7.1.1 Generalized Additive Models

Stat 5100: Dr. Bean

Example 1: Baseball Dataset from 4.1.1

Let's see if we can improve upon the penalized linear regression model to predict the log of salary for professional (non-pitcher) baseball players. Note that answers will differ slightly depending on the random seed set.

```
# Set a random seed for reproducibility
set.seed(830578)

# Load data
library(stat5100)
library(mgcv)

## Loading required package:  nlme
## This is mgcv 1.8-33.  For overview type 'help("mgcv-package")'.

data(baseball)

baseball_gam_all <-
  mgcv::gam(logSalary ~ s(nAtBat) + s(nHits) + s(nHome) +
              s(nRuns) + s(nRBI) + s(nBB) + s(YrMajor) +
              s(CrAtBat) + s(CrHits) + s(CrHome) + s(CrRuns) +
              s(CrRbi) + s(CrBB) + s(nOuts) + s(nAssts) +
              s(nError) + League + Division,
                          data = baseball)

summary(baseball_gam_all)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## logSalary ~ s(nAtBat) + s(nHits) + s(nHome) + s(nRuns) + s(nRBI) +
##     s(nBB) + s(YrMajor) + s(CrAtBat) + s(CrHits) + s(CrHome) +
##     s(CrRuns) + s(CrRbi) + s(CrBB) + s(nOuts) + s(nAssts) + s(nError) +
##     League + Division
##
## Parametric coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     5.90256    0.03572 165.223   <2e-16 ***
## LeagueNational  0.07437    0.04414   1.685   0.0936 .
## DivisionWest   -0.02042    0.04346  -0.470   0.6389
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

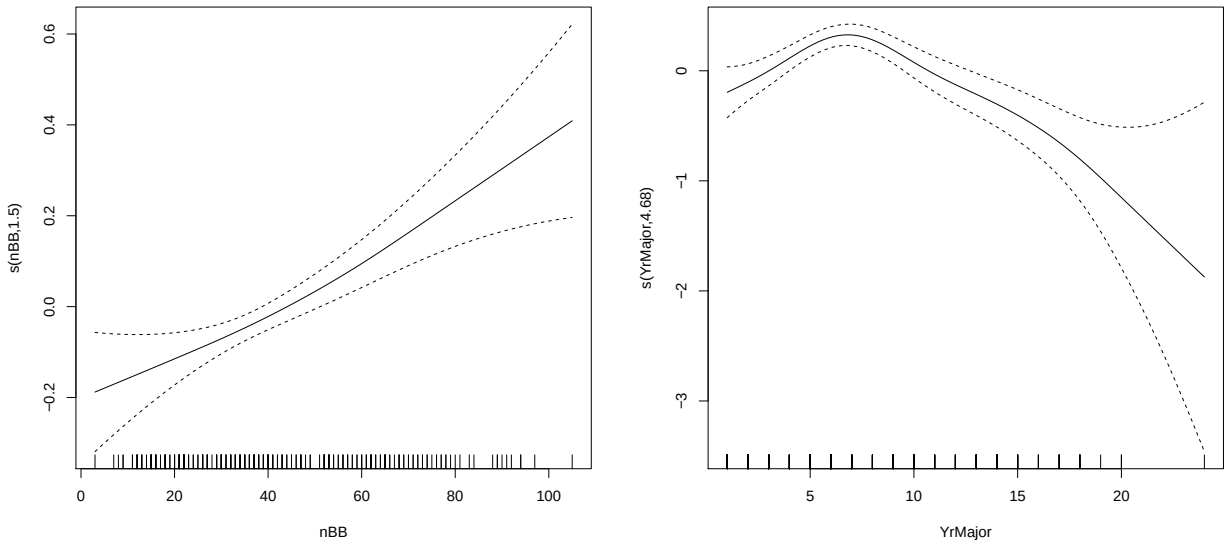
```
## Approximate significance of smooth terms:
##               edf Ref.df      F p-value
## s(nAtBat)   3.470  4.373  1.312 0.25485
## s(nHits)    1.000  1.000  0.130 0.71892
## s(nHome)    4.819  5.884  1.259 0.25922
## s(nRuns)    4.174  5.178  1.223 0.29111
## s(nRBI)     1.971  2.551  0.975 0.42890
## s(nBB)      1.245  1.445  3.885 0.04937 *
## s(YrMajor) 5.315  6.270 10.713 < 2e-16 ***
## s(CrAtBat) 7.385  8.063  2.822 0.00894 **
## s(CrHits)  3.895  4.930  2.157 0.05747 .
## s(CrHome)  8.094  8.510  3.217 0.00128 **
## s(CrRuns)  1.000  1.000  0.079 0.77944
## s(CrRbi)   8.447  8.770  2.961 0.00632 **
## s(CrBB)    1.000  1.000  0.138 0.71063
## s(nOuts)   5.957  7.070  1.982 0.05850 .
## s(nAssts)  1.000  1.000  0.325 0.56924
## s(nError)  3.601  4.463  1.224 0.25678
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.882   Deviance explained = 91.1%
