Model Building

April 2, 2019

library packages

```
library(caret)
theme1 <- trellis.par.get()
theme1$plot.symbol$col <- rgb(.2, .4, .2, .2)
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)
theme_set(theme_bw() + theme(legend.position = "bottom"))
library(tidyverse)
library(caret)
library(glmnet)
library(earth)
library(mgcv)
library(splines)
library(gam)
library(boot)
library(pdp)
```

Import the data

```
train = read_csv("./data/train.csv")
test = read_csv("./data/test.csv")

options(na.action = 'na.pass')
x <- model.matrix(transformed_value~., train)[,-1]
y <- train$transformed_value
```

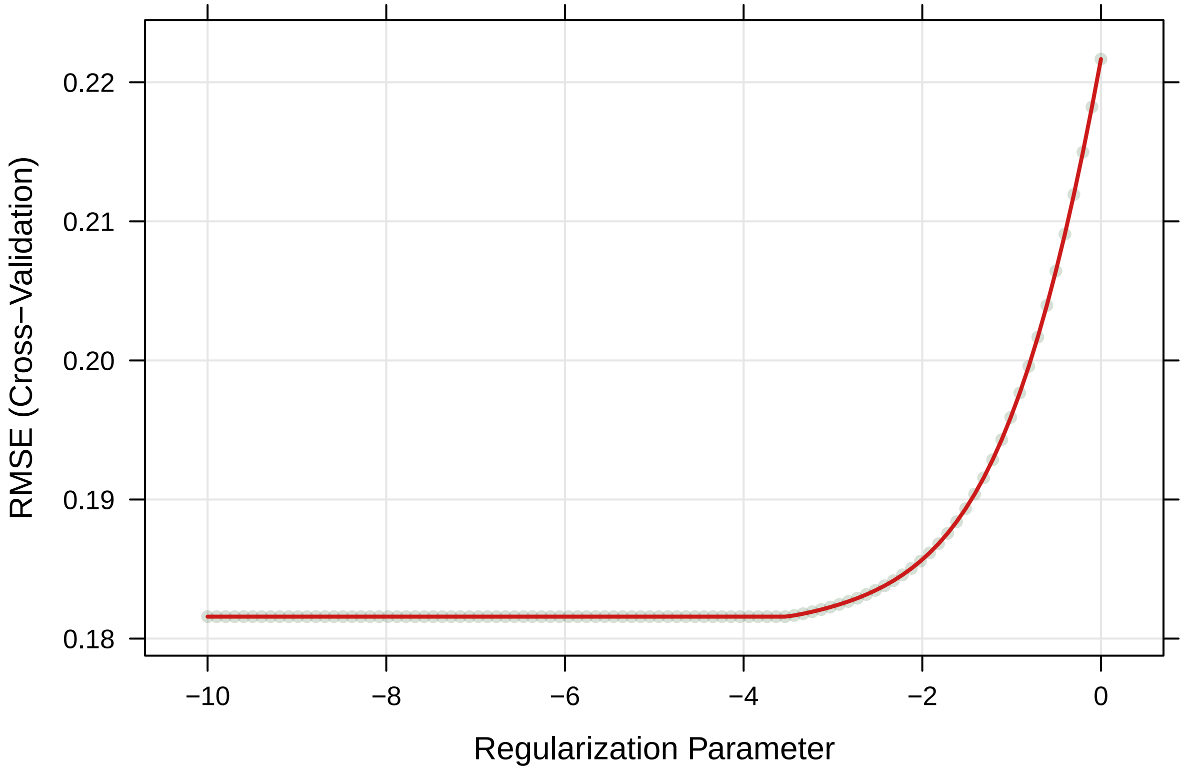
linear model

```
set.seed(2)
ctrl1 <- trainControl(method = "cv", number = 10)
lm.fit <- train(x, y,
                method = "lm",
                preProcess = "medianImpute",
                trControl = ctrl1)
```

ridge model

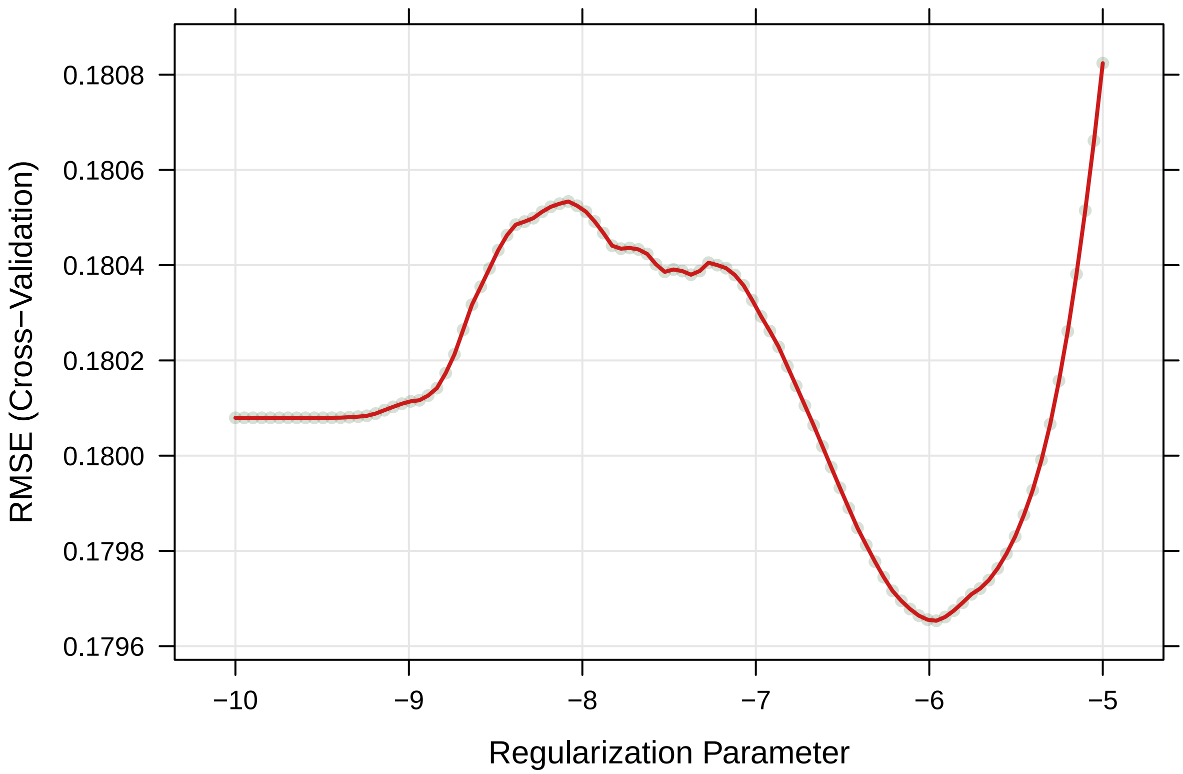
```
set.seed(2)
ridge.fit <- train(x, y,
```

```r
                    method = "glmnet",
                    tuneGrid = expand.grid(alpha = 0,
                                           lambda = exp(seq(-10, 0, length=100))),
                    preProc = c("center", "scale", "medianImpute"),
                    trControl = ctrl1)
plot(ridge.fit, xTrans = function(x) log(x))
```



