Nonlinear

JunLu

4/1/2019

Import the train dataset

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

y = train$transformed_value

x = model.matrix(transformed_value ~ ., train)[,-1]
```

Set a random seed

```
set.seed(1)
```

Generallized additive model (GAM)

Use caret package

```
ctrl1 <- trainControl(method = "cv", number = 10)
gam.fit = train(x, y,
                method = "gam",
                tuneGrid = data.frame(method = "GCV.Cp",
                                      select = c(TRUE,FALSE)),
                trControl = ctrl1)

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

```
load(file = "./gam_fit.rda")
gam.fit$bestTune
```

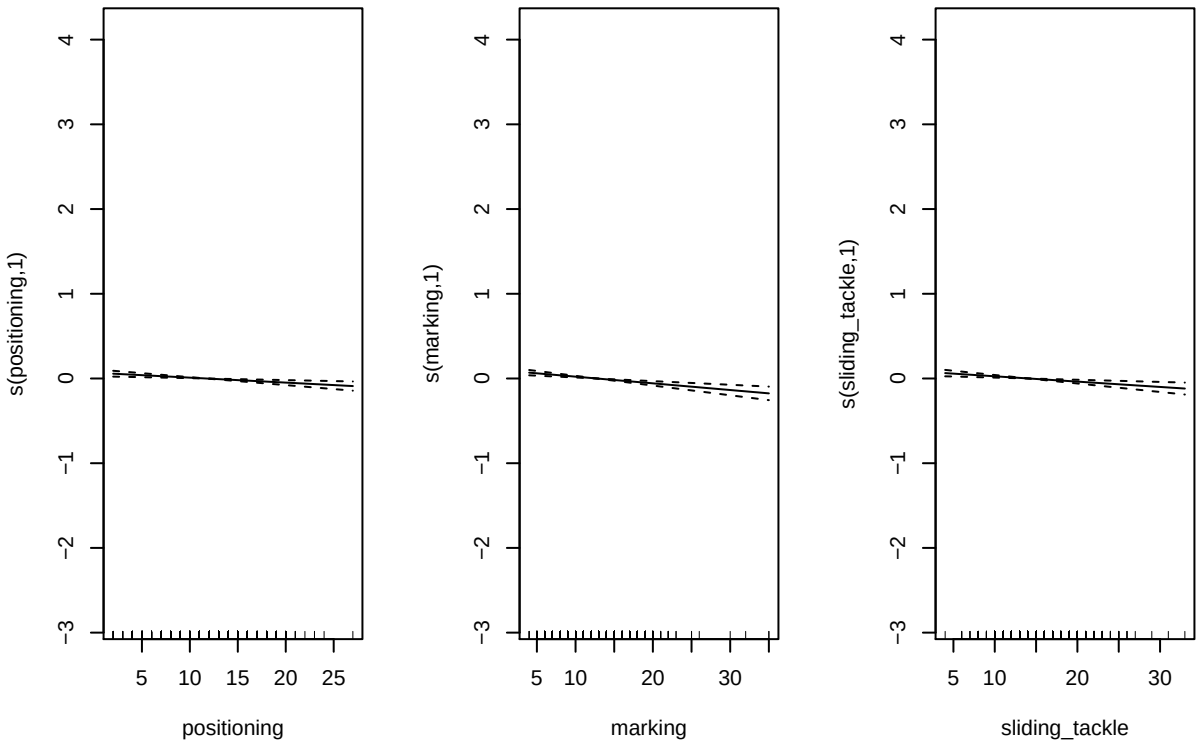
```
##   select method
## 1  FALSE GCV.Cp
```