## GCV = 0.12437  Scale est. = 0.093457  n = 263
```

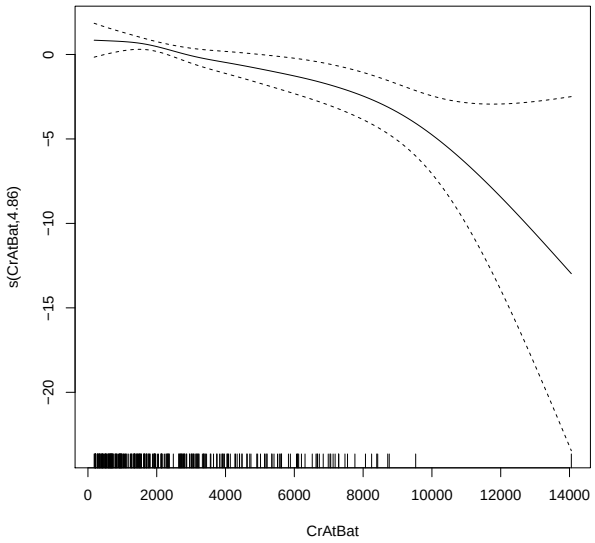
Now, let's refit the models but only using the significant terms:

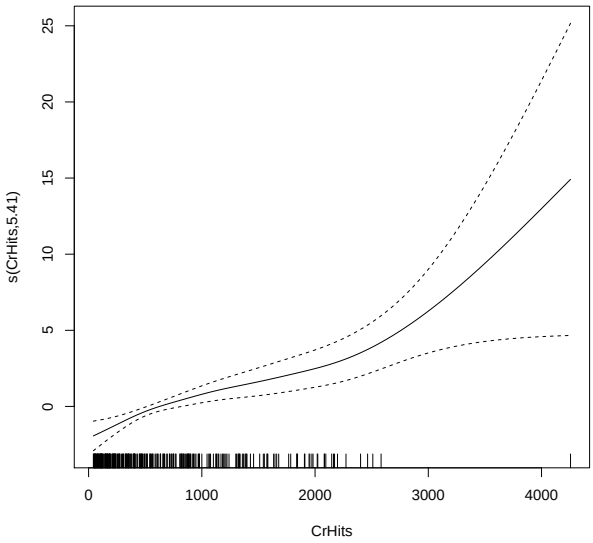
```
baseball_gam <- mgcv::gam(logSalary ~ s(nBB) + s(YrMajor) + s(CrAtBat) + s(CrHits) +
                          s(CrHome) + s(CrRbi) + s(nOuts), data = baseball)
```

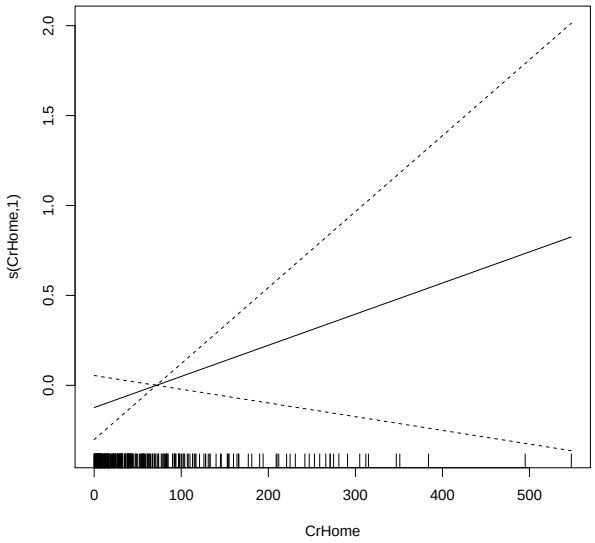
We can take a look at the estimated spline functions for each of the predictor variables. In each of the below plots, the x-axis contains the various levels of the predictor variables. On the y-axis, we see the estimated spline function (keep in mind that these are multiple different polynomial functions being concatenated together). Along the x-axis you will see little notches: these each indicate the unique points that went into creating the spline.

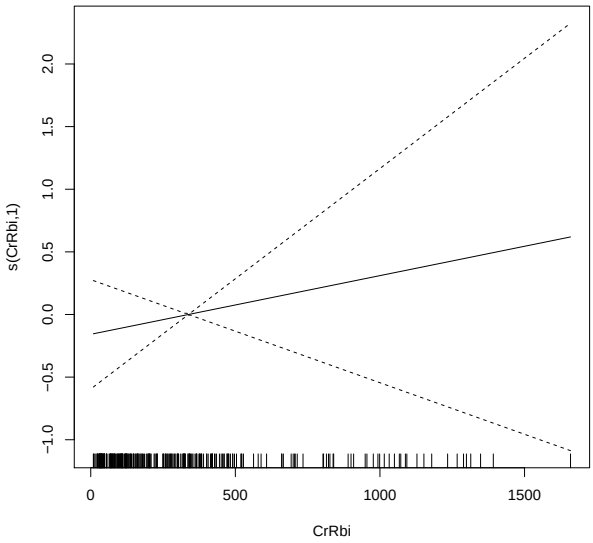
```
mgcv::plot.gam(baseball_gam, scale = 0)
```

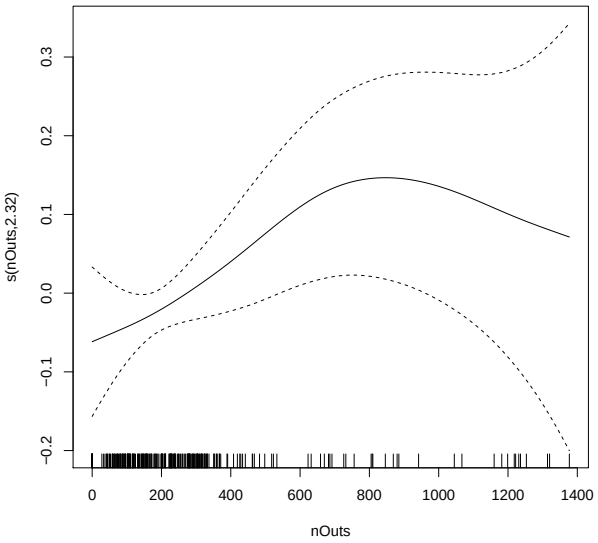



s(CrAtBat,4.86)
0
−5
−10
−15
−20
0
2000
4000
6000
8000
10000
12000
14000
CrAtBat

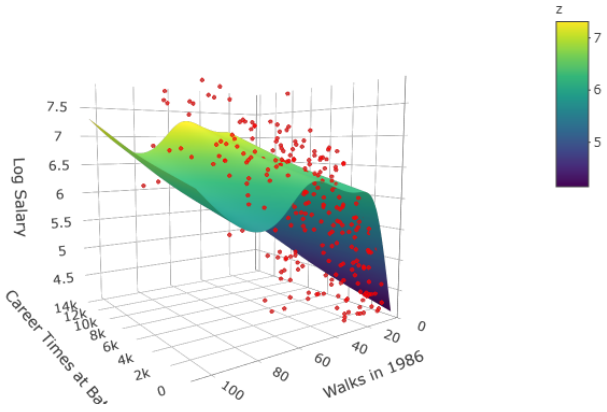

s(CrHits,5.41)
25
20
15
10
5
0
0
1000
2000
3000
4000
CrHits


s(CrHome,1)
2.0
1.5
1.0
0.5
0.0
0
100
200
300
400
500
CrHome


s(CrRbi,1)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
500
1000
1500
CrRbi



For simplicity, if we fit a GAM with just CrAtBat and nBB (like we did in the LOESS example), then we get the following surface plot:



This plot is comparable to the plot from the LOESS example in 4.3.1. Note that this plot uses an interactive plot feature available in the `plotly` package of R.

Example 2: Diabetes Dataset

The Pima Indians Diabetes dataset is a dataset from the National Institute of Diabetes and Digestive and Kidney Diseases. Our goal here is to predict whether or not a patient has diabetes. In this dataset, all patients are females that are at least 21 and are of Pima Indian heritage.

Let's split our data into a training and testing dataset and see how well we do on the testing dataset by training on the training dataset.

