Midterm Result

Yiying Wu (yw3996)

```
## R packages
library(tidyverse)
library(corrplot)
library(caret)
library(tidymodels)
library(gtsummary)
library(pls)
library(patchwork)
```

```
## Input dataset
load("./recovery.RData")
```

Data preparation

```
dat <- subset(dat, select = -id)
dat$study <- ifelse(dat$study == "A", 0, 1)
dat <- dat %>%
  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)
    )
dat <- na.omit(dat)
```

Exploratory analysis and data visualization

In this section, use appropriate visualization techniques to explore the dataset and identify any patterns or relationships in the data.

Summary statistics

```
summ_dat<-dat%>%
  mutate(
    gender = factor(case_when(
```

```
      gender == "1" ~ "female",
      gender == "0" ~ "male"),
      levels = c("male", "female")
    ),
    race= factor(case_when(
      race == "1" ~ "White",
      race == "2" ~ "Asian",
      race == "3" ~ "Black",
      race == "4" ~ "Hispanic"),
      levels = c("White", "Asian","Black","Hispanic")
    ),
    smoking=factor(case_when(
      smoking == "0" ~ "Never smoked",
      smoking == "1" ~ "Former smoker",
      smoking == "2" ~ "Current smoker"),
      levels = c("Never smoked", "Former smoker","Current smoker")
    ),
    hypertension=factor(case_when(
      hypertension == "0" ~ "No",
      hypertension == "1" ~ "Yes"),
      levels = c("No", "Yes")
    ),
    diabetes=factor(case_when(
      diabetes == "0" ~ "No",
      diabetes == "1" ~ "Yes"),
      levels = c("No", "Yes")
    ),
    vaccine=factor(case_when(
      vaccine == "0" ~ "Not vaccinated",
      vaccine == "1" ~ "Vaccinated"),
      levels = c("Not vaccinated", "Vaccinated")
    ),
    severity=factor(case_when(
      severity == "0" ~ "Not severe",
      severity == "1" ~ "Severe"),
      levels = c("Not severe", "Severe")
    )
    )

summ_dat %>%
  tbl_summary() %>%
  bold_labels()%>%
  as_gt(include = everything()) %>%
  gt::tab_header("Table 1: Summary of Dataset")
```

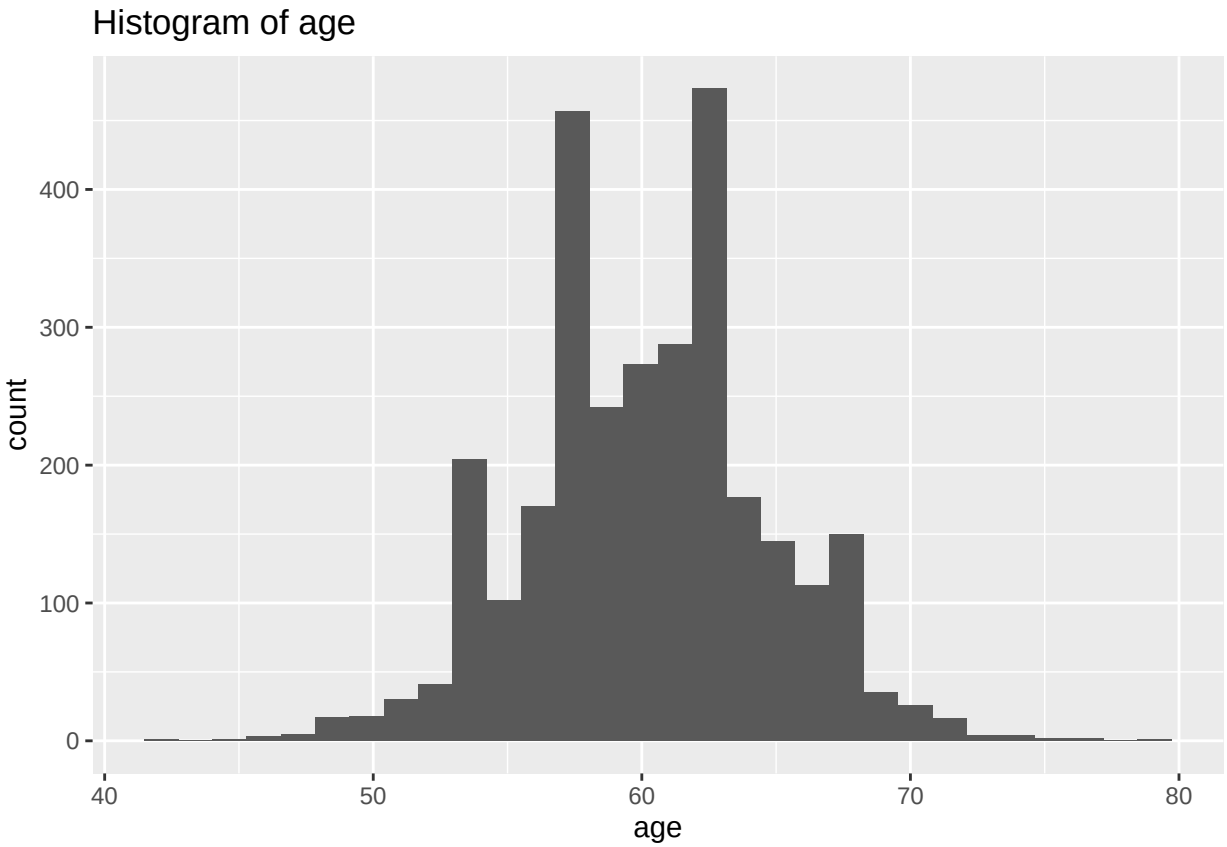
Table 1: Summary of Dataset

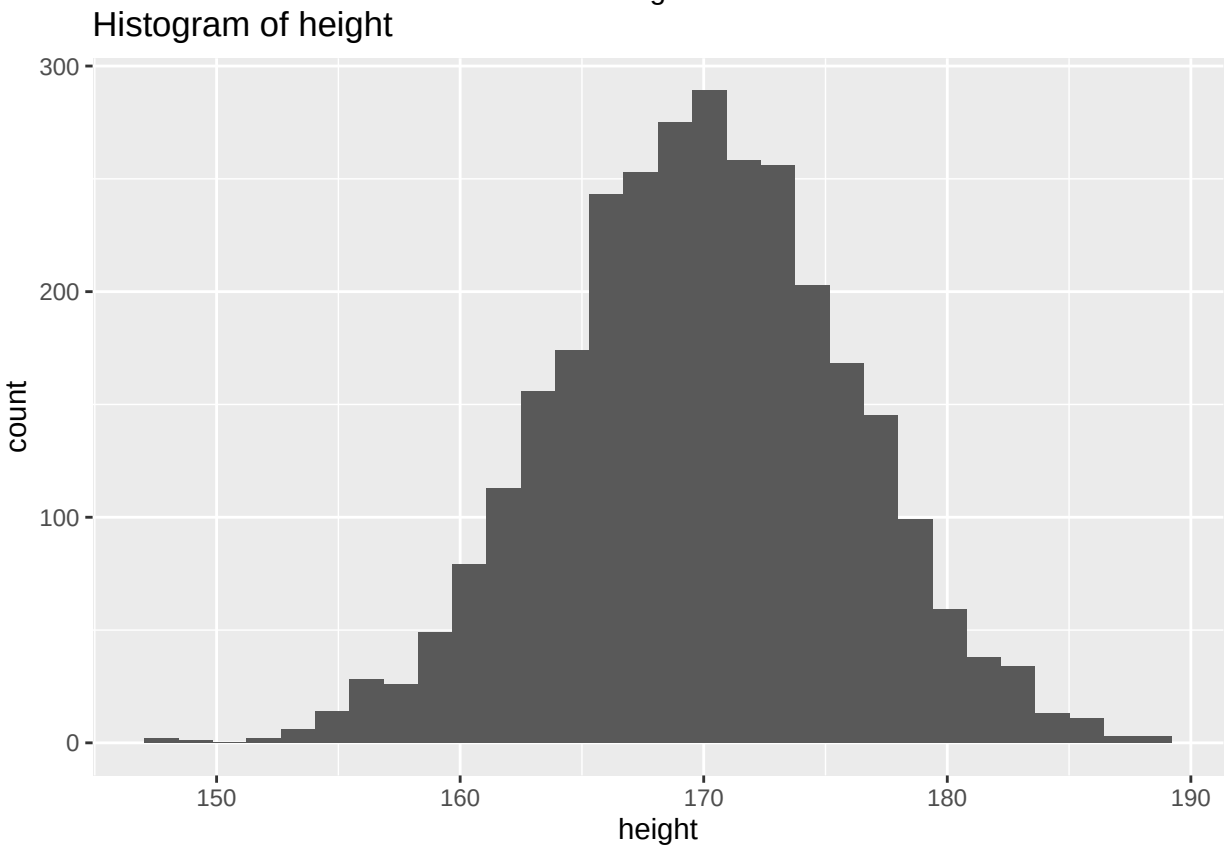
Characteristic	**N = 3,000**[1]
age	60.0 (57.0, 63.0)
gender	
male	1,544 (51%)
female	1,456 (49%)

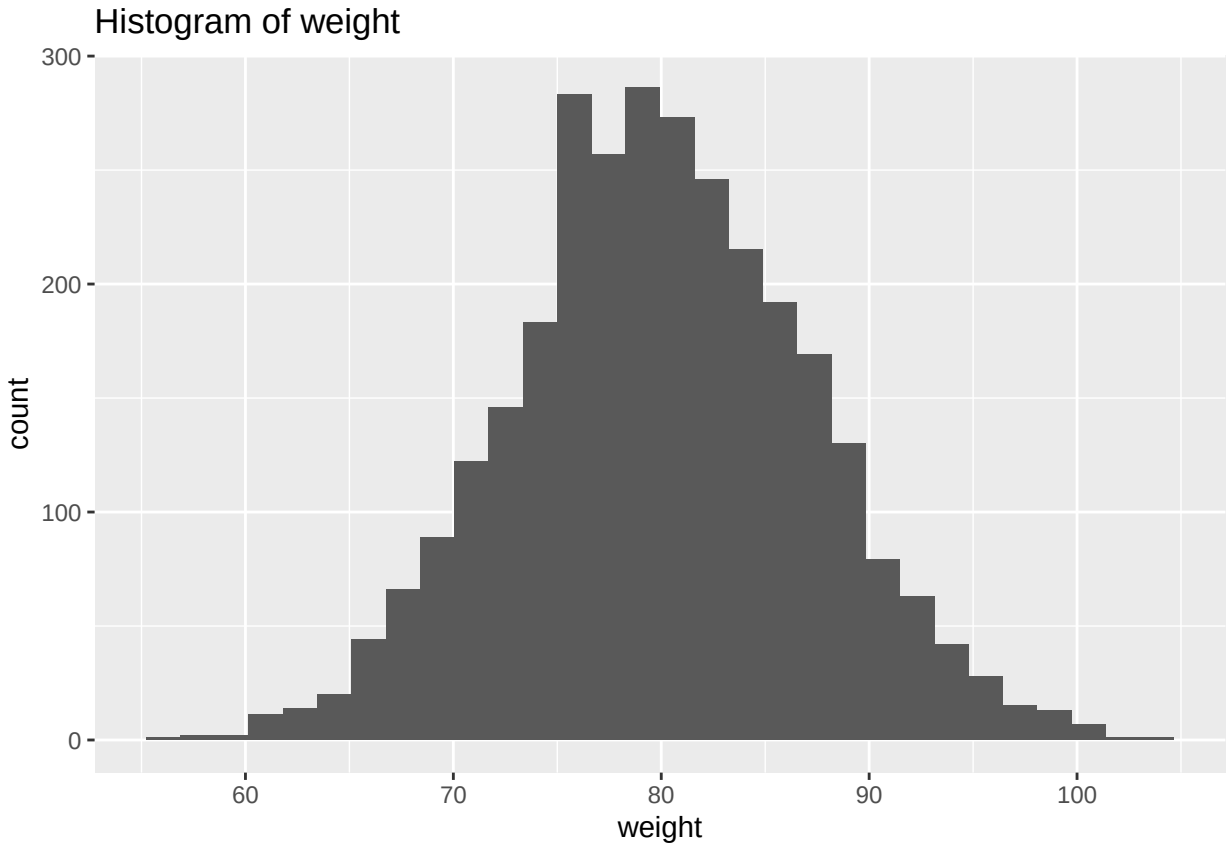
race	
White	1,967 (66%)
Asian	158 (5.3%)
Black	604 (20%)
Hispanic	271 (9.0%)
smoking	
Never smoked	1,822 (61%)
Former smoker	859 (29%)
Current smoker	319 (11%)
height	169.9 (166.0, 173.9)
weight	80 (75, 85)
bmi	27.65 (25.80, 29.50)
hypertension	1,492 (50%)
diabetes	463 (15%)
SBP	130 (125, 136)
LDL	110 (97, 124)
vaccine	
Not vaccinated	1,212 (40%)
Vaccinated	1,788 (60%)
severity	
Not severe	2,679 (89%)
Severe	321 (11%)
study	
0	2,000 (67%)
1	1,000 (33%)
recovery_time	39 (31, 49)

