2 + Expert) to build median lines and 95% bands.



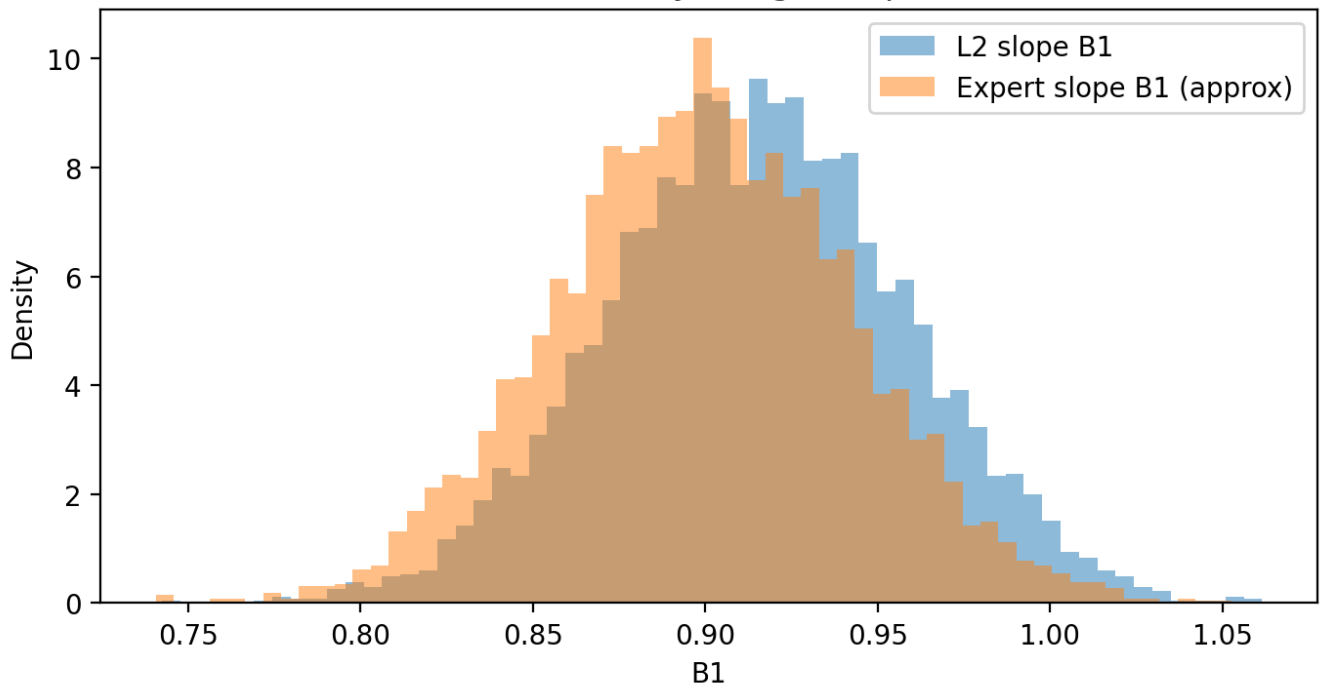
```
# L2 model fit (posterior draws used for regression line/bands and slope density)
fit_l2 <- sampling(
  object = model_l2,
  data   = data_l2,
  iter   = ITER,
  warmup = WARMUP,
  chains = CHAINS,
  refresh = 0
)

draws_l2 <- as_tibble(rstan::extract(fit_l2))
```

Posterior Density: Weight Slope (B1) (L2 vs Expert)

Histogram/density of B1 (weight slope) based on posterior draws.

