Nonlinear Methods

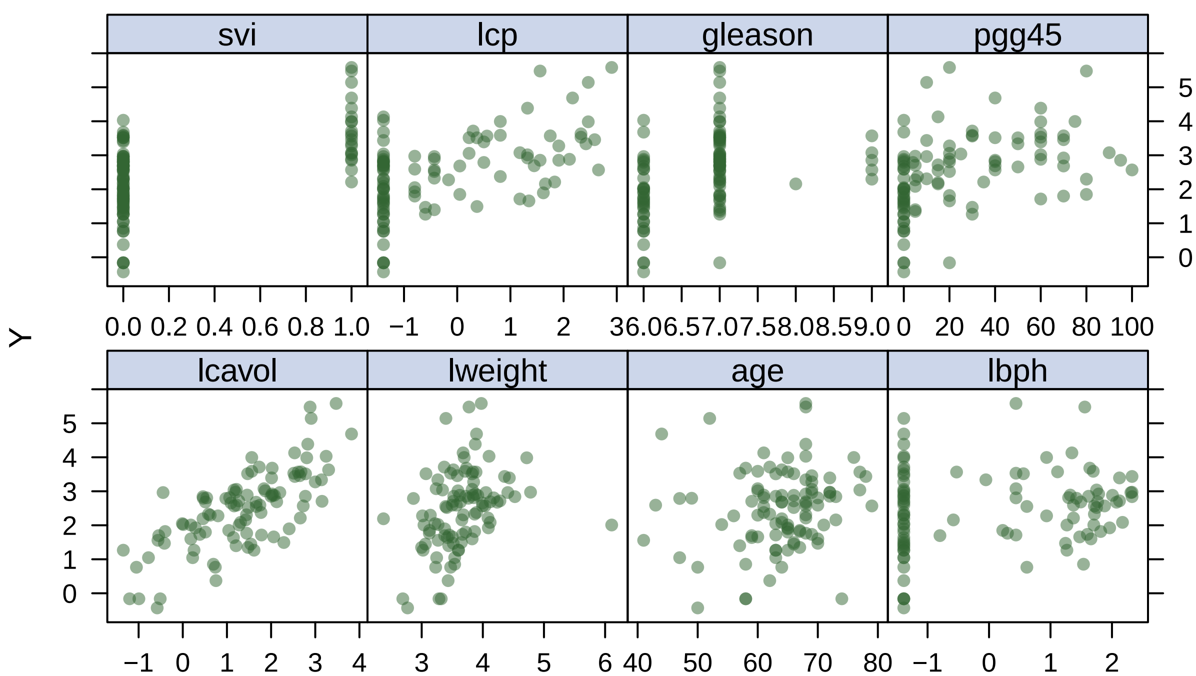
```r
library(caret)
library(splines)
library(lasso2)
library(mgcv)
library(ggplot2)
library(pdp)
library(earth)
```

We will use a prostate cancer dataset for illustration. The data come from a study that examined the correlation between the level of prostate specific antigen (PSA) and a number of clinical measures in men who were about to receive a radical prostatectomy. The dataset can be found in the package `lasso2`. The response is the log PSA level (`lpsa`).

```r
data(Prostate)
x <- model.matrix(lpsa~.,Prostate)[,-1] # matrix of predictors
y <- Prostate$lpsa # vector of response
```

We use scatterplot to explore the relationship between the log PSA level and other variables. The variable percentage Gleason score 4/5 (`pgg45`) shows some potentially nonlinear trend.

```r
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, y, plot = "scatter", labels = c("","Y"),
            type = c("p"), layout = c(4, 2))
```



