P8106 Midterm

Brian Jo Hsuan Lee

3/15/2022

Load packages

```
library(tidyverse)
library(corrplot)
library(caret)
library(splines)
library(mgcv)
library(earth)
```

Clean data. Consolidate the historic team names into their corresponding current names; replace NA values in weather temperature, humidity and wind with each of their means; expand the weather detail column into boolean `dome`, `rain`, `fog`, and `snow` columns; update the spread values to those for the home team; rid seasons before 1979 due to incomplete records; rid `schedule playoff` due to collinearity with the more informative `schedule week`; rid `stadium` due to speculated insignificant contribution; rid `spread_favored` and `weather details` as they are replaced by new predictors

```
data = read_csv("spreadspoke_score.csv",
                    col_types = "iffffiiffddffiiic",
                    col_select = c("schedule_season":"weather_detail")) %>%
  filter(
    schedule_season %in% (1979:2021)
  ) %>%
  mutate(
    schedule_season = droplevels(schedule_season),
    stadium = droplevels(stadium),
    dif = score_away - score_home,
    weather_detail = replace(weather_detail, is.na(weather_detail), "Dry"),
    weather_detail = factor(weather_detail),
    weather_temperature = replace(weather_temperature, is.na(weather_temperature), round(mean(weather_t
    weather_wind_mph = replace(weather_wind_mph, is.na(weather_wind_mph), round(mean(weather_wind_mph,
    weather_humidity = replace(weather_humidity, is.na(weather_humidity),round(mean(weather_humidity, na
    schedule_week = fct_collapse(schedule_week,
                                 "SuperBowl" = c("Superbowl","SuperBowl"),
                                 "WildCard" = c("Wildcard","WildCard")),
    schedule_week = fct_relevel(schedule_week, c(1:18, "WildCard", "Division", "Conference", "SuperBowl
    team_home = fct_collapse(team_home,
                             "Tennessee Titans" = c("Tennessee Titans", "Tennessee Oilers", "Houston Oi
                             "Washington Football Team" = c("Washington Football Team", "Washington Red
                             "Las Vegas Raiders" = c("Oakland Raiders", "Los Angeles Raiders", "Las Veg
                             "Indianapolis Colts" = c("Baltimore Colts", "Indianapolis Colts"),
                             "Los Angeles Chargers" = c("Los Angeles Chargers", "San Diego Chargers"),
                             "Arizona Cardinals" = c("St. Louis Cardinals", "Phoenix Cardinals", "Arizo
                             "Los Angeles Rams" = c("Los Angeles Rams", "St. Louis Rams"),
```

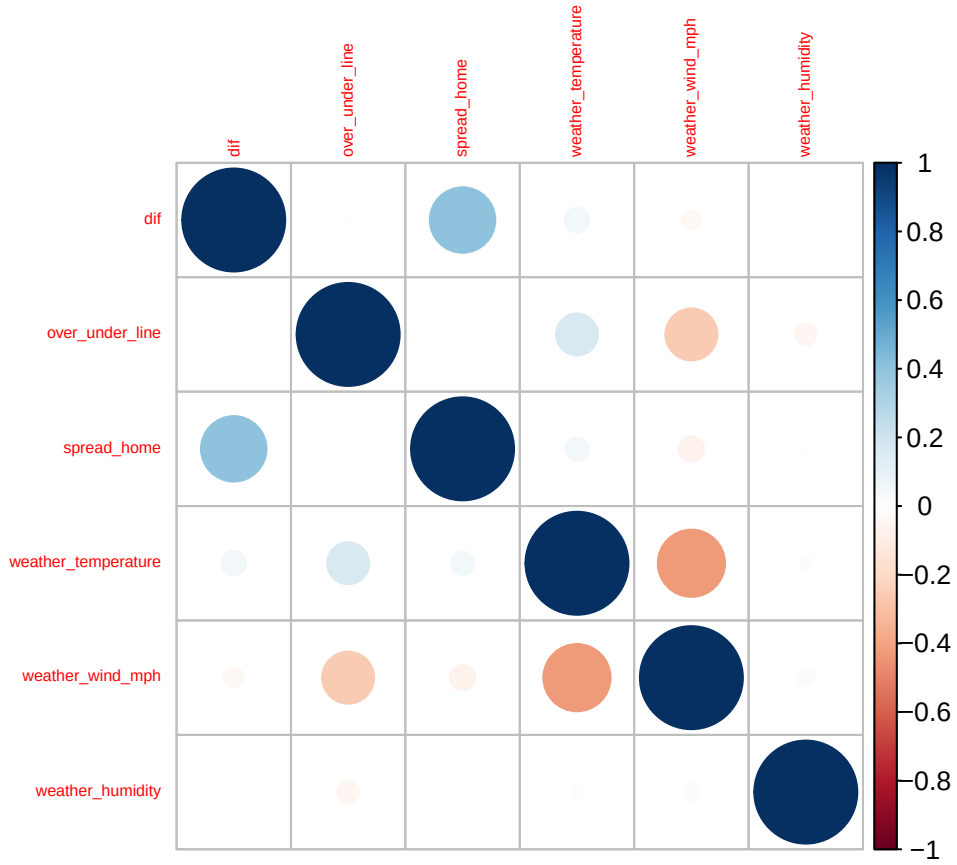
```
                            "New England Patriots" = c("New England Patriots", "Boston Patriots")),
  team_away = fct_collapse(team_away,
                            "Tennessee Titans" = c("Tennessee Titans", "Tennessee Oilers", "Houston Oi
                            "Washington Football Team" = c("Washington Football Team", "Washington Red
                            "Las Vegas Raiders" = c("Oakland Raiders", "Los Angeles Raiders", "Las Veg
                            "Indianapolis Colts" = c("Baltimore Colts", "Indianapolis Colts"),
                            "Los Angeles Chargers" = c("Los Angeles Chargers", "San Diego Chargers"),
                            "Arizona Cardinals" = c("St. Louis Cardinals", "Phoenix Cardinals", "Arizo
                            "Los Angeles Rams" = c("Los Angeles Rams", "St. Louis Rams"),
                            "New England Patriots" = c("New England Patriots", "Boston Patriots")),
  team_away = fct_relevel(team_away, levels(team_home)),
  team_favorite_id = recode_factor(team_favorite_id,
                                   "MIA" = "Miami Dolphins",
                                   "TEN" = "Tennessee Titans",
                                   "LAC" = "Los Angeles Chargers",
                                   "GB" = "Green Bay Packers",
                                   "ATL" = "Atlanta Falcons",
                                   "BUF" = "Buffalo Bills",
                                   "DET" = "Detroit Lions",
                                   "PIT" = "Pittsburgh Steelers",
                                   "SF" = "San Francisco 49ers",
                                   "ARI" = "Arizona Cardinals",
                                   "WAS" = "Washington Football Team",
                                   "LAR" = "Los Angeles Rams",
                                   "CLE" = "Cleveland Browns",
                                   "DAL" = "Dallas Cowboys",
                                   "DEN" = "Denver Broncos",
                                   "MIN" = "Minnesota Vikings",
                                   "NYJ" = "New York Jets",
                                   "LVR" = "Las Vegas Raiders",
                                   "PHI" = "Philadelphia Eagles",
                                   "IND" = "Indianapolis Colts",
                                   "NE" = "New England Patriots",
                                   "KC" = "Kansas City Chiefs",
                                   "NYG" = "New York Giants",
                                   "CHI" = "Chicago Bears",
                                   "NO"= "New Orleans Saints",
                                   "CIN" = "Cincinnati Bengals",
                                   "SEA" = "Seattle Seahawks",
                                   "TB" = "Tampa Bay Buccaneers",
                                   "JAX" = "Jacksonville Jaguars",
                                   "CAR" = "Carolina Panthers",
                                   "BAL" = "Baltimore Ravens",
                                   "HOU" = "Houston Texans",
                                   .default = "None"),
  spread_home = ifelse(as.character(team_away) == as.character(team_favorite_id), abs(spread_favorite
  dome = ifelse(((as.character(weather_detail) == "DOME") | (as.character(weather_detail) == "DOME (O
  fog = ifelse((as.character(weather_detail) == "Fog") | (as.character(weather_detail) == "Rain | Fog
  rain = ifelse((as.character(weather_detail) == "Rain") | (as.character(weather_detail) == "Rain | F
  snow = ifelse((as.character(weather_detail) == "Snow") | (as.character(weather_detail) == "Snow | F
) %>%
select(-schedule_season, -schedule_playoff, -score_home, -score_away, -team_favorite_id, -spread_favo
drop_na()
```

