# P8106 Midterm

Naomi Simon-Kumar, Ila Kanneboyina, Shayne Estill ns3782

03/24/2025

## Contents



### Libraries

```
# Load libraries
library(tidyverse)
library(caret)
library(ggplot2)
library(patchwork)
library(corrplot)
library(mgcv)
library(tidymodels)
library(earth)
```

### Partition into training and testing set

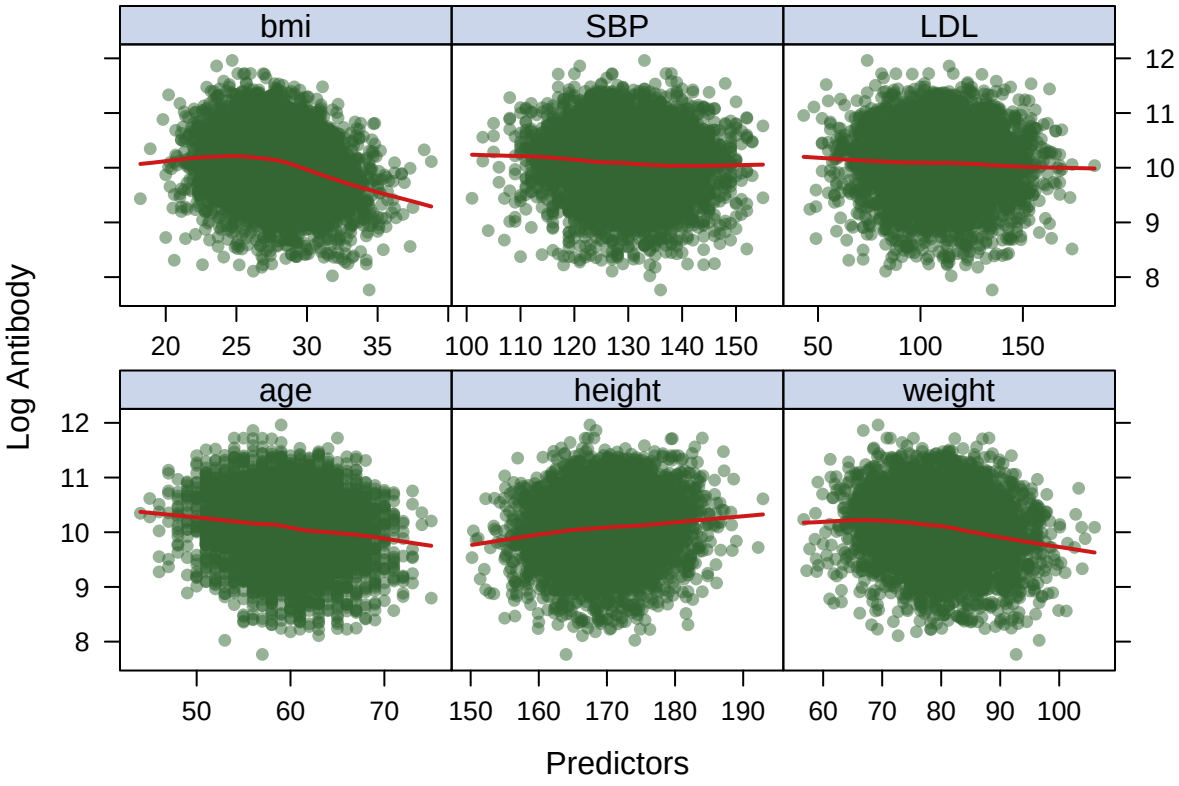
```
# Load training data
load("dat1.RData")
training_data <- dat1

# Load test data (dat2)
load("dat2.RData")
testing_data <- dat2

str(training_data)
```

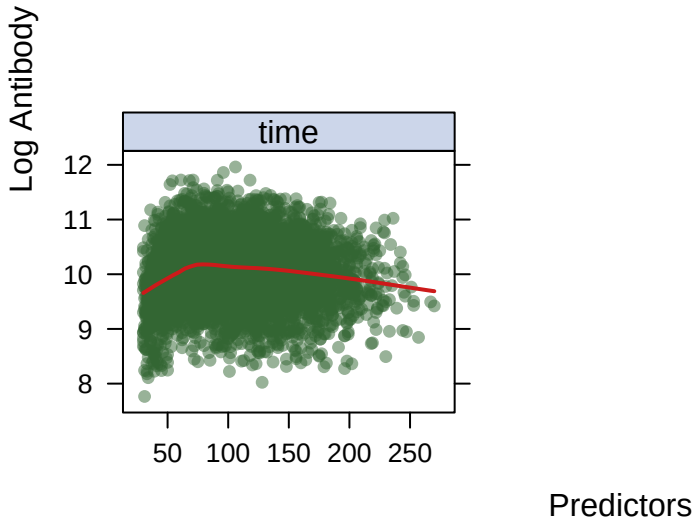
```
## 'data.frame':    5000 obs. of  14 variables:
##  $ id          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ age         : num  50 71 58 63 56 59 67 62 60 64 ...
##  $ gender      : int  0 1 1 0 1 1 0 1 0 1 ...
##  $ race        : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 3 4 1 4 1 ...
##  $ smoking     : Factor w/ 3 levels "0","1","2": 1 1 2 1 1 1 1 1 1 1 ...
##  $ height      : num  176 176 169 167 163 ...
##  $ weight      : num  68.3 69.6 76.9 90 83.9 86.8 91.4 87.7 85.7 76.6 ...
##  $ bmi         : num  22 22.6 27 32.1 31.7 30.8 29.7 28.1 29 31.5 ...
##  $ diabetes    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hypertension: num  0 1 0 1 0 1 1 0 0 1 ...
##  $ SBP         : num  130 149 127 138 123 132 133 130 129 134 ...
##  $ LDL         : num  82 129 101 93 97 108 89 96 120 135 ...
##  $ time        : num  76 82 168 105 193 143 63 78 61 88 ...
##  $ log_antibody: num  10.65 9.89 10.9 9.91 9.56 ...
```

```r
# Set seed for reproducibility
set.seed(299)

# Set 10-fold cross-validation
ctrl1 <- trainControl(method = "cv", number = 10)

# Remove non-predictor variables
training_data <- training_data %>% select(-id) # remove ID variable
testing_data <- testing_data %>% select(-id) # remove ID variable
```

