Midterm project

Tingyi Li

Contents


```
library(ISLR)
library(glmnet)
library(caret)
library(tidymodels)
library(pls)
library(earth)
library(bayesQR)
library(ggplot2)
library(plotmo)
library(corrplot)
library(patchwork)
```

```
load("./recovery.RData")
data <- subset(dat, select = -id)
data$study <- ifelse(data$study == "A", 0, 1)

data <- data |>
  mutate(
    gender = as.factor(gender),
    race = as.factor(race),
    smoking = as.factor(smoking),
    hypertension = as.factor(hypertension),
    diabetes = as.factor(diabetes),
    vaccine = as.factor(vaccine),
```

```
    severity = as.factor(severity),
    study = as.factor(study)
    )
```

Data Partition

Divide data into training data (80%) and testing data (20%)

```
set.seed(666)
data_split <- initial_split(data, prop = 0.8)

# training data
training_data <- training(data_split)
training_data <- na.omit(training_data)


# testing data
testing_data <- testing(data_split)
testing_data <- na.omit(testing_data)

# matrix of predictors
x <- model.matrix(recovery_time ~ ., training_data) [ ,-1]
y <- training_data$recovery_time

x2 <- model.matrix(recovery_time ~ ., testing_data) [ ,-1]
y2 <- testing_data$recovery_time

# 10-fold cross validation
ctrl1 <- trainControl(method = "repeatedcv", repeats = 10)
```

EDA

```
skimr::skim(data)
```

Table 1: Data summary

Name	data
Number of rows	3000
Number of columns	15
Column type frequency:	
factor	8
numeric	7
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
gender	0	1	FALSE	2	0: 1544, 1: 1456

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
race	0	1	FALSE	4	1: 1967, 3: 604, 4: 271, 2: 158
smoking	0	1	FALSE	3	0: 1822, 1: 859, 2: 319
hypertension	0	1	FALSE	2	0: 1508, 1: 1492
diabetes	0	1	FALSE	2	0: 2537, 1: 463
vaccine	0	1	FALSE	2	1: 1788, 0: 1212
severity	0	1	FALSE	2	0: 2679, 1: 321
study	0	1	FALSE	2	0: 2000, 1: 1000

Variable type: numeric

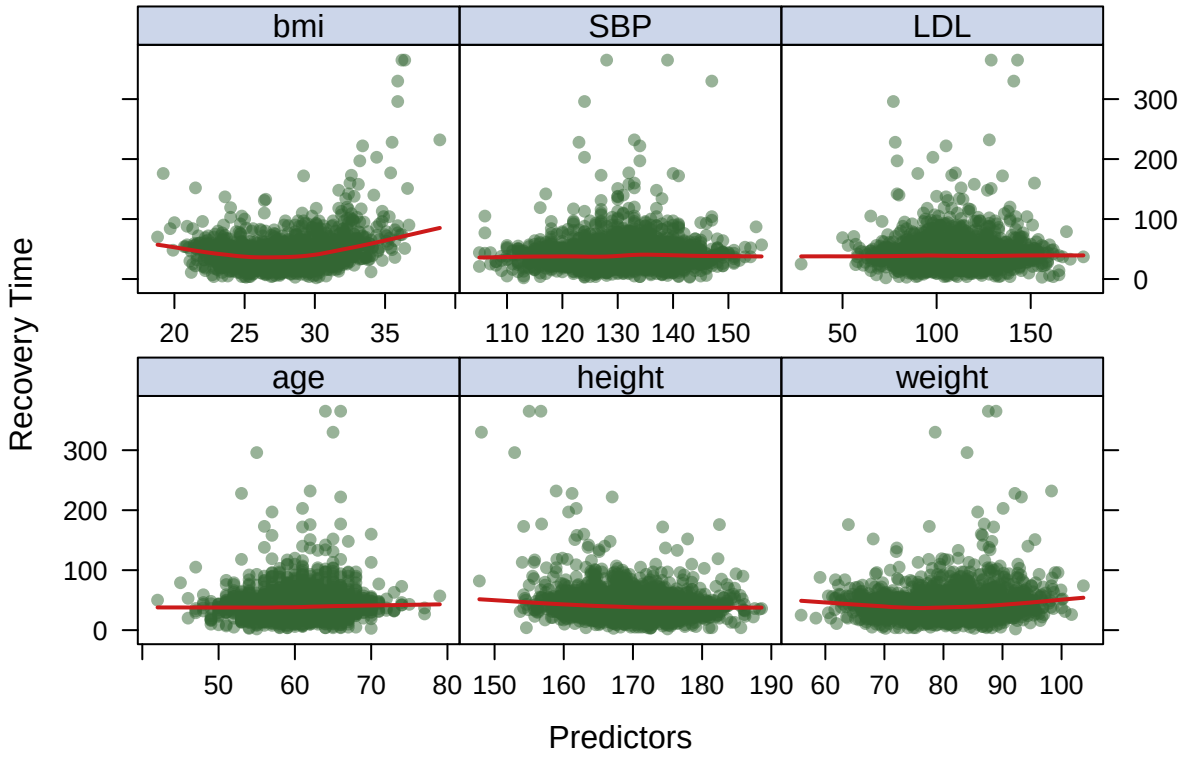
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	60.20	4.48	42.0	57.0	60.00	63.0	79.0	
height	0	1	169.90	5.97	147.8	166.0	169.90	173.9	188.6	
weight	0	1	79.96	7.14	55.9	75.2	79.80	84.8	103.7	
bmi	0	1	27.76	2.79	18.8	25.8	27.65	29.5	38.9	
SBP	0	1	130.47	7.97	105.0	125.0	130.00	136.0	156.0	
LDL	0	1	110.45	19.76	28.0	97.0	110.00	124.0	178.0	
recovery_time	0	1	42.17	23.15	2.0	31.0	39.00	49.0	365.0	

Feature plot for continuous variables

```
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)

trainData <- training_data|>
  dplyr::select('age', 'height', 'weight', 'bmi', 'SBP', 'LDL', 'recovery_time')
featurePlot(x = trainData[ , 1:6],
            y = trainData[ , 7],
            plot = "scatter",
            span = .5,
            labels = c("Predictors","Recovery Time"),
            main = "Figure 1:Relationship between Continuous Predictors and Recovery Time",
            type = c("p", "smooth"),
            layout = c(3, 2))
```

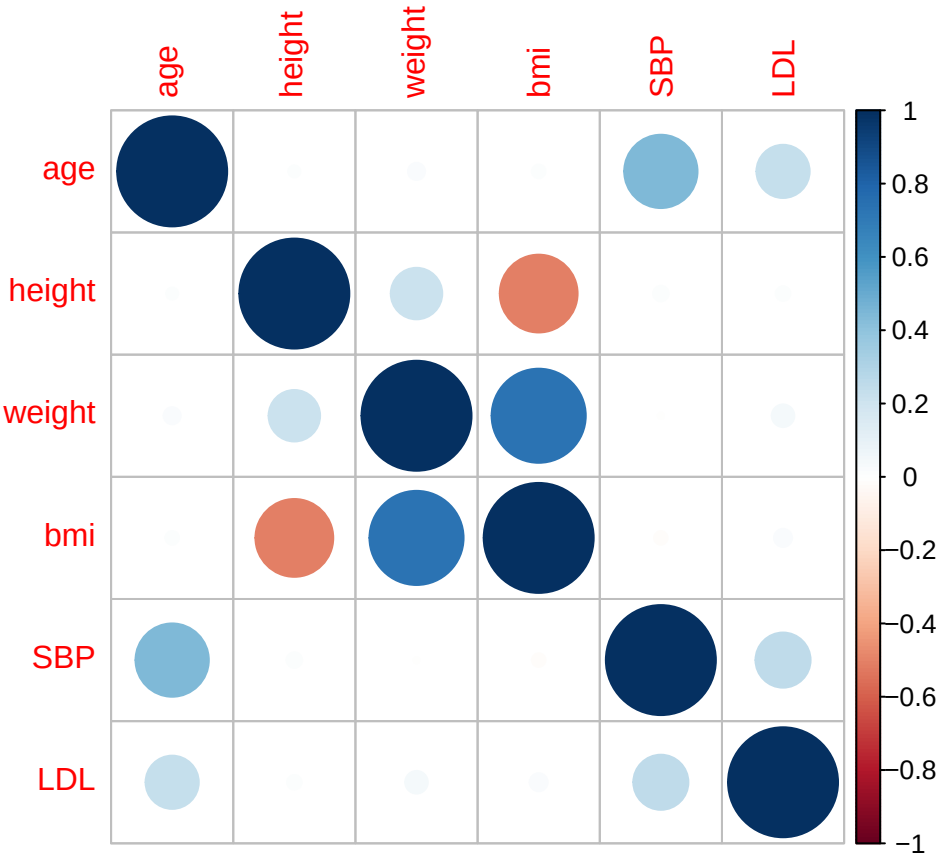
igure 1:Relationship between Continuous Predictors and Recovery Tim