Partition data into training and testing sets, and define resampling method.

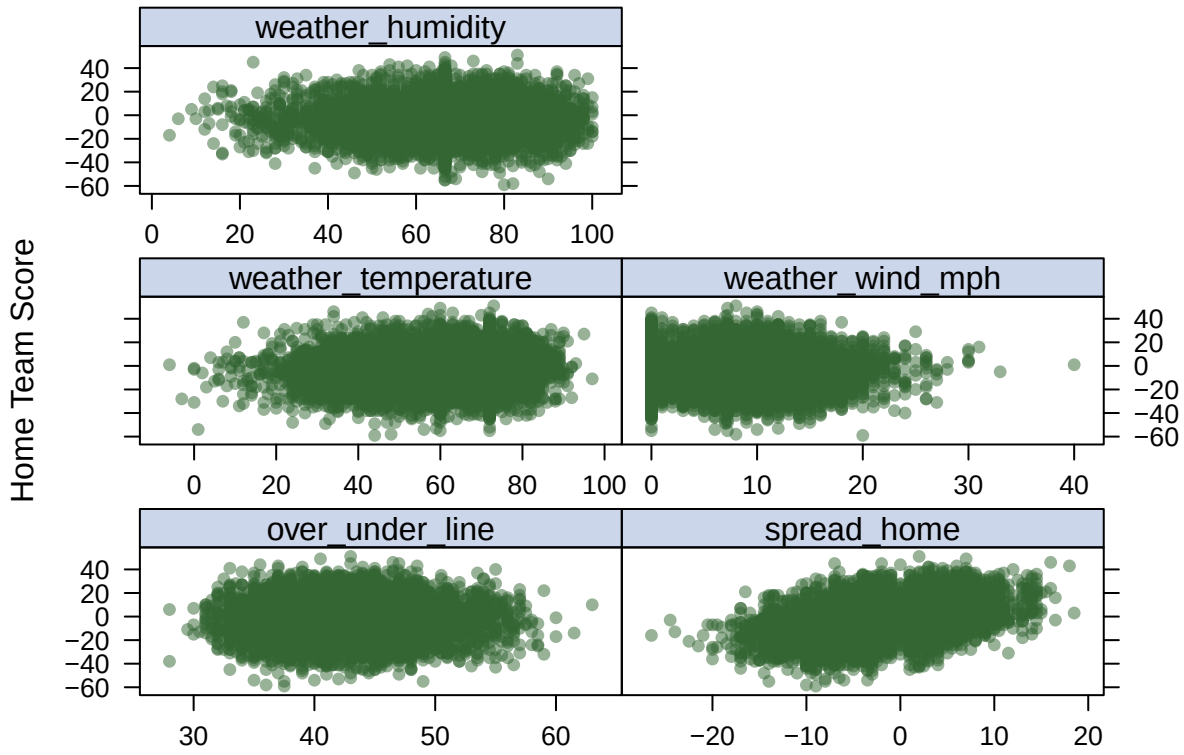
```
set.seed(2022)

# partition data into training and testing sets into randomized 4:1 splits
train_index = createDataPartition(y = data$dif, p = 0.8, list = FALSE)
train_data = data[train_index, ]
train_cont_data =
  train_data %>%
  select(dif, over_under_line, spread_home, weather_temperature, weather_wind_mph, weather_humidity)
test_data = data[-train_index, ]

# matrices of predictors
train_pred = model.matrix(dif ~ ., train_data)[ ,-1]
train_cont_pred = model.matrix(dif ~ ., train_cont_data)[ ,-1]
test_pred = model.matrix(dif ~ ., test_data)[ ,-1]

# vectors of response
train_resp = train_data$dif
test_resp = test_data$dif

# use 5 repeats of 10-Fold CV as the resampling method
ctrl = trainControl(method = "repeatedcv", repeats = 2, number = 5)
```

