# Problem Set 2

Nate Gonzales-Hess

2025-04-14

```r
# Load Data
data("Howell1", package = "rethinking")
data <- subset(Howell1, age <= 13)
```

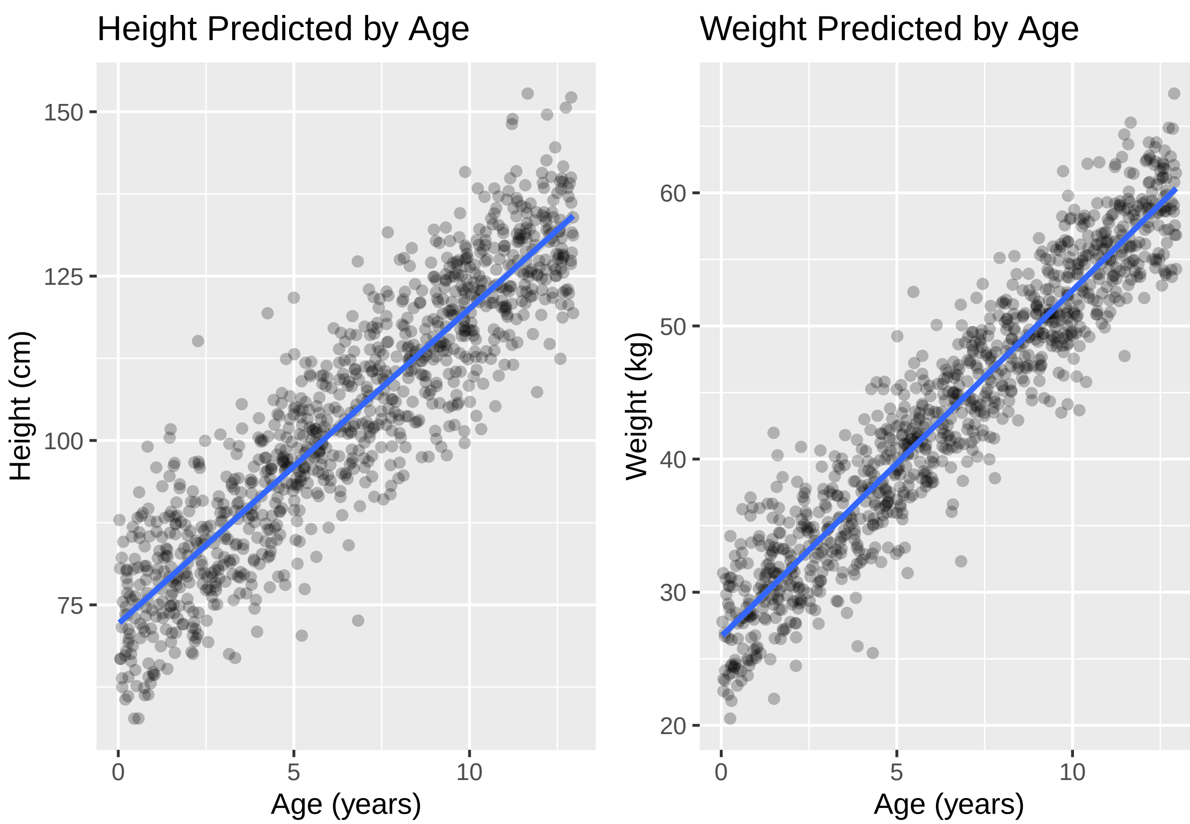
**Simulate data based on dataset and knowledge of the world**

```r
# Fit models to extract parameters
weight_age_model <- lm(weight ~ age, data = data)
height_age_model <- lm(height ~ age, data = data)
weight_height_model <- lm(weight ~ height, data = data)

# Extract coefficients
weight_rate <- coef(weight_age_model)["age"]  # Weight increase per year
weight_intercept <- coef(weight_age_model)["(Intercept)"]  # Weight at age 0
height_rate <- coef(height_age_model)["age"]  # Height increase per year
height_intercept <- coef(height_age_model)["(Intercept)"]  # Height at age 0
weight_by_height <- coef(weight_height_model)["height"]  # Weight per cm

# Extract standard deviations for noise
weight_std <- sigma(weight_age_model)  # Residual SD for weight
height_std <- sigma(height_age_model)  # Residual SD for height

# Generative simulation for children under 13
sim_child_data <- function(n = 100,
                           seed = 213,
                           height_intercept,
                           height_std,
                           height_rate,
                           weight_intercept,
                           weight_std,
                           weight_rate,
                           weight_by_height) {
  set.seed(seed)

  # Generate ages (uniform distribution of children under 13)
  age <- runif(n, 0, 13)

  # Generate height based on age
  height <- height_intercept + height_rate * age + rnorm(n, 0, height_std)
```

```r
  # Generate weight based on both height and age
  weight <- weight_intercept + weight_by_height * height +
            weight_rate * age + rnorm(n, 0, weight_std)

  # Return dataframe
  data.frame(age, height, weight)
}

# Create sample data with the extracted parameters
child_data <- sim_child_data(
  n = 1000,
  height_intercept = height_intercept,
  height_std = height_std,
  height_rate = height_rate,
  weight_intercept = weight_intercept,
  weight_std = weight_std,
  weight_rate = weight_rate,
  weight_by_height = weight_by_height
)
```

