EAS508-HW4

Saish Mandavkar

2022-10-20

Lab Code Homework

5.3 Cross Validation Labs

5.3.1 Validation Set Approach

```
# Setting the seed and loading the data

library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.0.5
```

```
set.seed(1)
train <- sample(392,196)
```

```
# Fitting a linear regression on the train data using subset option

lm.fit <- lm(mpg ~ horsepower, data = Auto, subset = train)
```

```
# Predicting the estimates for the 392 observations and calculate the MSE for 192 observations

mean((Auto$mpg - predict(lm.fit, Auto))[-train]^2)
```

```
## [1] 23.26601
```

```
# Fitting cubic regression and calculating the MSE

lm.fit2 <- lm(mpg ~poly(horsepower, 2), data = Auto, subset = train)

mean((Auto$mpg - predict(lm.fit2, Auto))[-train]^2)
```

```
## [1] 18.71646
```

```
# Fitting uadratic regression and calculating the MSE

lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto, subset = train)

mean((Auto$mpg - predict(lm.fit3, Auto))[-train]^2)
```

```
## [1] 18.79401
```

```r
# Using different seed and calculatiing the values for all the three regressions - will result into dif

set.seed(2)
train <- sample(392,196)

# Linear regression MSE

lm.fit <- lm(mpg ~ horsepower, data = Auto, subset = train)

mean((Auto$mpg - predict(lm.fit, Auto))[-train]^2)
```

```
## [1] 25.72651
```

```r
# Cubic regression MSE

lm.fit2 <- lm(mpg ~poly(horsepower, 2), data = Auto, subset = train)

mean((Auto$mpg - predict(lm.fit2, Auto))[-train]^2)
```

```
## [1] 20.43036
```

```r
# Quadratic regression MSE


lm.fit3 <- lm(mpg ~poly(horsepower, 3), data = Auto, subset = train)

mean((Auto$mpg - predict(lm.fit3, Auto))[-train]^2)
```

```
## [1] 20.38533
```

5.3.2 Leave One-Out Cross-Validation

```r
# LOOCV using glm() package

glm.fit <- glm(mpg ~ horsepower, data = Auto)

coef(glm.fit)
```

```
## (Intercept)  horsepower
##  39.9358610  -0.1578447
```

```r
# LOOCV using normal lm() function

lm.fit <- lm(mpg ~ horsepower, data = Auto)

coef(lm.fit)
```

```
## (Intercept)  horsepower
##  39.9358610  -0.1578447
```

```
# Cross-validation error using glm() package

library(boot)
```

```
## Warning: package 'boot' was built under R version 4.0.5
```

```
glm.fit <- glm(mpg ~ horsepower, data = Auto)

cv.err <- cv.glm(Auto, glm.fit)

cv.err$delta
```

```
## [1] 24.23151 24.23114
```

```
# Calculating CV error for for polynomial of order 1 to 10 using a for loop.

cv.error <- rep(0,10)

for (i in  1:10) {

  glm.fit <- glm(mpg ~ poly(horsepower, i), data = Auto)
  cv.error[i] <- cv.glm(Auto, glm.fit)$delta[1]

}

cv.error
```

```
##  [1] 24.23151 19.24821 19.33498 19.42443 19.03321 18.97864 18.83305 18.96115
##  [9] 19.06863 19.49093
```

5.3.3 k-Fold Cross Validation

```
# Calculating k-fold CV error for for polynomial of order 1 to 10 with k = 10

set.seed(17)
cv.error.10 <- rep(0,10)

for (i in  1:10) {

  glm.fit <- glm(mpg ~ poly(horsepower, i), data = Auto)
  cv.error.10[i] <- cv.glm(Auto, glm.fit, K = 10)$delta[1]

}

cv.error.10
```

```
##  [1] 24.27207 19.26909 19.34805 19.29496 19.03198 18.89781 19.12061 19.14666
##  [9] 18.87013 20.95520
```

6.5.3 PCR and PLS Regression

```r
# Creating model matrix for x and storing all salary values in y

# Omitting NA values

Hitters <- na.omit(Hitters)

x <- model.matrix(Salary ~ ., Hitters)[, -1]
y <- Hitters$Salary

# Creating train and test by setting the R seed

set.seed(1)
train <- sample(1:nrow(x), nrow(x) / 2)
test <- (-train)
y.test <- y[test]
```

```r
# Applying PCR to Hitters data to predcit Salary

library(pls)
```