Posterior Density: Weight Slope (B1)



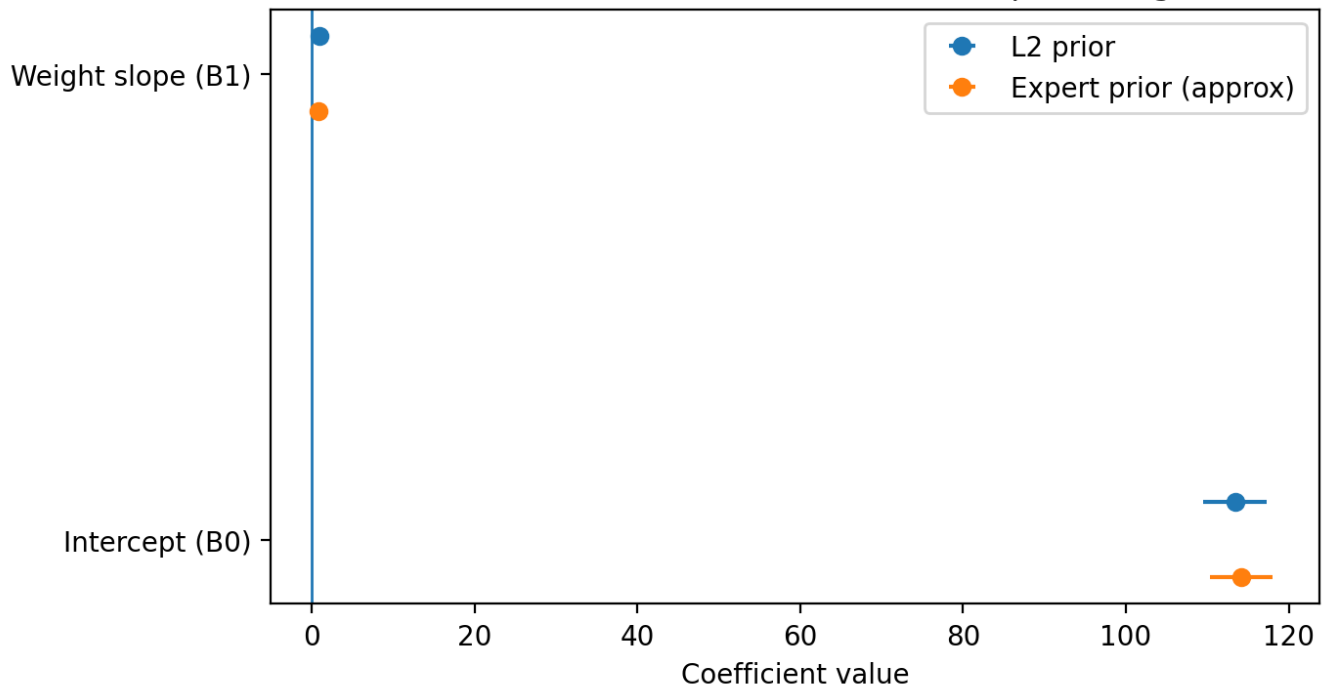
```
# L2 model fit (posterior draws used for regression line/bands and slope density)
fit_l2 <- sampling(
  object = model_l2,
  data   = data_l2,
  iter   = ITER,
  warmup = WARMUP,
  chains = CHAINS,
  refresh = 0
)

draws_l2 <- as_tibble(rstan::extract(fit_l2))
```

Posterior 95% Intervals (Adults, Intercept + Weight)

Credible intervals via `bayesplot::mcmc_intervals()`.

Posterior 95% Intervals (Adults, Intercept + Weight)

