# 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(gam)

## Loading required package:  splines
## Loading required package:  foreach
## Loaded gam 1.20

data(baseball)

baseball_gam_all <-
  gam::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)

##
## Call: gam::gam(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, data = baseball)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.94377 -0.14529  0.01674  0.19599  0.77441
##
## (Dispersion Parameter for gaussian family taken to be 0.1272)
##
##     Null Deviance: 207.1537 on 262 degrees of freedom
## Residual Deviance: 24.9385 on 195.9998 degrees of freedom
## AIC: 262.8024
## 59 observations deleted due to missingness
##
## Number of Local Scoring Iterations: NA
```

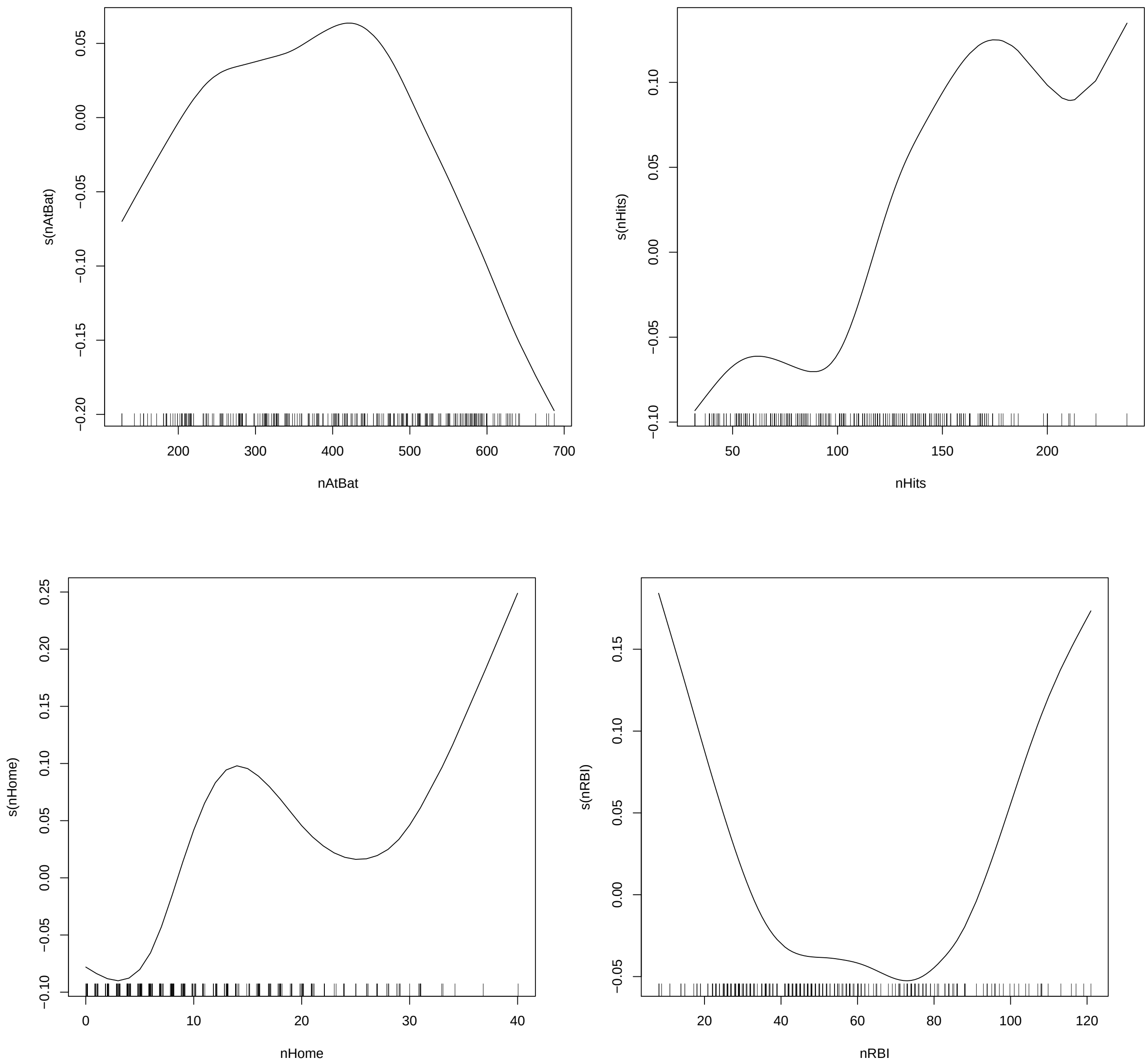
```
##
## Anova for Parametric Effects
##              Df Sum Sq Mean Sq  F value    Pr(>F)
## s(nAtBat)     1 23.931  23.931 188.0801 < 2.2e-16 ***
## s(nHits)      1  2.148   2.148  16.8800 5.848e-05 ***
## s(nHome)      1  7.285   7.285  57.2579 1.455e-12 ***
## s(nRuns)      1  0.059   0.059   0.4636    0.4967
## s(nRBI)       1  2.360   2.360  18.5464 2.619e-05 ***
## s(nBB)        1  9.493   9.493  74.6069 2.001e-15 ***
## s(YrMajor)    1 39.177  39.177 307.9045 < 2.2e-16 ***
## s(CrAtBat)    1 18.338  18.338 144.1265 < 2.2e-16 ***
## s(CrHits)     1  2.664   2.664  20.9347 8.420e-06 ***
## s(CrHome)     1  2.406   2.406  18.9076 2.204e-05 ***
## s(CrRuns)     1  0.051   0.051   0.4042    0.5256
## s(CrRbi)      1  0.286   0.286   2.2446    0.1357
## s(CrBB)       1  0.000   0.000   0.0003    0.9857
## s(nOuts)      1  0.158   0.158   1.2417    0.2665
## s(nAssts)     1  0.005   0.005   0.0398    0.8420
## s(nError)     1  0.079   0.079   0.6225    0.4311
## League        1  0.288   0.288   2.2619    0.1342
## Division      1  0.166   0.166   1.3014    0.2553
## Residuals   196 24.939   0.127
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##              Npar Df Npar F     Pr(F)
## (Intercept)
## s(nAtBat)          3  2.153  0.094866 .
## s(nHits)           3  1.122  0.341421
## s(nHome)           3  2.340  0.074667 .
## s(nRuns)           3  1.424  0.237075
## s(nRBI)            3  1.999  0.115515
## s(nBB)             3  1.769  0.154376
## s(YrMajor)         3 34.309 < 2.2e-16 ***
## s(CrAtBat)         3  4.826  0.002899 **
## s(CrHits)          3  4.655  0.003628 **
## s(CrHome)          3  1.888  0.132929
## s(CrRuns)          3  3.438  0.017912 *
## s(CrRbi)           3  3.383  0.019259 *
## s(CrBB)            3  3.036  0.030305 *
## s(nOuts)           3  2.231  0.085855 .
## s(nAssts)          3  0.845  0.470916
## s(nError)          3  1.743  0.159571
## League
## Division
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