Principal Components Regression

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

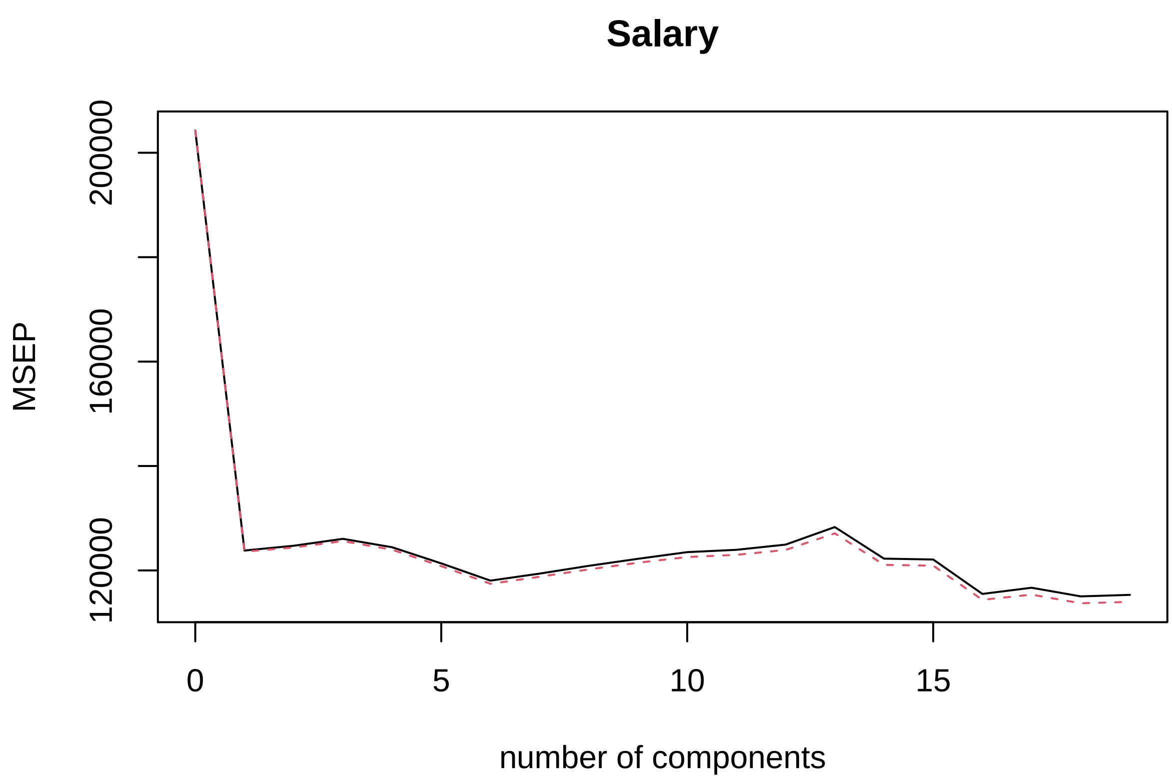
```r
set.seed(2)
pcr.fit <- pcr(Salary ~., data = Hitters, scale = TRUE, validation = "CV")
```

```r
# Checking summary of our fit

summary(pcr.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV             452    351.9    353.2    355.0    352.8    348.4    343.6
## adjCV          452    351.6    352.7    354.4    352.1    347.6    342.7
```