### Plot synthetic data

```r
# Plot relationships
p1 <- ggplot(child_data, aes(x = age, y = height)) +
  geom_point(alpha=0.25) +
  geom_smooth(method = "lm") +
  labs(title = "Height Predicted by Age", x = "Age (years)", y = "Height (cm)")

p2 <- ggplot(child_data, aes(x = age, y = weight)) +
  geom_point(alpha=0.25) +
  geom_smooth(method = "lm") +
  labs(title = "Weight Predicted by Age", x = "Age (years)", y = "Weight (kg)")

p1 + p2
```




**Mathematical Model of weight as a function of age**

```
# LaTeX refused to knit
# weight_i ~ normal(mu_i, sigma)
# mu_i = a + b * age_i

# Where:
# weight_i = individual weight observation
# age_i = individual age observation
# a = intercept (weight at age 0)
# b = slope (weight gain per year)
# sigma = residual standard deviation
```

**Extracting prior from simulated data**

```
# Extract parameters from the simulation
analyze_sim_params <- function(sim_data) {
  # Fit a simple linear model
  model <- lm(weight ~ age, data = sim_data)

  # Extract parameters
  intercept <- coef(model)[1]
```

```r
  slope <- coef(model)[2]
  sigma <- sigma(model)

  # Print results
  cat("From simulation:\n")
  cat("Intercept:", round(intercept, 2), "kg\n")
  cat("Slope:", round(slope, 2), "kg/year\n")
  cat("Residual SD:", round(sigma, 2), "kg\n\n")

  # Suggest priors
  cat("Suggested priors:\n")
  cat("prior(normal(", round(intercept, 1), ", ", max(1, round(intercept/5, 1)),
      "), class = \"Intercept\")\n", sep="")
  cat("prior(normal(", round(slope, 1), ", ", max(0.5, round(slope/4, 1)),
      "), class = \"b\")\n", sep="")
  cat("prior(exponential(", round(1/sigma, 2), "), class = \"sigma\")\n", sep="")

  # Return values
  return(list(intercept = intercept, slope = slope, sigma = sigma))
}

# Analyze simulated data
sim_params <- analyze_sim_params(child_data)
```

```
## From simulation:
## Intercept: 26.67 kg
## Slope: 2.6 kg/year
## Residual SD: 3.37 kg
##
## Suggested priors:
## prior(normal(26.7, 5.3), class = "Intercept")
## prior(normal(2.6, 0.6), class = "b")
## prior(exponential(0.3), class = "sigma")
```

```r
# Set priors for model
priors <- c(
  prior(normal(26.7, 5.3), class = "Intercept"),
  prior(normal(2.6, 0.6), class = "b"),
  prior(exponential(.3), class = "sigma")
)
```

```r
# Fit the model
weight_model <- brm(
  formula = weight ~ age,
  data = data,
  family = gaussian(),
  prior = priors,
  chains = 2,
  iter = 2000,
  warmup = 500,
  seed = 213,
)
```

```
## Running "C:/PROGRA~1/R/R-44~1.3/bin/x64/Rcmd.exe" SHLIB foo.c
## using C compiler: 'gcc.exe (GCC) 13.3.0'
## gcc  -I"C:/PROGRA~1/R/R-44~1.3/include" -DNDEBUG   -I"C:/Users/hssla/AppData/Local/R/win-library/4.4/
## cc1.exe: warning: command-line option '-std=c++14' is valid for C++/ObjC++ but not for C
## In file included from C:/Users/hssla/AppData/Local/R/win-library/4.4/RcppEigen/include/Eigen/Core:19
##                  from C:/Users/hssla/AppData/Local/R/win-library/4.4/RcppEigen/include/Eigen/Dense:1
##                  from C:/Users/hssla/AppData/Local/R/win-library/4.4/StanHeaders/include/stan/math/p
##                  from <command-line>:
## C:/Users/hssla/AppData/Local/R/win-library/4.4/RcppEigen/include/Eigen/src/Core/util/Macros.h:679:10
##   679 | #include <cmath>
##       |          ^~~~~~~
## compilation terminated.
## make: *** [C:/PROGRA~1/R/R-44~1.3/etc/x64/Makeconf:289: foo.o] Error 1
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000104 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.04 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:  501 / 2000 [ 25%]  (Sampling)
## Chain 1: Iteration:  700 / 2000 [ 35%]  (Sampling)
## Chain 1: Iteration:  900 / 2000 [ 45%]  (Sampling)
## Chain 1: Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1: Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1: Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1: Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1: Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.028 seconds (Warm-up)
## Chain 1:                0.038 seconds (Sampling)
## Chain 1:                0.066 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 7e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07 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:  501 / 2000 [ 25%]  (Sampling)
## Chain 2: Iteration:  700 / 2000 [ 35%]  (Sampling)
## Chain 2: Iteration:  900 / 2000 [ 45%]  (Sampling)
## Chain 2: Iteration: 1100 / 2000 [ 55%]  (Sampling)
```

```
## Chain 2: Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2: Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2: Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2: Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.024 seconds (Warm-up)
## Chain 2:                0.052 seconds (Sampling)
## Chain 2:                0.076 seconds (Total)
## Chain 2:
```