Visualize of the correlation among continuous variables, and the linearity among continuous predictors against the response.

```
corrplot(cor(train_cont_data), method = "circle", type="full", tl.cex = 0.5)
```

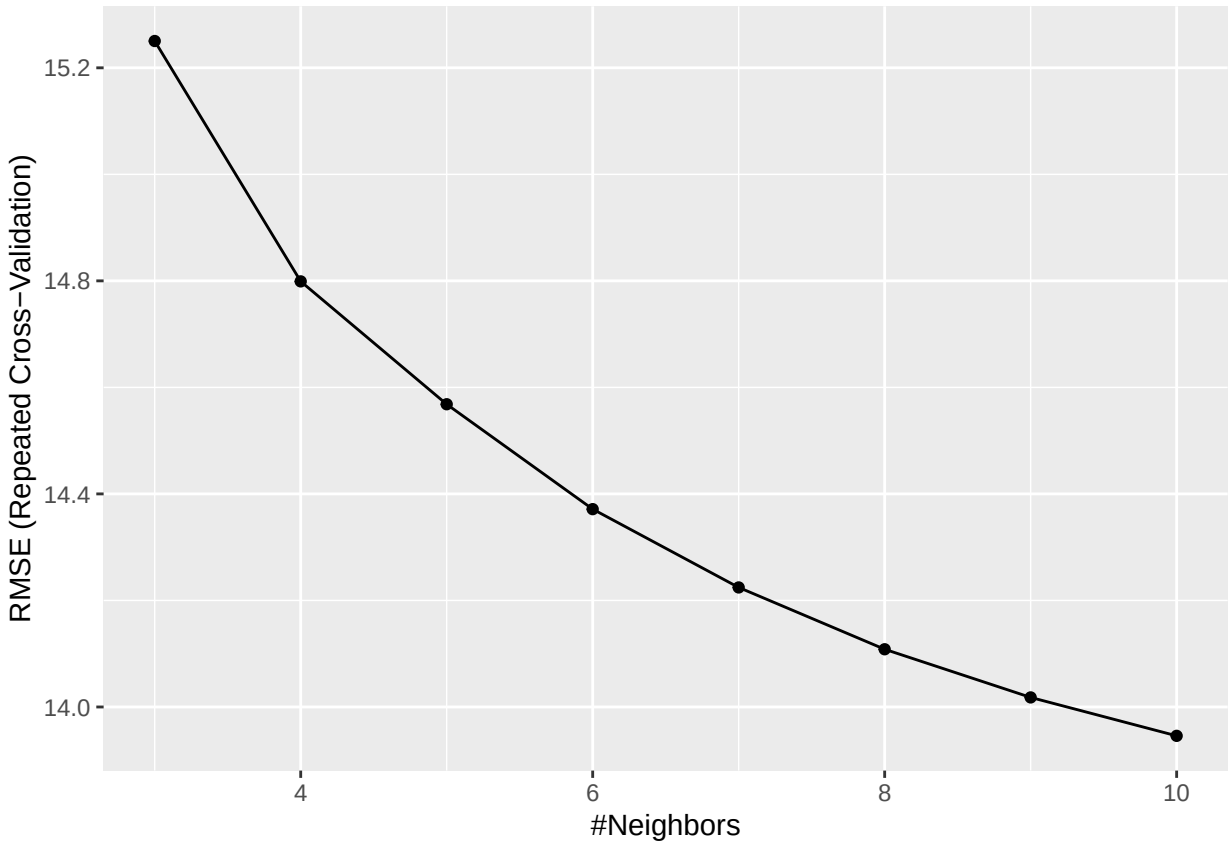



```
data_theme1 = trellis.par.get()
data_theme1$plot.symbol$col = rgb(.2, .4, .2, .5)
data_theme1$plot.symbol$pch = 16
data_theme1$plot.line$col = rgb(.8, .1, .1, 1)
data_theme1$plot.line$lwd = 2
data_theme1$strip.background$col = rgb(.0, .2, .6, .2)
trellis.par.set(data_theme1)

featurePlot(train_cont_pred, train_resp,
            plot = "scatter", labels = c("", "Home Team Score"),
            type = c("p"), layout = c(2, 3))
```



Fit a k-nearest neighbor model. Might have to rely on the Cluster.

```
set.seed(2022)
k_grid = expand.grid(k = seq(from = 3, to = 10, by = 1))

knn_fit = train(train_pred, train_resp,
                method = "knn",
                trControl = ctrl,
                tuneGrid = k_grid)
ggplot(knn_fit)
```