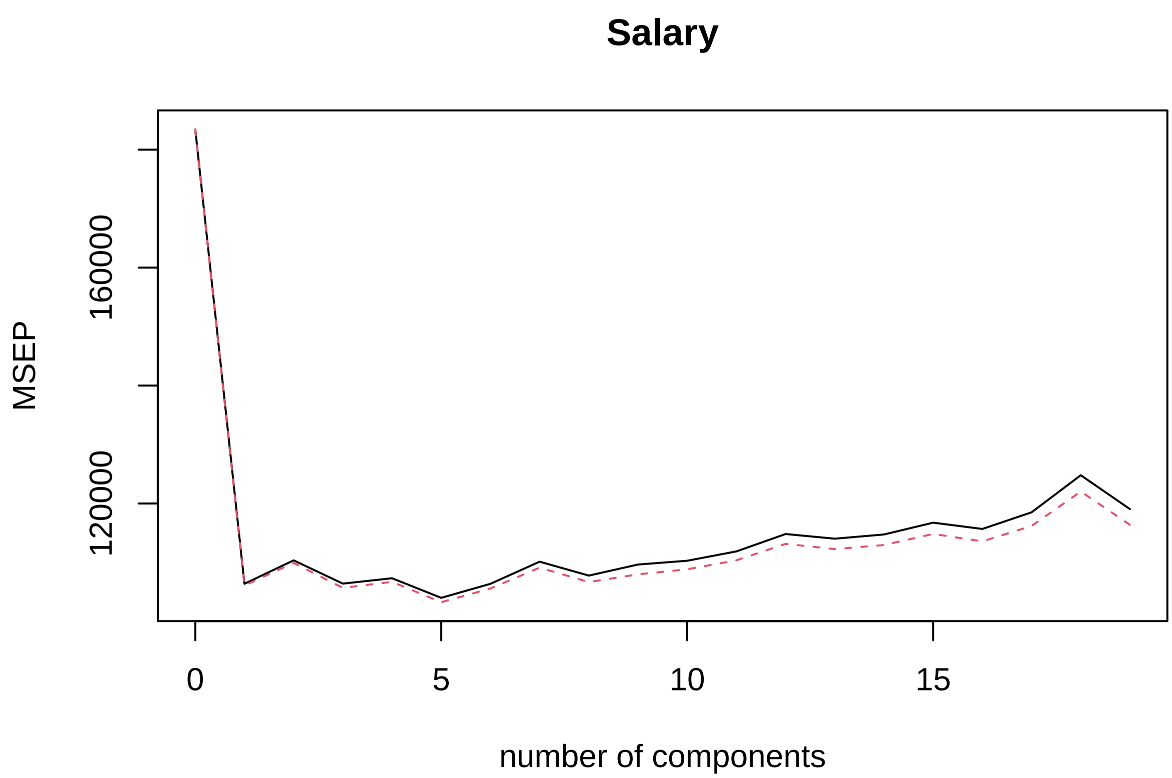
```
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       345.5    347.7    349.6     351.4     352.1     353.5     358.2
## adjCV    344.7    346.7    348.5     350.1     350.7     352.0     356.5
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        349.7     349.4     339.9     341.6     339.2     339.6
## adjCV     348.0     347.7     338.2     339.7     337.2     337.6
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         38.31    60.16    70.84    79.03    84.29    88.63    92.26    94.96
## Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69    46.75
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         96.28     97.26     97.98     98.65     99.15     99.47     99.75
## Salary    46.86     47.76     47.82     47.85     48.10     50.40     50.55
##         16 comps  17 comps  18 comps  19 comps
## X          99.89     99.97     99.99    100.00
## Salary     53.01     53.85     54.61     54.61
```

```r
# Plotting cross-validation MSE

validationplot(pcr.fit, val.type = "MSEP")
```




```r
# Performing PCR on the training data using subset function and plotting the CV MSE

set.seed(1)
```

```r
pcr.fit <- pcr(Salary ~., data = Hitters, subset = train, scale = TRUE, validation = "CV")

validationplot(pcr.fit, val.type = "MSEP")
```




```r
# Find the lowest CV error when M = 5


pcr.pred <- predict(pcr.fit, x[test, ], ncomp = 5)

mean((pcr.pred - y.test)^2)
```

```
## [1] 142811.8
```

```r
# Fit PCR on complete dataset using M = 5 identified by CV

pcr.fit <- pcr(y~x, scale = TRUE, ncomp = 5)

summary(pcr.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 5
```

```
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps
## X    38.31    60.16    70.84    79.03    84.29
## y    40.63    41.58    42.17    43.22    44.90
```

```r
# Implement PLS using plsr() function

set.seed(1)
pls.fit <- plsr(Salary ~ ., data = Hitters, subset = train, scale = TRUE, validation = "CV")

summary(pls.fit)
```

Partial Least Squares

```
## Data:    X dimension: 131 19
##  Y dimension: 131 1
## Fit method: kernelpls
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           428.3    325.5    329.9    328.8    339.0    338.9    340.1
## adjCV        428.3    325.0    328.2    327.2    336.6    336.1    336.6
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       339.0    347.1    346.4     343.4     341.5     345.4     356.4
## adjCV    336.2    343.4    342.8     340.2     338.3     341.8     351.1
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        348.4     349.1     350.0     344.2     344.5     345.0
## adjCV     344.2     345.0     345.9     340.4     340.6     341.1
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         39.13    48.80    60.09    75.07    78.58    81.12    88.21    90.71
## Salary    46.36    50.72    52.23    53.03    54.07    54.77    55.05    55.66
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         93.17     96.05     97.08     97.61     97.97     98.70     99.12
## Salary    55.95     56.12     56.47     56.68     57.37     57.76     58.08
##         16 comps  17 comps  18 comps  19 comps
## X          99.61     99.70     99.95    100.00
## Salary     58.17     58.49     58.56     58.62
```

```r
# Evaluating the coresponding test set MSE

pls.pred <- predict(pls.fit, x[test, ], ncomp = 1)

mean((pls.pred - y.test)^2)
```

```
## [1] 151995.3
```

```r
# Fitting PLS on complete dataset when M = 1

pls.fit <- plsr(Salary ~ ., data = Hitters, scale = TRUE, ncomp = 1)

summary(pls.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: kernelpls
## Number of components considered: 1
## TRAINING: % variance explained
##         1 comps
## X         38.08
## Salary    43.05
```

