final_recovery_time

2023-05-10

Data Cleaning

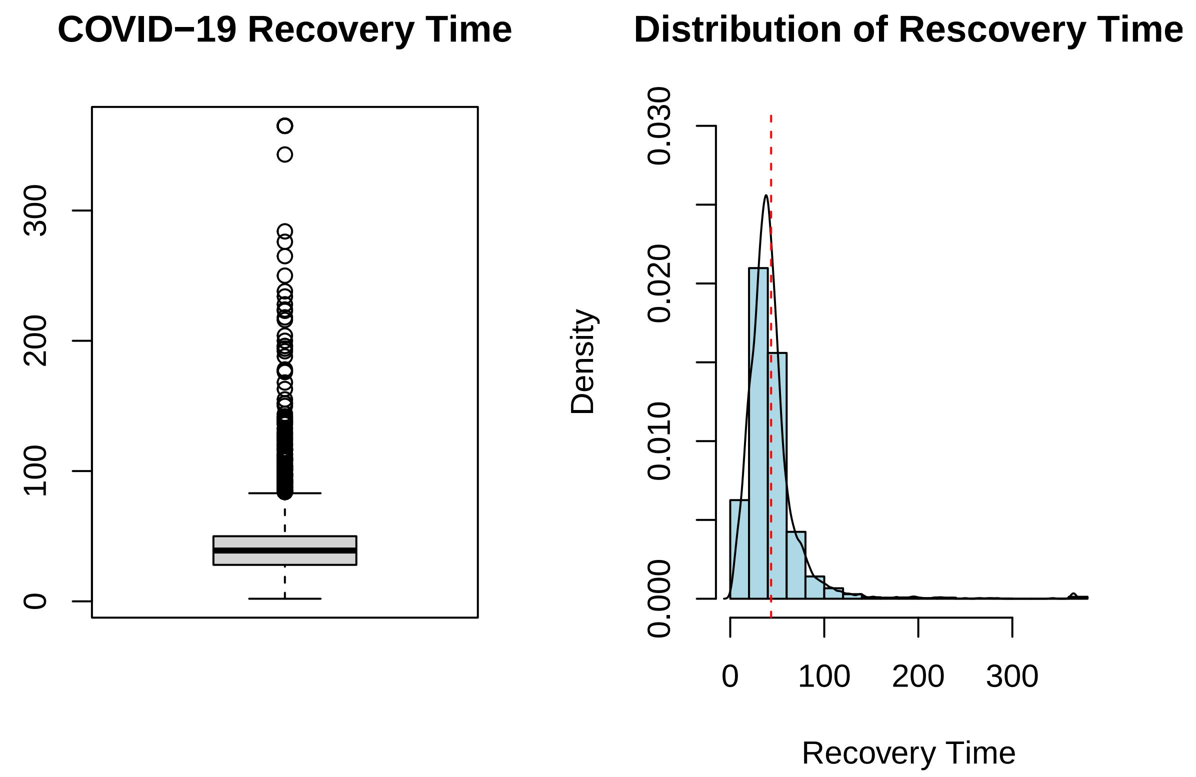
```
dat0 = read_csv(file = "final_used_data.csv")
```

```
## Rows: 3604 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (13): id, age, gender, race, smoking, bmi, hypertension, diabetes, sbp, ...
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

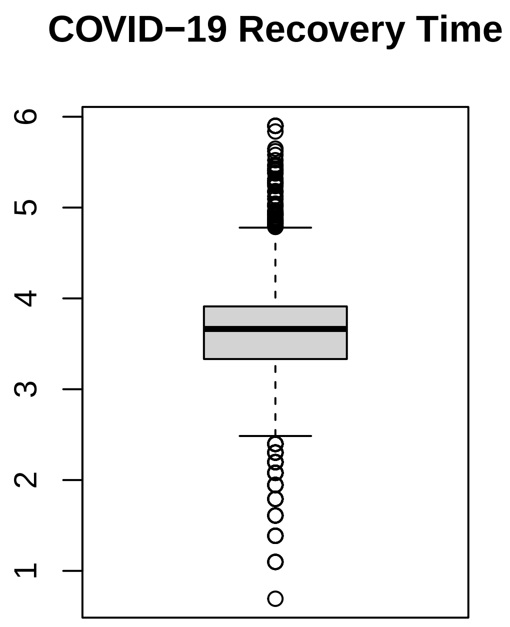
```
dat1 <- na.omit(dat0)

#Create a new data frame without id variable
dat <- dat1[ , !names(dat0) %in% c("id")]
attach(dat)
```

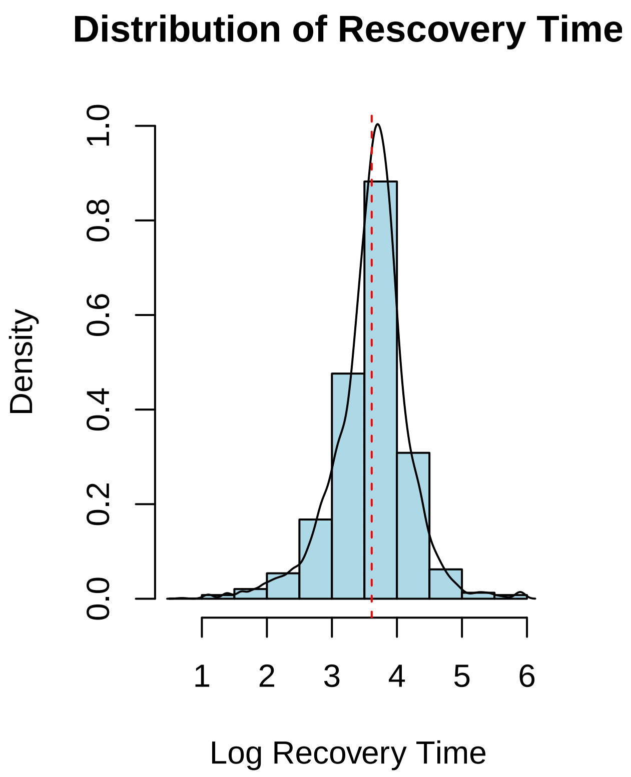
Data Preprocessing

```
#Visualize response variable
par(mfrow = c(1, 2))
boxplot(dat$recovery_time, main = "COVID-19 Recovery Time")
hist(dat$recovery_time, main = "Distribution of Rescovery Time", col = "lightblue",
                        xlab = "Recovery Time", prob = TRUE, ylim = c(0,0.03))
lines(density(dat$recovery_time))
abline(v = mean(dat$recovery_time), lty = "dashed", col = "red")
```



```
# The above plots show that the response variable is right-skewed, so log-transformation was performed.


dat$log_recovery_time <- log(dat$recovery_time)
par(mfrow = c(1, 2))
boxplot(dat$log_recovery_time, main = "COVID-19 Recovery Time")
hist(dat$log_recovery_time, main = "Distribution of Rescovery Time", col = "lightblue",
xlab = "Log Recovery Time", prob = TRUE, ylim = c(0,1))
lines(density(dat$log_recovery_time))
abline(v = mean(dat$log_recovery_time), lty = "dashed", col = "red")
```





```
#After the transformation, the outcome variable is normally distributed.
```

Data Partition

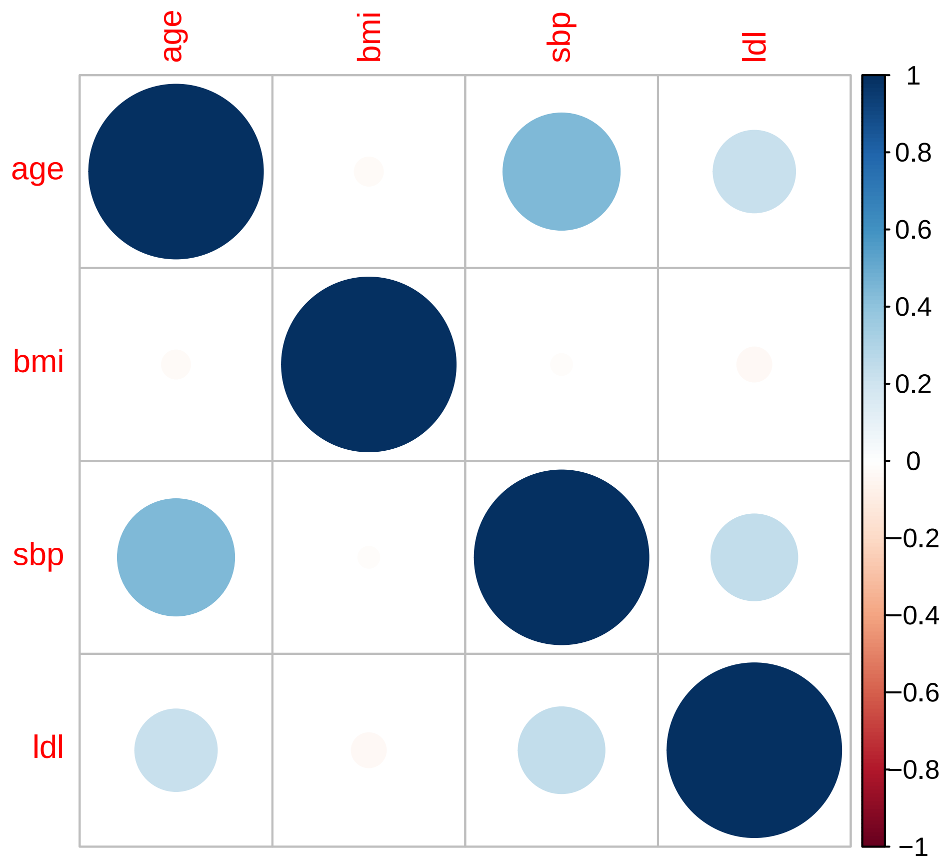
```
set.seed(5220)
trainRows <- createDataPartition(y = dat$log_recovery_time, p = 0.8, list = FALSE)

