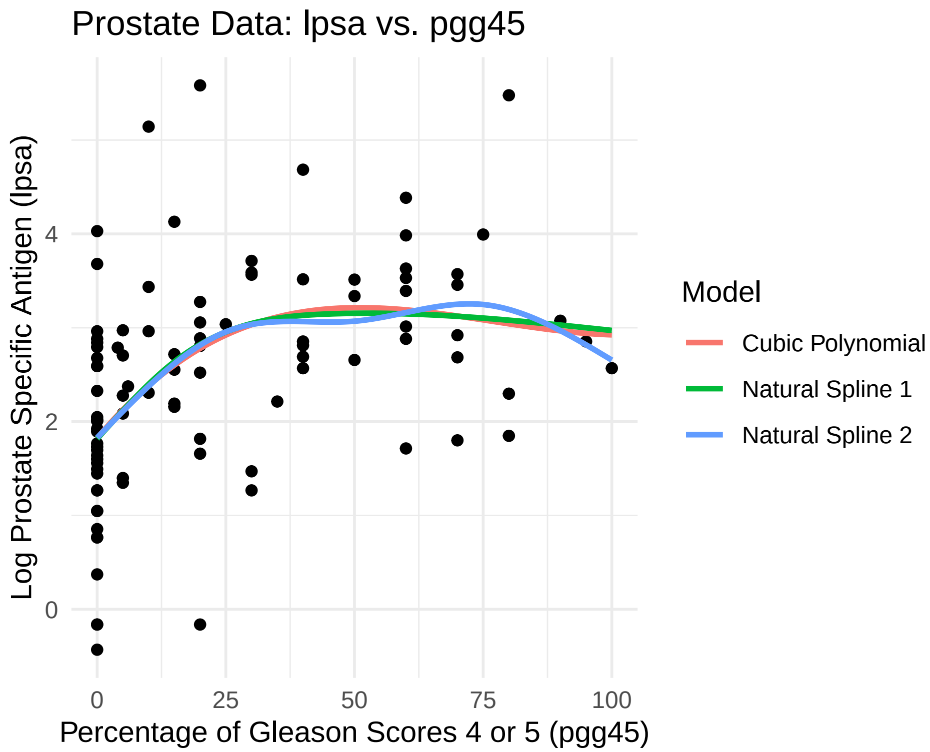
STAA 577: HW6

Your Name

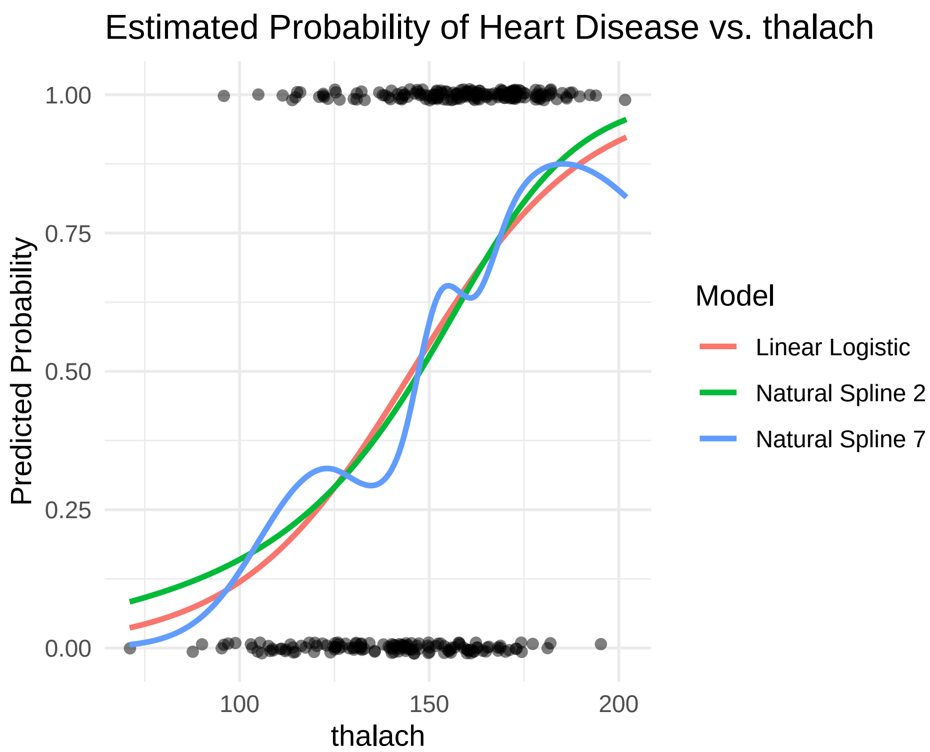
Problem 1

Regularization refers to adding a penalty term to the loss function that shrinks the estimated coefficients toward zero. This is used to prevent overfitting by discouraging overly complex models as such improves the model's generalization.

