P8106_midterm

lz2951

2024-03-28

```
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   3.4.3     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(ggcorrplot)
library(pheatmap)
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        1.2.1     v tune         1.1.2
## v infer        1.0.5     v workflows    1.1.4
## v modeldata    1.3.0     v workflowsets 1.0.1
## v parsnip      1.2.0     v yardstick    1.3.0
## v recipes      1.0.9
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard()        masks purrr::discard()
## 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 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()
## * Search for functions across packages at https://www.tidymodels.org/find/
```

Import Data

```
load("recovery.RData")

str(dat)
```

```
## 'data.frame':    3000 obs. of  16 variables:
##  $ id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ age          : num  56 70 57 53 59 60 56 58 60 60 ...
##  $ gender       : int  0 1 1 0 1 1 0 1 0 1 ...
##  $ race         : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 3 1 1 1 1 ...
##  $ smoking      : Factor w/ 3 levels "0","1","2": 3 2 1 1 3 2 1 1 2 1 ...
##  $ height       : num  170 170 168 167 174 ...
##  $ weight       : num  78.7 73.1 77.4 76.1 70.2 75.1 79.1 62.6 81.8 75.7 ...
##  $ bmi          : num  27.2 25.4 27.3 27.4 23.3 28.4 27.5 26.8 28.8 27.3 ...
##  $ hypertension : num  0 1 1 0 0 0 0 1 1 0 ...
##  $ diabetes     : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ SBP          : num  120 134 131 115 127 129 122 134 136 127 ...
##  $ LDL          : num  97 112 88 87 118 104 66 104 126 123 ...
##  $ vaccine      : int  0 0 1 0 1 0 0 1 1 1 ...
##  $ severity     : int  0 0 0 1 0 0 0 0 1 0 ...
##  $ study        : chr  "A" "A" "A" "A" ...
##  $ recovery_time: num  31 44 29 47 40 34 31 41 50 33 ...
```

```
recovery = dat |>
  janitor::clean_names() |>
  mutate(gender = as.factor(gender),
         hypertension = as.factor(hypertension),
         diabetes = as.factor(diabetes),
         vaccine = as.factor(vaccine),
         severity = as.factor(severity),
         study = as.factor(study)) |>
  select(-id)

str(recovery)
```

```
## 'data.frame':    3000 obs. of  15 variables:
##  $ age          : num  56 70 57 53 59 60 56 58 60 60 ...
##  $ gender       : Factor w/ 2 levels "0","1": 1 2 2 1 2 2 1 2 1 2 ...
##  $ race         : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 3 1 1 1 1 ...
##  $ smoking      : Factor w/ 3 levels "0","1","2": 3 2 1 1 3 2 1 1 2 1 ...
##  $ height       : num  170 170 168 167 174 ...
```

```
##  $ weight       : num  78.7 73.1 77.4 76.1 70.2 75.1 79.1 62.6 81.8 75.7 ...
##  $ bmi          : num  27.2 25.4 27.3 27.4 23.3 28.4 27.5 26.8 28.8 27.3 ...
##  $ hypertension : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 2 2 1 ...
##  $ diabetes     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...
##  $ sbp          : num  120 134 131 115 127 129 122 134 136 127 ...
##  $ ldl          : num  97 112 88 87 118 104 66 104 126 123 ...
##  $ vaccine      : Factor w/ 2 levels "0","1": 1 1 2 1 2 1 1 2 2 2 ...
##  $ severity     : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 2 1 ...
##  $ study        : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...
##  $ recovery_time: num  31 44 29 47 40 34 31 41 50 33 ...
```

Exploratory analysis and data visualization

```
skimr::skim(recovery) |>
  select(-numeric.hist)
```

Table 1: Data summary

Name	recovery
Number of rows	3000
Number of columns	15
Column type frequency:	
factor	8
numeric	7
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
gender	0	1	FALSE	2	0: 1544, 1: 1456
race	0	1	FALSE	4	1: 1967, 3: 604, 4: 271, 2: 158
smoking	0	1	FALSE	3	0: 1822, 1: 859, 2: 319
hypertension	0	1	FALSE	2	0: 1508, 1: 1492
diabetes	0	1	FALSE	2	0: 2537, 1: 463
vaccine	0	1	FALSE	2	1: 1788, 0: 1212
severity	0	1	FALSE	2	0: 2679, 1: 321
study	0	1	FALSE	2	A: 2000, B: 1000

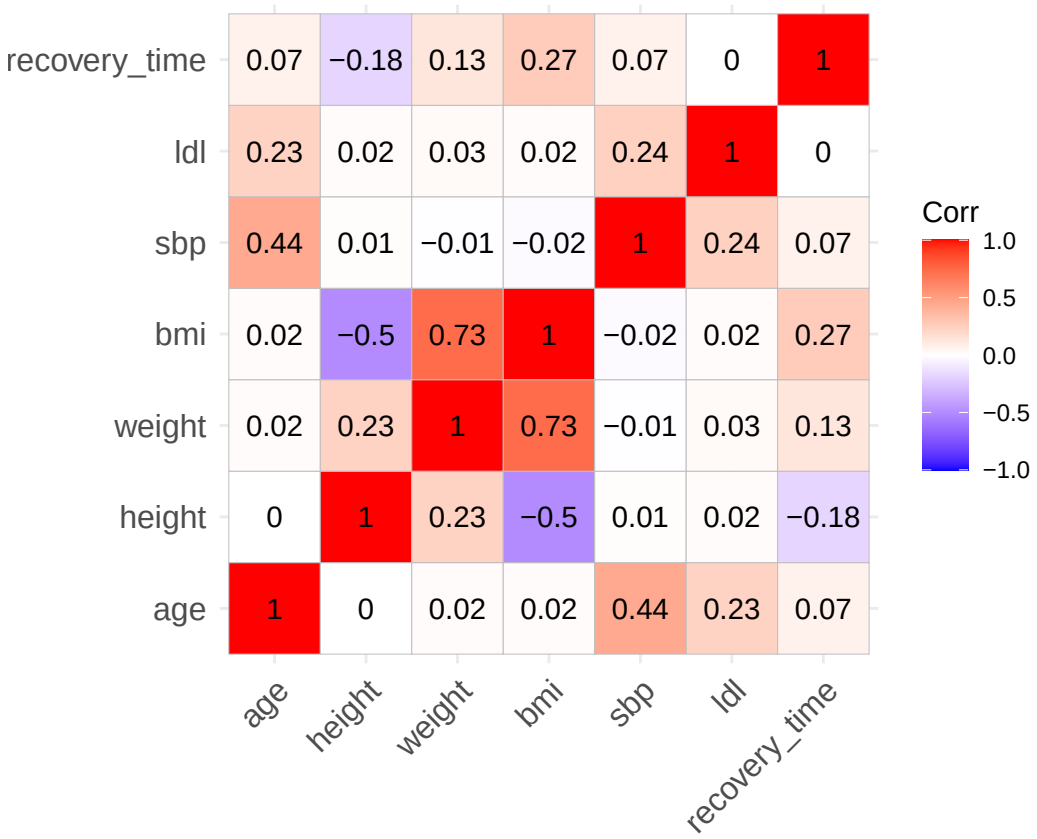
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
age	0	1	60.20	4.48	42.0	57.0	60.00	63.0	79.0
height	0	1	169.90	5.97	147.8	166.0	169.90	173.9	188.6
weight	0	1	79.96	7.14	55.9	75.2	79.80	84.8	103.7
bmi	0	1	27.76	2.79	18.8	25.8	27.65	29.5	38.9