# Training data
dat_train = dat[trainRows, ]
x_train = model.matrix(log_recovery_time~.,dat)[trainRows, -1]
y_train = dat$log_recovery_time[trainRows]
# Test data
dat_test = dat[-trainRows, ]
x_test = model.matrix(log_recovery_time~.,dat)[-trainRows, -1]
y_test = dat$log_recovery_time[-trainRows]
```

Exploratory Analysis & Data Visualization

```
# Summary statistics for each variable
summary(dat_train)
```

```
##       age            gender            race          smoking
##  Min.   :46.00   Min.   :0.0000   Min.   :1.000   Min.   :0.0000
##  1st Qu.:57.00   1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:0.0000
##  Median :60.00   Median :0.0000   Median :1.000   Median :0.0000
##  Mean   :60.11   Mean   :0.4884   Mean   :1.753   Mean   :0.4919
##  3rd Qu.:63.00   3rd Qu.:1.0000   3rd Qu.:3.000   3rd Qu.:1.0000
##  Max.   :79.00   Max.   :1.0000   Max.   :4.000   Max.   :2.0000
##       bmi        hypertension       diabetes           sbp
##  Min.   :19.7   Min.   :0.0000   Min.   :0.0000   Min.   :103.0
##  1st Qu.:25.8   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:125.0
##  Median :27.6   Median :0.0000   Median :0.0000   Median :130.0
##  Mean   :27.7   Mean   :0.4853   Mean   :0.1622   Mean   :130.3
##  3rd Qu.:29.5   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:136.0
##  Max.   :39.8   Max.   :1.0000   Max.   :1.0000   Max.   :157.0
##       ldl           vaccine          severity       recovery_time
##  Min.   : 45.0   Min.   :0.0000   Min.   :0.00000   Min.   :  3.00
##  1st Qu.: 97.0   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.: 28.00
##  Median :110.0   Median :1.0000   Median :0.00000   Median : 39.00
##  Mean   :110.1   Mean   :0.5934   Mean   :0.09185   Mean   : 43.38
##  3rd Qu.:124.0   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.: 50.00
##  Max.   :174.0   Max.   :1.0000   Max.   :1.00000   Max.   :365.00
##  log_recovery_time
##  Min.   :1.099
##  1st Qu.:3.332
##  Median :3.664
##  Mean   :3.612
##  3rd Qu.:3.912
##  Max.   :5.900
```

```
#Relocate columns putting non-discrete predictors together
dat1 =
  dat %>%
  select(age,bmi,sbp,ldl,log_recovery_time)


# Correlation Plot
dat2 <- model.matrix(log_recovery_time ~ ., dat1)[ ,-1]
x <- dat2[trainRows,]
corrplot(cor(x))
```



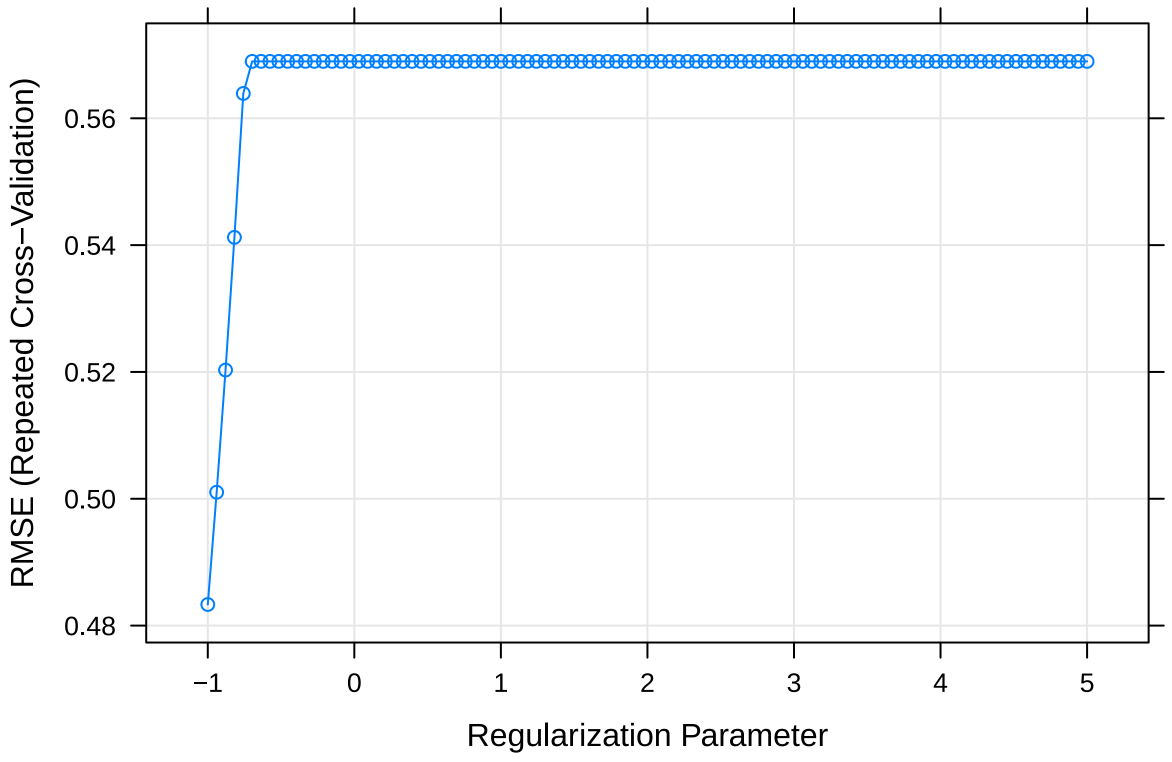
```
# Convert non-numeric columns to numeric
dat_train1 <- dat_train
non_numeric_cols <- sapply(dat_train1, function(x) !is.numeric(x))
dat_train1[, non_numeric_cols] <- lapply(dat_train1[, non_numeric_cols], as.numeric)
```

Fit LASSO

```
ctrl1 <- trainControl(method = "repeatedcv", number = 10, repeats = 5)

set.seed(5220)
lasso.fit <- train(x_train, y_train,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(5, -1, 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)
```



```r
lasso.fit$bestTune
```

```
##   alpha    lambda
## 1     1 0.3678794
```

```r
coef(lasso.fit$finalModel, lasso.fit$bestTune$lambda)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)   3.454454838
## age           .
## gender        .
## race          .
## smoking       .
## bmi           .
## hypertension  .
## diabetes      .
## sbp           .
## ldl           .
## vaccine       .
## severity      .
## recovery_time 0.003622746
```

Fit Elastic net

```
set.seed(5220)
enet.fit <- train(x_train, y_train,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                  lambda = exp(seq(2, -2, length = 50))),
                  trControl = ctrl1)
```

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

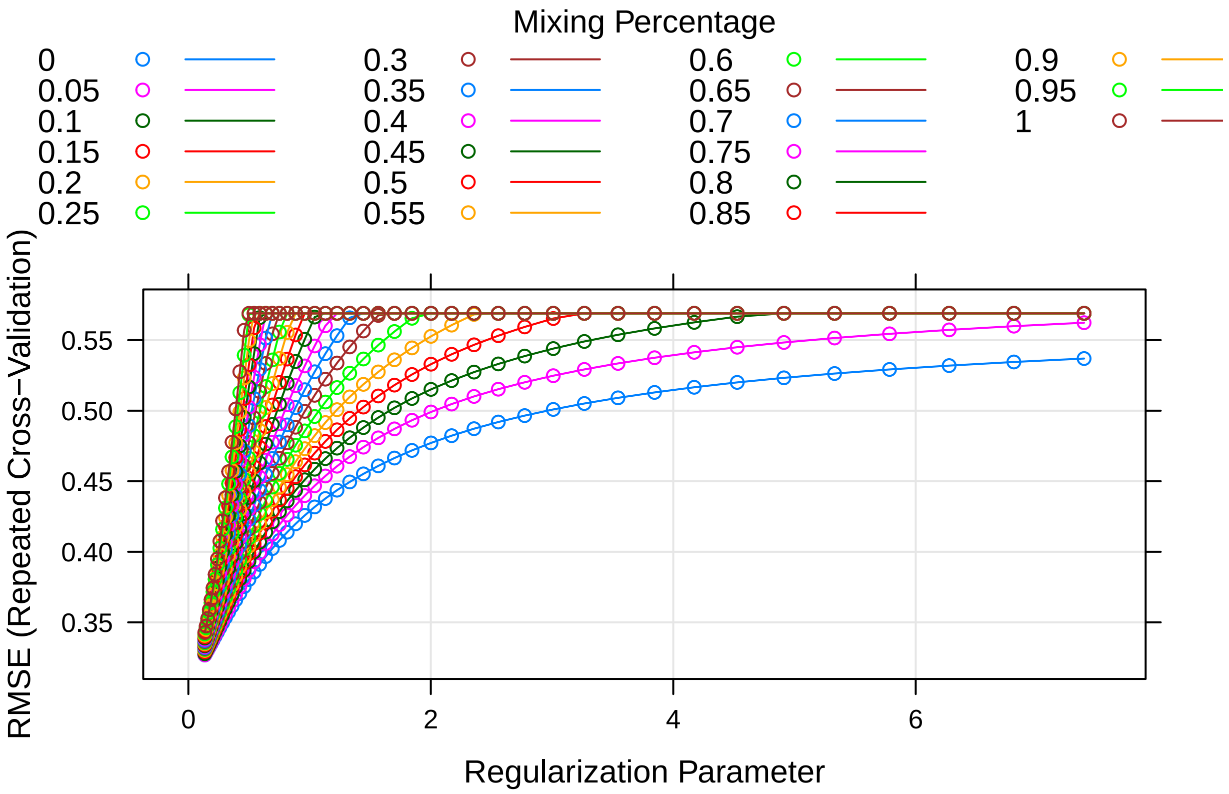
```
enet.fit$bestTune
```