Problem 2



Problem 3



Problem 4

```
library(mgcv)
library(splines)

loocv_accuracy <- function(formula, data) {
    n <- nrow(data)
    pred_cat <- rep(NA, n)
    for (i in 1:n) {
        fit <- gam(formula, data = data[-i, ], family = binomial)
        phat <- predict(fit, newdata = data[i, ], type = "response")
        pred_cat[i] <- ifelse(phat > 0.5, 1, 0)
    }
    return(mean(pred_cat == data$target))
}

formula1 <- target ~ s(age) + sex + cp + s(trestbps) + s(chol) + fbs + restecg +
    s(thalach) + exang + s(oldpeak) + slope + ca + thal
acc1 <- loocv_accuracy(formula1, heart)

formula2 <- target ~ ns(age, df = 4) + sex + cp + ns(trestbps, df = 4) + ns(chol,
    df = 4) + fbs + restecg + ns(thalach, df = 4) + exang + ns(oldpeak, df = 4) +
    slope + ca + thal
acc2 <- loocv_accuracy(formula2, heart)
```

```r
formula3 <- target ~ s(age) + sex + cp + s(trestbps) + s(chol) + s(thalach) + exang +
    s(oldpeak)
acc3 <- loocv_accuracy(formula3, heart)

cat("LOOCV accuracy for Model 1 (all predictors with s()):", acc1, "\n")
```

```
## LOOCV accuracy for Model 1 (all predictors with s()): 0.8052805
```

```r
cat("LOOCV accuracy for Model 2 (all predictors with ns()):", acc2, "\n")
```

```
## LOOCV accuracy for Model 2 (all predictors with ns()): 0.8019802
```

```r
cat("LOOCV accuracy for Model 3 (reduced model):", acc3, "\n")
```

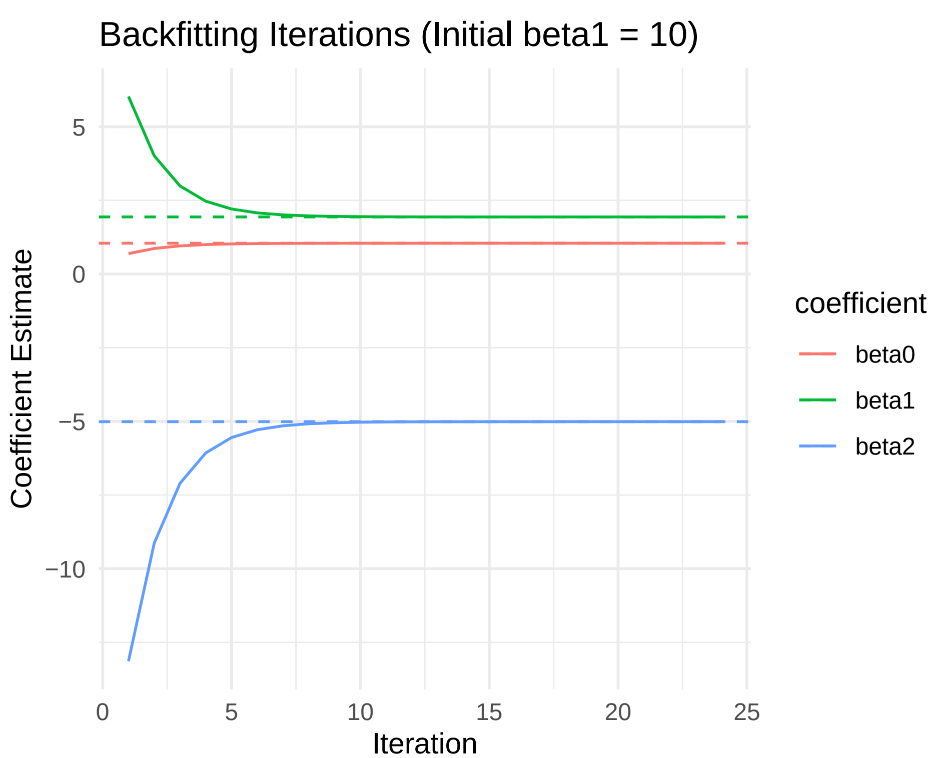
```
## LOOCV accuracy for Model 3 (reduced model): 0.8052805
```