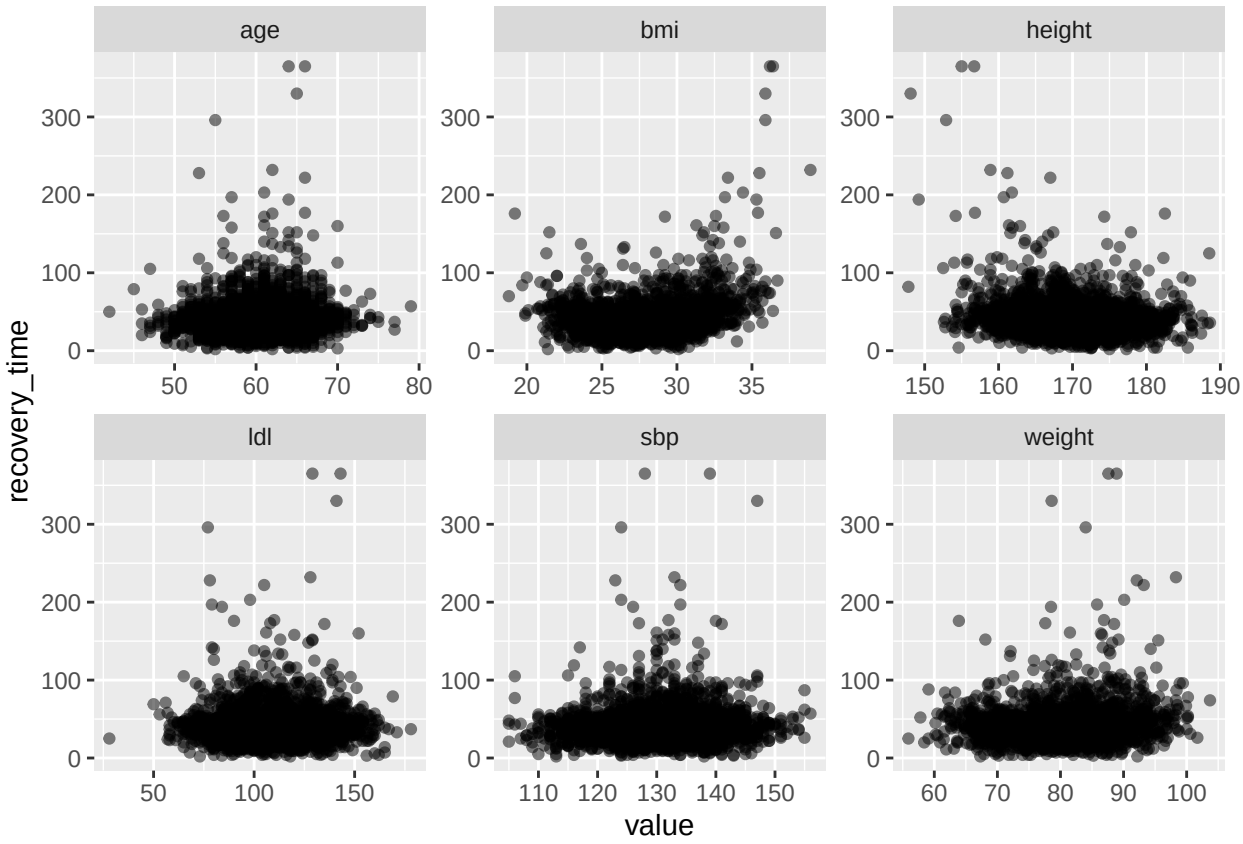
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
sbp	0	1	130.47	7.97	105.0	125.0	130.00	136.0	156.0
ldl	0	1	110.45	19.76	28.0	97.0	110.00	124.0	178.0
recovery_time	0	1	42.17	23.15	2.0	31.0	39.00	49.0	365.0

Analysis between numeric predictors

```
recovery_numeric =
  recovery |>
  select(where(is.numeric))

# recovery_numeric

ggcorrplot(cor(recovery_numeric), lab = T)
```



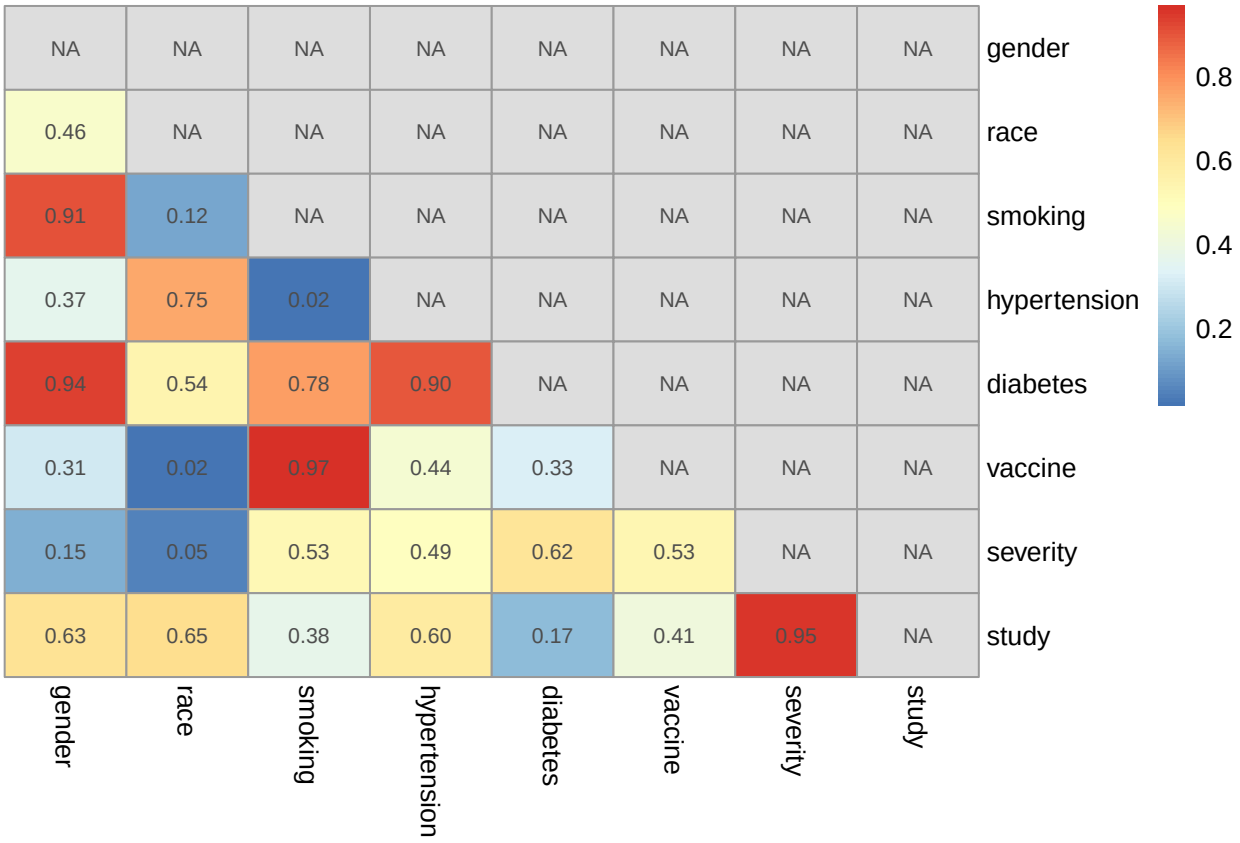

```
recovery_numeric_long =
  recovery_numeric |>
  gather(key = "predictor", value = "value", -recovery_time)

# recovery_numeric_long

ggplot(recovery_numeric_long, aes(x = value, y = recovery_time)) +
  geom_point(alpha = 0.5) +
  facet_wrap(~predictor, scales = "free")
```



Analysis between factor predictors

```
recovery_factor =
  recovery |>
  select(where(is.factor), recovery_time)

# recovery_factor

recovery_factor_nonresp =
  recovery |>
  select(where(is.factor))

# recovery_factor_nonresp

chi_sq_matrix = matrix(NA, ncol = ncol(recovery_factor_nonresp), nrow = ncol(recovery_factor_nonresp))
for (i in 1:(ncol(recovery_factor_nonresp)-1)) {
  for (j in (i+1):ncol(recovery_factor_nonresp)) {
    cross_table = table(recovery_factor_nonresp[,i],
                        recovery_factor_nonresp[,j])
    chi_sq_matrix[i,j] = chisq.test(cross_table)$p.value
  }
}

rownames(chi_sq_matrix) = colnames(recovery_factor_nonresp)
colnames(chi_sq_matrix) = colnames(recovery_factor_nonresp)
```