```
##   alpha    lambda
## 1     0 0.1353353
```

```
coef(enet.fit$finalModel, enet.fit$bestTune$lambda)
```

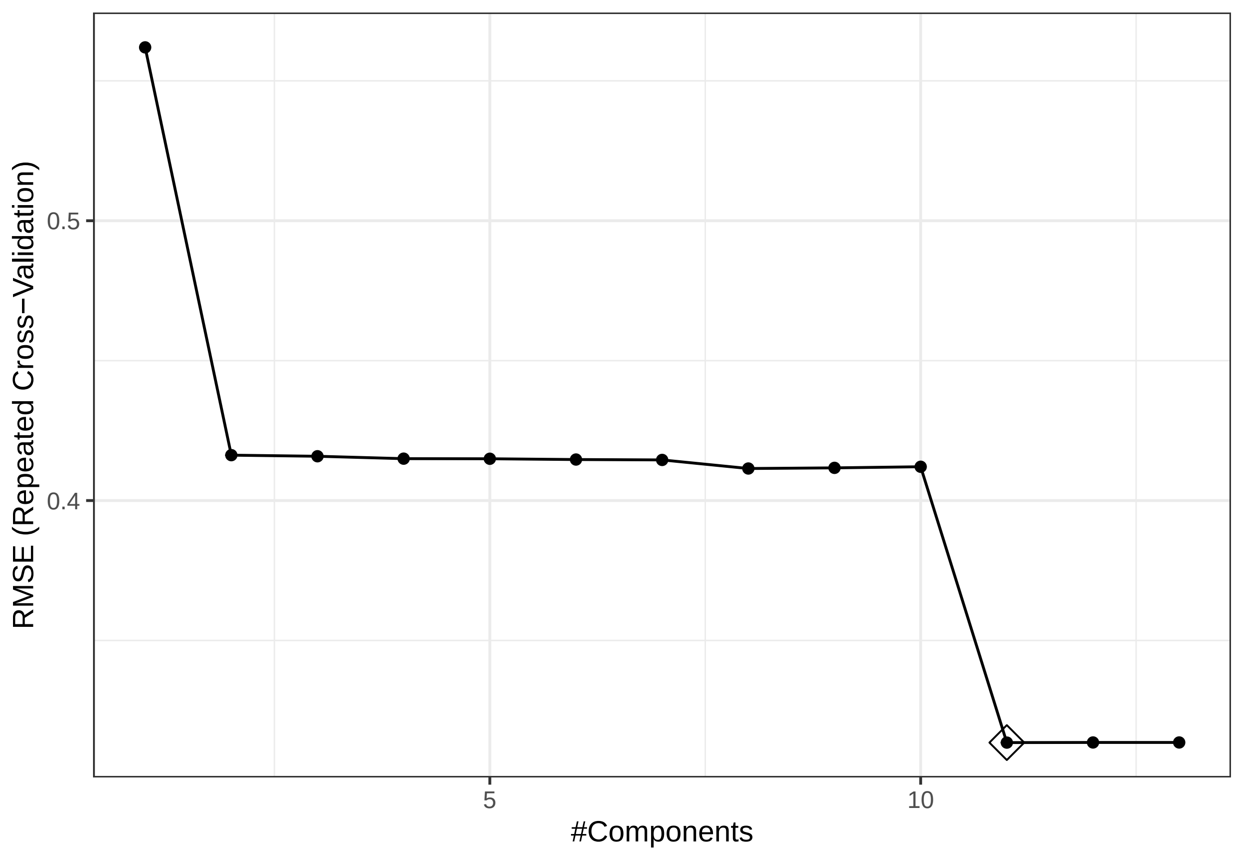
```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)    2.6435219747
## age            0.0018941343
## gender        -0.0402957253
## race          -0.0054230774
## smoking        0.0273289815
## bmi            0.0031212386
## hypertension   0.0120551948
## diabetes       0.0095582923
## sbp            0.0018610716
## ldl            0.0001388527
## vaccine       -0.0583915581
## severity       0.0563081990
## recovery_time  0.0126284871
```

```
plot(enet.fit)
```



Fit PCR

```r
ctrl2 <- trainControl(method = "repeatedcv",
                      number = 10,
                      repeats = 5,
                      selectionFunction = "best")

set.seed(5220)
pcr.fit <- train(x_train, y_train,
                 method = "pcr",
                 tuneGrid = data.frame(ncomp = 1:13),
                 trControl = ctrl2,
                 preProcess = c("center", "scale"))

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

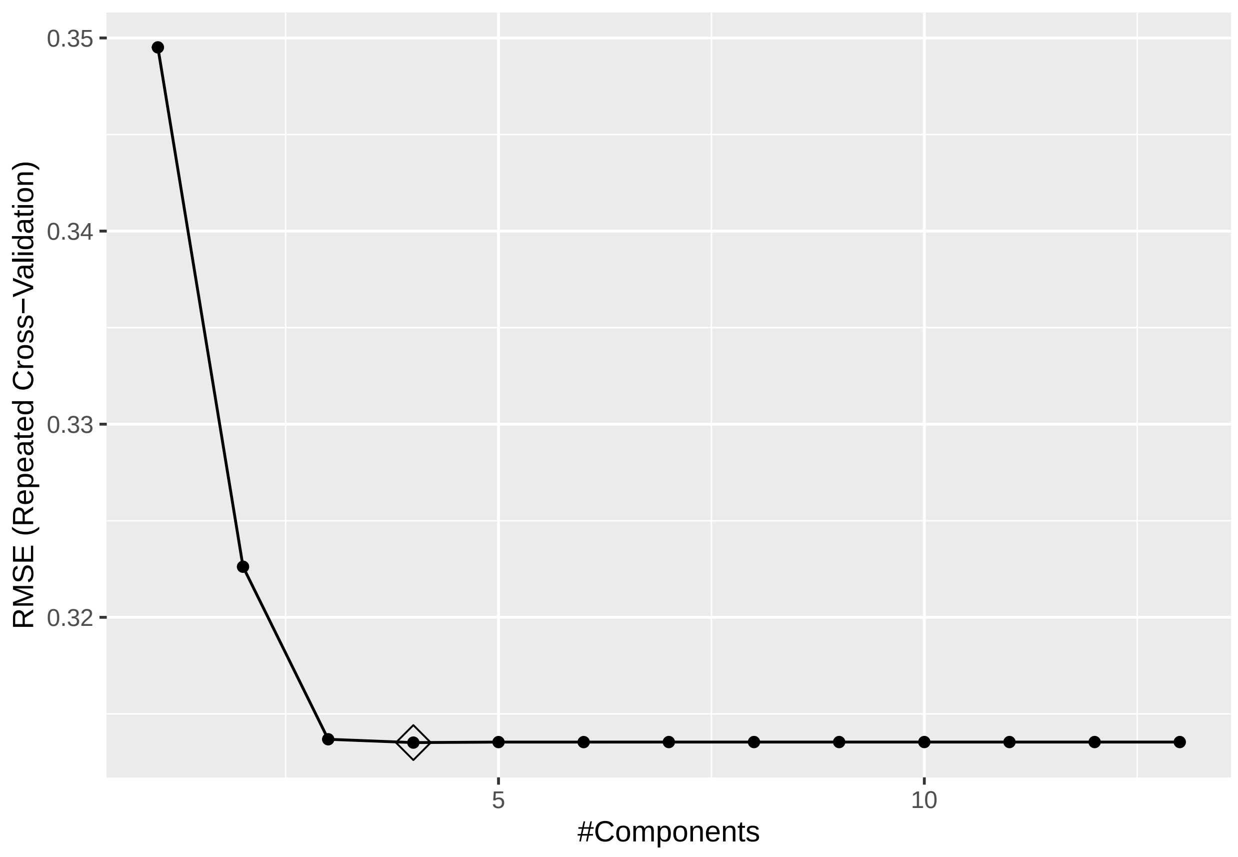


```
summary(pcr.fit)
```

```
## Data:    X dimension: 2885 12
##  Y dimension: 2885 1
## Fit method: svdpc
## Number of components considered: 11
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X          18.536    30.03    38.94    47.60    55.99    64.29    72.29
## .outcome    2.576    46.70    46.82    47.14    47.17    47.32    47.34
##           8 comps  9 comps  10 comps  11 comps
## X           80.14    87.41     93.17     98.43
## .outcome    48.22    48.23     48.25     69.98
```

Fit PLS

```
set.seed(5220)
pls.fit <- train(x_train, y_train,
                 method = "pls",
                 tuneGrid = data.frame(ncomp = 1:13),
                 trControl = ctrl2,
                 preProcess = c("center", "scale"))

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