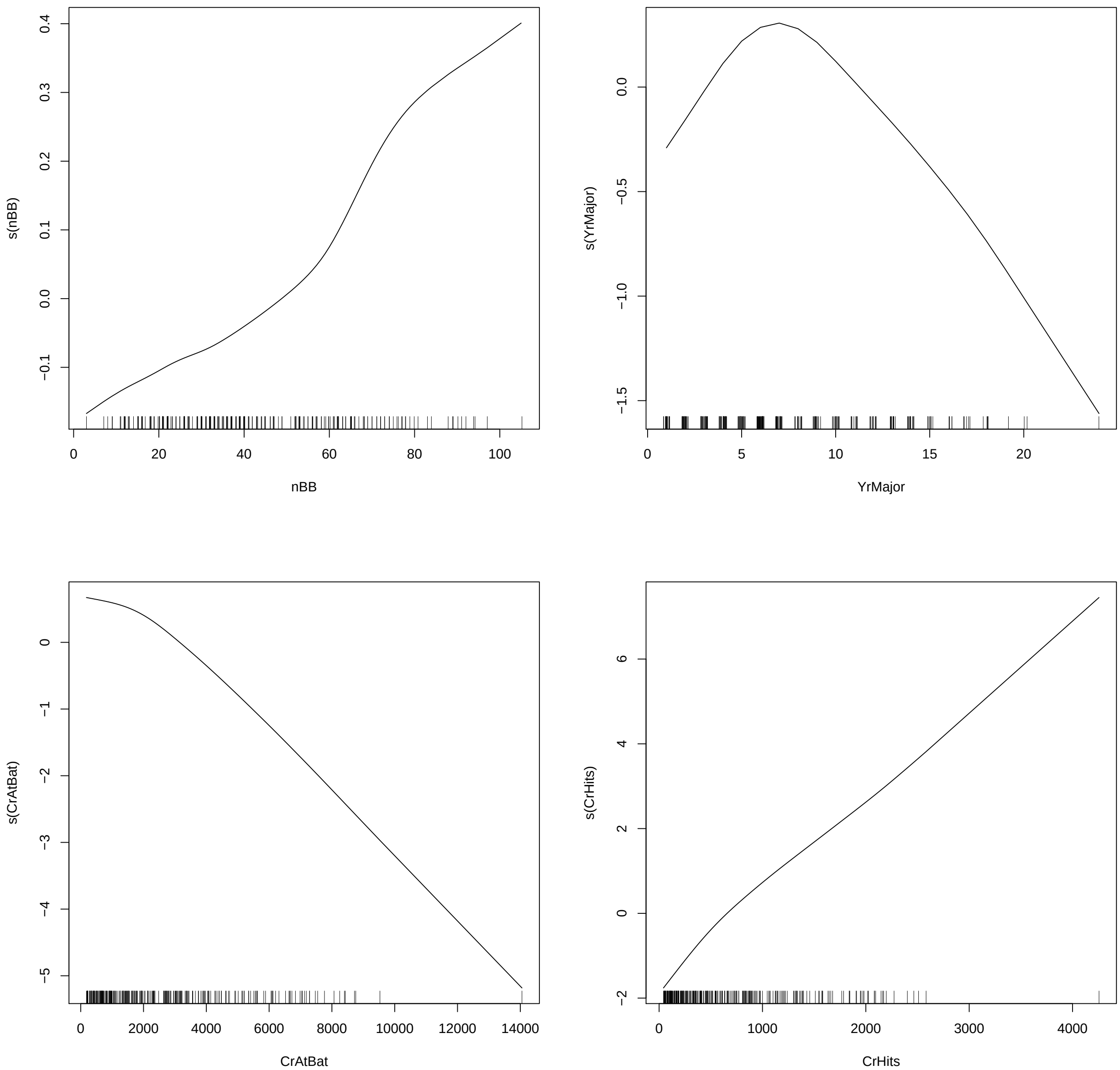
Now, let's refit the models but only using the significant terms:

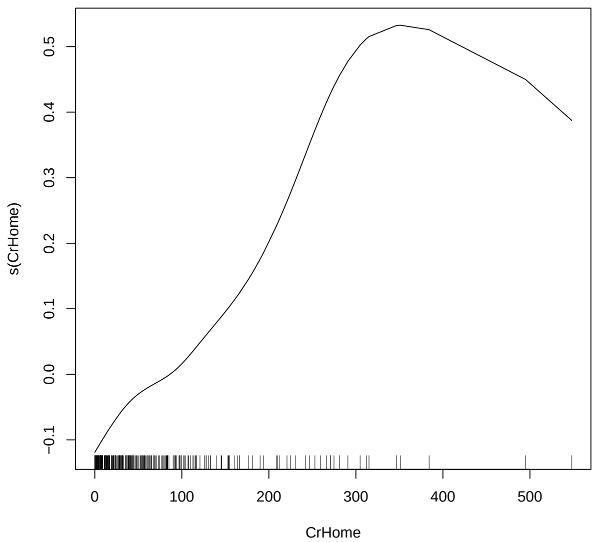
```
baseball_gam <- gam::gam(logSalary ~ s(nAtBat) + s(nHits) + s(nHome) + s(nRBI) +
                           s(nBB) + s(YrMajor) + s(CrAtBat) + s(CrHits) +
                           s(CrHome), data = baseball)
```

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.

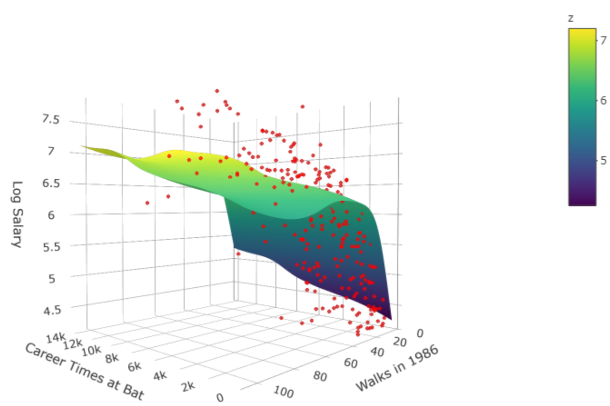
```r
plot(baseball_gam)
```




s(nBB)
nBB
s(YrMajor)
YrMajor
s(CrAtBat)
CrAtBat
s(CrHits)
CrHits




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.

---

## 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 <- gam::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] 0.3116883

# Successful prediction rate
total_correct / total_test

## [1] 0.6883117
```