P8106_Midterm_Code_yh3554

Yi Huang

2023-03-30

Contents

```
library(tidyverse)
library(dplyr)
library(gtsummary) # data summary table
library(ggplot2)
library(GGally) # ggplot, ggpair
library(viridis) # color and theme
library(caret)
library(doBy)
library(glmnet)
library(earth)
library(randomForest)
library(ranger)
library(gbm)
library(mgcv)
library(nlme)
library(vip)

options(
  ggplot2.continuous.colour = "viridis",
  ggplot2.continuous.fill = "viridis"
)

scale_colour_discrete = scale_colour_viridis_d
scale_fill_discrete = scale_fill_viridis_d
viridis::scale_fill_viridis()
```

```
## <ScaleContinuous>
##  Range:
##  Limits:    0 --    1
```

```
theme_set(theme_minimal() + theme(legend.position = "bottom"))
```

Background

Description of each variable:

Variable Name (Column Name): Description

ID (id) :Participant ID

Gender (gender): 1 = Male, 0 = Female

Race/ethnicity (race): 1 = White, 2 = Asian, 3 = Black, 4 = Hispanic

Smoking (smoking): Smoking status; 0 = Never smoked, 1 = Former smoker, 2 = Current smoker

Height (height): Height (in centimeters)

Weight (weight): Weight (in kilograms)

BMI (bmi): Body Mass Index; BMI = weight (in kilograms) / height (in meters) squared

Hypertension (hypertension): 0 = No, 1 = Yes

Diabetes (diabetes): 0 = No, 1 = Yes

Systolic blood pressure (SBP): Systolic blood pressure (in mm/Hg)

LDL cholesterol (LDL): LDL (low-density lipoprotein) cholesterol (in mg/dL)

Vaccination status at the time of infection (vaccine): 0 = Not vaccinated, 1 = Vaccinated

Severity of COVID-19 infection (severity): 0 = Not severe, 1= Severe

Study (study): The study (A/B/C) that the participant belongs to

Time to recovery (tt_recovery_time): Time from COVID-19 infection to recovery in days

The dataset in "recovery.RData" consists of 10000 participants.

In your analysis, please draw a random sample of 2000 participants using the following R code:

set.seed([last four digits of your UNI])

dat <- dat[sample(1:10000, 2000),]

The resulting dat object will contain a random sample of 2000 participants that you can use for your analysis.

Data Cleaning and Visulizations

Load data

```
# set seed for reproducibility
set.seed(3554)

# load data
load("recovery.RData")
dat <- dat[sample(1:10000, 2000),]
```

Data Cleaning

```
dat <- dat %>%
  mutate(gender = factor(gender),
         hypertension = factor(hypertension),
         diabetes = factor(diabetes),
         vaccine = factor(vaccine),
         severity = factor(severity),
         study = factor(study),
)
```

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.

EDA

```
summary(dat)
```

```
##        id             age        gender   race     smoking      height
##  Min.   :   2   Min.   :45.00   0:1015   1:1299   0:1215   Min.   :147.8
##  1st Qu.:2527   1st Qu.:57.00   1: 985   2:  96   1: 580   1st Qu.:165.8
##  Median :4952   Median :60.00            3: 422   2: 205   Median :170.0
##  Mean   :4969   Mean   :59.92            4: 183            Mean   :170.0
##  3rd Qu.:7366   3rd Qu.:63.00                              3rd Qu.:173.9
##  Max.   :9998   Max.   :77.00                              Max.   :195.9
```

```
##      weight            bmi        hypertension diabetes      SBP
##  Min.   : 54.90   Min.   :19.20   0:1052       0:1670   Min.   :104.0
##  1st Qu.: 75.40   1st Qu.:25.70   1: 948       1: 330   1st Qu.:125.0
##  Median : 80.00   Median :27.70                         Median :130.0
##  Mean   : 79.91   Mean   :27.72                         Mean   :130.1
##  3rd Qu.: 84.70   3rd Qu.:29.50                         3rd Qu.:136.0
##  Max.   :103.40   Max.   :37.00                         Max.   :158.0
##       LDL        vaccine  severity study    recovery_time
##  Min.   : 58.0   0: 785   0:1809   A: 402   Min.   :  1.00
##  1st Qu.: 97.0   1:1215   1: 191   B:1209   1st Qu.: 28.00
##  Median :110.0                     C: 389   Median : 39.00
##  Mean   :110.2                              Mean   : 42.82
##  3rd Qu.:124.0                              3rd Qu.: 50.00
##  Max.   :173.0                              Max.   :365.00
```

```
dat %>% select(age, height, weight, bmi, SBP, LDL,
               recovery_time, gender, race, smoking,
               hypertension, diabetes, vaccine, severity,
               study) %>% tbl_summary()
```

Characteristic	**N = 2,000**
age	60.0 (57.0, 63.0)
height	170 (166, 174)
weight	80 (75, 85)
bmi	27.70 (25.70, 29.50)
SBP	130 (125, 136)
LDL	110 (97, 124)
recovery_time	39 (28, 50)
gender	
0	1,015 (51%)
1	985 (49%)
race	
1	1,299 (65%)
2	96 (4.8%)
3	422 (21%)
4	183 (9.2%)
smoking	
0	1,215 (61%)
1	580 (29%)
2	205 (10%)
hypertension	
0	1,052 (53%)
1	948 (47%)
diabetes	
0	1,670 (84%)
1	330 (16%)
vaccine	
0	785 (39%)
1	1,215 (61%)
severity	
0	1,809 (90%)
1	191 (9.6%)
study	
A	402 (20%)

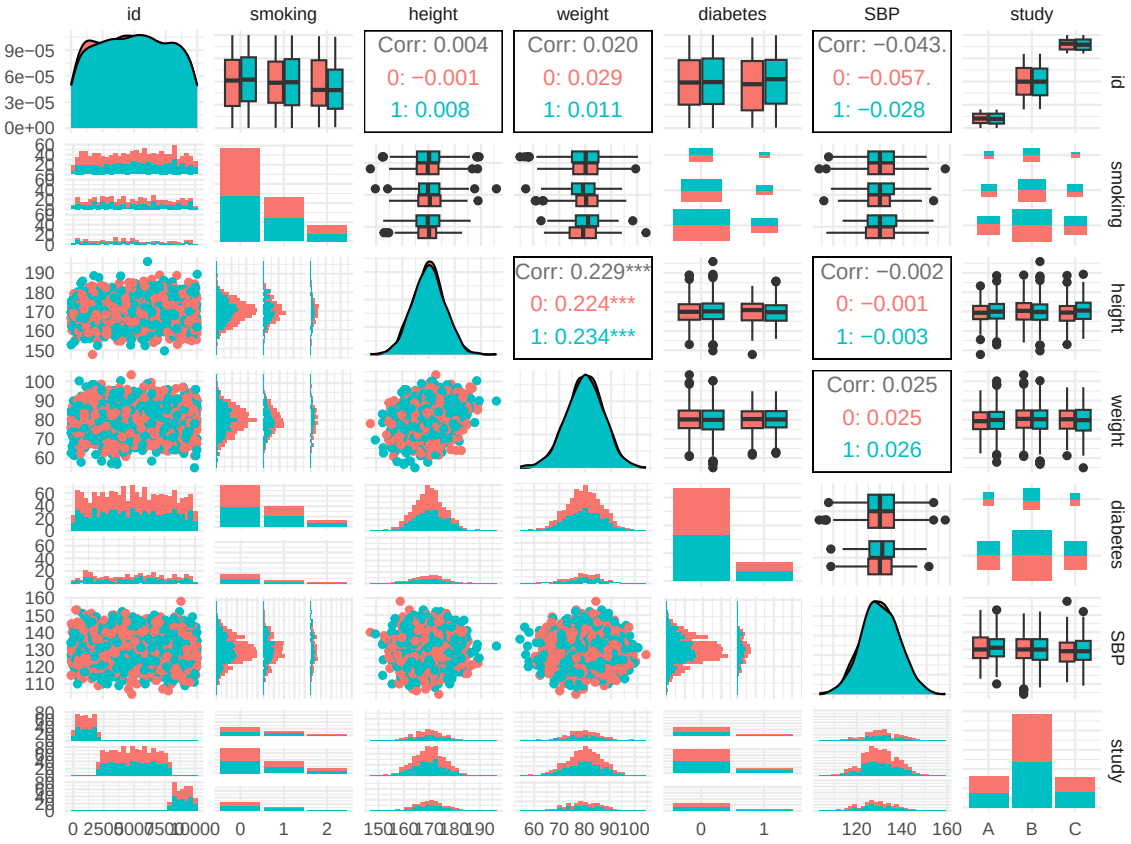
Characteristic	**N = 2,000**
B	1,209 (60%)
C	389 (19%)

```
dat %>% select(age, height, weight, bmi, SBP, LDL,
               recovery_time, gender, race, smoking,
               hypertension, diabetes, vaccine, severity,
               study) %>% tbl_summary(by = study)
```

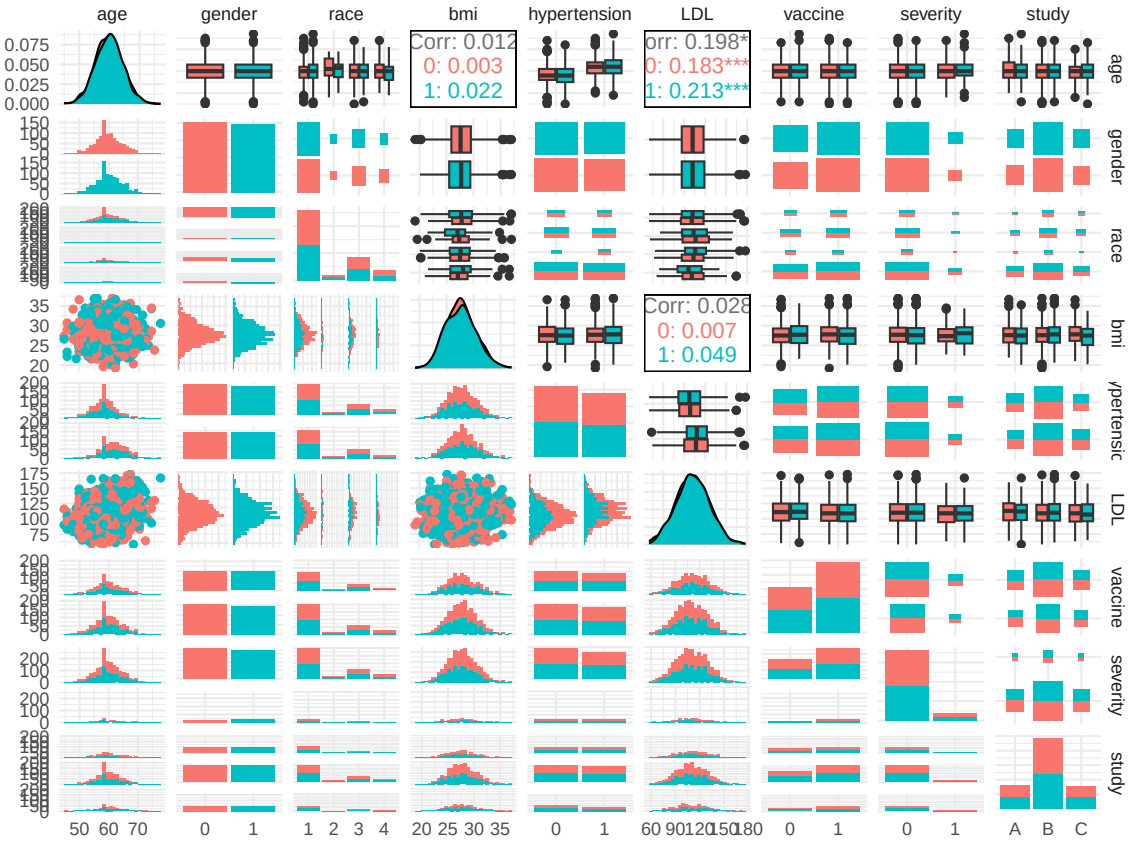
Characteristic	**A**, N = 402	**B**, N = 1,209	**C**, N = 389
age	60.0 (57.0, 63.0)	60.0 (57.0, 63.0)	60.0 (57.0, 63.0)
height	170 (166, 173)	170 (166, 174)	170 (166, 174)
weight	79 (75, 84)	80 (76, 85)	80 (75, 85)
bmi	27.60 (25.60, 29.30)	27.70 (25.80, 29.50)	27.60 (25.60, 29.40)
SBP	131 (126, 136)	130 (125, 136)	129 (124, 135)
LDL	112 (98, 125)	110 (97, 123)	108 (96, 123)
recovery_time	40 (33, 47)	37 (23, 56)	39 (33, 45)
gender			
0	211 (52%)	613 (51%)	191 (49%)
1	191 (48%)	596 (49%)	198 (51%)
race			
1	271 (67%)	779 (64%)	249 (64%)
2	21 (5.2%)	60 (5.0%)	15 (3.9%)
3	82 (20%)	257 (21%)	83 (21%)
4	28 (7.0%)	113 (9.3%)	42 (11%)
smoking			
0	242 (60%)	724 (60%)	249 (64%)
1	114 (28%)	357 (30%)	109 (28%)
2	46 (11%)	128 (11%)	31 (8.0%)
hypertension			
0	196 (49%)	629 (52%)	227 (58%)
1	206 (51%)	580 (48%)	162 (42%)
diabetes			
0	327 (81%)	1,017 (84%)	326 (84%)
1	75 (19%)	192 (16%)	63 (16%)
vaccine			
0	164 (41%)	471 (39%)	150 (39%)
1	238 (59%)	738 (61%)	239 (61%)
severity			
0	365 (91%)	1,091 (90%)	353 (91%)
1	37 (9.2%)	118 (9.8%)	36 (9.3%)