7.8.3 GAMs

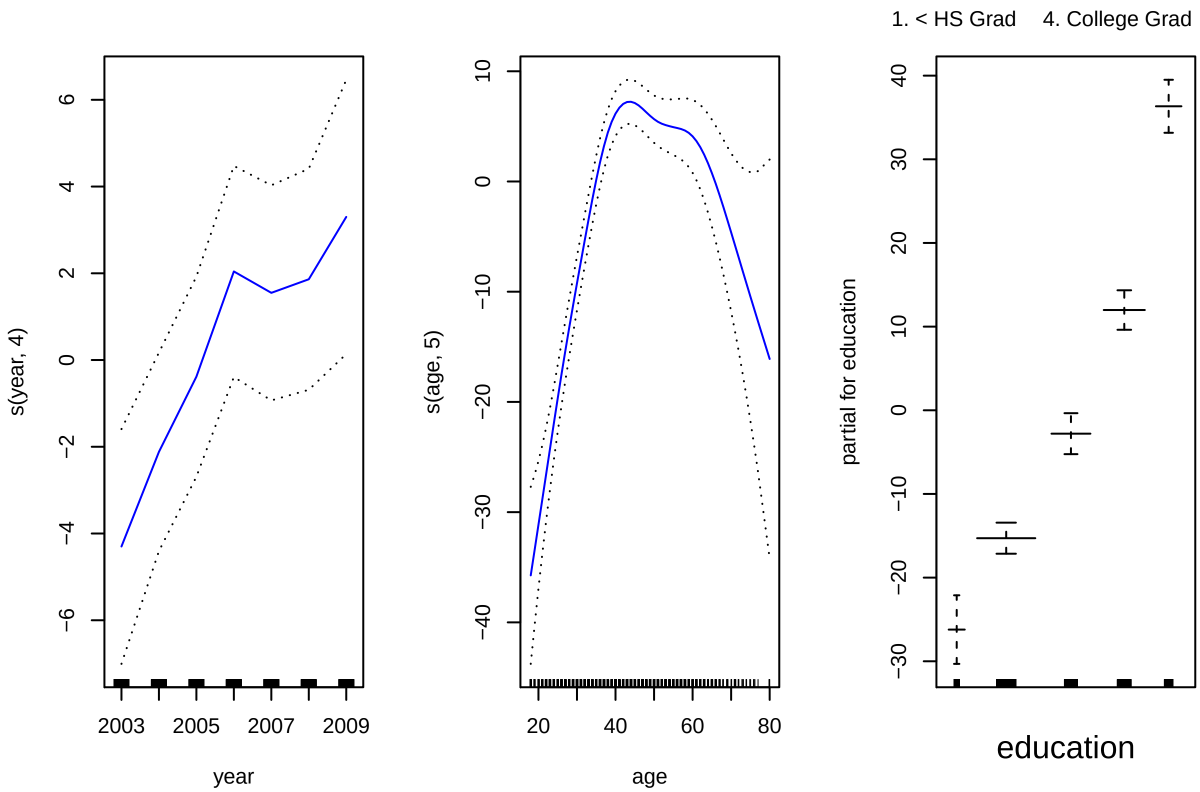
```r
# Fit a GAM or predict wage using natural spline functions of years and age.

gam1 <- lm(wage ~ splines::ns(year, 4) + splines::ns(age, 5) + education, data = Wage)
```

```r
# Fit the model using smoothing splines

library(gam)
```