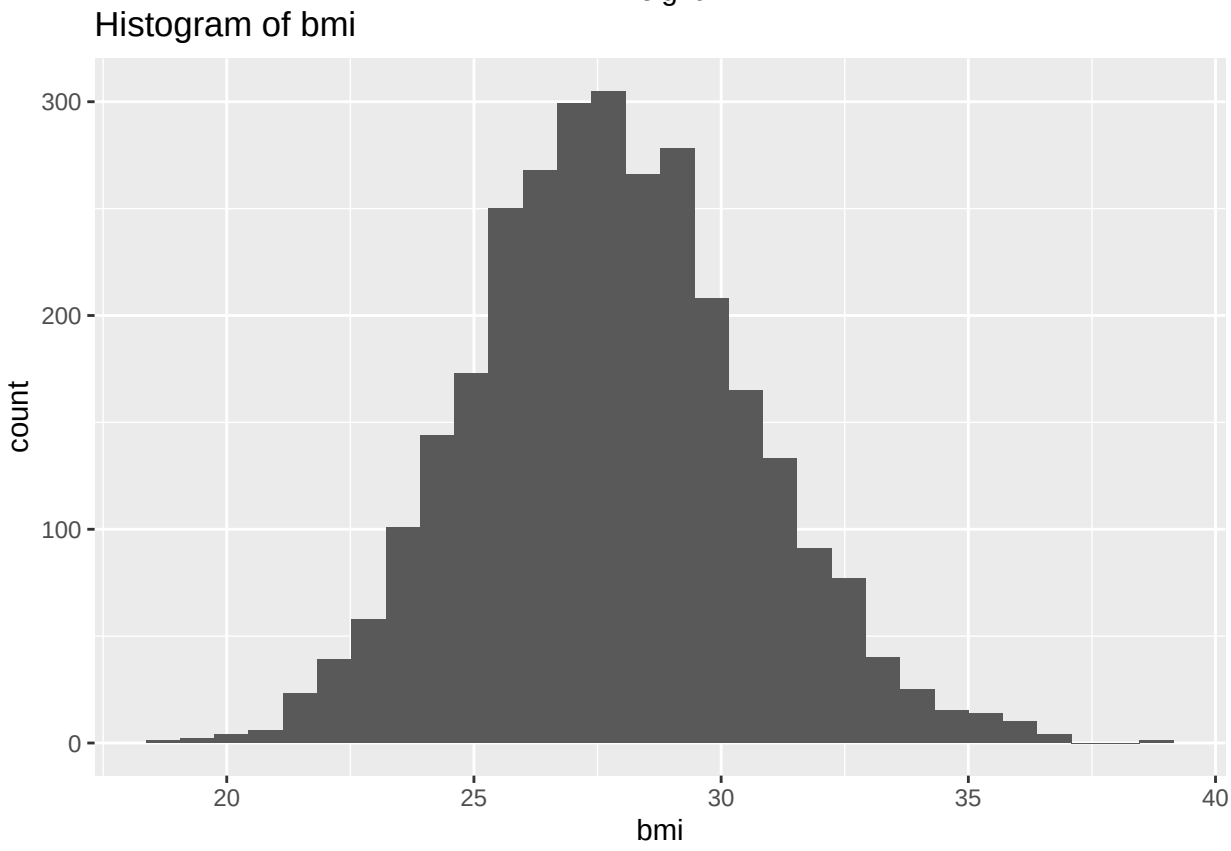
[1] Median (IQR); n (%)