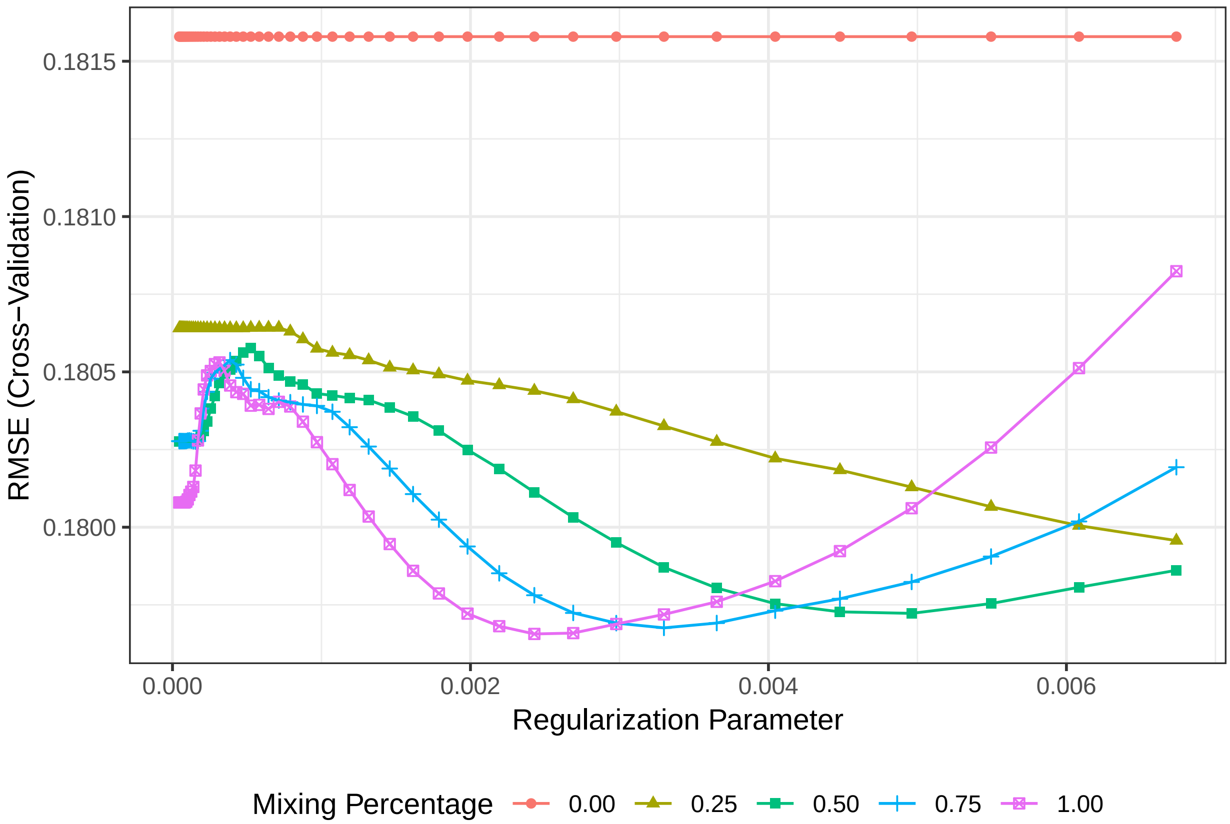
lasso model

```r
set.seed(2)
lasso.fit <- train(x, y,
                    method = "glmnet",
                    tuneGrid = expand.grid(alpha = 1,
                                           lambda = exp(seq(-10, -5, length=100))),
                    preProc = c("center", "scale", "medianImpute"),
                    trControl = ctrl1)
plot(lasso.fit, xTrans = function(x) log(x))
```



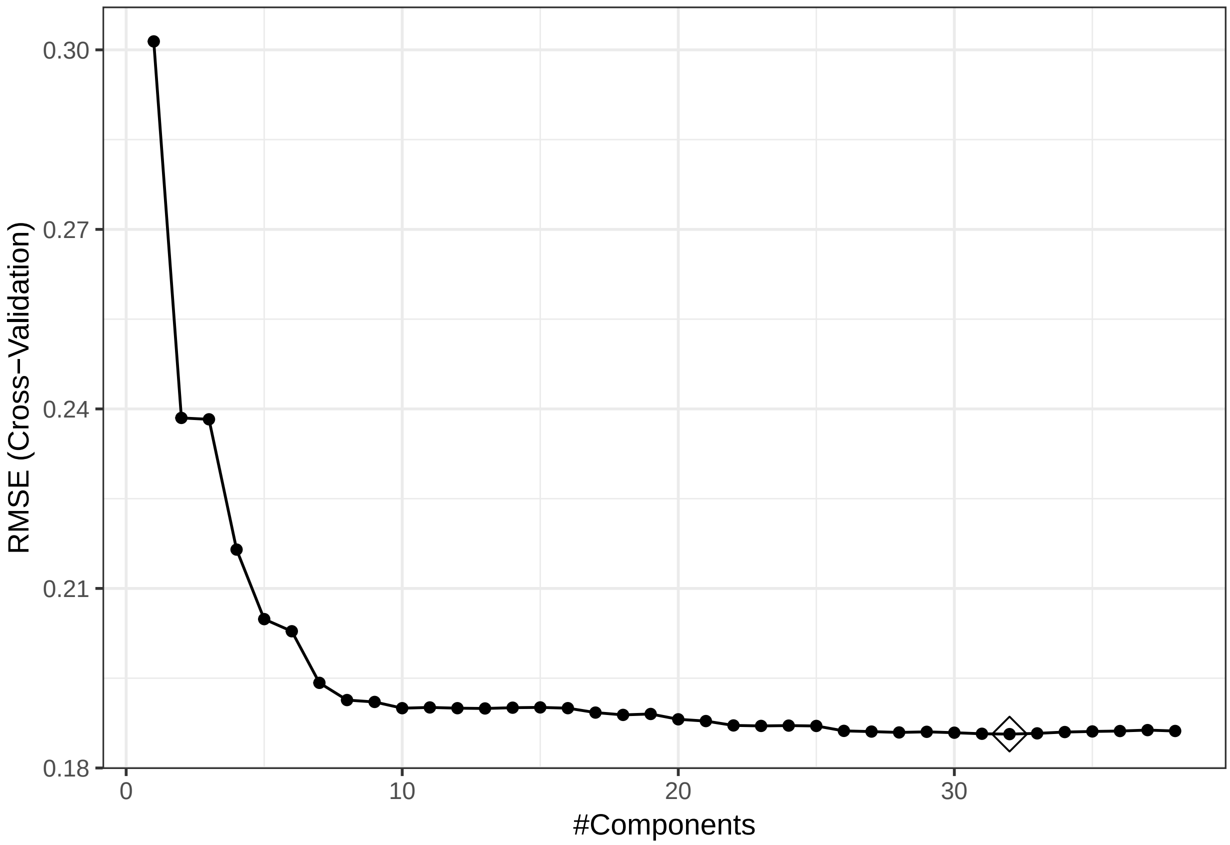
Elastic net

```
set.seed(2)
enet.fit <- train(x, y,
                  method = "glmnet",
                  tuneGrid = expand.grid(alpha = seq(0, 1, length = 5),
                                         lambda = exp(seq(-10, -5, length=50))),
                  preProc = c("center", "scale", "medianImpute"),
                  trControl = ctrl1)
ggplot(enet.fit)
```