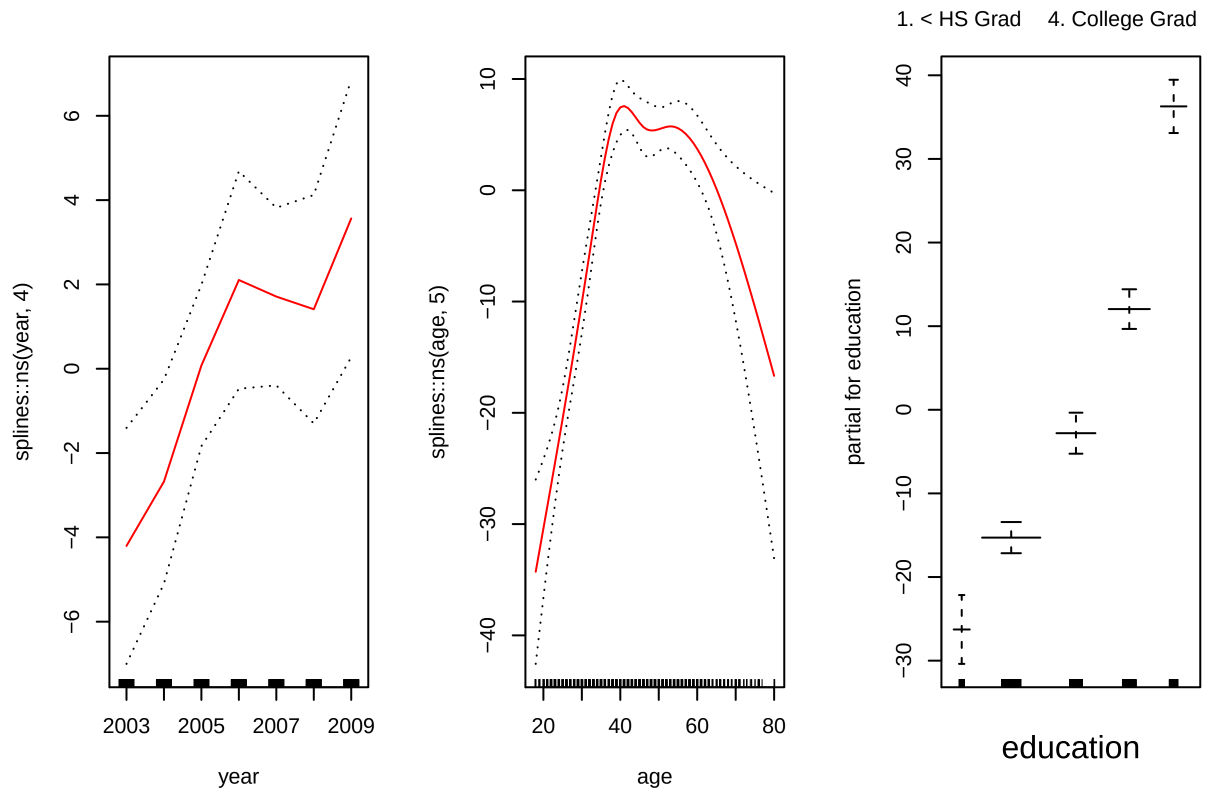
```
## Warning: package 'gam' was built under R version 4.0.5

## Loading required package: splines

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 4.0.5

## Loaded gam 1.20.1
```

```r
gam.m3 <- gam(wage ~ s(year, 4) + s(age, 5) + education, data = Wage)
```

```r
# Plot the model

par(mfrow = c(1,3))
plot(gam.m3, se = TRUE, col = "blue")
```




```r
# Plotting the GAM created using lm

par(mfrow = c(1,3))
plot.Gam(gam1, se = TRUE, col = "red")
```




```
# Performing ANOVA test to determine the best model

gam.m1 <- gam(wage ~ s(age, 5) + education,  data = Wage)
gam.m2 <- gam(wage ~ year + s(age, 5) + education, data = Wage)

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

```
## Analysis of Deviance Table
##
## Model 1: wage ~ s(age, 5) + education
## Model 2: wage ~ year + s(age, 5) + education
## Model 3: wage ~ s(year, 4) + s(age, 5) + education
##   Resid. Df Resid. Dev Df Deviance       F    Pr(>F)
## 1      2990    3711731
## 2      2989    3693842  1  17889.2 14.4771 0.0001447 ***
## 3      2986    3689770  3   4071.1  1.0982 0.3485661
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