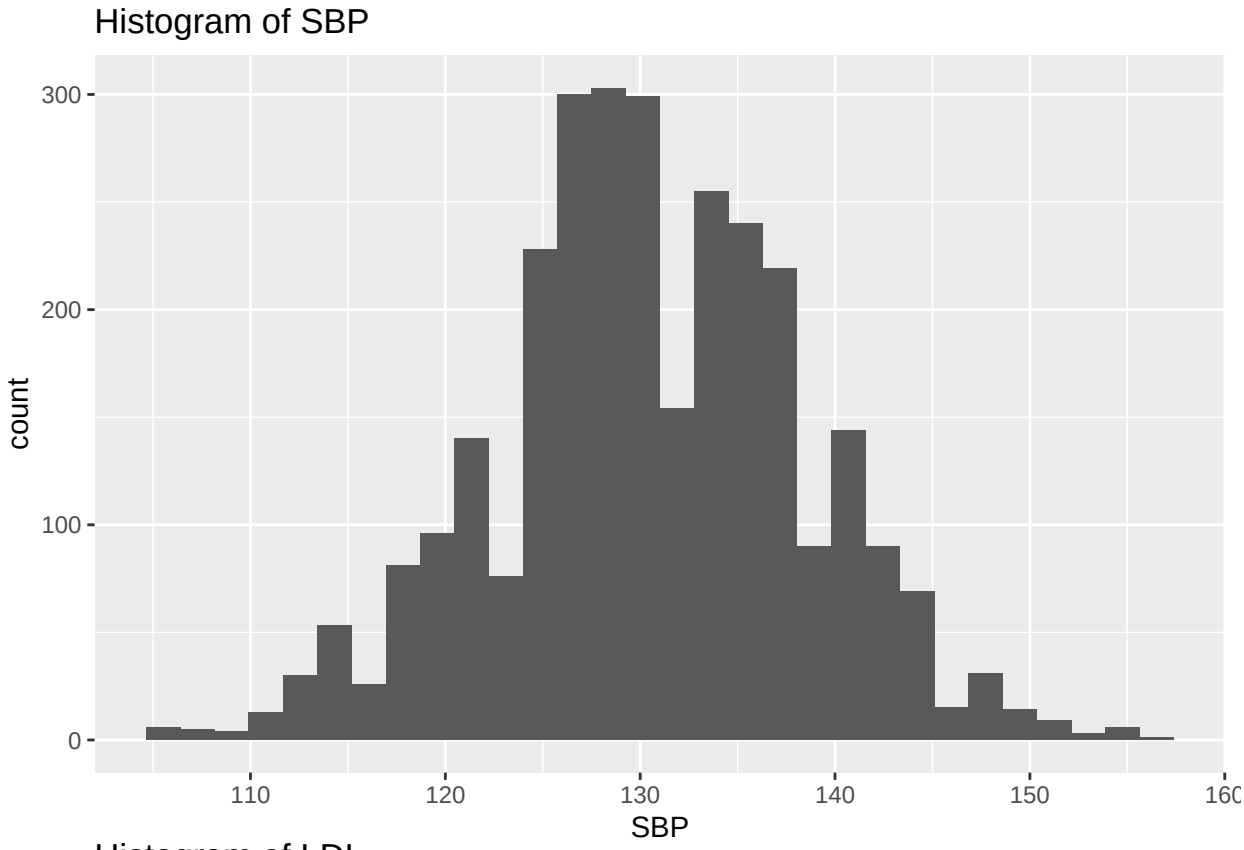
Histograms for the numerical variables

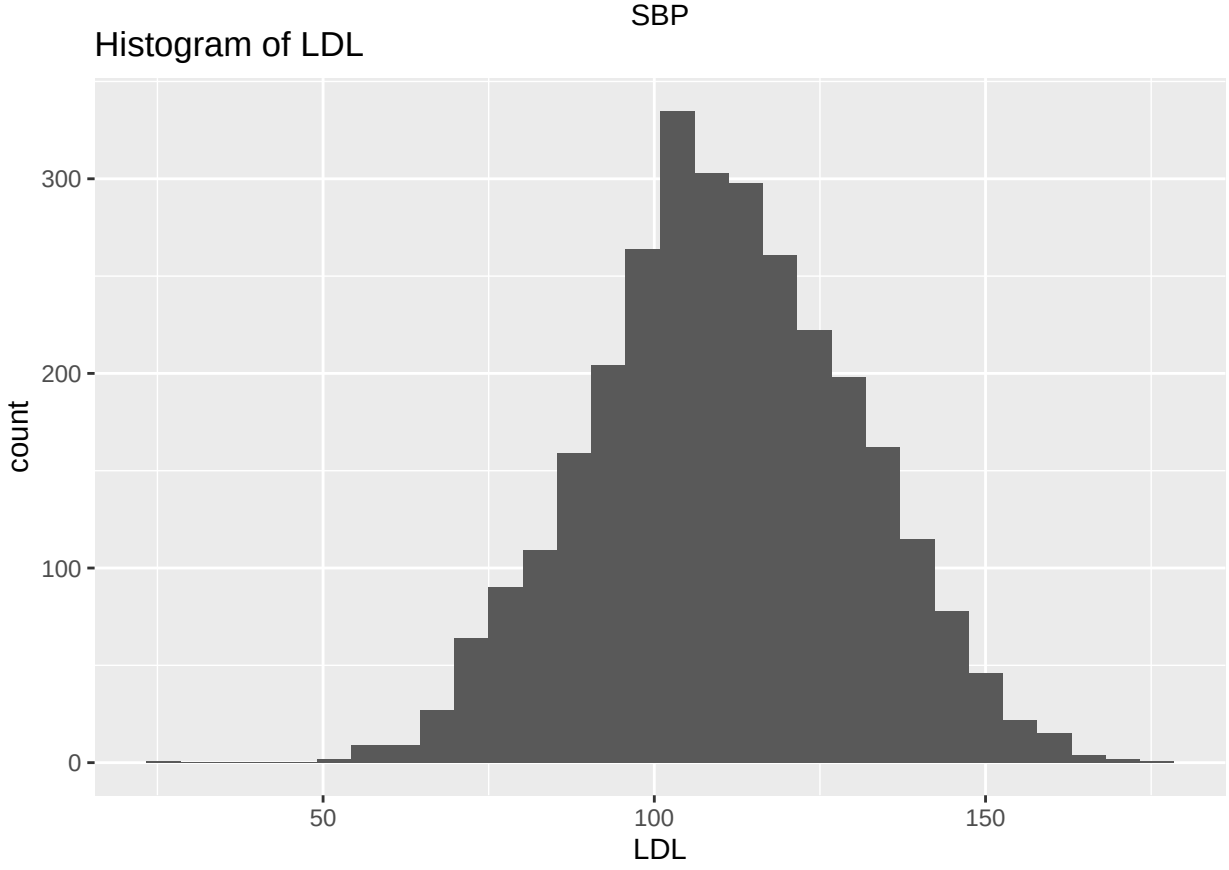


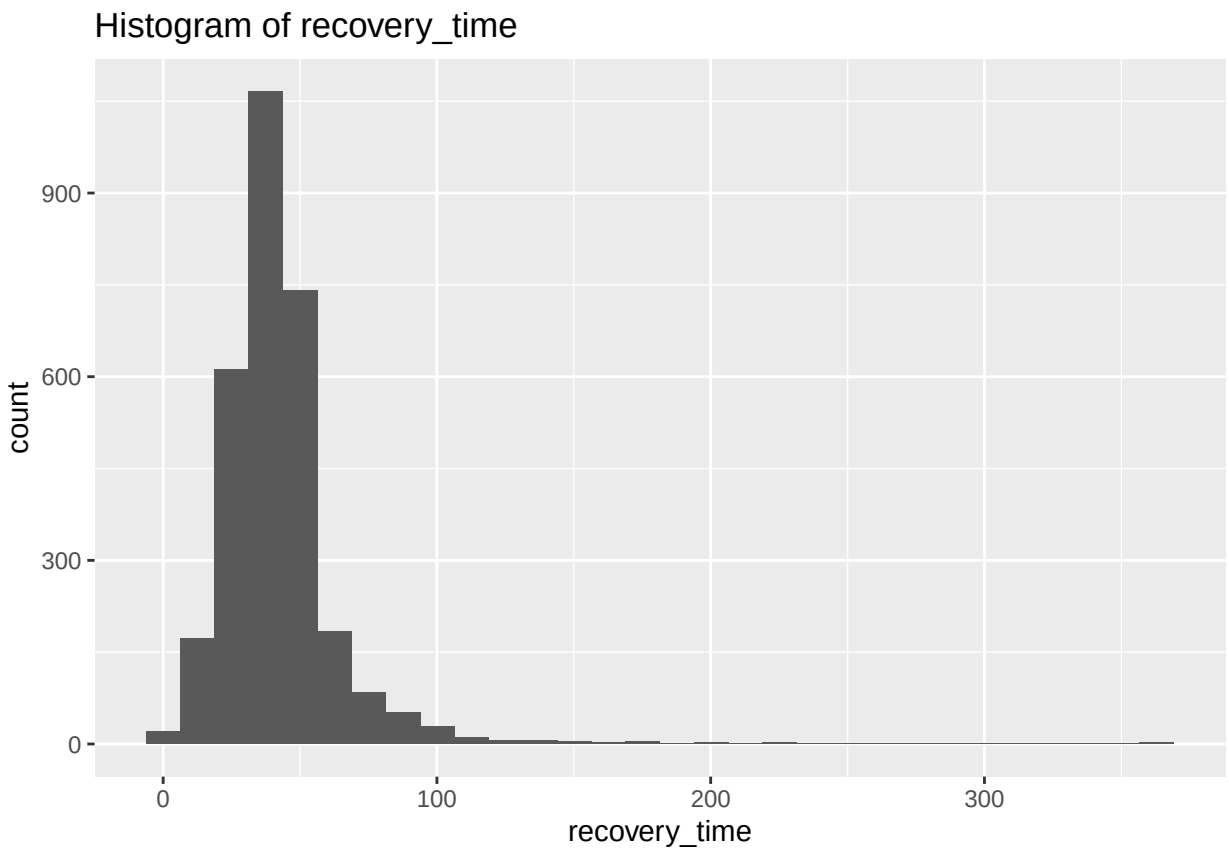



Histogram of weight
count
weight
300
200
100
0
60
70
80
90
100


Histogram of bmi
count
bmi
300
200
100
0
20
25
30
35
40


Histogram of SBP
count
300
200
100
0
110
120
130
140
150
160
SBP


Histogram of LDL
count
300
200
100
0
50
100
150
LDL



Feature plot for continuous variables

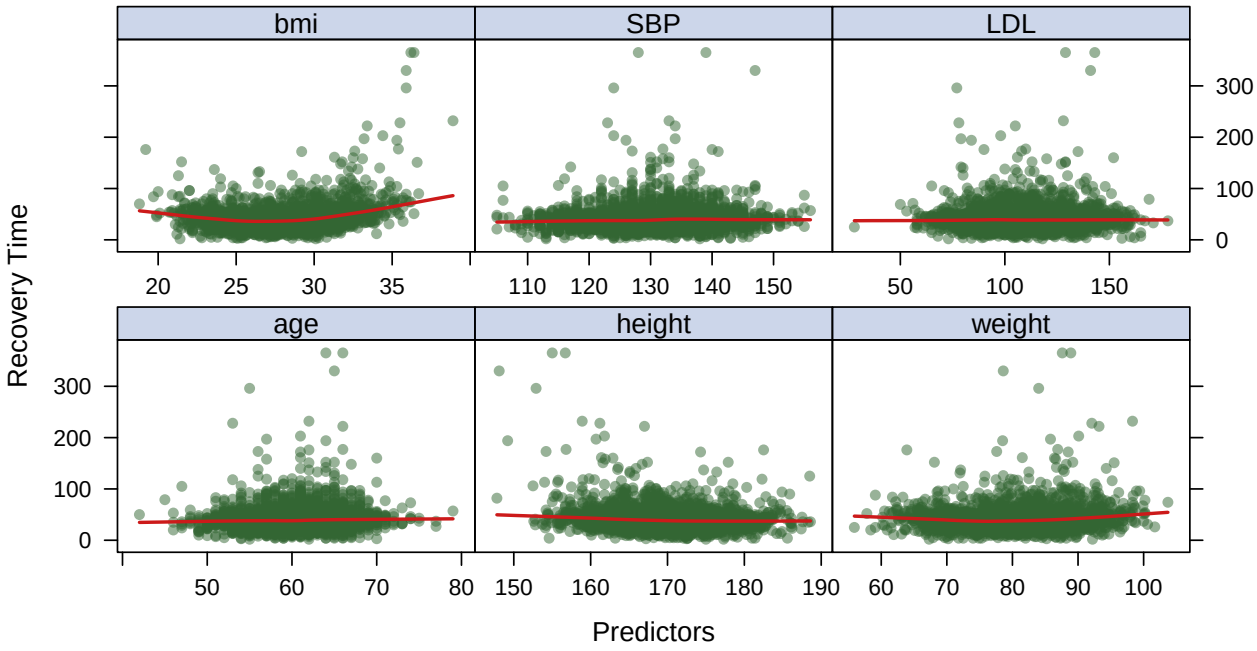
```
theme1 <- trellis.par.get()
theme1$plot.symbol$col <- rgb(.2, .4, .2, .5)
theme1$plot.symbol$pch <- 16
theme1$plot.line$col <- rgb(.8, .1, .1, 1)
theme1$plot.line$lwd <- 2
theme1$strip.background$col <- rgb(.0, .2, .6, .2)
trellis.par.set(theme1)

continuousData <- dat%>%
  dplyr::select('age', 'height', 'weight', 'bmi', 'SBP', 'LDL', 'recovery_time')

# Set up the plotting device to save the output to a file
png(filename = "./result_files/featurePlot.png", width = 8, height = 8, units = 'in', res = 300)

featurePlot(x = continuousData[ , 1:6],
            y = continuousData[ , 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))
```

```
# Turn off the plotting device
dev.off()
```

```
## pdf
##   2
```

```
featurePlot(x = continuousData[ , 1:6],
            y = continuousData[ , 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))
```

Figure 1:Relationship between Continuous Predictors and Recovery Time

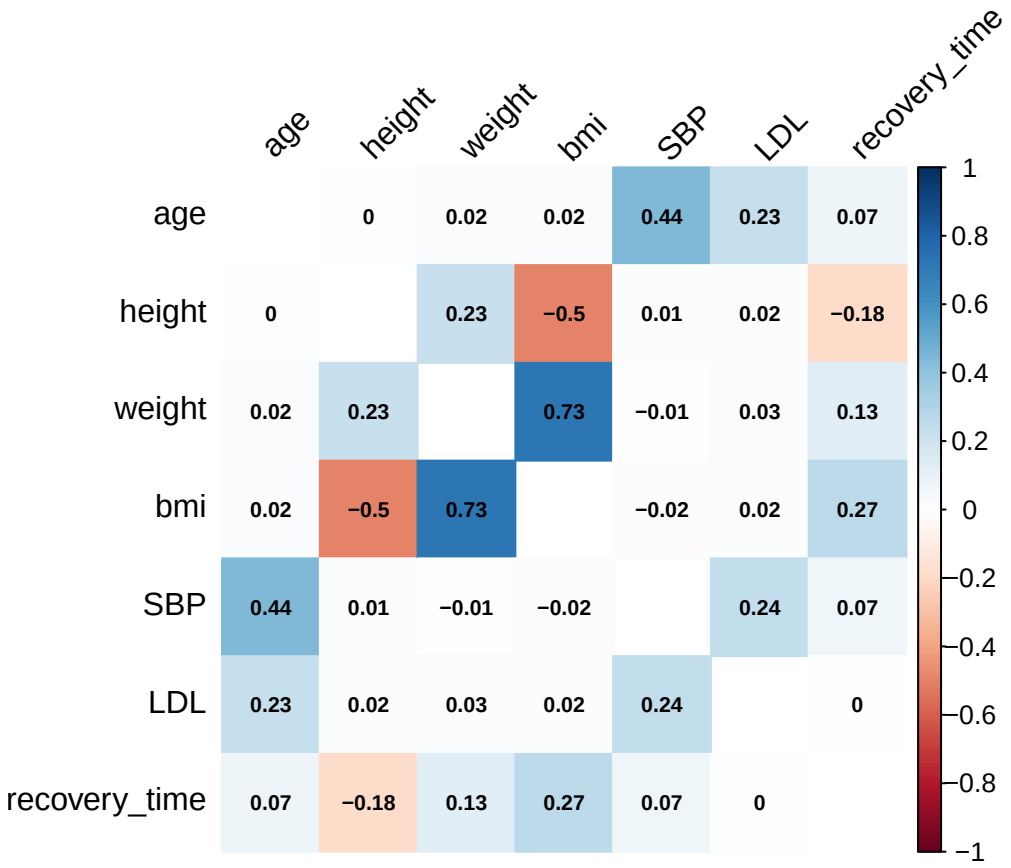


correlation plot

```
numerical_data <- summ_dat[, sapply(summ_dat, is.numeric)]
correlation_matrix <- cor(numerical_data)

# Set up the plotting device to save the output to a file
png(filename = "./result_files/cor_plot.png", width = 8, height = 8, units = 'in', res = 300)

# Create the correlation plot
corrplot::corrplot(correlation_matrix, method = "color", addCoef.col = "black",
                   tl.col = "black", tl.srt = 45, insig = "blank", number.cex = 0.7, diag = FALSE)

# Turn off the plotting device
dev.off()
```