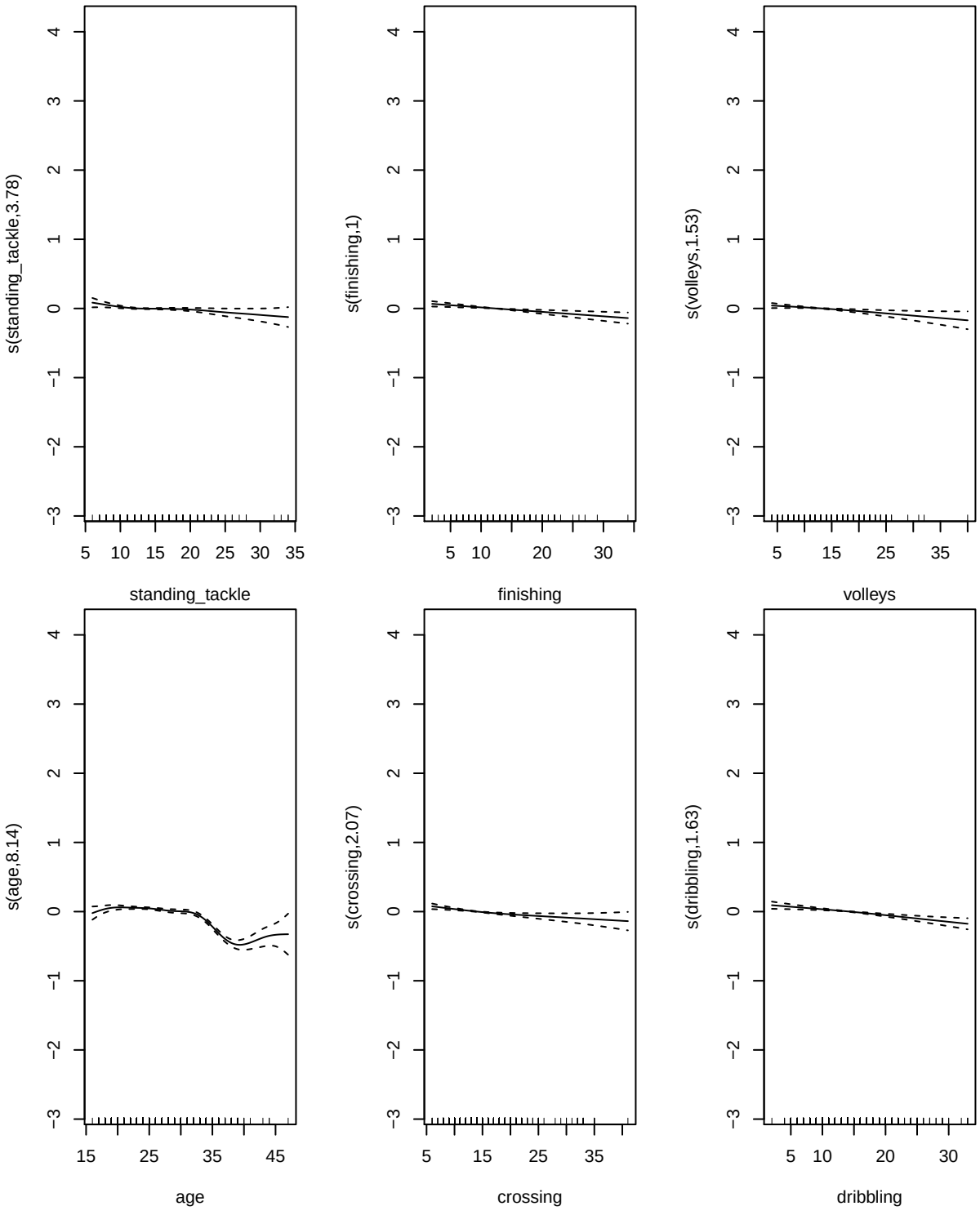
```
gam.fit$finalModel
```

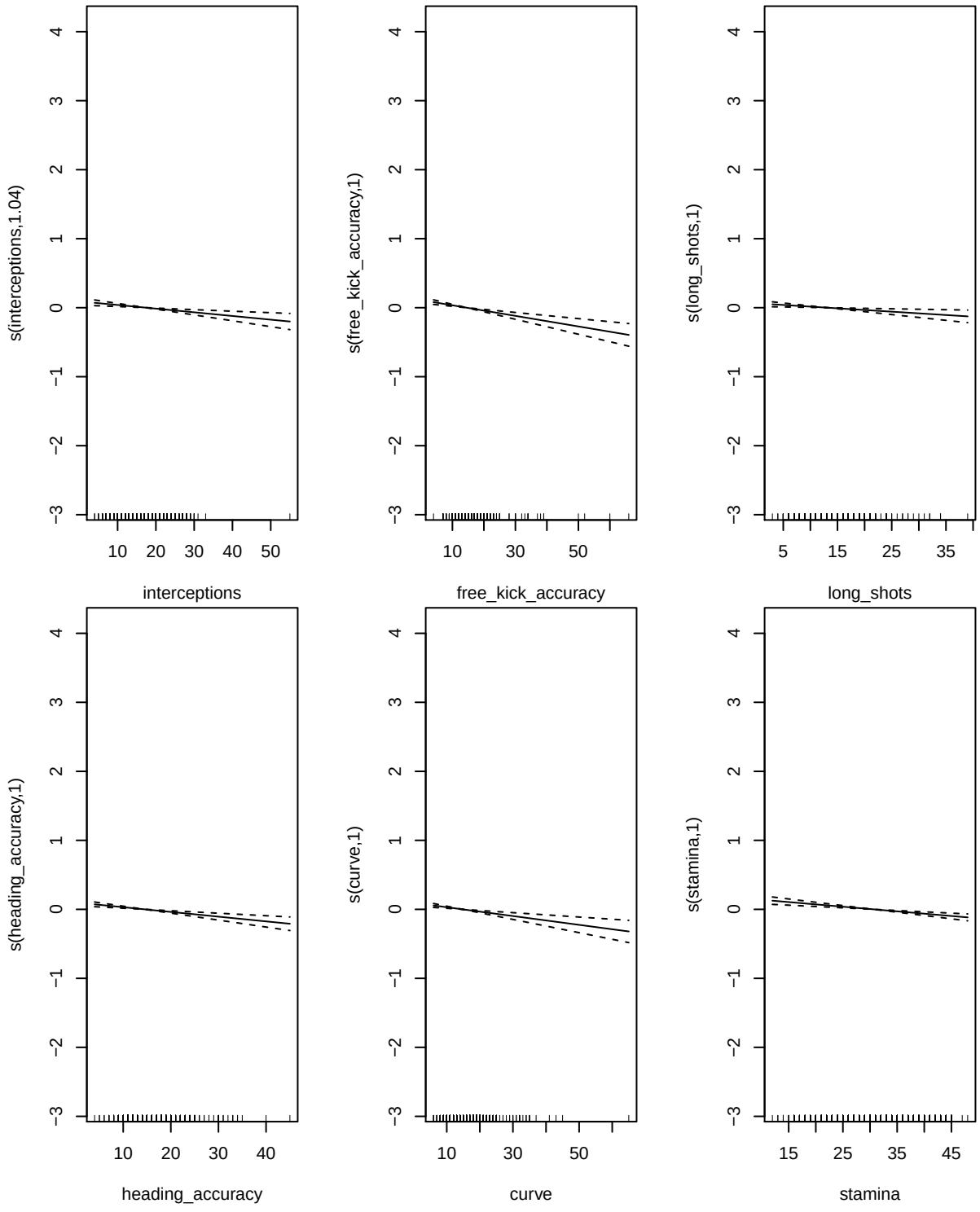
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ nationalityas + nationalityeu + nationalitysa + s(positioning) +
##     s(marking) + s(sliding_tackle) + s(standing_tackle) + s(finishing) +
##     s(volleys) + s(age) + s(crossing) + s(dribbling) + s(interceptions) +
##     s(free_kick_accuracy) + s(long_shots) + s(heading_accuracy) +
##     s(curve) + s(stamina) + s(aggression) + s(ball_control) +
##     s(potential) + s(penalties) + s(gk_handling) + s(short_passing) +
```