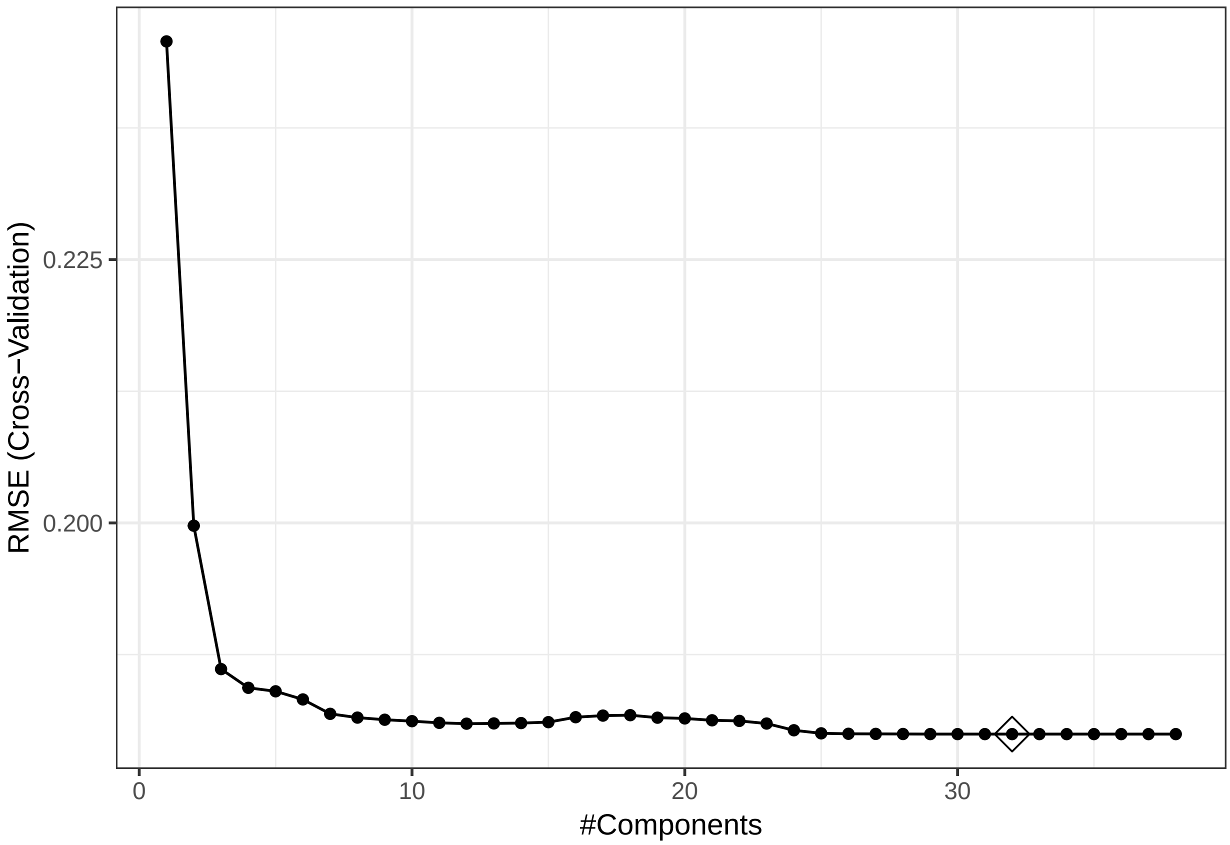
pcr model

```r
set.seed(2)
pcr.fit <- train(x, y,
                  method = "pcr",
                  tuneLength = 38,
                  trControl = ctrl1,
                  preProc = c("medianImpute"),
                  scale = TRUE)
ggplot(pcr.fit, highlight = TRUE)
```



pls model

```r
set.seed(2)
pls.fit <- train(x, y,
                 method = "pls",
                 tuneLength = 38,
                 trControl = ctrl1,
                 preProc = c("medianImpute"),
                 scale = TRUE)
ggplot(pls.fit, highlight = TRUE)
```



GAM

```r
set.seed(2)
gam.fit = train(x, y,
                preProcess = "medianImpute",
                method = "gam",
                tuneGrid = data.frame(method = "GCV.Cp",
                                      select = c(TRUE,FALSE)),
                trControl = ctrl1)

save(gam.fit, file = "./gam_fit.rda")
```

```r
load(file = "./non-linear/gam_fit.rda")
gam.fit$bestTune
```