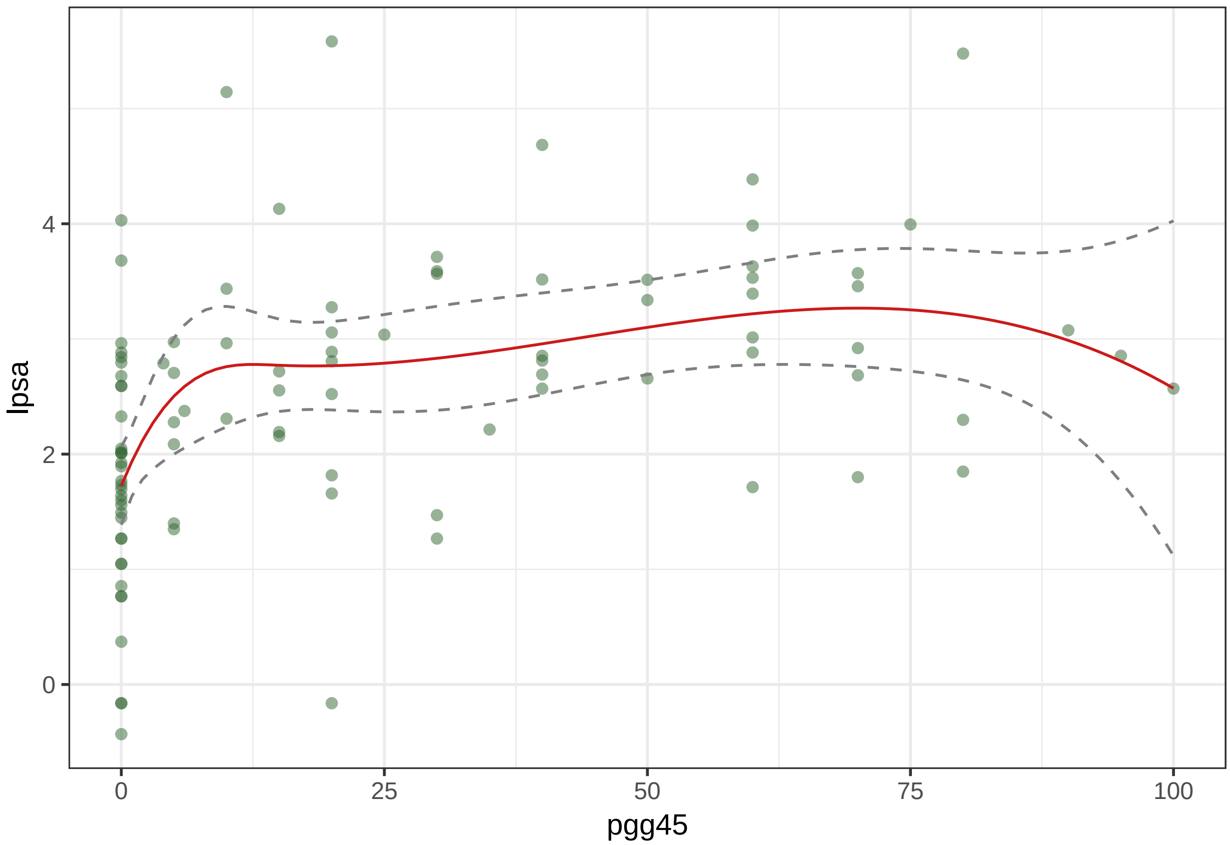
In what follows, we fit univariate nonlinear models and GAM to investage the association between `lpsa` and pgg45.

Splines

Cubic splines

We fit a cubic spline model. Degree of freedom `df` (or knots `knots`) need to be specified. The argument `degree` denotes the degree of the piecewise polynomial; default is 3 for cubic splines.

