

midterm project

Yixiao Sun

2024-03-19

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
## v dplyr      1.1.3      v readr      2.1.4
## v forcats    1.0.0      v stringr   1.5.0
## v ggplot2    3.4.4      v tibble    3.2.1
## v lubridate  1.9.2      v tidyr     1.3.0
## v purrr      1.0.2
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(summarytools)
```

```
##
## Attaching package: 'summarytools'
##
## The following object is masked from 'package:tibble':
##
##      view
```

```
library(leaps)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(dplyr)
library(ggplot2)
library(ISLR)
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##      expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##   lift
```

```
library(tidymodels)
```

```
## -- Attaching packages ----- tidymodels 1.1.1 --
## v broom      1.0.5      v rsample      1.2.0
## v dials      1.2.0      v tune       1.1.2
## v infer      1.0.5      v workflows  1.1.3
## v modeldata  1.2.0      v workflowsets 1.0.1
## v parsnip    1.1.1      v yardstick  1.2.0
## v recipes    1.0.8
## -- Conflicts ----- tidymodels_conflicts() --
## x scales::discard()      masks purrr::discard()
## x Matrix::expand()       masks tidyr::expand()
## x dplyr::filter()        masks stats::filter()
## x recipes::fixed()       masks stringr::fixed()
## x dplyr::lag()           masks stats::lag()
## x caret::lift()          masks purrr::lift()
## x Matrix::pack()         masks tidyr::pack()
## x yardstick::precision() masks caret::precision()
## x yardstick::recall()    masks caret::recall()
## x yardstick::sensitivity() masks caret::sensitivity()
## x yardstick::spec()      masks readr::spec()
## x yardstick::specificity() masks caret::specificity()
## x recipes::step()        masks stats::step()
## x Matrix::unpack()       masks tidyr::unpack()
## x recipes::update()      masks Matrix::update(), stats::update()
## x summarytools::view()   masks tibble::view()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```
library(plotmo)
```

```
## Loading required package: Formula
## Loading required package: plotrix
##
## Attaching package: 'plotrix'
##
## The following object is masked from 'package:scales':
##
##   rescale
##
## Loading required package: TeachingDemos
```

```
library(caret)
library(tidymodels)
library(splines)
library(mgcv)
```

```
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
##
## This is mgcv 1.9-0. For overview type 'help("mgcv-package")'.
```

```
library(pdp)
```

```
##
## Attaching package: 'pdp'
##
## The following object is masked from 'package:purrr':
##
##     partial
```

```
library(earth)
library(tidyverse)
library(ggplot2)
library(bayesQR)
```

Exploratory analysis and data visualization

```
load("recovery.RData")
st_options(plain.ascii = TRUE,
            style = "rmarkdown",
            dfSummary.silent = TRUE,
            footnote = NA,
            subtitle.emphasis = FALSE)
dfSummary(dat[, -1])
```

```
## Data Frame Summary
## dat
## Dimensions: 3000 x 15
## Duplicates: 0
##
```

##	## -----	##	##	##	## -----
## No	## Variable	## Stats / Values	## Freqs (% of Valid)	## Graph	## Valid
## ----	## -----	## -----	## -----	## -----	## -----
## 1	## age	## Mean (sd) : 60.2 (4.5)	## 34 distinct values	## : .	## 3000
##	## [numeric]	## min < med < max:		## : :	## (100.0%)
##		## 42 < 60 < 79		## : :	
##		## IQR (CV) : 6 (0.1)		## . : : .	

```

##                                     : : : :
##
## 2   gender      Min   : 0                0 : 1544 (51.5%)   I I I I I I I I I I   3000
##     [integer]   Mean : 0.5              1 : 1456 (48.5%)   I I I I I I I I      (100.0%)
##     Max   : 1
##
## 3   race        1. 1                1967 (65.6%)   I I I I I I I I I I I I   3000
##     [factor]    2. 2                158 ( 5.3%)    I                      (100.0%)
##     3. 3                604 (20.1%)   I I I I
##     4. 4                271 ( 9.0%)    I
##
## 4   smoking     1. 0                1822 (60.7%)   I I I I I I I I I I   3000
##     [factor]    2. 1                859 (28.6%)   I I I I                (100.0%)
##     3. 2                319 (10.6%)   I I
##
## 5   height      Mean (sd) : 169.9 (6)    313 distinct values      : :   3000
##     [numeric]   min < med < max:          : :   (100.0%)
##     147.8 < 169.9 < 188.6          . : : .
##     IQR (CV) : 7.9 (0)              : : : :
##     . : : : : .
##
## 6   weight      Mean (sd) : 80 (7.1)    364 distinct values      : .   3000
##     [numeric]   min < med < max:          : :   (100.0%)
##     55.9 < 79.8 < 103.7          : : : :
##     IQR (CV) : 9.6 (0.1)          . : : : : .
##     . : : : : : .
##
## 7   bmi         Mean (sd) : 27.8 (2.8)   163 distinct values      . :   3000
##     [numeric]   min < med < max:          : : :   (100.0%)
##     18.8 < 27.6 < 38.9          : : :
##     IQR (CV) : 3.7 (0.1)          : : : : :
##     . : : : : : .
##
## 8   hypertension Min   : 0                0 : 1508 (50.3%)   I I I I I I I I I I   3000
##     [numeric]   Mean : 0.5              1 : 1492 (49.7%)   I I I I I I I I      (100.0%)
##     Max   : 1
##
## 9   diabetes    Min   : 0                0 : 2537 (84.6%)   I I I I I I I I I I I I I I   3000
##     [integer]   Mean : 0.2              1 : 463 (15.4%)   I I I                  (100.0%)
##     Max   : 1
##
## 10  SBP         Mean (sd) : 130.5 (8)    52 distinct values      : .   3000
##     [numeric]   min < med < max:          : : .   (100.0%)
##     105 < 130 < 156          : : : :
##     IQR (CV) : 11 (0.1)          . : : : : .
##     . : : : : : .
##
## 11  LDL         Mean (sd) : 110.5 (19.8) 114 distinct values      :   3000
##     [numeric]   min < med < max:          : : .   (100.0%)
##     28 < 110 < 178          : : :
##     IQR (CV) : 27 (0.2)          . : : : .
##     . : : : : : .
##
## 12  vaccine     Min   : 0                0 : 1212 (40.4%)   I I I I I I I I   3000

```

```
##      [integer]      Mean : 0.6          1 : 1788 (59.6%)      IIIIIIIIIII      (100.0%)
##                                     Max : 1
##
## 13 severity      Min : 0          0 : 2679 (89.3%)      IIIIIIIIIIIIIIIII 3000
##      [integer]      Mean : 0.1        1 : 321 (10.7%)      II                (100.0%)
##                                     Max : 1
##
## 14 study          1. A          2000 (66.7%)      IIIIIIIIIIIIIII 3000
##      [character]      2. B          1000 (33.3%)      IIIIII            (100.0%)
##
## 15 recovery_time Mean (sd) : 42.2 (23.2) 140 distinct values : : 3000
##      [numeric]      min < med < max: : : (100.0%)
##                                     2 < 39 < 365 : :
##                                     IQR (CV) : 18 (0.5) : :
##                                     : : .
## -----
```