```
summary(pls.fit)
```

```
## Data:    X dimension: 2885 12
##  Y dimension: 2885 1
## Fit method: oscorespls
## Number of components considered: 4
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps
## X           11.78    25.05    35.33    43.41
## .outcome    62.66    68.25    69.97    69.99
```

Fit GAM

```
#ctrl3 <- trainControl(method = "cv", number = 10)

set.seed(5220)
gam.fit <- train(x_train, y_train,
                 method = "gam",
                 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 1.8-40. For overview type 'help("mgcv-package")'.
```

```
gam.fit$bestTune
```

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

```
gam.fit$finalModel
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ gender + hypertension + diabetes + vaccine + severity +
##     smoking + race + s(age) + s(sbp) + s(ldl) + s(recovery_time) +
##     s(bmi)
##
## Estimated degrees of freedom:
## 0.0004 0.0001 6.4058 8.9836 0.4052  total = 23.8
##
## GCV score: 0.003162374
```

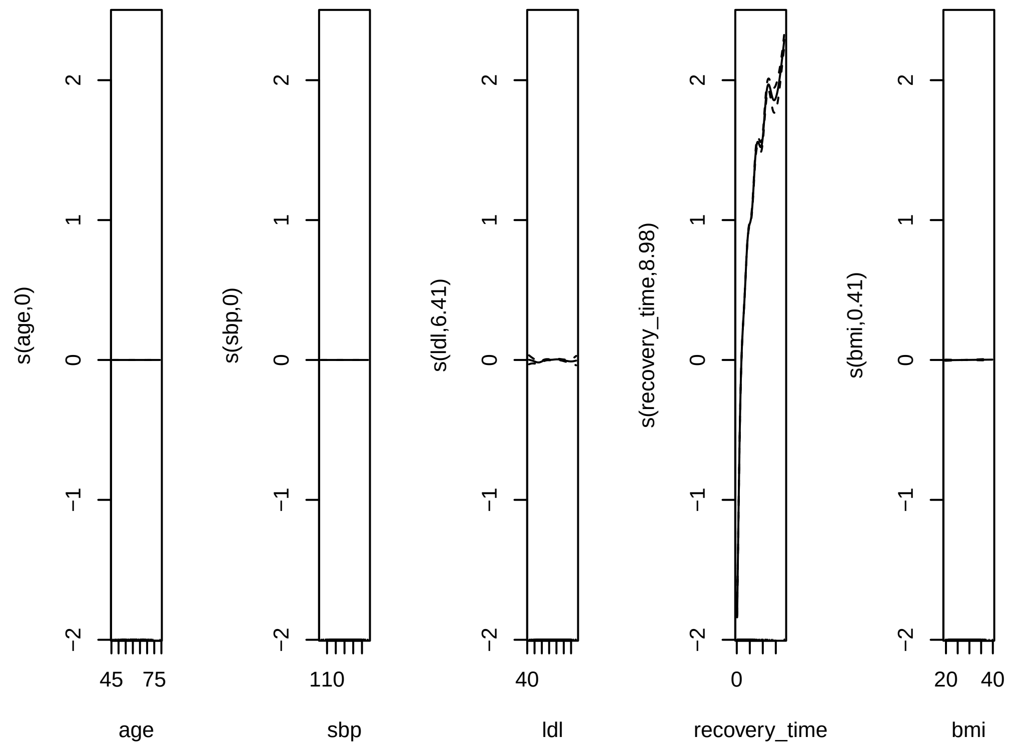
```
coef(gam.fit$finalModel)
```

```
##        (Intercept)             gender       hypertension           diabetes
##       3.612330e+00       9.722452e-04       4.146820e-04       1.305287e-03
##            vaccine           severity            smoking               race
##      -1.523296e-03      -4.759119e-03       2.982511e-04      -2.407263e-04
##           s(age).1           s(age).2           s(age).3           s(age).4
##      -2.718648e-09      -2.242637e-08       5.459337e-08      -8.260225e-08
##           s(age).5           s(age).6           s(age).7           s(age).8
##       1.966523e-08       6.613925e-08       1.678210e-08       4.973789e-07
##           s(age).9           s(sbp).1           s(sbp).2           s(sbp).3
##       2.112459e-10      -2.640486e-08      -4.580629e-09       1.930582e-08
##           s(sbp).4           s(sbp).5           s(sbp).6           s(sbp).7
##      -3.429016e-08      -1.392191e-08       2.878634e-08       1.522718e-08
##           s(sbp).8           s(sbp).9           s(ldl).1           s(ldl).2
##       1.531279e-07      -1.093297e-10       1.401524e-03       2.314941e-02
##           s(ldl).3           s(ldl).4           s(ldl).5           s(ldl).6
##      -5.147928e-03       1.687779e-02       8.290092e-03      -8.597029e-03
##           s(ldl).7           s(ldl).8           s(ldl).9 s(recovery_time).1
##      -1.070984e-02      -5.783819e-02      -1.133501e-11      -1.101218e+00
## s(recovery_time).2 s(recovery_time).3 s(recovery_time).4 s(recovery_time).5
##       2.482169e+00       1.071460e+00       1.411544e+00      -9.561915e-01
## s(recovery_time).6 s(recovery_time).7 s(recovery_time).8 s(recovery_time).9
##      -1.022372e+00       1.227444e+00       1.282190e+00       1.173501e+00
##           s(bmi).1           s(bmi).2           s(bmi).3           s(bmi).4
##      -1.855887e-07       2.174533e-08       3.572935e-08      -4.140910e-08
##           s(bmi).5           s(bmi).6           s(bmi).7           s(bmi).8
##      -1.157483e-08      -3.957406e-08       7.307461e-09       2.533496e-07
##           s(bmi).9
##       5.648109e-04
```

```
mod_gam <- gam(log_recovery_time ~ gender + race + smoking + hypertension +
    diabetes + vaccine + severity + s(age) +
    s(sbp) + s(ldl) + s(bmi),
               data = dat[trainRows,], method = "REML")

summary(mod_gam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log_recovery_time ~ gender + race + smoking + hypertension +
##     diabetes + vaccine + severity + s(age) + s(sbp) + s(ldl) +
##     s(bmi)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.728313   0.029602 125.947  < 2e-16 ***
## gender       -0.103779   0.018851  -5.505 4.02e-08 ***
## race         -0.016905   0.008700  -1.943   0.0521 .
## smoking       0.080160   0.013982   5.733 1.09e-08 ***
## hypertension  0.052285   0.033266   1.572   0.1161
## diabetes      0.005141   0.025571   0.201   0.8407
## vaccine      -0.197984   0.019165 -10.330  < 2e-16 ***
## severity      0.168279   0.032605   5.161 2.62e-07 ***
## ---
## 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) 2.665  3.401  1.892   0.110
## s(sbp) 1.779  2.233  1.248   0.221
## s(ldl) 1.003  1.005  0.004   0.967
## s(bmi) 5.677  6.794 80.886  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.216   Deviance explained = 22.1%
## -REML =   2161  Scale est. = 0.25444   n = 2885
```

```
par(mfrow = c(1,6))
plot(gam.fit$finalModel)
```

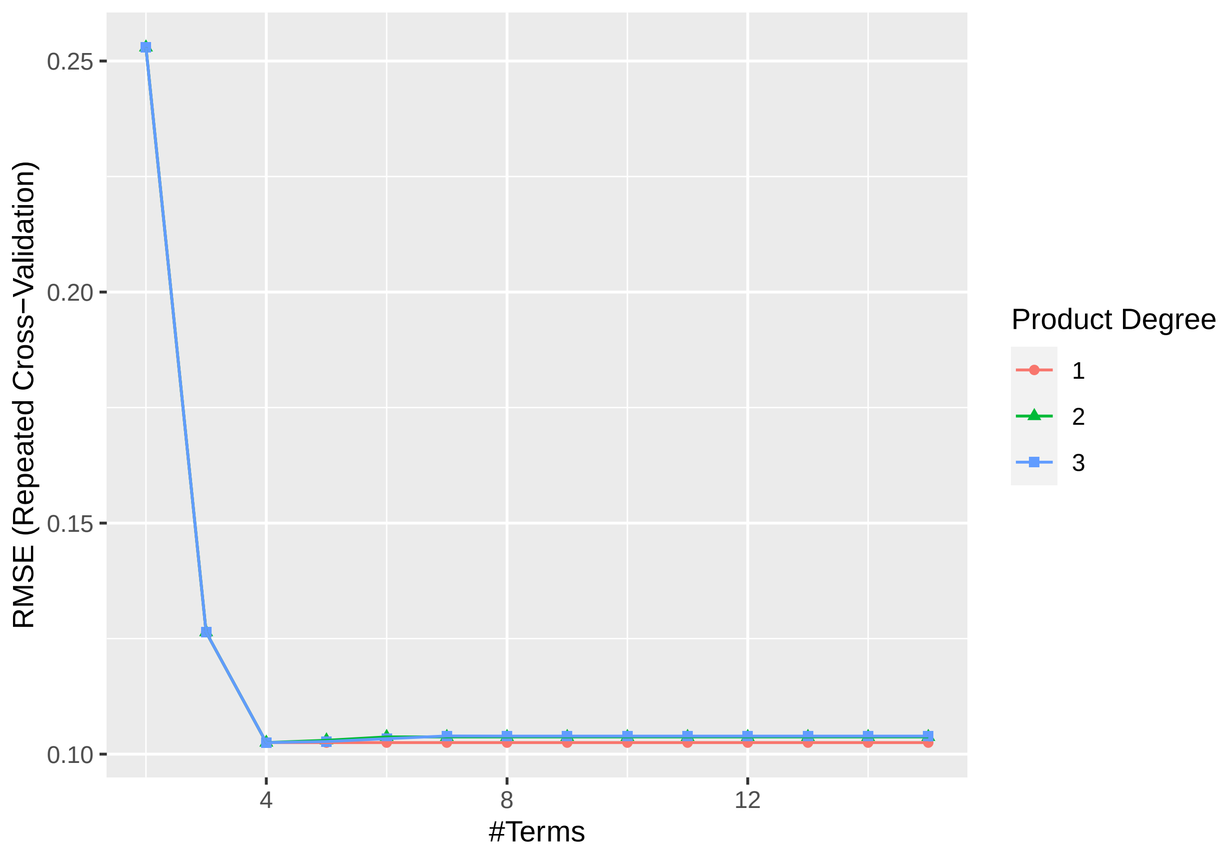


Fit MARS