```
data("diabetes")
head(diabetes)

##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
```

```
## 1           6     148            72            35       0 33.6
## 2           1      85            66            29       0 26.6
## 3           8     183            64             0       0 23.3
## 4           1      89            66            23      94 28.1
## 5           0     137            40            35     168 43.1
## 6           5     116            74             0       0 25.6
##   DiabetesPedigreeFunction Age Outcome
## 1                    0.627  50       1
## 2                    0.351  31       0
## 3                    0.672  32       1
## 4                    0.167  21       0
## 5                    2.288  33       1
## 6                    0.201  30       0

# How many observations are there?
nrow(diabetes)

## [1] 768

# Create a training and testing split with 80% training data
train_index <- sample(1:nrow(diabetes), size = 0.80*nrow(diabetes))
diabetes_train <- diabetes[train_index, ]
diabetes_test <- diabetes[-train_index, ]

diabetes_gam <- mgcv::gam(Outcome ~ s(Pregnancies) + s(Glucose) + s(BloodPressure) +
                            s(SkinThickness) + s(Insulin) + s(BMI) +
                            s(DiabetesPedigreeFunction) + s(Age), family = "binomial",
                          data = diabetes_train)
```

Now let's see how accurate we are on the testing dataset:

```
# Here are the predicted class probabilities
test_class_prob <- predict(diabetes_gam, diabetes_test, type = "response")

# If the probability is higher than 50% of having diabetes, mark it as a 1.
pred_class <- rep(0, nrow(diabetes_test))
pred_class[test_class_prob > 0.50] <- 1

# Now that we have our predicted class, let's get some statistics on our accuracy.
total_test <- nrow(diabetes_test)
total_correct <- sum(pred_class == diabetes_test$Outcome)

# Error rate
(total_test - total_correct) / total_test

## [1] 0.1948052

# Successful prediction rate
total_correct / total_test

## [1] 0.8051948
```