```
summary(dat)
```

```
##      id      age      gender      race      smoking
## Min.   : 1.0   Min.   :42.0   Min.   :0.0000   1:1967   0:1822
## 1st Qu.: 750.8 1st Qu.:57.0   1st Qu.:0.0000   2: 158   1: 859
## Median :1500.5 Median :60.0   Median :0.0000   3: 604   2: 319
## Mean   :1500.5 Mean   :60.2   Mean   :0.4853   4: 271
## 3rd Qu.:2250.2 3rd Qu.:63.0   3rd Qu.:1.0000
## Max.   :3000.0 Max.   :79.0   Max.   :1.0000
##      height      weight      bmi      hypertension
## Min.   :147.8   Min.   : 55.90   Min.   :18.80   Min.   :0.0000
## 1st Qu.:166.0   1st Qu.: 75.20   1st Qu.:25.80   1st Qu.:0.0000
## Median :169.9   Median : 79.80   Median :27.65   Median :0.0000
## Mean   :169.9   Mean   : 79.96   Mean   :27.76   Mean   :0.4973
## 3rd Qu.:173.9   3rd Qu.: 84.80   3rd Qu.:29.50   3rd Qu.:1.0000
## Max.   :188.6   Max.   :103.70   Max.   :38.90   Max.   :1.0000
##      diabetes      SBP      LDL      vaccine
## Min.   :0.0000   Min.   :105.0   Min.   : 28.0   Min.   :0.000
## 1st Qu.:0.0000   1st Qu.:125.0   1st Qu.: 97.0   1st Qu.:0.000
## Median :0.0000   Median :130.0   Median :110.0   Median :1.000
## Mean   :0.1543   Mean   :130.5   Mean   :110.5   Mean   :0.596
## 3rd Qu.:0.0000   3rd Qu.:136.0   3rd Qu.:124.0   3rd Qu.:1.000
## Max.   :1.0000   Max.   :156.0   Max.   :178.0   Max.   :1.000
##      severity      study      recovery_time
## Min.   :0.000   Length:3000   Min.   : 2.00
## 1st Qu.:0.000   Class :character 1st Qu.: 31.00
## Median :0.000   Mode  :character Median : 39.00
## Mean   :0.107           Mean   : 42.17
## 3rd Qu.:0.000           3rd Qu.: 49.00
## Max.   :1.000           Max.   :365.00
```

```
columns_to_convert <- c("gender", "race", "smoking", "hypertension", "diabetes", "vaccine", "severity")

dat$study <- as.character(dat$study)
unique(dat$study)
```

```
## [1] "A" "B"
```

```

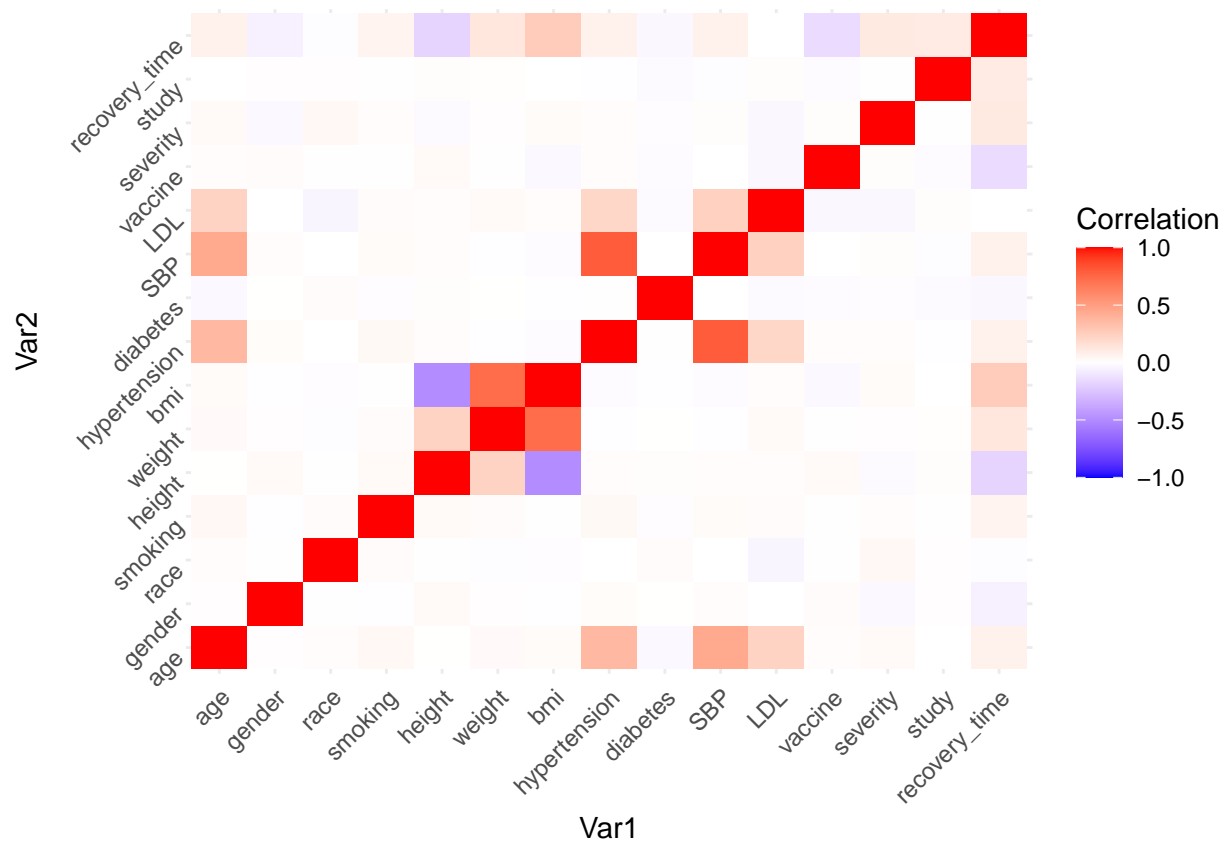
# Convert selected factor variables to numeric using mutate
dat <- dat %>%
  mutate(across(all_of(columns_to_convert), as.numeric)) %>%
  mutate(study = case_when(
    study == "A" ~ 1,
    study == "B" ~ 2
  ))

numeric_data <- dat[, c("age", "gender", "race", "smoking", "height", "weight", "bmi", "hypertension",
                        "diabetes", "SBP", "LDL", "vaccine", "severity", "study", "recovery_time")]

# Compute correlation matrix
correlation_matrix <- cor(numeric_data)
correlation_df <- as.data.frame(as.table(correlation_matrix))
names(correlation_df) <- c("Var1", "Var2", "Correlation")