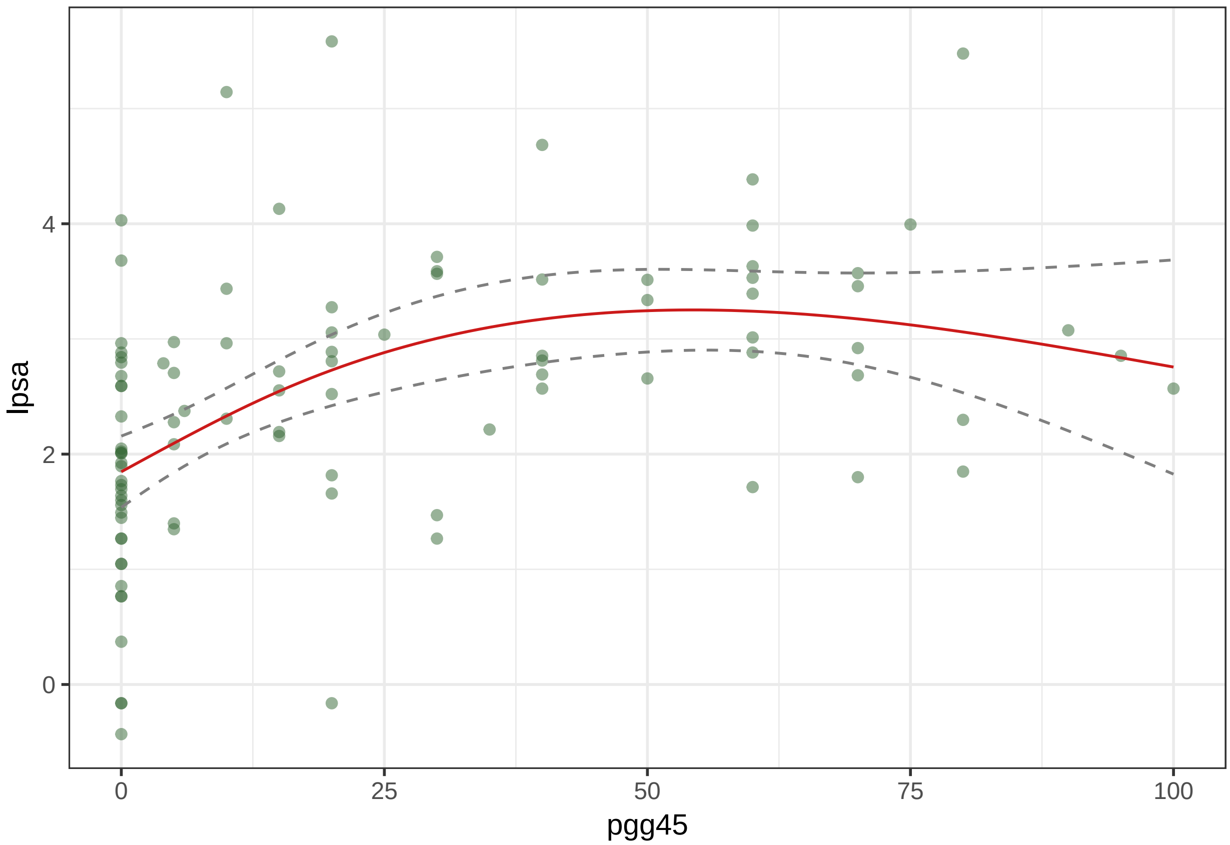
```
fit.bs <- lm(lpsa~bs(pgg45, df = 4), data = Prostate)
# fit.bs <- lm(lpsa~bs(pgg45, df = 5, intercept = TRUE)-1, data = Prostate)

pgg45lims <- range(Prostate$pgg45)
pgg45.grid <- seq(from = pgg45lims[1],to = pgg45lims[2])

pred.bs <- predict(fit.bs,
                   newdata = list(pgg45=pgg45.grid),
                   se = TRUE)

pred.bs.df <- data.frame(pred = pred.bs$fit,
                         pgg45 = pgg45.grid,
                         upper = pred.bs$fit+2*pred.bs$se,
                         lower = pred.bs$fit-2*pred.bs$se)

p <- ggplot(data = Prostate, aes(x = pgg45, y = lpsa)) +
     geom_point(color = rgb(.2, .4, .2, .5))

p + geom_line(aes(x = pgg45, y = pred), data = pred.bs.df,
              color = rgb(.8, .1, .1, 1)) +
    geom_line(aes(x = pgg45, y = upper), data = pred.bs.df,
              linetype = 2, col = "grey50") +
    geom_line(aes(x = pgg45, y = lower), data = pred.bs.df,
              linetype = 2, col = "grey50") + theme_bw()
```



Natural cubic splines

We then fit a natural cubic spline model that extraplate linearly beyond the boundary knots.

```
fit.ns <- lm(lpsa~ns(pgg45, df = 2), data = Prostate)

pred.ns <- predict(fit.ns,
                   newdata = list(pgg45=pgg45.grid),
                   se=TRUE)

pred.ns.df <- data.frame(pred = pred.ns$fit,
                         pgg45 = pgg45.grid,
                         upper = pred.ns$fit+2*pred.ns$se,
                         lower = pred.ns$fit-2*pred.ns$se)


p + geom_line(aes(x = pgg45, y = pred), data = pred.ns.df,
              color = rgb(.8, .1, .1, 1)) +
    geom_line(aes(x = pgg45, y = upper), data = pred.ns.df,
              linetype = 2, col = "grey50") +
    geom_line(aes(x = pgg45, y = lower), data = pred.ns.df,
              linetype = 2, col = "grey50") + theme_bw()
```