Correlation plot

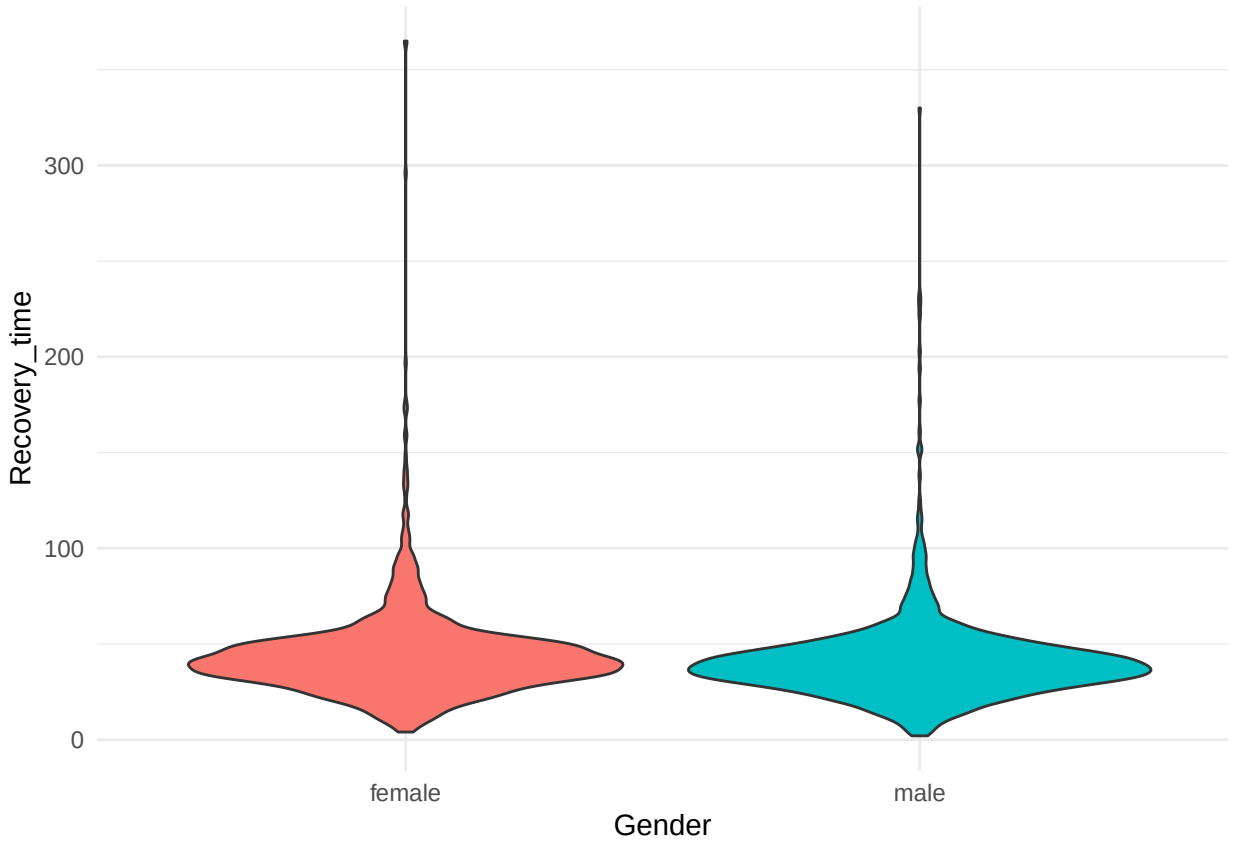
```
corr_data = training_data |>
  dplyr::select('age', 'height', 'weight', 'bmi', 'SBP', 'LDL')

corrplot(cor(corr_data), method = "circle", type = "full")
```

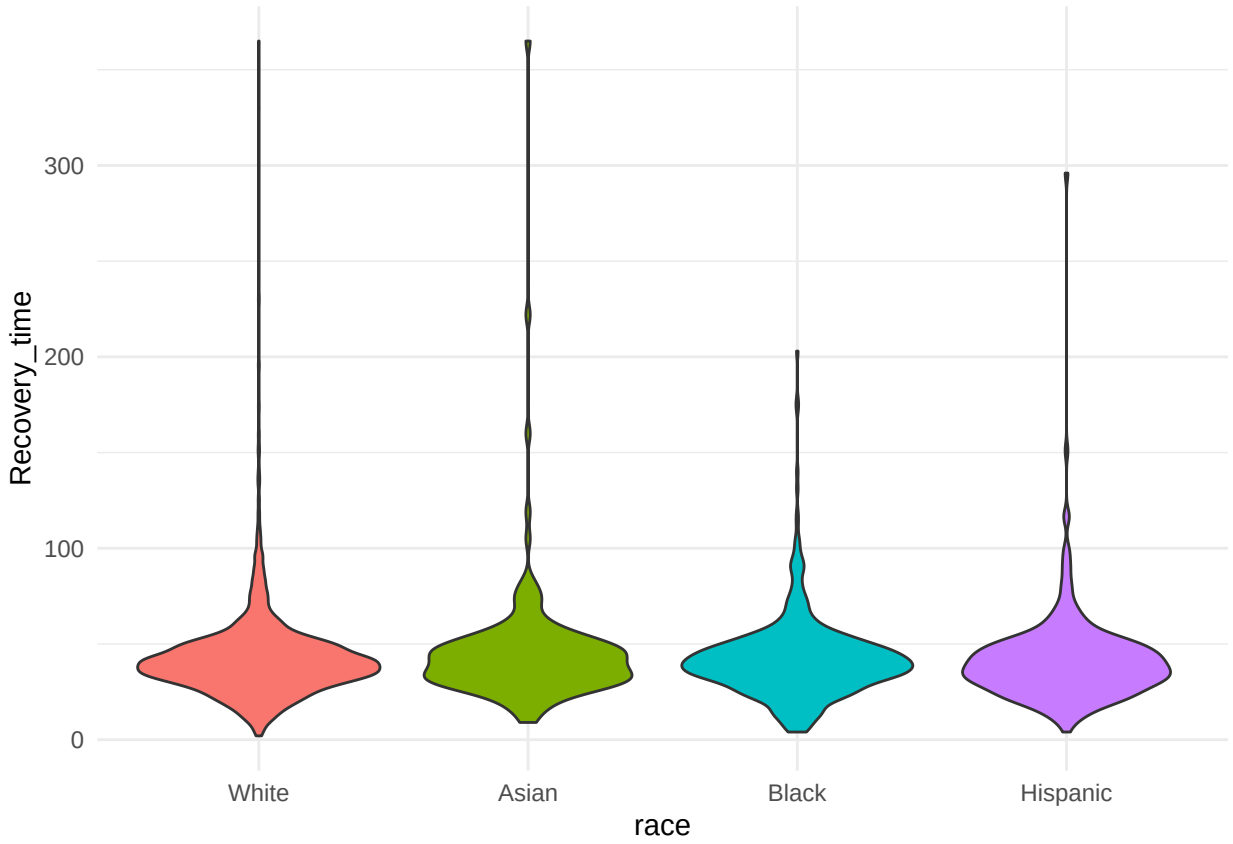


Violin plot for categorical variables

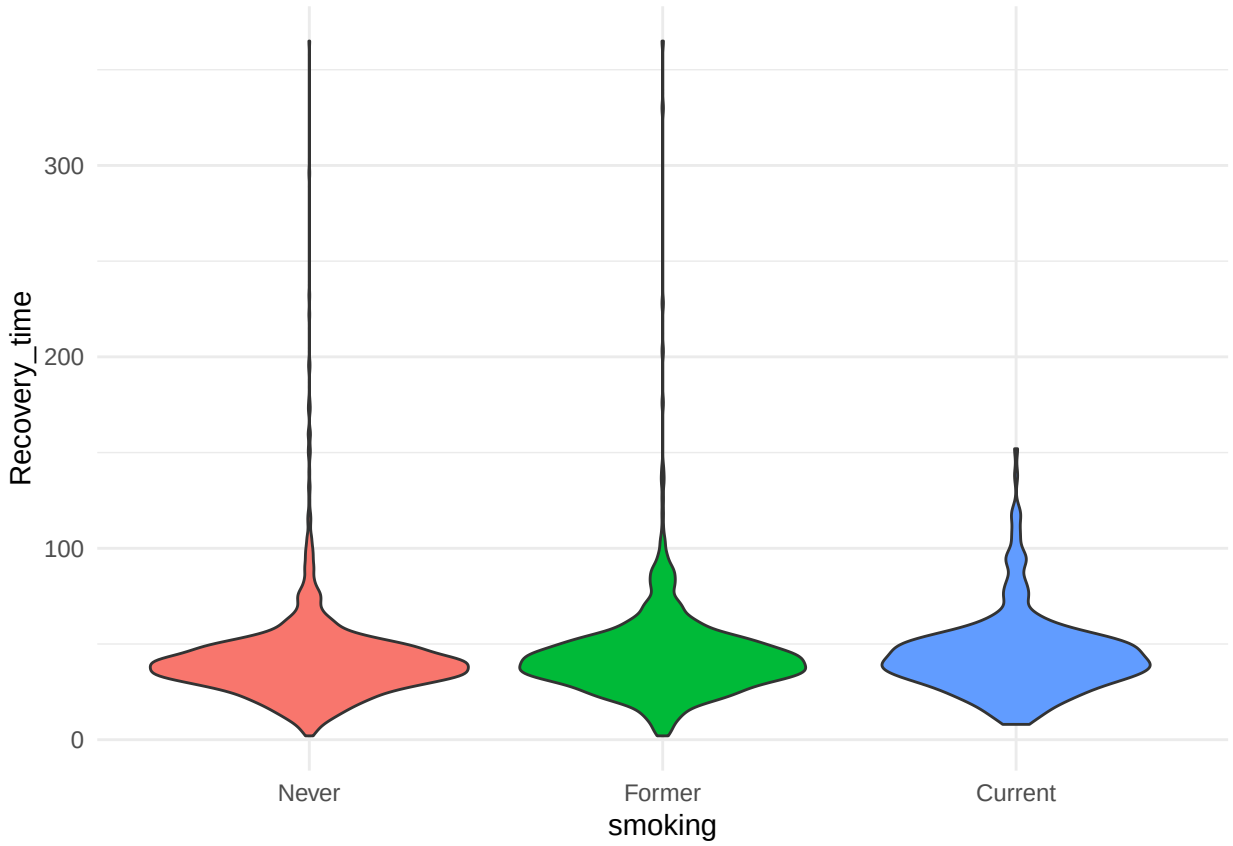
```
gender_plot <- data |>
  ggplot(aes(x = gender, y = recovery_time, fill = gender))+
  geom_violin() +
  scale_x_discrete(labels = c("female", "male")) +
  labs(
       x = "Gender",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
gender_plot
```