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

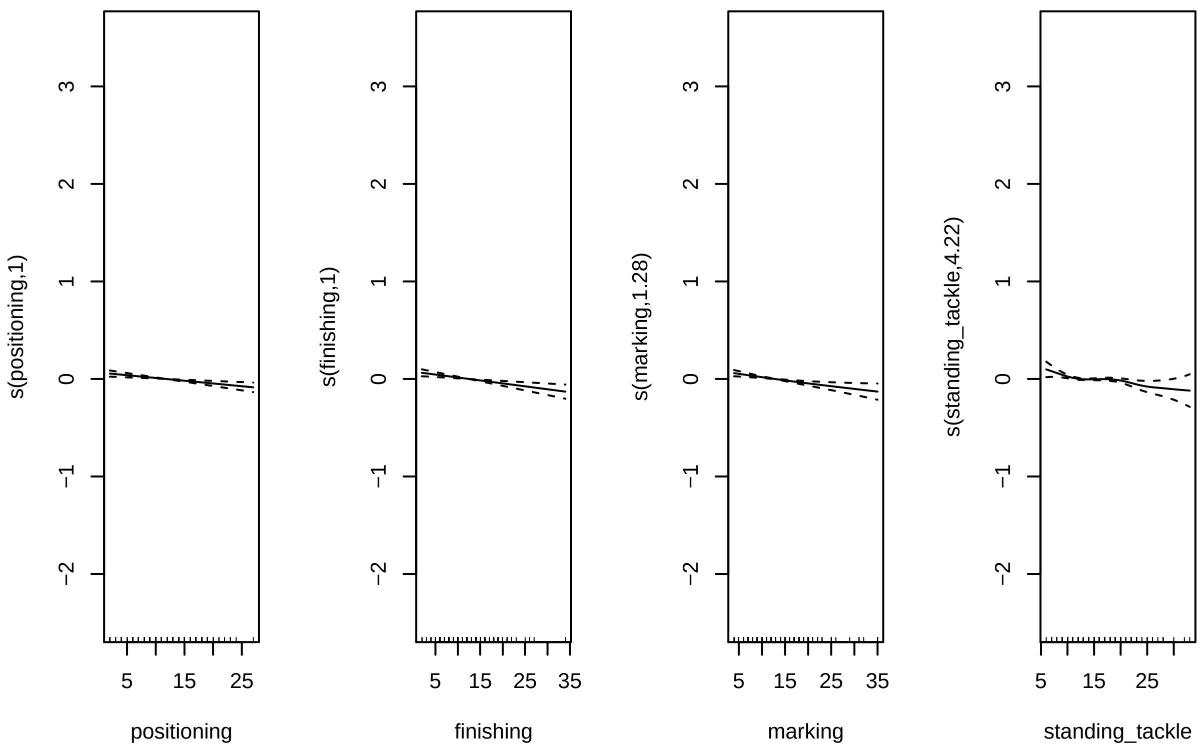
```r
gam.fit$finalModel
```

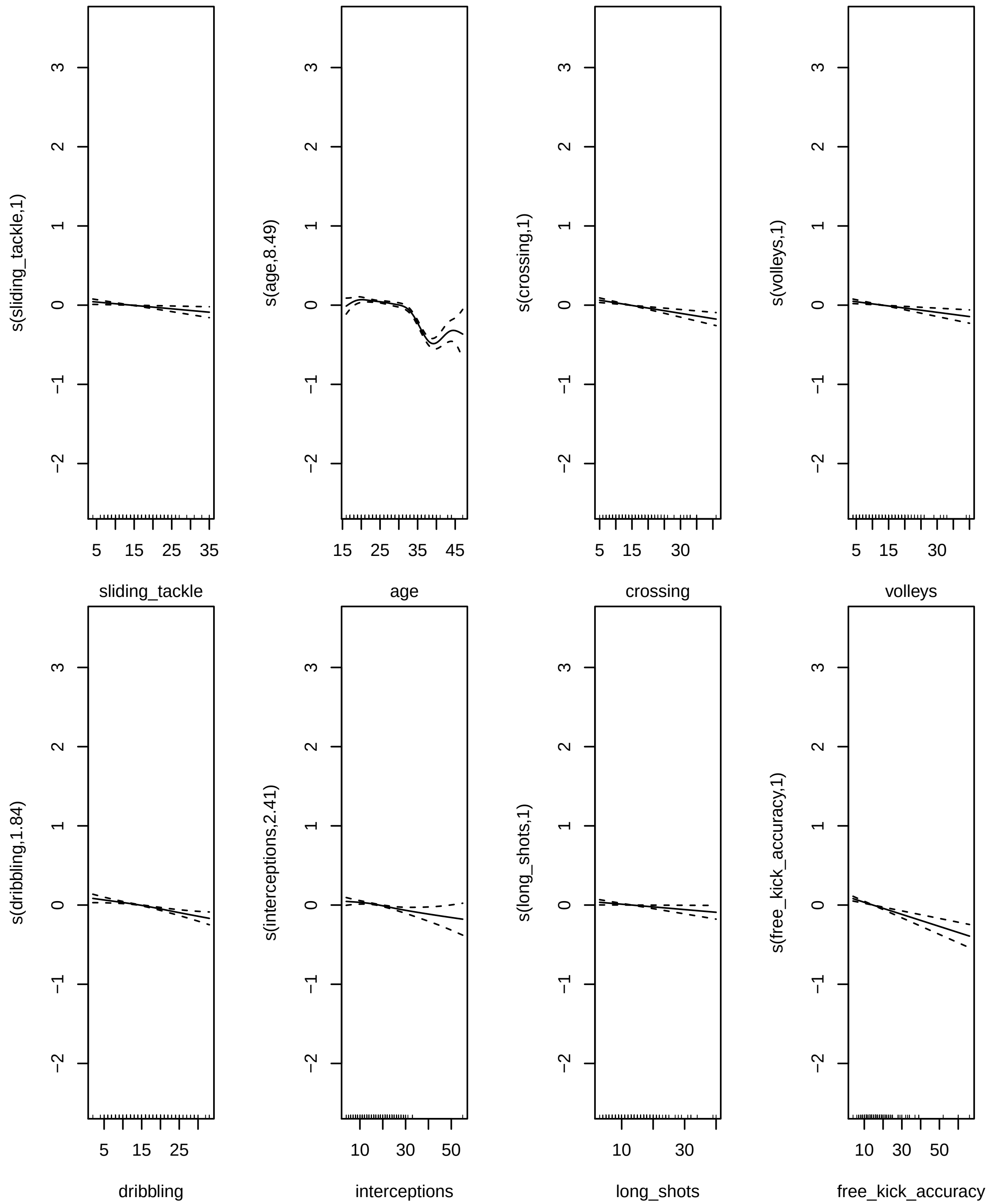
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ nationalityas + nationalityeu + nationalitysa + s(positioning) +
```

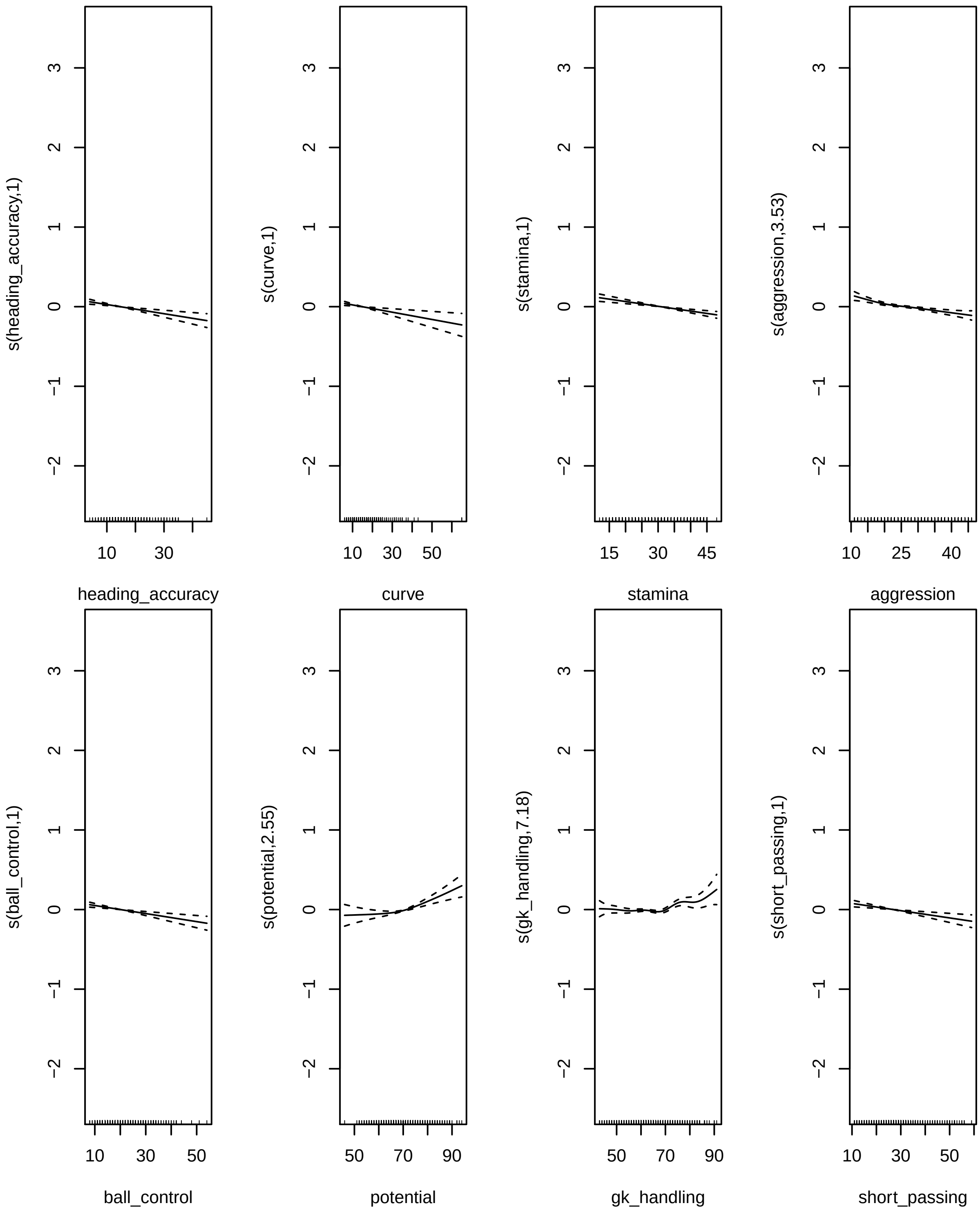
```
##     s(finishing) + s(marking) + s(standing_tackle) + s(sliding_tackle) +
##     s(age) + s(crossing) + s(volleys) + s(dribbling) + s(interceptions) +
##     s(long_shots) + s(free_kick_accuracy) + s(heading_accuracy) +
##     s(curve) + s(stamina) + s(aggression) + s(ball_control) +
##     s(potential) + s(gk_handling) + s(short_passing) + s(gk_diving) +
##     s(gk_reflexes) + s(penalties) + s(gk_kicking) + s(gk_positioning) +
##     s(long_passing) + s(shot_power) + s(acceleration) + s(sprint_speed) +
##     s(balance) + s(reactions) + s(agility) + s(composure) + s(strength) +
##     s(vision) + s(jumping) + s(special)
##
## Estimated degrees of freedom:
## 1.00 1.00 1.28 4.22 1.00 8.49 1.00
## 1.00 1.84 2.41 1.00 1.00 1.00 1.00
## 1.00 3.53 1.00 2.55 7.18 1.00 4.39
## 2.97 8.90 6.35 1.00 1.00 2.68 1.00
## 1.00 2.67 2.89 1.40 1.00 3.88 8.78
## 1.00 2.88  total = 101.29
##
## GCV score: 0.01923898
```