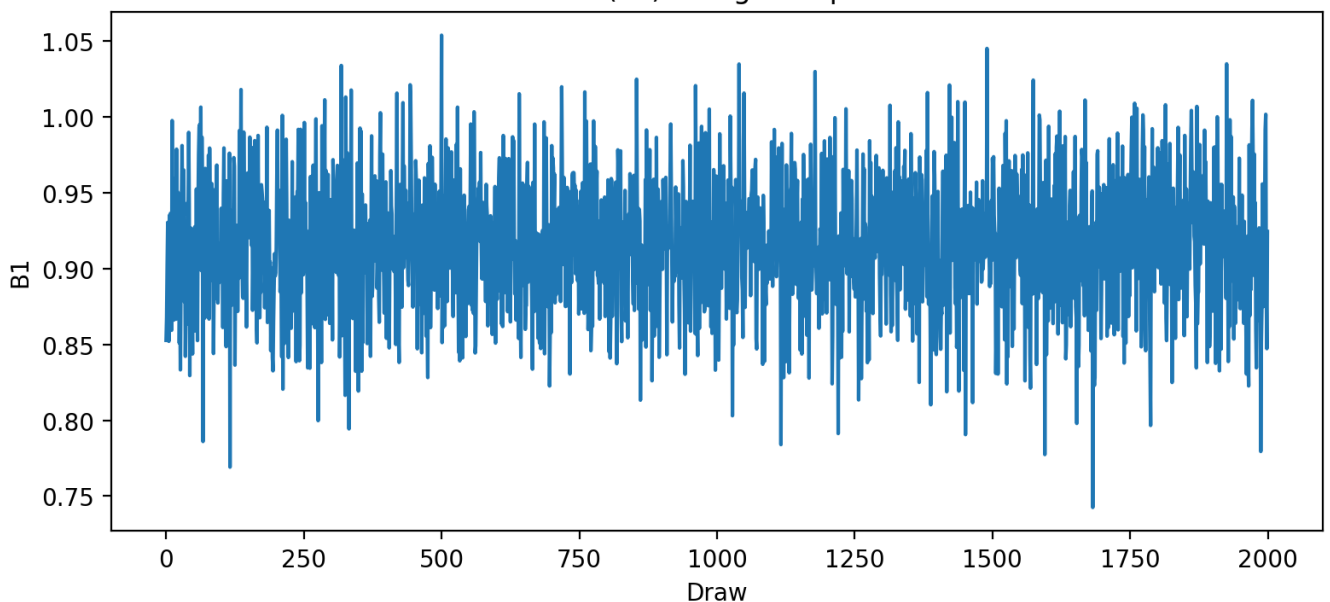


```
draws_l2 <- as_tibble(rstan::extract(fit_l2))
print(mcmc_intervals(draws_l2[c("sigma2", "B")]) + ggtitle("Credible Intervals (L2)"))
```

Trace (L2): Weight slope B1

Trace plot diagnostics from `bayesplot::mcmc_trace()`.

Trace (L2): Weight slope B1



```
print(mcmc_trace(as.array(fit_l2), pars = c("sigma2", "B[1]", "B[2]")) + ggtitle("Trace Plot (L2)"))
```

```
# =====
# Bayesian Regression with Shrinkage (L2 / L1) + Expert Prior
# Dataset: Howell1 (rethinking package)
# Output: EDA plots + posterior intervals + diagnostics + BF + BIC
# =====
# -----
```

```

# 0) Packages (install if needed)
# -----
pkgs <- c("rstan", "ggplot2", "dplyr", "tidyr", "tibble",
          "bayesplot", "coda", "bridgesampling", "rethinking")

to_install <- pkgs[!pkgs %in% installed.packages()[, "Package"]]
if (length(to_install) > 0) install.packages(to_install)

library(rstan)
library(ggplot2)
library(dplyr)
library(tidyr)
library(tibble)
library(bayesplot)
library(coda)
library(bridgesampling)
library(rethinking)

# Stan settings
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)

set.seed(2542)

# -----
# 1) Load dataset (Howell1)
# -----
data(Howell1, package = "rethinking")

df <- Howell1 %>%
  select(height, weight, age) %>%
  filter(age > 18) %>%
  as.data.frame()

print(head(df))
cat("\nRows:", nrow(df), "\n")
print(cor(df))

# -----
# 2) Part (a) - Exploratory Data Analysis
# -----
p1 <- ggplot(df, aes(x = weight, y = height)) +
  geom_point(size = 2) +
  labs(title = "Height vs Weight (Adults)",
       x = "Weight (kg)",
       y = "Height (cm)") +
  theme_minimal()

p2 <- ggplot(df, aes(x = age, y = height)) +
  geom_point(size = 2) +
  labs(title = "Height vs Age (Adults)",
       x = "Age (years)",
       y = "Height (cm)") +
  theme_minimal()

print(p1)
print(p2)

# -----
# Helper: safer sampling defaults (fast enough to run)
# If you want "heavy" MCMC like your original, increase iter/warmup.
# -----
ITER <- 6000
WARMUP <- 2000
CHAINS <- 3

# =====
# 3) Part (b) - L2 (Ridge-like) prior:  $B \sim \text{MVN}(m_0, L_0)$ 
# Model:  $y \sim \text{Normal}(XB, \sqrt{\text{sigma2}})$ ,  $\text{sigma2} \sim \text{Inv-Gamma}(a_0, b_0)$ 
# =====

stan_l2_code <- "

