Midterm Result

Yiying Wu (yw3996)

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.

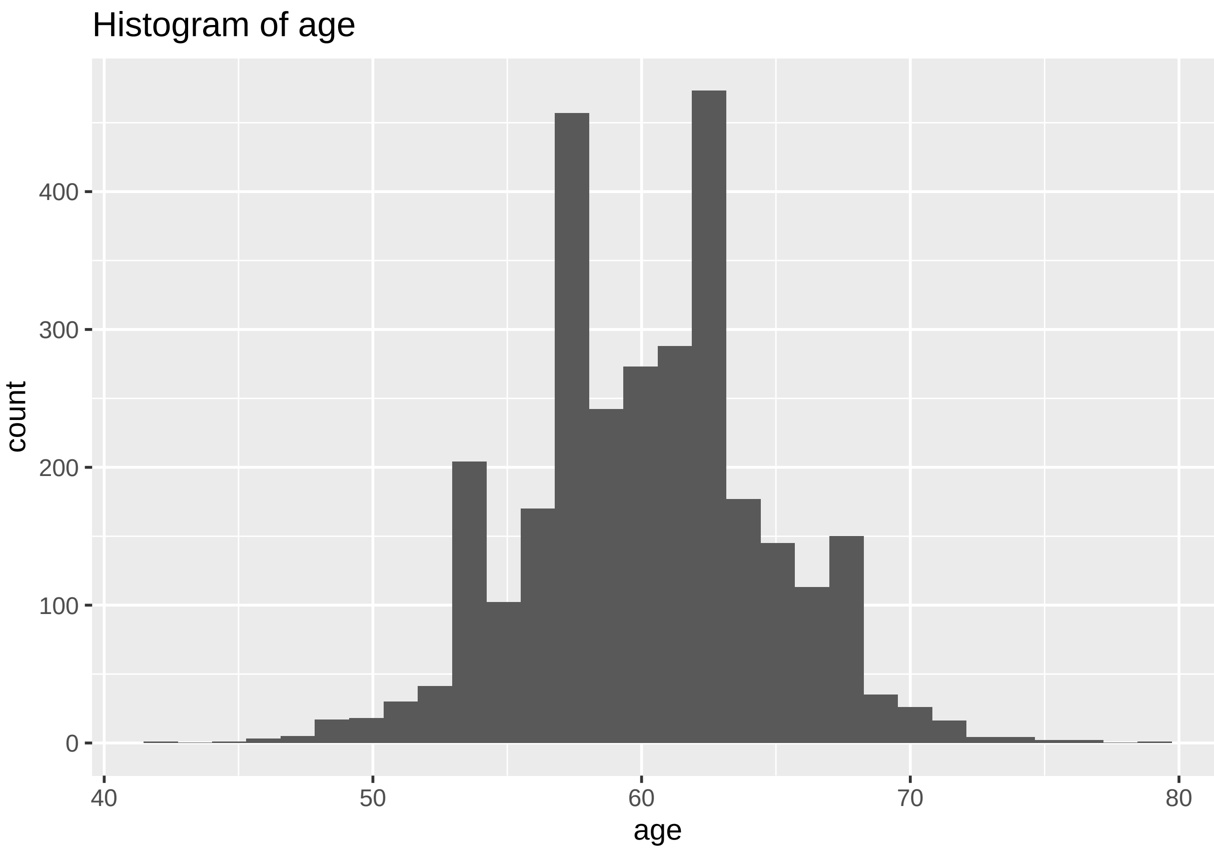
Summary statistics

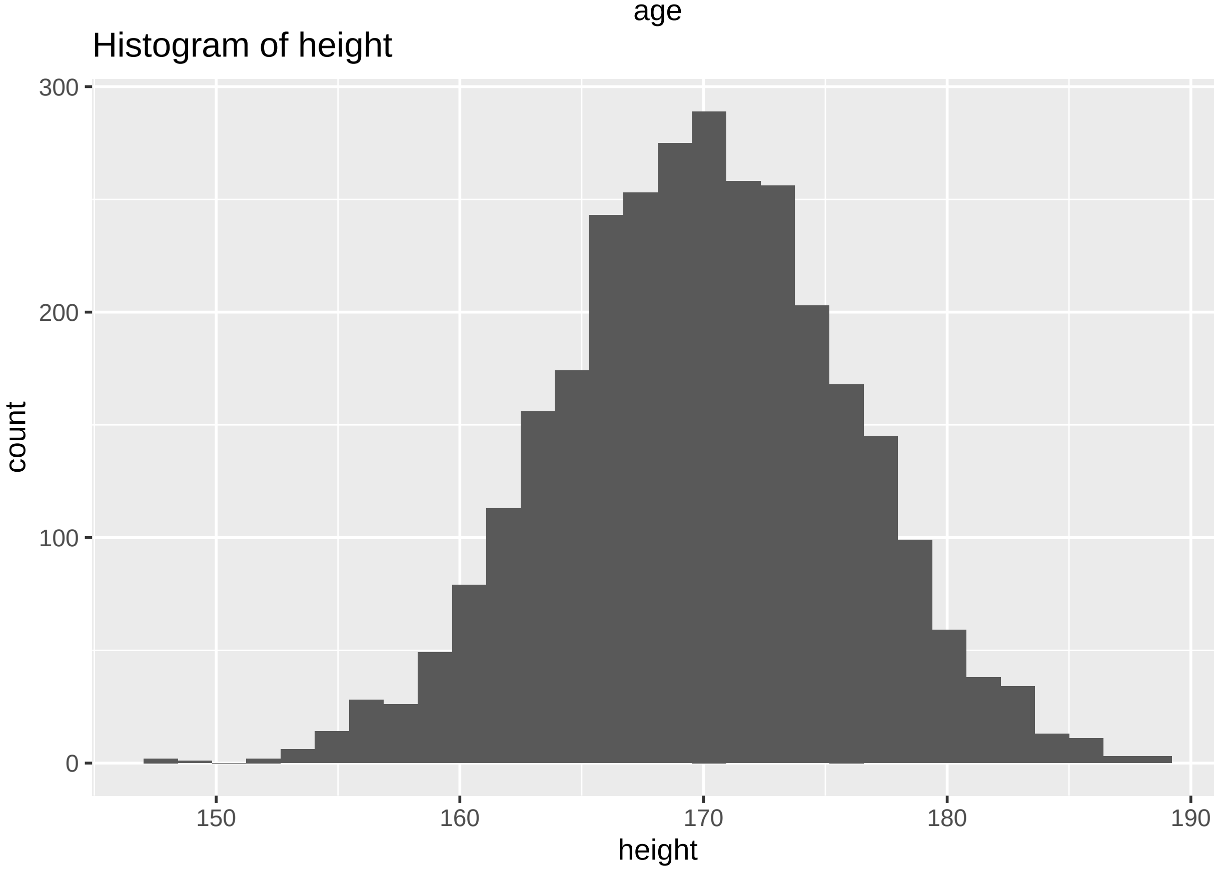
Table 1: Summary of Dataset

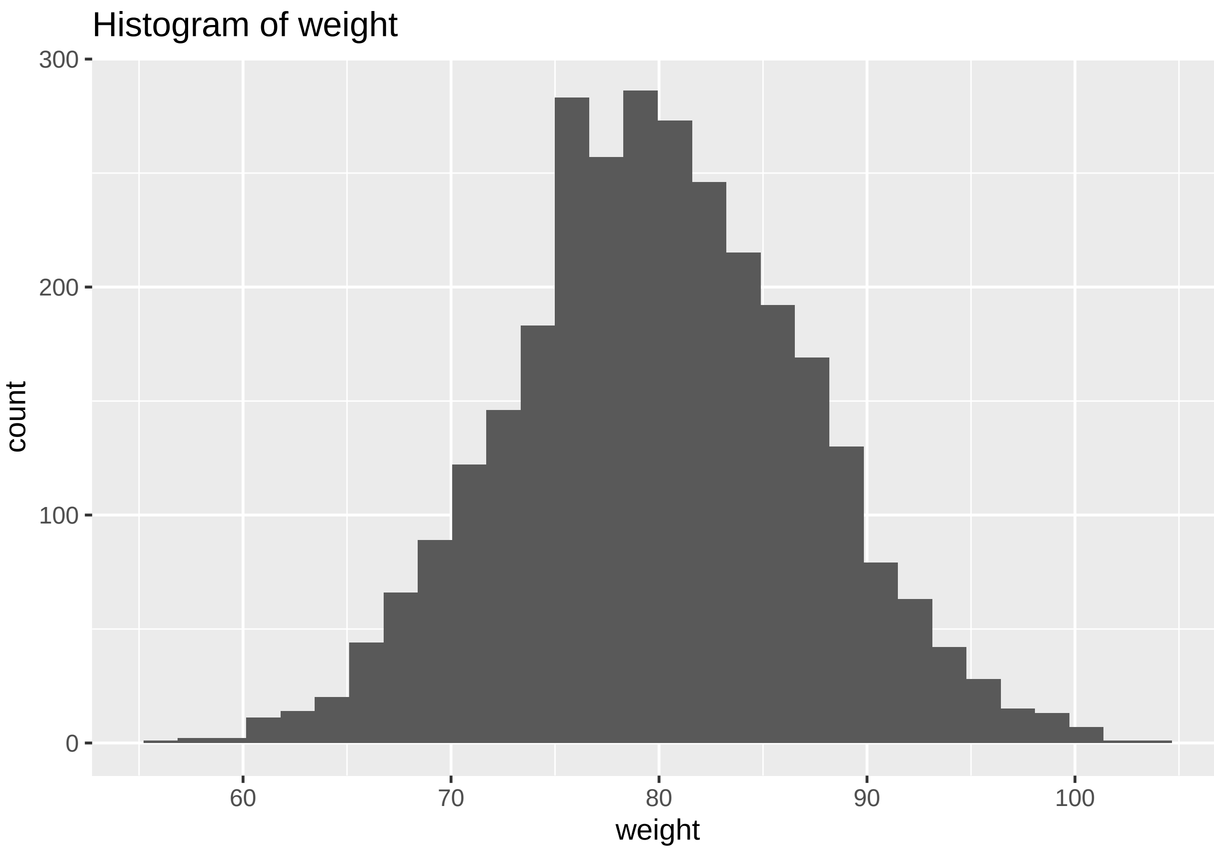
Characteristic	**N = 3,000**[1]
age	60.0 (57.0, 63.0)
gender	
male	1,544 (51%)
female	1,456 (49%)
race	