## Exploratory Analysis

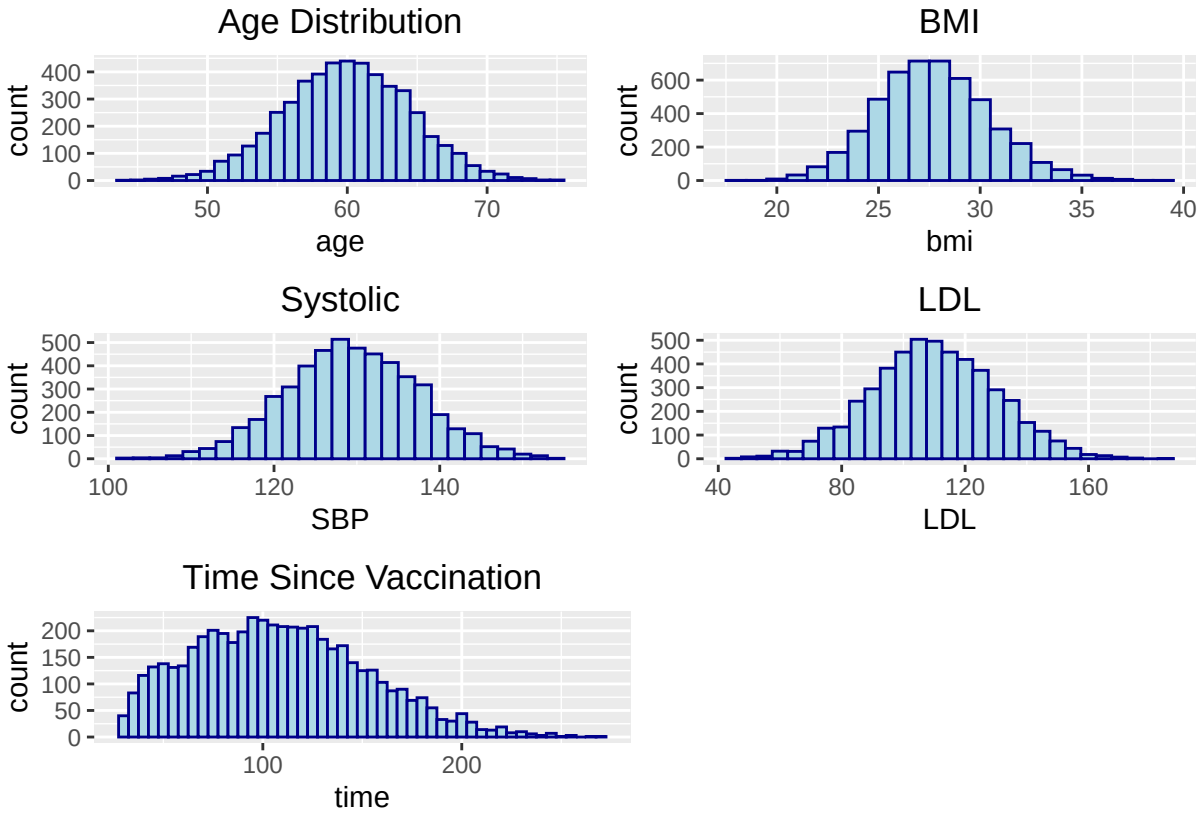
```r
## Set the plotting theme
theme1 <- trellis.par.get()
theme1$plot.symbol$col <- rgb(.2, .4, .2, .5)
theme1$plot.symbol$pch <- 16
theme1$plot.line$col <- rgb(.8, .1, .1, 1)
theme1$plot.line$lwd <- 2
theme1$strip.background$col <- rgb(.0, .2, .6, .2)
trellis.par.set(theme1)

## Feature Plots for numeric predictors (L2)
featurePlot(
  x = training_data[, c("age", "height", "weight", "bmi", "SBP", "LDL", "time")],
  y = training_data$log_antibody,
  plot = "scatter",
  span = 0.5,
  labels = c("Predictors", "Log Antibody"),
  type = c("p", "smooth"),
  layout = c(3, 2)
)
```


bmi
SBP
LDL
age
height
weight
Log Antibody
Predictors
12
11
10
9
8
20
25
30
35
100
110
120
130
140
150
50
100
150
60
70
80
90
160
170
180
190