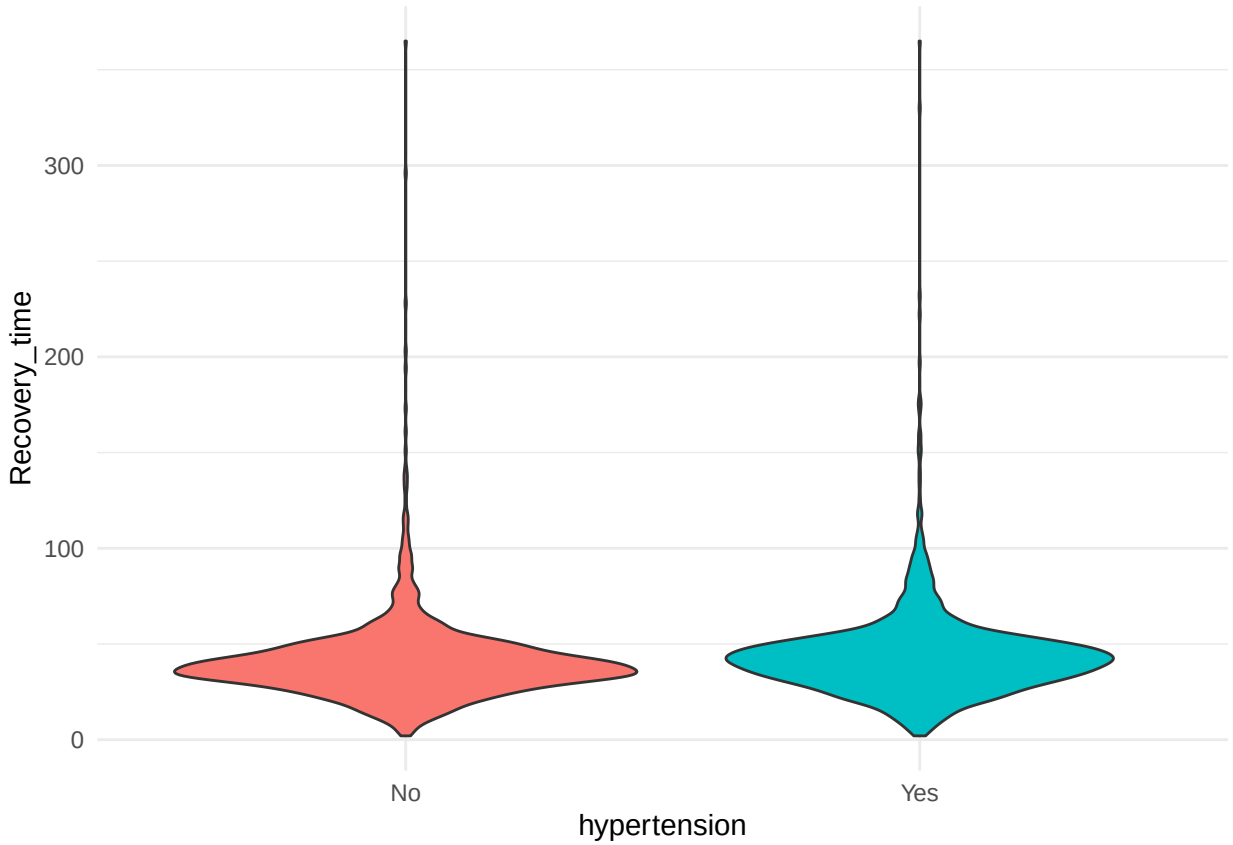

```
race_plot <- data |>
  ggplot(aes(x = race, y = recovery_time, fill = race)) +
  geom_violin() +
  scale_x_discrete(labels = c("White", "Asian", "Black", "Hispanic")) +
  labs(
       x = "race",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
race_plot
```



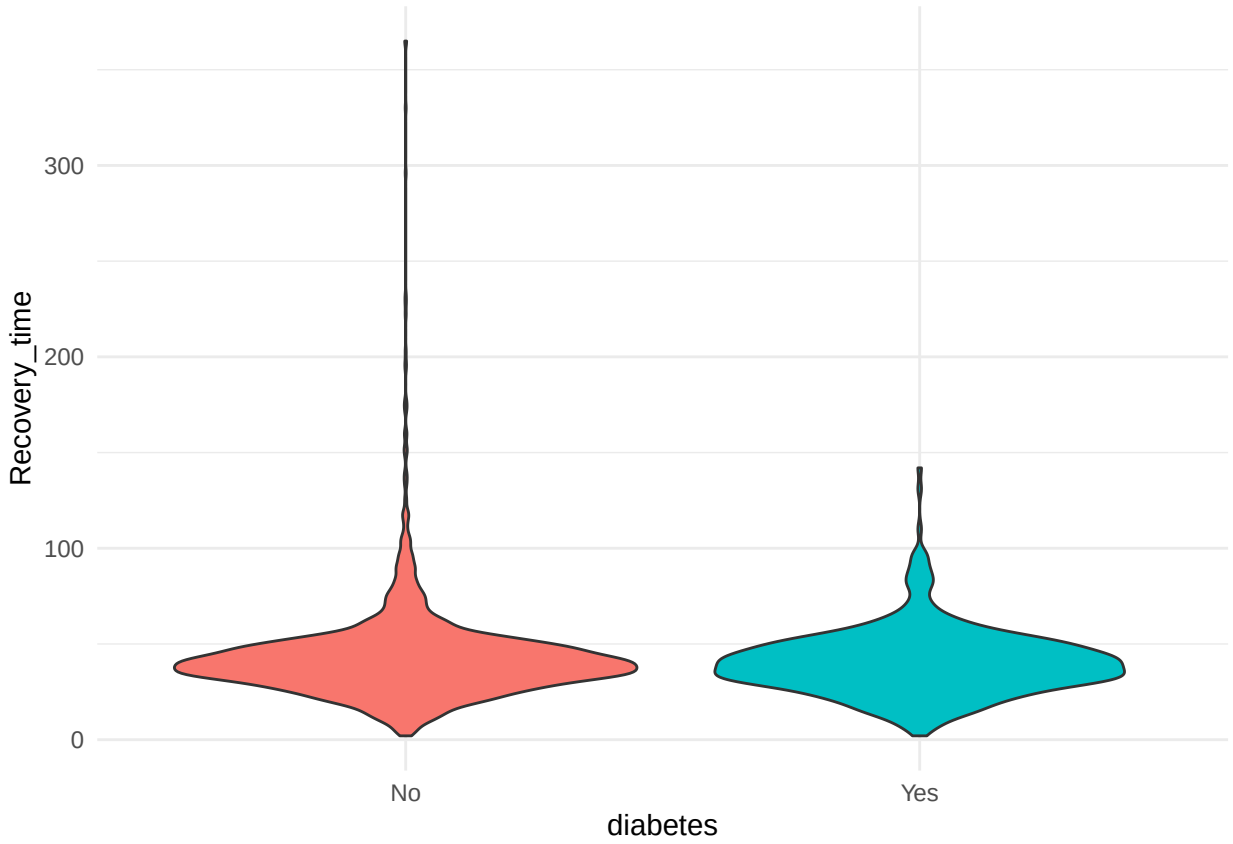

```
smoking_plot <- data |>
  ggplot(aes(x = smoking, y = recovery_time, fill = smoking)) +
  geom_violin() +
  scale_x_discrete(labels = c("Never", "Former", "Current")) +
  labs(
       x = "smoking",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
smoking_plot
```