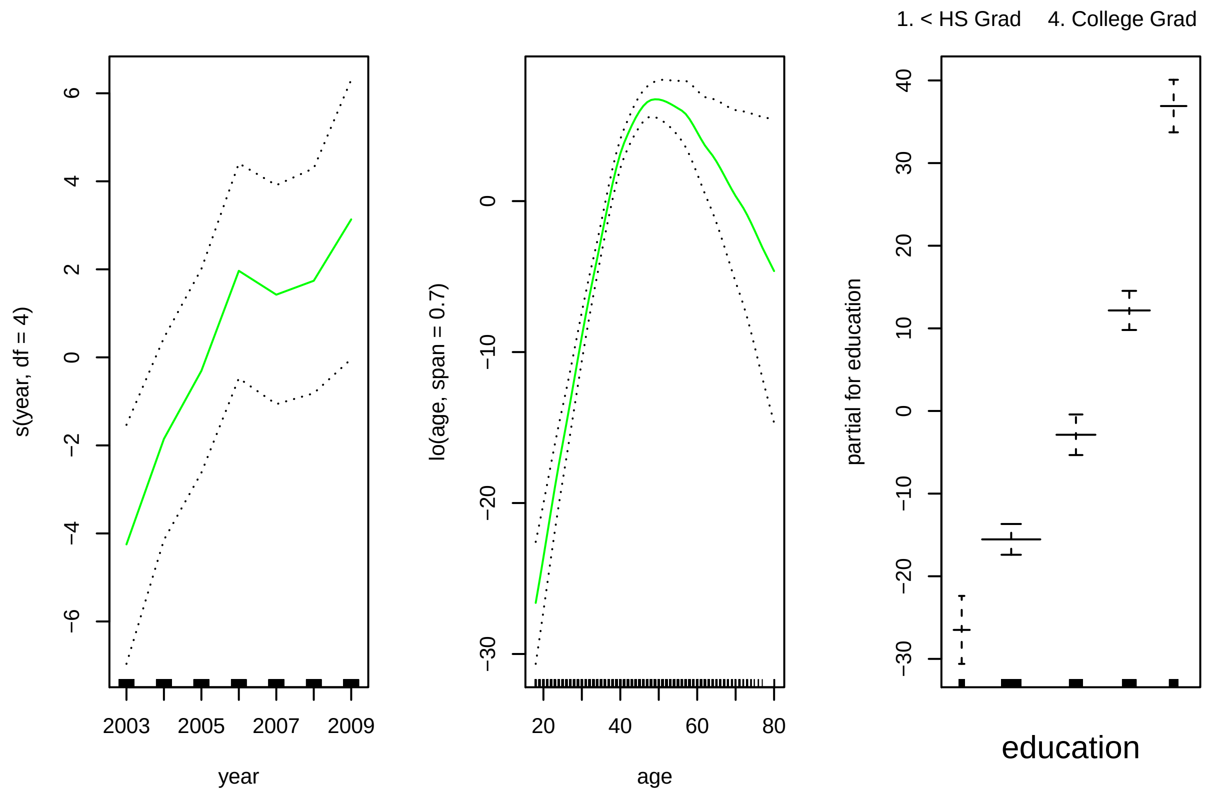
```
# summary of gam.m3

summary(gam.m3)
```

```
##
## Call: gam(formula = wage ~ s(year, 4) + s(age, 5) + education, data = Wage)
```

```
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -119.43  -19.70   -3.33   14.17  213.48
##
## (Dispersion Parameter for gaussian family taken to be 1235.69)
##
##     Null Deviance: 5222086 on 2999 degrees of freedom
## Residual Deviance: 3689770 on 2986 degrees of freedom
## AIC: 29887.75
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##               Df  Sum Sq Mean Sq F value    Pr(>F)
## s(year, 4)     1   27162   27162  21.981 2.877e-06 ***
## s(age, 5)      1  195338  195338 158.081 < 2.2e-16 ***
## education      4 1069726  267432 216.423 < 2.2e-16 ***
## Residuals   2986 3689770    1236
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##              Npar Df Npar F  Pr(F)
## (Intercept)
## s(year, 4)         3  1.086 0.3537
## s(age, 5)          4 32.380 <2e-16 ***
## education
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Using the predict method for class GAM

preds <- predict(gam.m2, newdata = Wage)
```

```r
# Using local regression fits in GAM using lo()

gam.lo <- gam(wage ~ s(year, df = 4) + lo(age, span = 0.7) + education, data = Wage)

par(mfrow = c(1,3))
plot.Gam(gam.lo, se = TRUE, col = "green")
```



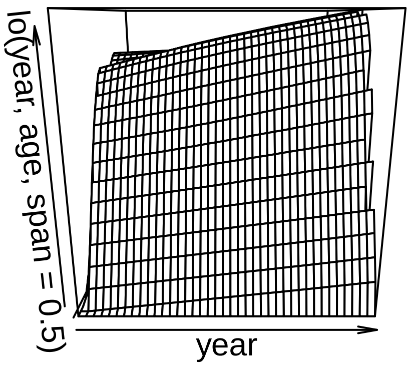

```r
# Using lo() to create interactions before calling gam

gam.lo.i <- gam(wage ~ lo(year, age, span = 0.5) + education, data = Wage)
```