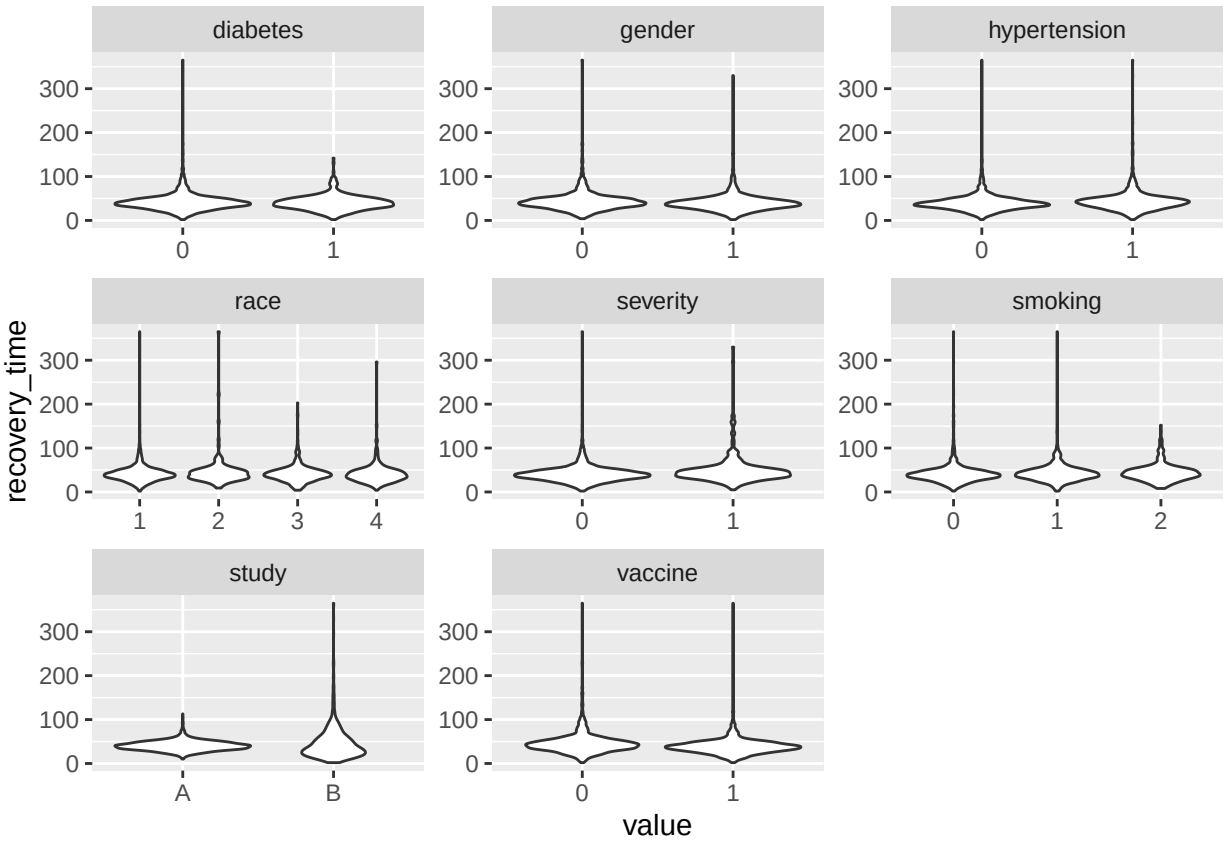
```
# chi_sq_matrix

chi_sq_matrix = t(chi_sq_matrix)

# chi_sq_matrix

pheatmap(chi_sq_matrix,
         cluster_rows = FALSE, cluster_cols = FALSE,
         show_rownames = TRUE, show_colnames = TRUE,
         legend = TRUE, display_numbers = TRUE)
```




```
recovery_factor_long =
  recovery_factor |>
  gather(key = "predictor", value = "value", -recovery_time)
```

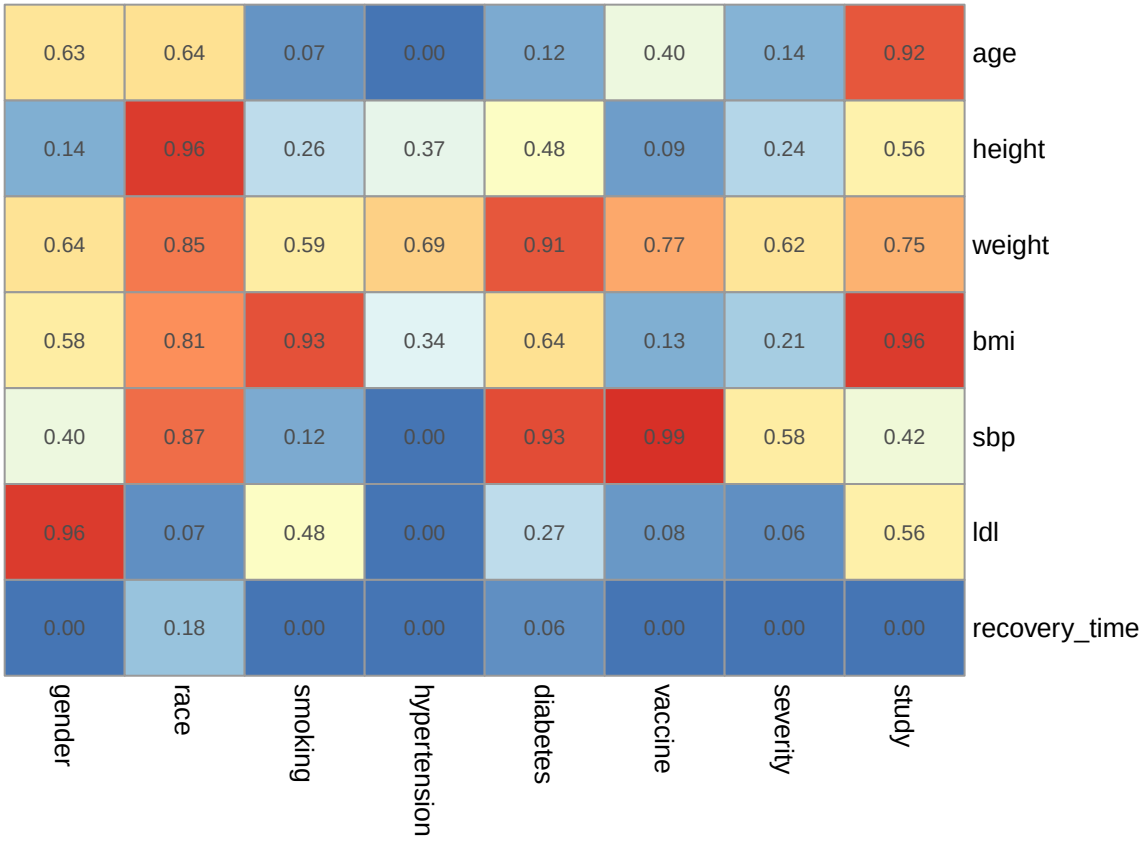
```
## Warning: attributes are not identical across measure variables; they will be
## dropped
```

```
# recovery_factor_long

ggplot(recovery_factor_long, aes(x = value, y = recovery_time)) +
  geom_violin() +
  facet_wrap(~predictor, scales = "free")
```



Analysis between numeric and factor predictors

```
anova_matrix = matrix(NA, ncol = ncol(recovery_factor_nonresp), nrow = ncol(recovery_numeric))
for (i in 1:(ncol(recovery_numeric))) {
  for (j in 1:ncol(recovery_factor_nonresp)) {
    cross_dat = data.frame(num = recovery_numeric[,i],
                           fac = recovery_factor_nonresp[,j])
    anova_matrix[i,j] = summary(aov(num ~ fac, data = cross_dat))[[1]]$"Pr(>F)"[[1]]
  }
}

# anova_matrix

rownames(anova_matrix) = colnames(recovery_numeric)
colnames(anova_matrix) = colnames(recovery_factor_nonresp)

pheatmap(anova_matrix,
         cluster_rows = FALSE, cluster_cols = FALSE,
         show_rownames = TRUE, show_colnames = TRUE,
         legend = TRUE, display_numbers = TRUE)
```