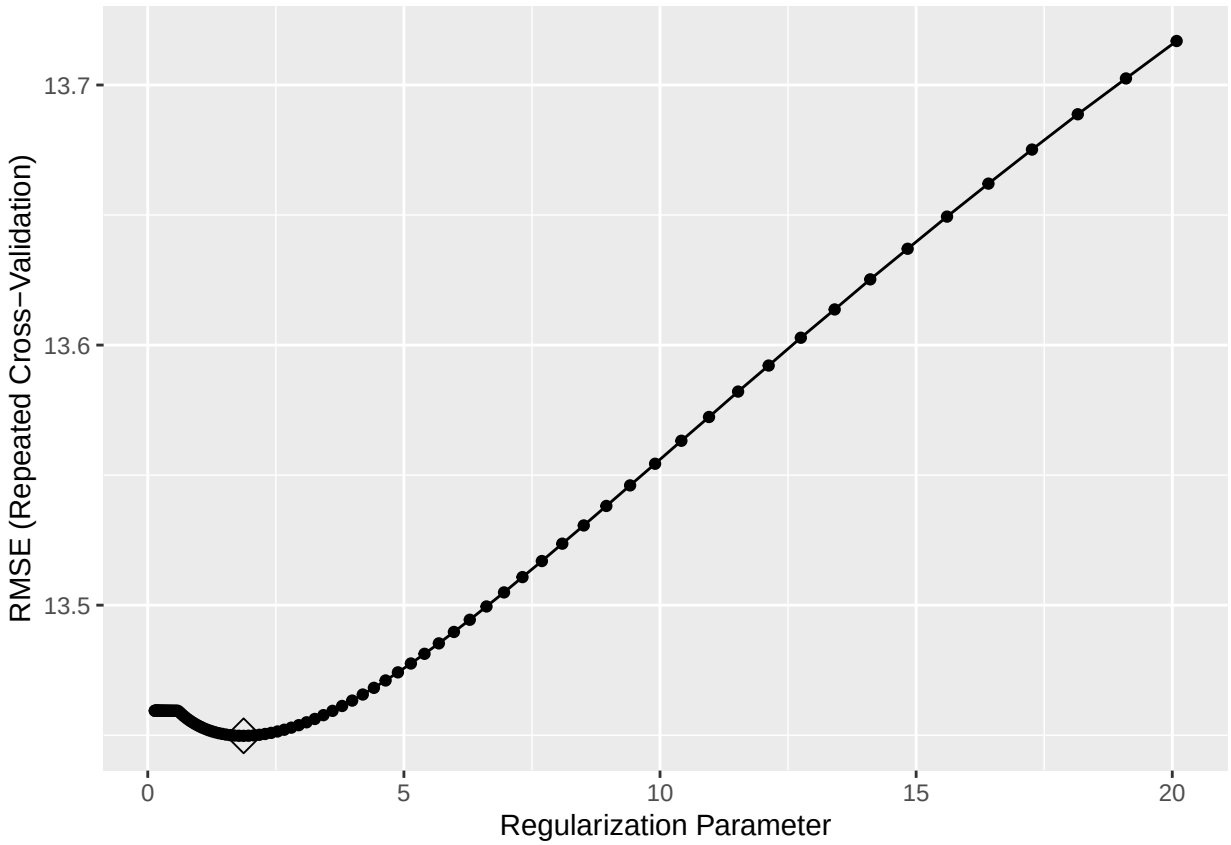



```
knn_fit$bestTune[1,1]
```

```
## [1] 10
```

Fit linear models

```
set.seed(2022)

# Fit a multiple linear model
lm_fit = train(train_pred, train_resp,
               method = "lm",
               trControl = ctrl)

# Make prediction on test data
lm_pred = predict(lm_fit, newdata = test_pred)

# Fit a ridge model (L2 regularization, alpha = 0)
ridge_fit = train(train_pred, train_resp,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = 0,
                                         lambda = exp(seq(-2, 3, length=100))),
                  preProc = c("center", "scale"),
                  trControl = ctrl)
ggplot(ridge_fit, highlight = TRUE) # could xTrans = log from plot() be implemented somehow?
```




```
# Get the optimal penalty term lambda
ridge_fit$bestTune
```

```
##    alpha   lambda
## 53     0 1.870606
```