Plots

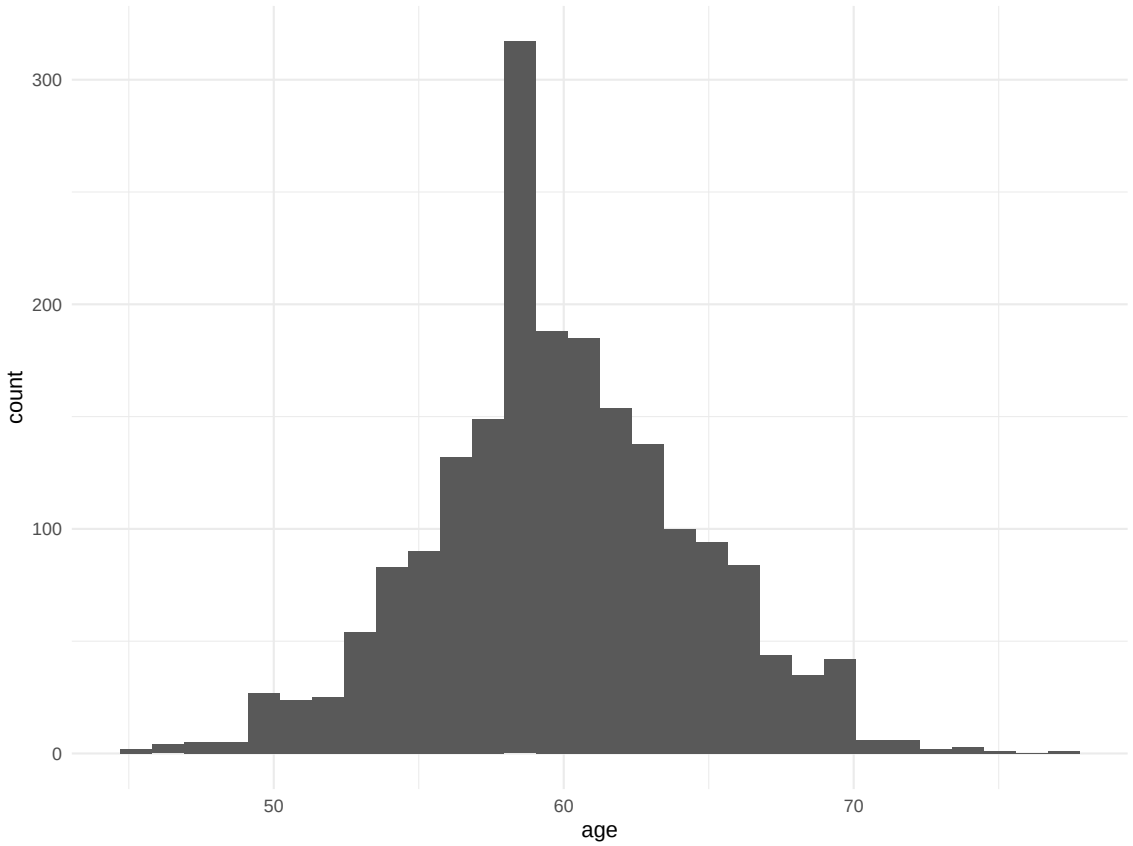
```
ggpairs(dat, columns = c(1, 5, 6, 7, 10, 11, 15), ggplot2::aes(colour=gender))
```



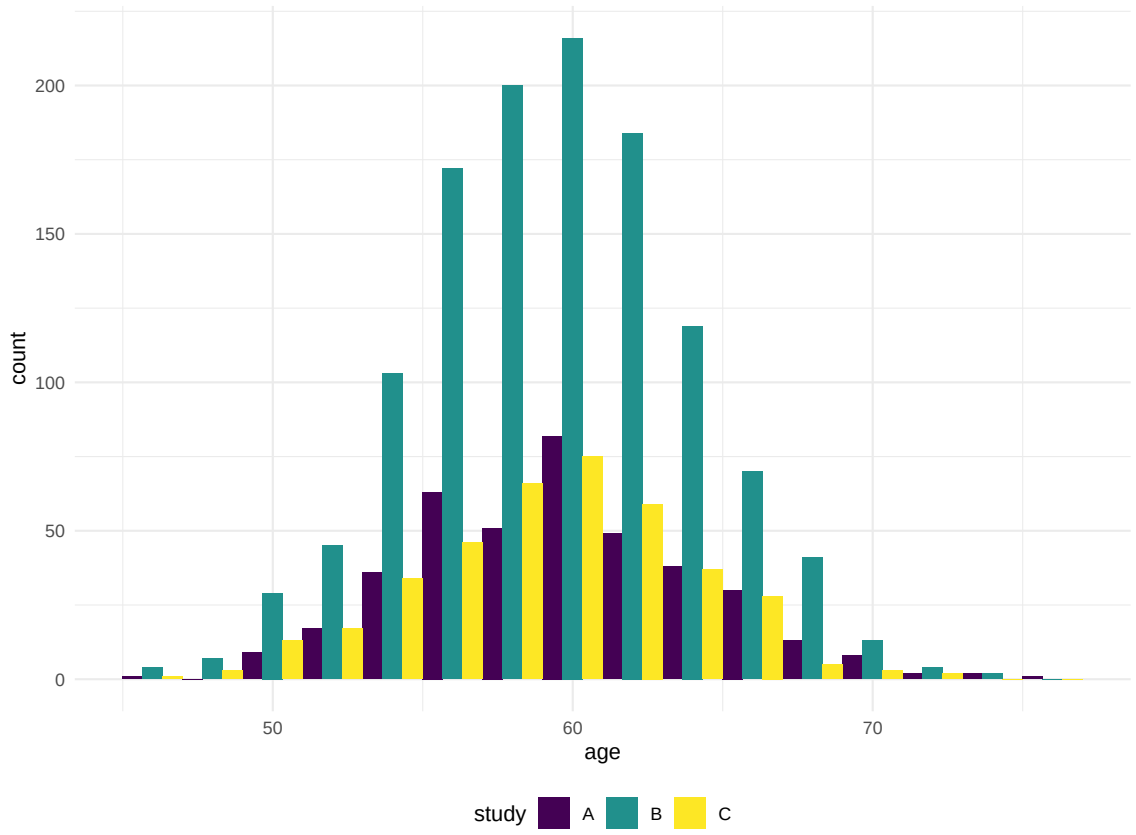

```
ggsave(file="image/gender_ggpair_cont.png",width=10,height=7)

ggpairs(dat, columns = c(2, 3, 4, 8, 9, 12, 13, 14, 15), ggplot2::aes(colour=gender))
```



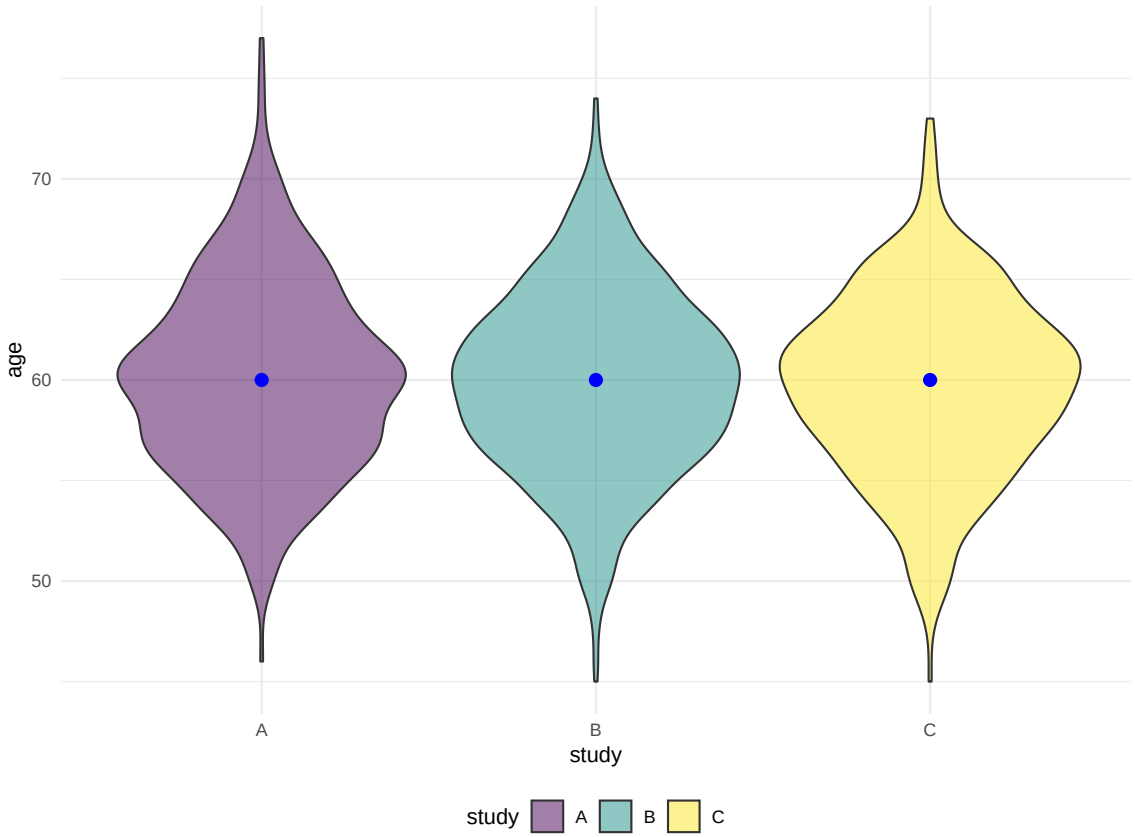

```
ggsave(file="image/gender_ggpair_cat.png",width=10,height=7)

# ggpairs(dat, title="correlogram with ggpairs()")

ggplot(dat, aes(x = age)) +
  geom_histogram()
```




```
ggsave(file="image/age_histogram.png",width=8,height=5)

ggplot(dat, aes(x = age, fill = study)) +
  geom_histogram(position = "dodge", binwidth = 2)
```




```
ggsave(file="image/age_histogram_by_study.png",width=8,height=5)

ggplot(dat, aes(x = study, y = age)) +
  geom_violin(aes(fill = study), alpha = .5) +
  stat_summary(fun = "median", color = "blue")
```




```
ggsave(file="image/age_violin_plot.png",width=8,height=5)
```

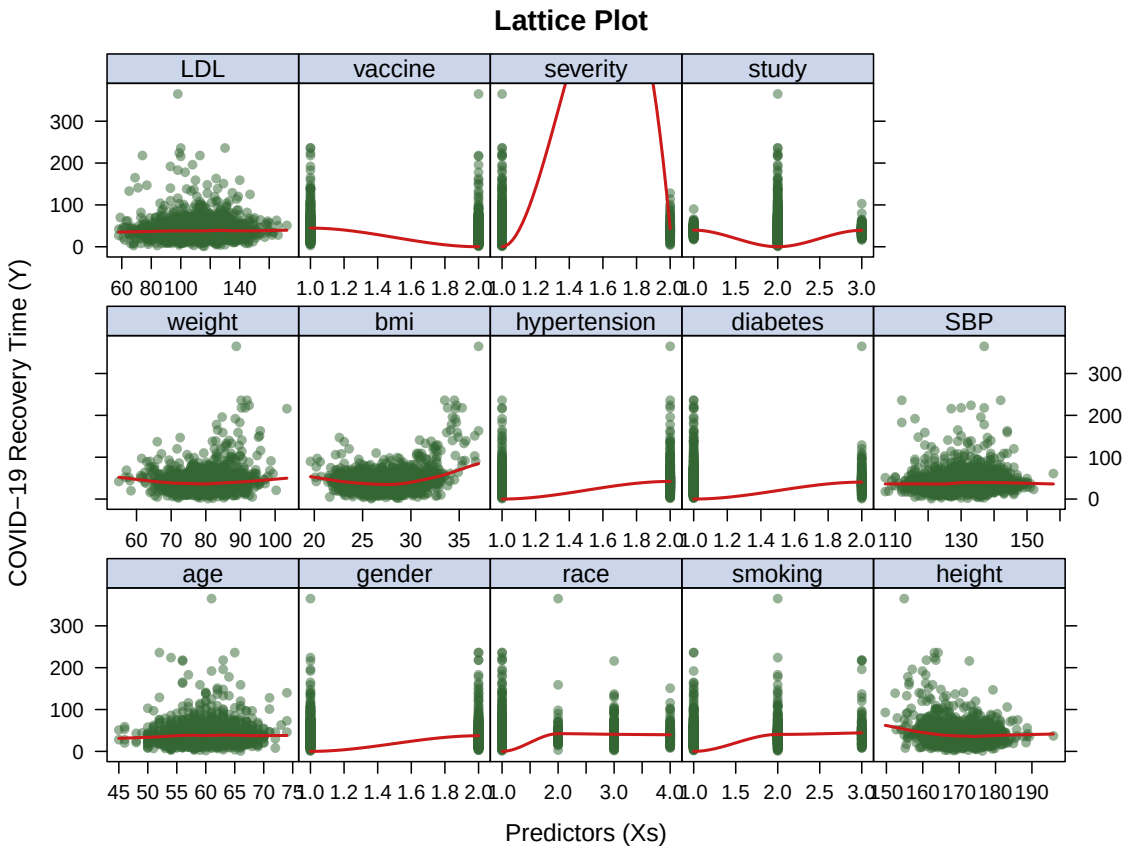
Split the data

```
set.seed(3554)
# split the data into training data (70%) and test data (30%)
# specify rows of training data (70% of the dataset)
train_rows <- createDataPartition(dat$recovery_time,
                                  p = 0.7,
                                  list = F)
dat <- dat[,-1]
dat2 <- model.matrix(recovery_time~., dat)[,-1]

# train for viz
dat_train <- dat[train_rows,]
dat_test <- dat[-train_rows,]

# training data
train_x <- dat2[train_rows,]
train_y <- dat$recovery_time[train_rows]
train_x1 <- dat[train_rows, -c(2, 3, 4, 8, 9, 12, 13, 14)] #continuous predictors

# test data
test_x <- dat2[-train_rows,]
test_y <- dat$recovery_time[-train_rows]
test_x1 <- dat[-train_rows, -c(2, 3, 4, 8, 9, 12, 13, 14)] #continuous predictors
```

```
# resampling method repeated cross-validation
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 10)

ctrl1 <- trainControl(method = "repeatedcv",
                      selectionFunction = "oneSE",
                      number = 10,
                      repeats = 10)

ctrl2 <- trainControl(method = "cv",
                      number = 10)

ctrl3 <- trainControl(method = "cv")
```