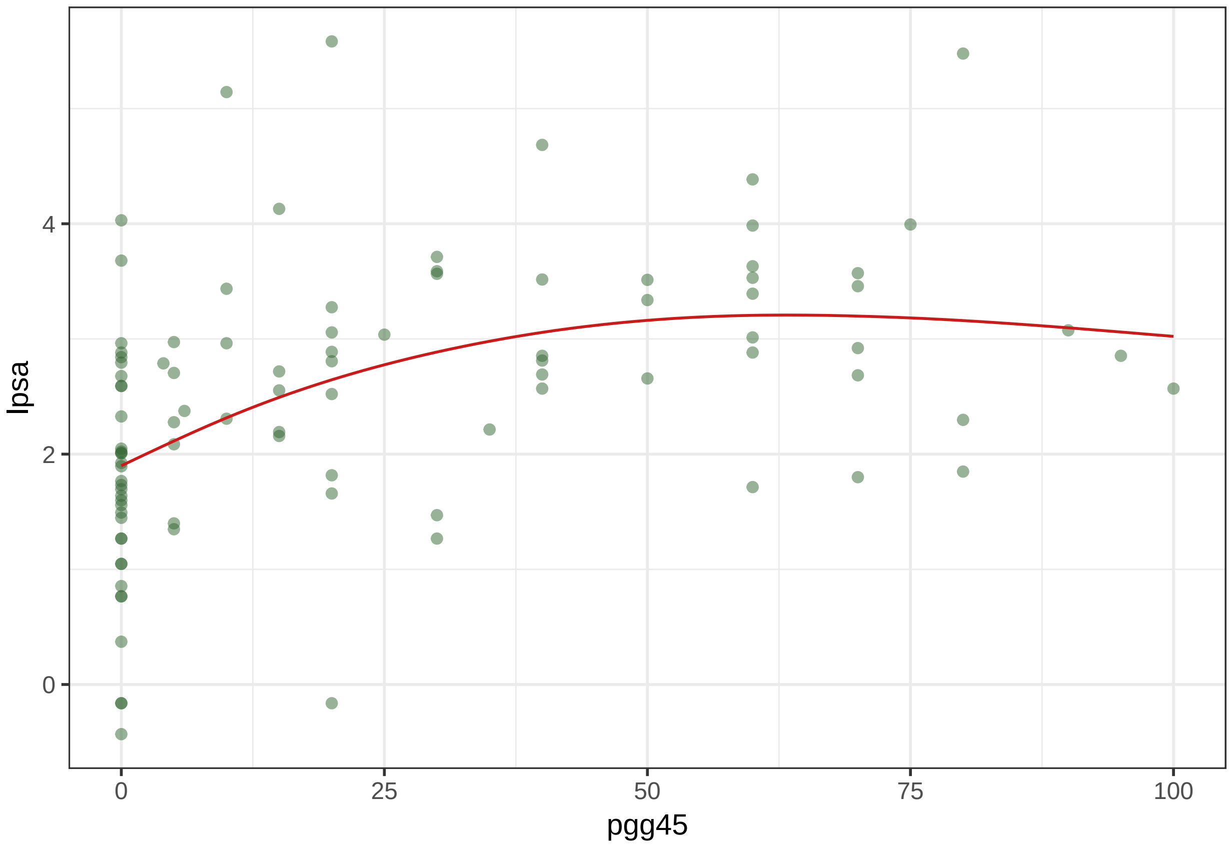
Smoothing splines

The function `smooth.spline()` can be used to fit smoothing spline models. Generalized cross-validation is used to select the degree of freedom (trace of the smoother matrix).

```
fit.ss <- smooth.spline(Prostate$pgg45, Prostate$lpsa)
fit.ss$df
```

```
## [1] 3.24361
```