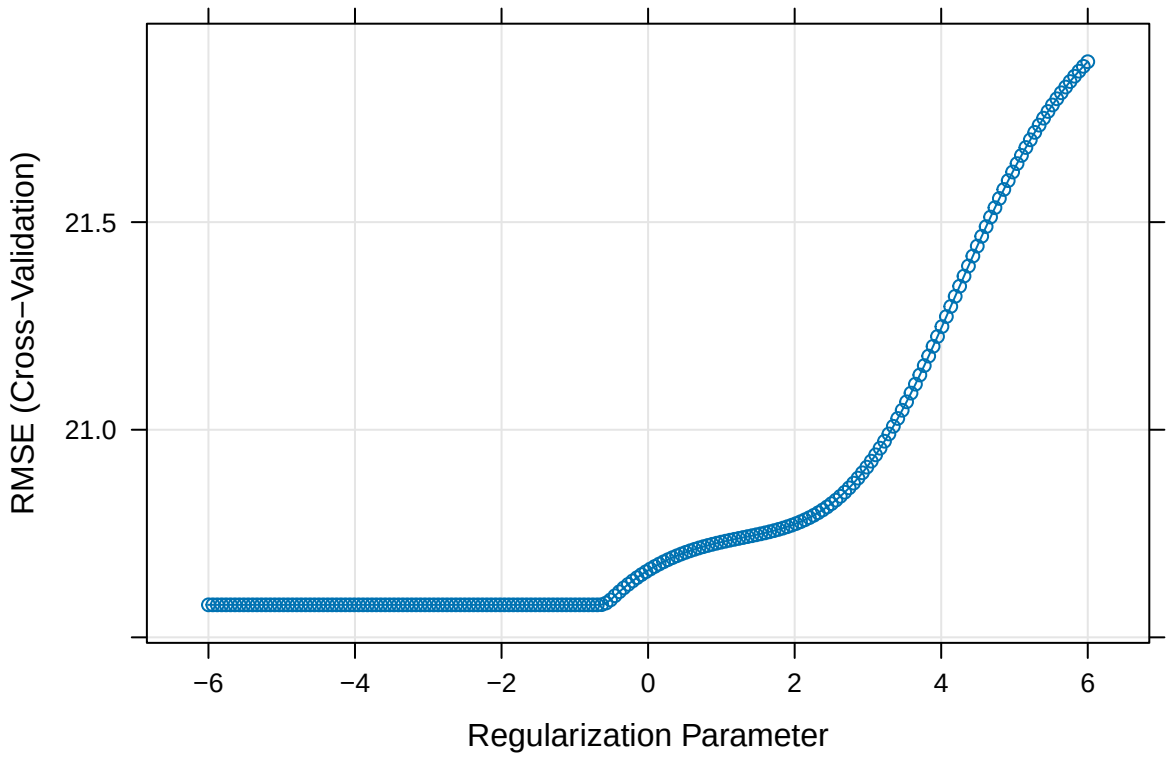
Model training

Split dataset into training and testing data.

```
set.seed(11)
data_split <- initial_split(recovery, prop = 0.8)

training_data <- training(data_split)
testing_data <- testing(data_split)
```

Ridge regression

```
ctrl1 <- trainControl(method = "cv", number = 10)

set.seed(11)
ridge.fit <- train(recovery_time ~ . ,
                   data = training_data,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 0,
                                          lambda = exp(seq(6, -6, length = 200))),
                   trControl = ctrl1)

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




```
ridge.fit$bestTune
```

```
##    alpha    lambda
## 89     0 0.4998399
```

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

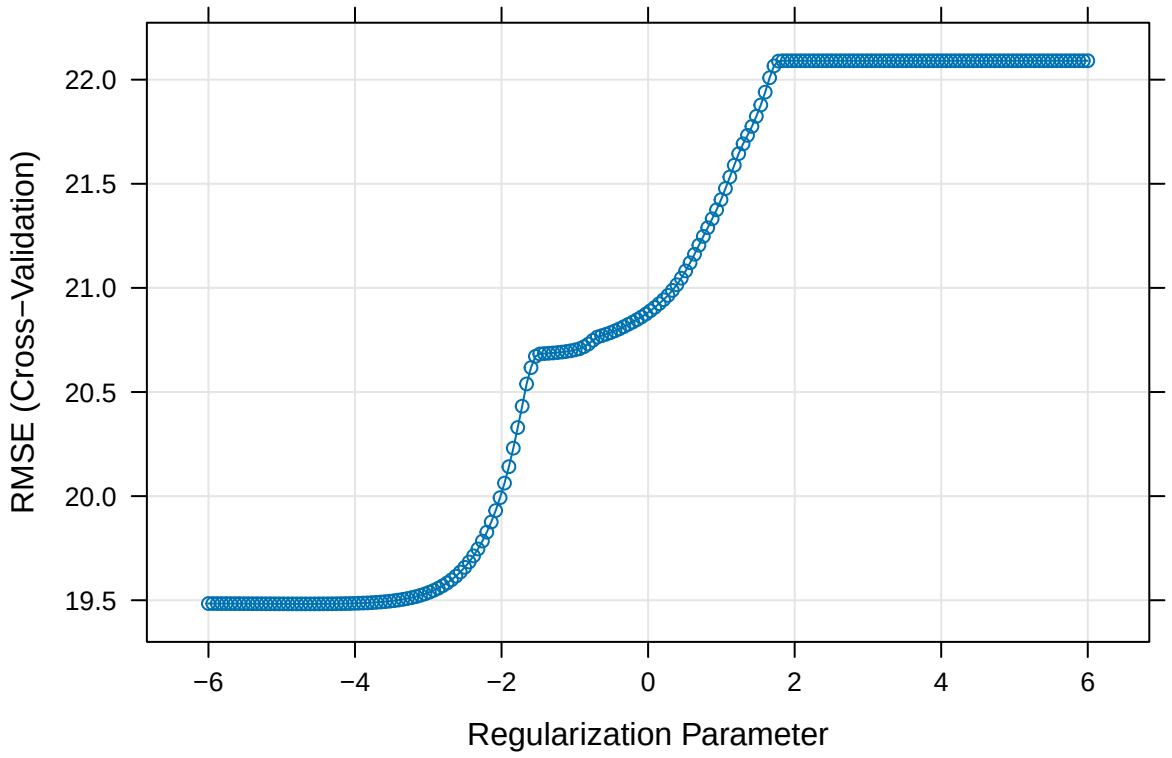
```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)   -114.65105734
## age              0.20004345
## gender1         -2.13115994
## race2            4.15477752
## race3           -0.62565171
## race4           -0.15246672
## smoking1         2.00533980
## smoking2         4.03330645
## height           0.59749832
## weight          -0.91582811
## bmi              4.20101819
## hypertension1    3.21624909
## diabetes1       -1.99034318
## sbp              0.01764196
## ldl             -0.02006663
## vaccine1        -6.71209585
## severity1        8.77312581
## studyB           4.81721562
```

Lasso

```
set.seed(11)
lasso.fit <- train(recovery_time ~ .,
                   data = training_data,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(6, -6, length = 200))),
                   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)
```




```
lasso.fit$bestTune
```

```
##    alpha      lambda
## 23     1 0.009340768
```

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

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)   -1.954480e+03
## age            1.937815e-01
```

```
## gender1       -2.292988e+00
## race2          4.102802e+00
## race3         -5.884611e-01
## race4          4.793890e-01
## smoking1       2.265972e+00
## smoking2       4.272929e+00
## height         1.145672e+01
## weight        -1.242565e+01
## bmi            3.731646e+01
## hypertension1  3.484604e+00
## diabetes1     -1.779152e+00
## sbp           -1.105410e-02
## ldl           -2.554712e-02
## vaccine1      -6.313273e+00
## severity1      9.155499e+00
## studyB         4.620366e+00
```

Elastic net

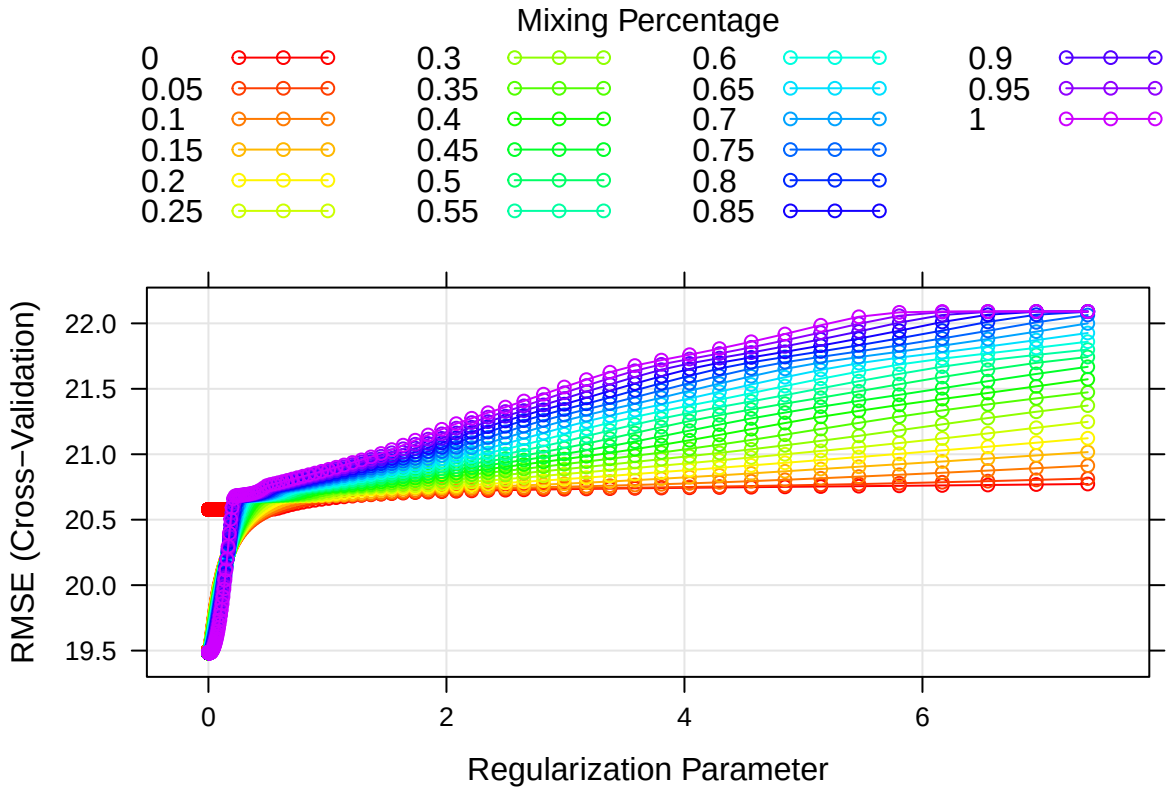
```
set.seed(11)
enet.fit <- train(recovery_time ~ .,
                  data = training_data,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                         lambda = exp(seq(2, -10, length = 200))),
                  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
## 868   0.2 0.002580431
```