```

```

data {
  int<lower=1> N;
  int<lower=1> K;
  matrix[N,K] X;
  vector[N] y;
  vector[K] m0;
  matrix[K,K] L0;
  real<lower=0> a0;
  real<lower=0> b0;
}
parameters {
  real<lower=0> sigma2;
  vector[K] B;
}
model{
  B ~ multi_normal(m0, L0);
  sigma2 ~ inv_gamma(a0, b0);
  y ~ normal(X * B, sqrt(sigma2));
}
"

model_l2 <- stan_model(model_code = stan_l2_code)

# Design matrix: Intercept + Weight
X_l2 <- cbind(1, df$weight)
N <- nrow(df)
K_l2 <- 2

data_l2 <- list(
  N = N,
  K = K_l2,
  X = X_l2,
  y = df$height,
  m0 = rep(0, K_l2),
  L0 = diag(c(50, 10)),
  a0 = 2,
  b0 = 0.5
)

fit_l2 <- sampling(
  object = model_l2,
  data = data_l2,
  iter = ITER,
  warmup = WARMUP,
  chains = CHAINS,
  refresh = 0
)

cat("\n--- L2 MODEL SUMMARY ---\n")
print(summary(fit_l2)$summary)

draws_l2 <- as_tibble(rstan::extract(fit_l2))
print(mcmc_intervals(draws_l2[c("sigma2", "B")]) + ggtitle("Credible Intervals (L2)"))

# Diagnostics
cat("\n--- Heidelberger-Welch diagnostics (L2) ---\n")
print(heidel.diag(draws_l2))

# Optional extra plots
print(mcmc_trace(as.array(fit_l2), pars = c("sigma2", "B[1]", "B[2]")) + ggtitle("Trace Plot (L2)"))

# =====
# 4) Part (b) - L1 (Lasso-like) prior: B ~ Double Exponential(0, L0)
# Start with Intercept + Weight + Age, then refit without Age
# =====

stan_l1_code <- "
data {
  int<lower=1> N;
  int<lower=1> K;
  matrix[N,K] X;
  vector[N] y;

```

```

    real<lower=0> L0;
    real<lower=0> a0;
    real<lower=0> b0;
  }
  parameters {
    real<lower=0> sigma2;
    vector[K] B;
  }
  model{
    for (k in 1:K) {
      B[k] ~ double_exponential(0, L0);
    }
    sigma2 ~ inv_gamma(a0, b0);
    y ~ normal(X * B, sqrt(sigma2));
  }
}
"

model_l1 <- stan_model(model_code = stan_l1_code)

# (i) Intercept + Weight + Age
X_l1_full <- cbind(1, df$weight, df$age)
K_l1_full <- 3

data_l1_full <- list(
  N = N,
  K = K_l1_full,
  X = X_l1_full,
  y = df$height,
  L0 = 10,
  a0 = 2,
  b0 = 0.5
)

fit_l1_full <- sampling(
  object = model_l1,
  data = data_l1_full,
  iter = ITER,
  warmup = WARMUP,
  chains = CHAINS,
  refresh = 0
)

cat("\n--- L1 (FULL) MODEL SUMMARY ---\n")
print(summary(fit_l1_full)$summary)

draws_l1_full <- as_tibble(rstan::extract(fit_l1_full))
print(mcmc_intervals(draws_l1_full[c("sigma2", "B")]) + ggtitle("Credible Intervals (L1 - Full)"))
print(mcmc_trace(as.array(fit_l1_full), pars = c("sigma2", "B[1]", "B[2]", "B[3]")) + ggtitle("Trace Plots (L1 - Full)"))

# (ii) Refit Intercept + Weight only (as in your original logic)
X_l1 <- cbind(1, df$weight)
K_l1 <- 2

data_l1 <- list(
  N = N,
  K = K_l1,
  X = X_l1,
  y = df$height,
  L0 = 10,
  a0 = 2,
  b0 = 0.5
)

fit_l1 <- sampling(
  object = model_l1,
  data = data_l1,
  iter = ITER,
  warmup = WARMUP,
  chains = CHAINS,
  refresh = 0
)

cat("\n--- L1 (WEIGHT ONLY) MODEL SUMMARY ---\n")