White	1,967 (66%)
Asian	158 (5.3%)
Black	604 (20%)
Hispanic	271 (9.0%)
smoking	
Never smoked	1,822 (61%)
Former smoker	859 (29%)
Current smoker	319 (11%)
height	169.9 (166.0, 173.9)
weight	80 (75, 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	
A	2,000 (67%)
B	1,000 (33%)
recovery_time	39 (31, 49)

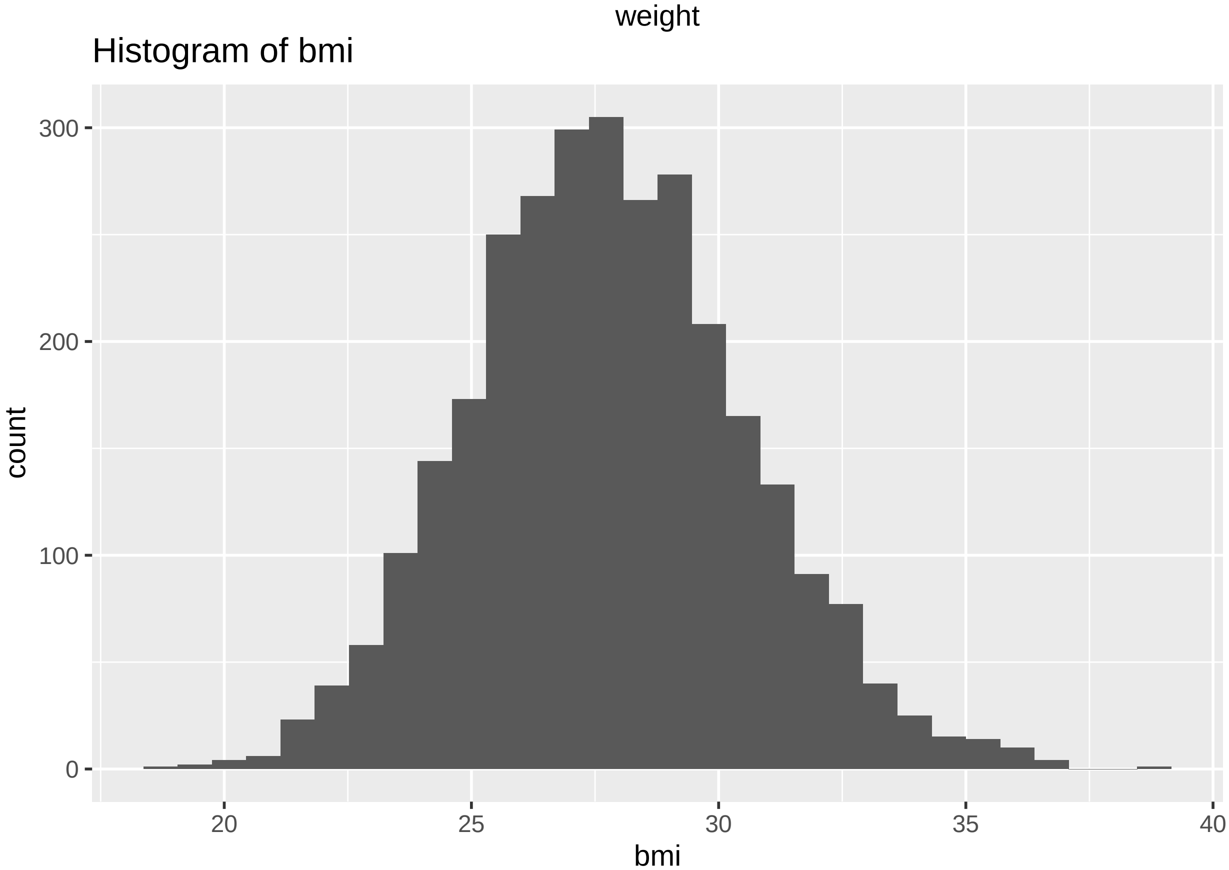
[1] Median (IQR); n (%)