ggplot(correlation_df, aes(x = Var1, y = Var2, fill = Correlation)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red",
                      midpoint = 0, limits = c(-1, 1),
                      name = "Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1),
        axis.text.y = element_text(angle = 45, vjust = 1, hjust = 1))

```



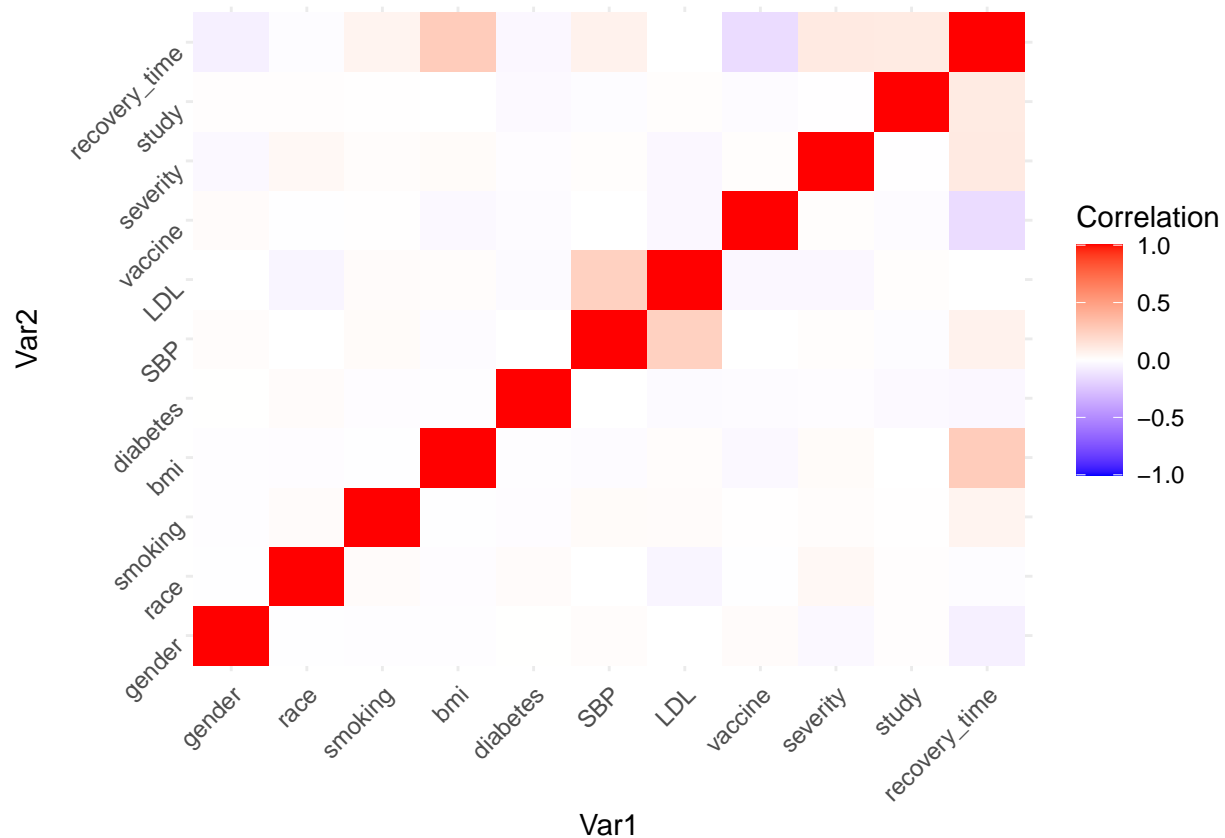
```

dat_subset <- dat %>%
  select(-id, -height, -weight, -hypertension, -age)
numeric_data <- dat_subset[, c("gender", "race", "smoking", "bmi", "diabetes", "SBP", "LDL", "vaccine",
                                "severity", "study", "recovery_time")]

# Compute correlation matrix
correlation_matrix <- cor(numeric_data)
correlation_df <- as.data.frame(as.table(correlation_matrix))
names(correlation_df) <- c("Var1", "Var2", "Correlation")

ggplot(correlation_df, aes(x = Var1, y = Var2, fill = Correlation)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red",
                      midpoint = 0, limits = c(-1, 1),
                      name = "Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1),
        axis.text.y = element_text(angle = 45, vjust = 1, hjust = 1))

```



Model training

```

ctrl1 <- trainControl(method = "cv", number = 10)
set.seed(1)
data <-
  dat_subset %>%
  mutate(gender = as.factor(gender),
         race = as.factor(race),

```

```

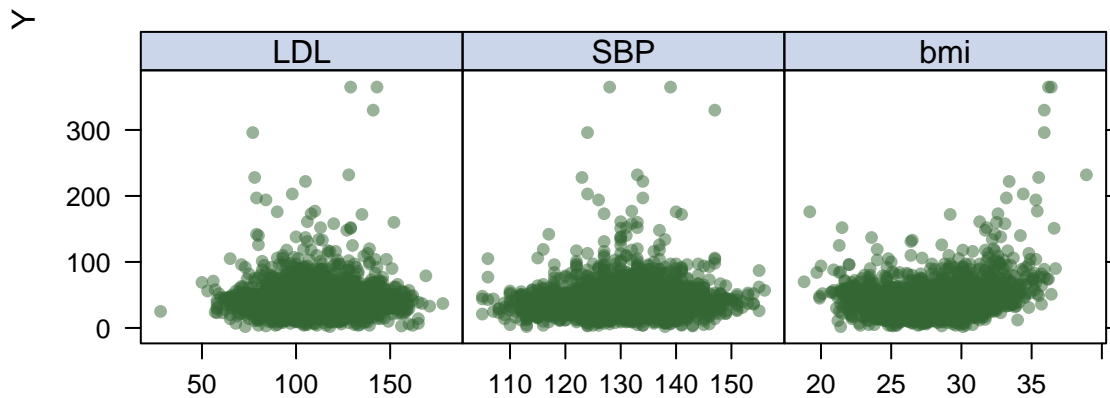
    smoking = as.factor(smoking),
    diabetes = as.factor(diabetes),
    vaccine = as.factor(vaccine),
    severity = as.factor(severity),
    study = as.factor(study))
data <- data %>% mutate(bmi2 = bmi^2,
                       bmi3 = bmi^3,
                       bmi4 = bmi^4)
data_split <- initial_split(data, prop = 0.8)
# Extract the training and test data
training_data <- training(data_split)
testing_data <- testing(data_split)

x<-model.matrix(recovery_time ~ ., data)[-1]
y<-data$recovery_time

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[, c('LDL','SBP','bmi')], y, plot = "scatter", labels = c("", "Y"), type = c("p"), layout =

```




```

fit1 <- lm(recovery_time ~ LDL, data = data)
fit2 <- lm(recovery_time ~ poly(LDL,2), data = data)
fit3 <- lm(recovery_time ~ poly(LDL,3), data = data)
fit4 <- lm(recovery_time ~ poly(LDL,4), data = data)
fit5 <- lm(recovery_time ~ poly(LDL,5), data = data)
anova(fit1, fit2, fit3, fit4, fit5)

```

```

## Analysis of Variance Table
##
## Model 1: recovery_time ~ LDL
## Model 2: recovery_time ~ poly(LDL, 2)
## Model 3: recovery_time ~ poly(LDL, 3)
## Model 4: recovery_time ~ poly(LDL, 4)
## Model 5: recovery_time ~ poly(LDL, 5)
##   Res.Df    RSS Df Sum of Sq    F Pr(>F)
## 1     2998 1607736
## 2     2997 1607696   1      39.47 0.0736 0.7862
## 3     2996 1607693   1       3.50 0.0065 0.9356
## 4     2995 1606441   1    1251.37 2.3326 0.1268
## 5     2994 1606218   1     222.94 0.4156 0.5192

```

```

fit1 <- lm(recovery_time ~ bmi, data = data)
fit2 <- lm(recovery_time ~ poly(bmi,2), data = data)
fit3 <- lm(recovery_time ~ poly(bmi,3), data = data)
fit4 <- lm(recovery_time ~ poly(bmi,4), data = data)
fit5 <- lm(recovery_time ~ poly(bmi,5), data = data)
anova(fit1, fit2, fit3, fit4, fit5)

```

```

## Analysis of Variance Table
##
## Model 1: recovery_time ~ bmi
## Model 2: recovery_time ~ poly(bmi, 2)
## Model 3: recovery_time ~ poly(bmi, 3)
## Model 4: recovery_time ~ poly(bmi, 4)
## Model 5: recovery_time ~ poly(bmi, 5)
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1     2998 1493906
## 2     2997 1225737   1    268169 672.9018 < 2.2e-16 ***
## 3     2996 1209491   1     16246  40.7645 1.984e-10 ***
## 4     2995 1193212   1     16279  40.8479 1.902e-10 ***
## 5     2994 1193186   1        26   0.0654   0.7981
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```

fit1 <- lm(recovery_time ~ SBP, data = data)
fit2 <- lm(recovery_time ~ poly(SBP,2), data = data)
fit3 <- lm(recovery_time ~ poly(SBP,3), data = data)
fit4 <- lm(recovery_time ~ poly(SBP,4), data = data)
fit5 <- lm(recovery_time ~ poly(SBP,5), data = data)
anova(fit1, fit2, fit3, fit4, fit5)

```