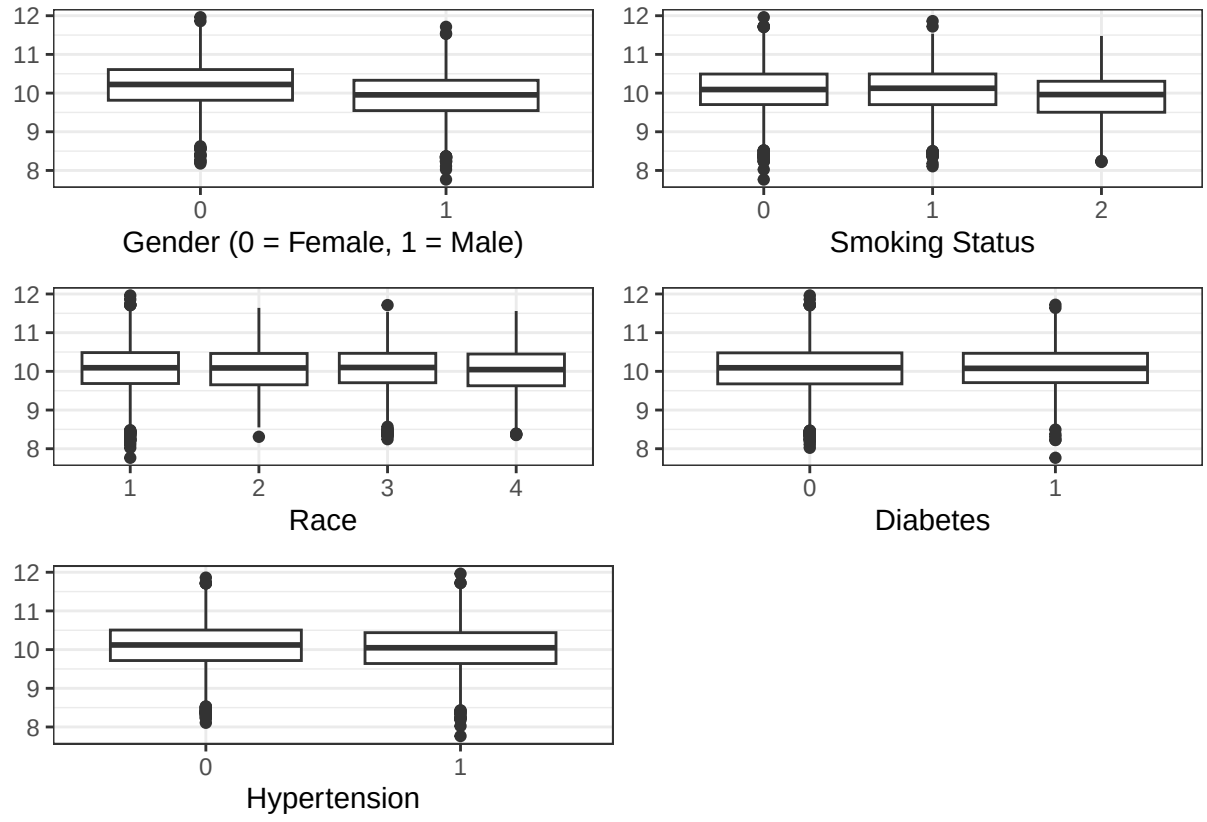

```
## Histograms for numeric predictors (need to ref code)

# Age histogram
h1 <- ggplot(training_data, aes(x = age)) +
  geom_histogram(binwidth = 1, color = "darkblue", fill = "lightblue") +
  ggtitle("Age Distribution") +
  theme(plot.title = element_text(hjust = 0.5))

# BMI histogram
h2 <- ggplot(training_data, aes(x = bmi)) +
  geom_histogram(binwidth = 1, color = "darkblue", fill = "lightblue") +
  ggtitle("BMI") +
  theme(plot.title = element_text(hjust = 0.5))

# SBP histogram
h3 <- ggplot(training_data, aes(x = SBP)) +
  geom_histogram(binwidth = 2, color = "darkblue", fill = "lightblue") +
  ggtitle("Systolic") +
  theme(plot.title = element_text(hjust = 0.5))

# LDL histogram
h4 <- ggplot(training_data, aes(x = LDL)) +
  geom_histogram(binwidth = 5, color = "darkblue", fill = "lightblue") +
  ggtitle("LDL") +
  theme(plot.title = element_text(hjust = 0.5))
```

```r
# Time since vaccination histogram
h5 <- ggplot(training_data, aes(x = time)) +
  geom_histogram(binwidth = 5, color = "darkblue", fill = "lightblue") +
  ggtitle("Time Since Vaccination") +
  theme(plot.title = element_text(hjust = 0.5))

# Combine using patchwork
(h1 + h2) / (h3 + h4) / (h5 + plot_spacer())
```