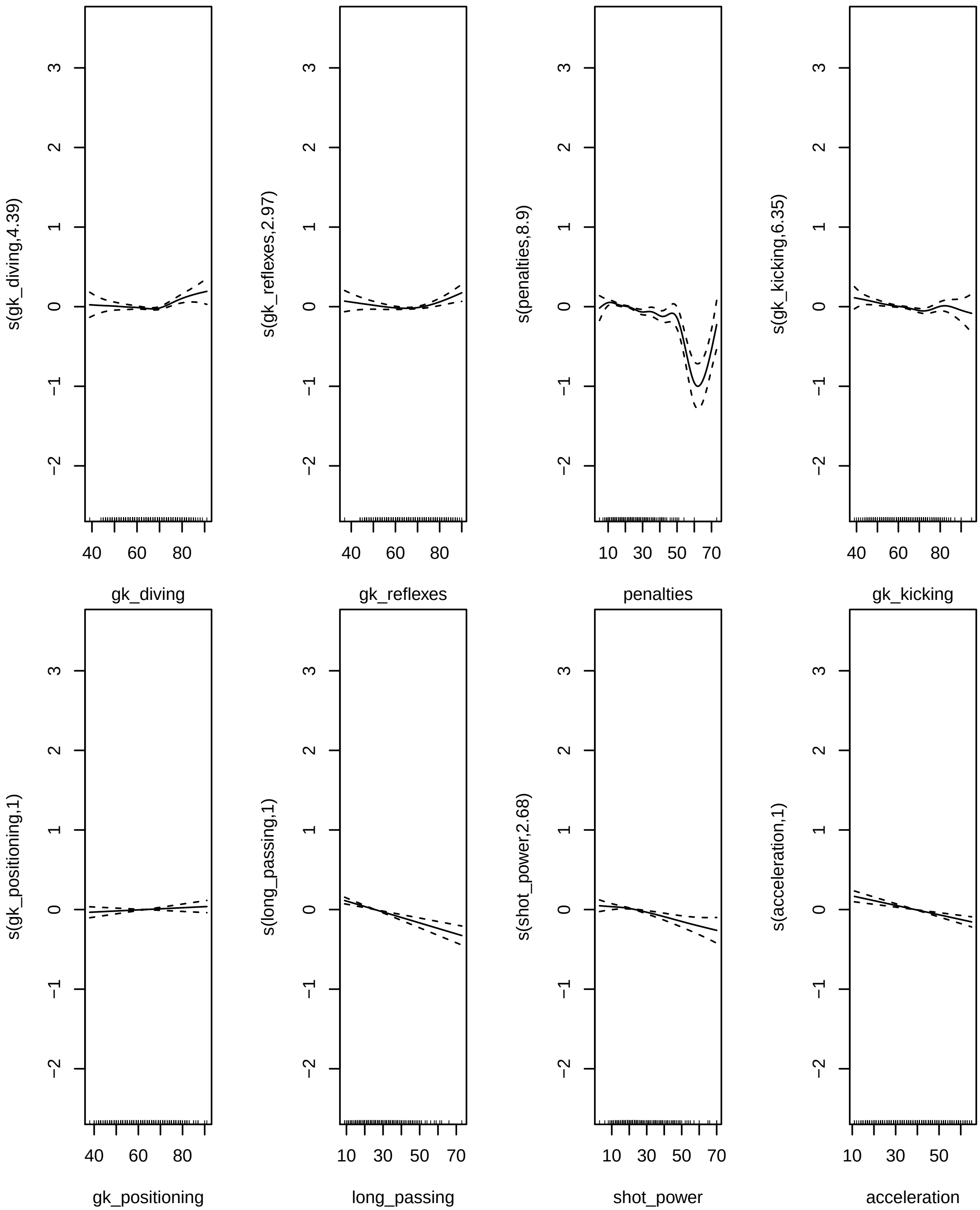
Plots of GAM

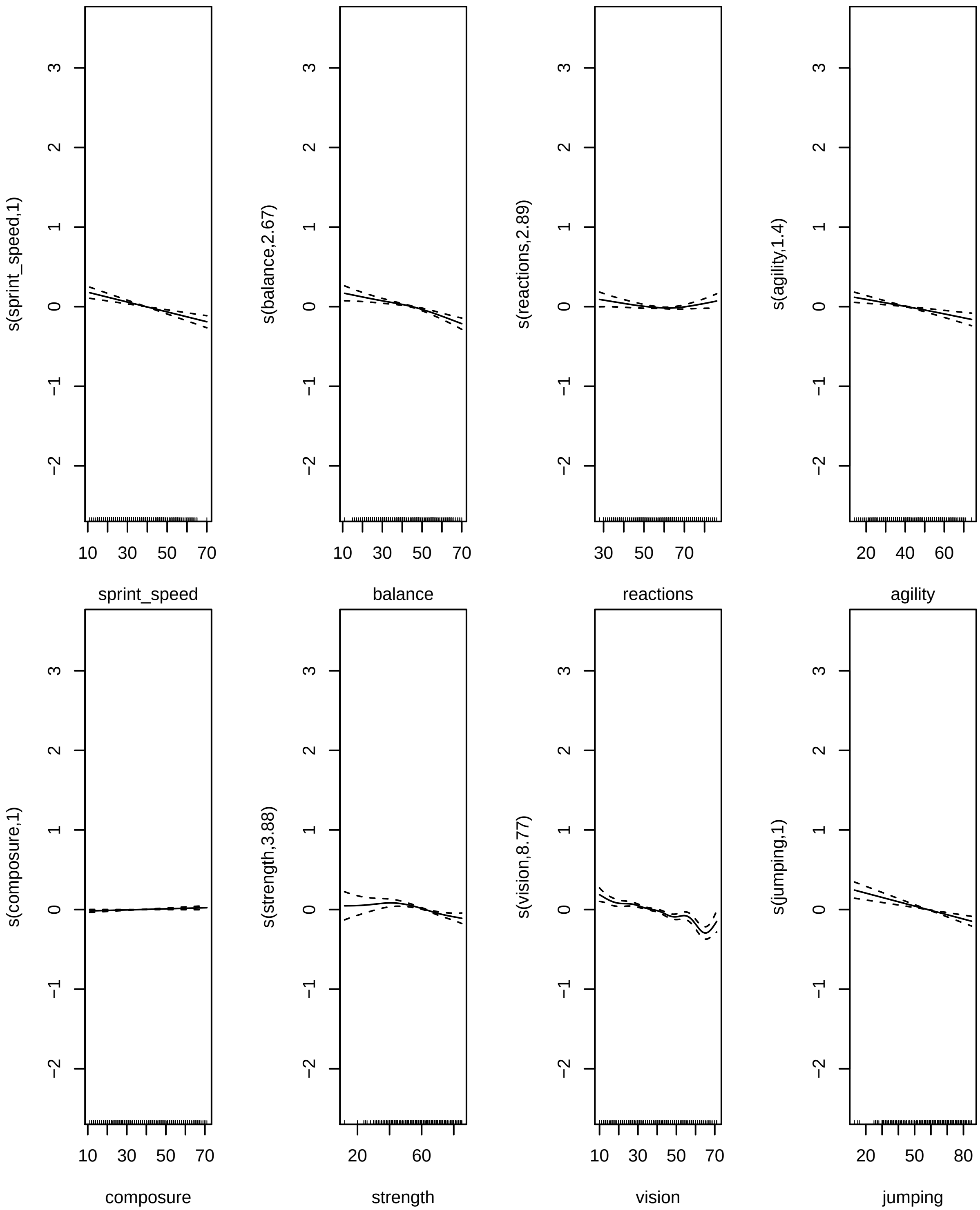
```r
par(mfrow = c(1,4))
plot(gam.fit$finalModel)
```



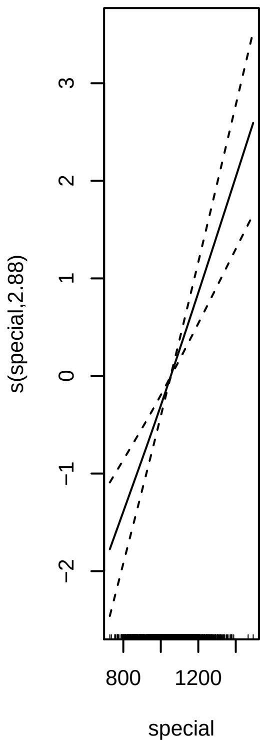

s(sliding_tackle,1)
sliding_tackle
s(age,8.49)
age
s(crossing,1)
crossing
s(volleys,1)
volleys
s(dribbling,1.84)
dribbling
s(interceptions,2.41)
interceptions
s(long_shots,1)
long_shots
s(free_kick_accuracy,1)
free_kick_accuracy


s(heading_accuracy,1)
heading_accuracy
s(curve,1)
curve
s(stamina,1)
stamina
s(aggression,3.53)
aggression
s(ball_control,1)
ball_control
s(potential,2.55)
potential
s(gk_handling,7.18)
gk_handling
s(short_passing,1)
short_passing


s(gk_diving,4.39)
gk_diving
s(gk_reflexes,2.97)
gk_reflexes
s(penalties,8.9)
penalties
s(gk_kicking,6.35)
gk_kicking
s(gk_positioning,1)
gk_positioning
s(long_passing,1)
long_passing
s(shot_power,2.68)
shot_power
s(acceleration,1)
acceleration



s(sprint_speed,1)
sprint_speed
s(balance,2.67)
balance
s(reactions,2.89)
reactions
s(agility,1.4)
agility
s(composure,1)
composure
s(strength,3.88)
strength
s(vision,8.77)
vision
s(jumping,1)
jumping



Importance of variables in GAM