```

## Analysis of Variance Table

```

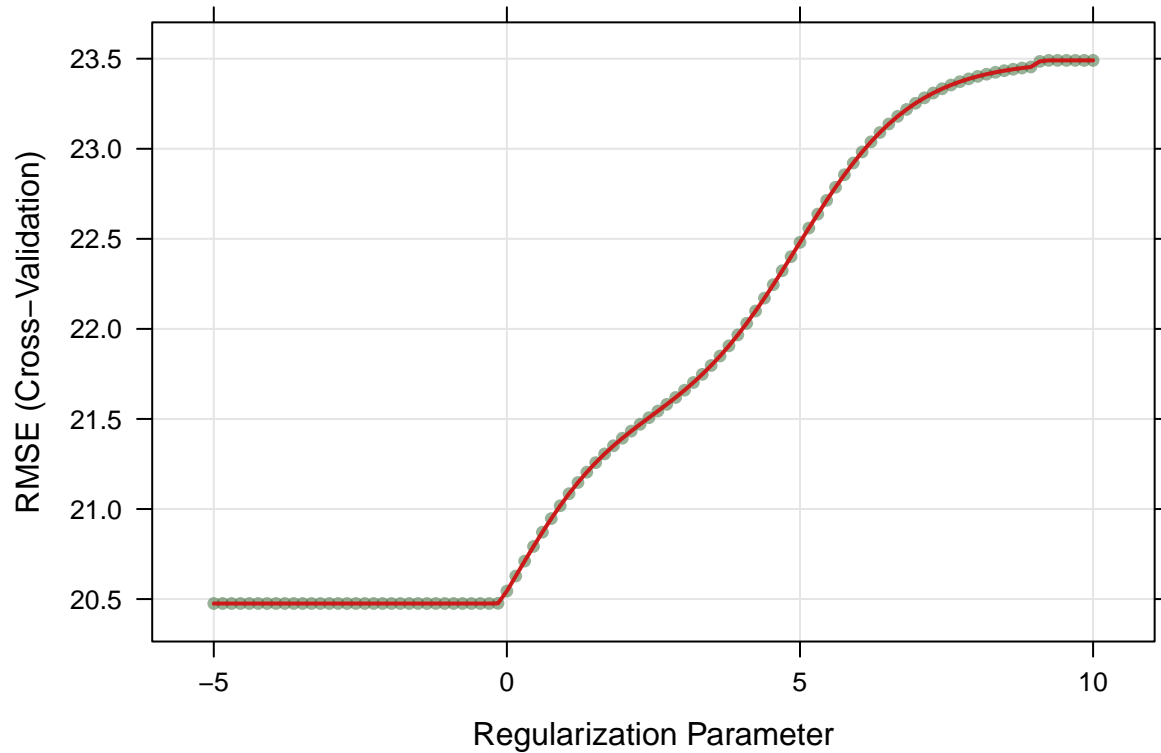
```
##
## Model 1: recovery_time ~ SBP
## Model 2: recovery_time ~ poly(SBP, 2)
## Model 3: recovery_time ~ poly(SBP, 3)
## Model 4: recovery_time ~ poly(SBP, 4)
## Model 5: recovery_time ~ poly(SBP, 5)
##   Res.Df    RSS Df Sum of Sq    F Pr(>F)
## 1   2998 1600160
## 2   2997 1599272   1    888.48 1.6649 0.1970
## 3   2996 1599086   1    185.73 0.3480 0.5553
## 4   2995 1597839   1   1247.49 2.3376 0.1264
## 5   2994 1597780   1     58.28 0.1092 0.7411
```

Ridge regression

```
set.seed(1)
ridge.fit <- train(recovery_time ~ . ,
  data = training_data,
  method = "glmnet",
  tuneGrid = expand.grid(alpha = 0,
    lambda = exp(seq(10, -5, length=100))),
  trControl = ctrl1)
```

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

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



```
ridge.fit$bestTune
```

```
##      alpha      lambda
## 32      0 0.7385767
```

```
coef(ridge.fit$finalModel, s = ridge.fit$bestTune$lambda)
```

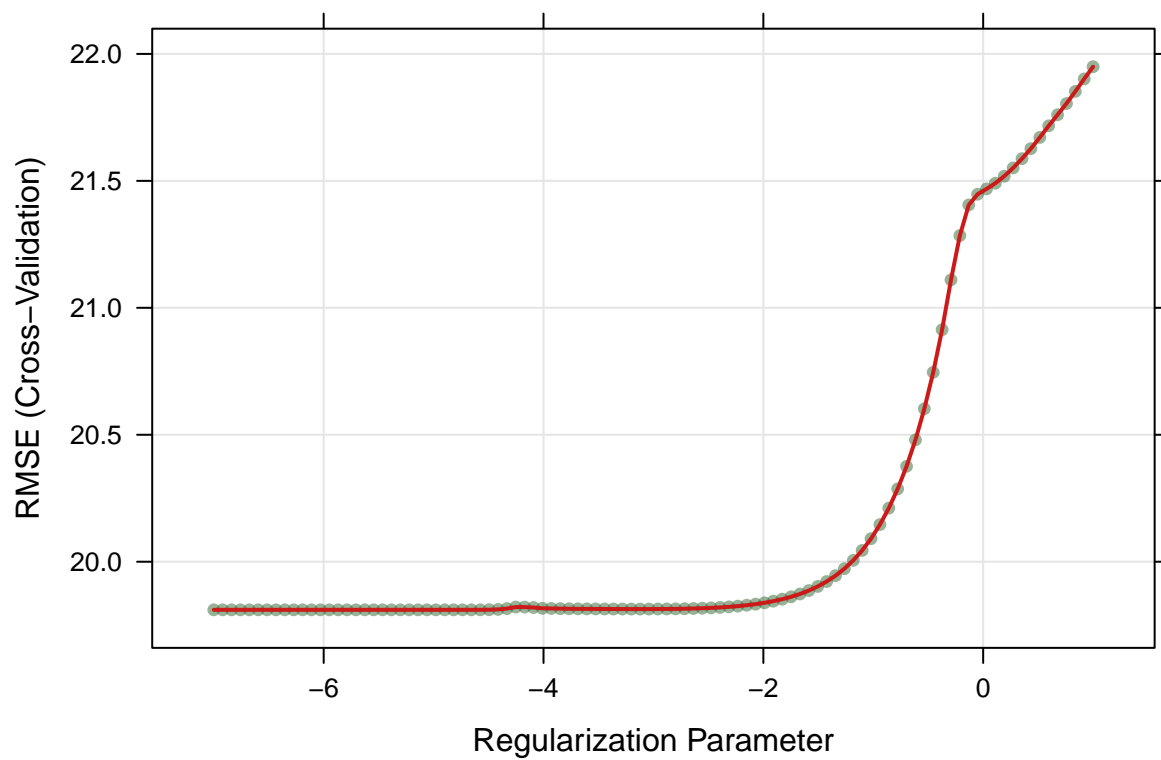
```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##              s1
## (Intercept) 1.019376e+02
## gender1     -2.983263e+00
## race2        9.675356e-01
## race3       -5.618009e-01
## race4       -1.027628e+00
## smoking2     2.802962e+00
## smoking3     3.505736e+00
## bmi          -4.734344e+00
## diabetes1    -1.554729e+00
## SBP          2.239282e-01
## LDL         -3.285523e-02
## vaccine1     -6.225018e+00
## severity1    7.638955e+00
## study2       4.646690e+00
## bmi2        -2.106367e-02
## bmi3         1.034046e-03
```