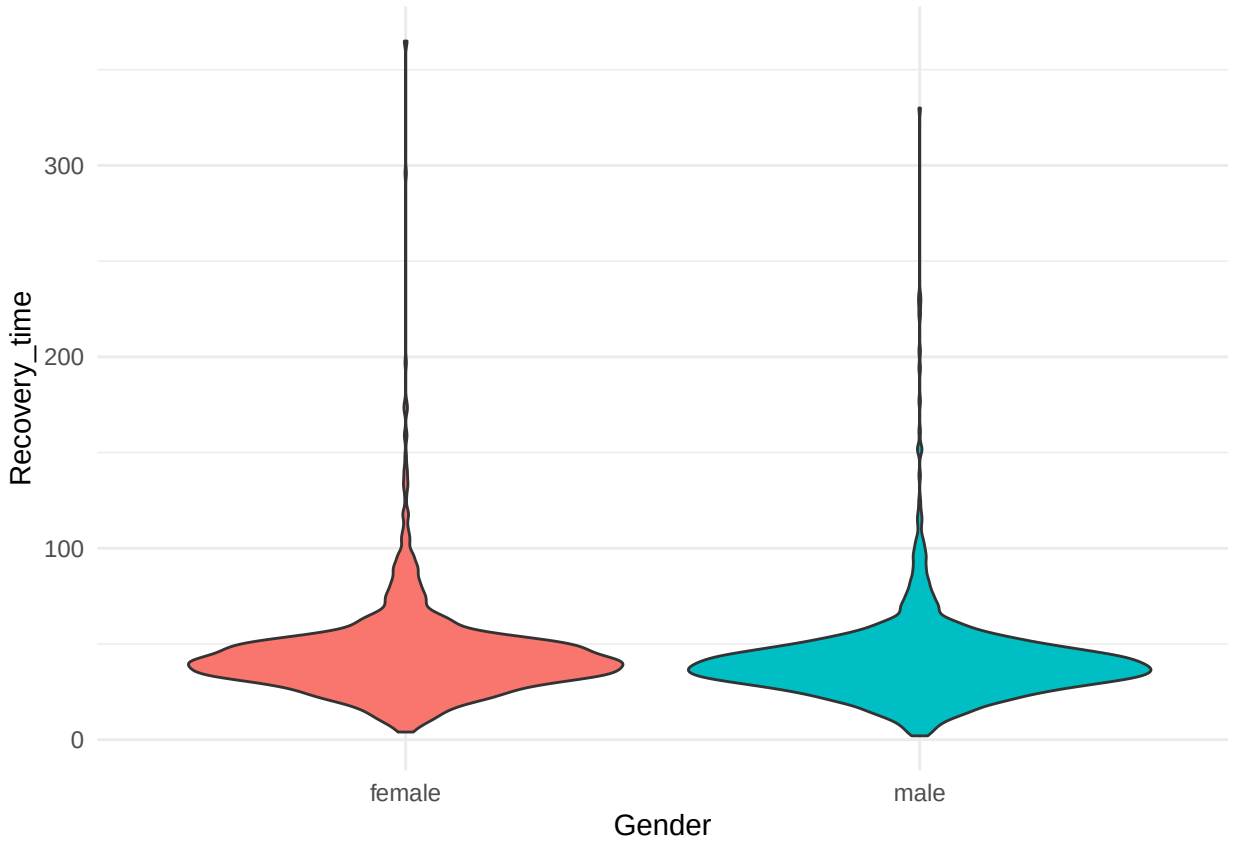
```
## pdf
##   2
```

```
corrplot::corrplot(correlation_matrix, method = "color", addCoef.col = "black",
                   tl.col = "black", tl.srt = 45, insig = "blank", number.cex = 0.7, diag = FALSE)
```

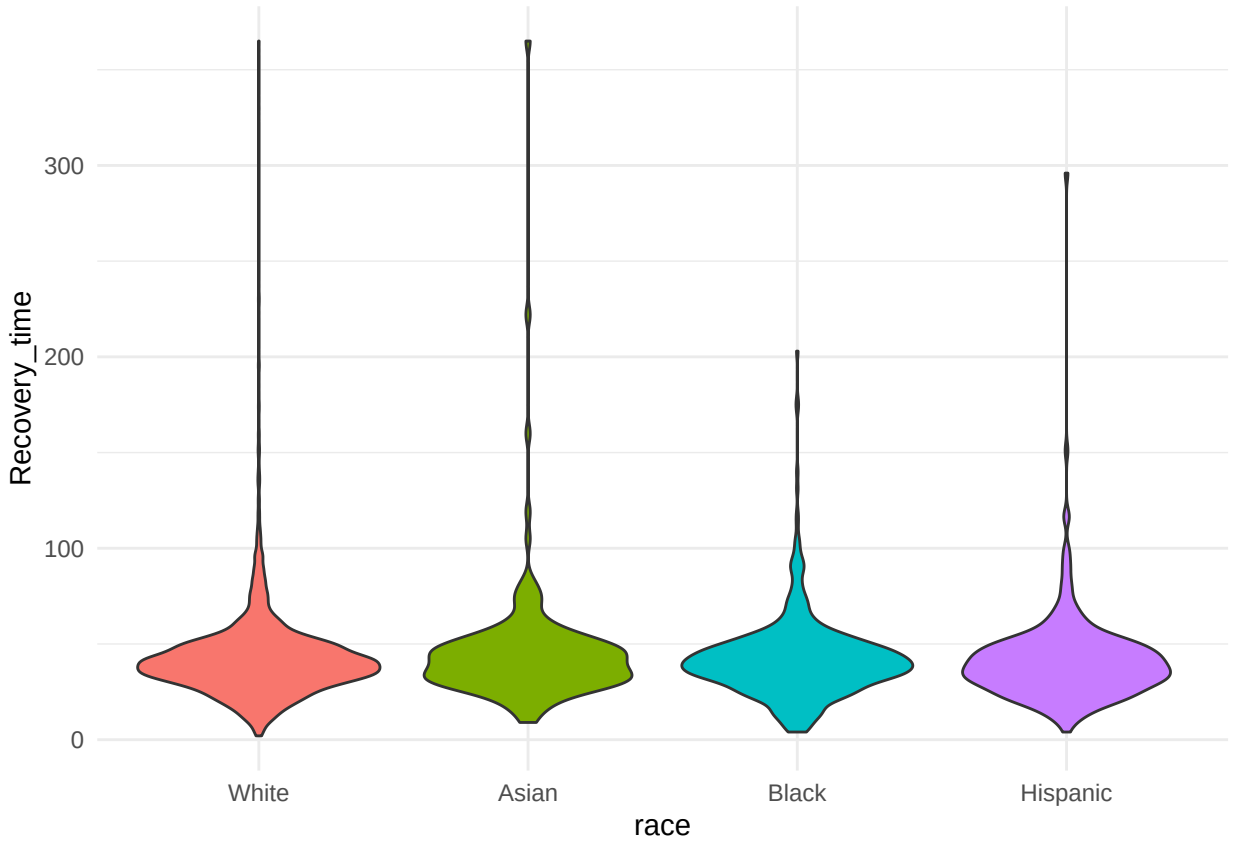


Violin plot for categorical variables

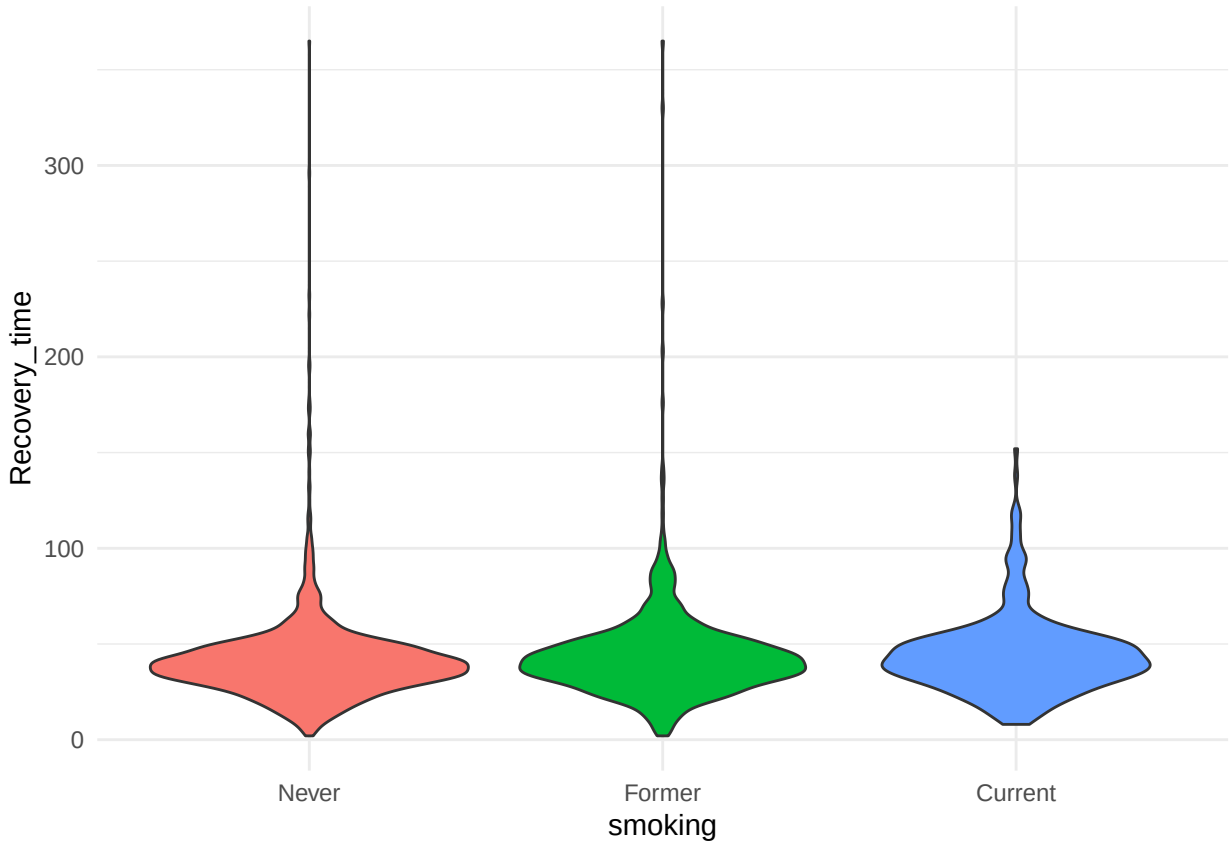
```
gender_plot <- dat %>%
  ggplot(aes(x = gender, y = recovery_time, fill = gender))+
  geom_violin() +
  scale_x_discrete(labels = c("female", "male")) +
  labs(
       x = "Gender",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
gender_plot
```