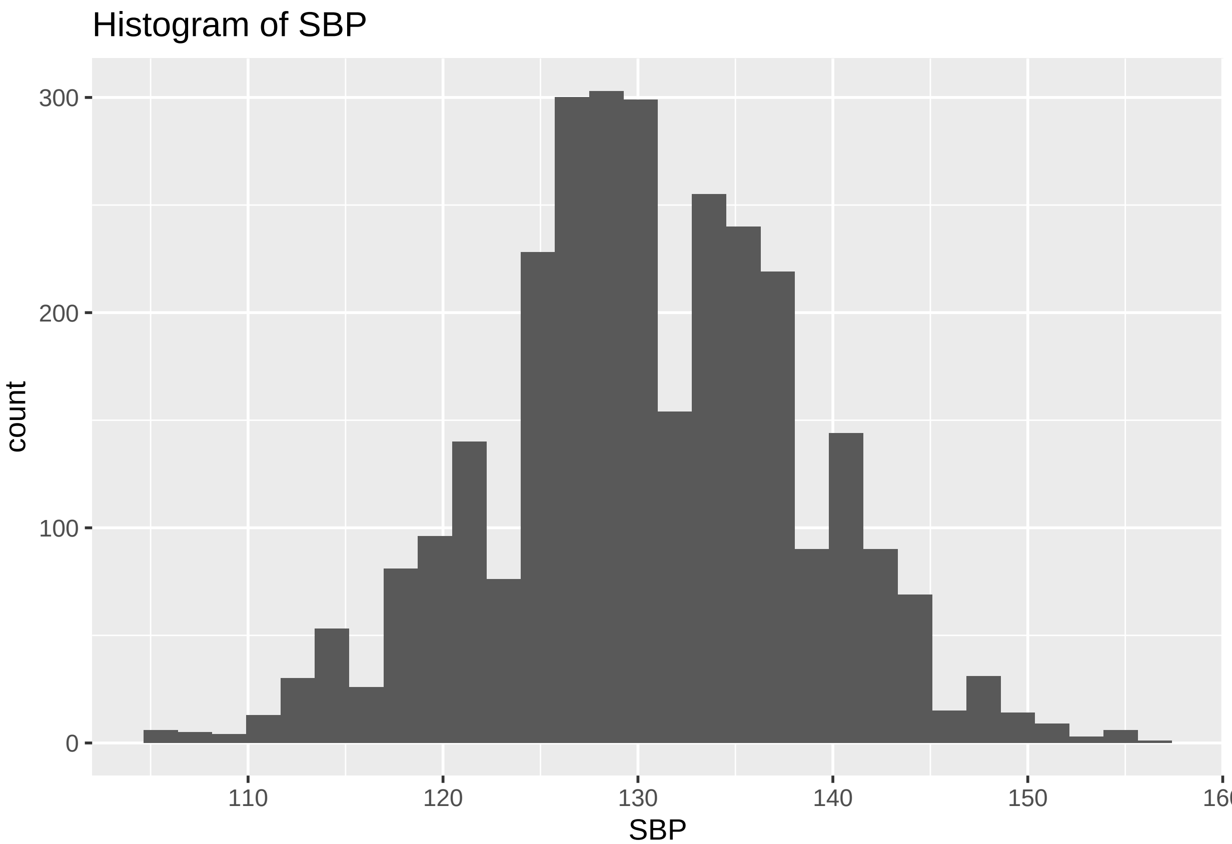
Visualizations for the numerical variables