```
## bmi4          6.611743e-05
```

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

```
## [1] 329.9334
```

```
set.seed(1)
lasso.fit <- train(recovery_time ~ .,
  data = training_data,
  method = "glmnet",
  tuneGrid = expand.grid(alpha = 1,
    lambda = exp(seq(1, -7, length = 100))),
  trControl = ctrl1)
plot(lasso.fit, xTrans = log)
```



```
lasso.fit$bestTune
```

```
##      alpha      lambda
## 30      1 0.009499029
```

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

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##              s1
## (Intercept) 169.155809970
## gender1     -3.574149920
## race2        0.156095718
## race3       -0.323448879
## race4       -0.675742585
## smoking2     3.274206336
## smoking3     4.098609493
## bmi         -1.753584488
## diabetes1   -1.217219485
## SBP          0.198048448
## LDL         -0.030632238
## vaccine1    -6.472767874
## severity1    7.713670120
## study2       4.593372747
## bmi2        -0.325460260
## bmi3         .
## bmi4         0.000245325
```

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

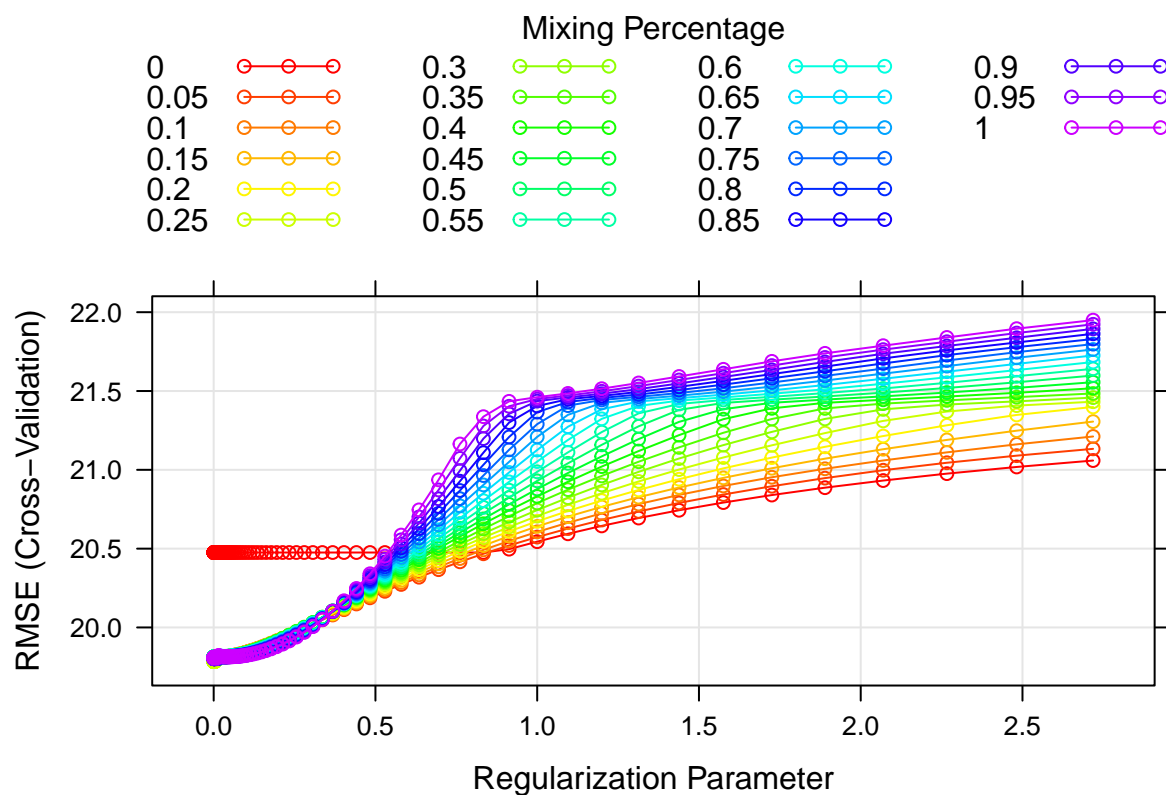
```
## [1] 304.8401
```

Elastic Net

```
set.seed(1)
enet.fit <- train(recovery_time ~ .,
                  data = training_data,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(1, -8, length = 100))),
                  trControl = ctrl1)
enet.fit$bestTune
```

```
##      alpha      lambda
## 526  0.25 0.003255945
```

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



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

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##              s1
## (Intercept) 193.901653987
## gender1     -3.592612004
## race2        0.183024786
## race3       -0.348096094
## race4       -0.712789065
## smoking2      3.304545993
## smoking3      4.135463922
## bmi          -4.469201129
## diabetes1    -1.242388991
## SBP           0.199417123
## LDL          -0.031210770
## vaccine1     -6.491738379
## severity1     7.736171655
## study2        4.611309984
## bmi2         -0.222775216
## bmi3         -0.001498305
## bmi4          0.000251091
```

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

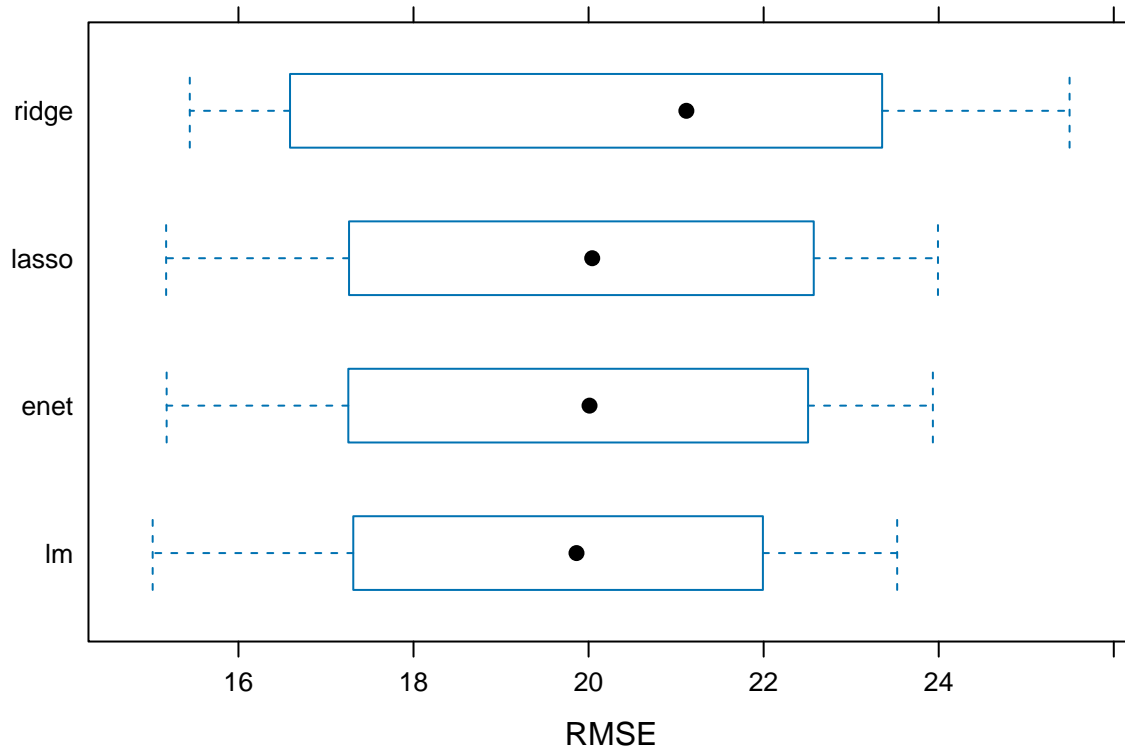
```
## [1] 304.6396
```

Comparison