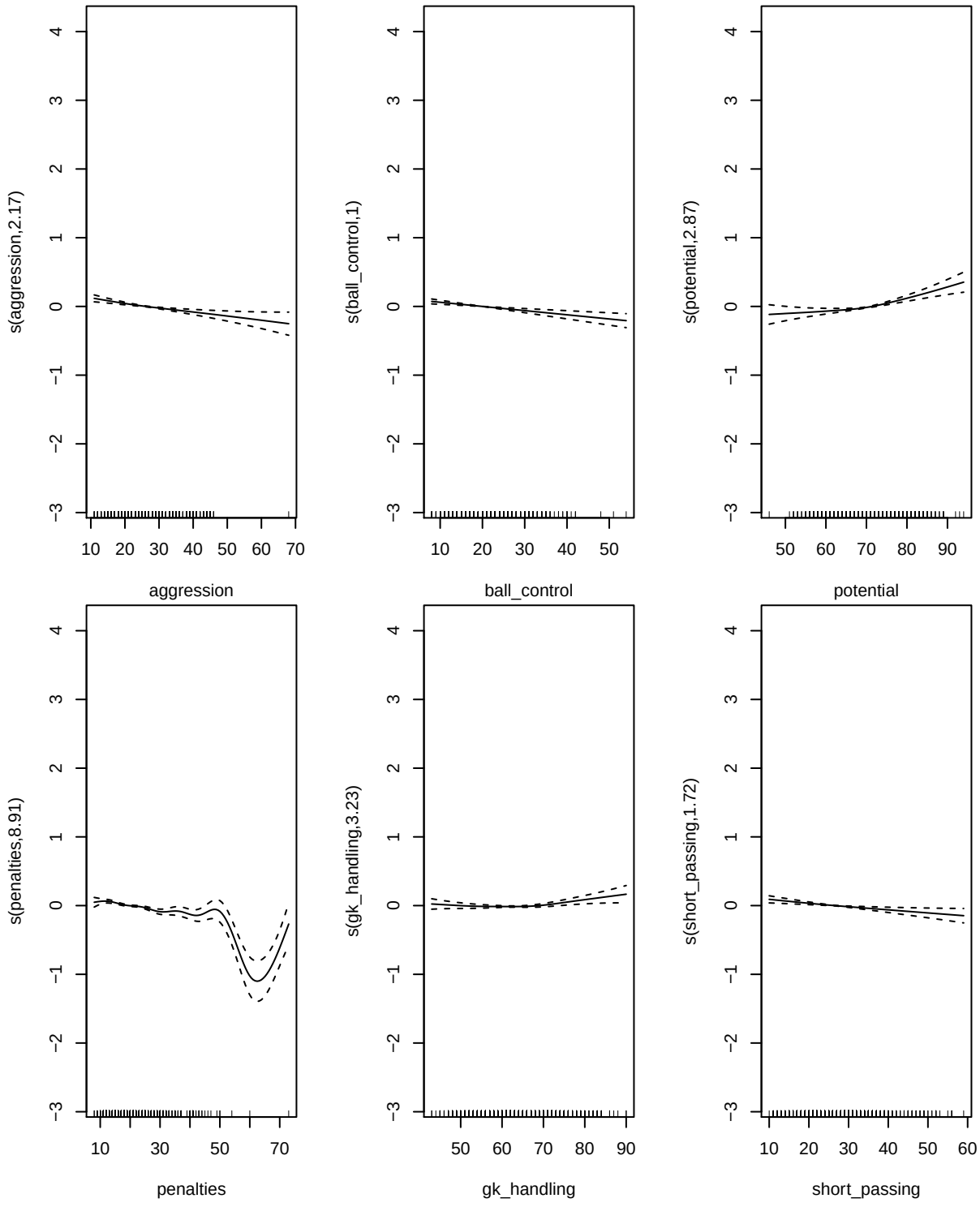
```
##     s(gk_diving) + s(gk_reflexes) + s(gk_kicking) + s(gk_positioning) +
##     s(shot_power) + s(long_passing) + s(acceleration) + s(sprint_speed) +
##     s(balance) + s(agility) + s(reactions) + s(strength) + s(jumping) +
##     s(composure) + s(vision) + s(special)
##
## Estimated degrees of freedom:
## 1.00 1.00 1.00 3.78 1.00 1.53 8.14
## 2.07 1.63 1.04 1.00 1.00 1.00 1.00
## 1.00 2.17 1.00 2.87 8.91 3.23 1.72
## 5.85 2.82 1.77 1.00 1.00 1.00 1.00
## 1.00 2.84 2.13 3.29 4.21 1.00 1.00
## 1.92 3.05  total = 85.96
##
## GCV score: 0.01906584
```