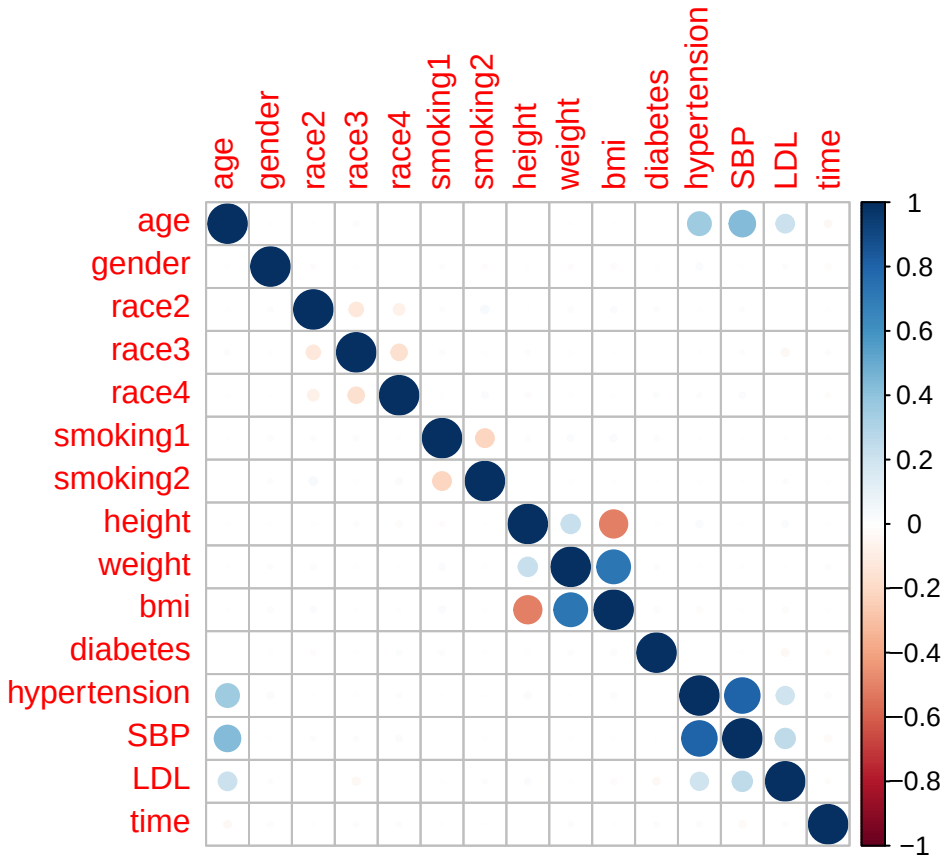



```r
## Boxplots for categorical predictors

p1 <- ggplot(training_data, aes(x = factor(gender), y = log_antibody)) +
  geom_boxplot() +
  labs(x = "Gender (0 = Female, 1 = Male)", y = NULL) +
  theme_bw()

p2 <- ggplot(training_data, aes(x = smoking, y = log_antibody)) +
  geom_boxplot() +
  labs(x = "Smoking Status", y = NULL) +
  theme_bw()

p3 <- ggplot(training_data, aes(x = race, y = log_antibody)) +
  geom_boxplot() +
  labs(x = "Race", y = NULL) +
  theme_bw()
```

```r
p4 <- ggplot(training_data, aes(x = factor(diabetes), y = log_antibody)) +
  geom_boxplot() +
  labs(x = "Diabetes", y = NULL) +
  theme_bw()

p5 <- ggplot(training_data, aes(x = factor(hypertension), y = log_antibody)) +
  geom_boxplot() +
  labs(x = "Hypertension", y = NULL) +
  theme_bw()

# Using patchwork
((p1 + p2) / (p3 + p4) / (p5 + plot_spacer()))
```




```r
## Correlation Plot

# Matrix of predictors
x <- model.matrix(log_antibody ~ ., training_data)[, -1]

# Vector of response
y <- training_data$log_antibody
corrplot(cor(x), method = 'circle', type = 'full')
```




Based on the exploratory analysis of continuous predictors, time since vaccination appears to be right-skewed.