```
set.seed(1)
lm.fit <- train(recovery_time ~ .,
               data = training_data,
               method = "lm",
               trControl = ctrl1)
resamp <- resamples(list(enet =enet.fit, lasso = lasso.fit, ridge = ridge.fit, lm = lm.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models:enet, lasso, ridge, lm
## Number of resamples: 10
##
## MAE
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max. NA's
## enet  11.19243 12.28303 12.87973 12.82832 13.28650 14.37349    0
## lasso 11.18210 12.28336 12.88231 12.83247 13.31513 14.37639    0
## ridge 11.39562 11.99400 12.89329 12.84582 13.69245 13.94910    0
## lm    11.03198 12.32874 12.67890 12.74138 13.13532 14.46433    0
##
## RMSE
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max. NA's
## enet  15.17942 17.28326 20.01063 19.78329 22.43991 23.93315    0
## lasso 15.17382 17.28987 20.04142 19.81051 22.50011 23.99149    0
## ridge 15.44401 17.11638 21.11696 20.47543 23.26148 25.49494    0
## lm    15.01978 17.52849 19.86189 19.79750 21.96821 23.52521    0
##
## Rsquared
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max. NA's
## enet  0.1574160 0.2137419 0.3165027 0.2951554 0.3364036 0.5131303    0
## lasso 0.1571902 0.2135190 0.3140797 0.2930746 0.3314665 0.5126260    0
## ridge 0.1666250 0.1876036 0.2471011 0.2486397 0.2566815 0.4165697    0
## lm    0.1527942 0.2141530 0.3067285 0.3054667 0.3788317 0.5158586    0
```

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



PCR

```
# show information about the model
modelLookup("pcr")
```

```
##   model parameter      label forReg forClass probModel
## 1  pcr      ncomp #Components   TRUE   FALSE   FALSE
```

```
modelLookup("pls")
```

```
##   model parameter      label forReg forClass probModel
## 1  pls      ncomp #Components   TRUE   TRUE   TRUE
```

```
#x <- model.matrix(recovery_time ~ ., training_data)[, -1]
#y <- training_data$recovery_time
# test data
#x2 <- model.matrix(recovery_time ~ ., testing_data)[, -1]
#y2 <- testing_data$recovery_time
```

```
set.seed(1)
pcr.fit <- train(recovery_time ~ .,
  data = training_data,
  method = "pcr",
  tuneGrid = data.frame(ncomp = 1:17),
  trControl = ctrl1,
```



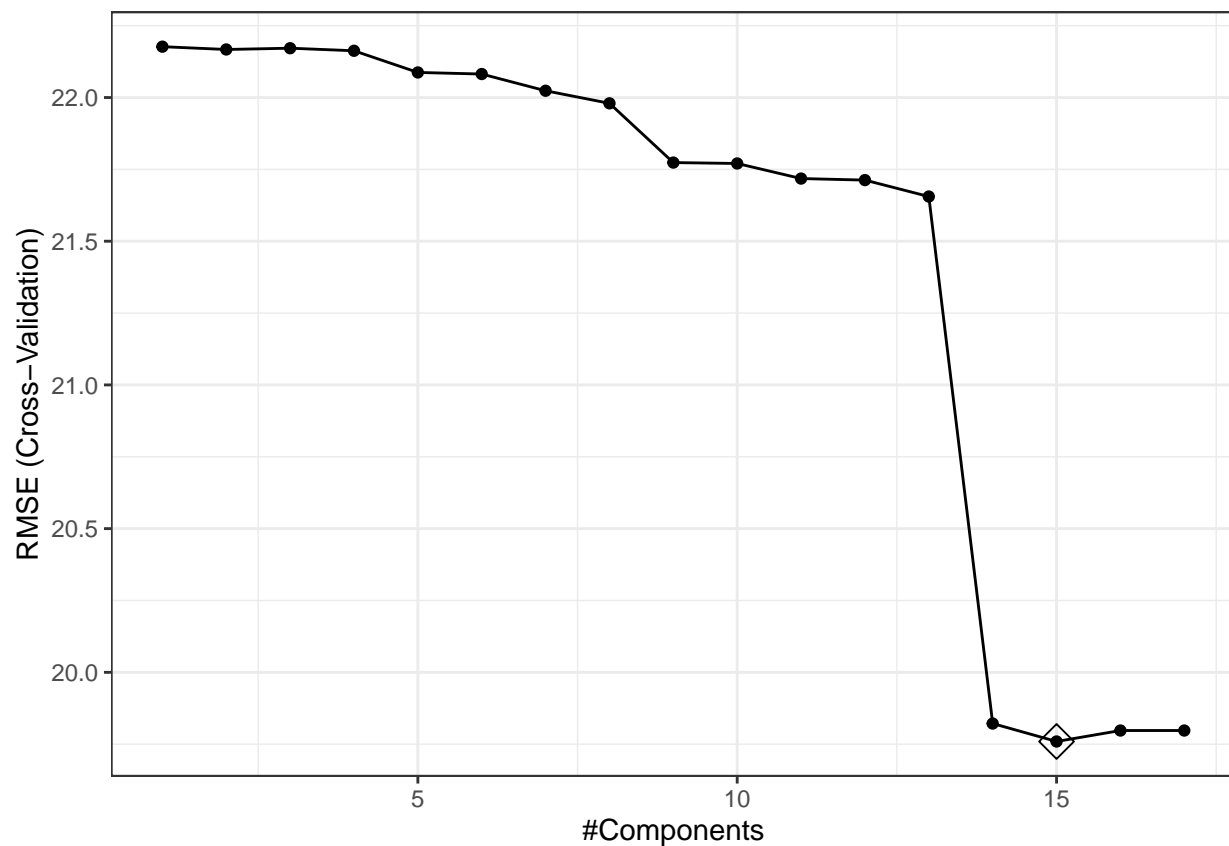
```
preProcess = c("center", "scale"))
pcr.fit$bestTune
```

```
##      ncomp
## 15      15
```

```
predy2.pcr2 <- predict(pcr.fit, newdata = testing_data)
mean((y - predy2.pcr2)^2)
```