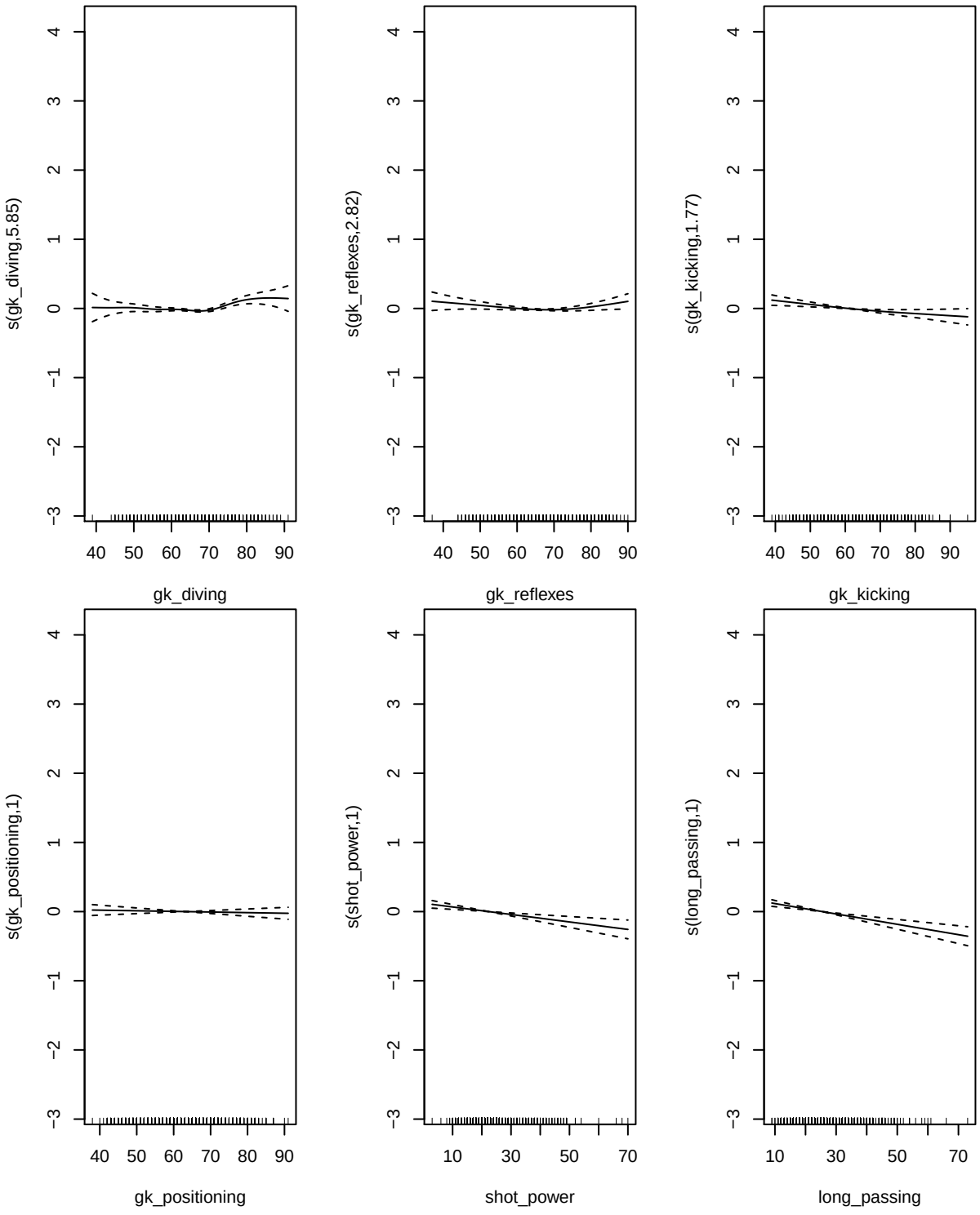
```
par(mfrow = c(1,3))
plot(gam.fit$finalModel)
```

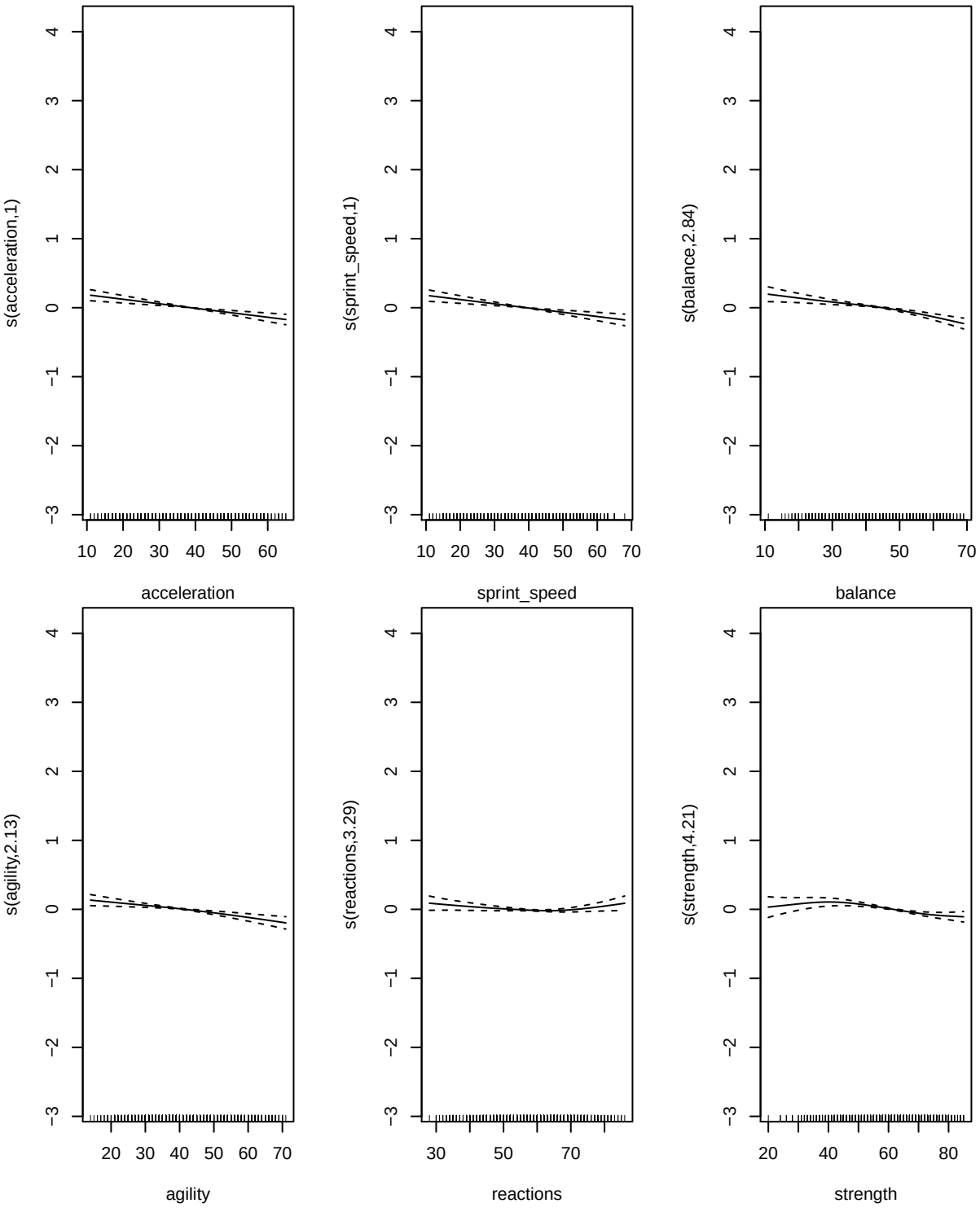


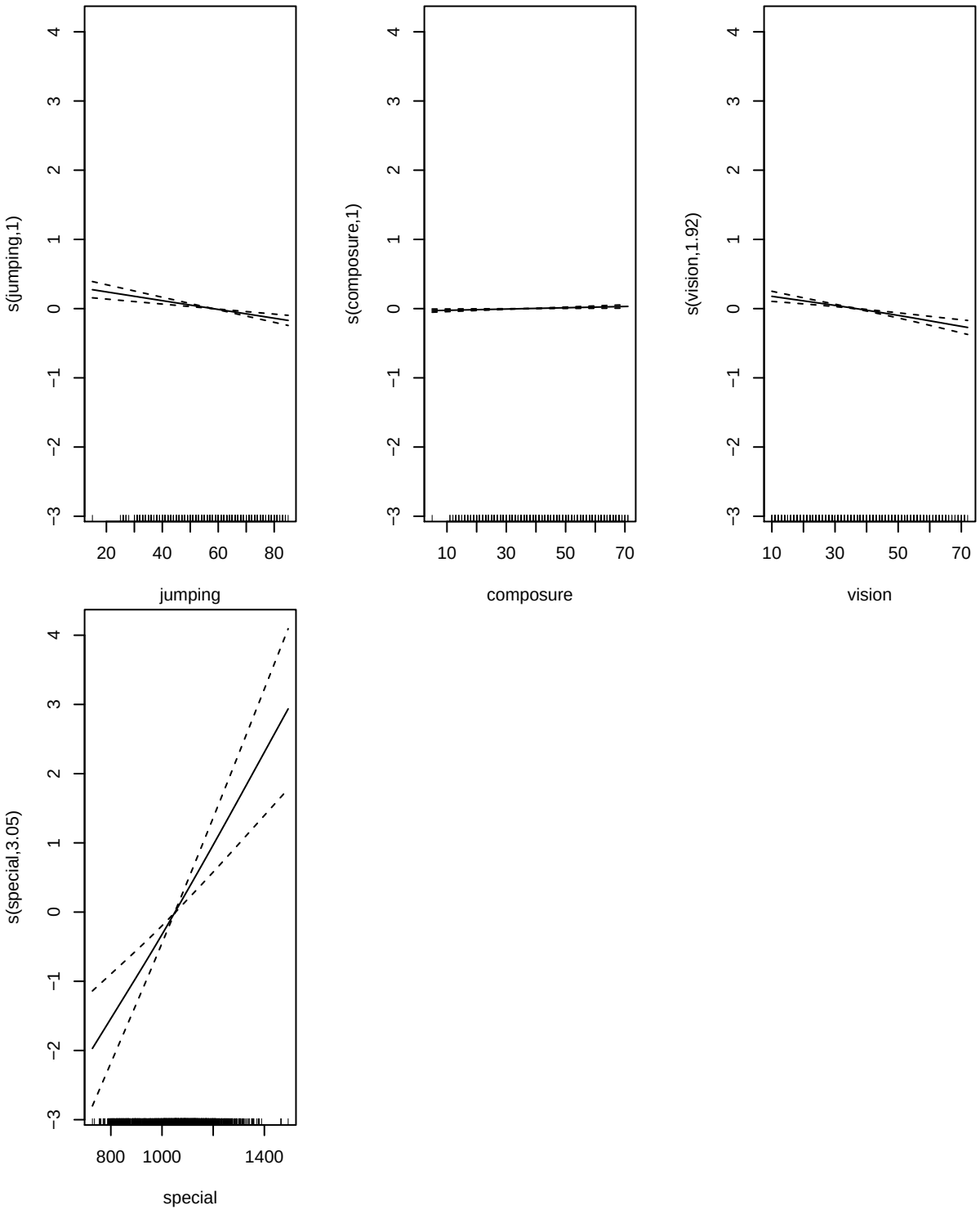

s(standing_tackle,3.78)
standing_tackle
s(finishing,1)
finishing
s(volleys,1.53)
volleys
s(age,8.14)
age
s(crossing,2.07)
crossing
s(dribbling,1.63)
dribbling


s(interceptions,1.04)
interceptions
s(free_kick_accuracy,1)
free_kick_accuracy
s(long_shots,1)
long_shots
s(heading_accuracy,1)
heading_accuracy
s(curve,1)
curve
s(stamina,1)
stamina


s(aggression,2.17)
aggression
s(ball_control,1)
ball_control
s(potential,2.87)
potential
s(penalties,8.91)
penalties
s(gk_handling,3.23)
gk_handling
s(short_passing,1.72)
short_passing


s(gk_diving,5.85)
gk_diving
s(gk_reflexes,2.82)
gk_reflexes
s(gk_kicking,1.77)
gk_kicking
s(gk_positioning,1)
gk_positioning
s(shot_power,1)
shot_power
s(long_passing,1)
long_passing


s(acceleration,1)
acceleration
s(sprint_speed,1)
sprint_speed
s(balance,2.84)
balance
s(agility,2.13)
agility
s(reactions,3.29)
reactions
s(strength,4.21)
strength



Multivariate Adaptive Regression Splines (MARS)