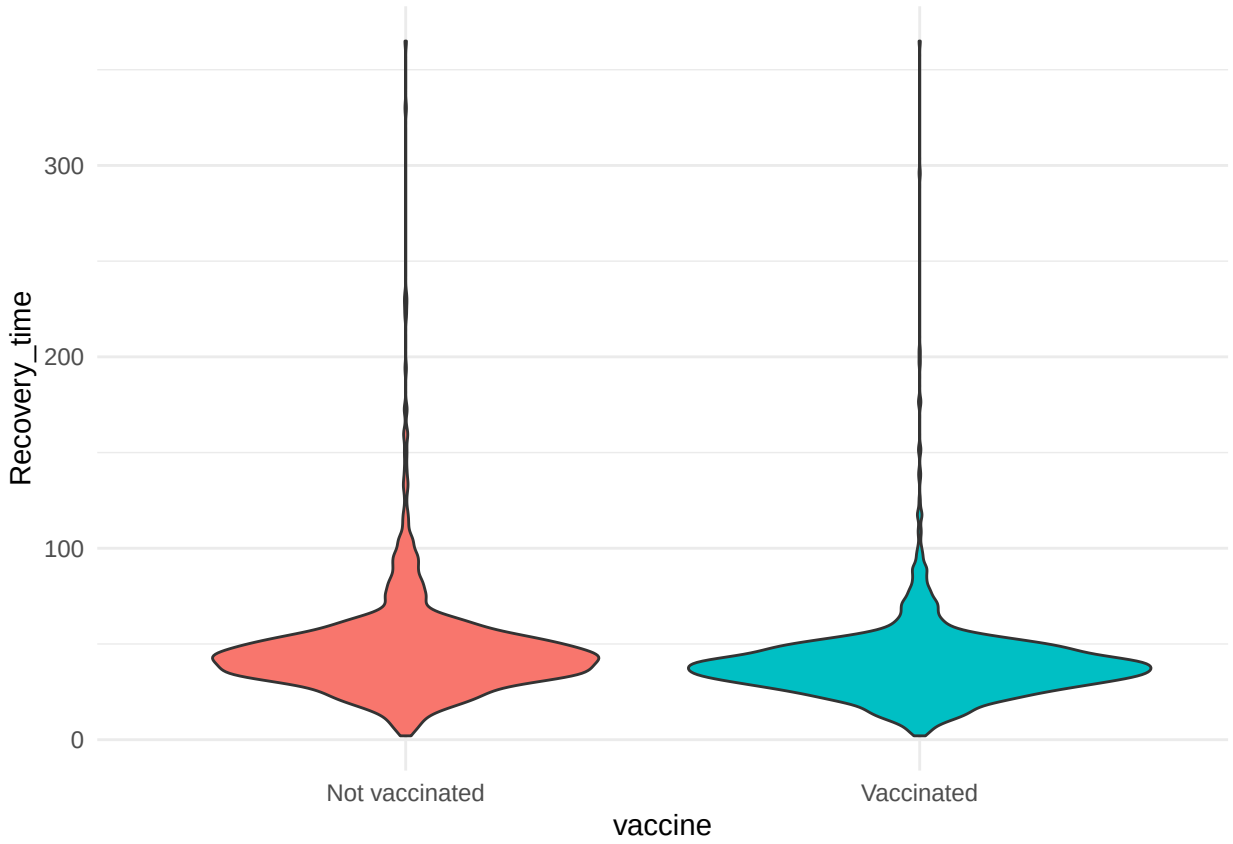

```
hypertension_plot <- data |>
  ggplot(aes(x = hypertension, y = recovery_time, fill = hypertension)) +
  geom_violin() +
  scale_x_discrete(labels = c("No", "Yes")) +
  labs(
       x = "hypertension",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
hypertension_plot
```



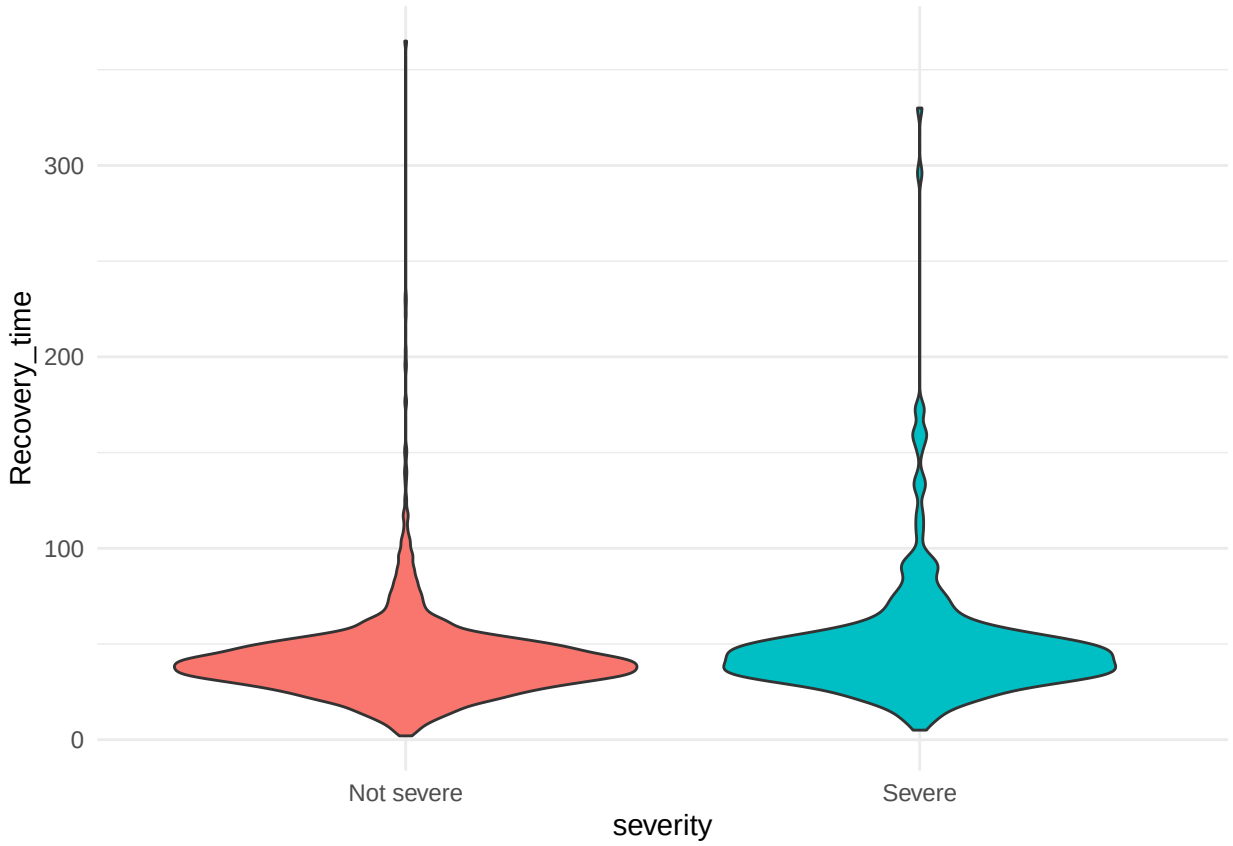

```
diabetes_plot <- data |>
  ggplot(aes(x = diabetes, y = recovery_time, fill = diabetes)) +
  geom_violin() +
  scale_x_discrete(labels = c("No", "Yes")) +
  labs(
       x = "diabetes",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
diabetes_plot
```



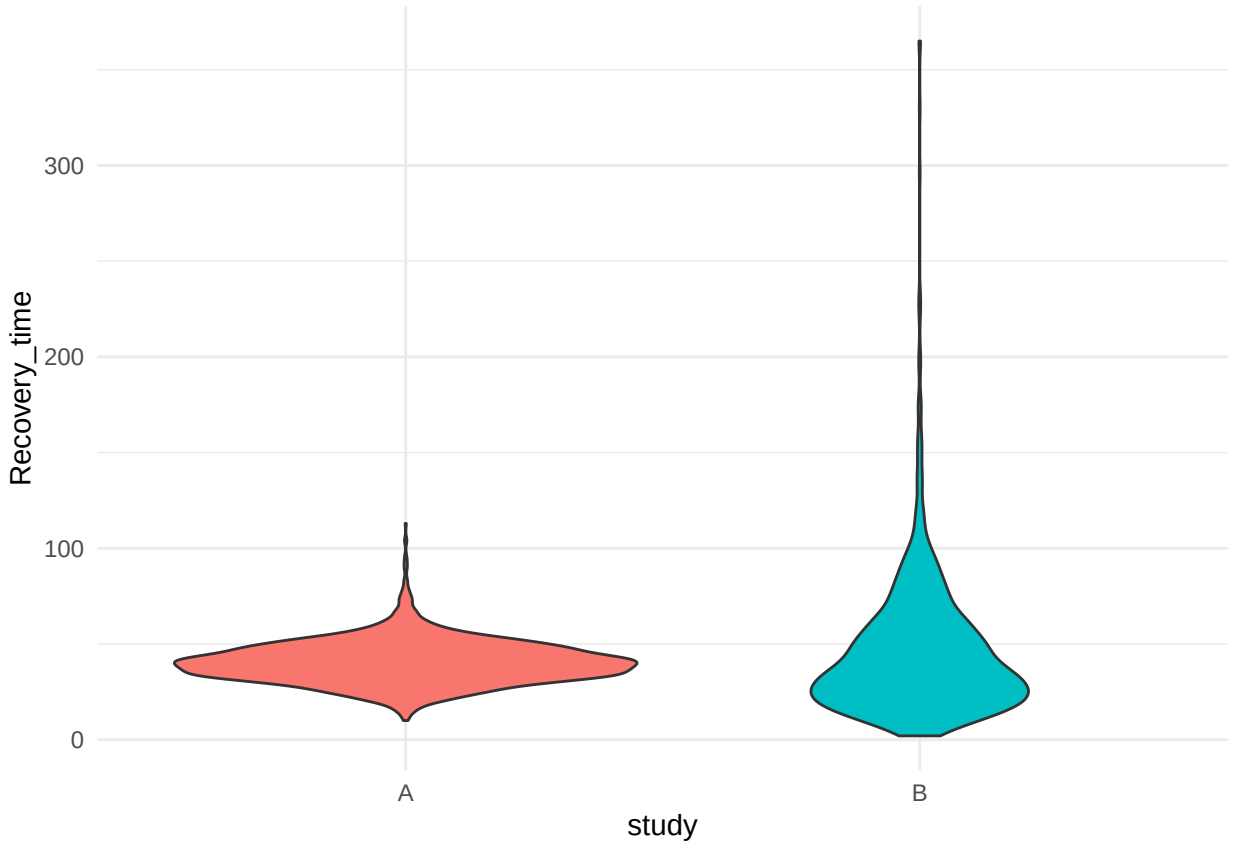

```
vaccine_plot <- data |>
   ggplot(aes(x = vaccine, y = recovery_time, fill = vaccine)) +
   geom_violin() +
   scale_x_discrete(labels = c("Not vaccinated", "Vaccinated")) +
   labs(
        x = "vaccine",
        y = "Recovery_time") +
   theme_minimal() + theme(legend.position = "none")
vaccine_plot
```




```
severity_plot <- data |>
  ggplot(aes(x = severity, y = recovery_time, fill = severity)) +
  geom_violin() +
  scale_x_discrete(labels = c("Not severe", "Severe")) +
  labs(
       x = "severity",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
severity_plot
```