```r
varImp(gam.fit)
```

```
## gam variable importance
##
##   only 20 most important variables shown (out of 40)
##
##                    Overall
## age                100.000
## vision               9.209
## penalties            7.638
## balance              7.021
## long_passing         6.349
## special              6.112
## free_kick_accuracy   6.105
## sprint_speed         5.358
## potential            5.107
## stamina              5.073
## acceleration         5.070
## jumping              4.968
## dribbling            4.891
## strength             4.882
## gk_handling          4.527
## aggression           4.331
## crossing             3.954
## gk_diving            3.585
## heading_accuracy     3.500
## ball_control         3.362
```

MARS

```r
set.seed(2)
mars_grid = expand.grid(degree = 1:2,
                        nprune = 2:38)
mars.fit = train(x, y,
                 method = "earth",
                 preProcess = "medianImpute",
                 tuneGrid = mars_grid,
                 trControl = ctrl1
                 )
save(mars.fit, file = "./earth.rda")
```

```r
load(file = "./non-linear/earth.rda")
summary(mars.fit)
```

```
## Call: earth(x=matrix[1523,42], y=c(2.795,2.772,1...), keepxy=TRUE,
##             degree=1, nprune=18)
##
##                       coefficients
## (Intercept)             0.70326337
## nationalityeu           0.04077075
## h(age-29)              -0.02402008
## h(age-33)              -0.05729609
## h(age-39)               0.12757144
## h(71-potential)        -0.00539790
## h(potential-71)         0.01690058
## h(61-agility)          -0.00171512
## h(balance-47)          -0.00319876
## h(68-gk_diving)        -0.00717104
## h(gk_diving-68)         0.01607073
## h(67-gk_handling)      -0.00482717
## h(gk_handling-67)       0.01440001
## h(gk_kicking-72)        0.00936914
## h(gk_positioning-43)    0.00790612
## h(gk_reflexes-71)       0.01509622
## h(reactions-63)         0.01070404
## h(41-strength)         -0.01290410
##
## Selected 18 of 28 terms, and 12 of 42 predictors
## Termination condition: RSq changed by less than 0.001 at 28 terms
## Importance: gk_diving, age, gk_positioning, potential, reactions, ...
## Number of terms at each degree of interaction: 1 17 (additive model)
## GCV 0.02047396    RSS 29.76514    GRSq 0.8598615    RSq 0.8660527
```

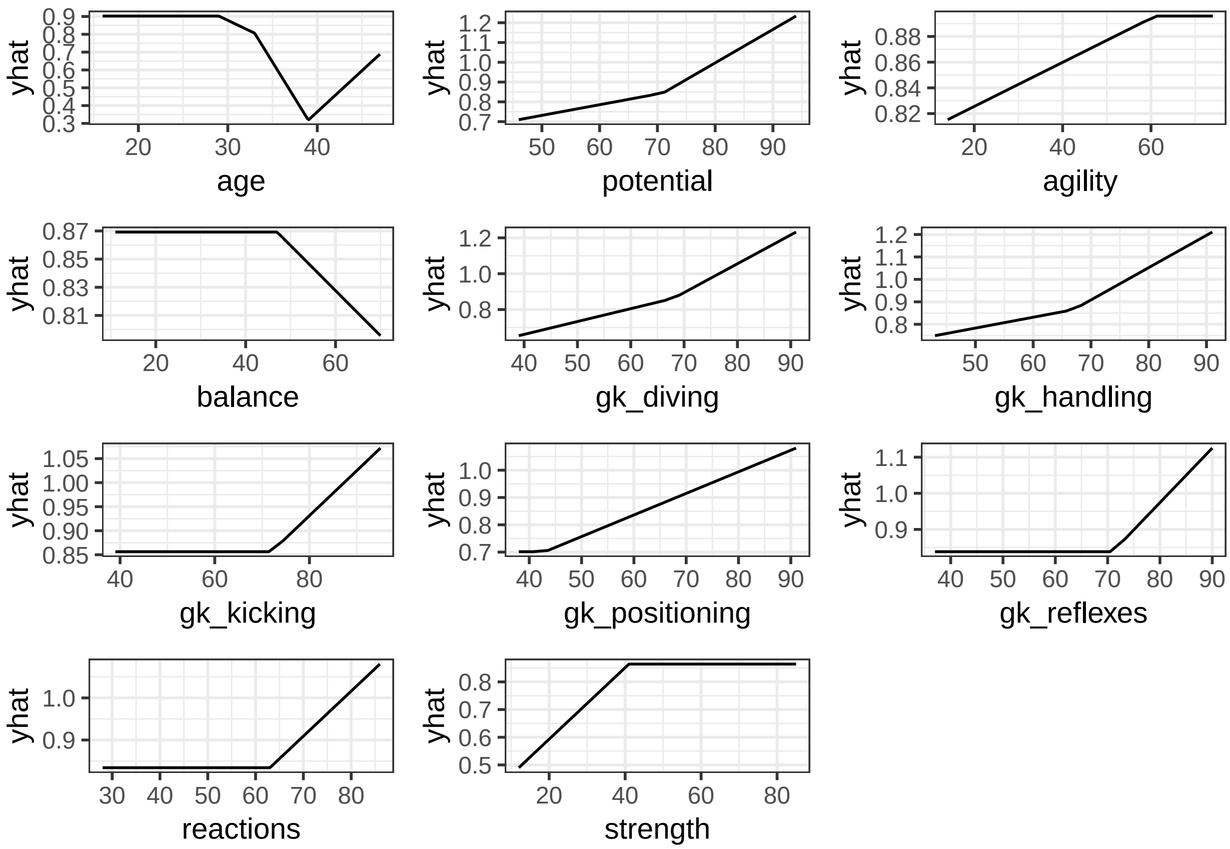
```r
mars.fit$bestTune
```

