

# Advanced Topics in Regression

## Kernels, Smoothers, and Generalized Additive Models

07-11-2023

### Kernels

In Statistical contexts: **a kernel is a symmetric PDF**

Examples:

- Normal distribution
- Uniform distribution

### Kernel Regression

The classical kernel regression estimator is the **Nadaraya-Watson** estimator:

$$\hat{y}_h(x) = \sum_{i=1}^n w_i(x) Y_i$$

where:

$$w_i(x) = \frac{K\left(\frac{x-X_i}{h}\right)}{\sum_{j=1}^n K\left(\frac{x-X_j}{h}\right)}$$

Regression estimate is the average of all the weighted observed response values

- Farther  $x$  is from the observation, the less weight that observation has in determining the regression estimate at  $x$ .

### Nadaraya-Watson

- given training data with explanatory variable  $x$  and continuous response  $y$
- bandwidth  $h > 0$
- and a new point

**Example of a linear smoother:** class of models where predictions are weighted sums of the response variable.

### Local Regression

We can fit a linear model **at each point**  $x_{new}$  with weights given by kernel function centered on  $x_{new}$

Local regression of the  $k$ th order with kernel function  $K$  solves the following:

$$\hat{\beta}(x_{new}) = \arg \min_{\beta} \left\{ \sum_i K_h(|x_{new} - x_i|) \cdot (y_i - \sum_{j=0}^k x_i^j \cdot \beta_j)^2 \right\}$$

So, every single observation has its own set of coefficients

The predicted value is then:

$$\hat{y}_{new} = \sum_{j=0}^k x_{new}^j \cdot \hat{\beta}_j(x_{new})$$

This is a smoother prediction than with kernel regression but comes at a higher computational cost

- LOESS replaces kernel with k nearest neighbors (discrete average)

## Smoothing Splines

Use a **smooth function**  $s(x)$  to predict  $y$ , control smoothness directly by minimizing the **spline objective function**:

$$\sum_{i=1}^n (y_i - s(x_i))^2 + \lambda \int (s''(x))^2 dx$$

= fit data + impose smoothness

$$\Rightarrow \text{model fit} = \text{likelihood} - \lambda \cdot \text{wiggliness}$$

Estimate the **smoothing spline**  $\hat{s}(x)$  that **balances the tradeoff between the model fit and the wiggliness**

## Basis Functions

Splines are piecewise cubic polynomials with **knots** (boundary points for functions) at every data point.

- Practical alternative: linear combination of a set of **basis functions**

Examples:

For a cubic polynomial:

- $B_1(x) = 1$ ,  $B_2(x) = x$ ,  $B_3(x) = x^2$ ,  $B_4(x) = x^3$

$$r(x) = \sum_j^4 \beta_j B_j(x)$$

\* Linear in the transformed variables  $B_1(x), B_2(x), B_3(x), B_4(x)$  but it is **nonlinear in  $x$**

We extend this idea for splines *piecewise* using indicator functions so the spline is a weighted sum:

$$s(x) = \sum_j^m \beta_j B_j(x)$$

## Generalized Additive Models (GAMS)

- relationships between individual explanatory variables and the response variable are smooth (either linear or nonlinear via basis functions)
- estimate the smooth relationships simultaneously to predict the response by just adding them up

**generalized** like GLMs where  $g()$  is the link function for the expected value of the response  $E(Y)$  and **additive** over the  $p$  variables.

$$g(E(Y)) = \beta_0 + s_1(x_1) + s_2(x_2) + \cdots + s_p(x_p)$$

\* can be a convenient balance between flexibility and interpretability

- you can combine linear and nonlinear terms!