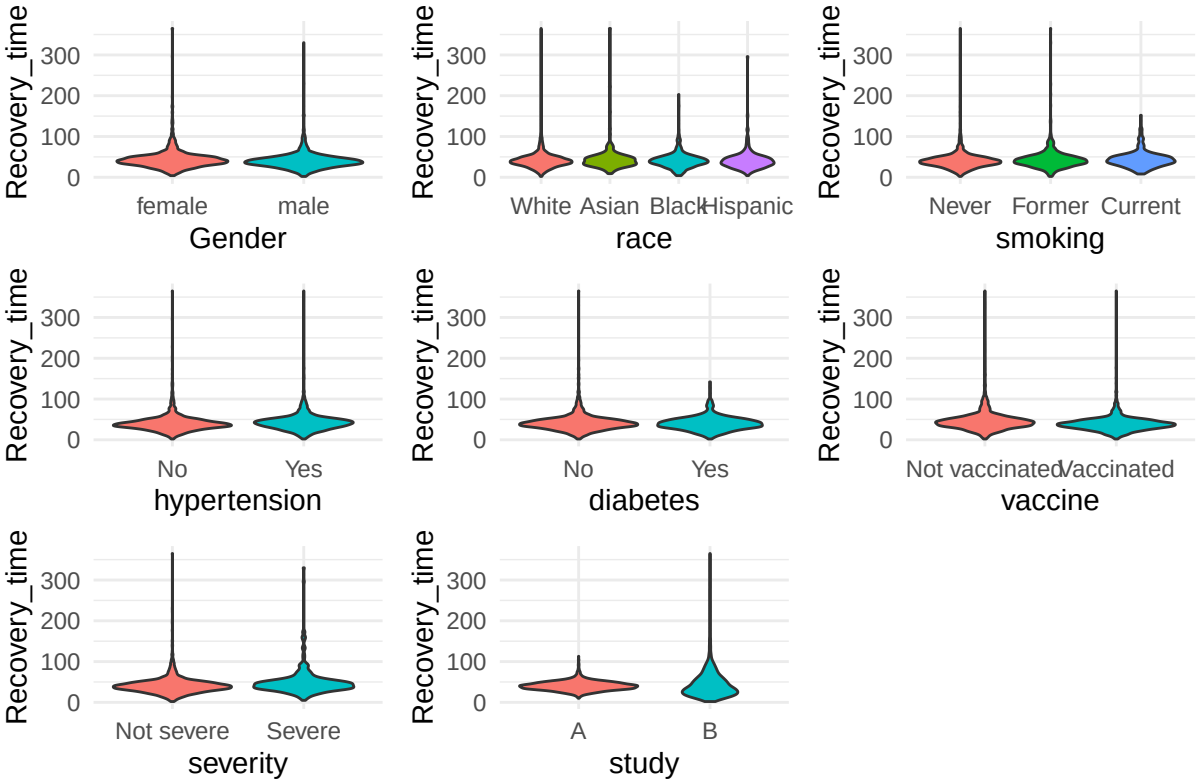

```
study_plot <- data |>
  ggplot(aes(x = study, y = recovery_time, fill = study)) +
  geom_violin() +
  scale_x_discrete(labels = c("A", "B")) +
  labs(
       x = "study",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
study_plot
```




```
combined <- gender_plot + race_plot + smoking_plot + hypertension_plot + diabetes_plot + vaccine_plot +

combined + plot_annotation(title = "Figure 2: Relationship between Categorical Predictors and Recovery T
```

Figure 2: Relationship between Categorical Predictors and Recovery Time



Linear models

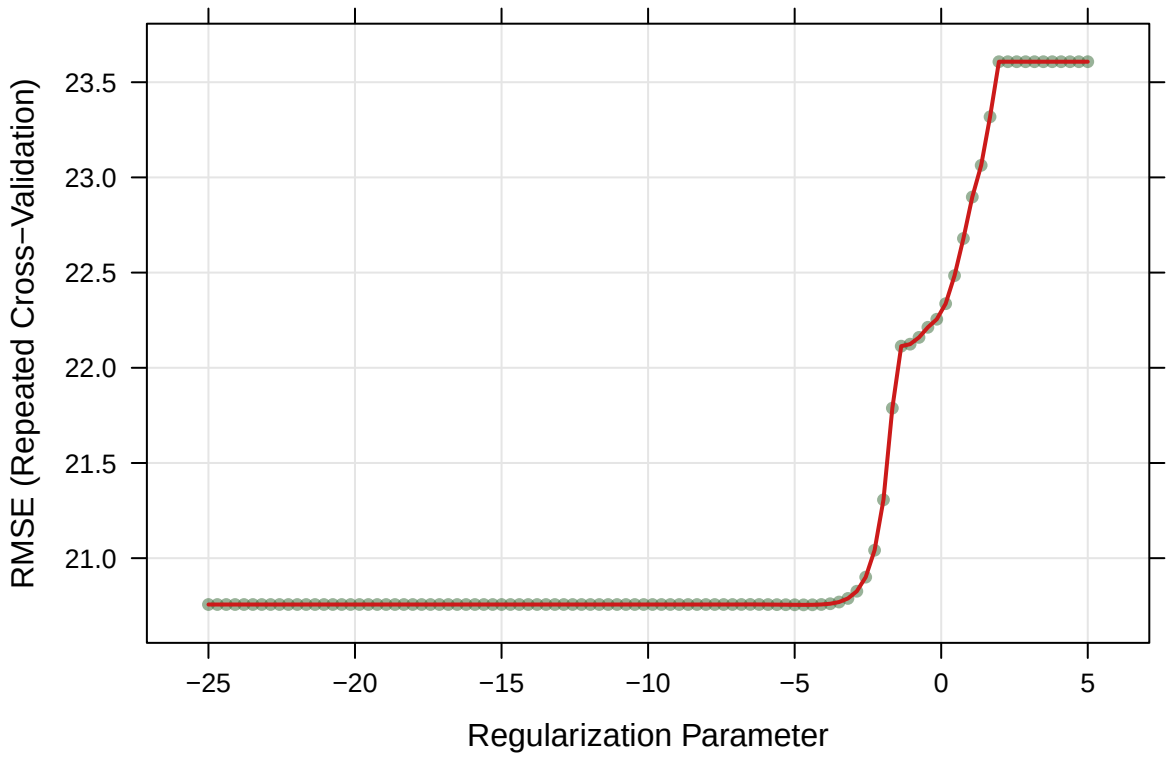
Fit a lasso model

```
set.seed(666)

lasso.fit <- train(recovery_time ~ .,
                   data = training_data,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(-25, 5, length = 100))),
                   trControl = ctrl1)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
plot(lasso.fit, xTrans = log)
```




```
## Select the best tuning parameter
lasso.fit$bestTune
```

```
##    alpha     lambda
## 68     1 0.00912288
```

```
# coefficients in the final model
coef(lasso.fit$finalModel, lasso.fit$bestTune$lambda)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                            s1
## (Intercept)   -2.229189e+03
## age            2.736904e-01
## gender1       -3.698081e+00
## race2          1.629521e+00
## race3         -7.478838e-01
## race4         -8.493986e-01
## smoking1       2.764637e+00
## smoking2       3.624109e+00
## height         1.302739e+01
## weight        -1.414814e+01
## bmi            4.250183e+01
## hypertension1  3.023576e+00
## diabetes1     -1.066279e+00
## SBP           -1.899186e-02
## LDL           -3.709871e-02
## vaccine1      -6.014455e+00
## severity1      7.390563e+00
## study1         5.298800e+00
```

The selected tuning parameter is 0.00912288.

Test errors

```
set.seed(666)
lasso.pred <- predict(lasso.fit, newdata = testing_data)
mean((lasso.pred - testing_data[, "recovery_time"])^2)
```

```
## [1] 298.3016
```

The test error is 298.3016.