Problem 5

```
## (Intercept)          X1          X3
##    1.045641    1.937950   -5.011689
```

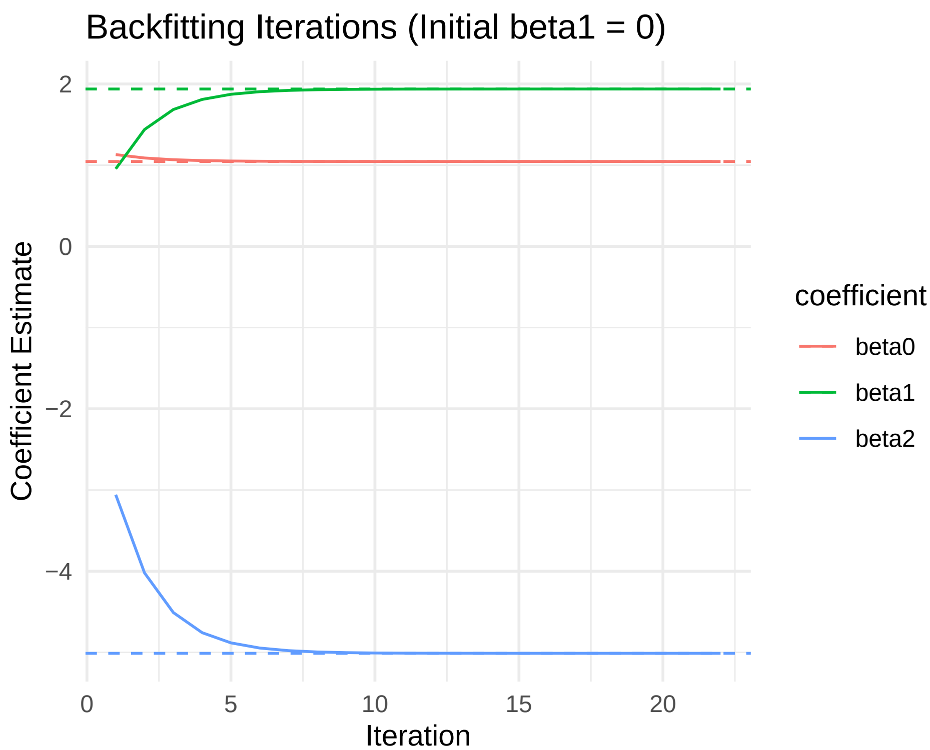
Problem 5(v.)

```
## Convergence reached at iteration 24 for initial beta1 = 10
```



Problem 5(vi.)

```
## Convergence reached at iteration 22 for initial beta1 = 0
```



Problem 5(vii.)

```
## For initial beta1 = 10, convergence was reached at iteration: 24

## For initial beta1 = 0, convergence was reached at iteration: 22
```

For initial beta1 = 10, convergence was reached at iteration: 24 For initial beta1 = 0, convergence was reached at iteration: 22

Problem 5(viii.)

when beta1 was initialized at 0 instead of 10 the early part of the path varied significantly. 0 got there a bit quicker and 10 took a lot longer. That said final convergence did happen at a similar time its jsut that the early change was different

Problem 6

Type down (tex) answers for each part or upload the picture of the handwritten answers.

Problem 6(a.)

Write

$$f_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3,$$

where

$$a_1 = \beta_0, \quad b_1 = \beta_1, \quad c_1 = \beta_2, \quad d_1 = \beta_3.$$

Problem 6(b.)