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

plot(enet.fit, par.settings = myPar)
```

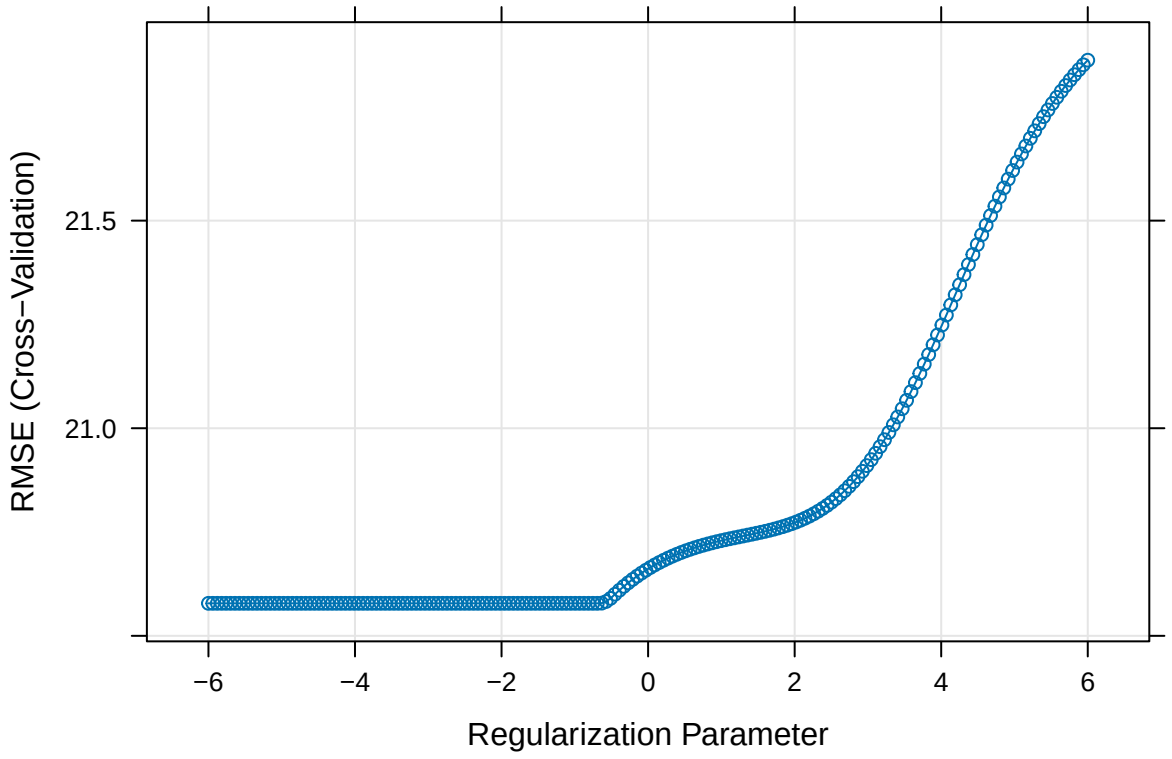



```
coef(enet.fit$finalModel, enet.fit$bestTune$lambda)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                            s1
## (Intercept)   -1.952199e+03
## age            1.972764e-01
## gender1       -2.306583e+00
## race2          4.139347e+00
## race3         -6.070652e-01
## race4          5.040683e-01
## smoking1       2.287723e+00
## smoking2       4.307013e+00
## height         1.144672e+01
## weight        -1.241459e+01
## bmi            3.728410e+01
## hypertension1  3.568559e+00
## diabetes1     -1.802445e+00
## sbp           -1.688603e-02
## ldl           -2.603621e-02
## vaccine1      -6.333775e+00
## severity1      9.177936e+00
## studyB         4.637170e+00
```

Ridge regression: one SE rule

```r
ctrl_1se <- trainControl(method = "cv", number = 10, selectionFunction = "oneSE")

set.seed(11)
ridge.fit_1se <- train(recovery_time ~ . ,
                       data = training_data,
                       method = "glmnet",
                       tuneGrid = expand.grid(alpha = 0,
                                              lambda = exp(seq(6, -6, length = 200))),
                       trControl = ctrl_1se)

plot(ridge.fit_1se, xTrans = log)
```




```r
ridge.fit_1se$bestTune
```

```
##     alpha   lambda
## 196     0 316.9658
```

```r
coef(ridge.fit_1se$finalModel, ridge.fit_1se$bestTune$lambda)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)   40.9035051207
## age            0.0225089928
## gender1       -0.1724040557
## race2          0.3772525684
## race3         -0.0318586237
## race4         -0.0098145100
```

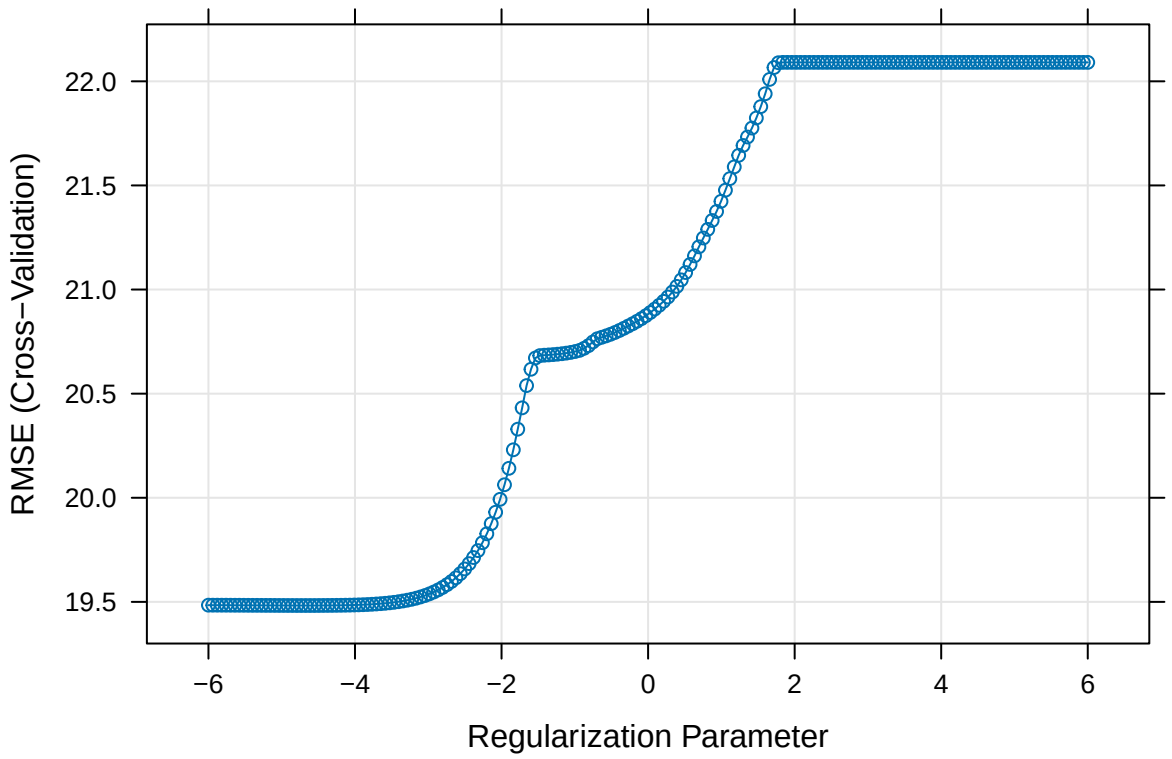
```
## smoking1       0.1309686338
## smoking2       0.2362534598
## height        -0.0411558082
## weight         0.0227943684
## bmi            0.1281912930
## hypertension1  0.2339598484
## diabetes1     -0.1635369673
## sbp            0.0118642244
## ldl            0.0006658349
## vaccine1      -0.4862538300
## severity1      0.6552374862
## studyB         0.3269409146
```

Lasso: one SE rule