```

```

print(summary(fit_l1)$summary)

draws_l1 <- as_tibble(rstan::extract(fit_l1))
print(mcmc_intervals(draws_l1[c("sigma2", "B")]) + ggtitle("Credible Intervals (L1 - Weight Only)"))
print(mcmc_trace(as.array(fit_l1), pars = c("sigma2", "B[1]", "B[2]")) + ggtitle("Trace Plot (L1 - Weight Only)"))

cat("\n--- Heidelberger-Welch diagnostics (L1) ---\n")
print(heidel.diag(draws_l1))

# =====
# 5) Part (c) - Expert prior model
# Priors:
#   B0 ~ Normal(168, 20)      (implemented as MVN with diag variances)
#   B1 ~ Normal(0.8, 1.5)
#   sigma2 ~ Uniform(0, 50)
# =====

stan_expert_code <- "
data {
  int<lower=1> N;
  int<lower=1> K;
  matrix[N,K] X;
  vector[N] y;
  vector[K] m0;
  matrix[K,K] L0;
}
parameters {
  real<lower=0> sigma2;
  vector[K] B;
}
model{
  B ~ multi_normal(m0, L0);
  sigma2 ~ uniform(0, 50);
  y ~ normal(X * B, sqrt(sigma2));
}
"

model_expert <- stan_model(model_code = stan_expert_code)

data_expert <- list(
  N = N,
  K = 2,
  X = X_l1,          # Intercept + Weight
  y = df$height,
  m0 = c(168, 0.8),
  L0 = diag(c(20, 1.5))
)

fit_expert <- sampling(
  object = model_expert,
  data   = data_expert,
  iter   = ITER,
  warmup = WARMUP,
  chains = CHAINS,
  refresh = 0
)

cat("\n--- EXPERT MODEL SUMMARY ---\n")
print(summary(fit_expert)$summary)

draws_expert <- as_tibble(rstan::extract(fit_expert))
print(mcmc_intervals(draws_expert[c("sigma2", "B")]) + ggtitle("Credible Intervals (Expert Prior)"))
print(mcmc_trace(as.array(fit_expert), pars = c("sigma2", "B[1]", "B[2]")) + ggtitle("Trace Plot (Expert Prior)"))

cat("\n--- Heidelberger-Welch diagnostics (Expert) ---\n")
print(heidel.diag(draws_expert))

# =====
# 6) Model comparison: Bayes Factor + BIC
# Compare: L1 (weight only) vs Expert model
# =====
cat("\n--- BRIDGE SAMPLING (this can take a bit) ---\n")

```

```

H0_bridge <- bridge_sampler(fit_l1, silent = TRUE)      # L1 weight-only
H1_bridge <- bridge_sampler(fit_expert, silent = TRUE)  # Expert

cat("\nLog marginal likelihoods:\n")
print(H0_bridge)
print(H1_bridge)

cat("\nApproximation error (%):\n")
cat("H0:", error_measures(H0_bridge)$percentage, "\n")
cat("H1:", error_measures(H1_bridge)$percentage, "\n")

BF10 <- bf(H1_bridge, H0_bridge) # Expert vs L1
BF01 <- bf(H0_bridge, H1_bridge) # L1 vs Expert

cat("\nBayes Factor BF10 (Expert / L1):\n")
print(BF10)

cat("\nBayes Factor BF01 (L1 / Expert):\n")
print(BF01)

post <- post_prob(H0_bridge, H1_bridge)
cat("\nPosterior model probabilities (given equal prior model probs):\n")
print(post)

# BIC (same structure as your original, using 3*log(N))
# Note: This is a rough comparison here; for full rigor, define k properly per model.
bic_l1      <- 3 * log(N) - 2 * H0_bridge$logml
bic_expert  <- 3 * log(N) - 2 * H1_bridge$logml

cat("\nBIC:\n")
cat("BIC (L1 weight-only):", bic_l1, "\n")
cat("BIC (Expert):", bic_expert, "\n")

cat("\nDONE.\n")

```