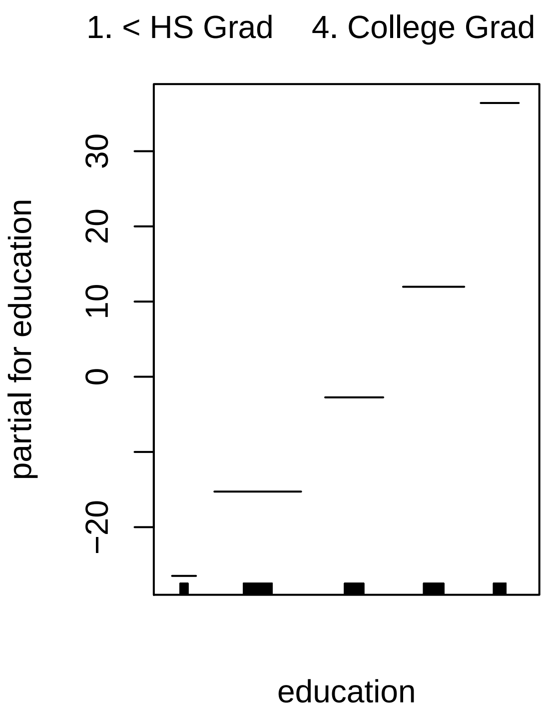
```
## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, : liv
## too small. (Discovered by lowesd)

## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, : lv
## too small. (Discovered by lowesd)

## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, : liv
## too small. (Discovered by lowesd)

## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, : lv
## too small. (Discovered by lowesd)
```

```r
# Plotting the 2D surface using akima package

library(akima)
```

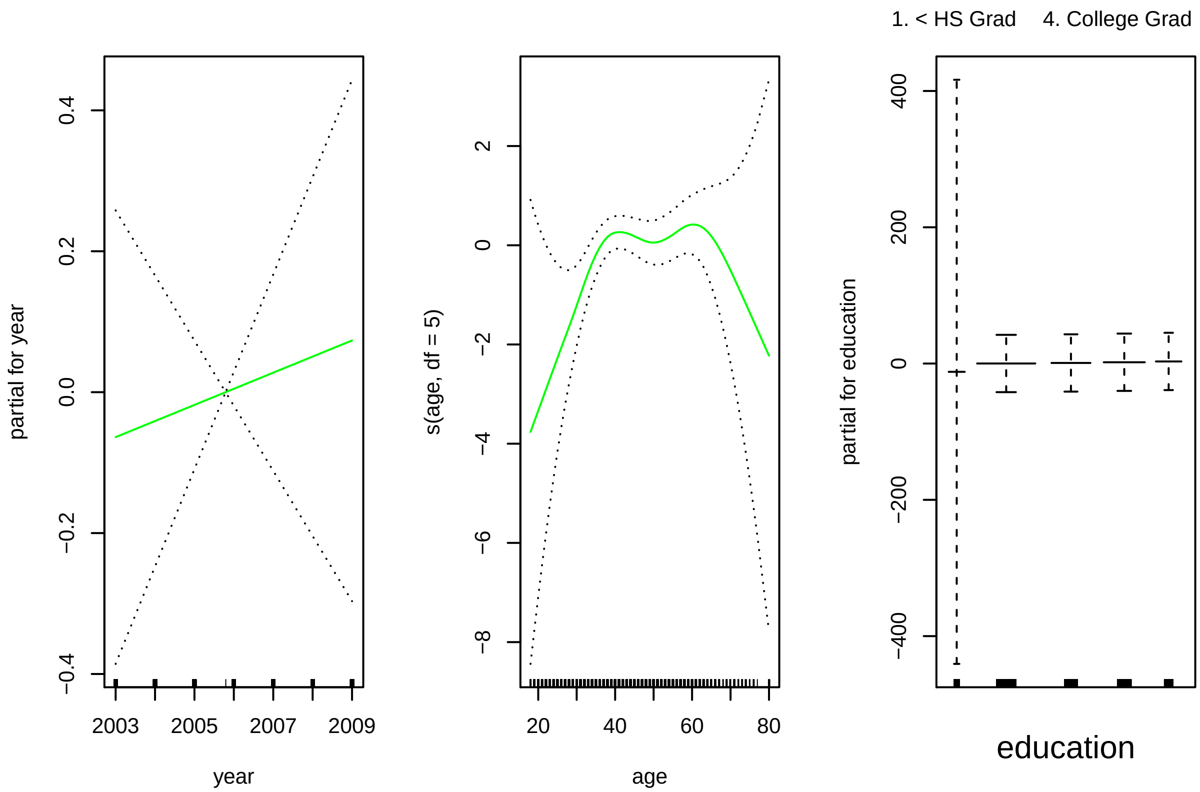
```
## Warning: package 'akima' was built under R version 4.0.5
```

```r
par(mfrow = c(1,2))
plot(gam.lo.i)
```