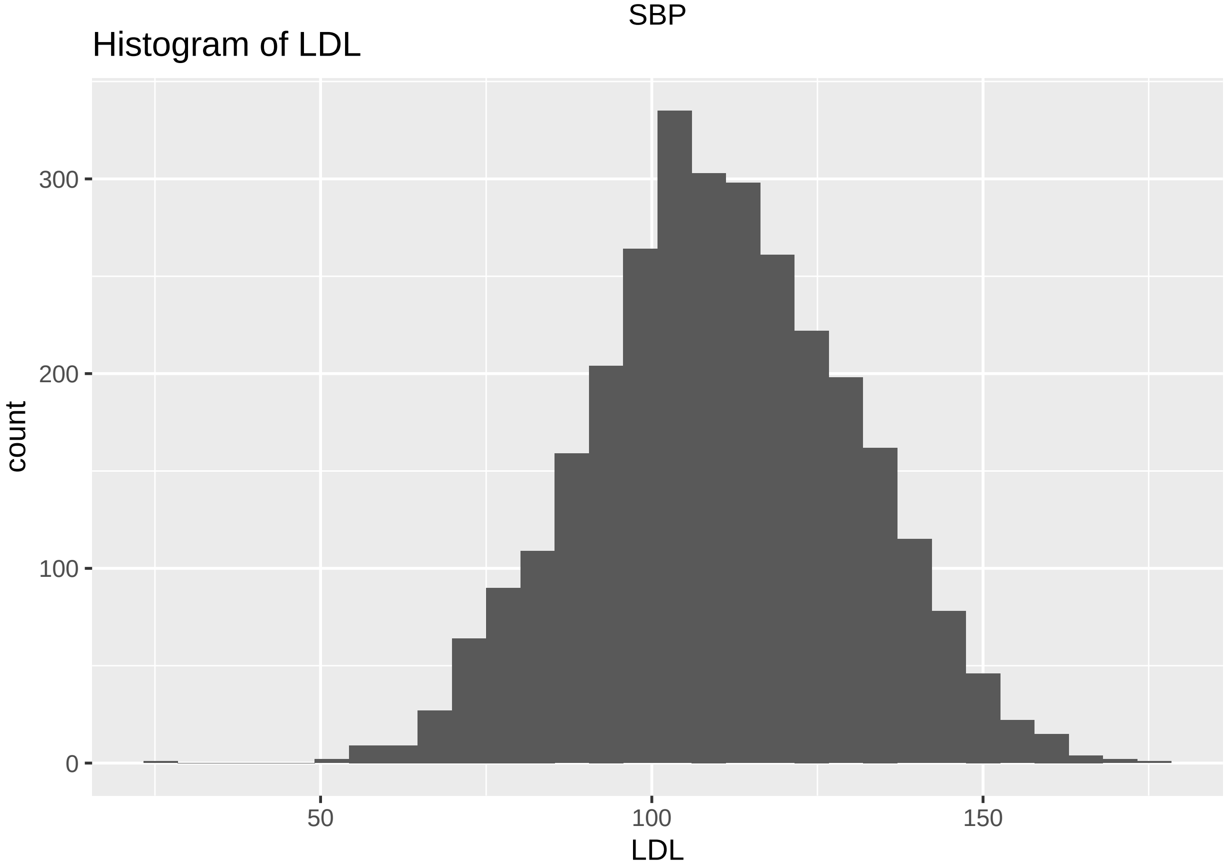

Histogram of age
count
0
100
200
300
400
40
50
60
70
80
age

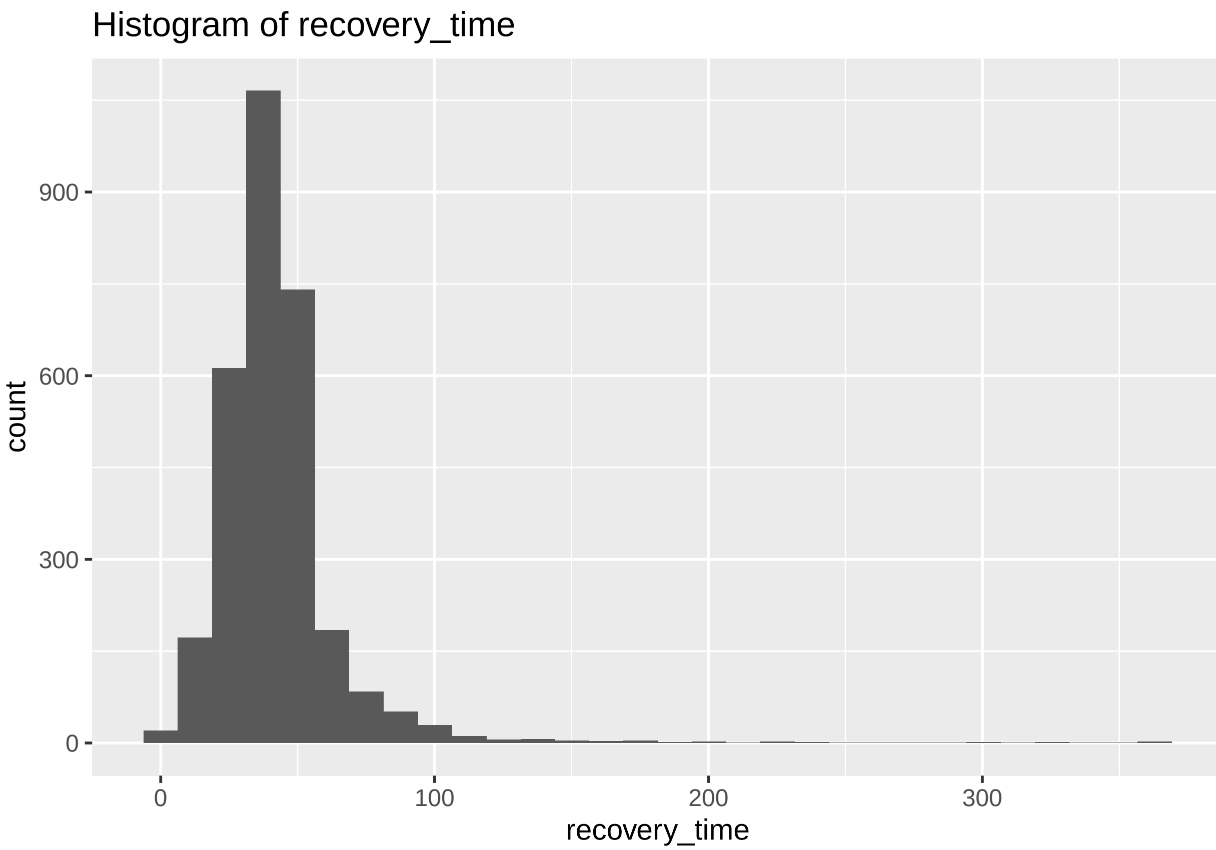

Histogram of height
count
0
100
200
300
150
160
170
180
190
height


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

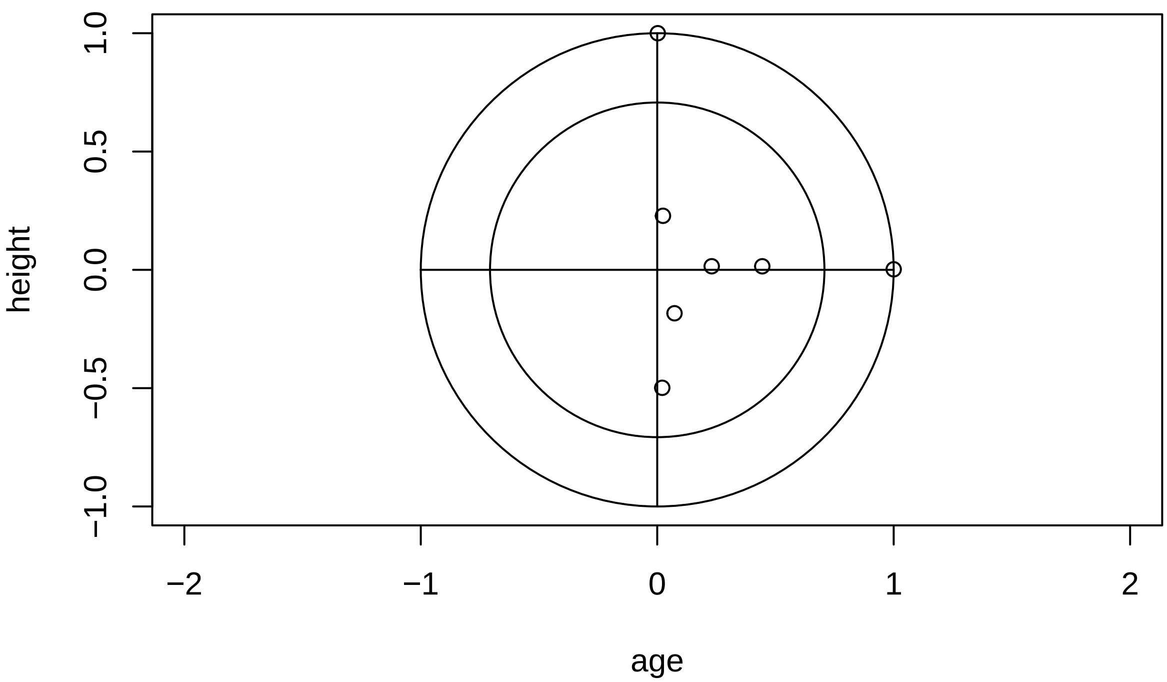

Histogram of bmi
300
200
100
0
count
20
25
30
35
40
bmi


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


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



correlation plot



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.