Fit an elastic net model

```
set.seed(666)
enet.fit <- train(recovery_time ~ .,
                  data = training_data,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(-25, 5, length = 100))),

                  trControl = ctrl1)
```

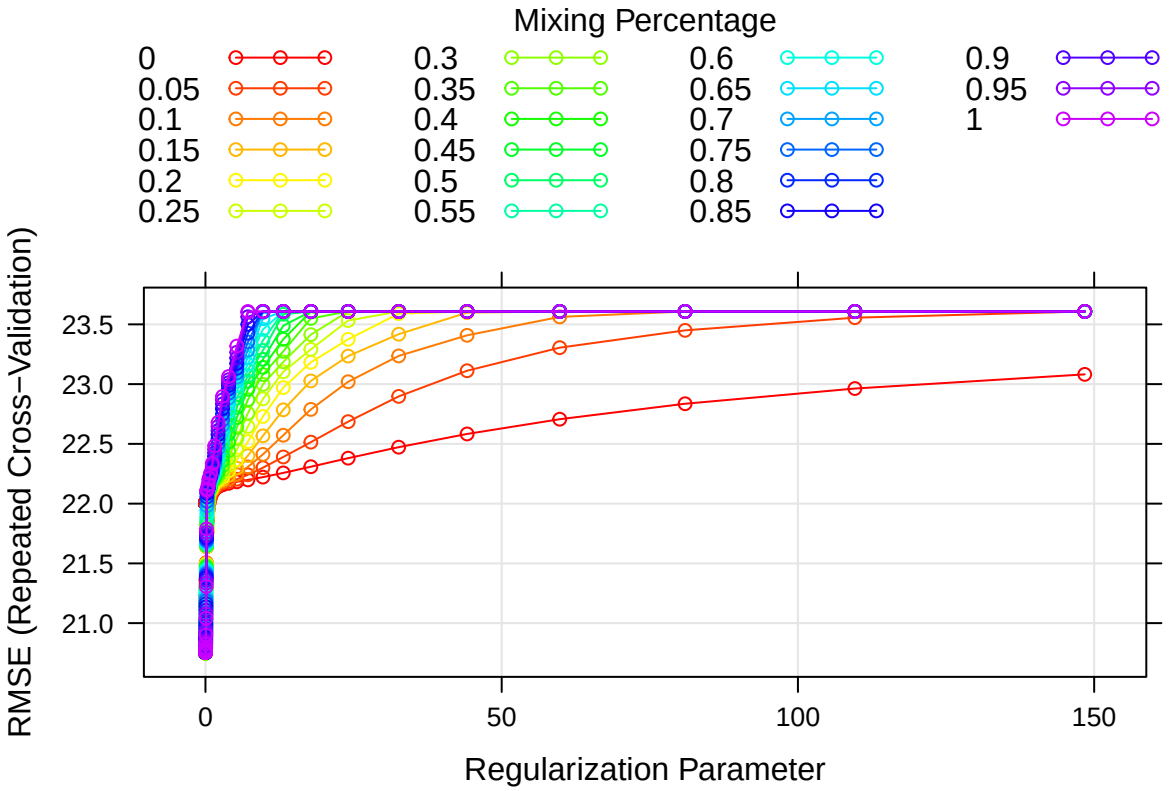
```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
## Select the best tuning parameter
enet.fit$bestTune
```

```
##     alpha     lambda
## 365  0.15 0.00367552
```

```
#Plot
myCol <- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))

plot(enet.fit, par.settings = myPar)
```




```
# coefficients in the final model
coef(enet.fit$finalModel, enet.fit$bestTune$lambda)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)   -2.157409e+03
## age            2.786152e-01
## gender1       -3.708843e+00
## race2          1.666442e+00
## race3         -7.696907e-01
## race4         -9.113345e-01
## smoking1       2.780175e+00
## smoking2       3.645642e+00
## height         1.260703e+01
## weight        -1.370319e+01
## bmi            4.122283e+01
## hypertension1  3.105223e+00
## diabetes1     -1.101528e+00
## SBP           -2.443694e-02
## LDL           -3.746249e-02
## vaccine1      -6.040658e+00
## severity1      7.430536e+00
## study1         5.315665e+00
```

```
# test error
enet.pred <- predict(enet.fit, newdata = testing_data)
mean((enet.pred - testing_data[, "recovery_time"])^2)
```

```
## [1] 297.1642
```

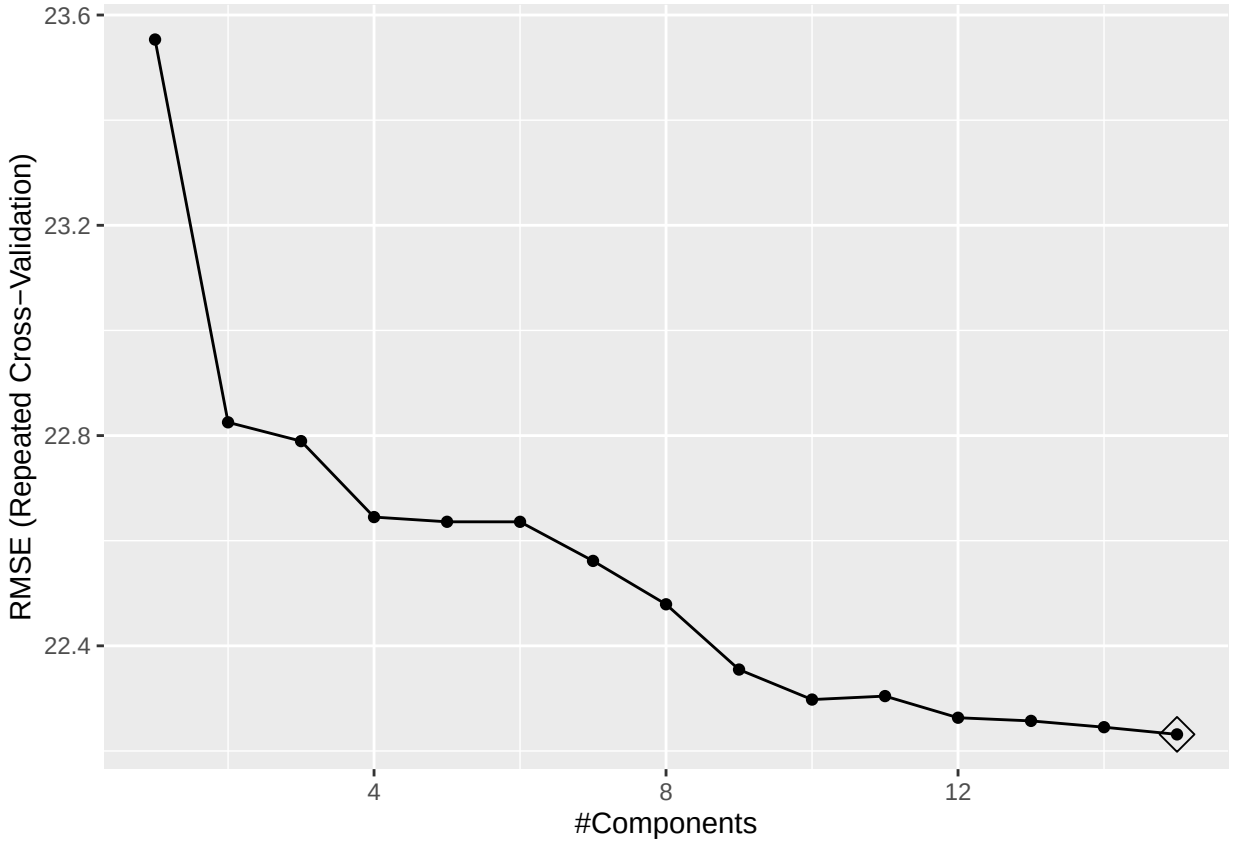
The selected tuning parameter is 0.00367552, and the test error is 297.1642.

Fit a PCR model

```
set.seed(666)
pcr.fit <- train(x, y,
                 method = "pcr",
                 tuneGrid = data.frame(ncomp = 1:15),
                 trControl = ctrl1,
                 scale = TRUE)
predy2.pcr2 <- predict(pcr.fit, newdata = x2)
mean((y2 - predy2.pcr2)^2)
```

```
## [1] 323.779
```

```
ggplot(pcr.fit, highlight = TRUE)
```




```
summary(pcr.fit)
```

```
## Data:    X dimension: 2400 17
##  Y dimension: 2400 1
## Fit method: svdpc
## Number of components considered: 15
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X          13.117   23.728   31.171   38.302   45.266   51.762    57.87
## .outcome    0.602    7.376    7.832    9.341    9.353    9.575    10.59
##           8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
```

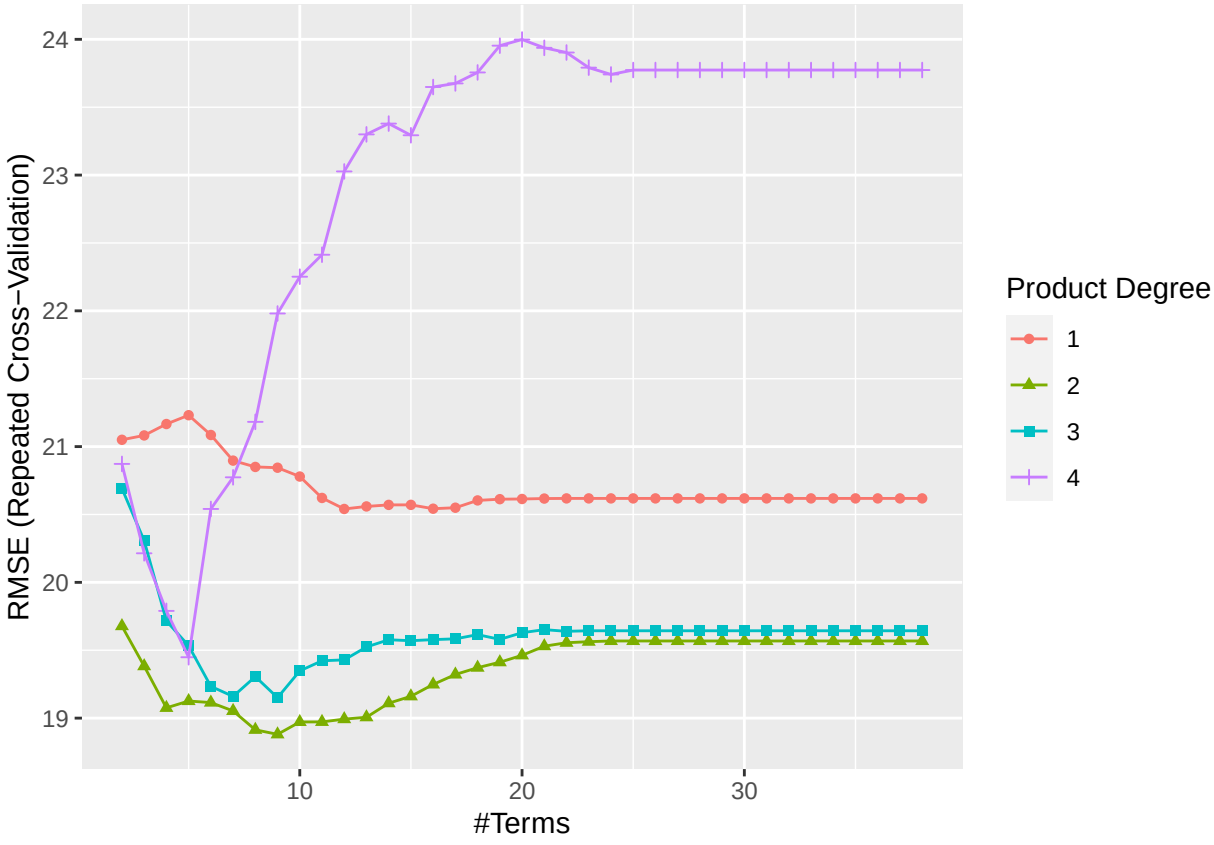
```
## X           63.91    69.84     75.59     81.01     85.98     90.51     94.88
## .outcome    11.16    12.02     12.51     12.51     12.88     12.98     13.16
##           15 comps
## X            98.82
## .outcome     13.32
```

There are 15 components in the model and the test error is 323.779.