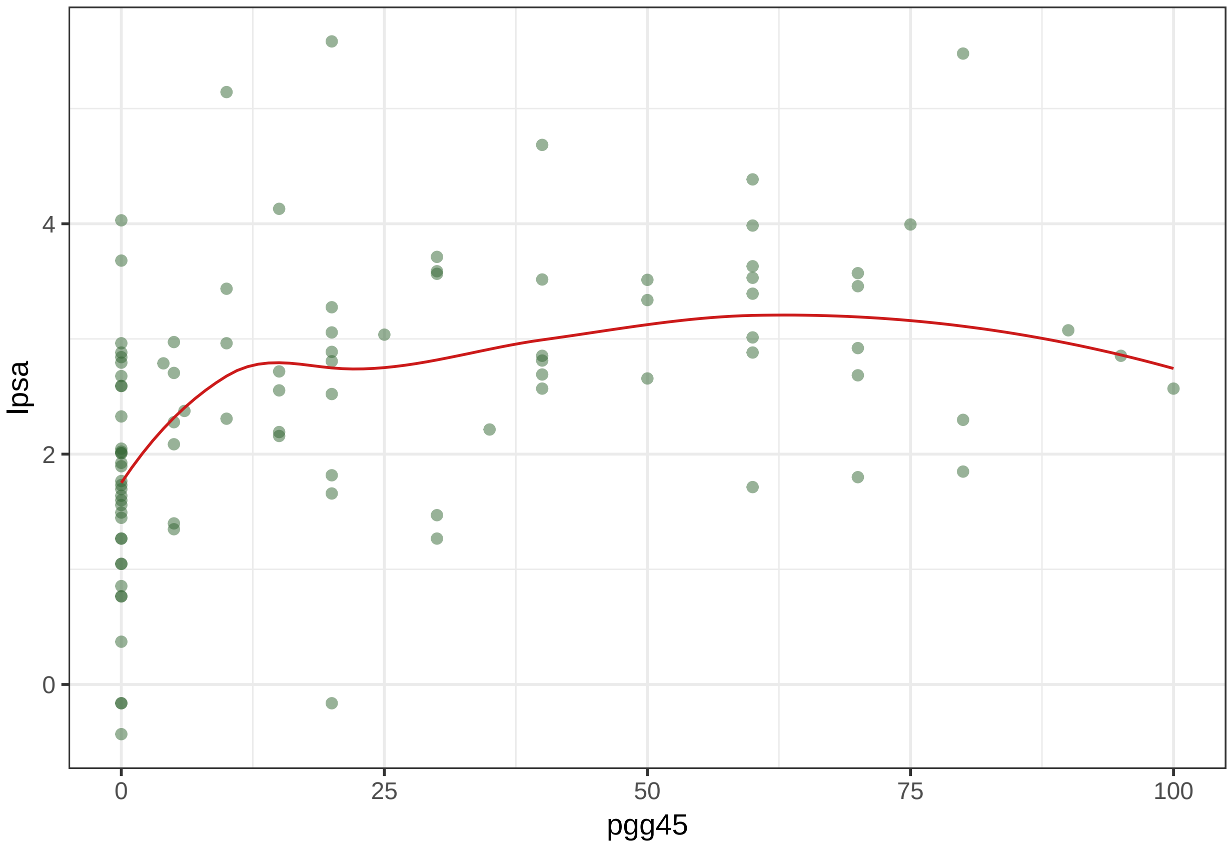
```
pred.ss <- predict(fit.ss,
                   x = pgg45.grid)

pred.ss.df <- data.frame(pred = pred.ss$y,
                         pgg45 = pgg45.grid)

p +
geom_line(aes(x = pgg45, y = pred), data = pred.ss.df,
          color = rgb(.8, .1, .1, 1)) + theme_bw()
```



Local regression

We perform a local linear regression using `loess()`.

```
fit.loess <- loess(lpsa~pgg45, data = Prostate)

pred.loess <- predict(fit.loess,
                      newdata = data.frame(pgg45 = pgg45.grid))

pred.loess.df <- data.frame(pred = pred.loess,
                            pgg45 = pgg45.grid)

p + geom_line(aes(x = pgg45, y = pred), data = pred.loess.df,
              color = rgb(.8, .1, .1, 1)) + theme_bw()
```