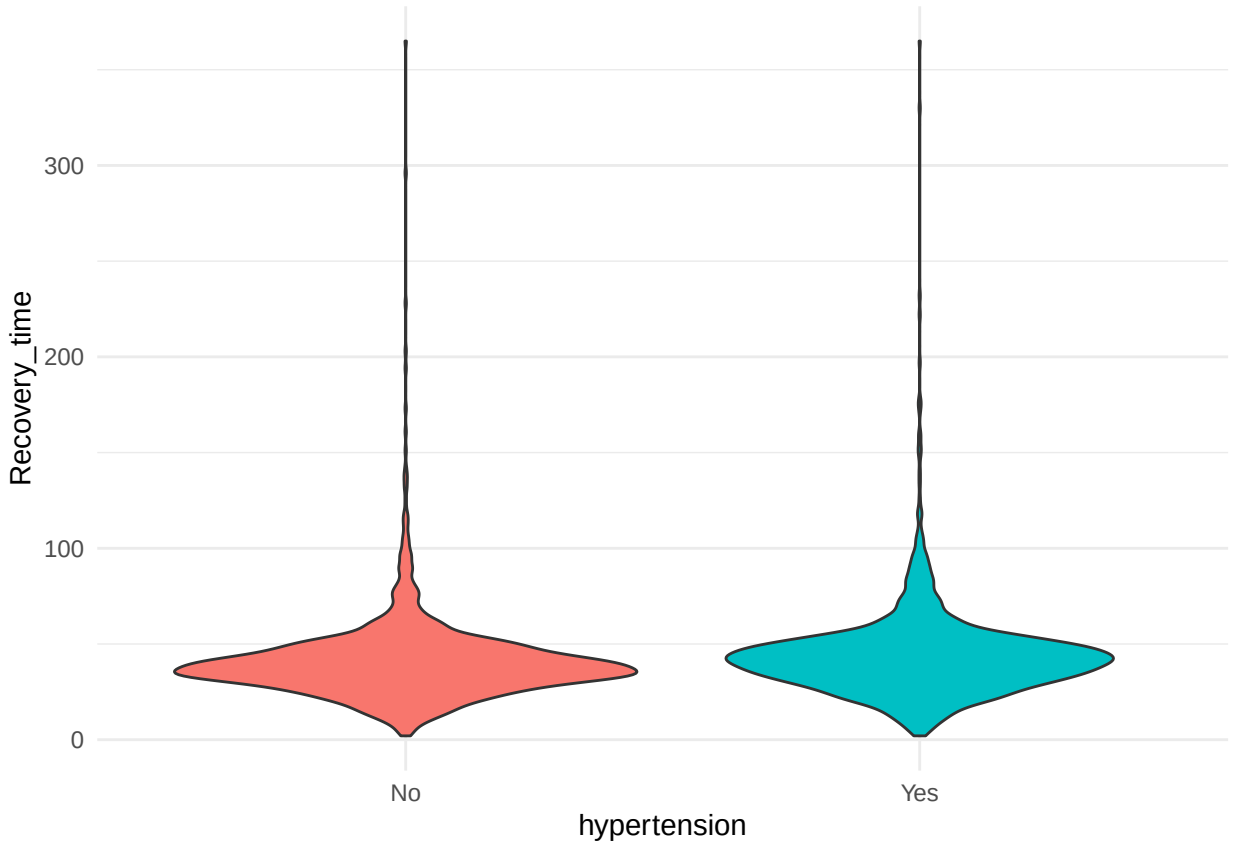

```
race_plot <- dat %>%
  ggplot(aes(x = race, y = recovery_time, fill = race)) +
  geom_violin() +
  scale_x_discrete(labels = c("White", "Asian", "Black", "Hispanic")) +
  labs(
       x = "race",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
race_plot
```



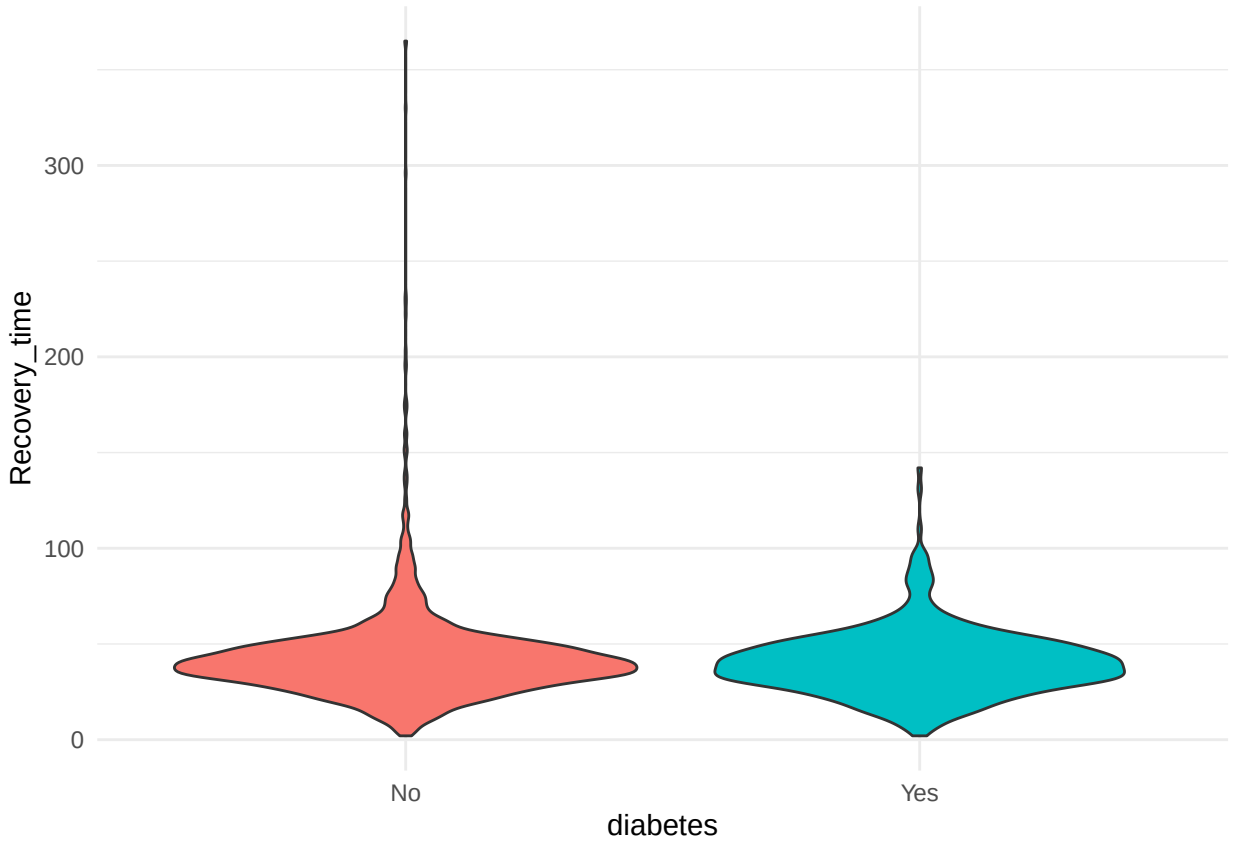

```
smoking_plot <- dat %>%
  ggplot(aes(x = smoking, y = recovery_time, fill = smoking)) +
  geom_violin() +
  scale_x_discrete(labels = c("Never", "Former", "Current")) +
  labs(
       x = "smoking",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
smoking_plot
```



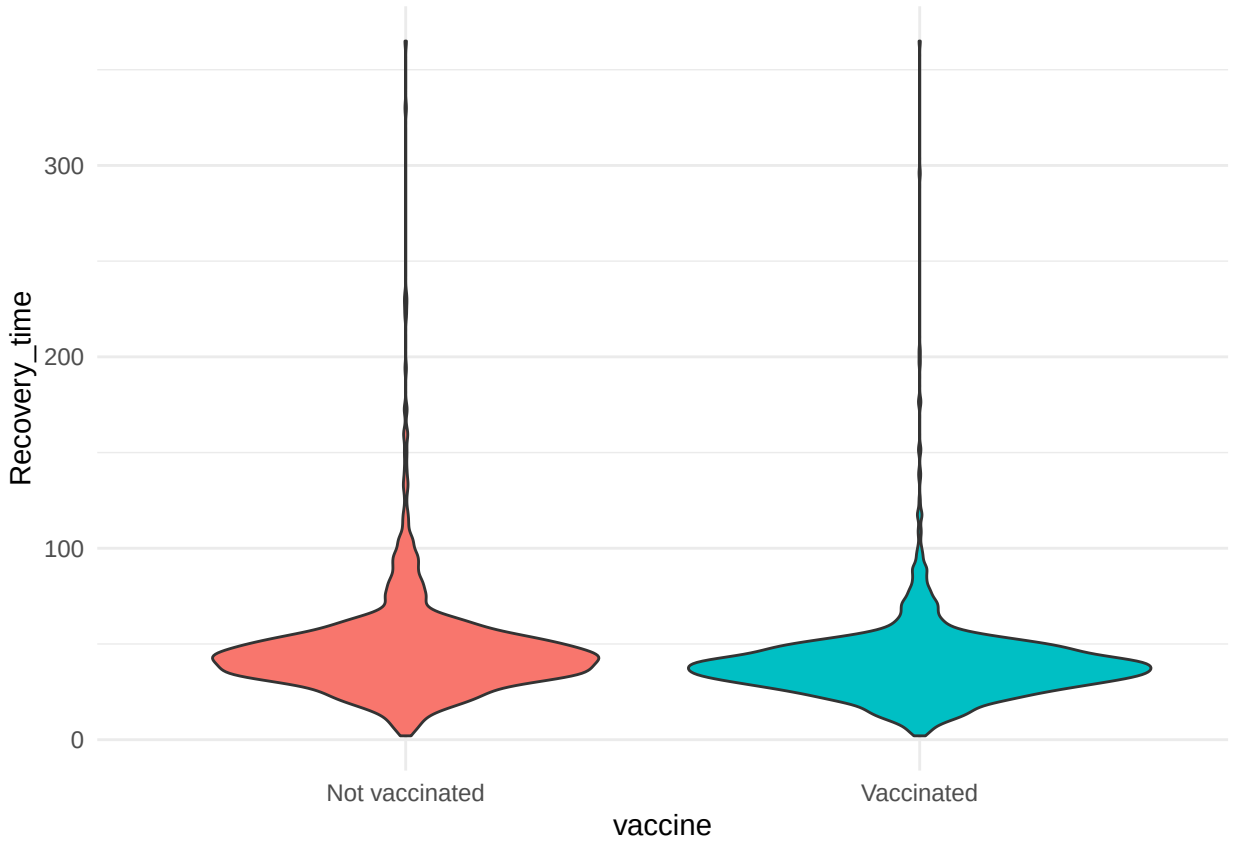

```
hypertension_plot <- dat %>%
  ggplot(aes(x = hypertension, y = recovery_time, fill = hypertension)) +
  geom_violin() +
  scale_x_discrete(labels = c("No", "Yes")) +
  labs(
       x = "hypertension",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
hypertension_plot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



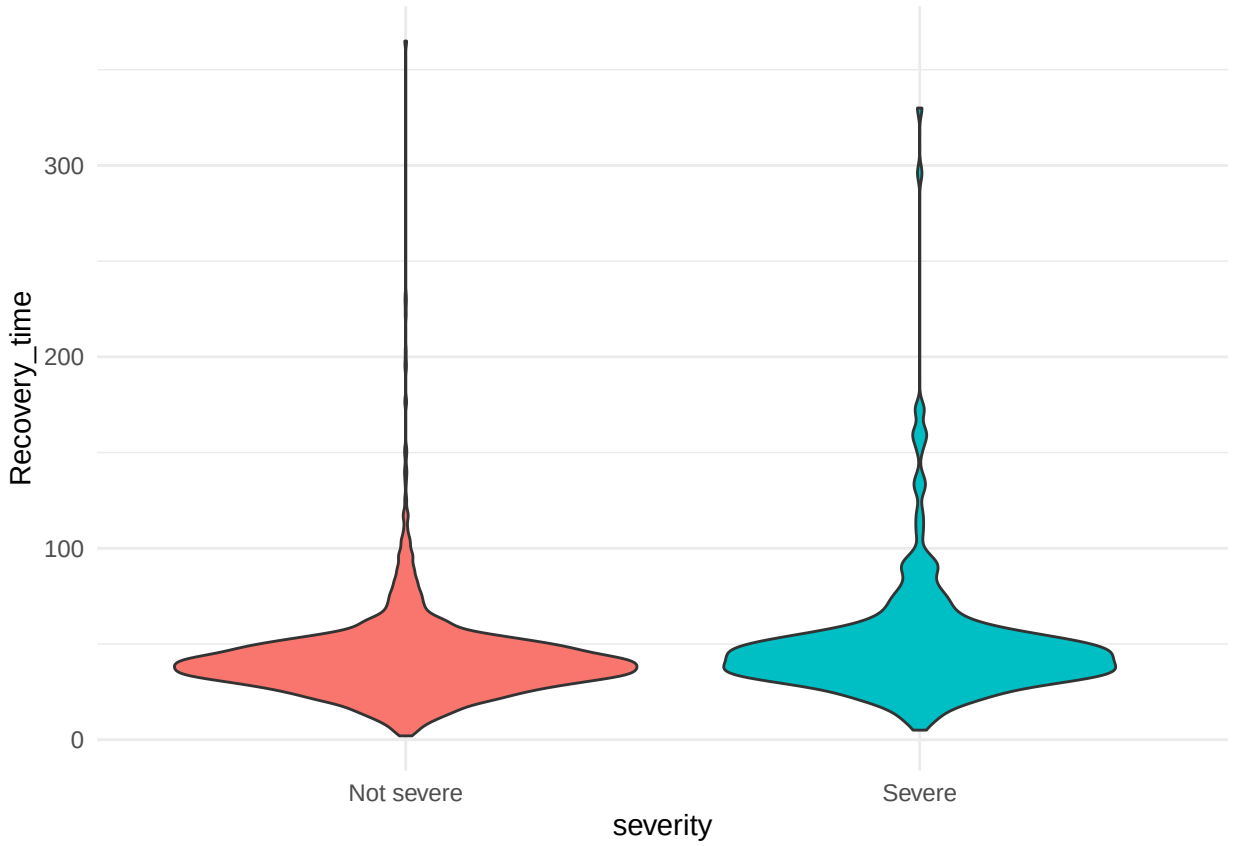