Scatter Plot use `feturePlot` on train data

```
# create dataset for exploratory analysis and data visualization
dat_train <- dat_train %>%
  mutate(study = case_when( # turn study (character variable) into a numeric variable
    study == "A" ~ 1,
    study == "B" ~ 2,
    study == "C" ~ 3))
# Find the remaining non-numeric columns
non_numeric_cols <- sapply(dat_train, function(x) !is.numeric(x))

# Convert non-numeric columns to numeric
dat_train[, non_numeric_cols] <- lapply(dat_train[, non_numeric_cols], as.numeric)
# turn factor variables into numeric variables
# set various graphical parameters (color, line type, background, etc)
# to control the look of trellis displays
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)
featurePlot(x = dat_train[ ,1:14],
            y = dat_train[ ,15],
            plot = "scatter",
            span = .5,
            labels = c("Predictors (Xs)", "COVID-19 Recovery Time (Y)"),
            main = "Lattice Plot",
            type = c("p", "smooth"))
```



preProcess in train()

```
# fit.lm <- train(x = train_x,
#                 y = train_y,
#                 preProcess = c("knnImpute"), # bagImpute/medianImpute
#                 method = "lm",
#                 trControl = trainControl(method = "none",
#                                          preProcOptions = list(k = 5)))
#
# pred.lm <- predict(fit.lm, newdata = test_x)
#
# mean((test_y - pred.lm)^2)
#
# # Imputation is performed within the resampling process
# fit.lm2 <- train(x = train_x,
#                  y = train_y,
#                  preProcess = c("knnImpute"),
#                  method = "lm",
#                  trControl = trainControl(method = "cv",
#                                           preProcOptions = list(k = 5)))
#
# pred.lm2 <- predict(fit.lm2, newdata = test_x)
#
# mean((test_y - pred.lm2)^2)
```

Model training

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

Linear regression model, KNN, ridge, lasso, lasso 1 se, elastic net, pls, gam, mars, bagging, random forest, boosting

1. linear model

```
## fit linear model on train data
linear_model <- train(train_x,
                      train_y,
                      method = "lm",
                      trControl = ctrl)
summary(linear_model)
```

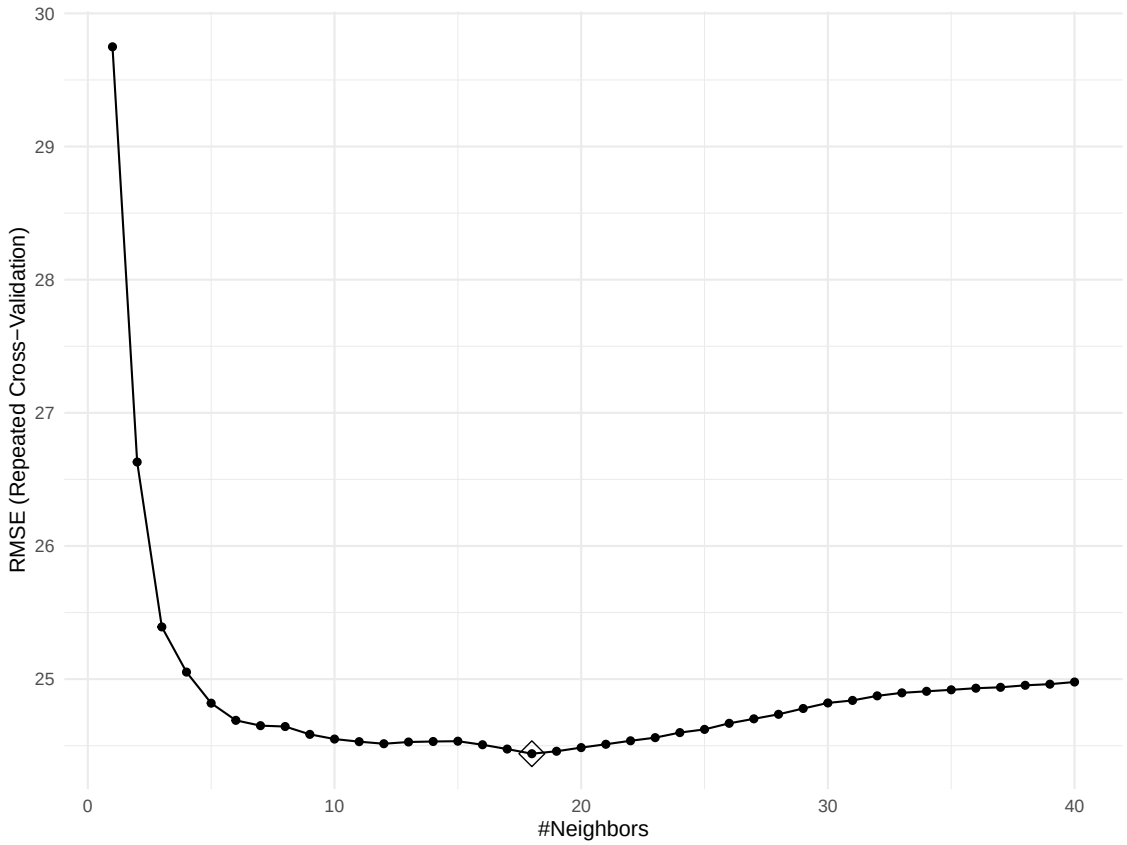
```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -75.437 -13.486  -1.977   9.491 228.042
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -2.579e+03  1.806e+02 -14.278  < 2e-16 ***
## age            1.318e-01  1.560e-01   0.845 0.398247
## gender1       -4.440e+00  1.264e+00  -3.512 0.000458 ***
## race2          3.940e+00  3.004e+00   1.312 0.189792
## race3         -1.614e+00  1.604e+00  -1.007 0.314255
## race4         -1.178e+00  2.196e+00  -0.536 0.591775
## smoking1       2.514e+00  1.432e+00   1.756 0.079286 .
## smoking2       8.192e+00  2.116e+00   3.871 0.000113 ***
## height         1.520e+01  1.064e+00  14.285  < 2e-16 ***
## weight        -1.663e+01  1.134e+00 -14.669  < 2e-16 ***
## bmi            4.991e+01  3.230e+00  15.451  < 2e-16 ***
## hypertension1  5.640e+00  2.183e+00   2.584 0.009863 **
## diabetes1     -5.910e-01  1.723e+00  -0.343 0.731684
## SBP           -1.558e-01  1.422e-01  -1.095 0.273519
## LDL           -4.769e-02  3.358e-02  -1.420 0.155796
## vaccine1      -7.641e+00  1.295e+00  -5.900 4.56e-09 ***
## severity1      1.741e+00  2.183e+00   0.797 0.425458
## studyB         4.771e+00  1.612e+00   2.961 0.003123 **
## studyC         1.541e-01  1.997e+00   0.077 0.938476
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.55 on 1384 degrees of freedom
## Multiple R-squared:   0.27,  Adjusted R-squared:  0.2605
## F-statistic: 28.43 on 18 and 1384 DF,  p-value: < 2.2e-16
```

```
# view performance on the test set (RMSE)
test_pred1 <- predict(linear_model, newdata = test_x) # test dataset
test_rmse1 <- sqrt(mean((test_pred1 - test_y)^2))
test_rmse1
```

```
## [1] 23.54638
```

2. KNN

```
# fit knn on train data use caret
kGrid <- expand.grid(k = seq(1, to = 40, by = 1))
knn_model <- train(train_x,
                   train_y,
                   method = "knn",
                   trControl = ctrl,
                   tuneGrid = kGrid)
ggplot(knn_model, highlight = TRUE)
```

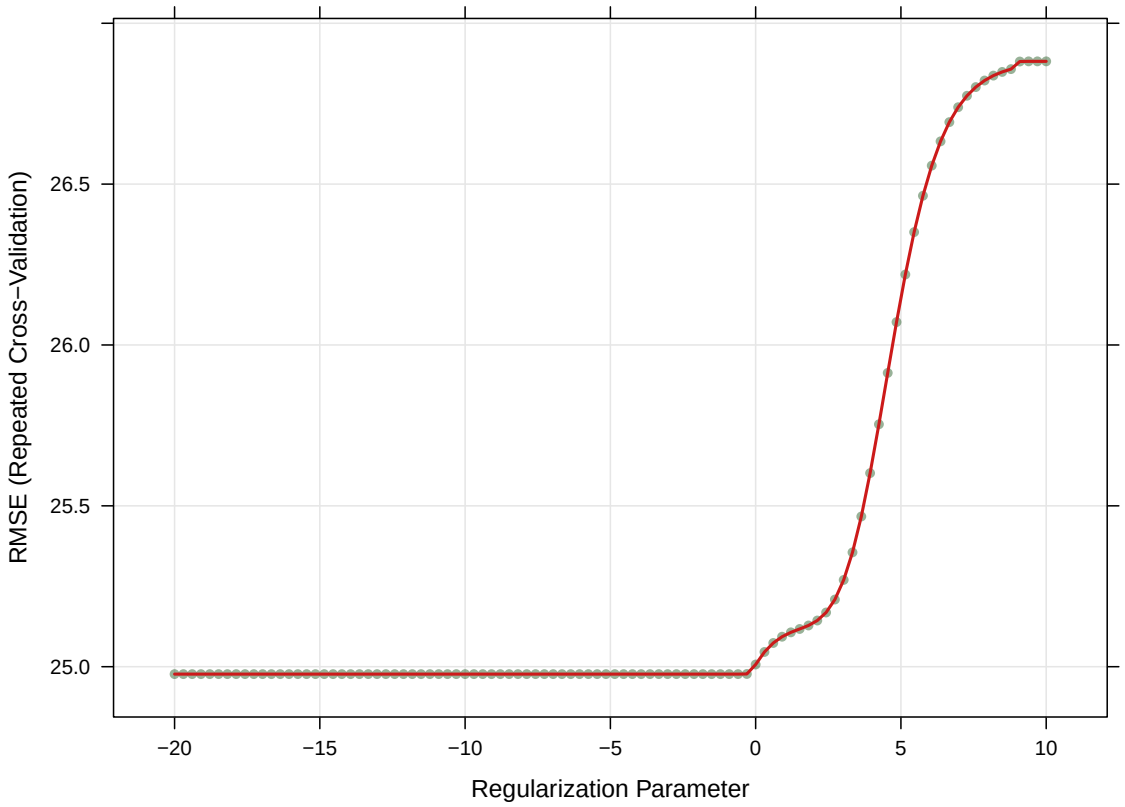



```
# knn with K = 18 was selected as the final model

# view performance on the test set (RMSE)
test_pred2 <- predict(knn_model, newdata = test_x) # test dataset
test_rmse2 <- sqrt(mean((test_pred2 - test_y)^2))
test_rmse2
```

```
## [1] 24.39294
```

3. Ridge regression

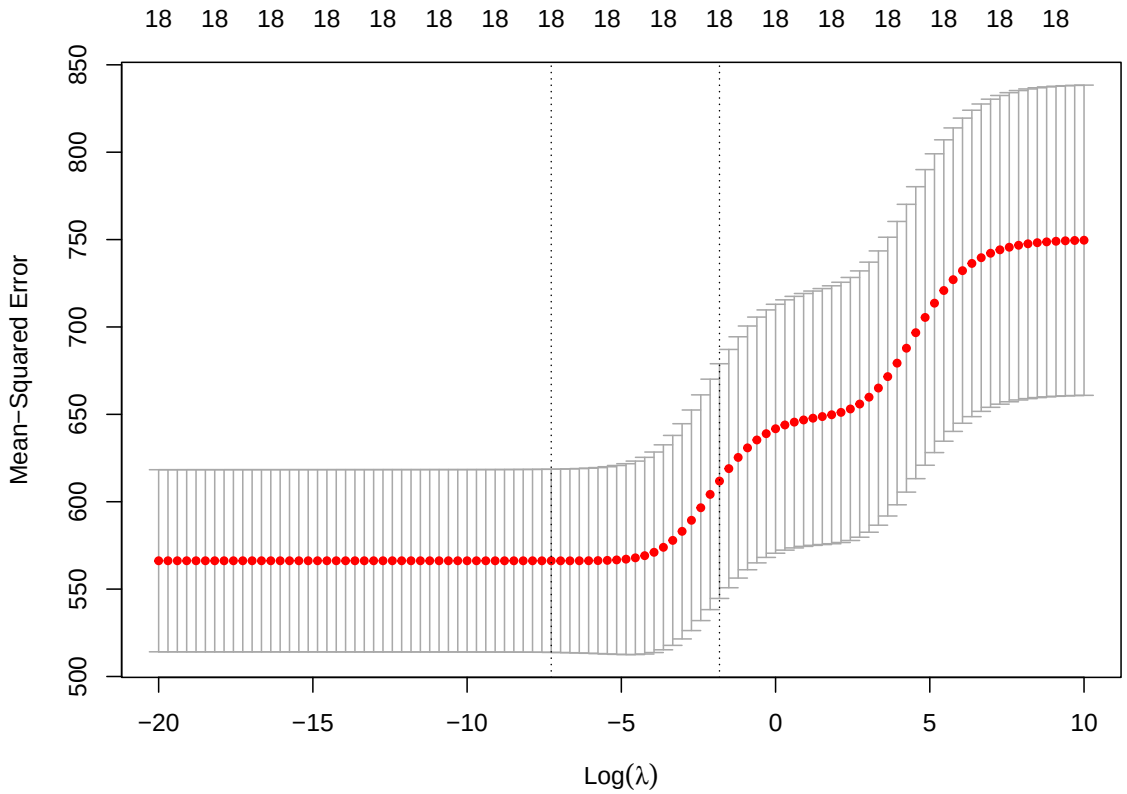
```
## fit ridge use caret
ridge_model <- train(train_x,
                     train_y,
                     method = "glmnet",
                     tuneGrid = expand.grid(alpha = 0,
                                            lambda = exp(seq(-20, 10, length = 100))),
                     trControl = ctrl)
plot(ridge_model, xTrans = log)
```




```
ridge_model$bestTune
```

```
##    alpha    lambda
## 65     0 0.5454956
```

```
## ridge use glmnet
cv.ridge_fit <- cv.glmnet(train_x,
                     train_y,
                     alpha = 0,
                     lambda = exp(seq(-20, 10, length = 100)))
plot(cv.ridge_fit)
```