GAM

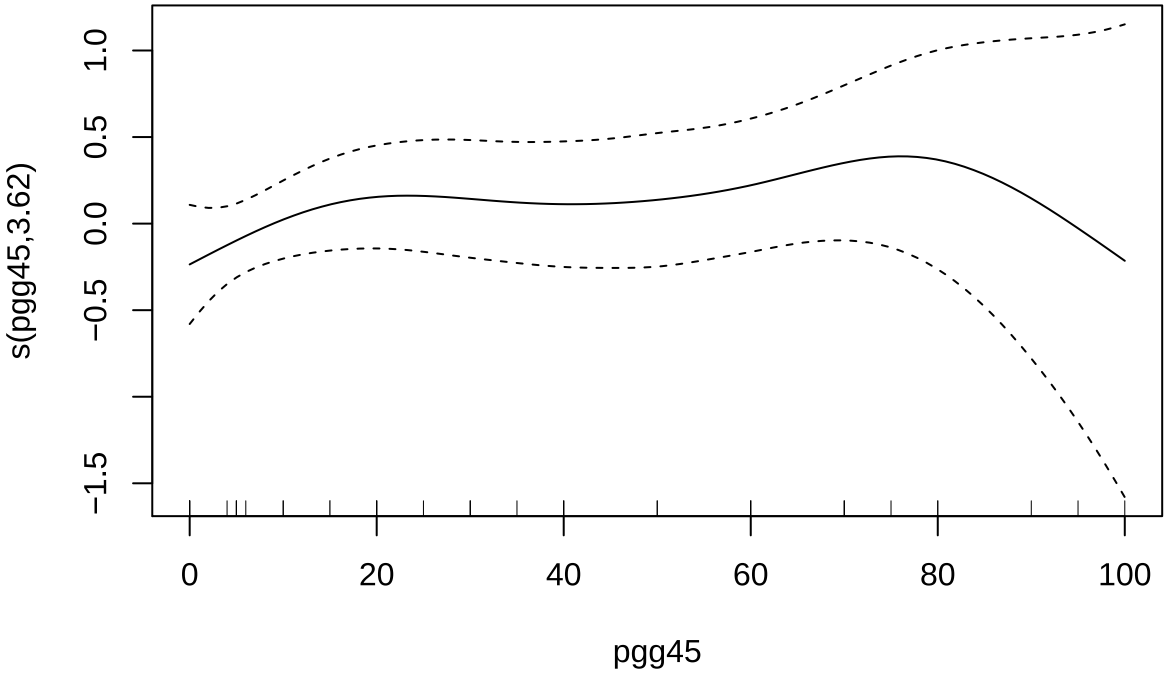
`gam()` fits a generalized additive model (GAM) to data, the term 'GAM' being taken to include any quadratically penalized GLM and a variety of other models estimated by a quadratically penalised likelihood type approach. In `gam()`, built-in nonparametric smoothing terms are indicated by `s` for smoothing splines. The package `gam` also provides a function `gam()`. GCV is used to select the degree of freedom. Confidence/credible intervals are readily available for any quantity predicted using a fitted model.

```
gam.m1 <- gam(lpsa~age+pgg45+lcavol+lweight+lbph+svi+lcp+gleason, data = Prostate)
gam.m2 <- gam(lpsa~age+s(pgg45)+lcavol+lweight+lbph+svi+lcp+gleason,
              data = Prostate)
gam.m3 <- gam(lpsa~age+s(pgg45)+te(lcavol,lweight)+lbph+svi+lcp+gleason,
              data = Prostate)

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

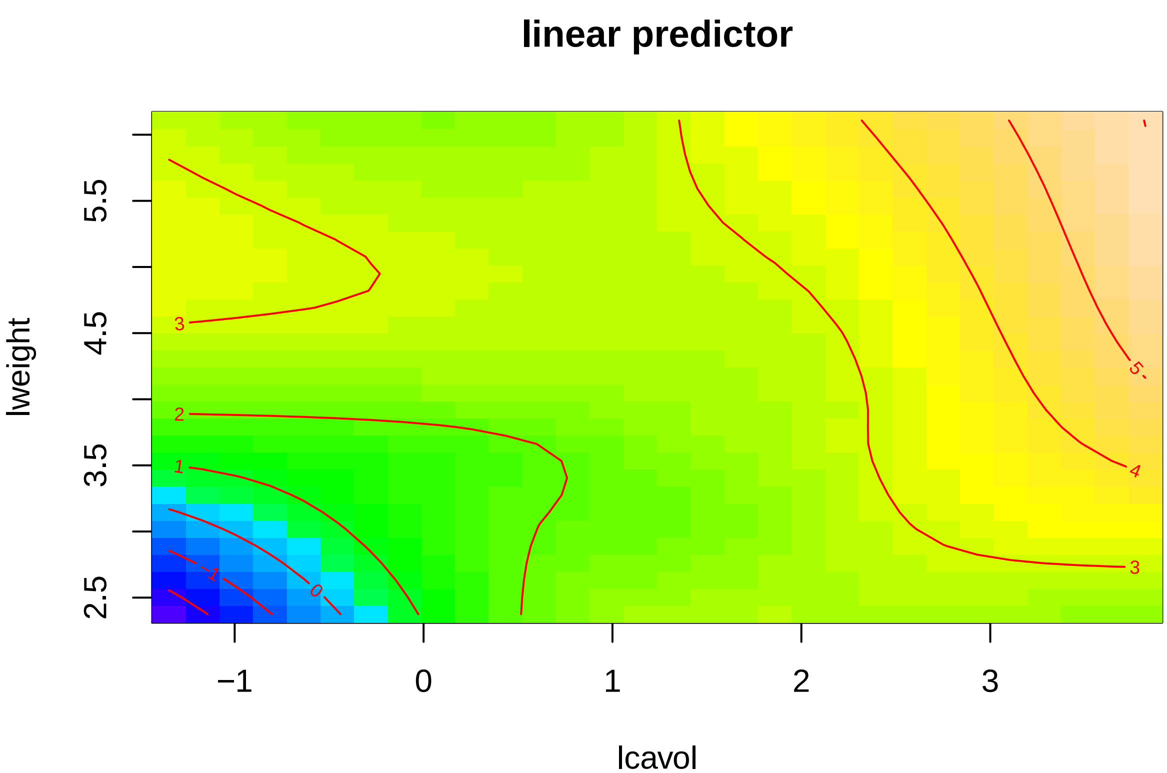
```
## Analysis of Deviance Table
##
## Model 1: lpsa ~ age + pgg45 + lcavol + lweight + lbph + svi + lcp + gleason
## Model 2: lpsa ~ age + s(pgg45) + lcavol + lweight + lbph + svi + lcp +
##     gleason
## Model 3: lpsa ~ age + s(pgg45) + te(lcavol, lweight) + lbph + svi + lcp +
##     gleason
##   Resid. Df Resid. Dev      Df Deviance      F  Pr(>F)
## 1    88.000     44.163
## 2    84.485     41.132  3.5154   3.0312 2.0734 0.10120
```