```
diabetes_plot <- dat %>%
  ggplot(aes(x = diabetes, y = recovery_time, fill = diabetes)) +
  geom_violin() +
  scale_x_discrete(labels = c("No", "Yes")) +
  labs(
       x = "diabetes",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
diabetes_plot
```



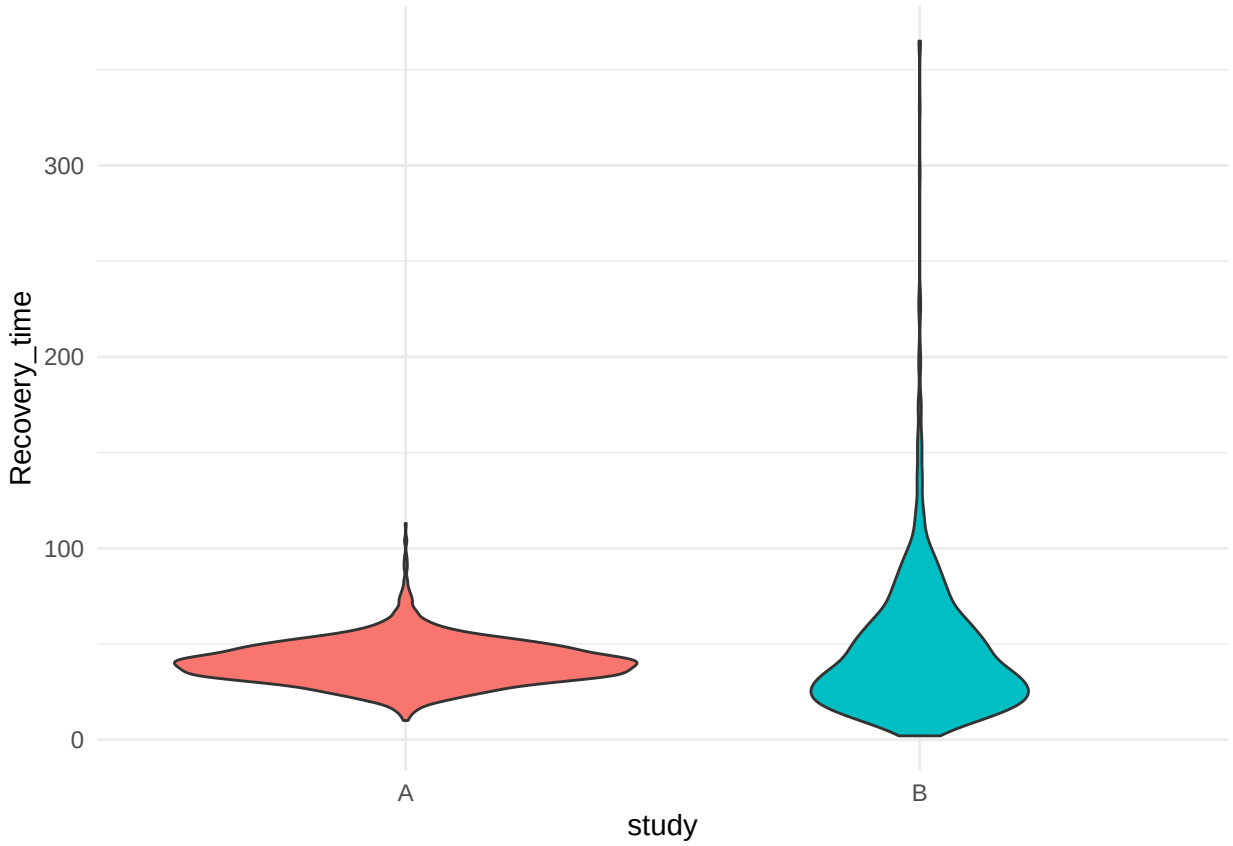

```
vaccine_plot <- dat %>%
   ggplot(aes(x = vaccine, y = recovery_time, fill = vaccine)) +
   geom_violin() +
   scale_x_discrete(labels = c("Not vaccinated", "Vaccinated")) +
   labs(
        x = "vaccine",
        y = "Recovery_time") +
   theme_minimal() + theme(legend.position = "none")
vaccine_plot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




```
severity_plot <- dat %>%
  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
```

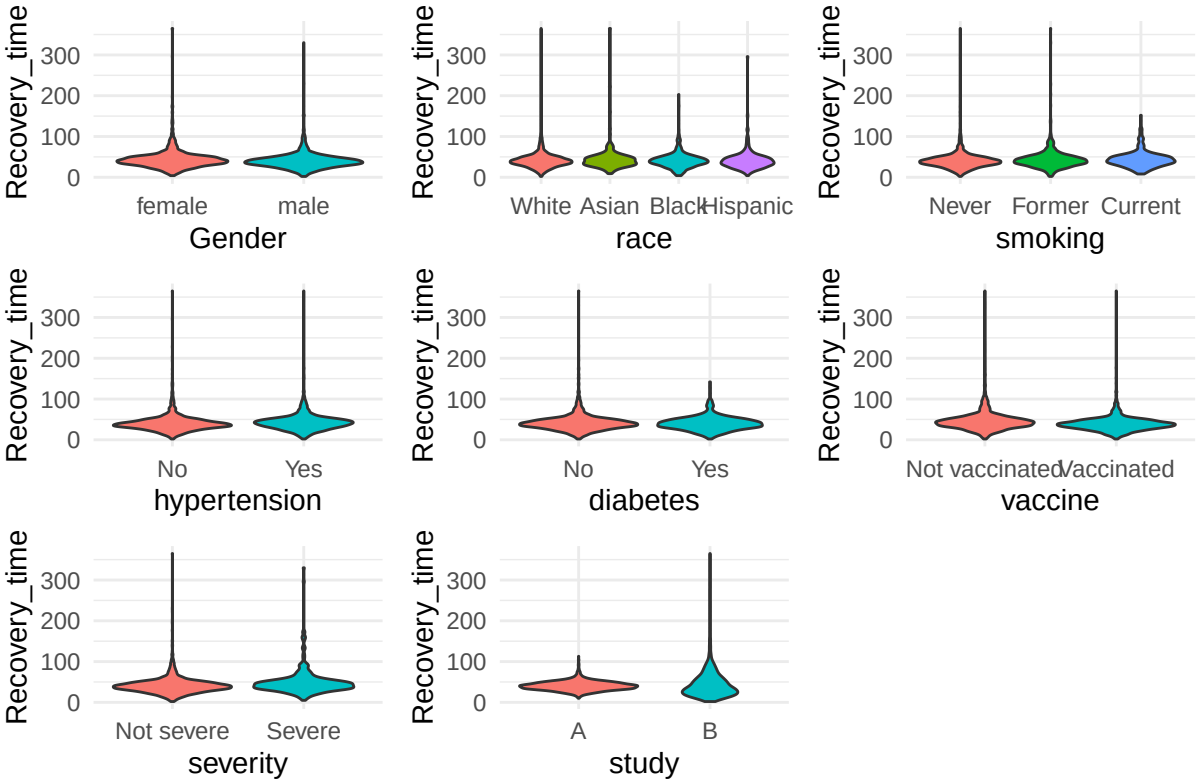



```
study_plot <- dat %>%
  ggplot(aes(x = study, y = recovery_time, fill = study)) +
  geom_violin() +
  scale_x_discrete(labels = c("A", "B")) +
  labs(
       x = "study",
       y = "Recovery_time") +
  theme_minimal() + theme(legend.position = "none")
study_plot
```




```
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
```

Figure 2: Relationship between Categorical Predictors and Recovery Time



Model training

In this section, describe the models you used to predict the time to recovery from COVID-19. Briefly state the assumptions made by using the models. Provide a detailed description of the model training procedure and how you obtained the final model.

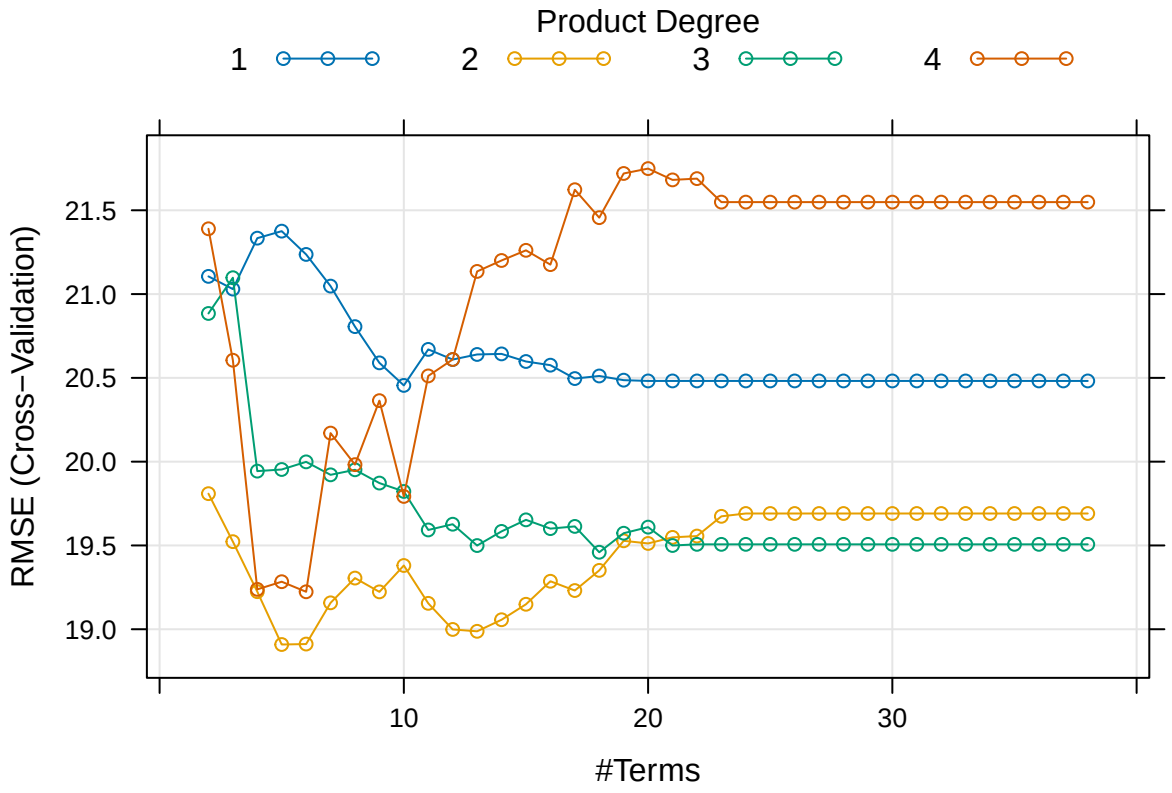
Outcome: recovery_time

Partition the dataset into two parts: training data (80%) and test data (20%).