```
set.seed(11)
lasso.fit_1se <- train(recovery_time ~ .,
                       data = training_data,
                       method = "glmnet",
                       tuneGrid = expand.grid(alpha = 1,
                                              lambda = exp(seq(6, -6, length = 200))),
                       trControl = ctrl_1se)
```

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

```
plot(lasso.fit_1se, xTrans = log)
```



```
lasso.fit_1se$bestTune
```

```
##    alpha    lambda
## 72     1 0.1793183
```

```
coef(lasso.fit_1se$finalModel, lasso.fit_1se$bestTune$lambda)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)   -2.643798e+02
## age            1.697354e-01
## gender1       -1.865753e+00
## race2          3.531337e+00
## race3         -1.916278e-01
## race4          .
## smoking1       1.619531e+00
## smoking2       3.457458e+00
## height         1.491542e+00
## weight        -1.876756e+00
## bmi            6.987201e+00
## hypertension1  3.152043e+00
## diabetes1     -1.521273e+00
## sbp            .
## ldl           -7.937861e-03
## vaccine1      -6.449607e+00
## severity1      8.500207e+00
## studyB         4.537838e+00
```

Elastic net: one SE rule

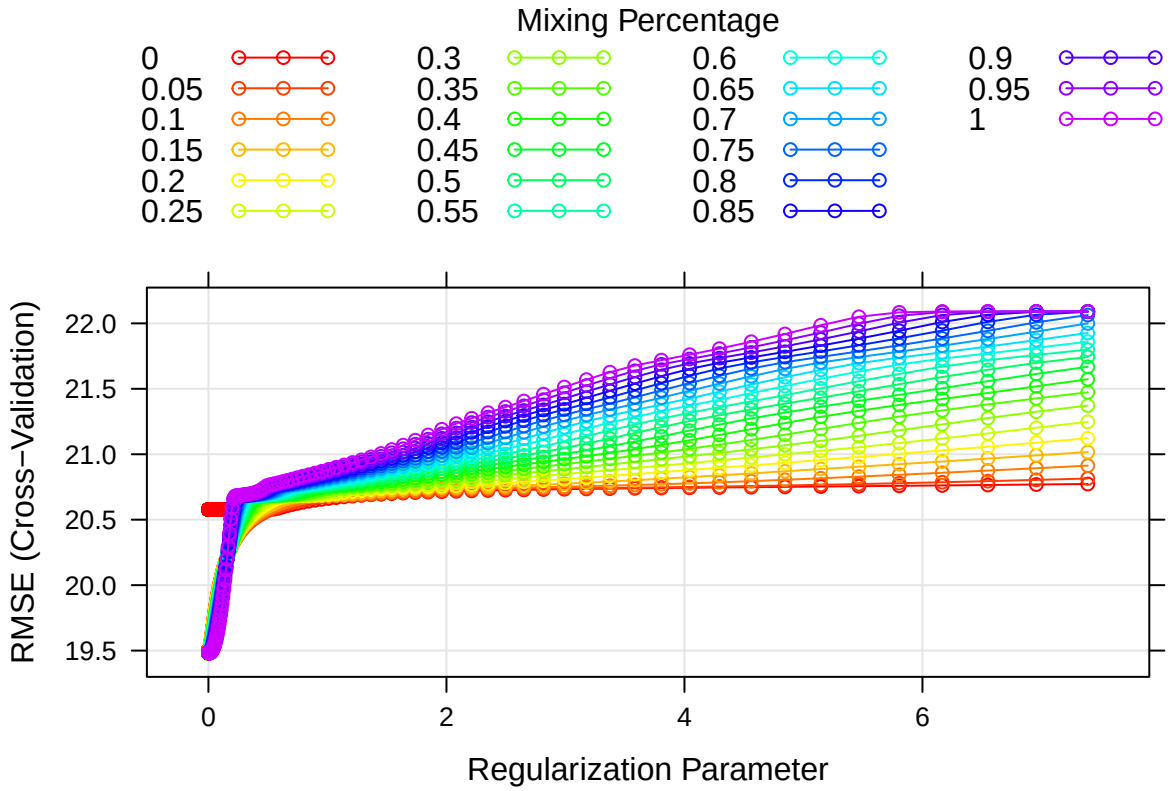
```
set.seed(11)
enet.fit_1se <- train(recovery_time ~ .,
                      data = training_data,
                      method = "glmnet",
                      tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                             lambda = exp(seq(2, -10, length = 200))),
                      trControl = ctrl_1se)
```

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

```
enet.fit_1se$bestTune
```

```
##     alpha    lambda
## 348  0.05 0.3212041
```

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

plot(enet.fit_1se, par.settings = myPar)
```

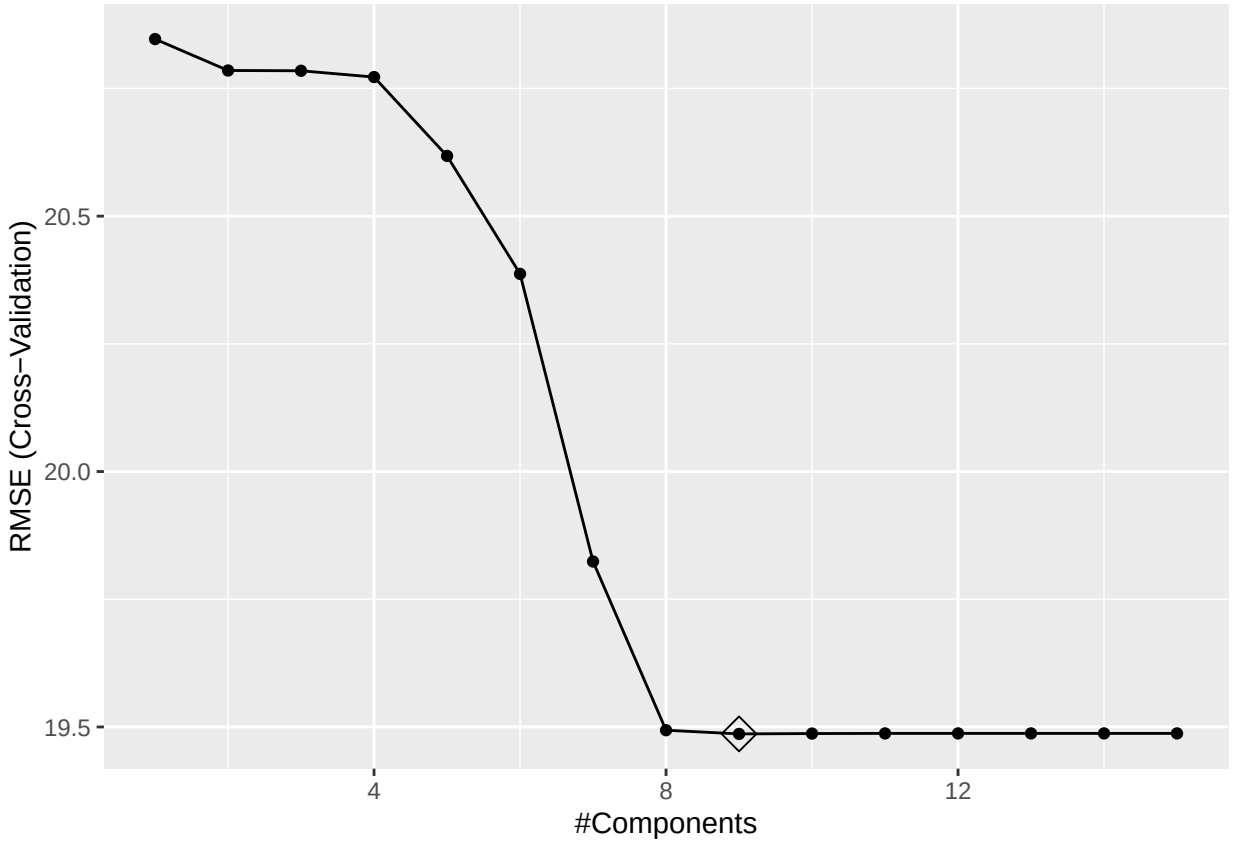



```
coef(enet.fit_1se$finalModel, enet.fit_1se$bestTune$lambda)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                            s1
## (Intercept)   -218.83869465
## age              0.19893628
## gender1         -2.12900018
## race2            4.12873024
## race3           -0.58678595
## race4           -0.05078926
## smoking1         2.00250853
## smoking2         4.03366016
## height           1.21461536
## weight          -1.57275172
## bmi              6.09912376
## hypertension1    3.29511924
## diabetes1       -1.95328996
## sbp              0.01084008
## ldl             -0.01952597
## vaccine1        -6.72347455
## severity1        8.83174186
## studyB           4.82138977
```