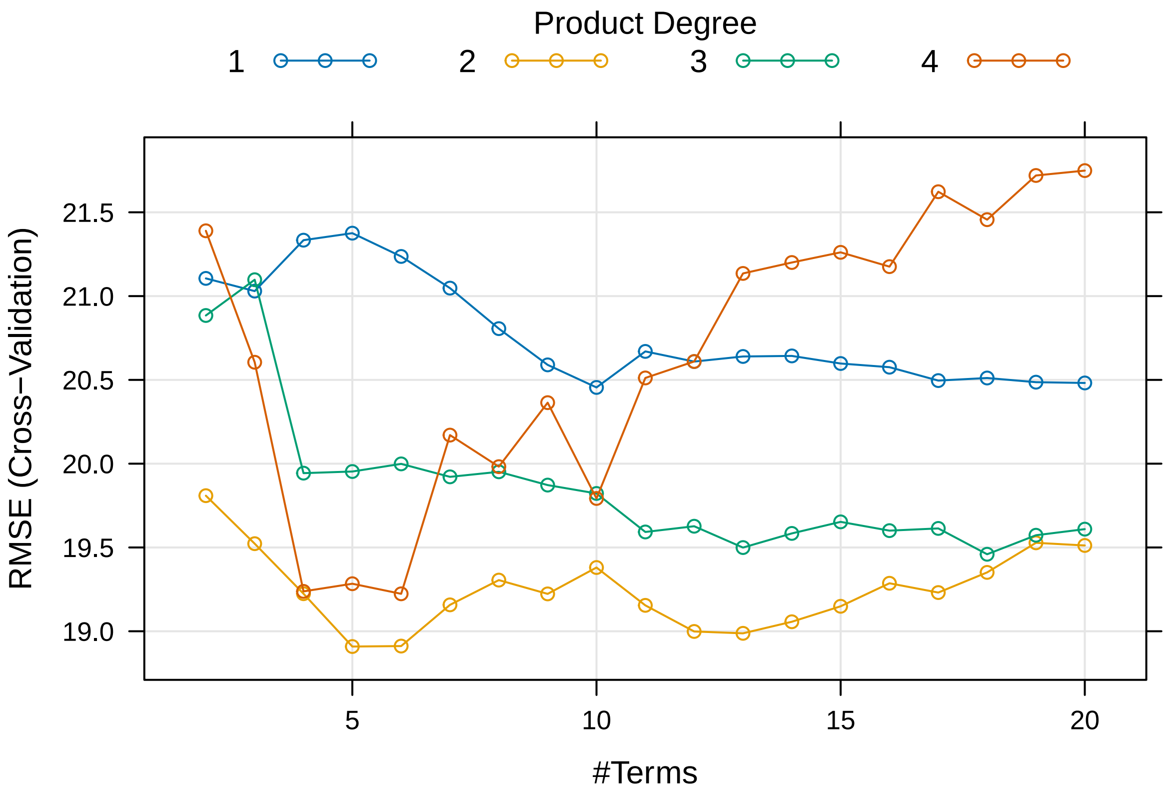
Outcome: recovery_time

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)%>% select(-id,-gender)
x_train <- training_data %>% select(-recovery_time)
y_train <- training_data$recovery_time

testing_data <- testing(data_split)%>% select(-id,-gender)
x_test <- testing_data %>% select(-recovery_time)
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 = 2:20),
                    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
## 23      5      2
```

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

```
##                (Intercept)                  h(31-bmi)
##                 -3.1983530                  6.3999877
##           h(bmi-31) * studyB              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)

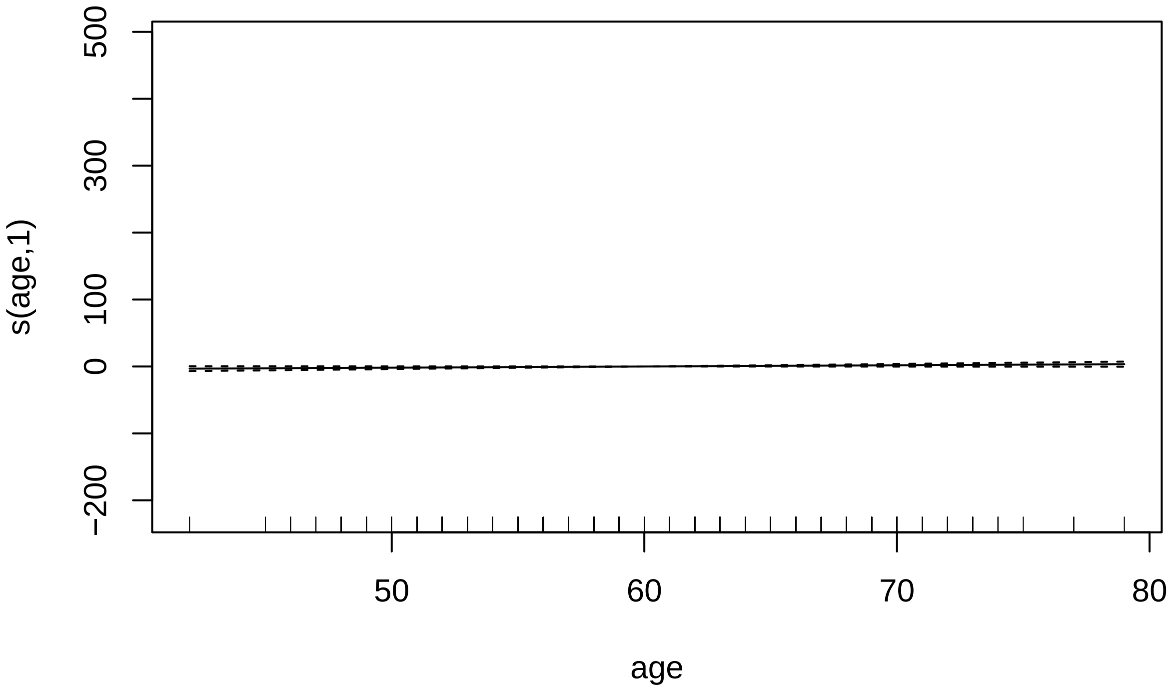
```
set.seed(666)
model.gam <- train(x = x_train,
                   y = y_train,
                   method = "gam",
                   trControl = ctrl)

model.gam$bestTune
```