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

# Extract the training and test data
training_data <- training(data_split)
x_train <- model.matrix(recovery_time ~ ., training_data) [ ,-1]
y_train <- training_data$recovery_time

testing_data <- testing(data_split)
x_test <- model.matrix(recovery_time ~ ., testing_data) [ ,-1]
y_test <- testing_data$recovery_time

# ctrl
ctrl <- trainControl(method = "cv", number = 10)
```

Multivariate Adaptive Regression Spline (MARS) Model

```
set.seed(666)
model.mars <- train(x = x_train,
                    y = y_train,
                    method = "earth", # earth is for mars
                    tuneGrid = expand.grid(degree = 1:4,
                                           nprune = 2:38),
                    trControl = ctrl)

# degree from 1~4 is sufficient
# nprune can be larger than the number of predictors, make it as large as possible

plot(model.mars)
```




```
# both number of terms and product degree are upper bounds

# best tune
model.mars$bestTune
```

```
##    nprune degree
## 41      5      2
```

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

```
##                (Intercept)                  h(31-bmi)
```

```
##                 -3.1983530                   6.3999877
##         h(bmi-31) * study1                h(bmi-25.2)
##                 25.6820131                   7.9260754
## h(weight-86.4) * h(bmi-31)
##                 -0.6277843
```

test error

```
mars.pred <- predict(model.mars, newdata = x_test)
test_error_mars <- mean((mars.pred - y_test)^2)
test_error_mars
```

```
## [1] 279.0367
```

```
RMSE_mars <- sqrt(test_error_mars)
RMSE_mars
```

```
## [1] 16.70439
```

The MSE of MARS model is 279.037.

Generalized Additive Model (GAM)

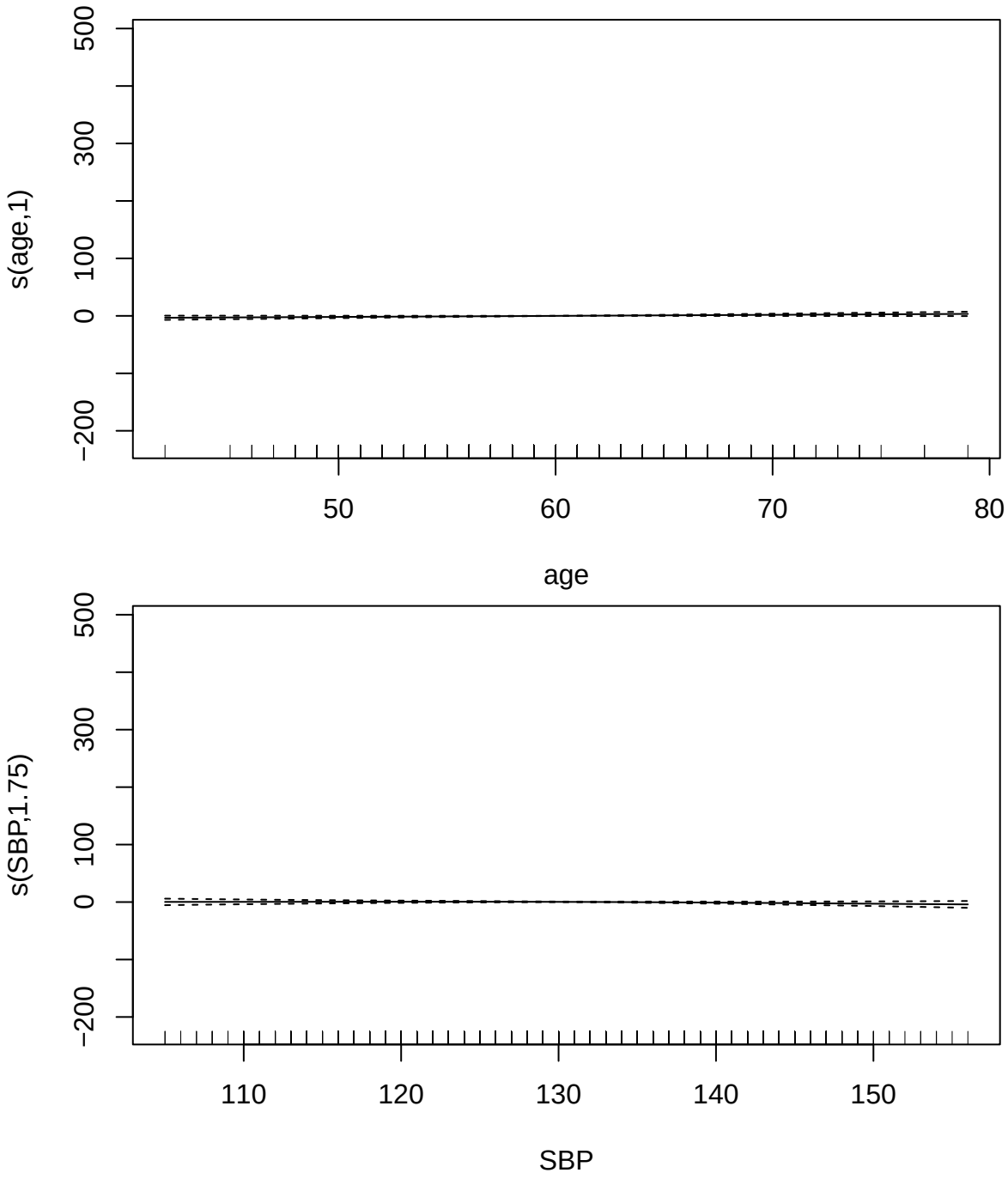
```
set.seed(666)
model.gam <- train(x = x_train,
                   y = y_train,
                   method = "gam",
                   tuneGrid = data.frame(method = "GCV.Cp", select = c(TRUE,FALSE)),
                   trControl = ctrl)

model.gam$bestTune
```

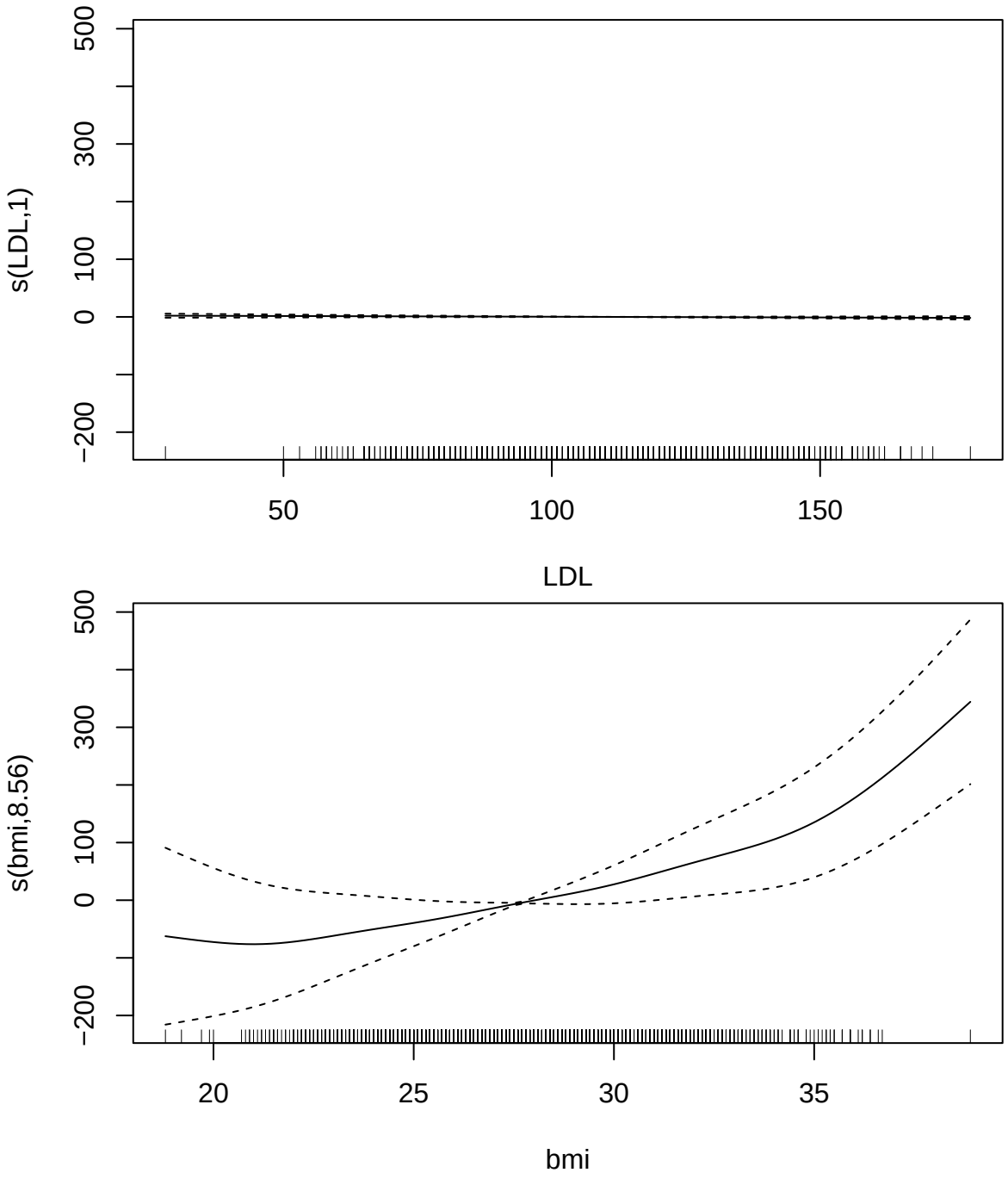
```
##   select method
## 1  FALSE GCV.Cp
```

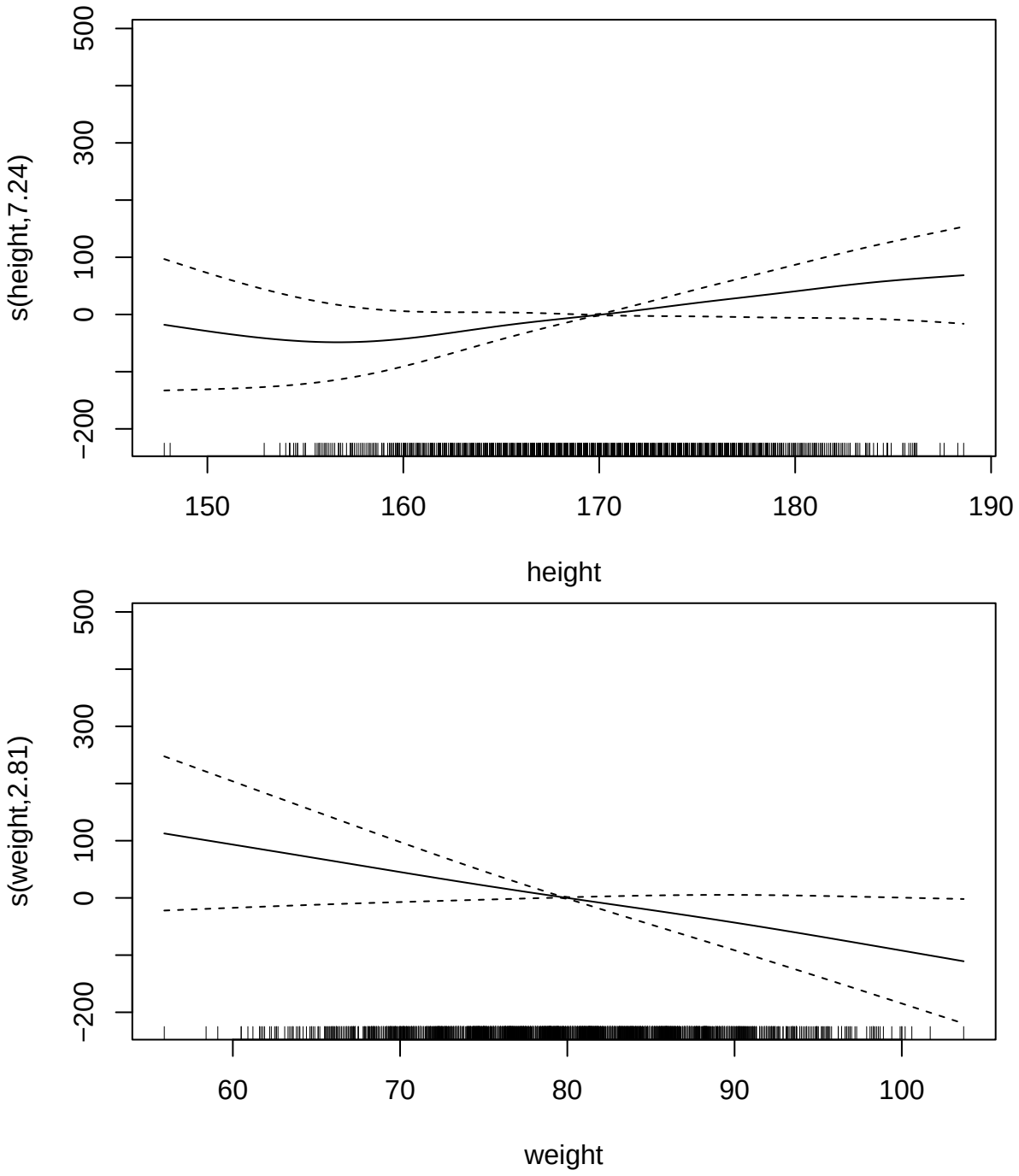
```
model.gam$finalModel
```