### Model Selection

#Linear Regression - Ila

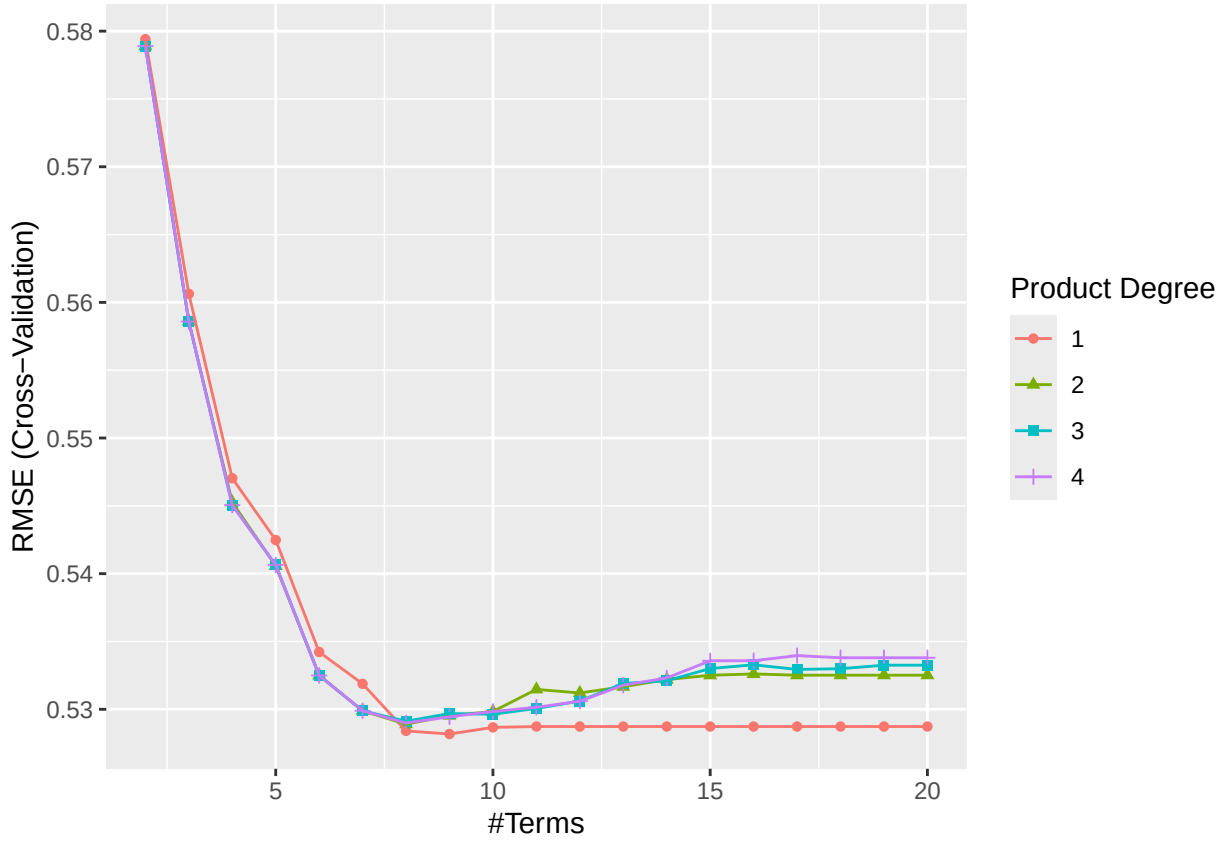
#Elastic Net - Shayne

#LDA - Shayne

## MARS - Naomi

```
# Set seed for reproducibility
set.seed(299)

# Specify MARS grid - first attempt
# mars_grid <- expand.grid(
#  degree = 1:3, # degree of interactions
#  nprune = 2:15  # no. of retained terms
# )

# MARS tuning grid - expanding grid
mars_grid <- expand.grid(
  degree = 1:4,      # interaction degrees
```

```
  nprune = 2:20      # number of terms
)

# Train MARS model to predict log_antibody
mars.fit <- train(log_antibody ~ .,
                  data = training_data,
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl1)

# Plot CV performance
ggplot(mars.fit)
```




```
# Optimal parameters
mars.fit$bestTune
```

```
##   nprune degree
## 8      9      1
```

```
# Final model coefficients
mars_coef <- coef(mars.fit$finalModel)

mars_coef
```

```
##  (Intercept)  h(27.8-bmi)   h(time-57)   h(57-time)       gender    h(age-59)
```

```
## 10.847446930 -0.061997354 -0.002254182 -0.033529326 -0.296290451 -0.022957648
##    h(59-age)     smoking2  h(bmi-23.7)
##  0.016138468 -0.205126851 -0.084380175
```

To evaluate the optimal complexity of the MARS model, I performed a grid search across degrees 1 to 4 and a range of 2 to 20 terms (`nprune`). Cross-validation results showed that models with degree = 1 consistently achieved the lowest RMSE, and increasing the interaction degree did not lead to further improvements.

As the interaction `degree` increased, RMSE values became more variable rather than decreasing steadily, suggesting there was no gain in prediction accuracy. The `nprune` range of 2 to 20 also appeared appropriate — RMSE decreased initially with more terms but then plateaued, indicating that adding further complexity would not significantly improve model performance. Based on this, I selected `degree = 1` as the optimal choice.

In other words, cross-validation results showed that a model with approximately 9 terms and no interactions (product degree = 1) minimized prediction error (RMSE  0.529).

**Optimal parameters and final model coefficients** The best-tuned model selected `degree = 1` and `nprune = 9`. Therefore, the final model includes 9 retained terms with no interaction effects.

Next, obtaining the test error:

```r
# Set seed for reproducibility
set.seed(299)

# Obtain response variable from testing data
y_testing <- testing_data$log_antibody

# Predict on test data using trained MARS model
y_pred_MARS <- predict(mars.fit, newdata = testing_data)

# Compute RMSE (Test Error)
test_rmse_mars <- sqrt(mean((y_testing - y_pred_MARS)^2))

# Print test RMSE
test_rmse_mars
```

```
## [1] 0.5327718
```

Therefore, the test error (RMSE) is `0.5327718`.

Next, obtaining the training error:

```r
# Set seed for reproducibility
set.seed(299)

# Predict on training data using MARS model
y_pred_mars_train <- predict(mars.fit, newdata = training_data)

# Compute Training RMSE for MARS Model
train_rmse_mars <- sqrt(mean((training_data$log_antibody - y_pred_mars_train)^2))

# Print Training RMSE
print(train_rmse_mars)
```

```
## [1] 0.5261998
```

Therefore, the training error (RMSE) is **0.5261998**.