Nonlinear models

MARS

```
set.seed(666)
mars.grid <- expand.grid(degree = 1:4,
                         nprune = 2:38)
mars.fit <- train(x, y,
                  method = "earth",
                  tuneGrid = mars.grid,
                  trControl = ctrl1)
ggplot(mars.fit)
```




```
mars.fit$bestTune
```

```
##    nprune degree
## 45      9      2
```

```
coef(mars.fit$finalModel)
```

```
##                 (Intercept)                   h(31-bmi)
##                   10.021897                    4.853416
##          h(bmi-31) * study1 h(height-162.4) * h(bmi-31)
##                   21.543750                    2.352927
##                  h(bmi-25.2)                   vaccine1
##                    5.933235                   -5.494258
##  h(weight-86.4) * h(bmi-31)                 h(bmi-33.8)
##                   -2.316318                   39.271654
##          severity1 * study1
##                   14.094456
```

```
# Test error
mars.pred <- predict(mars.fit, newdata = x2)
mars.test.error <- mean((mars.pred - y2)^2)
mars.test.error
```

```
## [1] 251.9653
```

The regression function should be: 10.021897 + 4.853416 * h(31-bmi) + 21.54375 * h(bmi-31) * study + 2.352927 * h(height-162.4) * h(bmi-31) + 5.5933235 * h(bmi-25.2) - 5.494258 * vaccine - 2.316318 * h(weight-86.4) * h(bmi-31) + 39.271654 * h(bmi-33.8) + 14.094456 * severity * study

The test error is 251.9653

GAM

```
set.seed(666)

gam.fit <- train(x, y,
                 method = "gam",
                 tuneGrid = data.frame(method = "GCV.Cp", select = c(TRUE,FALSE)),
                 trControl = ctrl1)
```

```
## Loading required package: mgcv
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## This is mgcv 1.9-0. For overview type 'help("mgcv-package")'.
```

```
gam.fit$bestTune
```

```
##   select method
## 1  FALSE GCV.Cp
```

```
gam.fit$finalModel
```