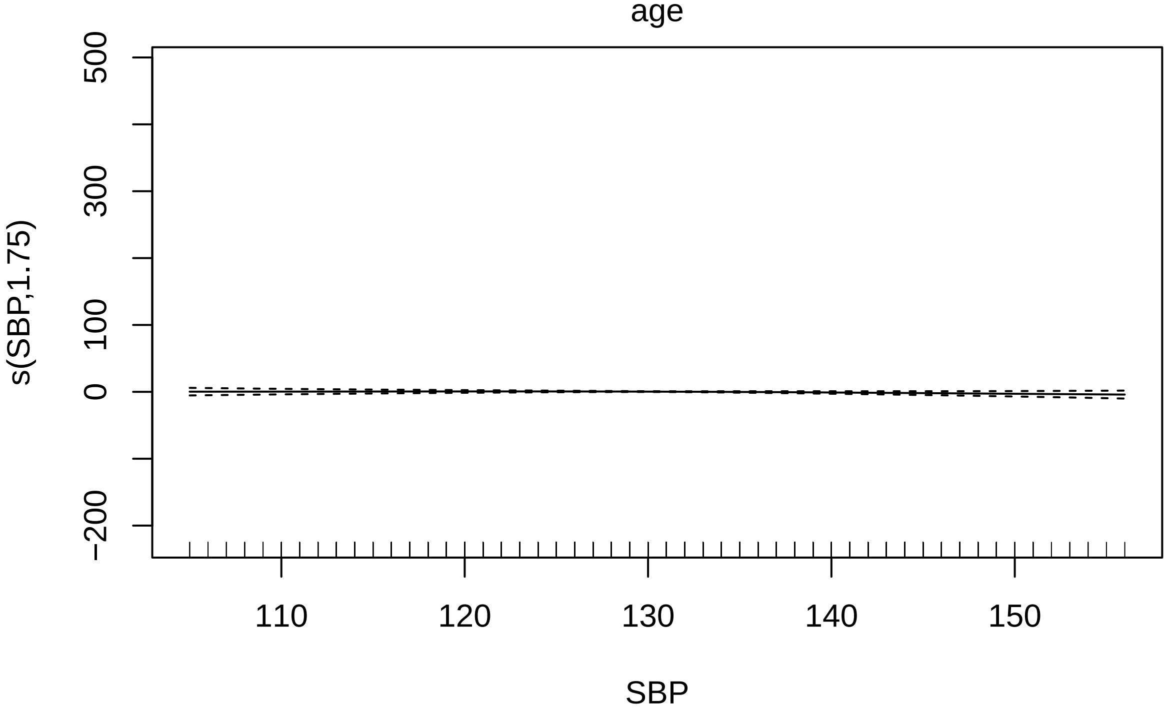
```
##   select method
## 1  FALSE GCV.Cp
```