```
ctrl1 <- trainControl(method = "cv", number = 10)
mars_grid = expand.grid(degree = 1:2,
                        nprune = 2:38)
```

```
mars.fit = train(x, y,
                 method = "earth",
                 tuneGrid = mars_grid,
                 trControl = ctrl1
                 )
save(mars.fit, file = "./earth.rda")
```

```
load(file = "./earth.rda")
summary(mars.fit)
```

```
## Call: earth(x=matrix[1516,42], y=c(2.795,1.456,2...), keepxy=TRUE,
##             degree=1, nprune=21)
##
##                       coefficients
## (Intercept)             0.86329518
## nationalityeu           0.03060648
## h(age-29)              -0.06283675
## h(age-30)               0.08508926
## h(age-32)              -0.10065954
## h(age-39)               0.11445155
## h(72-potential)        -0.00689246
## h(potential-72)         0.01876997
## h(special-961)          0.00031866
## h(balance-49)          -0.00468522
## h(69-gk_diving)        -0.00529335
## h(gk_diving-69)         0.01933709
## h(66-gk_handling)      -0.00473194
## h(gk_handling-66)       0.01664262
## h(70-gk_positioning)   -0.00568760
## h(gk_reflexes-71)       0.01593299
## h(63-reactions)        -0.00296193
## h(reactions-63)         0.00966418
## h(40-strength)         -0.01369018
## h(vision-58)           -0.01917415
## h(vision-67)            0.10347135
##
## Selected 21 of 30 terms, and 12 of 42 predictors
## Termination condition: RSq changed by less than 0.001 at 30 terms
## Importance: gk_diving, age, gk_positioning, potential, gk_handling, ...
## Number of terms at each degree of interaction: 1 20 (additive model)
## GCV 0.01956414    RSS 28.07667    GRSq 0.8680612    RSq 0.8749363
```

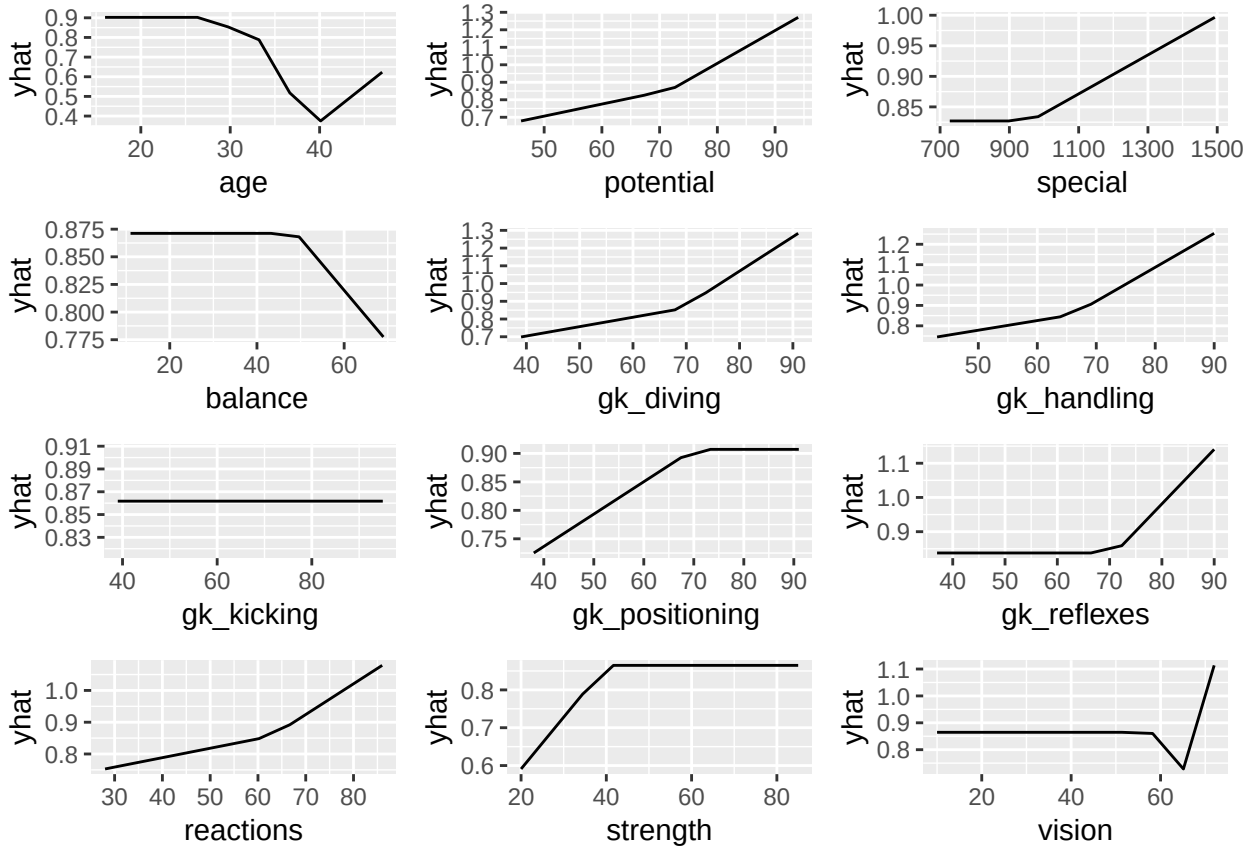
```
mars.fit$bestTune
```