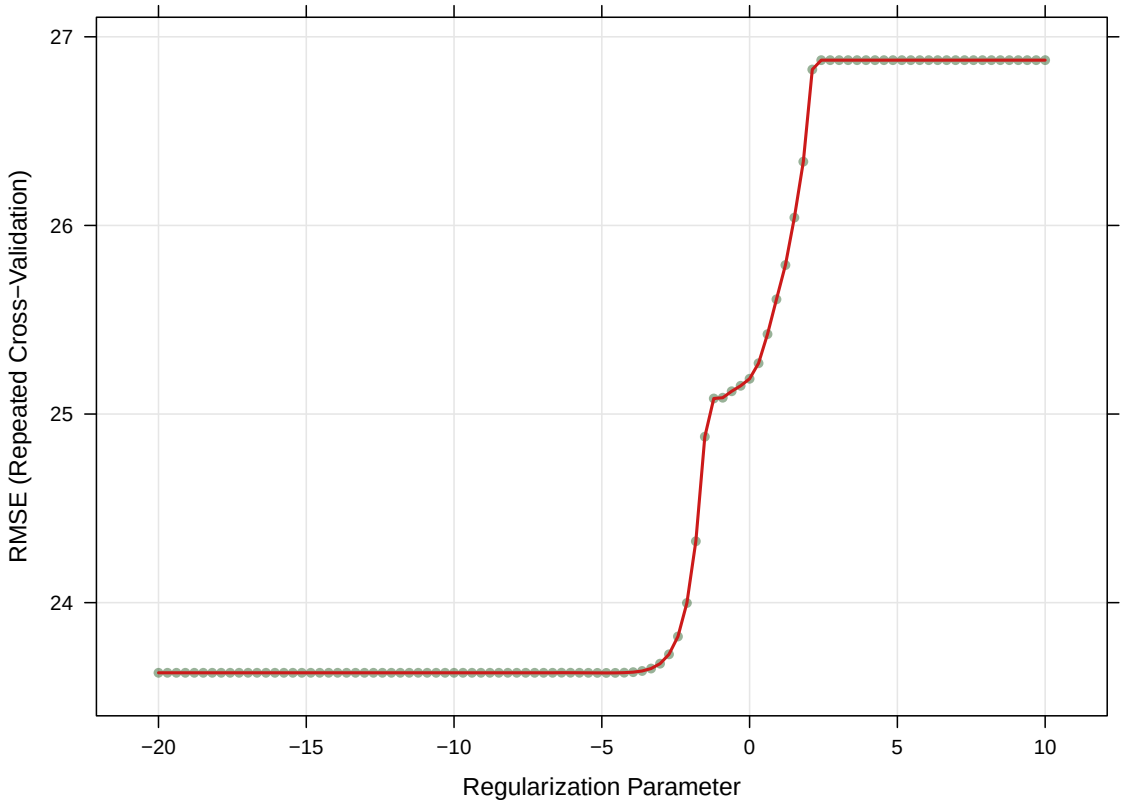



```
# view performance on the test set (RMSE)
test_pred3 <- predict(ridge_model, newdata = test_x) # test dataset
test_rmse3 <- sqrt(mean((test_pred3 - test_y)^2))
test_rmse3
```

```
## [1] 24.51672
```

4. Lasso regression

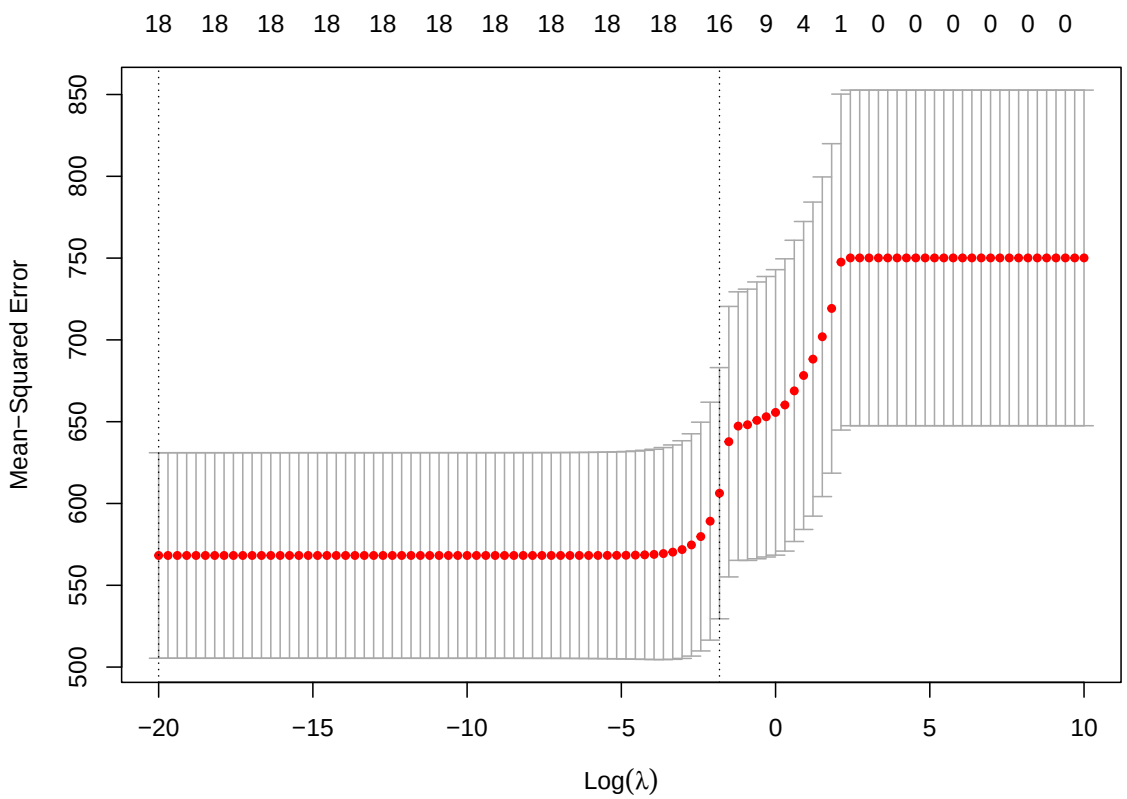
```
## fit lasso use caret
lasso_model <- train(train_x, train_y,
                     method = "glmnet",
                     tuneGrid = expand.grid(alpha = 1,
                                            lambda = exp(seq(-20, 10, length = 100))),
                     trControl = ctrl)
plot(lasso_model, xTrans = log)
```




```
lasso_model$bestTune
```

```
##    alpha      lambda
## 51     1 0.007840248
```

```
## fit lasso use glmnet
cv.lasso_fit <- cv.glmnet(train_x,
                          train_y,
                          alpha = 1,
                          lambda = exp(seq(-20, 10, length = 100)))
plot(cv.lasso_fit)
```

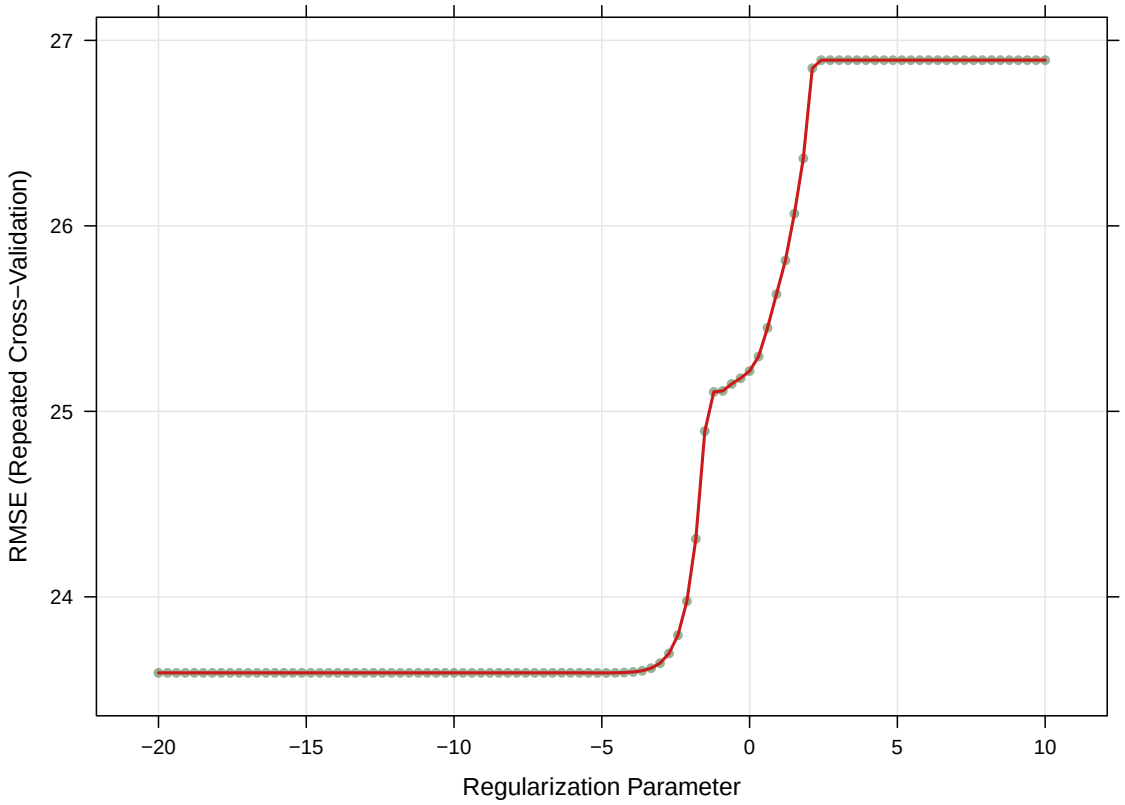



```
# view performance on the test set (RMSE)
test_pred4 <- predict(lasso_model, newdata = test_x) # test dataset
test_rmse4 <- sqrt(mean((test_pred4 - test_y)^2))
test_rmse4
```

```
## [1] 23.50932
```

5. Lasso 1se regression

```
lasso_1se <- train(train_x, train_y,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(-20, 10, length = 100))),
                   trControl = ctrl1)
plot(lasso_1se, xTrans = log)
```




```
lasso_1se$bestTune
```

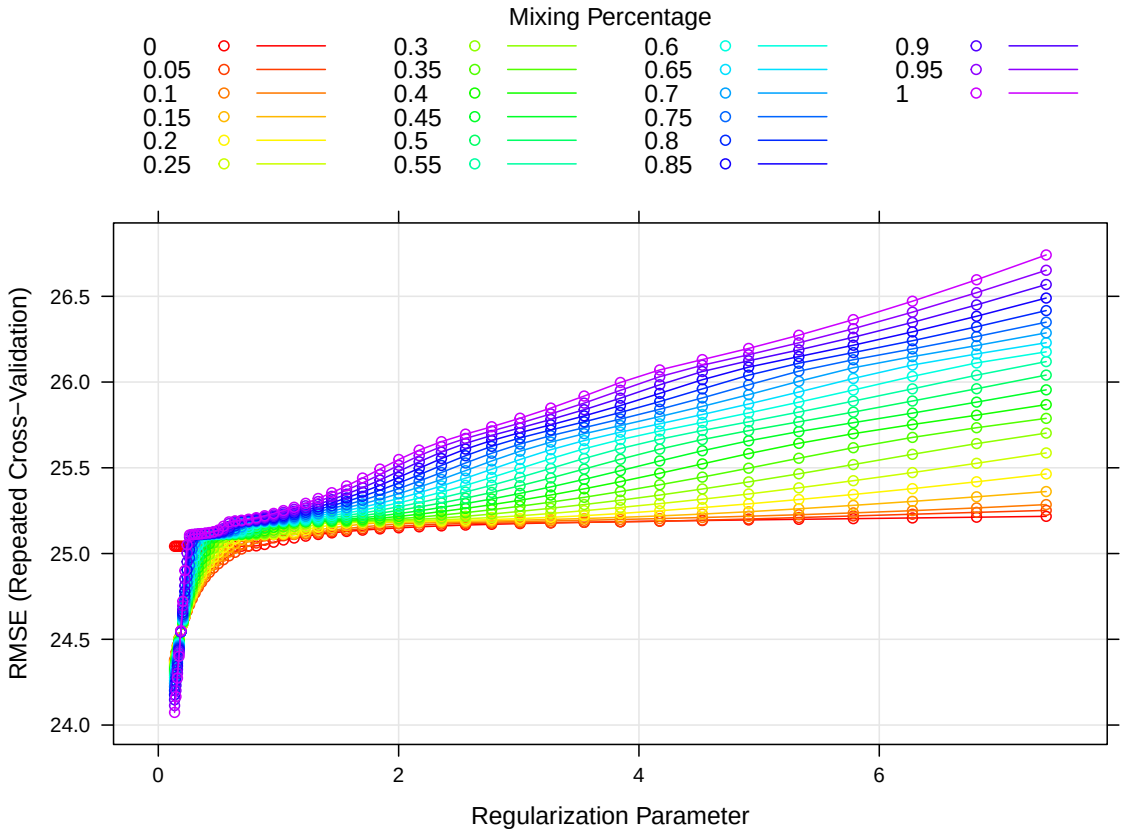
```
##    alpha     lambda
## 59     1 0.08854517
```

```
# view performance on the test set (RMSE)
test_pred5 <- predict(lasso_1se, newdata = test_x) # test dataset
test_rmse5 <- sqrt(mean((test_pred5 - test_y)^2))
test_rmse5
```

```
## [1] 23.48364
```