**Plot Results of prior and posterior linear fits**

```r
# Grabbing prior lines
n_lines <- 50
alpha_samples <- rnorm(n_lines, 5.2, 1)  # Using the prior mean and SD
beta_samples <- rnorm(n_lines, 2.2, 0.6) # Using the prior mean and SD

# Create a grid of ages
age_grid <- seq(0, 13, length.out = 100)

# Create plot data for prior lines
prior_lines_data <- data.frame()
for (i in 1:n_lines) {
  line_data <- data.frame(
    age = age_grid,
    weight = alpha_samples[i] + beta_samples[i] * age_grid,
    line_id = i
  )
  prior_lines_data <- rbind(prior_lines_data, line_data)
}

# Plot prior lines WITH data points
prior_plot <- ggplot() +
  geom_line(data = prior_lines_data, aes(x = age, y = weight, group = line_id), alpha = 0.2) +
  geom_point(data = data, aes(x = age, y = weight), alpha = 0.1) +  # Added data points
  labs(title = "Prior Regression Lines",
       x = "Age (years)", y = "Weight (kg)") +
  ylim(0, 50)

# For posterior lines - extract posterior samples directly
posterior_samples <- as.data.frame(weight_model)
n_posterior <- min(n_lines, nrow(posterior_samples))
sample_indices <- sample(1:nrow(posterior_samples), n_posterior)

# Create plot data for posterior lines
posterior_lines_data <- data.frame()
for (i in 1:n_posterior) {
  idx <- sample_indices[i]
  intercept <- posterior_samples$b_Intercept[idx]
  slope <- posterior_samples$b_age[idx]
```

```
  line_data <- data.frame(
    age = age_grid,
    weight = intercept + slope * age_grid,
    line_id = i
  )
  posterior_lines_data <- rbind(posterior_lines_data, line_data)
}

# Plot posterior lines with data points
posterior_plot <- ggplot() +
  geom_line(data = posterior_lines_data, aes(x = age, y = weight, group = line_id), alpha = 0.2) +
  geom_point(data = data, aes(x = age, y = weight), alpha = 0.1) +
  labs(title = "Posterior Regression Lines with Data",
       x = "Age (years)", y = "Weight (kg)") +
  ylim(0, 50)

# Display plots side by side
prior_plot + posterior_plot
```