We expand $(x-\xi)^3$:

$$(x-\xi)^3 = x^3 - 3\xi x^2 + 3\xi^2 x - \xi^3.$$

Substituting this expansion into $f(x)$ gives:

$$\begin{aligned} f(x) &= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 \left(x^3 - 3\xi x^2 + 3\xi^2 x - \xi^3\right) \\ &= \left(\beta_0 - \beta_4 \xi^3\right) + \left(\beta_1 + 3\beta_4 \xi^2\right) x \\ &\quad + \left(\beta_2 - 3\beta_4 \xi\right) x^2 + \left(\beta_3 + \beta_4\right) x^3. \end{aligned}$$

Thus, define

$$f_2(x) = a_2 + b_2 x + c_2 x^2 + d_2 x^3,$$

with

$$a_2 = \beta_0 - \beta_4 \xi^3, \quad b_2 = \beta_1 + 3\beta_4 \xi^2, \quad c_2 = \beta_2 - 3\beta_4 \xi, \quad d_2 = \beta_3 + \beta_4.$$

Problem 6(c.)

For $x > \xi$:

$$\begin{aligned} f_2(\xi) &= (\beta_0 - \beta_4 \xi^3) + (\beta_1 + 3\beta_4 \xi^2)\xi \\ &\quad + (\beta_2 - 3\beta_4 \xi)\xi^2 + (\beta_3 + \beta_4)\xi^3 \\ &= \beta_0 + \beta_1 \xi + \beta_2 \xi^2 + \beta_3 \xi^3, \end{aligned}$$

since the β_4-terms cancel. Therefore,

$$f_1(\xi) = f_2(\xi).$$

Problem 6(d.)

Differentiate $f_1(x)$:

$$f_1'(x) = \beta_1 + 2\beta_2 x + 3\beta_3 x^2,$$

so that

$$f_1'(\xi) = \beta_1 + 2\beta_2 \xi + 3\beta_3 \xi^2.$$

Differentiate $f_2(x)$:

$$f_2'(x) = b_2 + 2c_2 x + 3d_2 x^2,$$

and thus,

$$\begin{aligned} f_2'(\xi) &= (\beta_1 + 3\beta_4\xi^2) + 2(\beta_2 - 3\beta_4\xi)\xi + 3(\beta_3 + \beta_4)\xi^2 \\ &= \beta_1 + 3\beta_4\xi^2 + 2\beta_2\xi - 6\beta_4\xi^2 + 3\beta_3\xi^2 + 3\beta_4\xi^2 \\ &= \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2. \end{aligned}$$

Thus, we have

$$f_1'(\xi) = f_2'(\xi).$$

Problem 6(e.)

Differentiate $f_1'(x)$:

$$f_1''(x) = 2\beta_2 + 6\beta_3 x, \quad \text{so} \quad f_1''(\xi) = 2\beta_2 + 6\beta_3\xi.$$

Differentiate $f_2'(x)$:

$$f_2''(x) = 2c_2 + 6d_2x, \quad \text{so} \quad f_2''(\xi) = 2(\beta_2 - 3\beta_4\xi) + 6(\beta_3 + \beta_4)\xi.$$