```
## 3    73.739     31.975 10.7461   9.1569 2.0489 0.03636 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
plot(gam.m2)
```




```r
vis.gam(gam.m3, view = c("lcavol","lweight"),
        plot.type = "contour", color = "topo")
```



With the current support from `caret`, you may lose a significant amount of flexibility in `mgcv`.

```
ctrl1 <- trainControl(method = "cv", number = 10)
# you can try other options

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

gam.fit$bestTune
```

```
##   select method
## 2   TRUE GCV.Cp
```

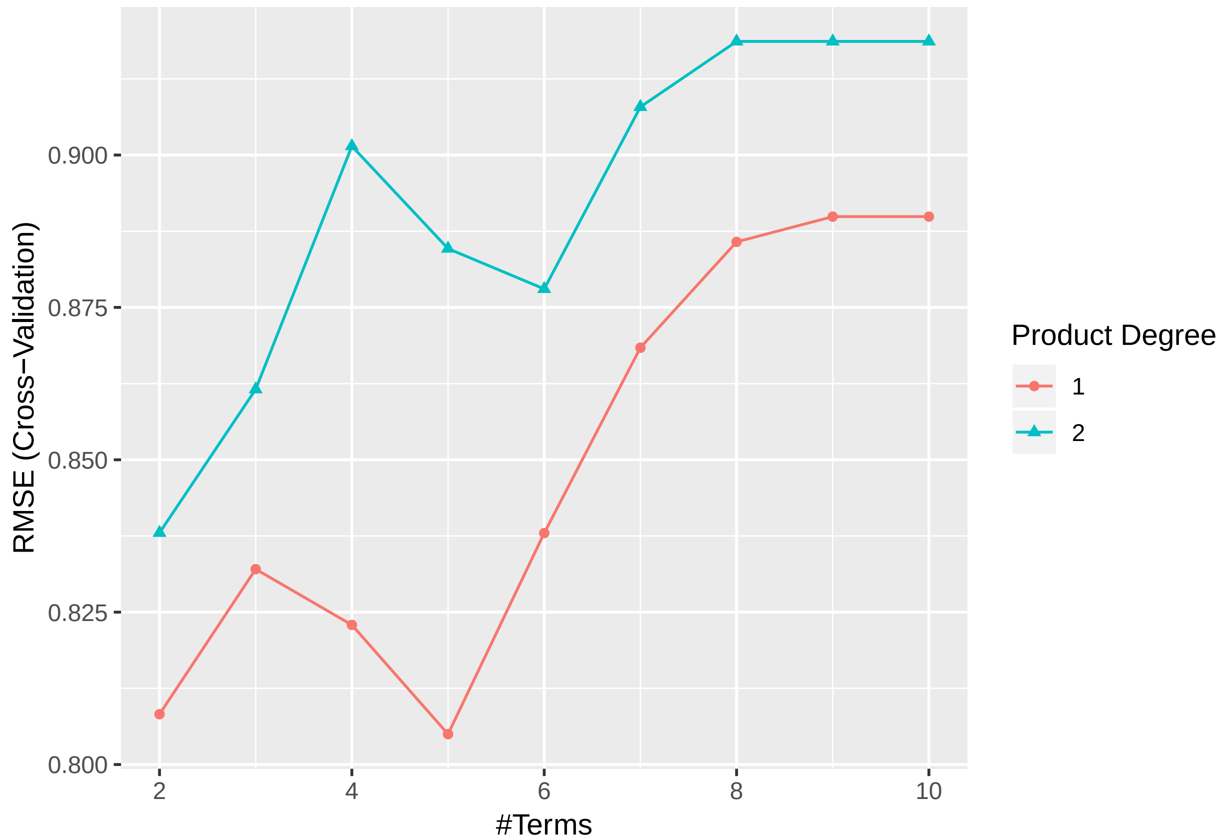
```
gam.fit$finalModel
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ svi + gleason + s(pgg45) + s(lcp) + s(age) + s(lbph) +
##     s(lweight) + s(lcavol)
##
## Estimated degrees of freedom:
## 3.651 0.000 1.470 0.716 1.520 4.582  total = 14.94
##
## GCV score: 0.5357211
```

Multivariate Adaptive Regression Splines (MARS)

We next create a piecewise linear model using multivariate adaptive regression splines (MARS). Since there are two tuning parameters associated with the MARS model: the degree of interactions and the number of retained terms, we need to perform a grid search to identify the optimal combination of these hyperparameters that minimize prediction error