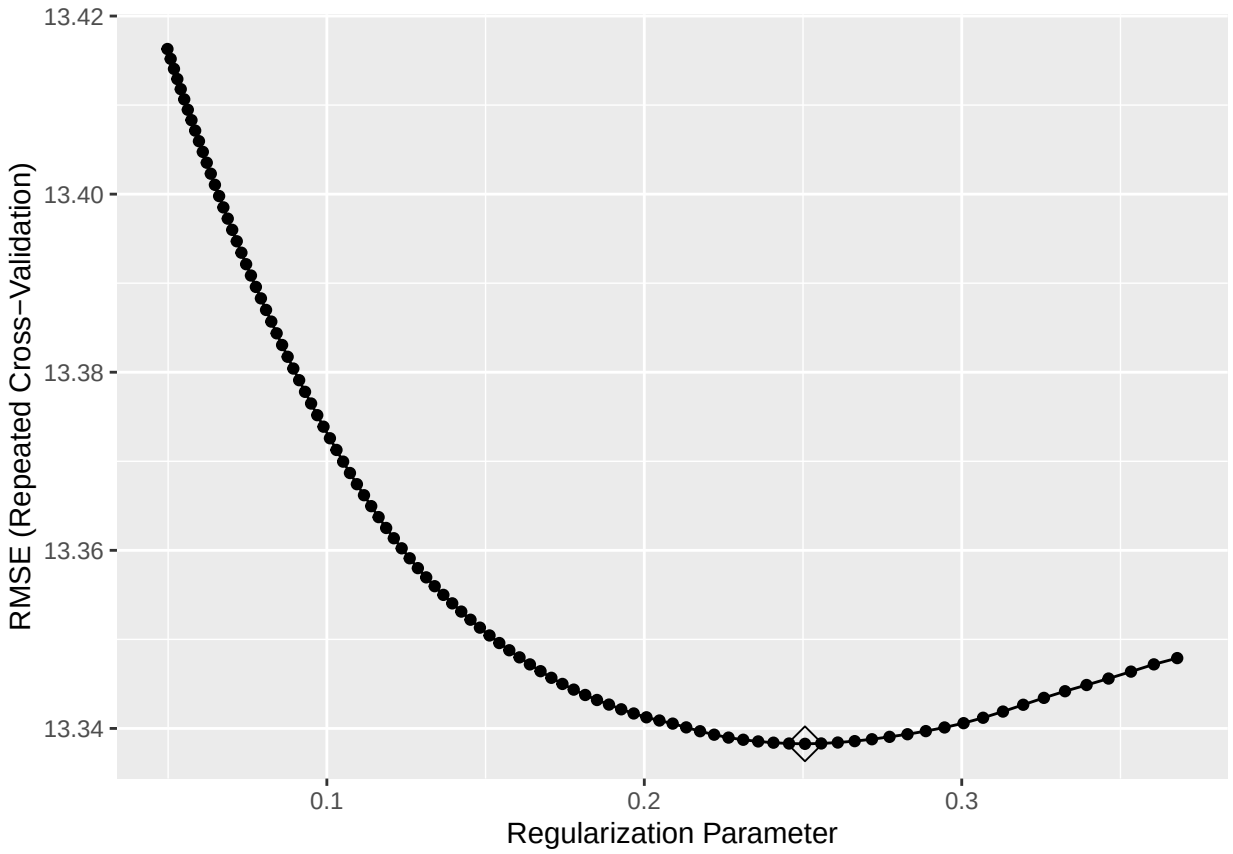
```
# Make prediction on test data and evaluate model using test MSE
ridge_pred = predict(ridge_fit, newdata = test_pred)
ridge_tmse = mean((test_resp - ridge_pred)^2); ridge_tmse
```

```
## [1] 186.4885
```

```
# Count the number of non-zeroed predictors are left in the trained model
ridge_coef = coef(ridge_fit$finalModel, ridge_fit$bestTune$lambda)
ridge_coef_count = ridge_coef@p[2]; ridge_coef_count
```

```
## [1] 187
```

```
# Fit a lasso model (L1 regularization, alpha = 1)
lasso_fit = train(train_pred, train_resp,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = 1,
                                         lambda = exp(seq(-3, -1, length=100))),
                  preProc = c("center", "scale"),
                  trControl = ctrl)
ggplot(lasso_fit, highlight = TRUE)
```




```
# Get the optimal penalty term lambda
lasso_fit$bestTune
```

```
##    alpha    lambda
## 81     1 0.2506147
```

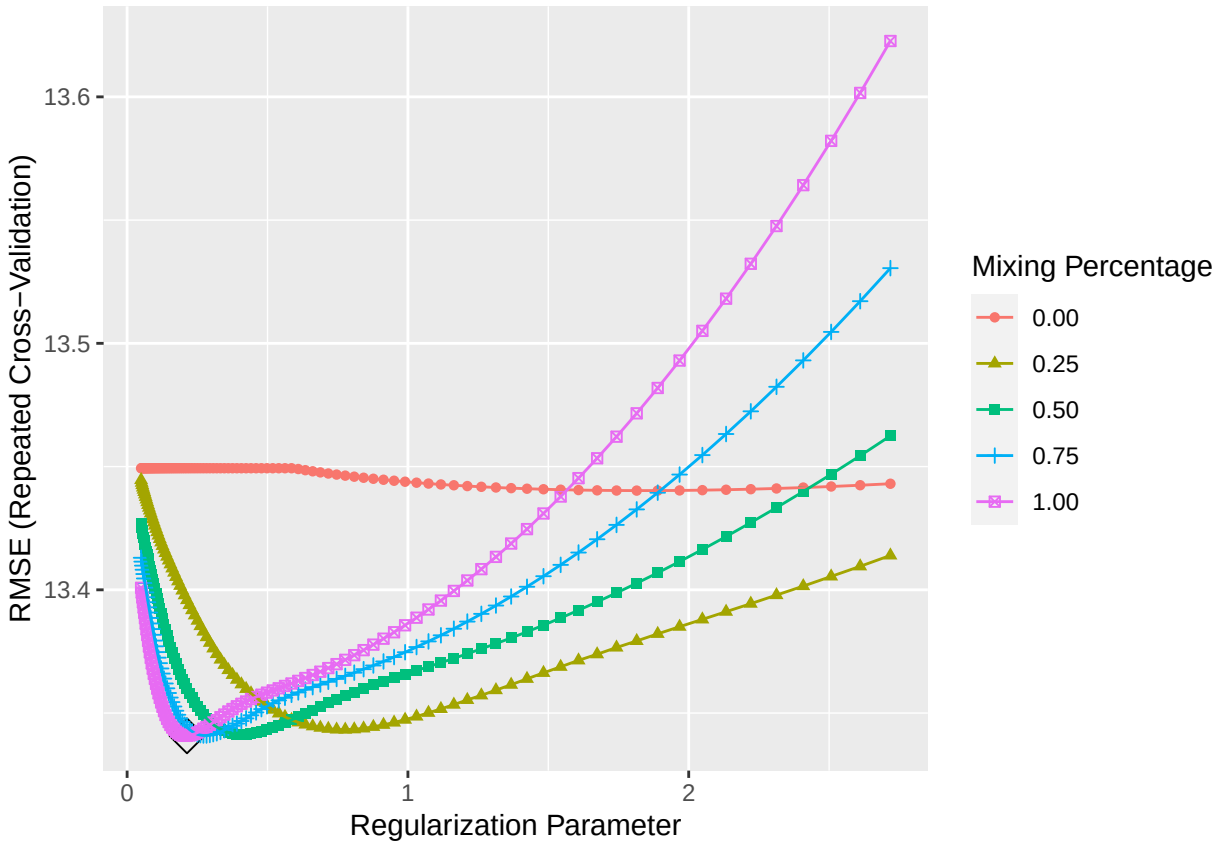
```
# Make prediction on test data and evaluate model using test MSE
lasso_pred = predict(lasso_fit, newdata = test_pred)
lasso_tmse = mean((test_resp - lasso_pred)^2); lasso_tmse
```

```
## [1] 183.1347
```

```
# Count the number of non-zeroed predictors are left in the trained model
lasso_coef = coef(lasso_fit$finalModel, lasso_fit$bestTune$lambda)
lasso_coef_count = lasso_coef@p[2]; lasso_coef_count
```

```
## [1] 30
```

```
# Fit an elastic net model (L1 & L2 regularization, alpha = [0,1])
enet_fit = train(train_pred, train_resp,
                 method = "glmnet",
                 tuneGrid = expand.grid(alpha = seq(0, 1, length = 5),
                                        lambda = exp(seq(1, -3, length = 100))),
                 trControl = ctrl)
ggplot(enet_fit, highlight = TRUE)
```