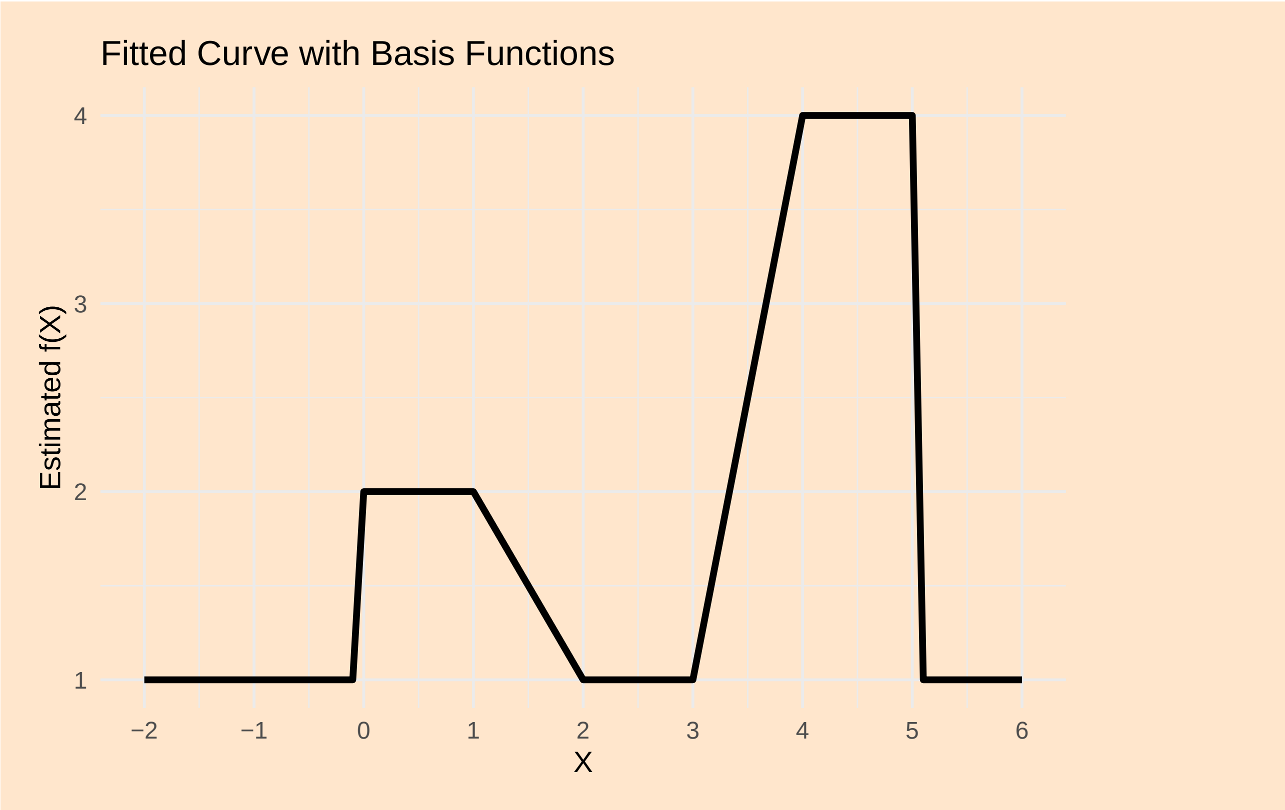
Simplifying,

$$\begin{aligned} f_2''(\xi) &= 2\beta_2 - 6\beta_4\xi + 6\beta_3\xi + 6\beta_4\xi \\ &= 2\beta_2 + 6\beta_3\xi. \end{aligned}$$