```
##    nprune degree
## 17     18      1
```

Plots of MARS

```r
p1 = partial(mars.fit, pred.var = c("age"), grid.resolution = 200) %>% autoplot()
p2 = partial(mars.fit, pred.var = c("potential"), grid.resolution = 20) %>% autoplot()
```

```r
p3 = partial(mars.fit, pred.var = c("agility"), grid.resolution = 20) %>% autoplot()
p4 = partial(mars.fit, pred.var = c("balance"), grid.resolution = 70) %>% autoplot()
p5 = partial(mars.fit, pred.var = c("gk_diving"), grid.resolution = 20) %>% autoplot()
p6 = partial(mars.fit, pred.var = c("gk_handling"), grid.resolution = 20) %>% autoplot()
p7 = partial(mars.fit, pred.var = c("gk_kicking"), grid.resolution = 20) %>% autoplot()
p8 = partial(mars.fit, pred.var = c("gk_positioning"), grid.resolution = 20) %>% autoplot()
p9 = partial(mars.fit, pred.var = c("gk_reflexes"), grid.resolution = 20) %>% autoplot()
p10 = partial(mars.fit, pred.var = c("reactions"), grid.resolution = 200) %>% autoplot()
p11 = partial(mars.fit, pred.var = c("strength"), grid.resolution = 200) %>% autoplot()
grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, ncol = 3, nrow = 4)
```



Importance of variables in MARS

```r
varImp(mars.fit)
```

```
## earth variable importance
##
##   only 20 most important variables shown (out of 42)
##
##                    Overall
## gk_diving          100.000
## age                 44.619
## gk_positioning      44.619
## gk_reflexes         32.682
## potential           30.563
```

```
## reactions         24.640
## gk_handling       18.626
## strength          14.856
## nationalityeu     10.442
## gk_kicking         7.197
## agility            3.178
## balance            3.178
## composure          0.000
## ball_control       0.000
## acceleration       0.000
## shot_power         0.000
## interceptions      0.000
## nationalityna      0.000
## crossing           0.000
## heading_accuracy   0.000
```

Formula

$$Y_{transformed} = 0.703 + 0.041X_{Europe} - 0.024h(X_{age} - 29) - 0.057h(X_{age} - 33) + 0.128h(X_{age} - 39) - 0.005h(71 - X_{potential}) + 0.017h(X_{potential} - 71) - 0.002h(61 - X_{agility}) - 0.003h(X_{balance} - 47) - 0.007h(68 - X_{gk\_diving}) + 0.016h(X_{gk\_diving} - 68) - 0.005h(67 - X_{gk\_handling}) + 0.014h(X_{gk\_handling} - 67) + 0.009h(X_{gk\_kicking} - 72) + 0.008h(X_{gk\_positioning} - 43) + 0.015h(X_{gk\_reflexes} - 71) + 0.011h(X_{reactions} - 63) - 0.013h(41 - X_{strength})$$

$$h(x) = x_+$$

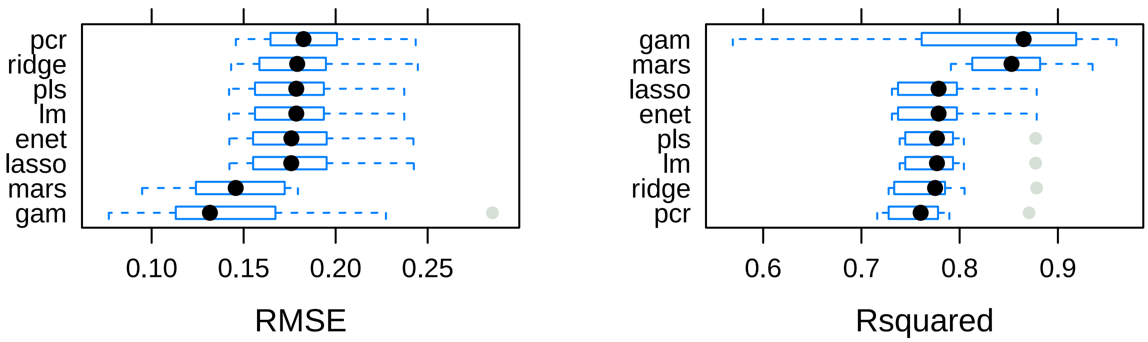
summarize

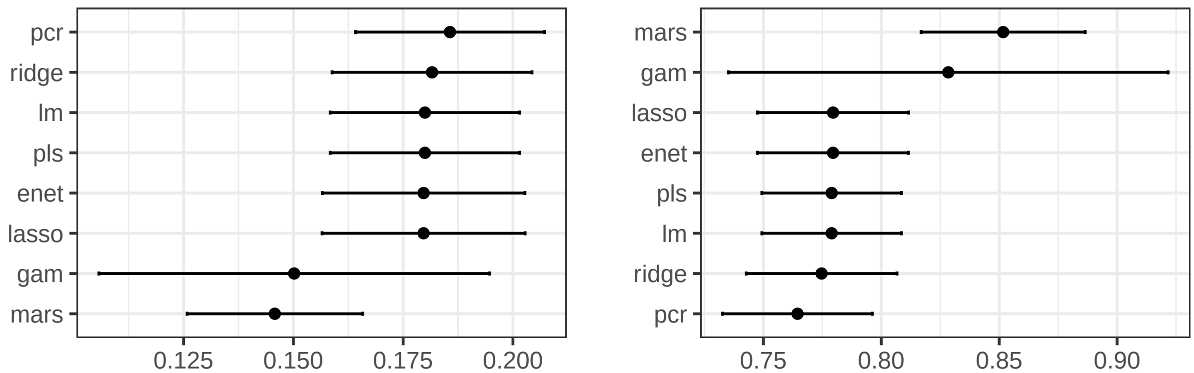
```r
load(file = "./non-linear/gam_fit.rda")
load(file = "./non-linear/earth.rda")
resamp <- resamples(list(lasso = lasso.fit,
                         ridge = ridge.fit,
                         enet = enet.fit,
                         pcr = pcr.fit,
                         pls = pls.fit,
                         lm = lm.fit,
                         gam = gam.fit,
                         mars = mars.fit
                         ))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lasso, ridge, enet, pcr, pls, lm, gam, mars
## Number of resamples: 10
##
## MAE
##             Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## lasso 0.09719222 0.10129245 0.10590401 0.10956587 0.11255805 0.13502906
```