6. Elastic net

```
## fit elastic net
enet_model <- train(train_x,
                    train_y,
                    method = "glmnet",
                    tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                           lambda = exp(seq(-2, 2, length = 50))),
                    trControl = ctrl)

myCol <- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
                      superpose.line = list(col = myCol))
plot(enet_model, par.settings = myPar)
```




```
enet_model$bestTune
```

```
##      alpha    lambda
## 1001     1 0.1353353
```

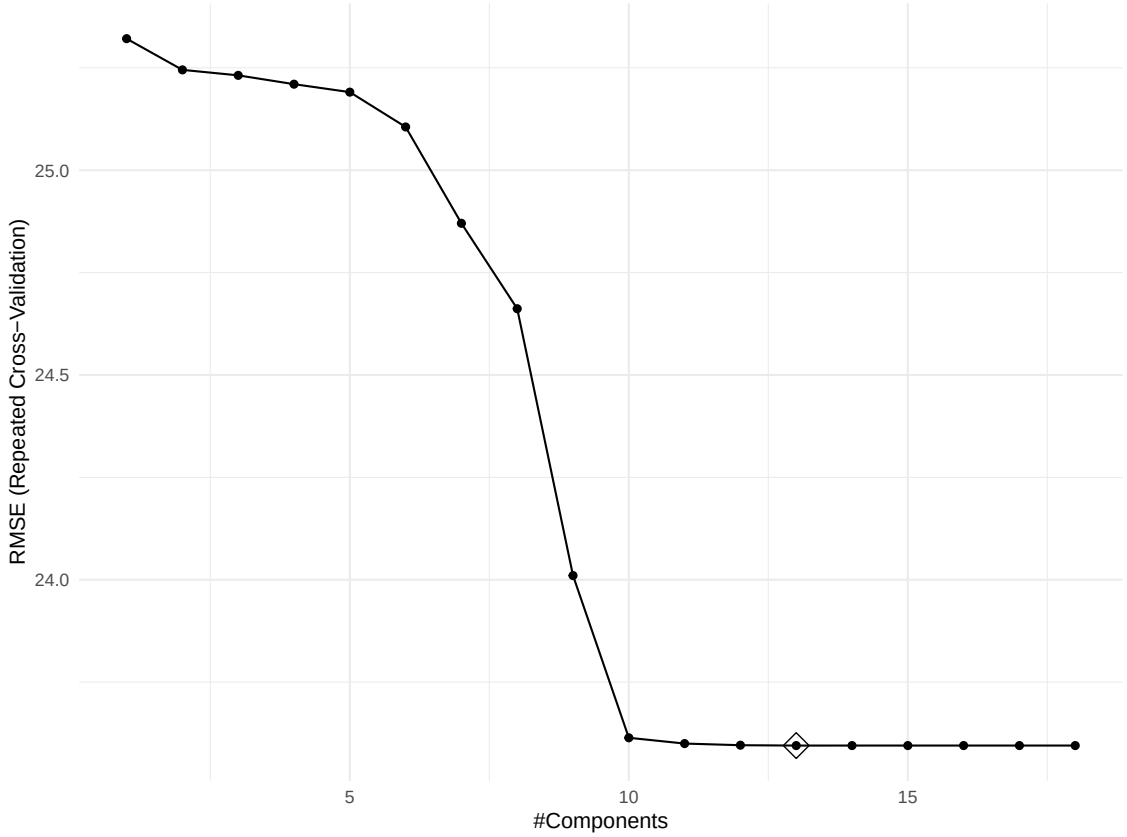
```
# enet_2 <- train(train_x, train_y,
#                     method = "glmnet",
#                     tuneGrid = expand.grid(alpha = seq(0, 1, length = 10),
#                                            lambda = exp(seq(-30,5, length = 100))),
#                     trControl = ctrl)
# plot(enet_2, par.settings = myPar)
# enet_2$bestTune

# view performance on the test set (RMSE)
test_pred6 <- predict(enet_model, newdata = test_x) # test dataset
test_rmse6 <- sqrt(mean((test_pred6 - test_y)^2))
test_rmse6
```

```
## [1] 23.66684
```

7. Partial least squares regression

```
pls_model <- train(train_x,
                   train_y,
                   method = "pls",
                   tuneGrid = data.frame(ncomp = 1:18),
                   trControl = ctrl,
                   preProcess = c("center", "scale"))
```

```
ggplot(pls_model, highlight = TRUE)
```




```
ggsave(file="image/pls_number_of_component.png",width=10,height=7)

# view performance on the test set (RMSE)
test_pred7 <- predict(pls_model, newdata = test_x) # test dataset
test_rmse7 <- sqrt(mean((test_pred7 - test_y)^2))
test_rmse7
```

```
## [1] 23.5464
```

8. Generalised additive regression

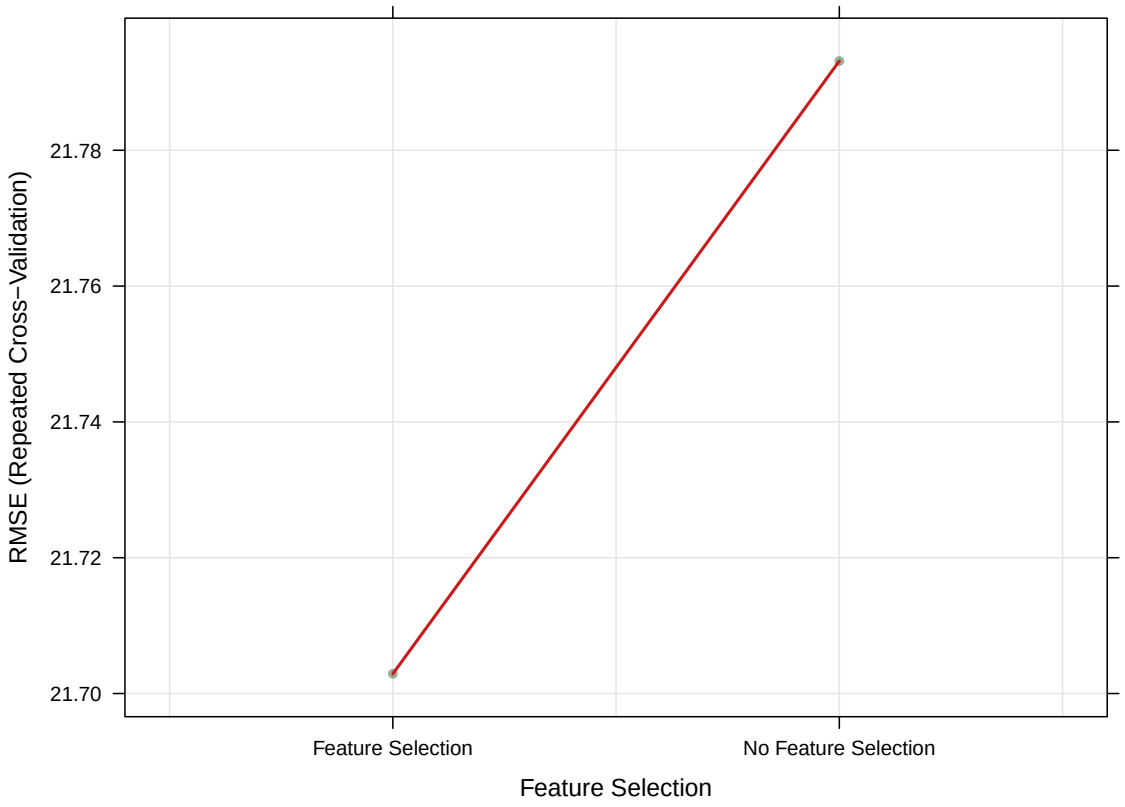
```
# gam use caret
# fit GAM model using all predictors
gam_model <- train(train_x, train_y, # training dataset
                   method = "gam",
                   trControl = ctrl,
                   control = gam.control(maxit = 200))

# gam_model <- train(train_x,
#                    train_y,
#                    trControl = ctrl2,
#                    control = gam.control(maxit =100))

gam_model$bestTune
```

```
##   select method
## 2   TRUE GCV.Cp
```

```
gam_model$finalModel
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ gender1 + race3 + race4 + smoking1 + smoking2 + hypertension1 +
##     diabetes1 + vaccine1 + severity1 + studyB + studyC + s(age) +
##     s(SBP) + s(LDL) + s(bmi) + s(height) + s(weight)
##
## Estimated degrees of freedom:
## 0.000 5.031 0.253 7.536 1.838 0.000  total = 26.66
##
## GCV score: 443.6718
```

```
plot(gam_model)
```




```
summary(gam_model$finalModel)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ gender1 + race3 + race4 + smoking1 + smoking2 + hypertension1 +
##     diabetes1 + vaccine1 + severity1 + studyB + studyC + s(age) +
```

```
##     s(SBP) + s(LDL) + s(bmi) + s(height) + s(weight)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    44.6541     2.1776  20.506  < 2e-16 ***
## gender1        -4.9874     1.1275  -4.423 1.05e-05 ***
## race3          -1.0788     1.4146  -0.763  0.44581
## race4          -3.1360     1.9439  -1.613  0.10692
## smoking1        3.8020     1.2744   2.983  0.00290 **
## smoking2        7.8978     1.8810   4.199 2.86e-05 ***
## hypertension1   2.5157     3.0870   0.815  0.41525
## diabetes1       1.4298     1.5307   0.934  0.35043
## vaccine1       -7.6852     1.1497  -6.684 3.36e-11 ***
## severity1       3.0003     1.9374   1.549  0.12169
## studyB          3.7248     1.4328   2.600  0.00943 **
## studyC          0.1299     1.7720   0.073  0.94159
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                 edf Ref.df      F p-value
## s(age)    5.629e-07      9  0.000  0.5748
## s(SBP)    5.031e+00      9  0.783  0.2005
## s(LDL)    2.531e-01      9  0.037  0.2477
## s(bmi)    7.536e+00      9 80.723  <2e-16 ***
## s(height) 1.838e+00      9  0.460  0.0832 .
## s(weight) 8.572e-07      9  0.000  0.7901
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.419   Deviance explained =   43%
## GCV = 443.67  Scale est. = 435.24    n = 1403
```

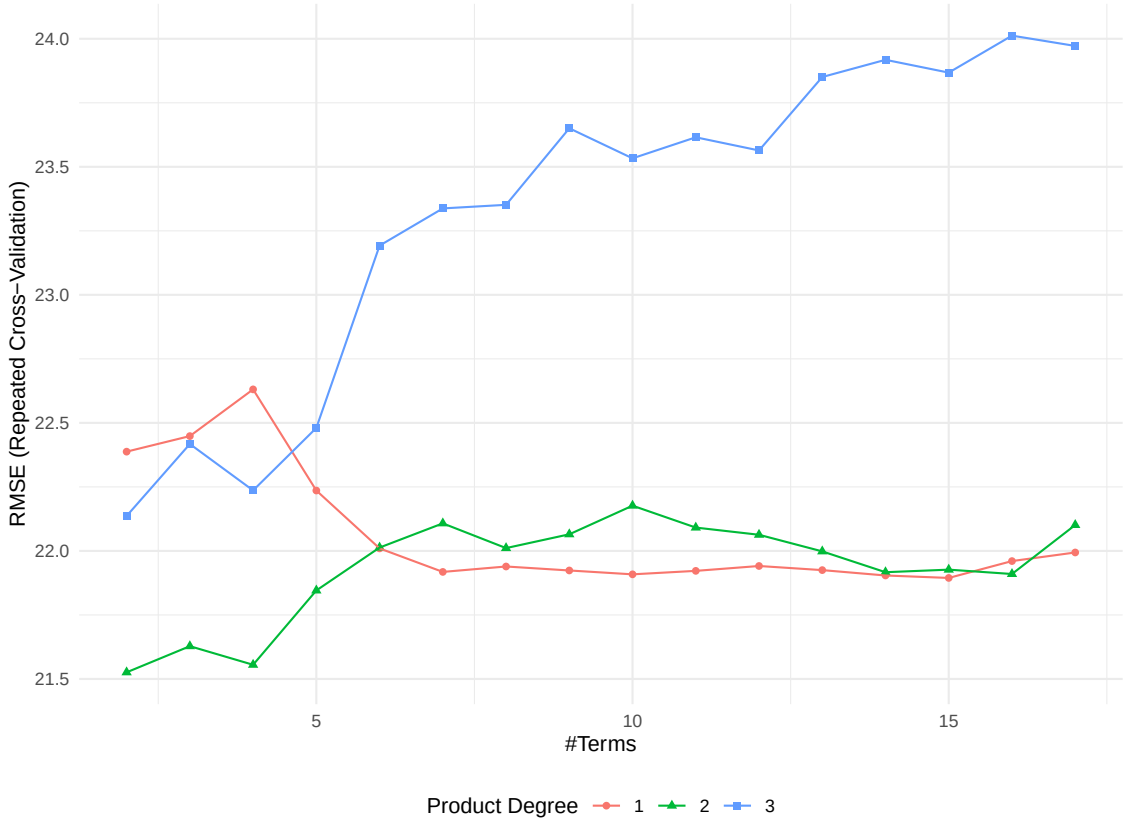
```
# view performance on the test set (RMSE)
test_pred8 <- predict(gam_model, newdata = test_x) # test dataset
test_rmse8 <- sqrt(mean((test_pred8 - test_y)^2))
test_rmse8
```

```
## [1] 22.47637
```