```
## [1] 662.9783
```

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



PLS

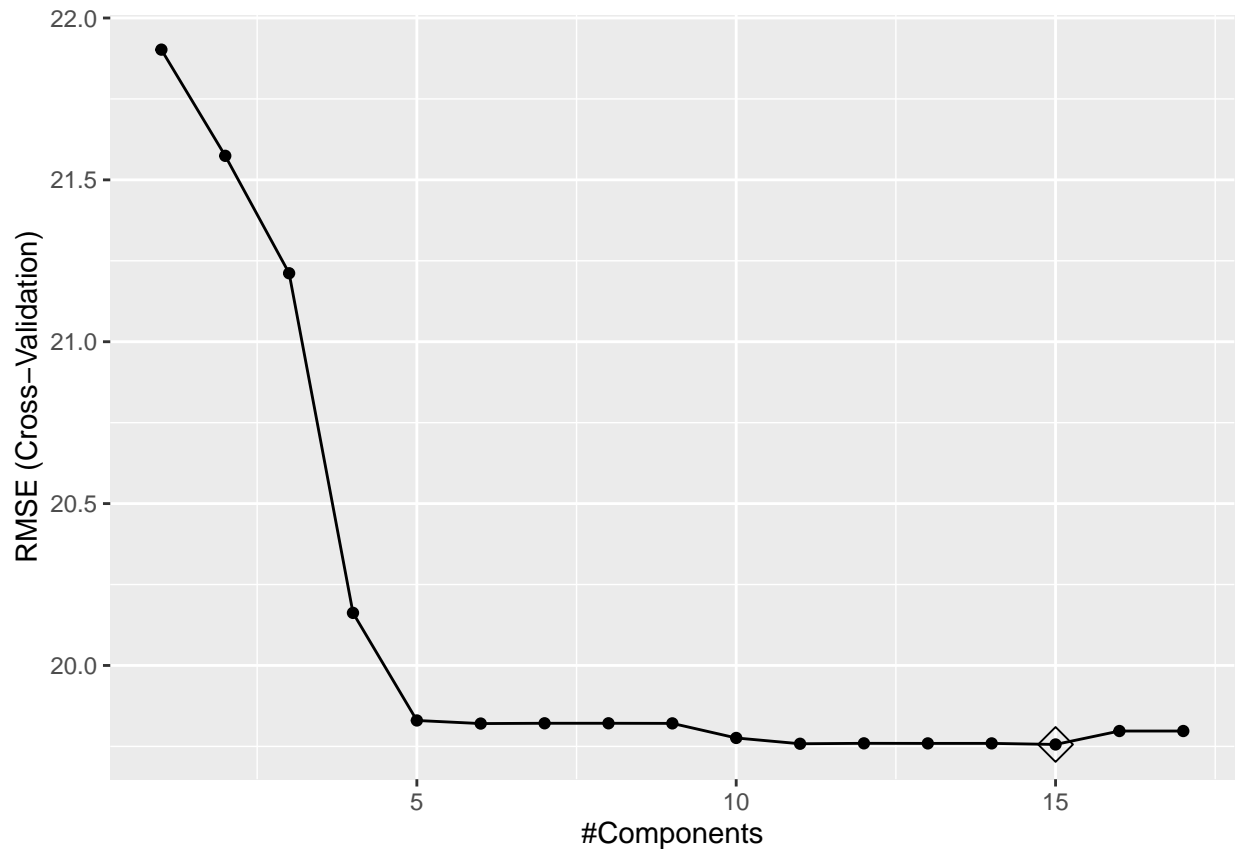
```
set.seed(1)
pls.fit <- train(recovery_time ~ .,
  data = training_data,
  method = "pls",
  tuneGrid = data.frame(ncomp = 1:17),
  trControl = ctrl1,
  preProcess = c("center", "scale"))
predy2.pls2 <- predict(pls.fit, newdata = testing_data)
mean((y - predy2.pls2)^2)
```

```
## [1] 662.9475
```

```
pls.fit$bestTune
```

```
##      ncomp  
## 15      15
```

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

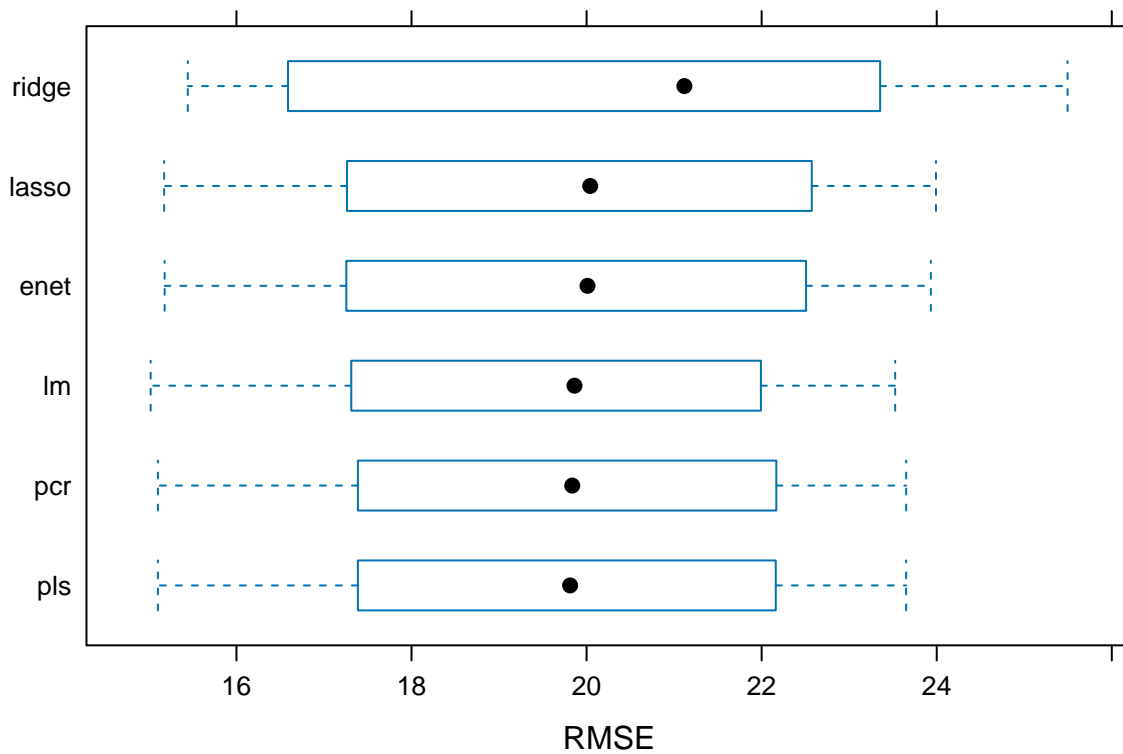


```
resamp <- resamples(list(enet =enet.fit, lasso = lasso.fit, ridge = ridge.fit, lm = lm.fit,pls = pls.f  
summary(resamp)
```