```
## ridge 0.09765152 0.10304667 0.10684230 0.11134534 0.11391107 0.13559869
## enet  0.09724934 0.10137448 0.10598078 0.10966325 0.11269508 0.13507619
## pcr   0.10107108 0.10906116 0.11408949 0.11637942 0.12054945 0.13650323
## pls   0.09835892 0.10207320 0.10887506 0.11137731 0.11487307 0.13469098
## lm    0.09835898 0.10207339 0.10887487 0.11137730 0.11487300 0.13469094
## gam   0.05668337 0.06203842 0.07195778 0.07242399 0.08033341 0.09352210
## mars  0.06070972 0.06560657 0.06925750 0.06864674 0.07180254 0.07405092
##       NA's
## lasso    0
## ridge    0
## enet     0
## pcr      0
## pls      0
## lm       0
## gam      0
## mars     0
##
## RMSE
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso 0.14226260 0.1562634 0.1758557 0.1796533 0.1945172 0.2424922    0
## ridge 0.14318847 0.1600487 0.1790817 0.1815792 0.1937781 0.2446725    0
## enet  0.14220929 0.1562548 0.1759364 0.1796564 0.1945613 0.2422556    0
## pcr   0.14574380 0.1666581 0.1825369 0.1856613 0.1984917 0.2435124    0
## pls   0.14206567 0.1577992 0.1785876 0.1799509 0.1926953 0.2372858    0
## lm    0.14206597 0.1577995 0.1785876 0.1799510 0.1926953 0.2372858    0
## gam   0.07671379 0.1161527 0.1316582 0.1501811 0.1599001 0.2851988    0
## mars  0.09481377 0.1276158 0.1456808 0.1457794 0.1702043 0.1794824    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso 0.7310921 0.7429010 0.7785803 0.7795687 0.7956519 0.8784677    0
## ridge 0.7277164 0.7391545 0.7749554 0.7746877 0.7838059 0.8783238    0
## enet  0.7310657 0.7428786 0.7785426 0.7795444 0.7955673 0.8784539    0
## pcr   0.7161349 0.7340152 0.7603207 0.7644997 0.7751822 0.8706531    0
## pls   0.7390087 0.7468739 0.7768999 0.7789712 0.7905430 0.8773768    0
## lm    0.7390086 0.7468734 0.7768995 0.7789712 0.7905434 0.8773763    0
## gam   0.5691469 0.7800303 0.8651427 0.8284229 0.9178897 0.9595658    0
## mars  0.7911018 0.8133015 0.8528088 0.8516841 0.8787272 0.9353345    0
```

```r
a=bwplot(resamp, metric = "RMSE")
b=bwplot(resamp, metric = "Rsquared")
c=ggplot(resamp, metric = "RMSE")
d=ggplot(resamp, metric = "Rsquared")
gridExtra::grid.arrange(a,b,c,d,ncol=2,nrow=2)
```





calculate the train and test error

```
x2 <- model.matrix(transformed_value~., test)[,-1]
y2 <- test$transformed_value
## lm
trans <- preProcess(x, method = c("medianImpute"))
predy.lm <- predict(lm.fit$finalModel)
lm_train=mean((predy.lm-y)^2)
predy2.lm <- predict(lm.fit$finalModel, newdata = data.frame(predict(trans, x2)))
lm_test=mean((predy2.lm-y2)^2)

## ridge, lasso and enet
trans <- preProcess(x, method = c("center", "scale","medianImpute"))
predy.ridge <- predict(ridge.fit$finalModel, newx = predict(trans, x),
                       s = ridge.fit$bestTune$lambda, type = "response")
ridge_train=mean((predy.ridge-y)^2)
predy2.ridge <- predict(ridge.fit$finalModel, newx = predict(trans, x2),
                       s = ridge.fit$bestTune$lambda, type = "response")
ridge_test=mean((predy2.ridge-y2)^2)

predy.lasso <- predict(lasso.fit$finalModel, newx = predict(trans, x),
                       s = lasso.fit$bestTune$lambda, type = "response")
lasso_train=mean((predy.lasso-y)^2)
predy2.lasso <- predict(lasso.fit$finalModel, newx = predict(trans, x2),
                       s = lasso.fit$bestTune$lambda, type = "response")
lasso_test=mean((predy2.lasso-y2)^2)

predy.enet <- predict(enet.fit$finalModel, newx = predict(trans, x),
```

```
                            s = enet.fit$bestTune$lambda, type = "response")
enet_train=mean((predy.enet-y)^2)
predy2.enet <- predict(enet.fit$finalModel, newx = predict(trans, x2),
                            s = enet.fit$bestTune$lambda, type = "response")
enet_test=mean((predy2.enet-y2)^2)

## pcr, pls
trans <- preProcess(x, method = c("medianImpute"))
predy.pcr <- predict(pcr.fit$finalModel, ncomp = pcr.fit$bestTune$ncomp)
pcr_train=mean((predy.pcr-y)^2)
predy2.pcr <- predict(pcr.fit$finalModel, newdata = predict(trans, x2),
                       ncomp = pcr.fit$bestTune$ncomp)
pcr_test=mean((predy2.pcr-y2)^2)

predy.pls <- predict(pls.fit$finalModel, ncomp = pls.fit$bestTune$ncomp)
pls_train=mean((predy.pls-y)^2)
predy2.pls <- predict(pls.fit$finalModel, newdata = predict(trans, x2),
                       ncomp = pls.fit$bestTune$ncomp)
pls_test=mean((predy2.pls-y2)^2)

## gam
trans <- preProcess(x, method = c("medianImpute"))
predy.gam <- predict(gam.fit$finalModel)
gam_train=mean((predy.gam-y)^2)
predy2.gam <- predict(gam.fit$finalModel, newdata = data.frame(predict(trans, x2)))
gam_test=mean((predy2.gam-y2)^2)

## mars
predy.mars <- predict(mars.fit$finalModel, type = "earth")
mars_train=mean((predy.mars-y)^2)
predy2.mars <- predict(mars.fit$finalModel, newdata = data.frame(predict(trans, x2)),
                       type = "earth")
mars_test=mean((predy2.mars-y2)^2)

tibble(
    model = c("linear","ridge","lasso","elastic net","PCR","PLS","MARS","GAM"),
    train_error = c(lm_train,ridge_train,lasso_train,enet_train,pcr_train,pls_train,mars_train,gam_trai
    test_error = c(lm_test,ridge_test,lasso_test,enet_test,pcr_test,pls_test,mars_test,gam_test)
) %>% knitr::kable(digits = 4)
```

model	train_error	test_error
linear	0.0306	0.0399
ridge	0.0317	0.0393
lasso	0.0315	0.0392
elastic net	0.0314	0.0392
PCR	0.0332	0.0401
PLS	0.0306	0.0399
MARS	0.0195	0.0260
GAM	0.0168	0.0272