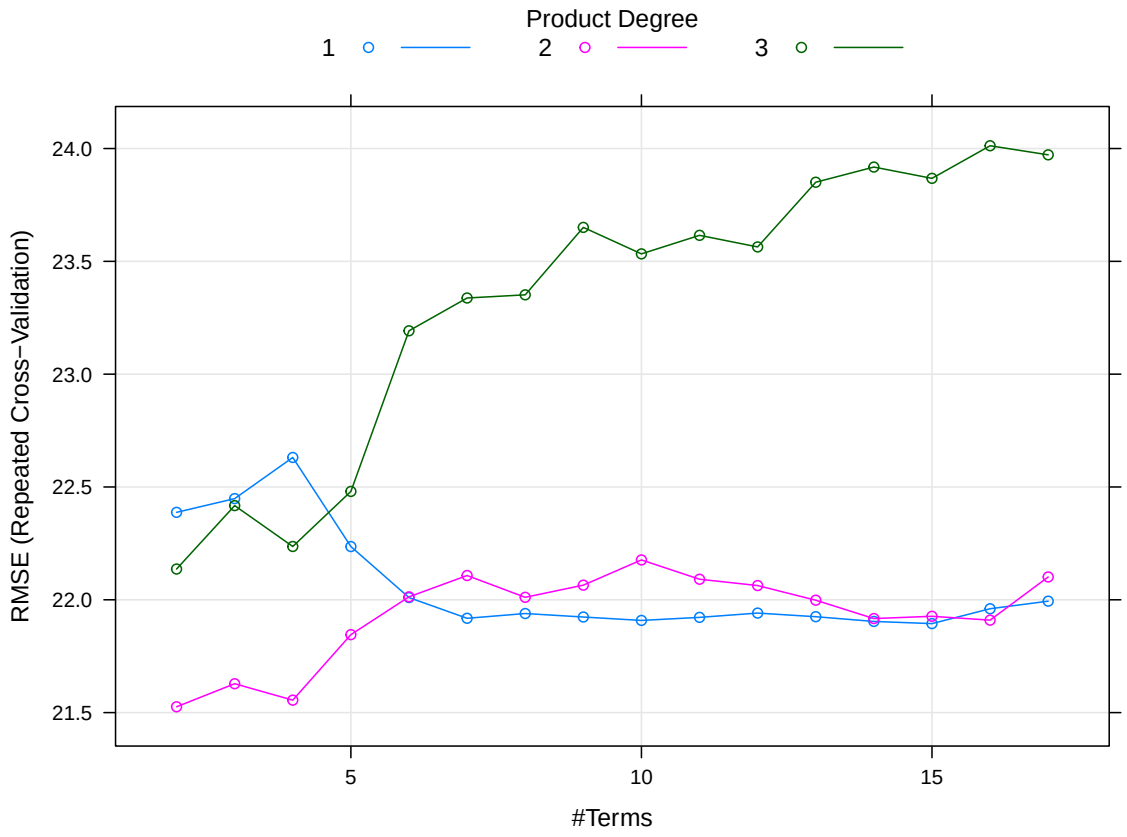
9. Multivariate adaptive regression

```
set.seed(3554)

# create dummy variables for categorical variables
df_dummies <- data.frame(model.matrix(~ . - 1,
                                      # exclude ID and continuous variables
                                      data = dat[, c("gender", "race", "smoking", "hypertension", "diabe
                                                     "vaccine", "severity", "study")]),
                         # add continuous variables back to the data frame
                         age = dat$age,
                         height = dat$height,
                         weight = dat$weight,
                         bmi = dat$bmi,
```

```
                          SBP = dat$SBP,
                          LDL = dat$LDL,
                          recovery_time = dat$recovery_time)

# rename df_dummies dataset as dat
dat_mars <- df_dummies

# training data
dat_train_mars <- dat_mars[train_rows, ]
x_mars <- model.matrix(recovery_time~.,dat_mars)[train_rows,-1]
y_mars <- dat_mars$recovery_time[train_rows]

# test data
dat_test_mars <- dat_mars[-train_rows, ]
x2_mars <- model.matrix(recovery_time~.,dat_mars)[-train_rows,-1]
y2_mars <- dat_mars$recovery_time[-train_rows]

# mars_grid <- expand.grid(degree = 1:5, # number of possible product hinge functions in 1 term
#                          nprune = -5:17) # upper bound of number of terms in model

# create grid of all possible pairs that can take degree and nprune values
mars_grid <- expand.grid(degree = 1:3, # number of possible product hinge functions in 1 term
                         nprune = 2:17) # upper bound of number of terms in model
mars_model <- train(x_mars, y_mars, # training dataset
                    method = "earth",
                    tuneGrid = mars_grid,
                    trControl = ctrl)
ggplot(mars_model)
```




```
print(plot(mars_model))
```



```
summary(mars_model$finalModel)
```

```
## Call: earth(x=matrix[1403,19], y=c(14,36,50,65,2...), keepxy=TRUE, degree=2,
##             nprune=2)
##
##                      coefficients
## (Intercept)             39.49104
## studyB * h(bmi-31.3)    32.34752
##
## Selected 2 of 30 terms, and 2 of 19 predictors (nprune=2)
## Termination condition: Reached nk 39
## Importance: studyB, bmi, gender0-unused, gender1-unused, race2-unused, ...
## Number of terms at each degree of interaction: 1 0 1
## GCV 455.5987    RSS 636019.8    GRSq 0.3927496    RSq 0.3949133
```

```
# view performance on the test set (RMSE)
test_pred9 <- predict(mars_model, newdata = x2_mars) # test dataset
test_rmse9 <- sqrt(mean((test_pred9 - dat_test_mars$recovery_time)^2))
test_rmse9
```

```
## [1] 23.15086
```

10. Bagging

```
set.seed(3554)
bag_model <- randomForest(train_x,
                          train_y,
                          mtry = 18)

# view performance on the test set (RMSE)
test_pred10 <- predict(bag_model, newdata = test_x) # test dataset
test_rmse10 <- sqrt(mean((test_pred10 - test_y)^2))
test_rmse10
```

```
## [1] 23.08393
```

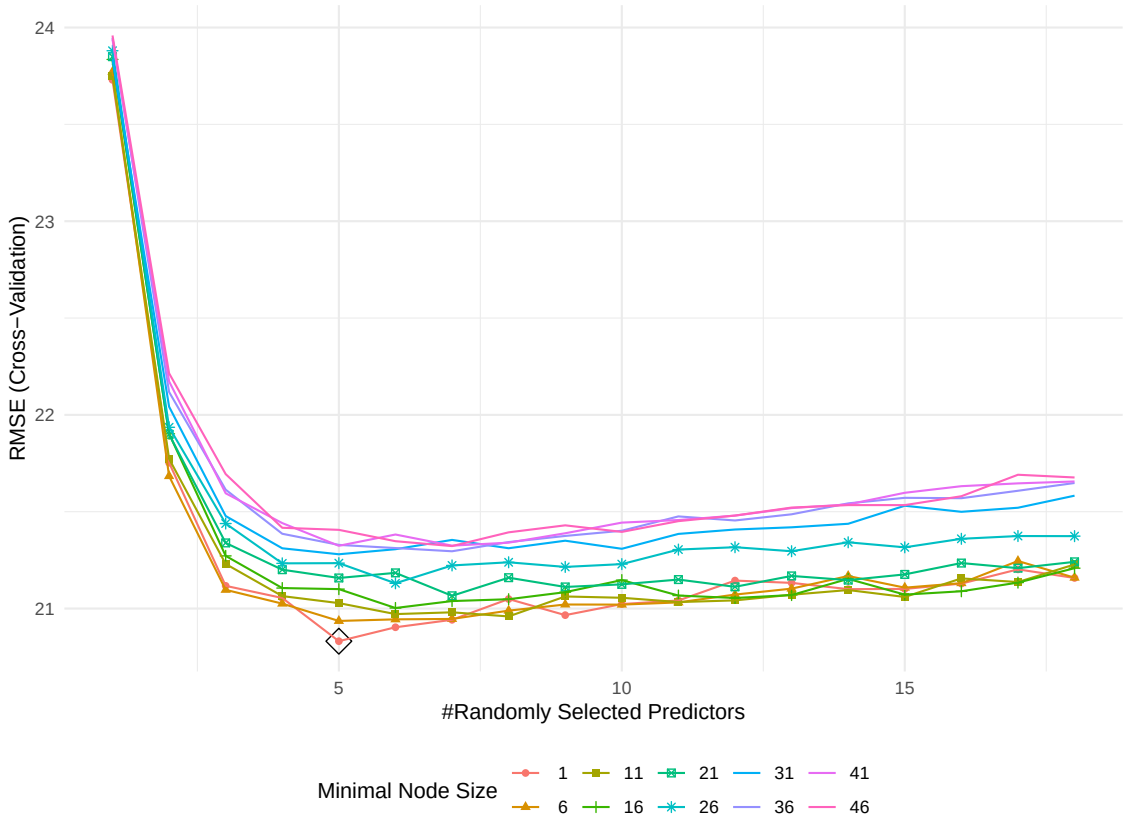
11. Random forest

```
set.seed(3554)
rf_fit <- randomForest(train_x,
                       train_y,
                       mtry = 6)

# view performance on the test set (RMSE)
test_pred11 <- predict(rf_fit, newdata = test_x) # test dataset
test_rmse11 <- sqrt(mean((test_pred11 - test_y)^2))
test_rmse11
```

```
## [1] 22.56521
```

```
# dat3 <- dat
# # use caret
# # Try more if possible
# rf.grid <- expand.grid(mtry = 1:18,
#                        splitrule = "variance",
```

```
#                         min.node.size = 1:6)
# set.seed(3554)
# rf.model <- train(recovery_time~.,
#                   dat3[train_rows,],
#                 method = "ranger",
#                 tuneGrid = rf.grid,
#                 trControl = ctrl3)
#
# ggplot(rf.model, highlight = TRUE)

rf.grid <- expand.grid(mtry = 1:18,
                       splitrule = "variance",
                       min.node.size = seq(from = 1, to = 50, by = 5))
rf_model <- train(train_x,
                  train_y,
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl3)
ggplot(rf_model, highlight = TRUE)
```




```
rf_model$bestTune
```

```
##    mtry splitrule min.node.size
## 41    5  variance             1
```

```
# view performance on the test set (RMSE)
test_pred_rf <- predict(rf_model, newdata = test_x) # test dataset
test_rmse_rf <- sqrt(mean((test_pred_rf - test_y)^2))
```

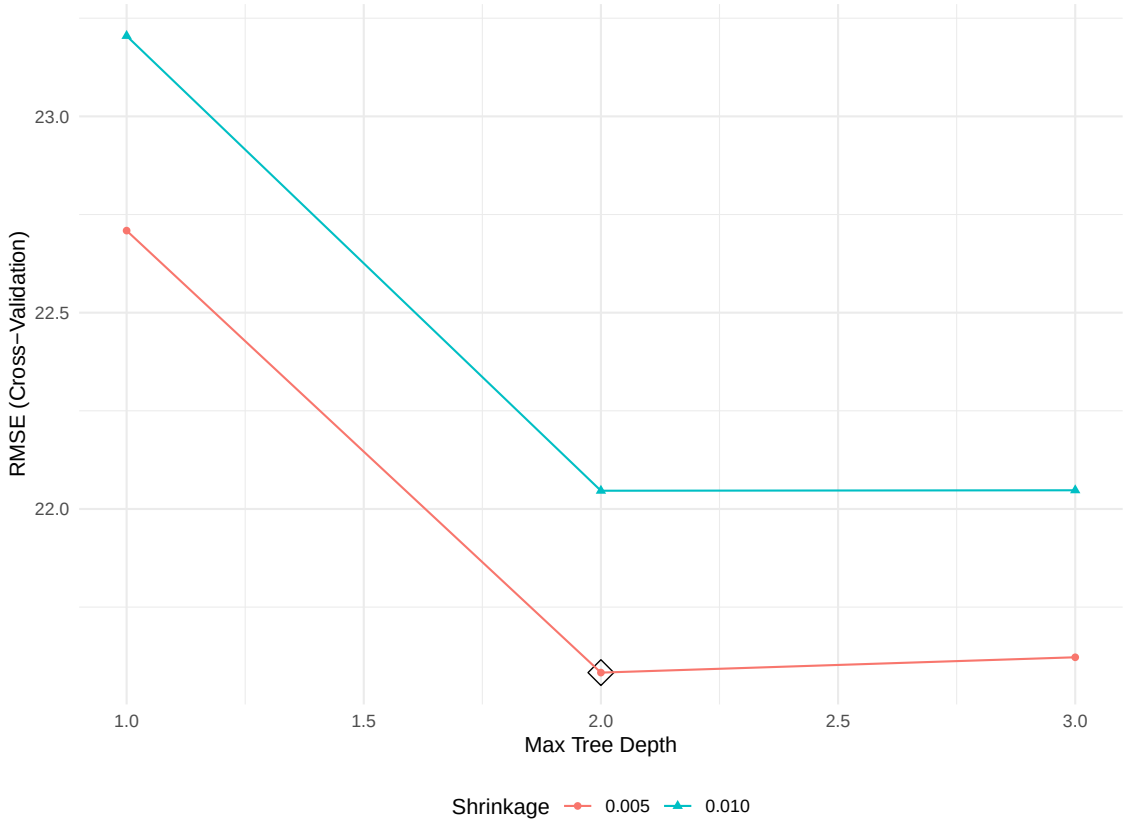
```
test_rmse_rf
```

```
## [1] 22.39369
```