```
##
## 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
```

```
# degree of freedom=1 means linear

# Plotting
plot(model.gam$finalModel)
```




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




```
# compute and report the test error
predictions <- predict(model.gam, x_test)
test_error <- mean((predictions - y_test)^2) # Mean Squared Error (MSE)
test_error # Reporting the test error
```

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

The MSE of GAM model is 272.001

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 = ctrl)
```

Here's the selected tuning parameter when the minimal MSE rule is applied

```
lasso.fit$bestTune
```

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

The best tuning parameter is 0.009

And the test error is

```
lasso.pred <- predict(lasso.fit, newdata = testing_data)
# test error
mean((lasso.pred - testing_data$recovery_time)^2)
```

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

The MSE of lasso model is 298.302

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 = ctrl)
```

Here's the selected tuning parameter

```
enet.fit$bestTune
```

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

The best tuning parameter is 0.004

And the test error is

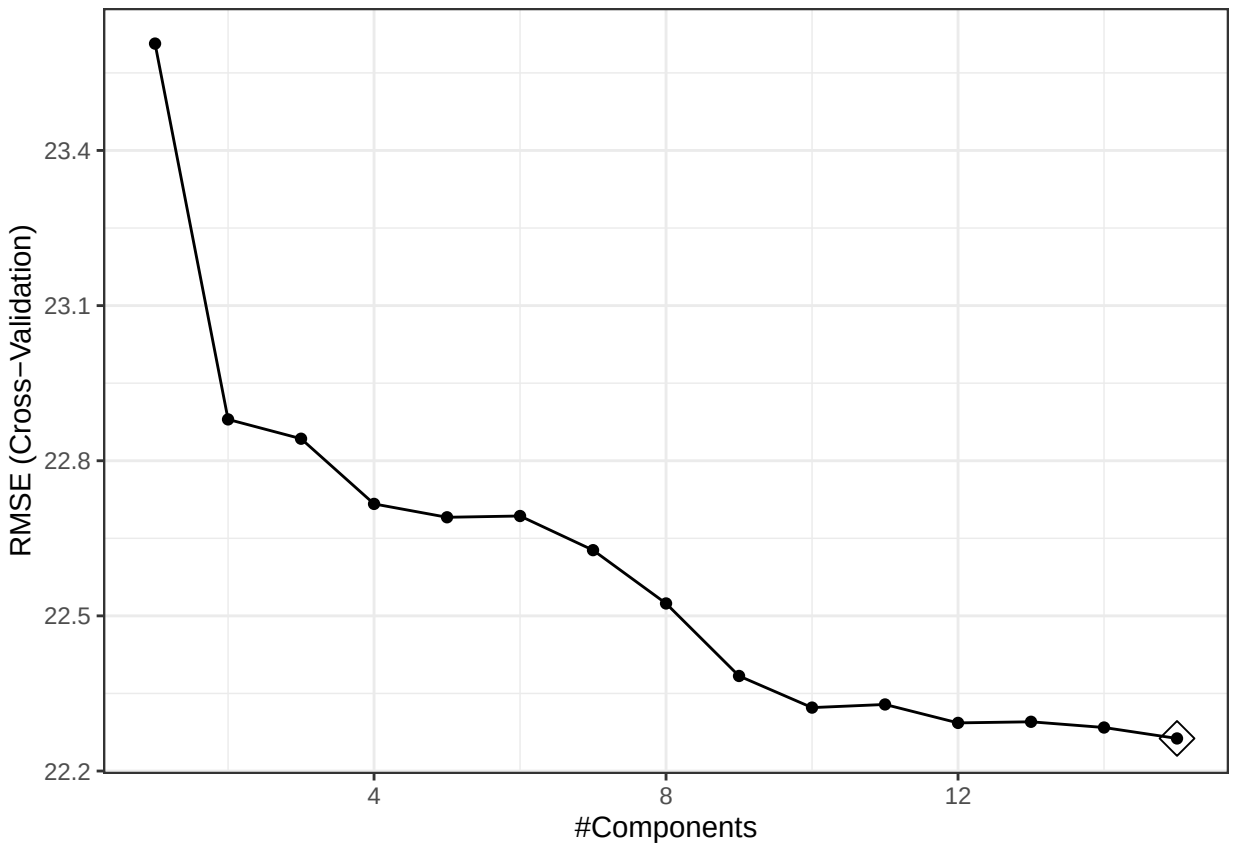
```
enet.pred <- predict(enet.fit, newdata = testing_data)
# test error
mean((enet.pred - testing_data$recovery_time)^2)
```

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

The MSE of elastic net model is 297.164

Principal components regression (PCR)

```
set.seed(666)

pcr.fit <- train(x = x_train,
                 y = y_train,
                 method = "pcr",
                 tuneGrid = data.frame(ncomp = 1:15),
                 trControl = ctrl,
                 preProcess = c("center", "scale"))
predy.pcr <- predict(pcr.fit, newdata = x_test)


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



```
# test MSE
mean((y_test - predy.pcr)^2)
```

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

The MSE of pcr model is 323.779

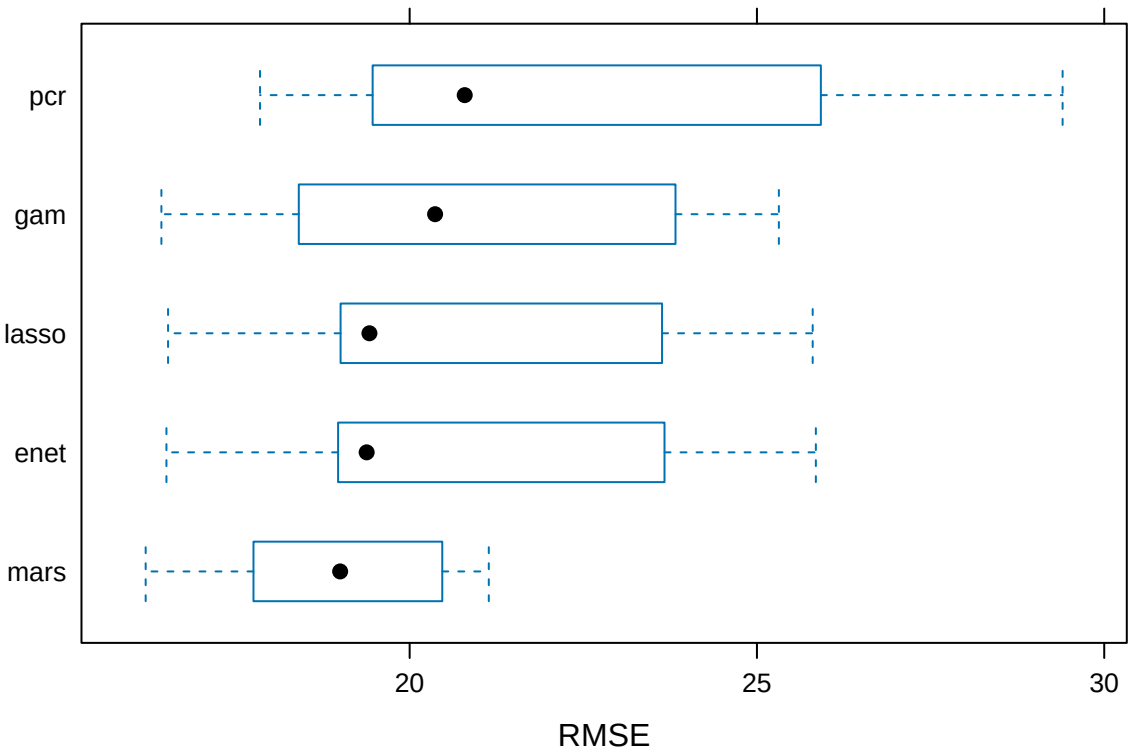
Model Comparison

compare the RMSE

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

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: mars, gam, lasso, enet, pcr
## Number of resamples: 10
##
## MAE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## mars  11.10656 12.33992 12.52758 12.48434 12.95145 13.22857    0
## gam   11.95625 12.48434 12.90413 13.04720 13.67832 14.43838    0
## lasso 12.43437 13.34989 13.55760 13.60117 13.83628 14.80446    0
## enet  12.39330 13.33858 13.50759 13.56655 13.79648 14.79199    0
## pcr   12.79843 13.31630 13.65714 13.79429 14.32701 15.26141    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## mars  16.19940 17.79482 18.99934 18.90878 20.37825 21.13959    0
## gam   16.42592 18.51665 20.36608 20.61337 23.26164 25.31540    0
## lasso 16.52182 19.01349 19.42116 20.76489 22.92278 25.80217    0
## enet  16.49848 18.99457 19.38126 20.76102 22.93865 25.84810    0
## pcr   17.84596 19.61842 20.79266 22.26300 24.74262 29.40059    0
##
## Rsquared
##             Min.    1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## mars  0.12610733 0.18362545 0.2804839 0.3515976 0.5442009 0.6258534    0
## gam   0.11451833 0.18716669 0.2610268 0.2888378 0.3891996 0.4926768    0
## lasso 0.12468127 0.17202217 0.2314610 0.2404963 0.2896872 0.3926191    0
## enet  0.12572907 0.17252570 0.2295363 0.2401821 0.2906348 0.3898854    0
## pcr   0.05140385 0.09539054 0.1072900 0.1273979 0.1285069 0.2390132    0
```

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



The MARS model is preferred since it has a lower mean value of RMSE compared to other models.