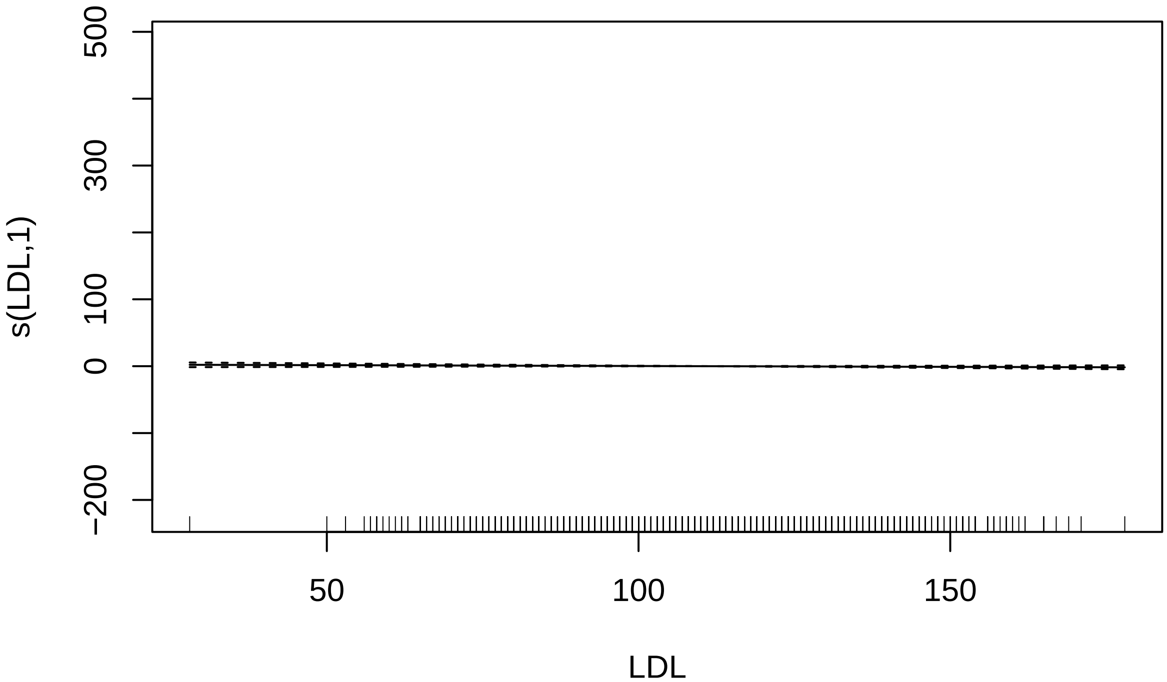
```
model.gam$finalModel
```

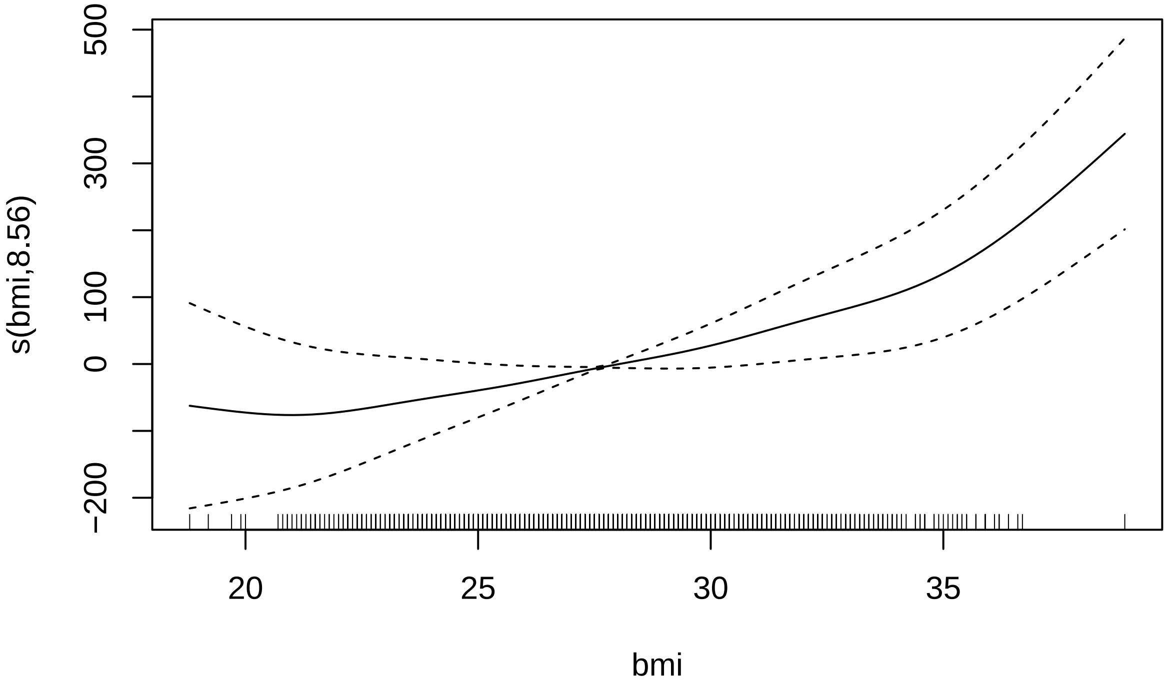
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ hypertension + diabetes + vaccine + severity + study +
##     male + smoking + race + s(age) + s(SBP) + s(LDL) + s(bmi) +
##     s(height) + s(weight)
##
## Estimated degrees of freedom:
## 1.00 1.75 1.00 8.56 7.24 2.81  total = 34.36
##
## GCV score: 384.8296
```

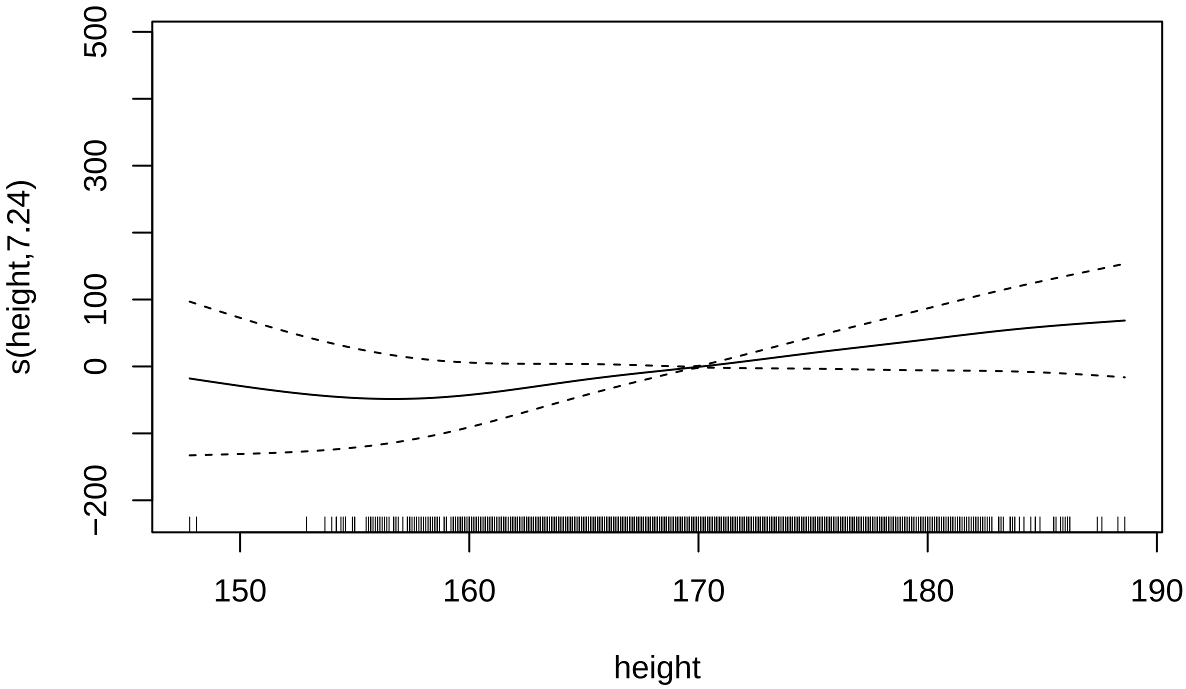
```
# degree of freedom=1 means linear