12. Boosting

```
set.seed(3554)
# use gbm() function
# dat3 <- dat
# bst_fit <- gbm(recovery_time~.,
#                  dat3[train_rows,],
#                  distribution = "gaussian",
#                  n.trees = 5000,
#                  interaction.depth = 2,
#                  shrinkage = 0.005,
#                  cv.folds = 10,
#                  n.cores = 2)
# gbm.perf(bst_fit, method = "cv")
# plot(bst_fit)
#
# # view performance on the test set (RMSE)
# test_pred12 <- predict(bst_fit, newdata = dat3[-train_rows,]) # test dataset
# test_rmse12 <- sqrt(mean((test_pred12 - test_y)^2))
# test_rmse12

# # use caret
# gbm.grid <- expand.grid(n.trees = c(5000,10000),
#                         interaction.depth = 1:3,
#                         shrinkage = c(0.005,0.01),
#                         n.minobsinnode = c(1))
#
# bst_model <- train(recovery_time~.,
#                    dat3[train_rows,],
#                    method = "gbm",
#                    tuneGrid = gbm.grid,
#                    trControl = ctrl3,
#                    verbose = FALSE)
#
# ggplot(gbm_model, highlight = TRUE)
#
# # view performance on the test set (RMSE)
# test_pred13 <- predict(bst_model, newdata = dat3[-train_rows,]) # test dataset
# test_rmse13 <- sqrt(mean((test_pred13 - test_y)^2))
# test_rmse13

gbm.grid <- expand.grid(n.trees = 5000,
                        interaction.depth = 1:3,
                        shrinkage = c(0.005,0.01),
                        n.minobsinnode = c(1))
gbm_model <- train(train_x,
                   train_y,
                  method = "gbm",
                  tuneGrid = gbm.grid,
                  trControl = ctrl3,
```

```
                  verbose = FALSE)

ggplot(gbm_model, highlight = TRUE)
```




```
# view performance on the test set (RMSE)
test_pred13 <- predict(gbm_model, newdata = test_x) # test dataset
test_rmse13 <- sqrt(mean((test_pred13 - test_y)^2))
test_rmse13
```

```
## [1] 21.8394
```

Results and Discussion:

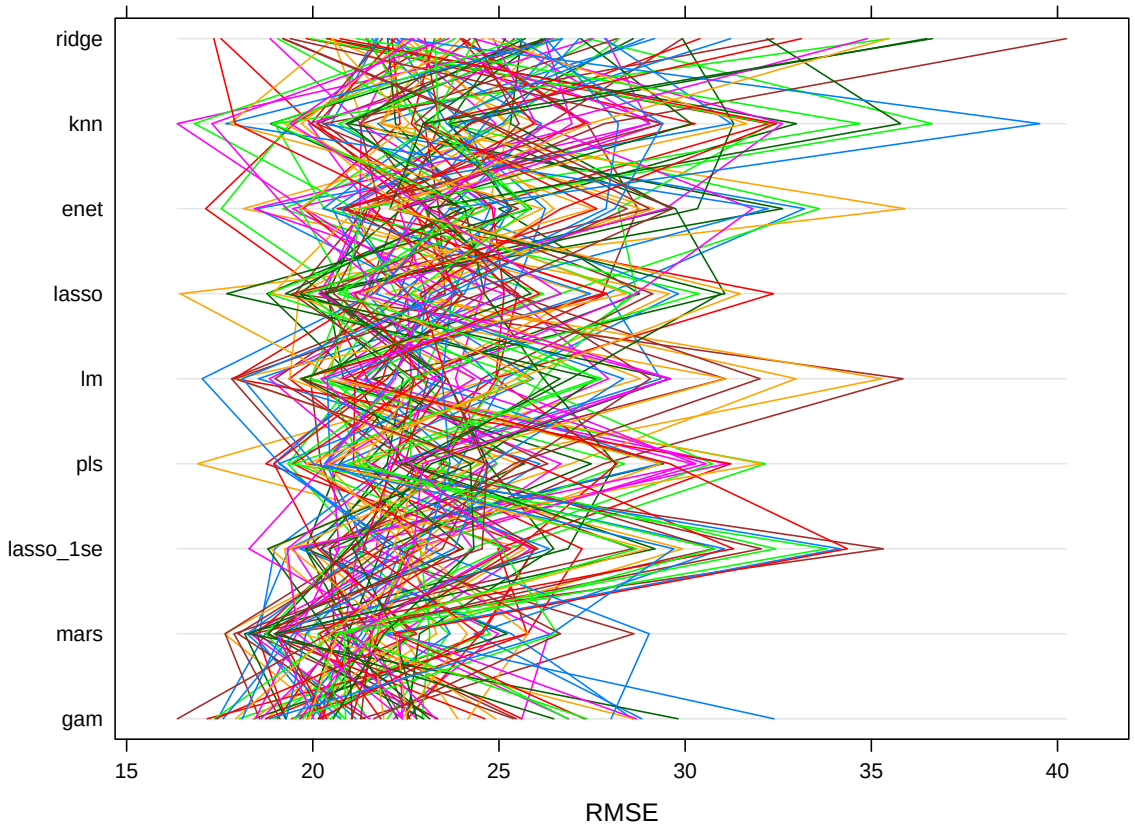
In this section, report the final model that you built for predicting time to recovery from COVID-19. Interpret the results. Assess the model's training/test performance.

Select model use CV result on train data

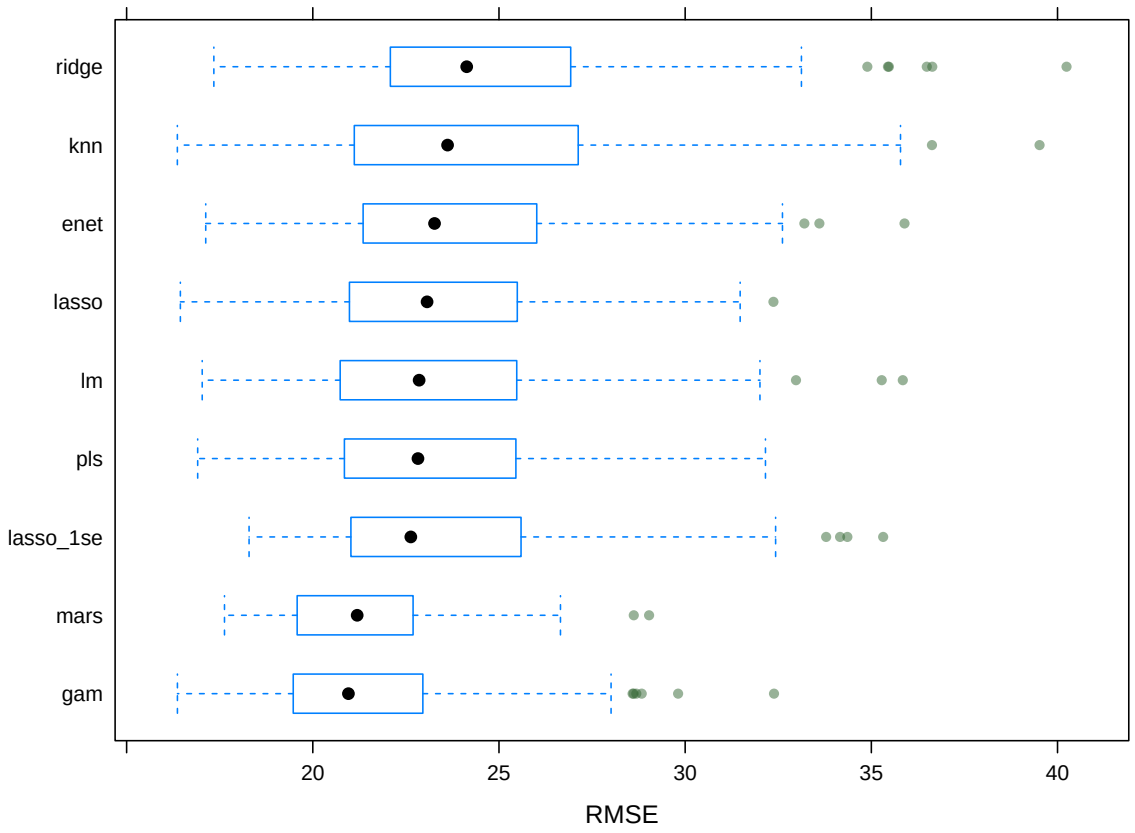
Comparing linear regression model, KNN, ridge, lasso, lasso 1 se, elastic net, pls, gam, mars, regression trees, bagging, random forest, boosting using cv results on train data.

```
set.seed(3554)

resamp <- resamples(list(lm = linear_model,
                         knn = knn_model,
                         ridge = ridge_model,
                         lasso = lasso_model,
                         lasso_1se = lasso_1se,
                         enet = enet_model,
```

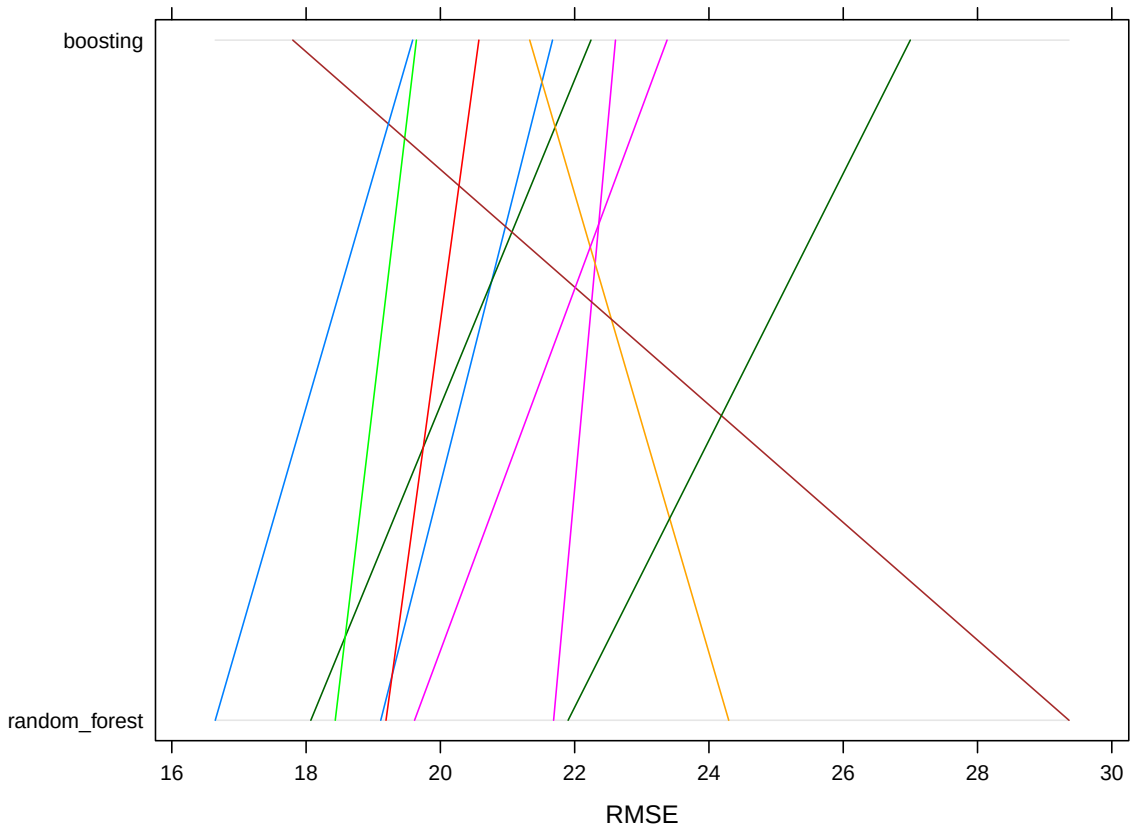
```
                         pls = pls_model,
                         gam = gam_model,
                         mars = mars_model))
# summary(resamp)
parallelplot(resamp, metric = "RMSE")
```