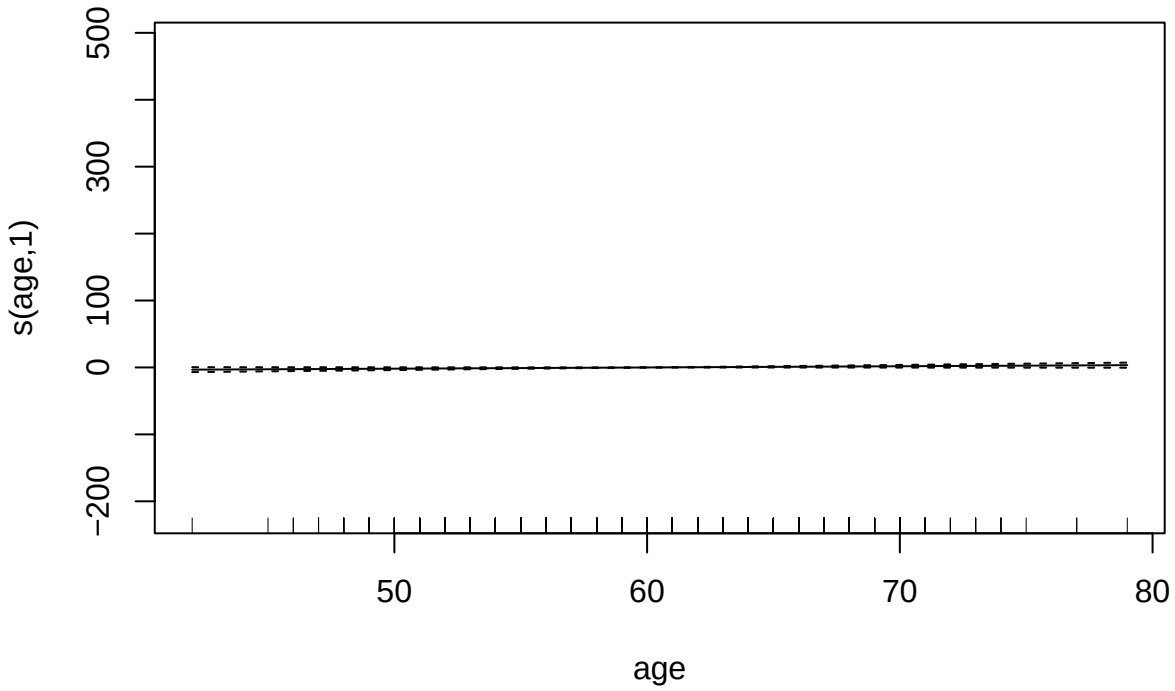
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ gender1 + race2 + race3 + race4 + smoking1 + smoking2 +
```

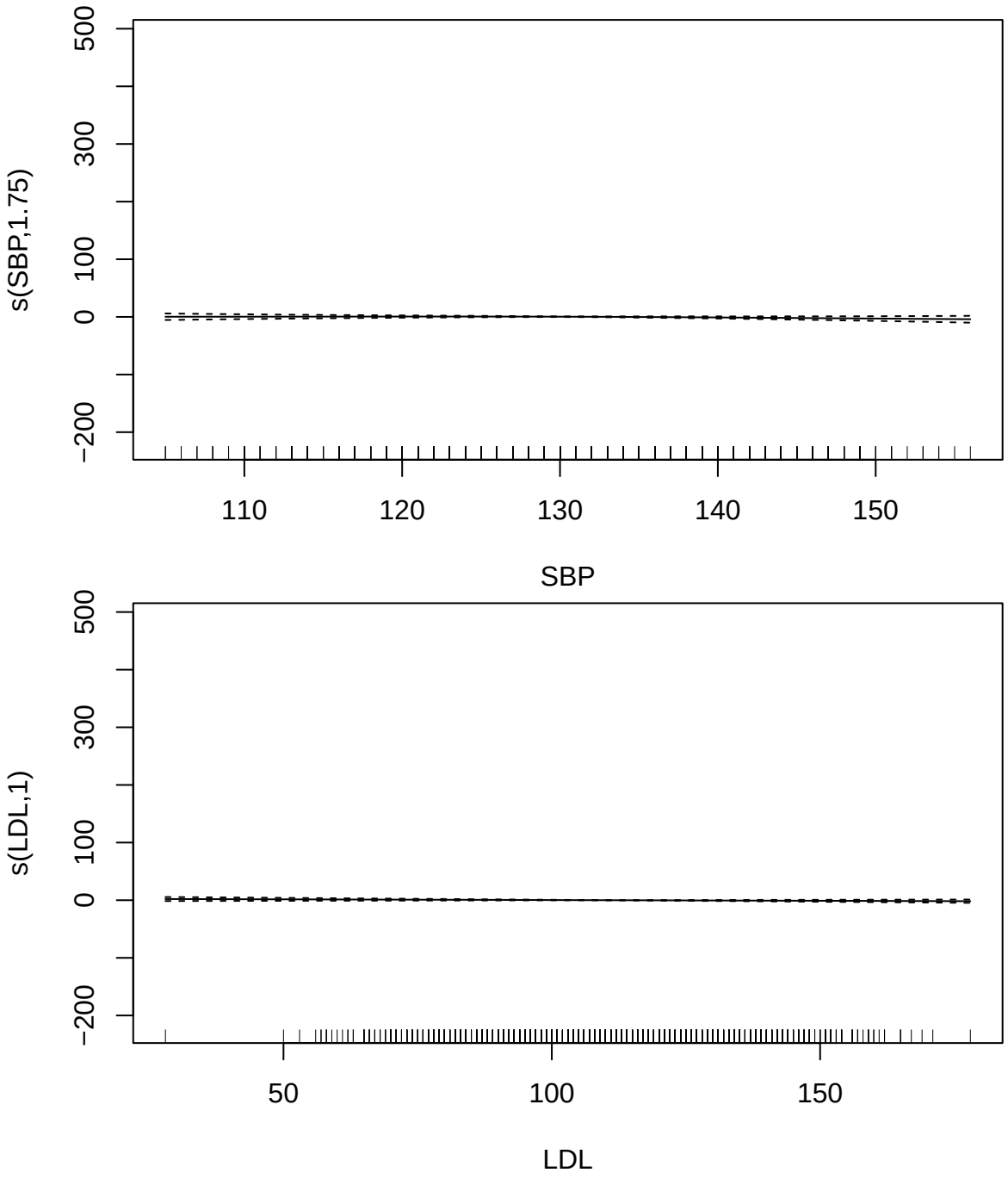
```
##     hypertension1 + diabetes1 + vaccine1 + severity1 + study1 +
##     s(age) + s(SBP) + s(LDL) + s(bmi) + s(height) + s(weight)
##
## Estimated degrees of freedom:
## 1.00 1.75 1.00 8.56 7.24 2.81  total = 34.36
##
## GCV score: 384.8296
```

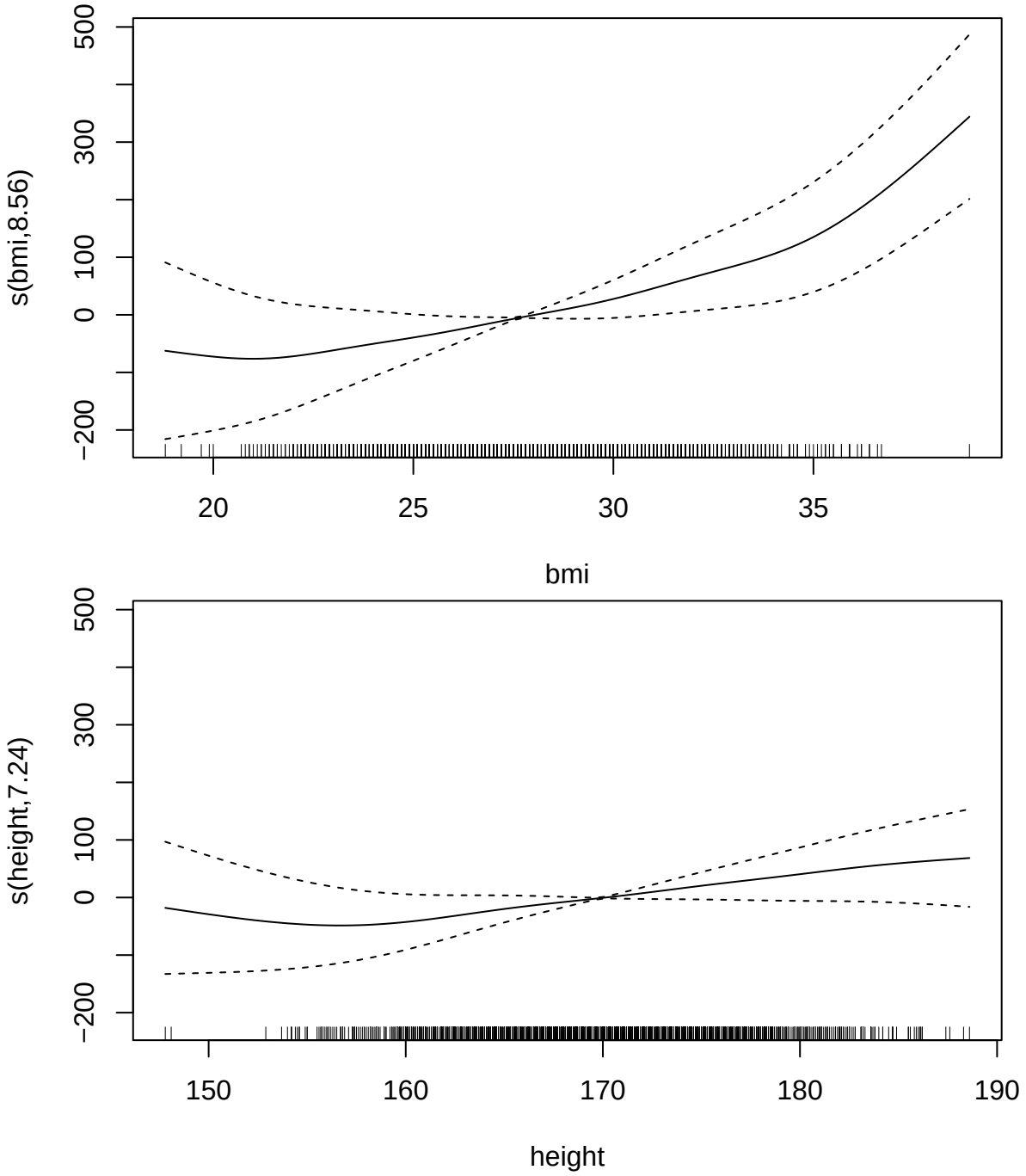
The GAM model includes all the predictors.

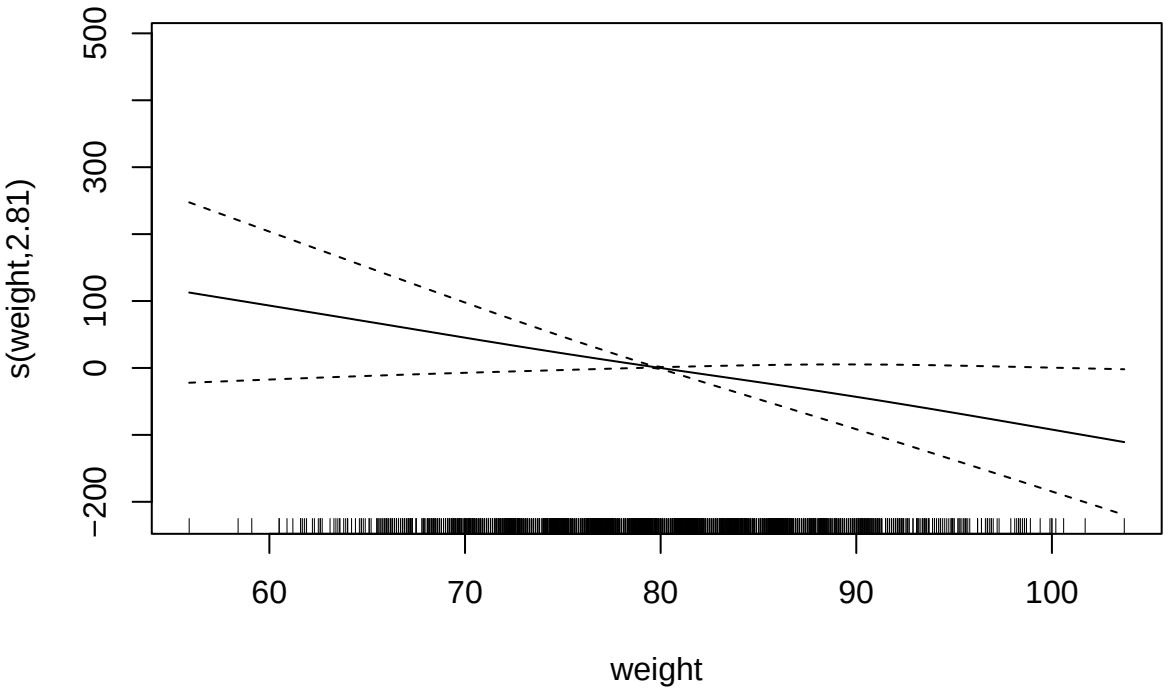
Generate plot for GAM

```
plot(gam.fit$finalModel)
```




s(SBP,1.75)
500
300
100
0
−200
110
120
130
140
150
SBP
s(LDL,1)
500
300
100
0
−200
50
100
150
LDL


s(bmi,8.56)
500
300
100
0
−200
20
25
30
35
bmi
s(height,7.24)
500
300
100
0
−200
150
160
170
180
190
height



Test error for GAM

```
gam.pred <- predict(gam.fit, newdata = x2)
gam.test.error <- mean((gam.pred - y2)^2)
gam.test.error
```

```
## [1] 272.0012
```

The test error for GAM is 272.0012.

Model comparison

```
resamp <- resamples(list(lasso = lasso.fit,
                         elastic_net = enet.fit,
                         pcr = pcr.fit,
                         MARS = mars.fit,
                         GAM = gam.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lasso, elastic_net, pcr, MARS, GAM
## Number of resamples: 100
##
## MAE
##                 Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso       11.92509 13.20141 13.61045 13.60346 14.03680 15.39327    0
## elastic_net 11.89773 13.15707 13.54537 13.56920 13.99818 15.38983    0
## pcr         12.04304 13.13403 13.80067 13.80337 14.41091 16.35188    0
## MARS        11.10058 11.96619 12.40141 12.45876 12.93784 14.07187    0
```

```
## GAM         11.57819 12.45809 12.98892 13.00159 13.52263 14.60480    0
##
## RMSE
##                   Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso       15.46600 18.53032 19.92715 20.75493 23.51240 30.77897    0
## elastic_net 15.42411 18.50613 19.94910 20.75059 23.54282 30.83453    0
## pcr         16.08147 18.75094 21.01156 22.23157 25.96932 34.12917    0
## MARS        14.77247 17.17671 18.85769 18.88053 20.27041 26.27734    0
## GAM         15.32431 18.14365 20.10992 20.36925 22.55963 29.53195    0
##
## Rsquared
##                    Min.    1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso       0.06726381 0.16474700 0.2213977 0.2401690 0.3198366 0.4749550    0
## elastic_net 0.06776640 0.16561439 0.2217672 0.2398895 0.3187296 0.4719851    0
## pcr         0.03090682 0.09335809 0.1265943 0.1251093 0.1508245 0.2390132    0
## MARS        0.03916963 0.21410778 0.3033209 0.3541771 0.5170313 0.6810754    0
## GAM         0.05260116 0.17792496 0.2566156 0.2855890 0.3909223 0.5621205    0
```

```
bwplot(resamp, metric = "RMSE")
```