PLS

```
# training data
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(11)
pls.fit <- train(x, y,
                 method = "pls",
                 tuneGrid = data.frame(ncomp = 1:15),
                 trControl = ctrl1,
                 preProcess = c("center", "scale"))

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




```
pls.fit$finalModel$ncomp
```

```
## [1] 9
```

MARS

```
mars_grid <- expand.grid(degree = 1:3,
                         nprune = 2:15)

set.seed(11)
mars.fit <- train(x, y,
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl1)
```

```
## Loading required package: earth
```

```
## Loading required package: Formula
```

```
## Loading required package: plotmo
```

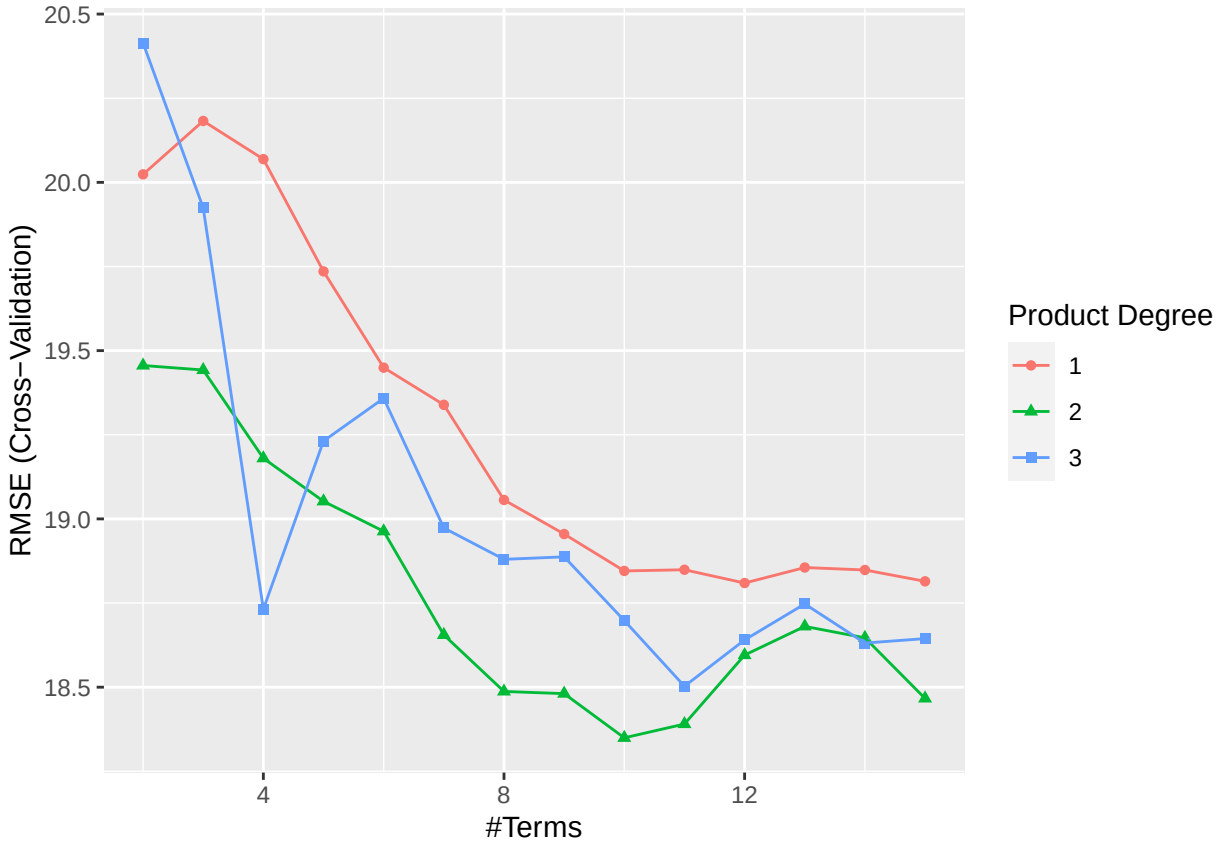
```
## Loading required package: plotrix
```

```
##
## Attaching package: 'plotrix'
```

```
## The following object is masked from 'package:scales':
##
##     rescale
```

```
## Loading required package: TeachingDemos
```

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



```
mars.fit$bestTune
```

```
##    nprune degree
## 23     10      2
```

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

```
##                  (Intercept)                    h(31-bmi)
##                   14.2893606                    4.1770989
##           h(bmi-31) * studyB                  h(bmi-25.3)
##                   15.2507664                    4.9496560
##                     vaccine1     h(bmi-25.3) * severity1
##                   -6.2289789                    2.1690932
## h(bmi-22.3) * hypertension1 h(22.3-bmi) * hypertension1
##                    0.6581062                   14.7719808
##         race2 * h(bmi-33.9)     h(bmi-33.9) * severity1
##                   77.8665195                   70.9836429
```

GAM

```
set.seed(1)
gam.fit <- train(x, y,
                 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 1.9-0. For overview type 'help("mgcv-package")'.
```

```
gam.fit$bestTune
```

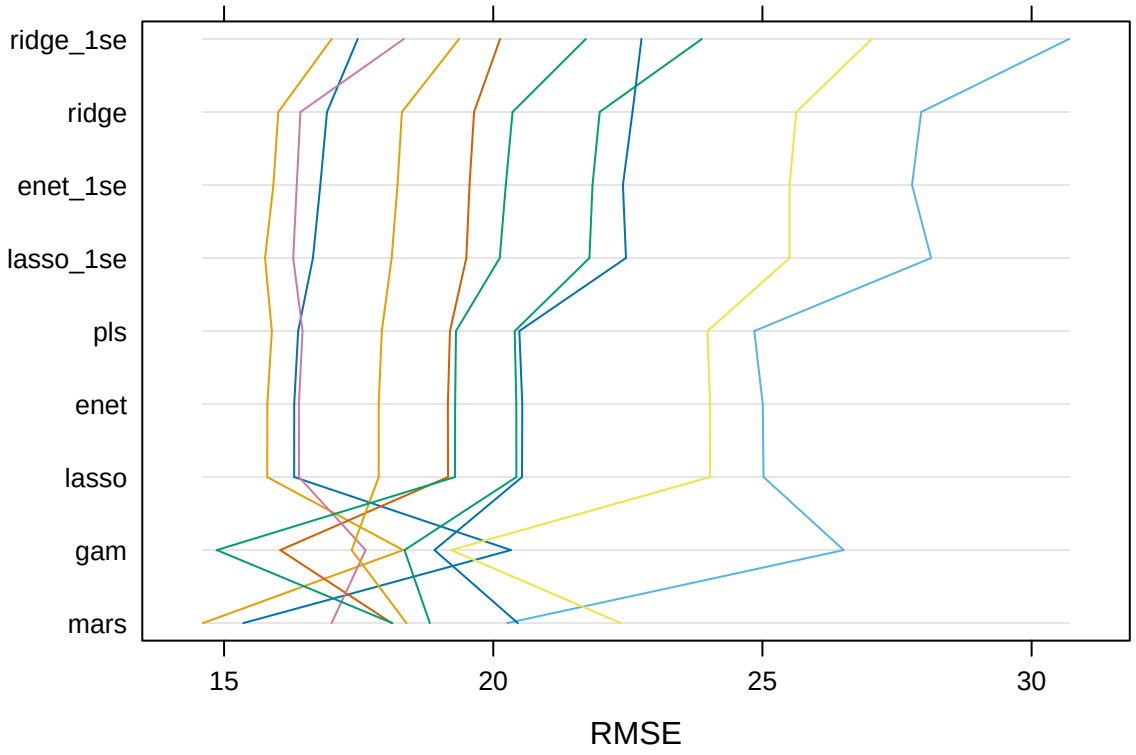
```
##   select method
## 1  FALSE GCV.Cp
```

Comparing different models

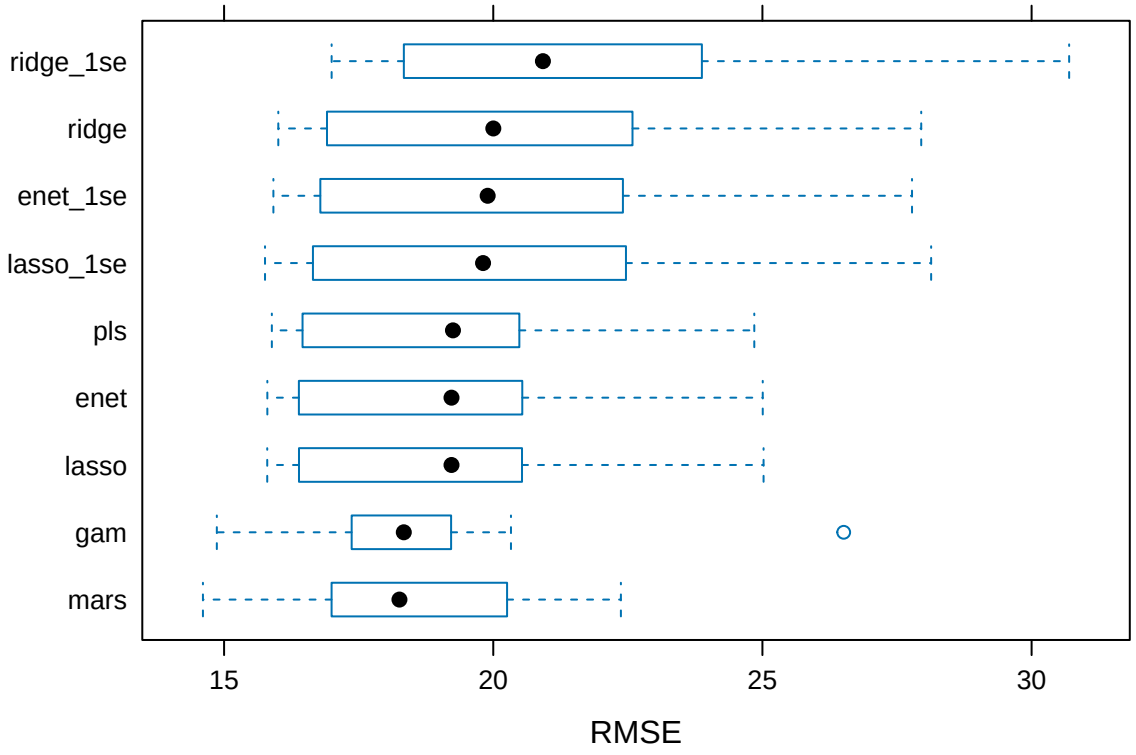
```r
set.seed(11)