```r
mars_grid <- expand.grid(degree = 1:3,
                         nprune = 2:15)
set.seed(5220)
mars.fit <- train(x_train, y_train,
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl1)
```

```
## Loading required package: earth

## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos
```

```r
ggplot(mars.fit)
```



```r
mars.fit$bestTune
```

```
##   nprune degree
## 3      4      1
```

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

```
##          (Intercept)  h(recovery_time-41)  h(41-recovery_time)
##          3.806773377          0.014172314         -0.043827302
## h(recovery_time-110)
##         -0.009387009
```

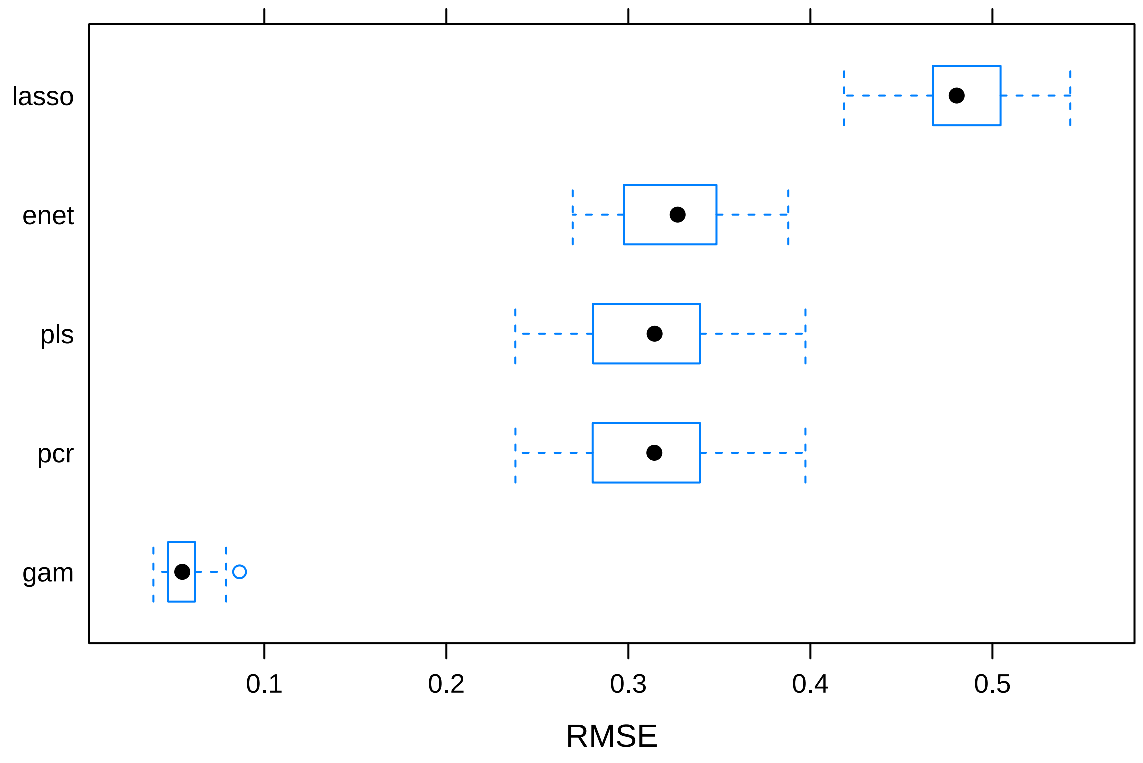
Model Comparison

```r
set.seed(5220)
resamp <- resamples(list(enet = enet.fit, lasso = lasso.fit, pcr = pcr.fit, pls = pls.fit, gam=gam.fit)
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: enet, lasso, pcr, pls, gam
## Number of resamples: 50
```

```
##
## MAE
##             Min.    1st Qu.     Median       Mean   3rd Qu.      Max. NA's
## enet  0.20448398 0.21367310 0.22406282 0.22380451 0.2316555 0.2466645    0
## lasso 0.31613346 0.34329906 0.35120802 0.35226982 0.3635228 0.3773298    0
## pcr   0.17894427 0.18824442 0.19810593 0.19799056 0.2062234 0.2174753    0
## pls   0.17894328 0.18816450 0.19819321 0.19800937 0.2062770 0.2176440    0
## gam   0.03090753 0.03313992 0.03433917 0.03432081 0.0353659 0.0400894    0
##
## RMSE
##             Min.    1st Qu.     Median       Mean    3rd Qu.       Max. NA's
## enet  0.26938541 0.29772082 0.32704779 0.32684611 0.34811402 0.38787385    0
## lasso 0.41847990 0.46740296 0.48034811 0.48332882 0.50435816 0.54279532    0
## pcr   0.23794846 0.28077381 0.31427071 0.31349502 0.33886646 0.39727401    0
## pls   0.23790094 0.28094331 0.31440206 0.31351276 0.33884139 0.39728622    0
## gam   0.03905939 0.04728467 0.05492136 0.05513242 0.06175933 0.08638066    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## enet  0.6144698 0.6730869 0.7022847 0.7168417 0.7619367 0.8361868    0
## lasso 0.6072387 0.6688274 0.7144653 0.7218806 0.7724960 0.8578139    0
## pcr   0.6161576 0.6779777 0.7119874 0.7224607 0.7702780 0.8427547    0
## pls   0.6161506 0.6781600 0.7118646 0.7224473 0.7700829 0.8426681    0
## gam   0.9811190 0.9891391 0.9905767 0.9907163 0.9931621 0.9954404    0
```

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

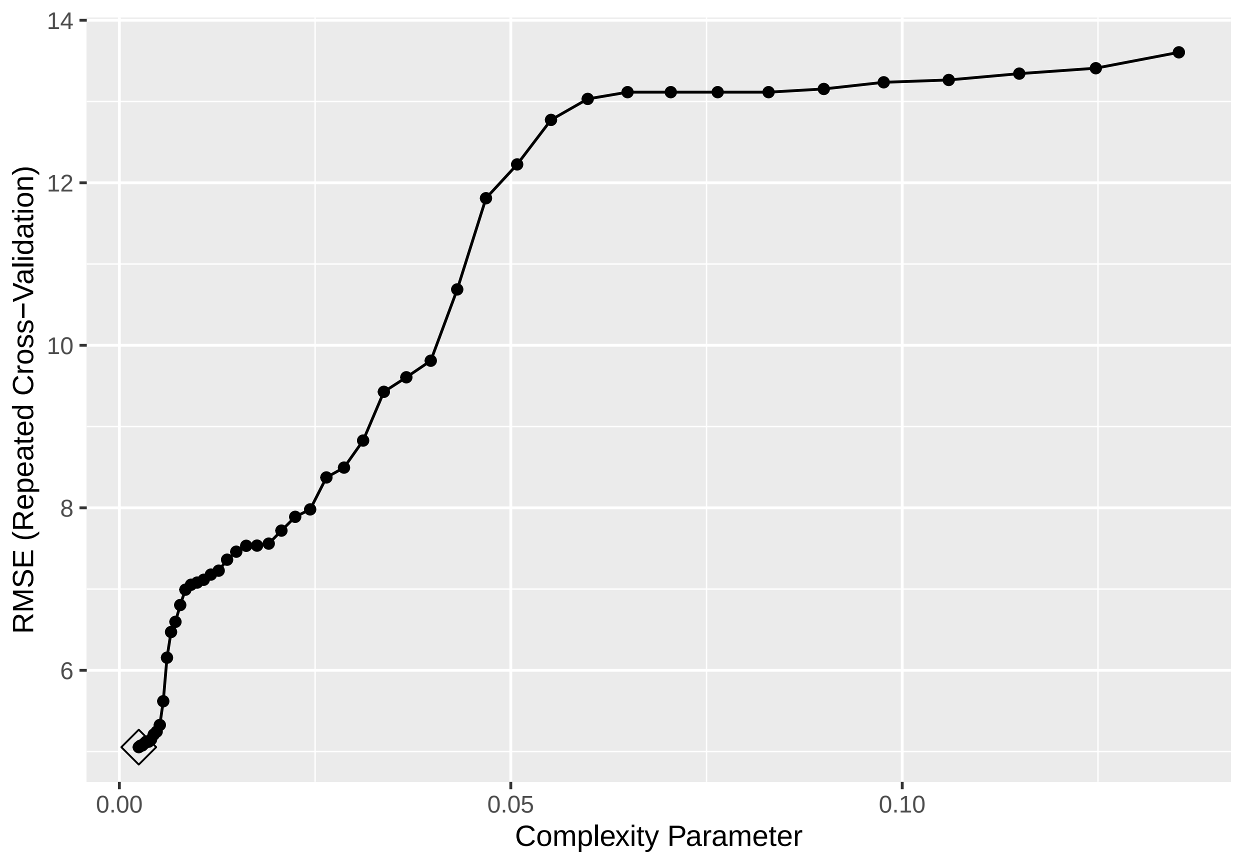


Regression tree

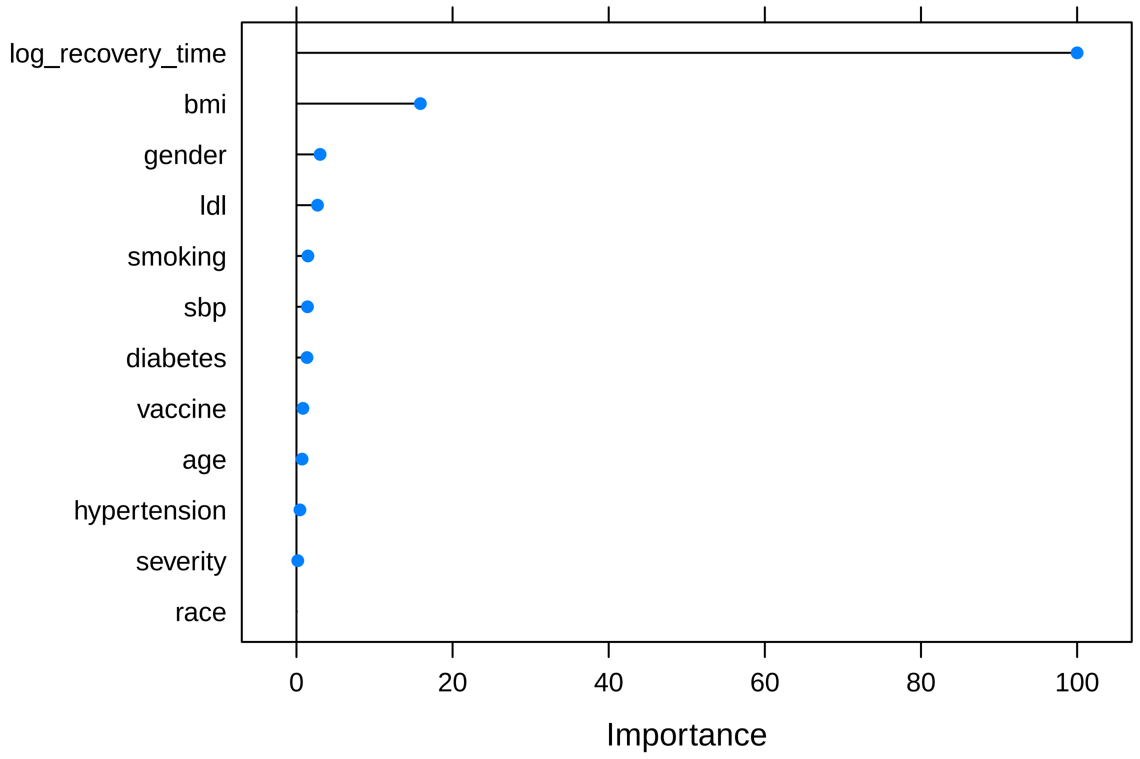
```r
set.seed(5220)