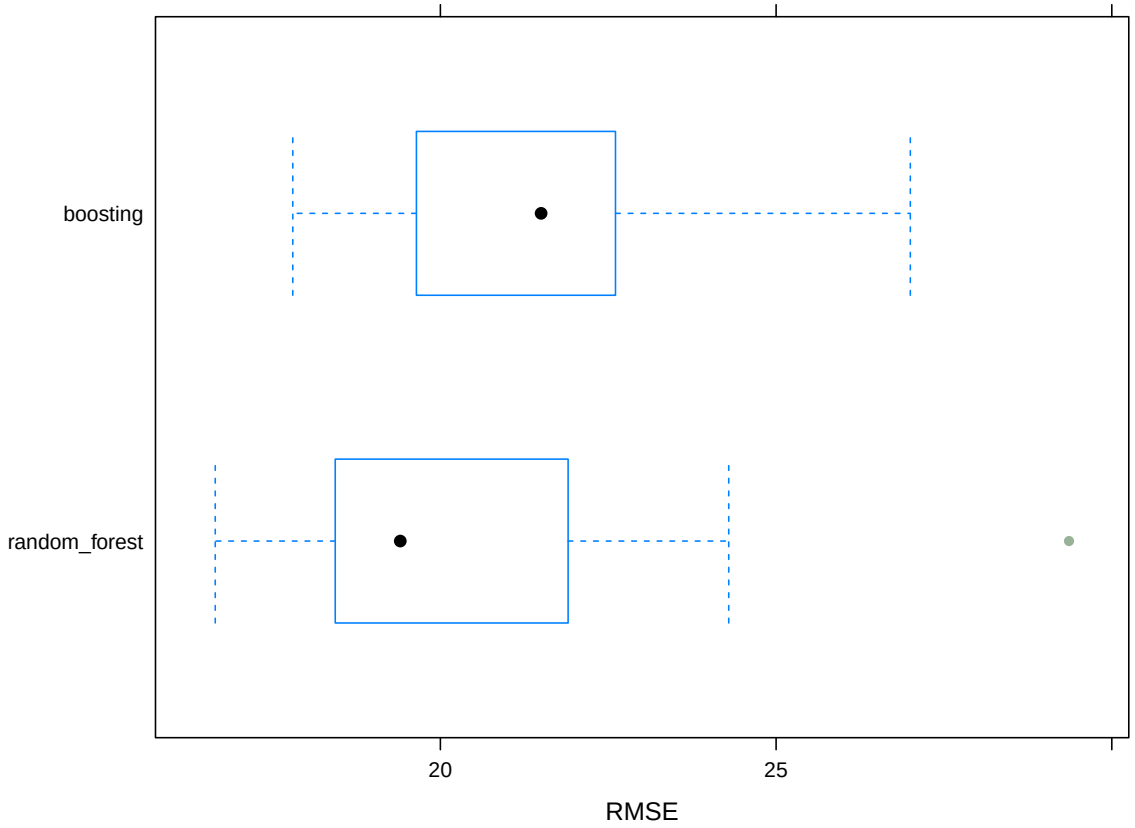

```
bwplot(resamp, metric = "RMSE")
```




```
resamp2 <- resamples(list(random_forest = rf_model,
                          boosting = gbm_model))
# summary(resamp2)
parallelplot(resamp2, metric = "RMSE")
```




```
bwplot(resamp2, metric = "RMSE")
```



```
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lm, knn, ridge, lasso, lasso_1se, enet, pls, gam, mars
## Number of resamples: 100
##
## MAE
##               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm        13.25527 15.31284 16.18176 16.19196 16.96599 20.18931    0
## knn       11.95675 14.55670 15.38613 15.51119 16.45506 19.47642    0
## ridge     13.17652 15.60712 16.45497 16.48082 17.27249 19.49263    0
## lasso     12.53397 15.21661 16.15249 16.15601 17.01374 18.99334    0
## lasso_1se 13.42755 15.07887 15.94312 16.00990 16.71698 19.88082    0
## enet      12.91408 15.21466 16.11015 16.01797 16.58587 19.94585    0
## pls       13.37905 15.22017 16.01148 16.18143 17.03679 18.90724    0
## gam       12.56935 14.22449 14.99938 14.96347 15.74925 17.86917    0
## mars      12.92768 14.30993 14.93712 15.05904 15.66805 17.43668    0
##
## RMSE
##               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm        17.02997 20.74039 22.85549 23.54169 25.47095 35.84485    0
## knn       16.36248 21.18660 23.61918 24.43978 27.04378 39.51748    0
## ridge     17.34282 22.10545 24.13304 24.97694 26.81373 40.24213    0
## lasso     16.44313 20.99082 23.06880 23.62749 25.48268 32.37075    0
## lasso_1se 18.28886 21.02407 22.63259 23.79331 25.54317 35.31773    0
## enet      17.12464 21.36901 23.26906 24.07358 25.94086 35.89088    0
## pls       16.90868 20.89813 22.82448 23.59512 25.43083 32.15666    0
## gam       16.36429 19.47772 20.95709 21.70290 22.94316 32.38591    0
## mars      17.62860 19.58672 21.19206 21.52585 22.68794 29.03128    0
##
## Rsquared
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lm        0.018796215 0.1809941 0.2400568 0.2495059 0.3105276 0.5040570    0
## knn       0.002135919 0.1272458 0.1994278 0.2150528 0.3169047 0.4488650    0
## ridge     0.029571044 0.1151916 0.1434132 0.1498434 0.1871175 0.2898008    0
## lasso     0.013187603 0.1773322 0.2403706 0.2465342 0.3039978 0.5415068    0
## lasso_1se 0.041274137 0.1631339 0.2267068 0.2329225 0.2922317 0.5580490    0
## enet      0.057334425 0.1708427 0.2090892 0.2188720 0.2647018 0.4387715    0
## pls       0.014350229 0.1696385 0.2402519 0.2459587 0.3018495 0.5521645    0
## gam       0.076645559 0.2795984 0.3787056 0.3646757 0.4401829 0.6315266    0
## mars      0.006565592 0.2458449 0.3563683 0.3591779 0.4888207 0.7427425    0
```

```
summary(resamp2)
```

```
##
## Call:
## summary.resamples(object = resamp2)
##
## Models: random_forest, boosting
## Number of resamples: 10
##
```

```
## MAE
##                    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## random_forest 12.03393 13.48289 14.18241 14.33381 15.37494 16.03850    0
## boosting      13.63396 14.13914 14.66023 14.56597 15.05770 15.14378    0
##
## RMSE
##                   Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## random_forest 16.6469 18.60361 19.40277 20.83151 21.84843 29.36269    0
## boosting      17.8019 19.87561 21.50028 21.58318 22.51652 26.99898    0
##
## Rsquared
##                     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## random_forest 0.1690466 0.3076896 0.3660373 0.3941938 0.4646946 0.7050781    0
## boosting      0.1124967 0.2878436 0.3956597 0.3837432 0.4978955 0.5951724    0
```

The best model is random forest since this model has the lowest mean value of Cross-validation RMSE 20.8315 on train data comparing to all other models. According to the cv results on train data, the second best model is MARS with mean value of RMSE 21.5259. We should always choose the model using CV results on train data rather than prediction error.

Prediction: Evaluating performance on test data

```
# Linear regression model, KNN, ridge, lasso, lasso 1 se, elastic net,
# pls, gam, mars, bagging, random forest, boosting
# lm_rmse <- test_rmse1
# lm_rmse
#
# knn_rmse <- test_rmse2
# knn_rmse
#
# ridge_rmse <- test_rmse3
# ridge_rmse
#
# lasso_rmse <- test_rmse4
# lasso_rmse
#
# lasso_1se_rmse <- test_rmse5
# lasso_1se_rmse
#
# enet_rmse <- test_rmse6
# enet_rmse
#
# pls_rmse <- test_rmse7
# pls_rmse
#
# gam_rmse <- test_rmse8
# gam_rmse

mars_rmse <- test_rmse9
mars_rmse
```

```
## [1] 23.15086
```

```
# bag_rmse <- test_rmse10
# bag_rmse
```

```
rf_rmse <- test_rmse_rf
rf_rmse
```

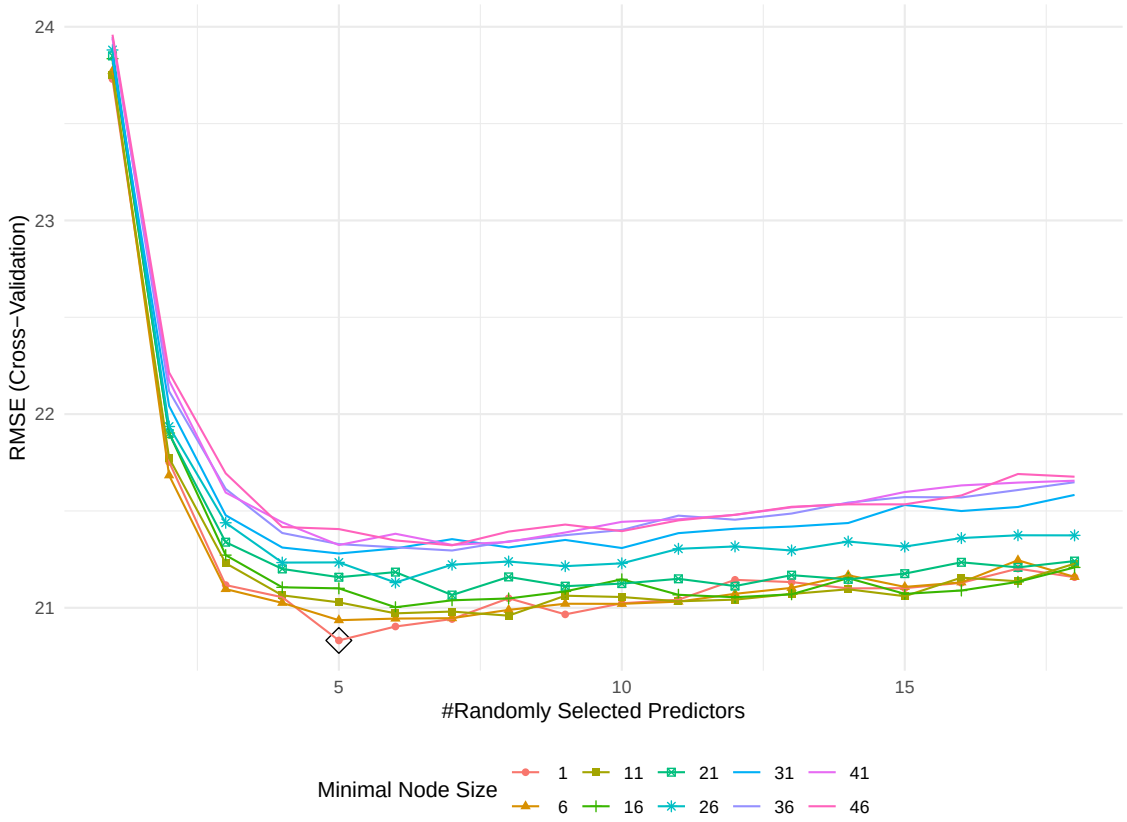
```
## [1] 22.39369
```

```
# gbm_rmse <- test_rmse13
# gbm_rmse
```

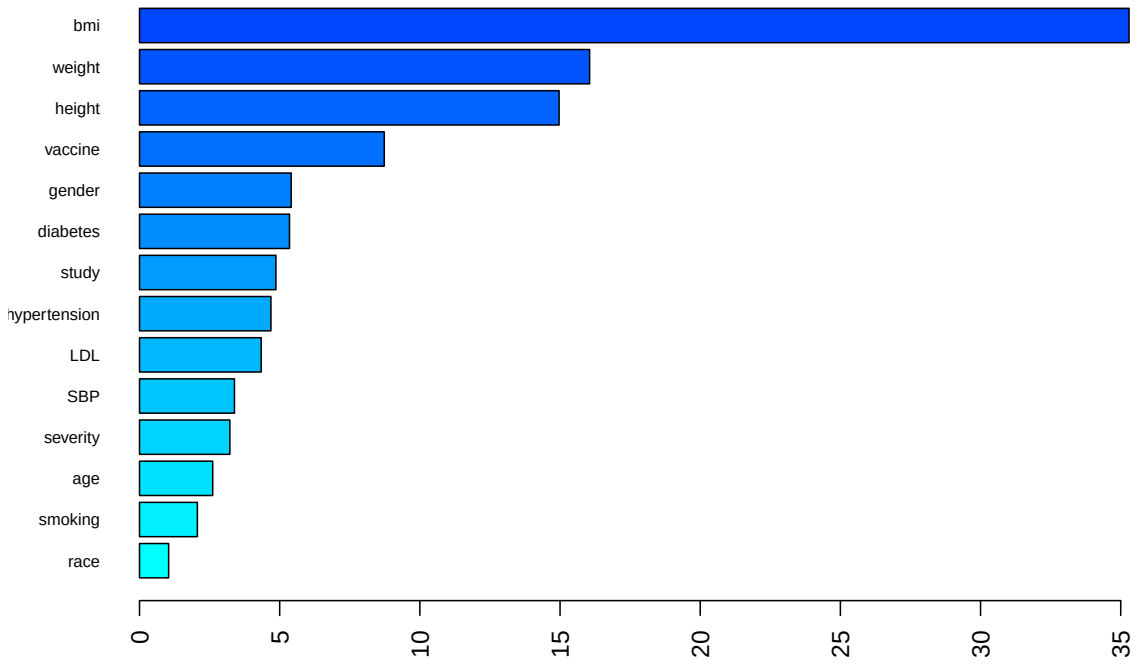
Interpretation on best and second best model obtained from cv results on train data

Random forest Variable importance

```
# number of node
ggplot(rf_model, highlight = TRUE)
```




```
# variable importance
set.seed(3554)
dat3 <- dat
rf2.final.per <- ranger(recovery_time~.,
                        dat3[train_rows,],
                        mtry = rf_model$bestTune[[1]],
                        splitrule = "variance",
                        min.node.size = rf_model$bestTune[[3]],
                        importance = "permutation",
                        scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(rf2.final.per), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan","blue"))(19))
```




```
# set.seed(3554)
# rf2.final.imp <- ranger(recovery_time~.,
#                         dat3[train_rows,],
#                         mtry = rf_model$bestTune[[1]],
#                         splitrule = "variance",
#                         min.node.size = rf_model$bestTune[[3]],
#                         importance = "impurity")
#
# barplot(sort(ranger::importance(rf2.final.imp), decreasing = FALSE),
#         las = 2, horiz = TRUE, cex.names = 0.7,
#         col = colorRampPalette(colors = c("cyan","blue"))(19))
```

Mars

```
summary(mars_model)
```

```
## Call: earth(x=matrix[1403,19], y=c(14,36,50,65,2...), keepxy=TRUE, degree=2,
##             nprune=2)
##
##                      coefficients
## (Intercept)             39.49104
## studyB * h(bmi-31.3)    32.34752
##
## Selected 2 of 30 terms, and 2 of 19 predictors (nprune=2)
## Termination condition: Reached nk 39
## Importance: studyB, bmi, gender0-unused, gender1-unused, race2-unused, ...
## Number of terms at each degree of interaction: 1 0 1
## GCV 455.5987    RSS 636019.8    GRSq 0.3927496    RSq 0.3949133
```

Conclusions

In this section, summarize your findings from the model analysis and discuss the insights gained into predicting time to recovery from COVID-19.