Hence,

$$f_1''(\xi) = f_2''(\xi).$$

Problem 7



Appendix

```r
library(knitr)
# install the tidyverse library (do this once) install.packages('tidyverse')
library(tidyverse)
library(splines)
library(ggplot2)
library(tidyr)
# set chunk and figure default options
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE, fig.width = 5.5,
    fig.height = 4, tidy = TRUE)
# problem 2

library(ggplot2)
library(splines)

prostate <- read.csv("prostate.csv")

model_poly <- lm(lpsa ~ poly(pgg45, 3), data = prostate)

model_ns1 <- lm(lpsa ~ ns(pgg45, knots = c(15, 45)), data = prostate)

model_ns2 <- lm(lpsa ~ ns(pgg45, knots = c(25, 50, 75)), data = prostate)

grid <- data.frame(pgg45 = seq(min(prostate$pgg45), max(prostate$pgg45), length.out = 200))
grid$pred_poly <- predict(model_poly, newdata = grid)
```

```r
grid$pred_ns1 <- predict(model_ns1, newdata = grid)
grid$pred_ns2 <- predict(model_ns2, newdata = grid)

ggplot(prostate, aes(x = pgg45, y = lpsa)) + geom_point() + geom_line(data = grid,
    aes(y = pred_poly, color = "Cubic Polynomial"), size = 1) + geom_line(data = grid,
    aes(y = pred_ns1, color = "Natural Spline 1"), size = 1) + geom_line(data = grid,
    aes(y = pred_ns2, color = "Natural Spline 2"), size = 1) + labs(title = "Prostate Data: lpsa vs. pgg
    x = "Percentage of Gleason Scores 4 or 5 (pgg45)", y = "Log Prostate Specific Antigen (lpsa)",
    color = "Model") + theme_minimal()


# Problem 3
library(ggplot2)
library(splines)
library(tidyr)

heart_train <- read.csv("heart_training.csv")
heart_test <- read.csv("heart_test.csv")

heart <- rbind(heart_train, heart_test)

heart$sex <- as.factor(heart$sex)
heart$cp <- as.factor(heart$cp)
heart$exang <- as.factor(heart$exang)
heart$fbs <- as.factor(heart$fbs)
heart$restecg <- as.factor(heart$restecg)

model_logit <- glm(target ~ thalach, data = heart, family = binomial)
model_ns2 <- glm(target ~ ns(thalach, df = 2), data = heart, family = binomial)
model_ns6 <- glm(target ~ ns(thalach, df = 6), data = heart, family = binomial)

thalach_grid <- data.frame(thalach = seq(min(heart$thalach), max(heart$thalach),
    length.out = 200))
thalach_grid$pred_logit <- predict(model_logit, newdata = thalach_grid, type = "response")
thalach_grid$pred_ns2 <- predict(model_ns2, newdata = thalach_grid, type = "response")
thalach_grid$pred_ns6 <- predict(model_ns6, newdata = thalach_grid, type = "response")

pred_data <- pivot_longer(thalach_grid, cols = starts_with("pred_"), names_to = "model",
    values_to = "pred")
pred_data$model <- factor(pred_data$model, levels = c("pred_logit", "pred_ns2", "pred_ns6"),
    labels = c("Linear Logistic", "Natural Spline 2", "Natural Spline 7"))

ggplot(heart, aes(x = thalach, y = target)) + geom_point(alpha = 0.5, position = position_jitter(height
    geom_line(data = pred_data, aes(x = thalach, y = pred, color = model), size = 1) +
    labs(title = "Estimated Probability of Heart Disease vs. thalach", x = "thalach",
        y = "Predicted Probability", color = "Model") + theme_minimal()

library(mgcv)
library(splines)

loocv_accuracy <- function(formula, data) {
    n <- nrow(data)
    pred_cat <- rep(NA, n)
```

```
    for (i in 1:n) {
        fit <- gam(formula, data = data[-i, ], family = binomial)
        phat <- predict(fit, newdata = data[i, ], type = "response")
        pred_cat[i] <- ifelse(phat > 0.5, 1, 0)
    }
    return(mean(pred_cat == data$target))
}

formula1 <- target ~ s(age) + sex + cp + s(trestbps) + s(chol) + fbs + restecg +
    s(thalach) + exang + s(oldpeak) + slope + ca + thal
acc1 <- loocv_accuracy(formula1, heart)

formula2 <- target ~ ns(age, df = 4) + sex + cp + ns(trestbps, df = 4) + ns(chol,
    df = 4) + fbs + restecg + ns(thalach, df = 4) + exang + ns(oldpeak, df = 4) +
    slope + ca + thal
acc2 <- loocv_accuracy(formula2, heart)

formula3 <- target ~ s(age) + sex + cp + s(trestbps) + s(chol) + s(thalach) + exang +
    s(oldpeak)
acc3 <- loocv_accuracy(formula3, heart)

cat("LOOCV accuracy for Model 1 (all predictors with s()):", acc1, "\n")
cat("LOOCV accuracy for Model 2 (all predictors with ns()):", acc2, "\n")
cat("LOOCV accuracy for Model 3 (reduced model):", acc3, "\n")