rpart.fit <- train(recovery_time ~ . ,
                   data = dat_train,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-2, length = 50))),
                   trControl = ctrl1)
rpart.fit$bestTune
```

```
##            cp
## 1 0.002478752
```

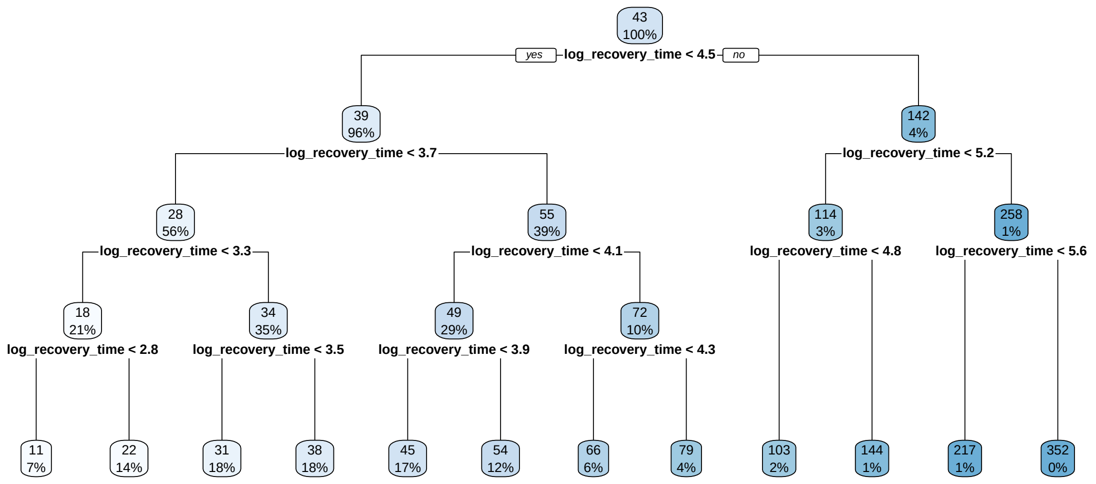
```r
# Plot of the complexity parameter
ggplot(rpart.fit, highlight = TRUE)
```



```r
# Variable importance
plot(varImp(rpart.fit, scale = TRUE))
```



```r
rpart.plot(rpart.fit$finalModel)
```

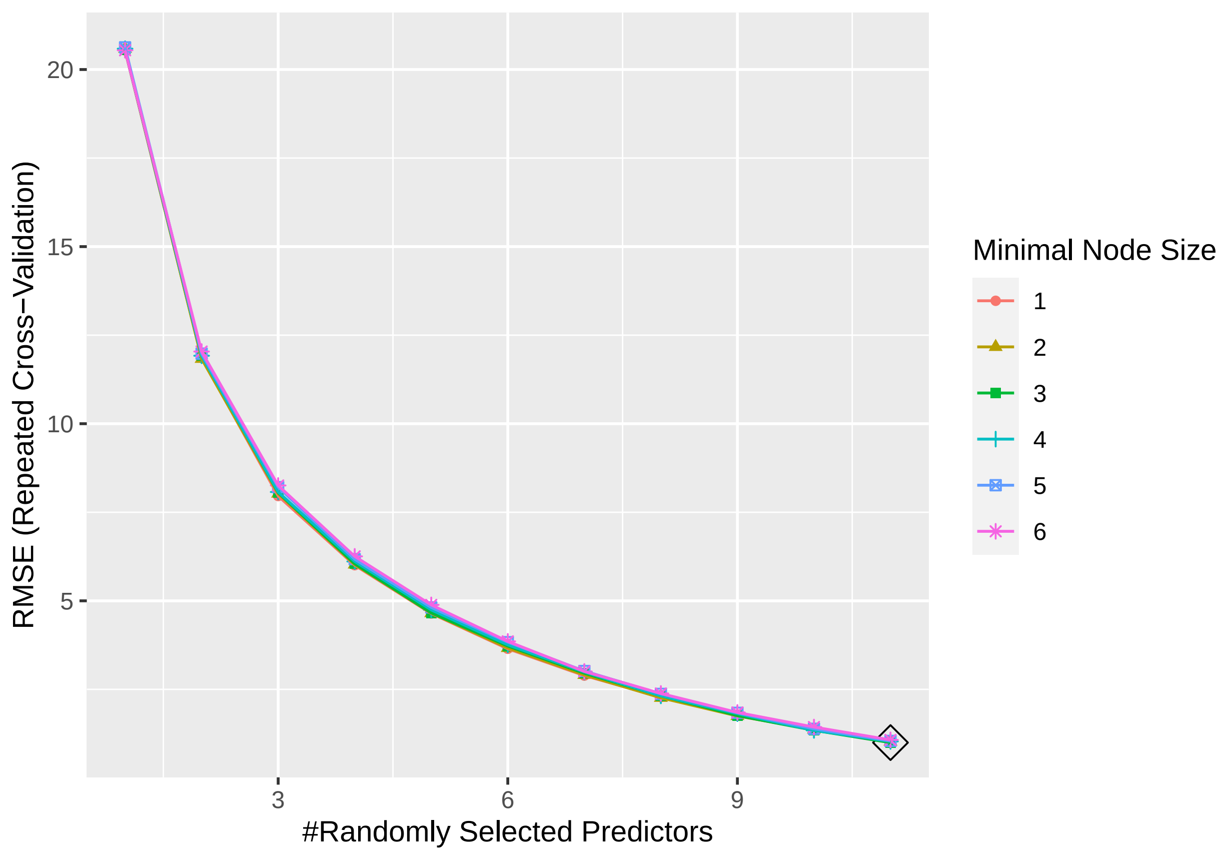


```
pred.rf <- predict(rpart.fit, newdata = dat_test)
RMSE(pred.rf, dat_test$recovery_time)
```