resamp <- resamples(list(ridge = ridge.fit, enet = enet.fit, lasso = lasso.fit, pls = pls.fit,
                         ridge_1se = ridge.fit_1se, enet_1se = enet.fit_1se, lasso_1se = lasso.fit_1se,
                         mars = mars.fit, gam = gam.fit))

summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: ridge, enet, lasso, pls, ridge_1se, enet_1se, lasso_1se, mars, gam
## Number of resamples: 10
##
## MAE
##               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## ridge     11.90469 12.54695 13.00443 13.04325 13.58196 14.27625    0
## enet      11.96564 12.39041 13.26113 13.02431 13.52946 13.91211    0
## lasso     11.96549 12.38961 13.25917 13.02185 13.53109 13.90959    0
## pls       12.03096 12.47340 13.31264 13.09057 13.60297 13.95535    0
## ridge_1se 12.13485 13.05752 13.27856 13.41577 13.88641 14.58846    0
## enet_1se  11.88284 12.46541 12.96441 12.98752 13.52810 14.22101    0
## lasso_1se 11.79052 12.33593 12.88245 12.92246 13.58637 14.20098    0
## mars      10.91572 11.71431 12.24786 12.07252 12.51720 12.89125    0
## gam       11.22297 12.00089 12.55518 12.51730 13.00707 13.63910    0
##
## RMSE
##               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## ridge     16.00722 17.25925 20.00110 20.57831 22.43405 27.94991    0
## enet      15.80354 16.76022 19.22445 19.48193 20.51149 25.00597    0
## lasso     15.80145 16.76142 19.22422 19.48264 20.50791 25.02117    0
## pls       15.88697 16.82547 19.25178 19.48638 20.46293 24.84868    0
## ridge_1se 16.99777 18.59551 20.92268 21.83867 23.59467 30.69770    0
## enet_1se  15.91600 17.14625 19.89601 20.46035 22.26728 27.77887    0
## lasso_1se 15.76046 17.01619 19.81021 20.43188 22.29529 28.13379    0
## mars      14.60691 17.27601 18.25745 18.34948 19.89803 22.37118    0
## gam       14.86521 17.43440 18.33833 18.75470 19.13950 26.50970    0
##
## Rsquared
##                Min.    1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## ridge     0.05279118 0.10526923 0.1392760 0.1464777 0.1743017 0.2716057    0
## enet      0.12616732 0.18960775 0.2147002 0.2331675 0.2402409 0.4420076    0
## lasso     0.12600091 0.18978794 0.2149894 0.2332728 0.2401815 0.4421833    0
## pls       0.12596192 0.18925914 0.2172799 0.2334081 0.2384395 0.4438751    0
## ridge_1se 0.03767866 0.08892062 0.1202529 0.1252861 0.1585655 0.2179548    0
## enet_1se  0.06250011 0.11564685 0.1479151 0.1563262 0.1853543 0.2895023    0
## lasso_1se 0.05808079 0.12335619 0.1569717 0.1588148 0.1946356 0.2574671    0
## mars      0.19275996 0.22166420 0.2753658 0.3132764 0.3459672 0.6407233    0
## gam       0.15303888 0.19941341 0.2226602 0.2861746 0.3903267 0.4986527    0
```

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




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



Results

```
pred_ridge = predict(ridge.fit, newdata = testing_data)
rmse_ridge = sqrt(mean((testing_data$recovery_time - pred_ridge)^2))
rmse_ridge
```

```
## [1] 24.0026
```

```
pred_lasso = predict(lasso.fit, newdata = testing_data)
rmse_lasso = sqrt(mean((testing_data$recovery_time - pred_lasso)^2))
rmse_lasso
```

```
## [1] 22.54878
```

```
pred_enet = predict(enet.fit, newdata = testing_data)
rmse_enet = sqrt(mean((testing_data$recovery_time - pred_enet)^2))
rmse_enet
```

```
## [1] 22.55381
```

```
pred_ridge_1se = predict(ridge.fit_1se, newdata = testing_data)
rmse_ridge_1se = sqrt(mean((testing_data$recovery_time - pred_ridge_1se)^2))
rmse_ridge_1se
```

```
## [1] 25.40867
```

```
pred_lasso_1se = predict(lasso.fit_1se, newdata = testing_data)
rmse_lasso_1se = sqrt(mean((testing_data$recovery_time - pred_lasso_1se)^2))
rmse_lasso_1se
```

```
## [1] 23.80956
```

```
pred_enet_1se = predict(enet.fit_1se, newdata = testing_data)
rmse_enet_1se = sqrt(mean((testing_data$recovery_time - pred_enet_1se)^2))
rmse_enet_1se
```

```
## [1] 23.88015
```

```
pred_pls = predict(pls.fit, newdata = x2)
rmse_pls = sqrt(mean((y2 - pred_pls)^2))
rmse_pls
```

```
## [1] 22.52326
```

```
pred_mars = predict(mars.fit, new_data = x2)
rmse_mars = sqrt(mean((y2 - pred_mars)^2))
rmse_mars
```

```
## [1] 29.3692
```

```r
pred_gam = predict(gam.fit, new_data = x2)
rmse_gam = sqrt(mean((y2 - pred_gam)^2))
rmse_gam
```

```
## [1] 28.80383
```

```r
models = c("ridge_regression", "lasso", "elastic_net", "ridge_regression_1se", "lasso_1se",
           "elastic_net_1se", "pls", "mars", "gam")
rmse_on_testing = c(rmse_ridge, rmse_lasso, rmse_enet, rmse_ridge_1se, rmse_lasso_1se,
                    rmse_enet_1se, rmse_pls, rmse_mars, rmse_gam)
knitr::kable(data.frame(models, rmse_on_testing))
```

models	rmse_on_testing
ridge_regression	24.00260
lasso	22.54878
elastic_net	22.55381
ridge_regression_1se	25.40867
lasso_1se	23.80956
elastic_net_1se	23.88015
pls	22.52326
mars	29.36920
gam	28.80383