```
##    nprune degree
## 20     21      1
```

```
p1 = partial(mars.fit, pred.var =c("age"), grid.resolution = 10) %>% autoplot()
p2 = partial(mars.fit, pred.var =c("potential"), grid.resolution = 10) %>% autoplot()
p3 = partial(mars.fit, pred.var =c("special"), grid.resolution = 10) %>% autoplot()
p4 = partial(mars.fit, pred.var =c("balance"), grid.resolution = 10) %>% autoplot()
p5 = partial(mars.fit, pred.var =c("gk_diving"), grid.resolution = 10) %>% autoplot()
p6 = partial(mars.fit, pred.var =c("gk_handling"), grid.resolution = 10) %>% autoplot()
p7 = partial(mars.fit, pred.var =c("gk_kicking"), grid.resolution = 10) %>% autoplot()
p8 = partial(mars.fit, pred.var =c("gk_positioning"), grid.resolution = 10) %>% autoplot()
```

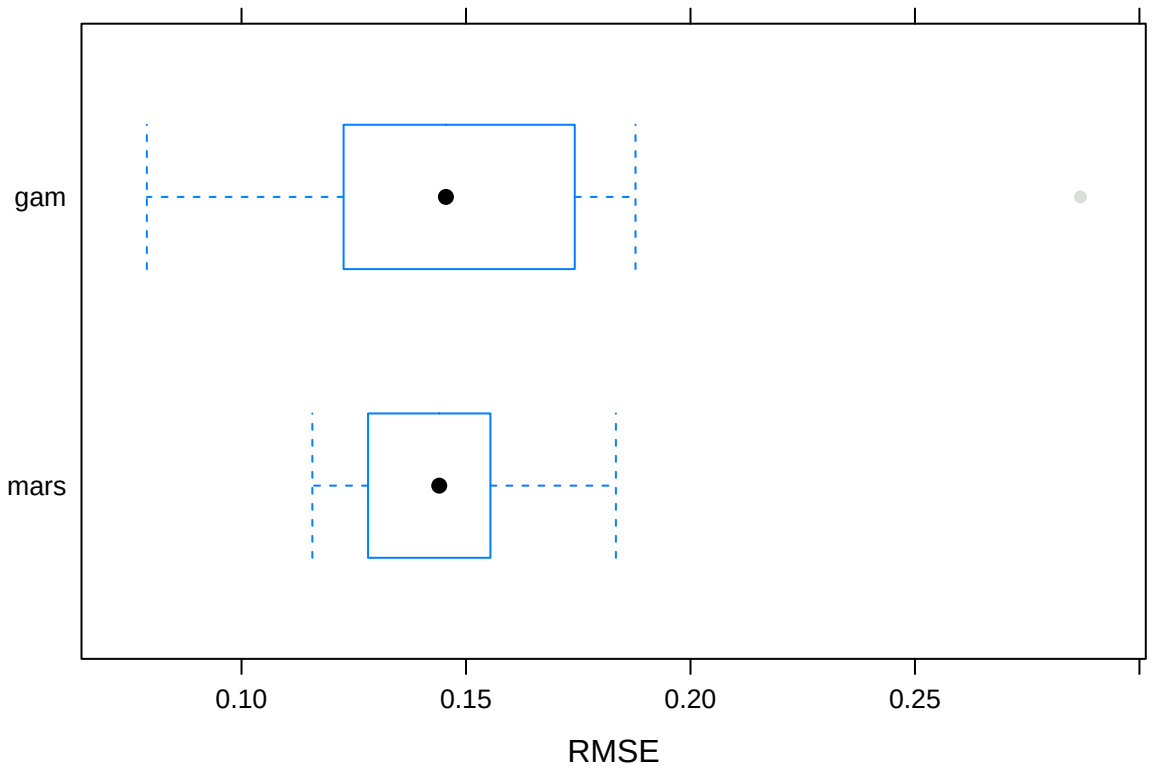
```
p9 = partial(mars.fit, pred.var =c("gk_reflexes"), grid.resolution = 10) %>% autoplot()
p10 = partial(mars.fit, pred.var =c("reactions"), grid.resolution = 10) %>% autoplot()
p11 = partial(mars.fit, pred.var =c("strength"), grid.resolution = 10) %>% autoplot()
p12 = partial(mars.fit, pred.var =c("vision"), grid.resolution = 10) %>% autoplot()

grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, ncol = 3, nrow = 4)
```



Compare those models