```
## [1] 4.298444
```

Random forest

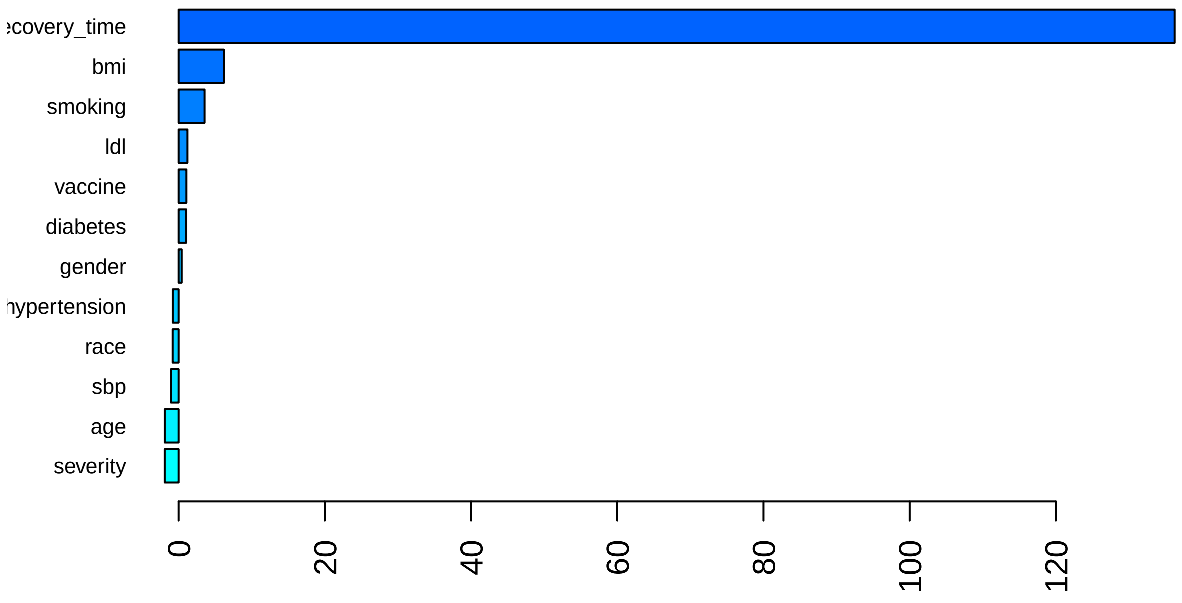
```
rf.grid <- expand.grid(mtry = 1:11,
                       splitrule = "variance",
                       min.node.size = 1:6)

set.seed(5220)

rf.fit <- train(recovery_time ~ .,
               data = dat_train,
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl1)

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



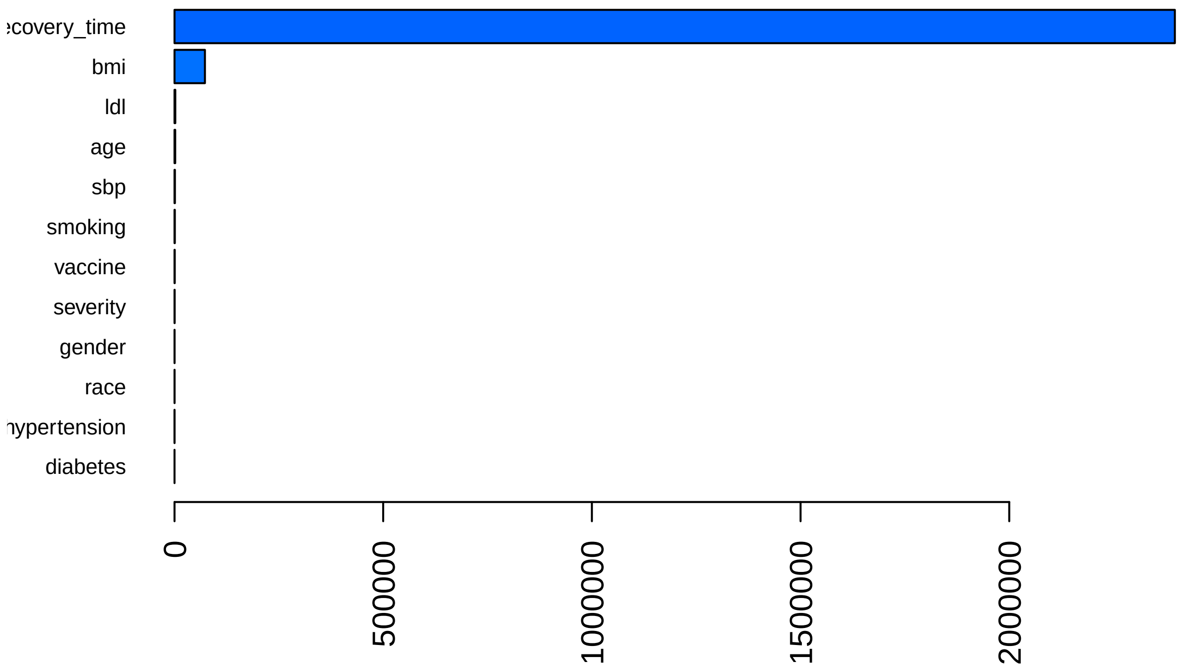
```
rf.fit$bestTune
```

```
##    mtry splitrule min.node.size
## 63   11  variance             3
```

```
# variable importance using permutation methods
set.seed(5220)
rf.perm = ranger(recovery_time ~ .,
                      data = dat_train,
                      mtry = rf.fit$bestTune[[1]],
                      splitrule = "variance",
                      min.node.size = rf.fit$bestTune[[3]],
                      importance = "permutation",
                      scale.permutation.importance = TRUE)

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



```
# variable importance using impurity methods
set.seed(5220)
rf.impu <- ranger(recovery_time ~ .,
                        data = dat_train,
                        mtry = rf.fit$bestTune[[1]],
                        splitrule = "variance",
                        min.node.size = rf.fit$bestTune[[3]],
                        importance = "impurity")

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

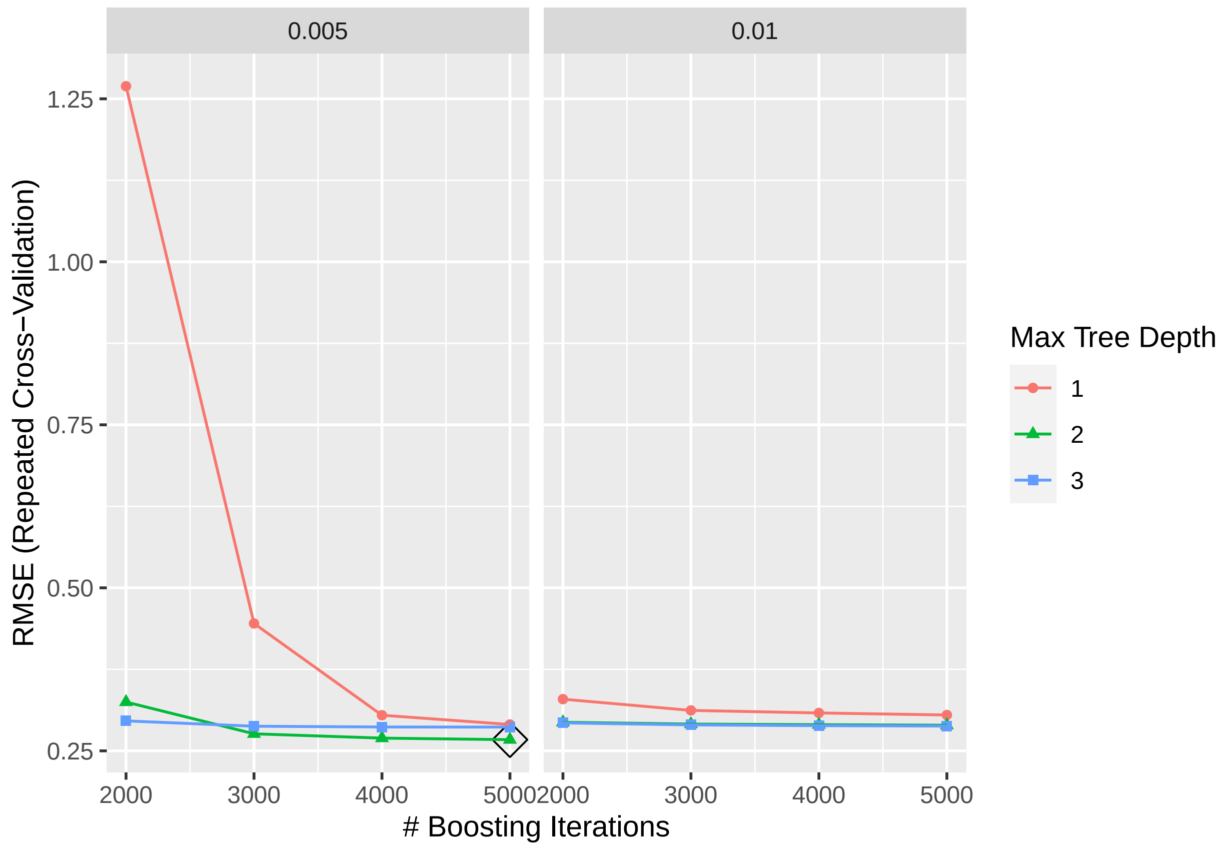


```
# test error
pred.rf <- predict(rf.fit, newdata = dat_test)
RMSE(pred.rf, dat_test$recovery_time)
```

```
## [1] 1.278566
```