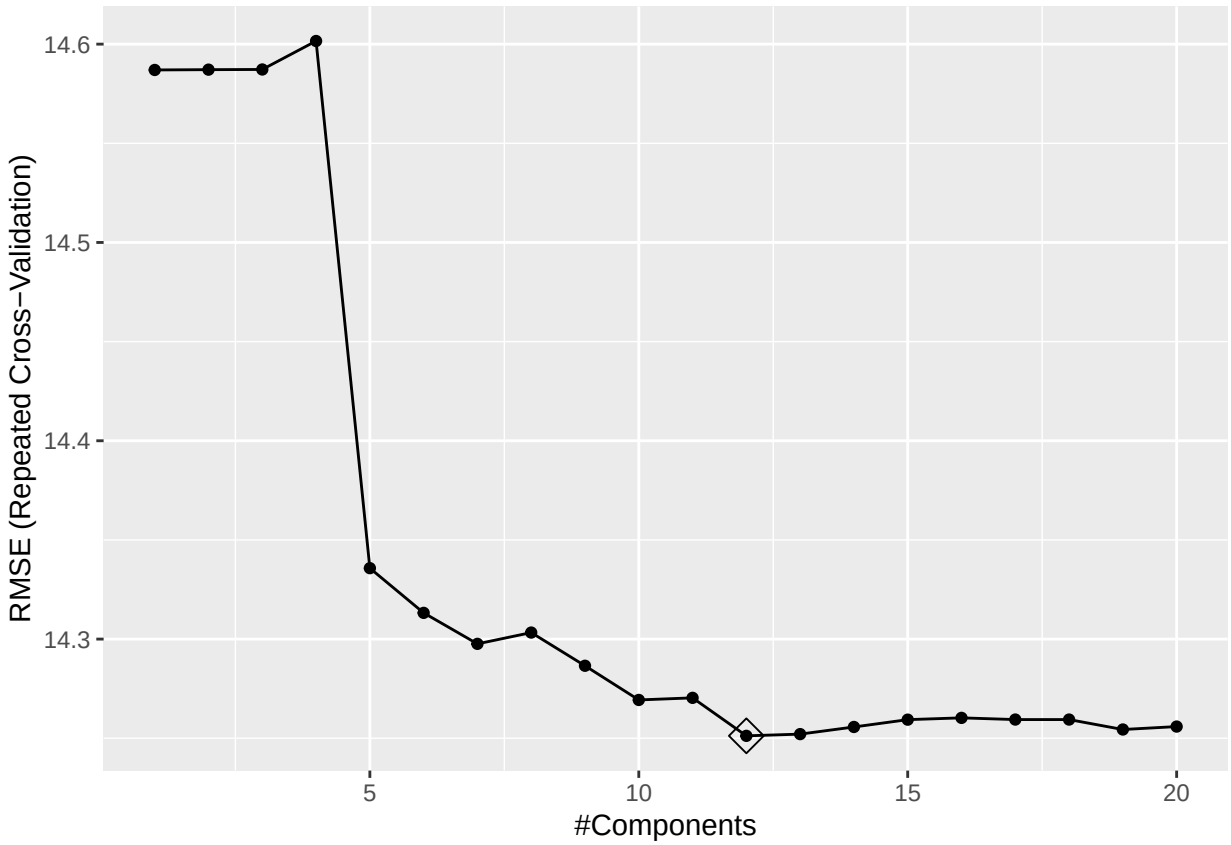

```
# Get tuning parameters alpha and lambda values. Alpha = 1, which matches lasso?
enet_fit$bestTune
```

```
##     alpha    lambda
## 437     1 0.2132149
```

```
# Make prediction on test data and evaluate model using test MSE
enet_pred = predict(enet_fit, newdata = test_pred)
enet_tmse = mean((test_resp - enet_pred)^2); enet_tmse
```

```
## [1] 183.2322
```