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

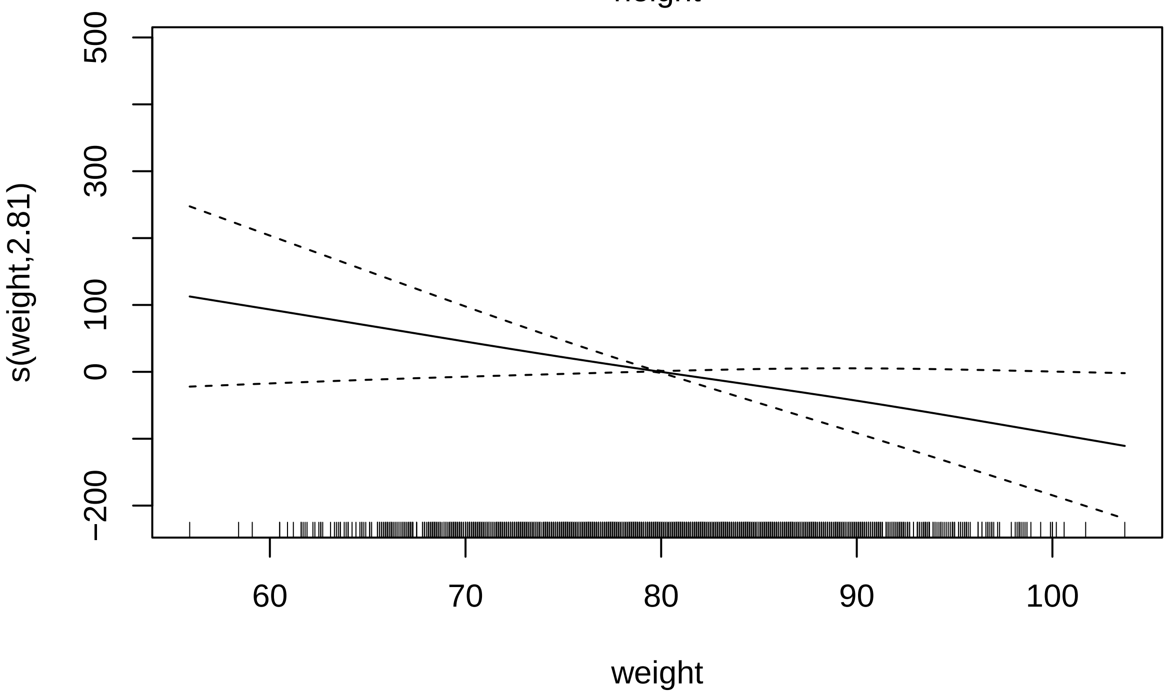

s(age,1)
500
300
100
0
−200
50
60
70
80
age


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


s(LDL,1)
−200
0
100
300
500
50
100
150
LDL


s(bmi,8.56)
−200
0
100
300
500
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] 298.3018
```

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

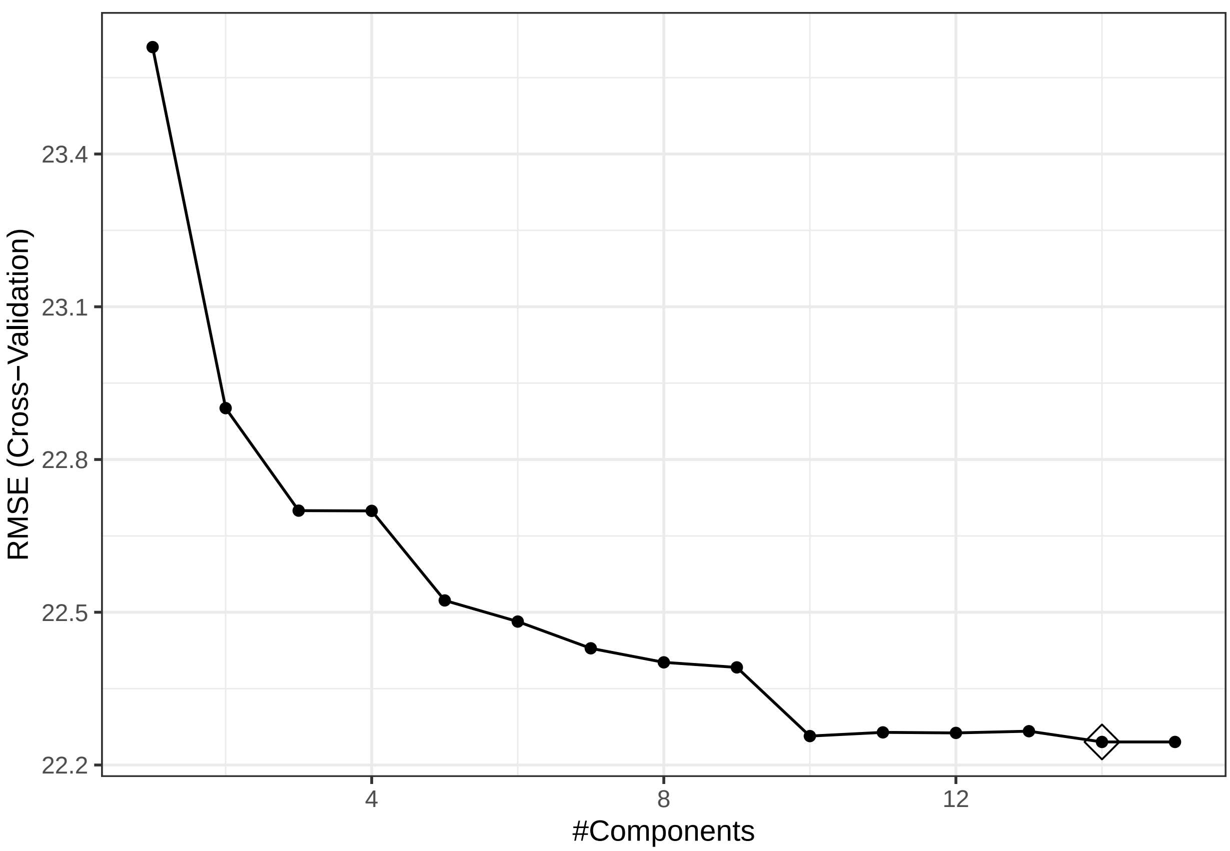
```
enet.pred <- predict(enet.fit, newdata = testing_data)
# test error
mean((enet.pred - testing_data$recovery_time)^2)
```

```
## [1] 297.1645
```

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()
```



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

```
## [1] 326.7915
```

The MSE of pcr model is 326.792

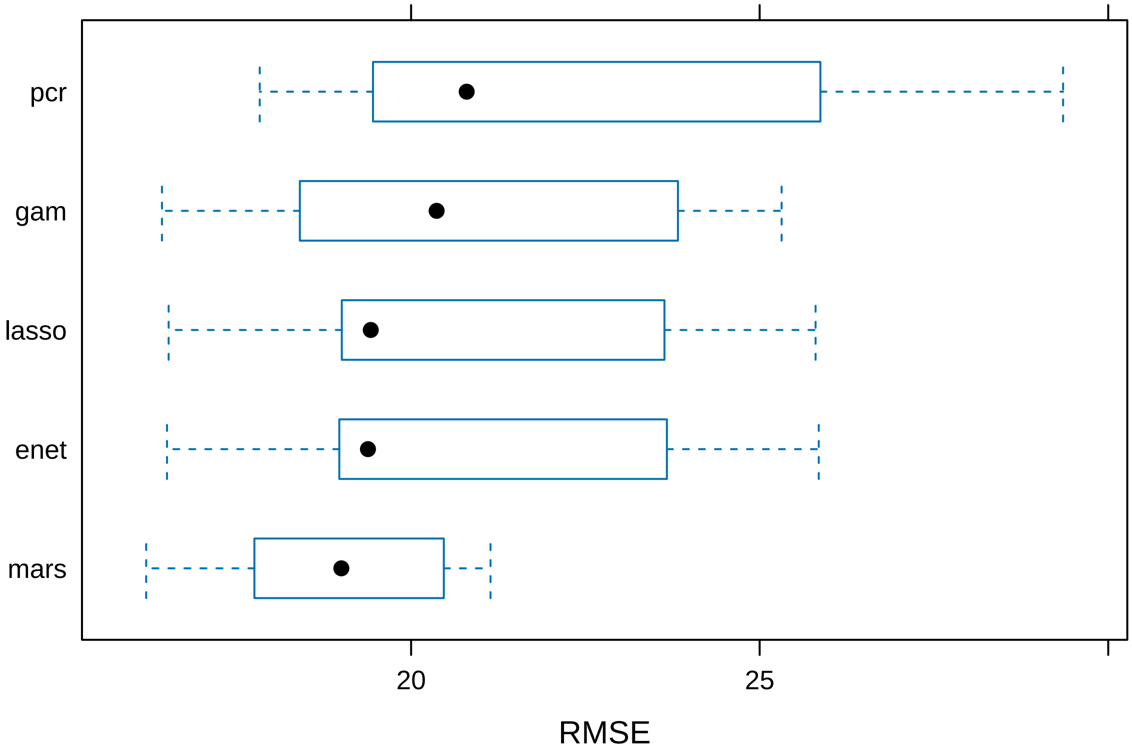
Model Comparison

compare the RMSE

```r
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 12.43438 13.34989 13.55761 13.60118 13.83628 14.80448    0
## enet  12.39331 13.33858 13.50760 13.56655 13.79648 14.79201    0
## pcr   12.72240 13.31222 13.55369 13.74933 14.24483 15.34219    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 16.52183 19.01349 19.42117 20.76489 22.92277 25.80218    0
## enet  16.49849 18.99457 19.38126 20.76102 22.93865 25.84811    0
## pcr   17.82883 19.58486 20.79741 22.24532 24.69032 29.35286    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 0.12468177 0.17202246 0.2314607 0.2404962 0.2896867 0.3926191    0
## enet  0.12572956 0.17252601 0.2295363 0.2401820 0.2906342 0.3898855    0
## pcr   0.05353634 0.09951385 0.1073118 0.1279829 0.1298992 0.2421886    0
```

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



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