```
mars_grid <- expand.grid(degree = 1:2,
                         nprune = 2:10)

set.seed(2)
mars.fit <- train(x, y,
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl1)

ggplot(mars.fit)
```




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

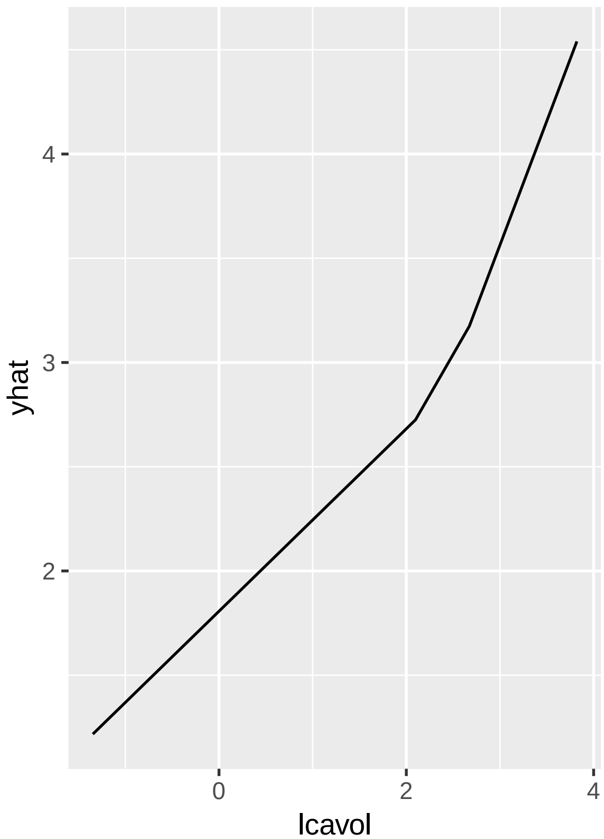
```
##   nprune degree
## 4      5      1
```

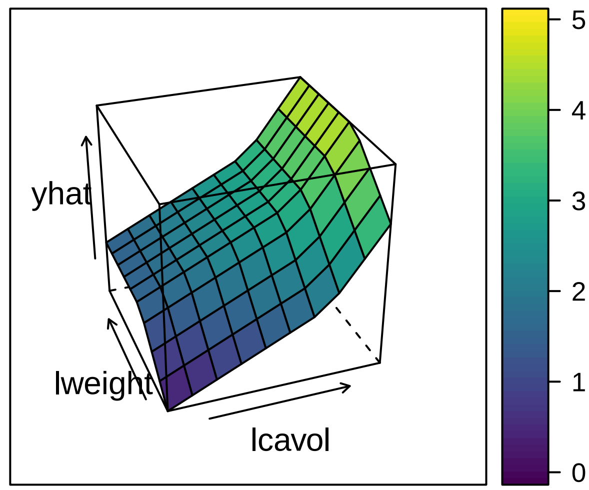
```r
coef(mars.fit$finalModel)
```

```
##        (Intercept)  h(lcavol-2.40964)  h(2.40964-lcavol)
##         3.31668457         1.18965538        -0.43756141
## h(3.83622-lweight)        h(10-pgg45)
##        -0.88094773        -0.04983056
```

To better understand the relationship between these features and `lpsa`, we can create partial dependence plots (PDPs) for each feature individually and also an interaction PDP. This is used to examine the marginal effects of predictors.

```r
p1 <- partial(mars.fit, pred.var = c("lcavol"), grid.resolution = 10) %>% autoplot()

p2 <- partial(mars.fit, pred.var = c("lcavol", "lweight"), grid.resolution = 10) %>%
      plotPartial(levelplot = FALSE, zlab = "yhat", drape = TRUE,
                  screen = list(z = 20, x = -60))

grid.arrange(p1, p2, ncol = 2)
```






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