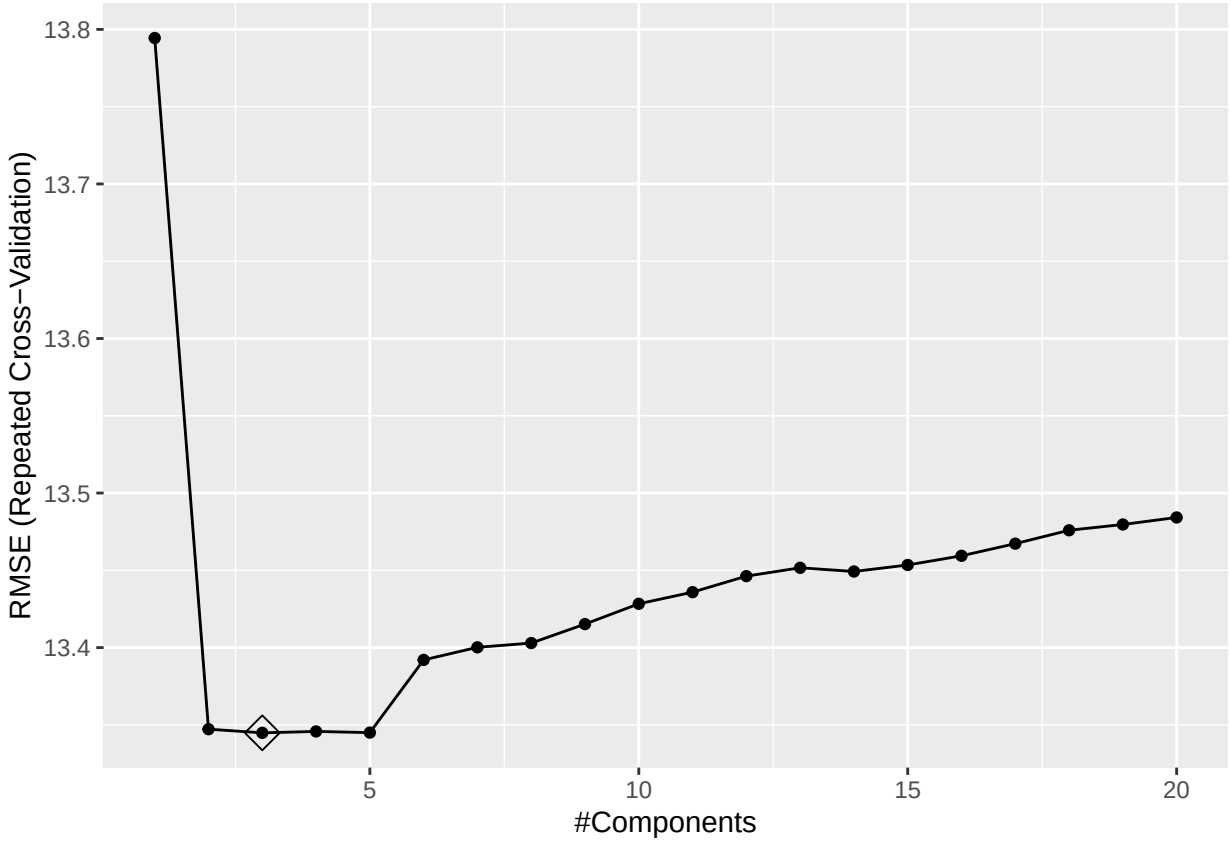
```
# Fit a principle component model
pcr_fit = train(train_pred, train_resp,
                  method = "pcr",
                  tuneGrid = data.frame(ncomp = 1:20),
                  trControl = ctrl,
                  scale = TRUE)
ggplot(pcr_fit, highlight = TRUE)
```




```
# Make prediction on test data and evaluate model using test MSE
pcr_pred = predict(pcr_fit, newdata = test_pred)
pcr_tmse = mean((test_resp - pcr_pred)^2); pcr_tmse
```

```
## [1] 208.0477
```

```
# Fit a partial least squares model
pls_fit = train(train_pred, train_resp,
                method = "pls",
                tuneGrid = data.frame(ncomp = 1:20), ## ncomp range shortened for better graphing
                trControl = ctrl)
ggplot(pls_fit, highlight = TRUE)
```




```
# Make prediction on test data and evaluate model using test MSE
pls_pred = predict(pls_fit, newdata = test_pred)
pls_tmse = mean((test_resp - pls_pred)^2); pls_tmse
```

```
## [1] 182.7374
```

```
# Get the number of model components
pls_fit$bestTune
```