```
bwplot(resamples(list(mars = mars.fit,gam = gam.fit)),
       metric = "RMSE")
```



Importance

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

```
## earth variable importance
##
##   only 20 most important variables shown (out of 42)
##
##                Overall
## gk_diving      100.000
## age             43.209
## gk_positioning  43.209
## potential       33.810
## gk_handling     26.189
## reactions       19.768
## strength        13.791
## gk_reflexes     12.171
## special          8.858
## vision           8.858
## balance          7.914
## nationalityeu    4.214
## positioning      0.000
## aggression       0.000
## long_shots       0.000
## ball_control     0.000
## volleys          0.000
## nationalityas    0.000
## agility          0.000
```

```
## jumping          0.000
```

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

```
## gam variable importance
##
##   only 20 most important variables shown (out of 40)
##
##                     Overall
## age                 100.000
## penalties             9.470
## potential             6.659
## balance               6.658
## gk_diving             6.587
## long_passing          6.191
## special               6.086
## strength              5.631
## vision                5.491
## free_kick_accuracy    5.366
## dribbling             5.208
## stamina               5.119
## jumping               5.061
## acceleration          4.781
## marking               4.547
## heading_accuracy      4.282
## sprint_speed          4.247
## ball_control          3.946
## aggression            3.891
## curve                 3.768
```

Questions

1. As we can't use the test dataset to choose our final model, do we need to calculate test error for each model or just our final model.

Do we write (linear -> nonlinear, if we know it is nonlinear why we try linear firstly)

2. How to know our model is good or not, just with a train error and a test error. (close and both small -> good, train << test overfitted, both large underfitted) However, how do we know it is small or large (MSE also depends on y)
3. Which variables are important? Different models have different important variables. Do we only use the result of our final model?
4. how do we present our model in the report (coefficients? tunning parameter? plot?)
5. Model assumptions and limitations

linear Multi linear: model linear in parameter (error term mean 0 constant variance uncorrelated)

lasso and ridge: linear (Anything else? Multicollinearity?)

pcr: linear (Anything else? Multicollinearity?)

non-linear gam: Nonlinear relationship (Anything else?) mars: Nonlinear relationship (Anything else)

6. How to choose tunning parameters? like we try lambda from (e^-10, e^10), then we choose the best lambda by 10-fold cross vaildation.

7.How do we know model is enough flexible? How do we make prediction?

Set the model by exploratory analysis

```
gam.m1 = gam(transformed_value ~ age + nationality + potential + special + acceleration
             + aggression + agility + balance + ball_control + composure + crossing + curve +
                 dribbling + finishing + free_kick_accuracy + gk_diving + gk_handling + gk_kicking +
                 gk_positioning + gk_reflexes + heading_accuracy + interceptions +
                 jumping + long_passing + long_shots + marking + penalties + positioning +
                 reactions + short_passing + shot_power + sliding_tackle + sprint_speed + stamina +
                 standing_tackle + strength + vision + volleys, data = train)

gam.m2 = gam(transformed_value ~ s(age) + nationality  + potential + special + acceleration
             + aggression + agility + balance + ball_control + composure + crossing + curve +
                 dribbling + finishing + free_kick_accuracy + gk_diving + gk_handling + gk_kicking +
                 gk_positioning + gk_reflexes + heading_accuracy + interceptions +
                 jumping + long_passing + long_shots + marking + penalties + positioning +
                 reactions + short_passing + shot_power + sliding_tackle + sprint_speed + stamina +
                 standing_tackle + strength + vision + volleys, data = train)

gam.m3 = gam(transformed_value ~ s(age) + nationality + potential + special + acceleration
             + aggression + agility + balance + ball_control + composure + crossing + curve +
                 dribbling + finishing + free_kick_accuracy + gk_diving + gk_handling + gk_kicking +
                 gk_positioning + gk_reflexes + heading_accuracy + interceptions +
                 jumping + long_passing + long_shots + marking + penalties + positioning +
                 s(reactions) + short_passing + shot_power + sliding_tackle + sprint_speed +
                 stamina + standing_tackle + strength + vision + volleys, data = train)

gam.m4 = gam(transformed_value ~ s(age) + nationality + s(potential) + s(special) + acceleration
             + aggression + agility + balance + ball_control + composure + crossing + curve +
                 dribbling + finishing + s(free_kick_accuracy) + s(gk_diving) + gk_handling + gk_kickin
                 s(gk_positioning) + s(gk_reflexes) + heading_accuracy + interceptions +
                 jumping + long_passing + long_shots + marking + penalties + positioning +
                 s(reactions) + short_passing + shot_power + sliding_tackle + sprint_speed +
                 stamina + standing_tackle + strength + vision + volleys, data = train)

anova(gam.m1, gam.m2, gam.m3, gam.m4, test = "F")
```