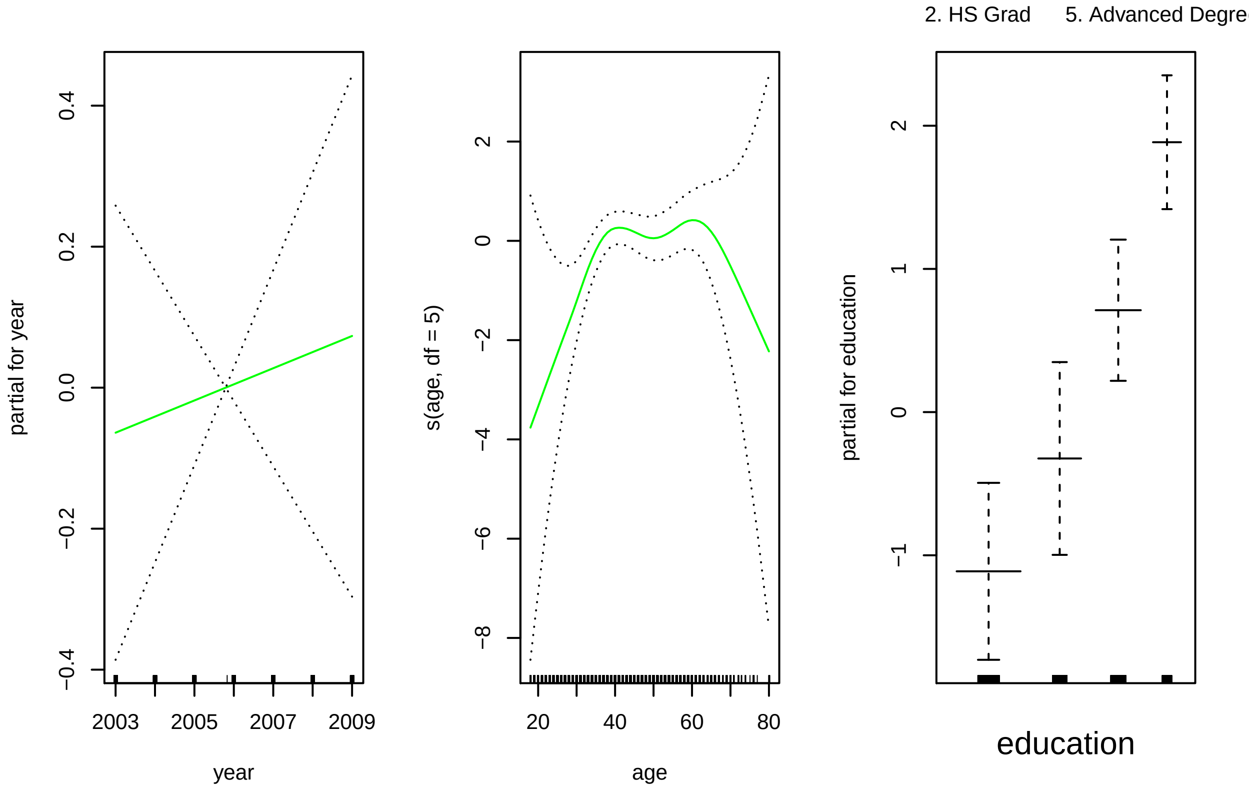

```r
# Fittinga logistic regression GAM using I() function

gam.lr <- gam(I(wage > 250) ~ year + s(age, df = 5) + education, family = binomial, data = Wage)

par(mfrow = c(1,3))
plot(gam.lr, se = TRUE, col = "green")
```




```r
# Creeate a table with high earnes in the < HS category

attach(Wage)
table(education, I(wage > 250))
```

```
##
## education             FALSE TRUE
##   1. < HS Grad          268    0
##   2. HS Grad            966    5
##   3. Some College       643    7
##   4. College Grad       663   22
##   5. Advanced Degree    381   45
```

```r
# Fitting a logsitic regression GAM by skipping the education category

gam.lr.s <- gam( I(wage > 250) ~ year + s(age, df = 5) + education, family = binomial, subset = (educati

par(mfrow = c(1,3))
plot(gam.lr.s, se = TRUE, col = "green")
```


2. HS Grad
5. Advanced Degre
partial for year
0.4
0.2
0.0
−0.2
−0.4
2003
2005
2007
2009
year
s(age, df = 5)
2
0
−2
−4
−6
−8
20
40
60
80
age
partial for education
2
1
0
−1
education