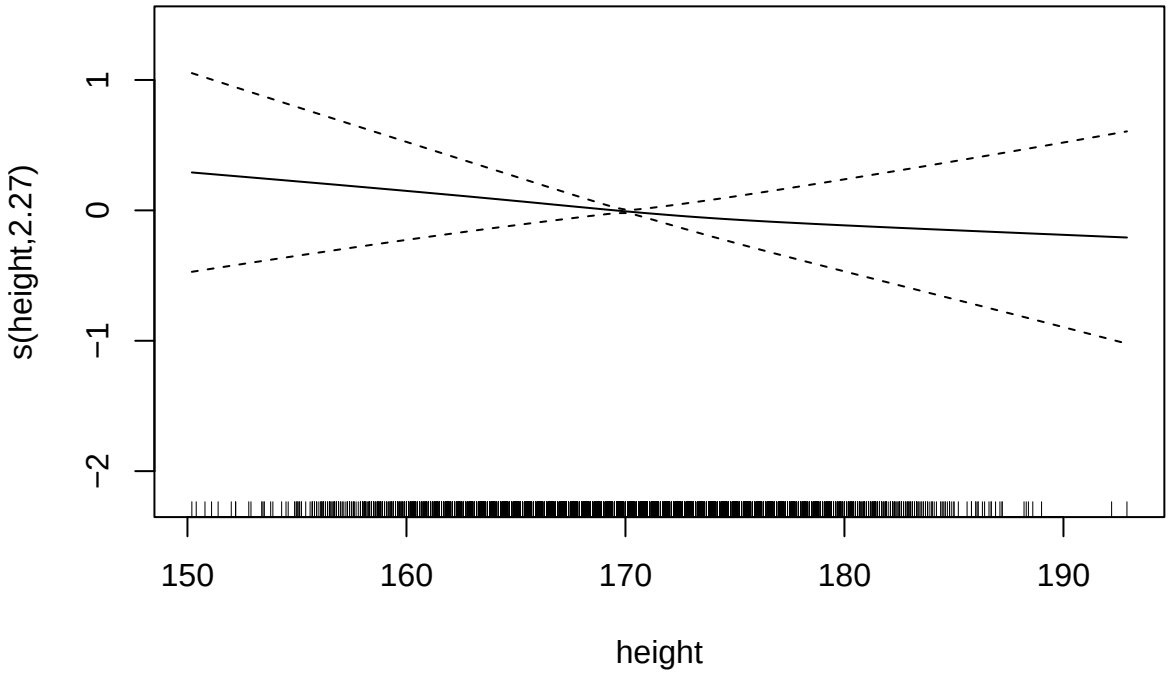
## GAM - Naomi

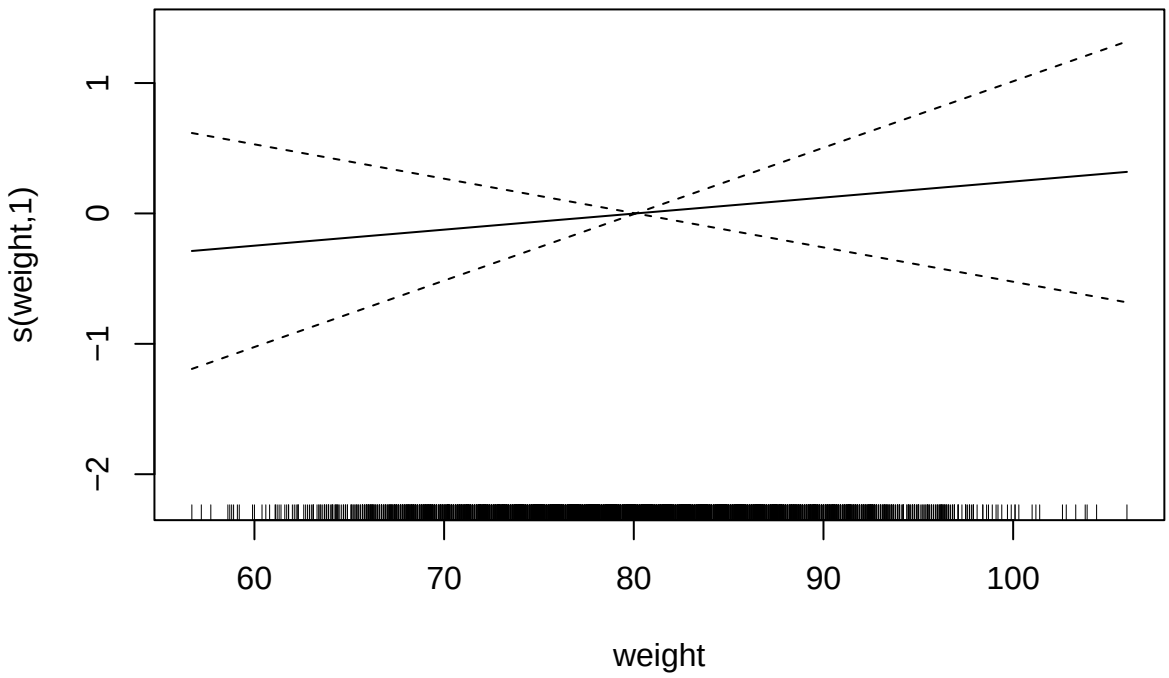
I will proceed with constructing a GAM model, allowing us to mix non-linear and linear terms and build a model estimating the relationship between the outcome (`log_antibody`) and predictors in the provided dataset.

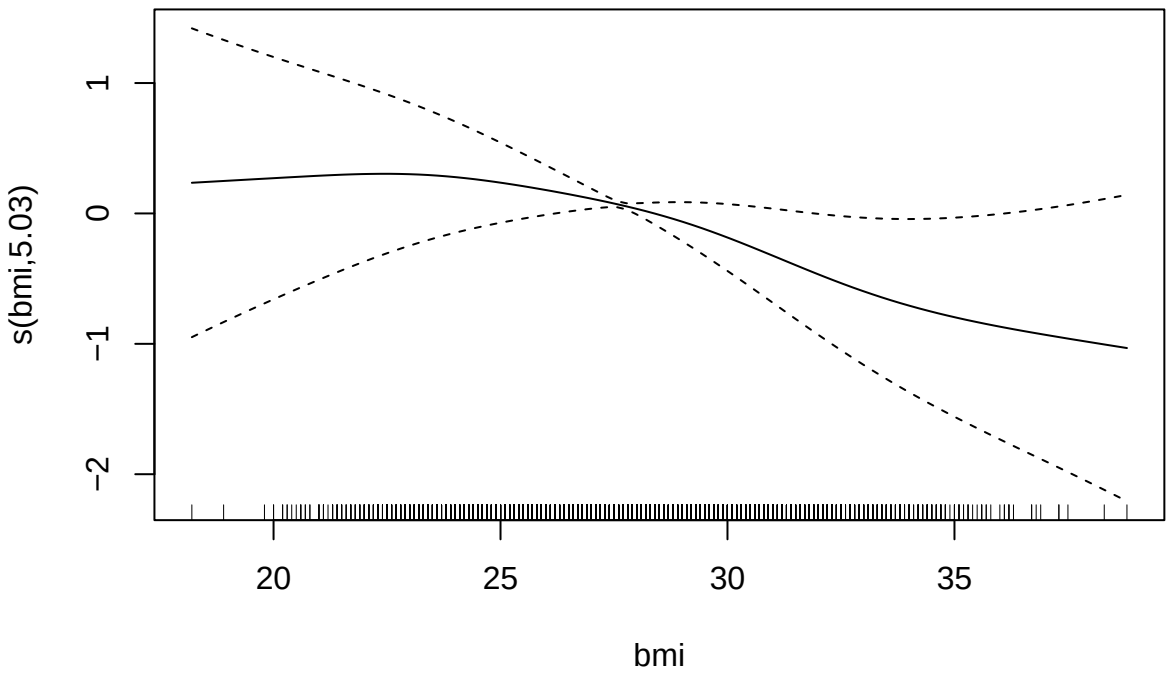
```
# Set seed for reproducibility
set.seed(299)

# Fit a GAM model, using training data
gam_antibody <- gam(log_antibody ~ gender + race + smoking +
                      s(height) + s(weight) + s(bmi) +
                      diabetes + hypertension + s(SBP) +
                      s(LDL) + s(time),
                    data = training_data)

# Summary
summary(gam_antibody)
```