# Problem 5i-iv
set.seed(577)
n <- 150
X1 <- rnorm(n)
X2 <- rnorm(n)
X3 <- 0.5 * X1 + 0.5 * X2  # X3 is generated from X1 and X2
epsilon <- rnorm(n, mean = 0, sd = 0.5)
Y <- 1 + 2 * X1 - 5 * X3 + epsilon

lm_fit <- lm(Y ~ X1 + X3)
coef_true <- coef(lm_fit)
print(coef_true)

tol <- 1e-06
max_iter <- 100

beta1 <- 10
beta0_est <- numeric(max_iter)
beta1_est <- numeric(max_iter)
beta2_est <- numeric(max_iter)
conv_iter_10 <- NA

for (i in 1:max_iter) {
    fit_beta2 <- lm(I(Y - beta1 * X1) ~ X3)
    beta0 <- coef(fit_beta2)[1]
    beta2 <- coef(fit_beta2)[2]
```

```
    fit_beta1 <- lm(I(Y - beta2 * X3) ~ X1)
    beta0_new <- coef(fit_beta1)[1]
    beta1_new <- coef(fit_beta1)[2]

    beta0_est[i] <- beta0_new
    beta1_est[i] <- beta1_new
    beta2_est[i] <- beta2

    if (i > 1 && abs(beta1_new - beta1) < tol) {
        conv_iter_10 <- i
        cat("Convergence reached at iteration", i, "for initial beta1 = 10\n")
        break
    }
    beta1 <- beta1_new
}
if (is.na(conv_iter_10)) conv_iter_10 <- max_iter

iter_vals <- 1:(ifelse(is.na(conv_iter_10), max_iter, conv_iter_10))
backfit_df <- data.frame(iter = iter_vals, beta0 = beta0_est[iter_vals], beta1 = beta1_est[iter_vals],
    beta2 = beta2_est[iter_vals])
backfit_long <- pivot_longer(backfit_df, cols = c("beta0", "beta1", "beta2"), names_to = "coefficient",
    values_to = "estimate")

p1 <- ggplot(backfit_long, aes(x = iter, y = estimate, color = coefficient)) + geom_line() +
    labs(title = "Backfitting Iterations (Initial beta1 = 10)", x = "Iteration",
        y = "Coefficient Estimate") + theme_minimal() + geom_hline(aes(yintercept = coef_true["(Interce
    color = "beta0"), linetype = "dashed") + geom_hline(aes(yintercept = coef_true["X1"],
    color = "beta1"), linetype = "dashed") + geom_hline(aes(yintercept = coef_true["X3"],
    color = "beta2"), linetype = "dashed")
print(p1)
beta1 <- 0
beta0_est2 <- numeric(max_iter)
beta1_est2 <- numeric(max_iter)
beta2_est2 <- numeric(max_iter)
conv_iter_0 <- NA

for (i in 1:max_iter) {
    fit_beta2 <- lm(I(Y - beta1 * X1) ~ X3)
    beta0 <- coef(fit_beta2)[1]
    beta2 <- coef(fit_beta2)[2]

    fit_beta1 <- lm(I(Y - beta2 * X3) ~ X1)
    beta0_new <- coef(fit_beta1)[1]
    beta1_new <- coef(fit_beta1)[2]

    beta0_est2[i] <- beta0_new
    beta1_est2[i] <- beta1_new
    beta2_est2[i] <- beta2

    if (i > 1 && abs(beta1_new - beta1) < tol) {
        conv_iter_0 <- i
        cat("Convergence reached at iteration", i, "for initial beta1 = 0\n")
        break
```

```
    }
    beta1 <- beta1_new
}
if (is.na(conv_iter_0)) conv_iter_0 <- max_iter
iter_vals2 <- 1:(ifelse(is.na(conv_iter_0), max_iter, conv_iter_0))
backfit_df2 <- data.frame(iter = iter_vals2, beta0 = beta0_est2[iter_vals2], beta1 = beta1_est2[iter_va
    beta2 = beta2_est2[iter_vals2])
backfit_long2 <- pivot_longer(backfit_df2, cols = c("beta0", "beta1", "beta2"), names_to = "coefficient
    values_to = "estimate")

p2 <- ggplot(backfit_long2, aes(x = iter, y = estimate, color = coefficient)) + geom_line() +
    labs(title = "Backfitting Iterations (Initial beta1 = 0)", x = "Iteration", y = "Coefficient Estima
    theme_minimal() + geom_hline(aes(yintercept = coef_true["(Intercept)"], color = "beta0"),
    linetype = "dashed") + geom_hline(aes(yintercept = coef_true["X1"], color = "beta1"),
    linetype = "dashed") + geom_hline(aes(yintercept = coef_true["X3"], color = "beta2"),
    linetype = "dashed")
print(p2)

cat("For initial beta1 = 10, convergence was reached at iteration:", conv_iter_10,
    "\n")
cat("For initial beta1 = 0, convergence was reached at iteration:", conv_iter_0,
    "\n")



x_seq <- seq(-2, 6, by = 0.1)

b1 <- ifelse(x_seq >= 0 & x_seq <= 2, 1, 0) - ifelse(x_seq >= 1 & x_seq <= 2, (x_seq -
    1), 0)
b2 <- ifelse(x_seq >= 3 & x_seq <= 4, (x_seq - 3), 0) + ifelse(x_seq > 4 & x_seq <=
    5, 1, 0)


fhat <- 1 + 1 * b1 + 3 * b2

df <- data.frame(x = x_seq, fhat = fhat)

ggplot(df, aes(x = x, y = fhat)) + geom_line(size = 1.2) + labs(title = "Fitted Curve with Basis Functi
    x = "X", y = "Estimated f(X)") + theme_minimal() + scale_x_continuous(breaks = seq(-2,
    6, by = 1)) + scale_y_continuous(breaks = seq(0, 5, by = 1))
```