```
##  
## Call:  
## summary.resamples(object = resamp)  
##  
## Models:enet, lasso, ridge, lm, pls, pcr  
## Number of resamples: 10  
##  
## MAE  
##           Min. 1st Qu.  Median    Mean 3rd Qu.    Max. NA's  
## enet  11.19243 12.28303 12.87973 12.82832 13.28650 14.37349    0  
## lasso 11.18210 12.28336 12.88231 12.83247 13.31513 14.37639    0  
## ridge 11.39562 11.99400 12.89329 12.84582 13.69245 13.94910    0
```

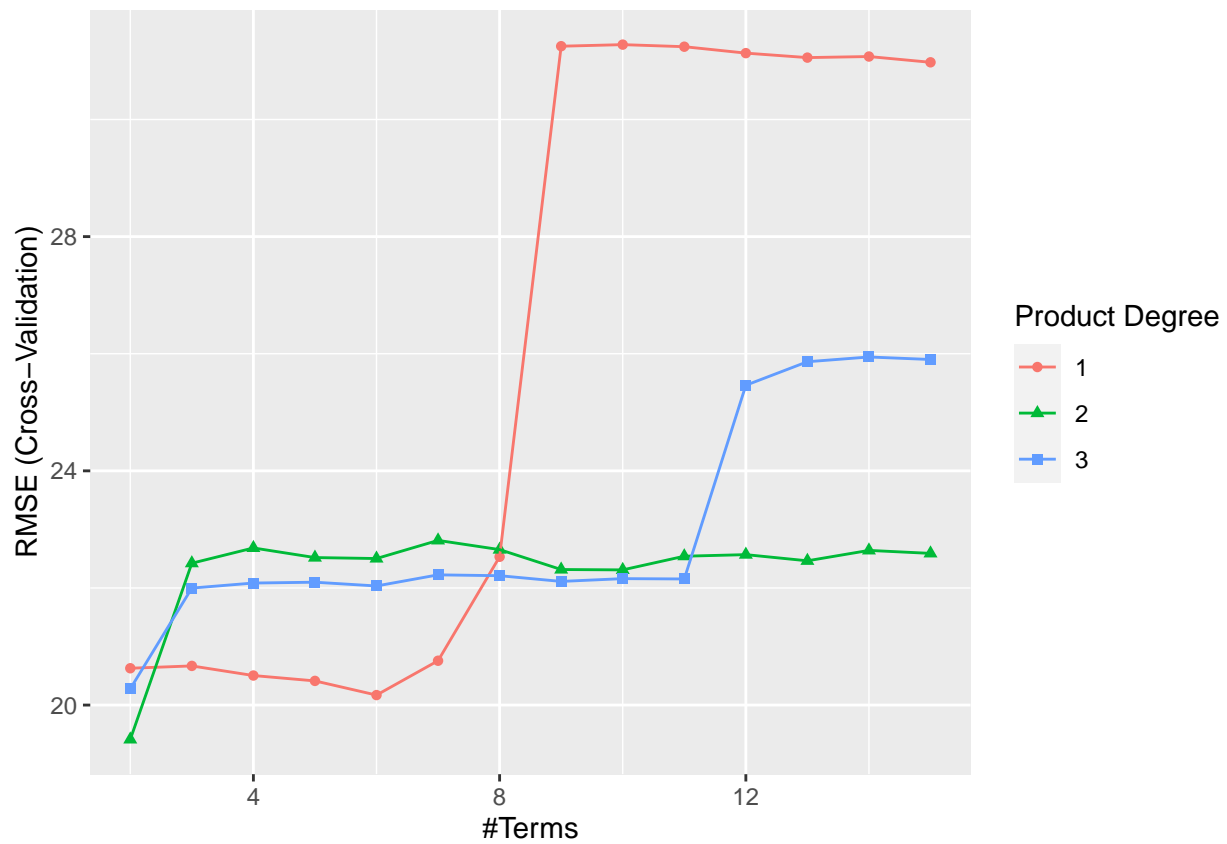
```
## lm      11.03198 12.32874 12.67890 12.74138 13.13532 14.46433    0
## pls     11.09836 12.30310 12.87270 12.79128 13.19942 14.42377    0
## pcr     11.09891 12.30337 12.85146 12.78847 13.19881 14.42453    0
##
## RMSE
##           Min.   1st Qu.   Median     Mean   3rd Qu.     Max. NA's
## enet  15.17942 17.28326 20.01063 19.78329 22.43991 23.93315    0
## lasso 15.17382 17.28987 20.04142 19.81051 22.50011 23.99149    0
## ridge 15.44401 17.11638 21.11696 20.47543 23.26148 25.49494    0
## lm     15.01978 17.52849 19.86189 19.79750 21.96821 23.52521    0
## pls     15.10279 17.43433 19.81192 19.75604 22.13012 23.65080    0
## pcr     15.10280 17.43486 19.83610 19.75938 22.14057 23.65119    0
##
## Rsquared
##           Min.   1st Qu.   Median     Mean   3rd Qu.     Max. NA's
## enet  0.1574160 0.2137419 0.3165027 0.2951554 0.3364036 0.5131303    0
## lasso 0.1571902 0.2135190 0.3140797 0.2930746 0.3314665 0.5126260    0
## ridge 0.1666250 0.1876036 0.2471011 0.2486397 0.2566815 0.4165697    0
## lm     0.1527942 0.2141530 0.3067285 0.3054667 0.3788317 0.5158586    0
## pls     0.1534147 0.2135833 0.3136269 0.3008433 0.3627103 0.5067731    0
## pcr     0.1533912 0.2114601 0.3137550 0.3005294 0.3625788 0.5067576    0
```

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



MARS

```
mars_grid <- expand.grid(degree = 1:3, nprune = 2:15)
set.seed(1)
mars.fit <- train(recovery_time ~ ., data = training_data,
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl1)
ggplot(mars.fit)
```



```
mars.fit$bestTune
```

```
##      nprune degree
## 15         2      2
```

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

```
## [1] 350.9301
```

```
set.seed(1)
gam.fit <- train(recovery_time ~ .,
```

```

        data = training_data,
        method = "gam",
        trControl = ctrl1)
gam.fit$bestTune

```

```

## select method
## 2 TRUE GCV.Cp

```

```
gam.fit$finalModel
```

```

##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ gender1 + race2 + race3 + race4 + smoking2 + smoking3 +
## diabetes1 + vaccine1 + severity1 + study2 + s(SBP) + s(LDL) +
## s(bmi) + s(bmi2) + s(bmi3) + s(bmi4)
##
## Estimated degrees of freedom:
## 0.9019 4.6205 0.5665 0.0619 8.0001 0.8229 total = 25.97
##
## GCV score: 382.9841

```

```

gam_pred <- predict(gam.fit, newdata = testing_data)
mean((gam_pred - testing_data$recovery_time)^2)

```

```
## [1] 307.2322
```

```

set.seed(1)
resamp <- resamples(list(enet =enet.fit, lasso = lasso.fit, ridge = ridge.fit, lm = lm.fit,pls = pls.f
summary(resamp)

```

```

##
## Call:
## summary.resamples(object = resamp)
##
## Models:enet, lasso, ridge, lm, pls, pcr, mars, gam
## Number of resamples: 10
##
## MAE
##      Min.   1st Qu.   Median     Mean  3rd Qu.   Max. NA's
## enet  11.19243 12.28303 12.87973 12.82832 13.28650 14.37349    0
## lasso 11.18210 12.28336 12.88231 12.83247 13.31513 14.37639    0
## ridge 11.39562 11.99400 12.89329 12.84582 13.69245 13.94910    0
## lm    11.03198 12.32874 12.67890 12.74138 13.13532 14.46433    0
## pls   11.09836 12.30310 12.87270 12.79128 13.19942 14.42377    0
## pcr   11.09891 12.30337 12.85146 12.78847 13.19881 14.42453    0
## mars  12.03188 12.39737 12.74942 12.84437 13.40388 13.69395    0
## gam   11.01263 12.33175 12.53916 12.81030 13.12889 15.67349    0
##

```

```
## RMSE
##           Min.   1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## enet  15.17942 17.28326 20.01063 19.78329 22.43991 23.93315    0
## lasso 15.17382 17.28987 20.04142 19.81051 22.50011 23.99149    0
## ridge 15.44401 17.11638 21.11696 20.47543 23.26148 25.49494    0
## lm    15.01978 17.52849 19.86189 19.79750 21.96821 23.52521    0
## pls   15.10279 17.43433 19.81192 19.75604 22.13012 23.65080    0
## pcr   15.10280 17.43486 19.83610 19.75938 22.14057 23.65119    0
## mars  16.69585 17.22443 20.26374 19.41309 21.18100 21.44825    0
## gam   15.06042 17.64721 19.69238 20.15536 21.28434 29.43546    0
##
## Rsquared
##           Min.   1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## enet  0.15741603 0.2137419 0.3165027 0.2951554 0.3364036 0.5131303    0
## lasso 0.15719022 0.2135190 0.3140797 0.2930746 0.3314665 0.5126260    0
## ridge 0.16662495 0.1876036 0.2471011 0.2486397 0.2566815 0.4165697    0
## lm    0.15279421 0.2141530 0.3067285 0.3054667 0.3788317 0.5158586    0
## pls   0.15341473 0.2135833 0.3136269 0.3008433 0.3627103 0.5067731    0
## pcr   0.15339123 0.2114601 0.3137550 0.3005294 0.3625788 0.5067576    0
## mars  0.02656562 0.1765903 0.3581283 0.3234134 0.4885344 0.5032351    0
## gam   0.14930906 0.1801864 0.3050085 0.3057842 0.4056275 0.5161894    0
```

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