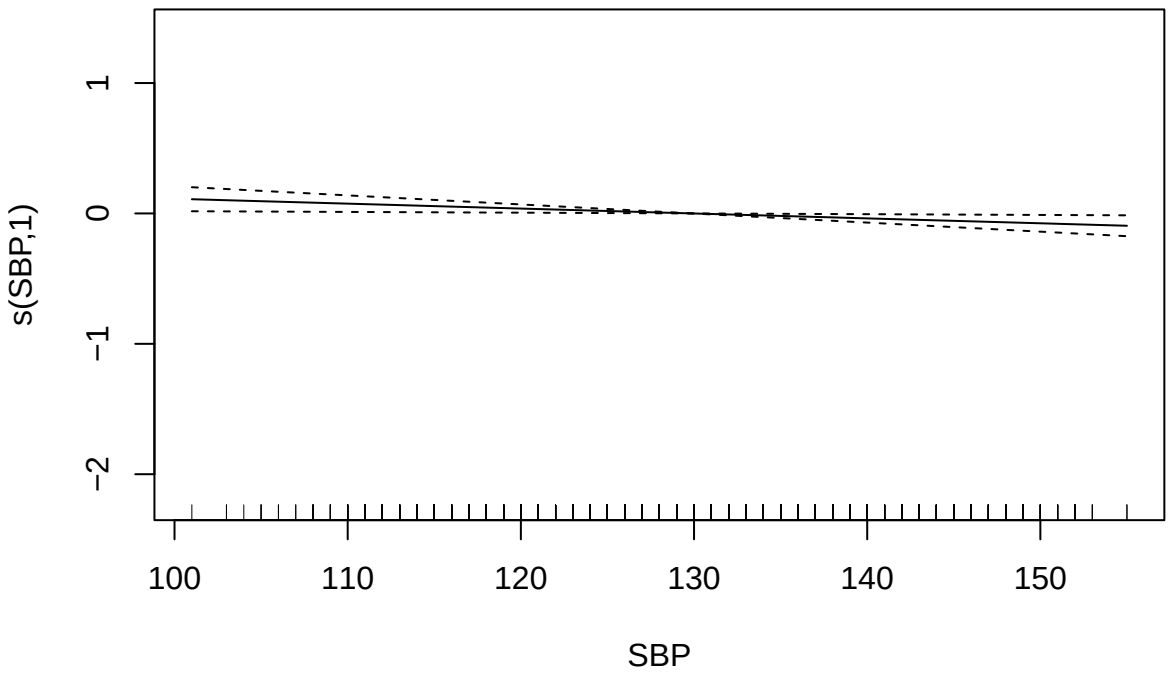
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log_antibody ~ gender + race + smoking + s(height) + s(weight) +
##     s(bmi) + diabetes + hypertension + s(SBP) + s(LDL) + s(time)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.231010   0.017812 574.396  < 2e-16 ***
## gender       -0.295517   0.015112 -19.555  < 2e-16 ***
## race2        -0.004687   0.033401  -0.140    0.888
## race3        -0.009166   0.019068  -0.481    0.631
## race4        -0.034991   0.026485  -1.321    0.187
## smoking1      0.021818   0.016858   1.294    0.196
## smoking2     -0.195446   0.026150  -7.474 9.14e-14 ***
## diabetes      0.013428   0.020892   0.643    0.520
## hypertension -0.016208   0.025289  -0.641    0.522
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df      F p-value
## s(height) 2.272  2.938  0.902  0.3736
## s(weight) 1.000  1.000  0.405  0.5245
## s(bmi)    5.032  6.110 18.024  <2e-16 ***
## s(SBP)    1.000  1.000  5.582  0.0182 *
## s(LDL)    1.000  1.000  3.334  0.0679 .
## s(time)   7.835  8.544 46.085  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.203   Deviance explained = 20.7%
## GCV = 0.28529  Scale est. = 0.28374   n = 5000
```

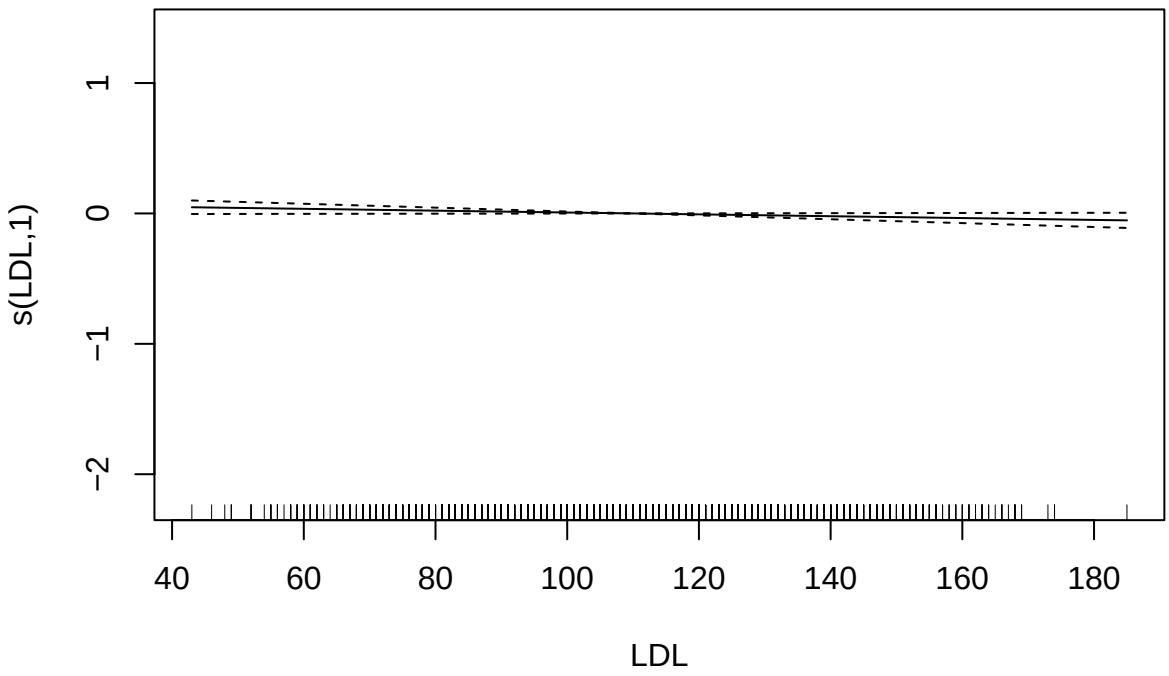
```r
# Plot GAM
plot(gam_antibody)
```

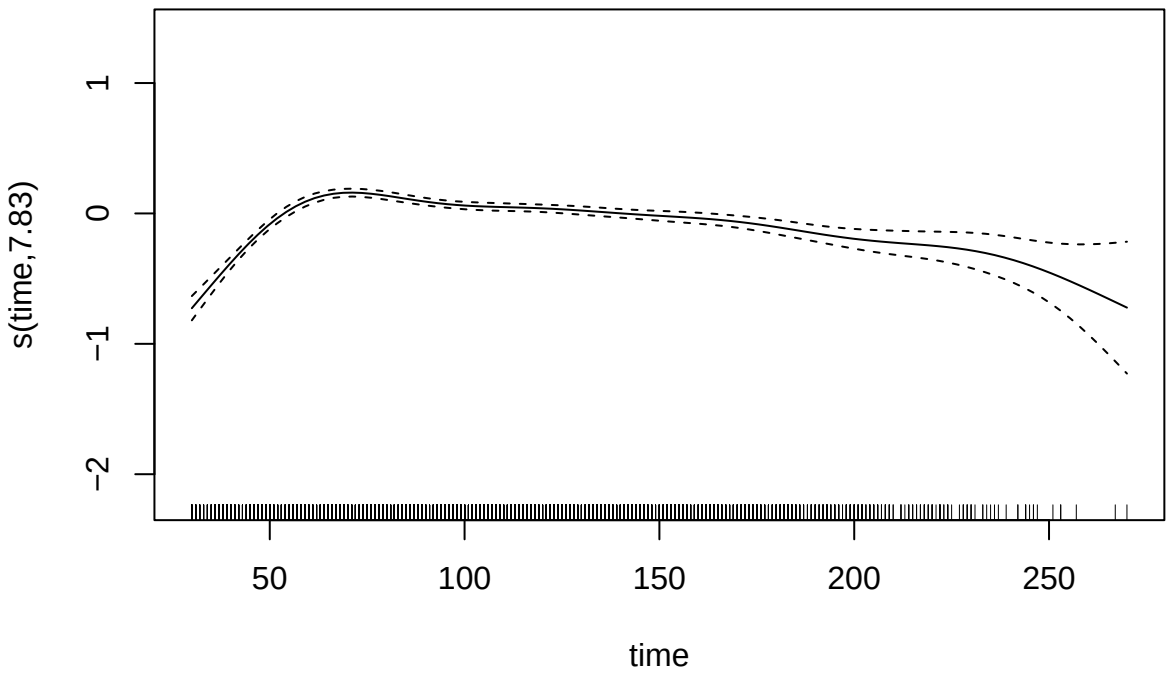



s(weight,1)
1
0
−1
−2
60
70
80
90
100
weight