Boosting

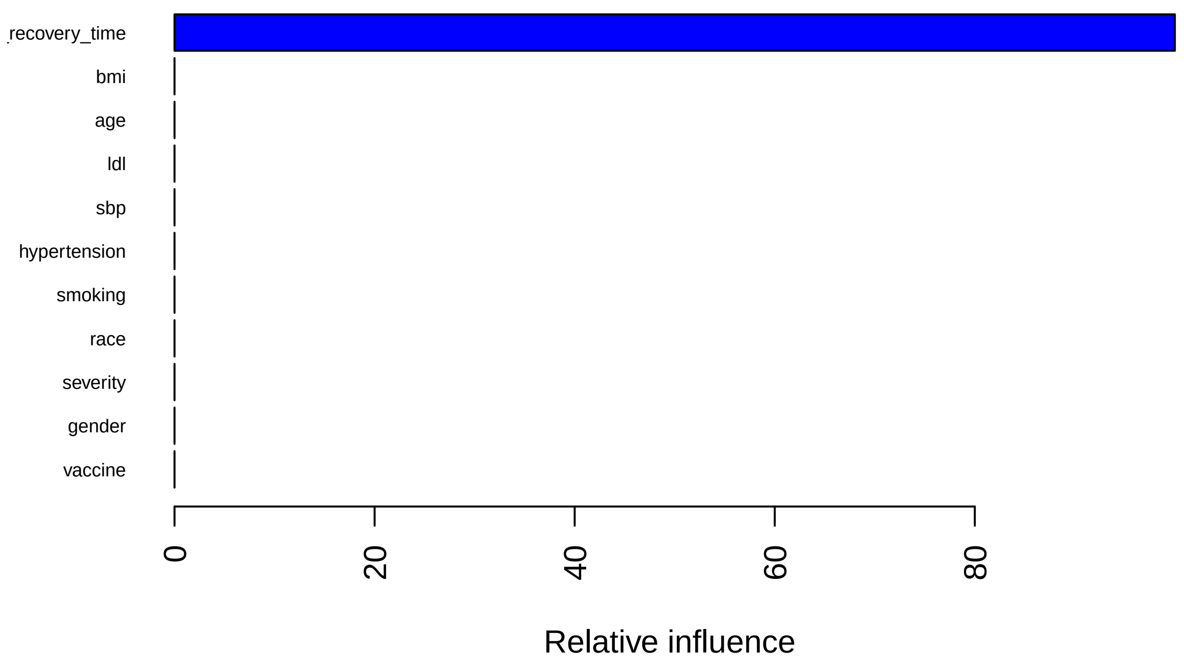
```
bst.grid = expand.grid(n.trees = c(2000,3000,4000,5000),
                        interaction.depth = 1:3,
                        shrinkage = c(0.005,0.01),
                        n.minobsinnode = c(1))

set.seed(5220)
bst.fit <- train(recovery_time ~.,
                 data = dat_train,
                 method = "gbm",
                 tuneGrid = bst.grid,
                 trControl = ctrl1,
                 verbose = FALSE)

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



```
bst.fit$bestTune
```

```
##   n.trees interaction.depth shrinkage n.minobsinnode
## 8    5000                 2     0.005              1
```

```
# variable importance
summary(bst.fit$finalModel, las = 2, cBars = 11, cex.names = 0.6)
```



```
##                                   var      rel.inf
## log_recovery_time log_recovery_time 9.999825e+01
## bmi                               bmi 5.949987e-04
## age                               age 5.065697e-04
## ldl                               ldl 3.628965e-04
## sbp                               sbp 1.217016e-04
## hypertension             hypertension 7.082340e-05
## smoking                       smoking 4.067221e-05
## race                             race 1.696229e-05
## severity                     severity 1.588669e-05
## gender                         gender 1.236924e-05
## vaccine                       vaccine 4.149641e-06
## diabetes                     diabetes 0.000000e+00
```

```
# test error
pred.bst <- predict(bst.fit, newdata = dat_test)
RMSE(pred.bst, dat_test$recovery_time)
```

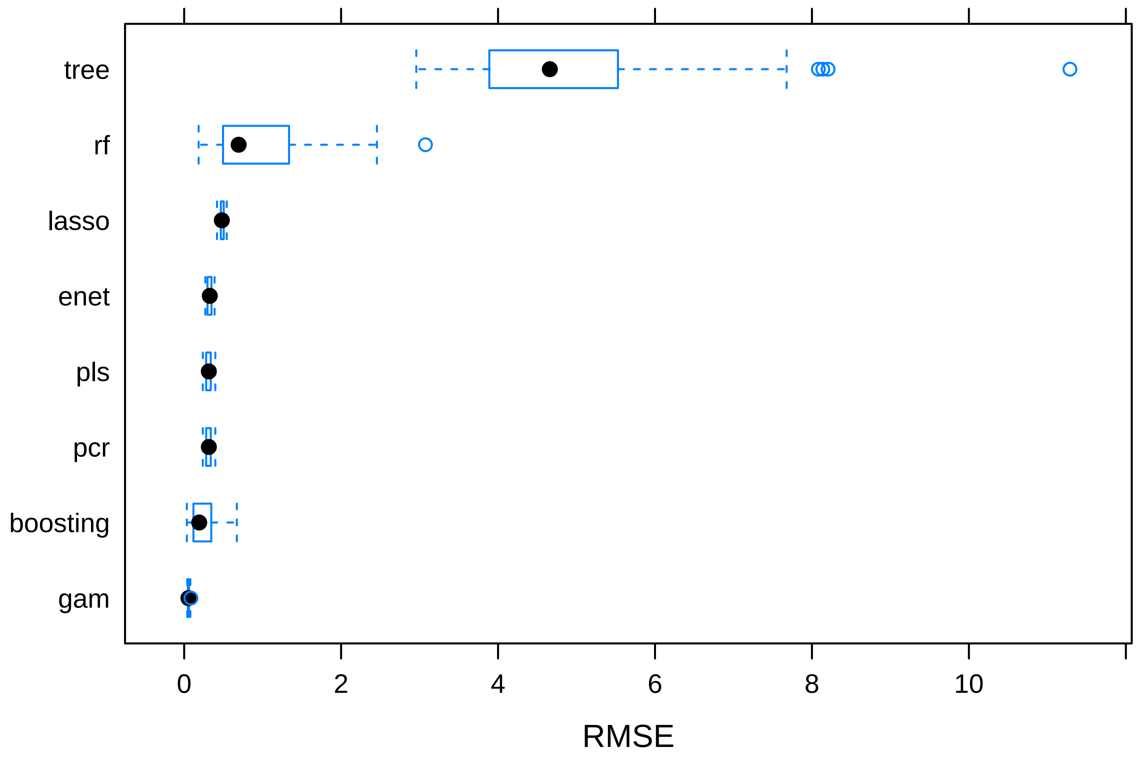
```
## [1] 0.8662751
```

Model Comparison

```
set.seed(5220)
resamp <- resamples(list(enet = enet.fit, lasso = lasso.fit, pcr = pcr.fit, pls = pls.fit, gam=gam.fit,
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: enet, lasso, pcr, pls, gam, tree, rf, boosting
## Number of resamples: 50
##
## MAE
##                 Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## enet     0.204483979 0.21367310 0.22406282 0.22380451 0.23165551 0.24666446
## lasso    0.316133461 0.34329906 0.35120802 0.35226982 0.36352276 0.37732984
## pcr      0.178944265 0.18824442 0.19810593 0.19799056 0.20622345 0.21747534
## pls      0.178943285 0.18816450 0.19819321 0.19800937 0.20627703 0.21764404
## gam      0.030907527 0.03313992 0.03433917 0.03432081 0.03536590 0.04008940
## tree     2.511276986 2.74260652 2.84829024 2.88825200 3.00423418 3.68344338
## rf       0.041952710 0.07184448 0.10474823 0.12563785 0.17484122 0.31405421
## boosting 0.009534808 0.02372112 0.03227388 0.03738471 0.04544817 0.08821171
##          NA's
## enet        0
## lasso       0
## pcr         0
## pls         0
## gam         0
## tree        0
## rf          0
## boosting    0
##
## RMSE
##                Min.    1st Qu.     Median       Mean    3rd Qu.        Max.
## enet     0.26938541 0.29772082 0.32704779 0.32684611 0.34811402  0.38787385
## lasso    0.41847990 0.46740296 0.48034811 0.48332882 0.50435816  0.54279532
## pcr      0.23794846 0.28077381 0.31427071 0.31349502 0.33886646  0.39727401
## pls      0.23790094 0.28094331 0.31440206 0.31351276 0.33884139  0.39728622
## gam      0.03905939 0.04728467 0.05492136 0.05513242 0.06175933  0.08638066
## tree     2.95780787 3.89787450 4.65972092 5.05439196 5.50287083 11.28748560
## rf       0.18453821 0.50448945 0.69436904 0.99714001 1.30883301  3.07490317
## boosting 0.03440117 0.11885121 0.19221470 0.26713252 0.34138146  0.67141072
##          NA's
## enet        0
## lasso       0
## pcr         0
## pls         0
## gam         0
## tree        0
## rf          0
## boosting    0
##
## Rsquared
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## enet     0.6144698 0.6730869 0.7022847 0.7168417 0.7619367 0.8361868    0
## lasso    0.6072387 0.6688274 0.7144653 0.7218806 0.7724960 0.8578139    0
## pcr      0.6161576 0.6779777 0.7119874 0.7224607 0.7702780 0.8427547    0
## pls      0.6161506 0.6781600 0.7118646 0.7224473 0.7700829 0.8426681    0
## gam      0.9811190 0.9891391 0.9905767 0.9907163 0.9931621 0.9954404    0
## tree     0.9167779 0.9664709 0.9762732 0.9718544 0.9815702 0.9891014    0
## rf       0.9931318 0.9990826 0.9994931 0.9988237 0.9996636 0.9999535    0
## boosting 0.9995095 0.9998702 0.9999628 0.9998967 0.9999834 0.9999977    0
```

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


tree
rf
lasso
enet
pls
pcr
boosting
gam
RMSE