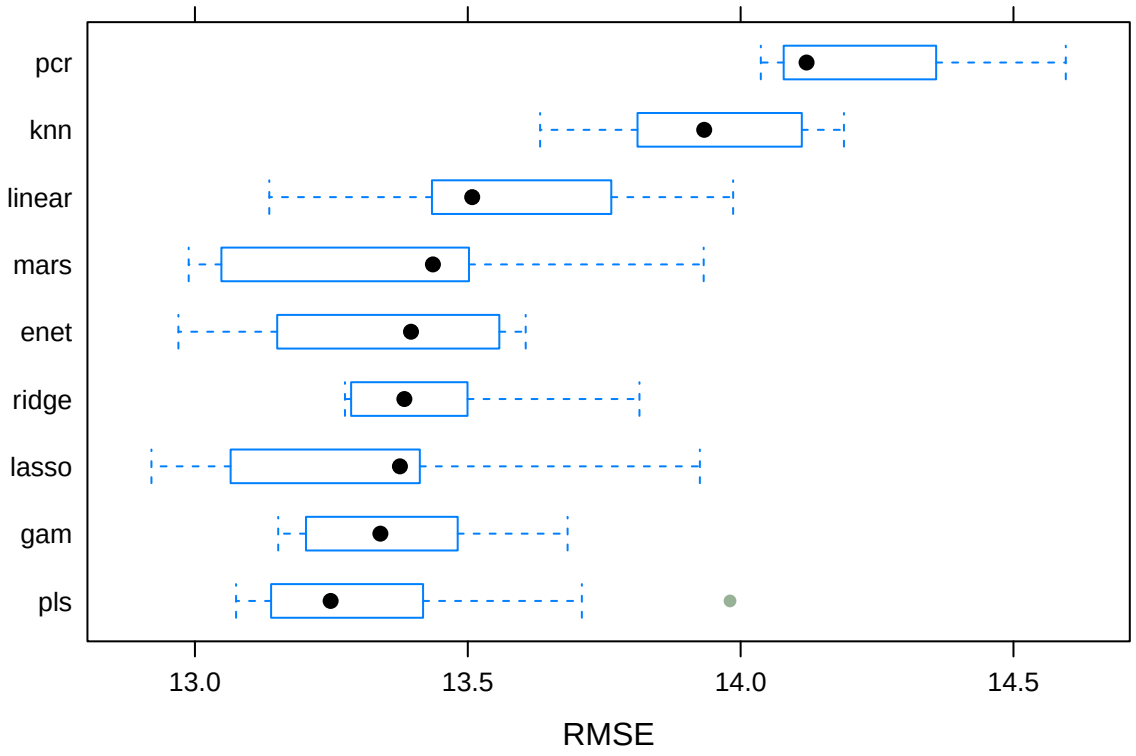
```
##   ncomp
## 3     3
```

Fit nonlinear models. Might have to rely on the Cluster.

```
gam_fit = train(train_pred, train_resp,
                method = "gam",
                tuneGrid = data.frame(method = "GCV.Cp", select = c(TRUE, FALSE)),
                trControl = ctrl)

mars_fit = train(train_pred, train_resp,
                 method = "earth",
                 tuneGrid = expand.grid(degree = 1:3, nprune = 2:20),
                 trControl = ctrl)
```

```
bwplot(
  resamples(
    list(linear = lm_fit,
         knn = knn_fit,
         ridge = ridge_fit,
```

```
        lasso = lasso_fit,
        enet = enet_fit,
        pcr = pcr_fit,
        pls = pls_fit,
        gam = gam_fit,
        mars = mars_fit)
  ),
metric = "RMSE")
```




```
summary(resamples(
    list(linear = lm_fit,
         knn = knn_fit,
         ridge = ridge_fit,
         lasso = lasso_fit,
         enet = enet_fit,
         pcr = pcr_fit,
         pls = pls_fit,
         gam = gam_fit,
         mars = mars_fit)
    ))
```

```
##
## Call:
## summary.resamples(object = resamples(list(linear = lm_fit, knn = knn_fit,
##  ridge = ridge_fit, lasso = lasso_fit, enet = enet_fit, pcr = pcr_fit, pls
##  = pls_fit, gam = gam_fit, mars = mars_fit)))
##
## Models: linear, knn, ridge, lasso, enet, pcr, pls, gam, mars
## Number of resamples: 10
##
```

```
## MAE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## linear 10.24216 10.44864 10.58979 10.56850 10.66370 10.89478    0
## knn    10.74894 10.86056 10.96461 10.95011 11.01587 11.12262    0
## ridge  10.33602 10.36719 10.41556 10.48973 10.56808 10.85939    0
## lasso  10.17295 10.26763 10.40514 10.38981 10.44047 10.74512    0
## enet   10.08150 10.26467 10.44438 10.39193 10.52456 10.60996    0
## pcr    11.03947 11.04040 11.04132 11.13222 11.17859 11.31586    7
## pls    10.10920 10.28612 10.34093 10.39466 10.45852 10.82748    0
## gam    10.30659 10.34206 10.38000 10.43385 10.49093 10.68507    0
## mars   10.06541 10.24016 10.50537 10.42035 10.53646 10.79772    0
##
## RMSE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## linear 13.13626 13.44278 13.50810 13.56691 13.74235 13.98627    0
## knn    13.63257 13.83122 13.93322 13.94581 14.08716 14.18933    0
## ridge  13.27494 13.29545 13.38383 13.44976 13.49717 13.81456    0
## lasso  12.92028 13.13159 13.37565 13.33827 13.40776 13.92526    0
## enet   12.96969 13.16577 13.39594 13.34075 13.53690 13.60611    0
## pcr    14.03697 14.07888 14.12079 14.25121 14.35833 14.59586    7
## pls    13.07539 13.14463 13.24865 13.34486 13.40681 13.98054    0
## gam    13.15261 13.20936 13.33982 13.36288 13.45522 13.68290    0
## mars   12.98853 13.09560 13.43608 13.36570 13.50059 13.93228    0
##
## Rsquared
##               Min.    1st Qu.    Median       Mean   3rd Qu.       Max. NA's
## linear 0.11909559 0.13699594 0.1408481 0.14183356 0.1464810 0.16313434    0
## knn    0.09590953 0.09780164 0.1022351 0.10487281 0.1130401 0.11462174    0
## ridge  0.13515565 0.14608379 0.1497185 0.15119866 0.1538830 0.17076612    0
## lasso  0.13718239 0.15279991 0.1704806 0.16558339 0.1754417 0.19081735    0
## enet   0.15183212 0.15557559 0.1640447 0.16518823 0.1712400 0.18415196    0
## pcr    0.04199468 0.04454114 0.0470876 0.04965256 0.0534815 0.05987541    7
## pls    0.13324538 0.15568315 0.1729499 0.16491362 0.1764384 0.18633158    0
## gam    0.13632390 0.14353541 0.1630195 0.16304636 0.1793351 0.19196907    0
## mars   0.13421789 0.14562032 0.1658069 0.16217349 0.1731540 0.19144730    0
```