s(bmi,5.03)
1
0
−1
−2
20
25
30
35
bmi


s(SBP,1)
1
0
−1
−2
100
110
120
130
140
150
SBP


s(LDL,1)
1
0
−1
−2
40
60
80
100
120
140
160
180
LDL




- Some of the predictors have estimated degrees of freedom (edf) greater than 1, specifically height, BMI, and time, indicating that they are nonlinear smooth functions. This is supported by the GAM predictor plots.
- Others appear approximately linear (edf  1), including weight, SBP, and LDL.
- Among the nonlinear terms, BMI and time are highly significant ($p < 2e\text{-}16$), suggesting meaningful nonlinear associations with the log antibody levels.
- However, not all nonlinear terms with edf $> 1$ are statistically significant. i.e, height has an edf of 2.272 but a non-significant p-value ($p = 0.3736$).
- SBP is a linear term (edf $= 1.00$) and also statistically significant ($p = 0.0182$), suggesting a meaningful linear relationship. It is the only linear term that is statistically significant.

Next, obtaining the training error:

```
# Note I have not done test error
# Set seed for reproducibility
set.seed(299)

# Prediction using training data
y_train_pred_gam <- predict(gam_antibody, newdata = training_data)

# Training RMSE
train_rmse_gam <- sqrt(mean((training_data$log_antibody - y_train_pred_gam)^2))
```

```r
print(train_rmse_gam)
```

```
## [1] 0.5312249
```

Therefore, the training set error (RMSE) is **0.5312249**.