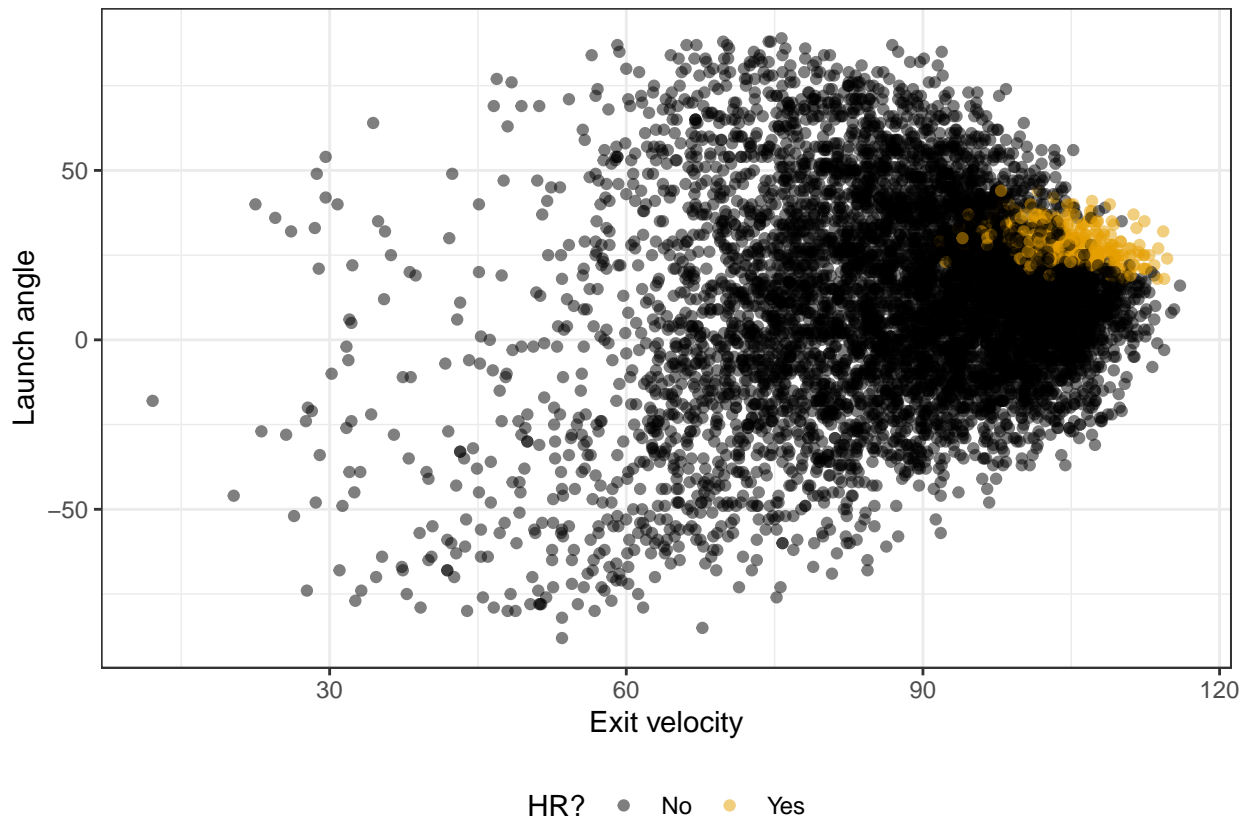
## Example

```
batted_ball_data <- read_csv("https://shorturl.at/moty2") %>%
  mutate(is_hr = as.numeric(events == "home_run")) %>%
  filter(!is.na(launch_angle), !is.na(launch_speed), !is.na(is_hr))

head(batted_ball_data)

## # A tibble: 6 x 32
##   player_name    batter stand events    hc_x  hc_y hit_distance_sc launch_speed
##   <chr>          <dbl> <chr> <chr>    <dbl> <dbl>         <dbl>         <dbl>
## 1 Daza, Yonathan 602074 R      force_out 103.  150.           18           97.4
## 2 Robles, Victor 645302 R      single    58.6 120.          158           80.2
## 3 Hoerner, Nico  663538 R      field_out  99.3 166.           20          101.
## 4 Clemens, Kody  665019 L      field_out 126.  191.          165            84
## 5 Rosario, Amed  642708 R      field_out  97.4 170.            9          94.3
## 6 Castro, Willi  650489 L      sac_fly   178.  58.9          369            96
## # i 24 more variables: launch_angle <dbl>, hit_location <dbl>, bb_type <chr>,
## #   barrel <dbl>, pitch_type <chr>, release_speed <dbl>, effective_speed <dbl>,
## #   if_fielding_alignment <chr>, of_fielding_alignment <chr>, game_date <date>,
## #   balls <dbl>, strikes <dbl>, outs_when_up <dbl>, on_1b <dbl>, on_2b <dbl>,
## #   on_3b <dbl>, inning <dbl>, inning_topbot <chr>, home_score <dbl>,
## #   away_score <dbl>, post_home_score <dbl>, post_away_score <dbl>, des <chr>,
## #   is_hr <dbl>

batted_ball_data %>% ggplot(aes(x = launch_speed, y = launch_angle,
                                color = as.factor(is_hr))) +
  geom_point(alpha = 0.5) +
  ggthemes::scale_color_colorblind(labels = c("No", "Yes")) +
  labs(x = "Exit velocity", y = "Launch angle", color = "HR?") +
  theme_bw() +
  theme(legend.position = "bottom")
```



```
# setting up the training data

set.seed(2004)

batted_ball_data <- batted_ball_data %>%
  mutate(is_train = sample(rep(0:1, length.out = nrow(batted_ball_data))))

init_logit_gam <- gam(is_hr ~ s(launch_speed) + s(launch_angle),
  data = filter(batted_ball_data, is_train == 1), family = binomial, method = "REML")

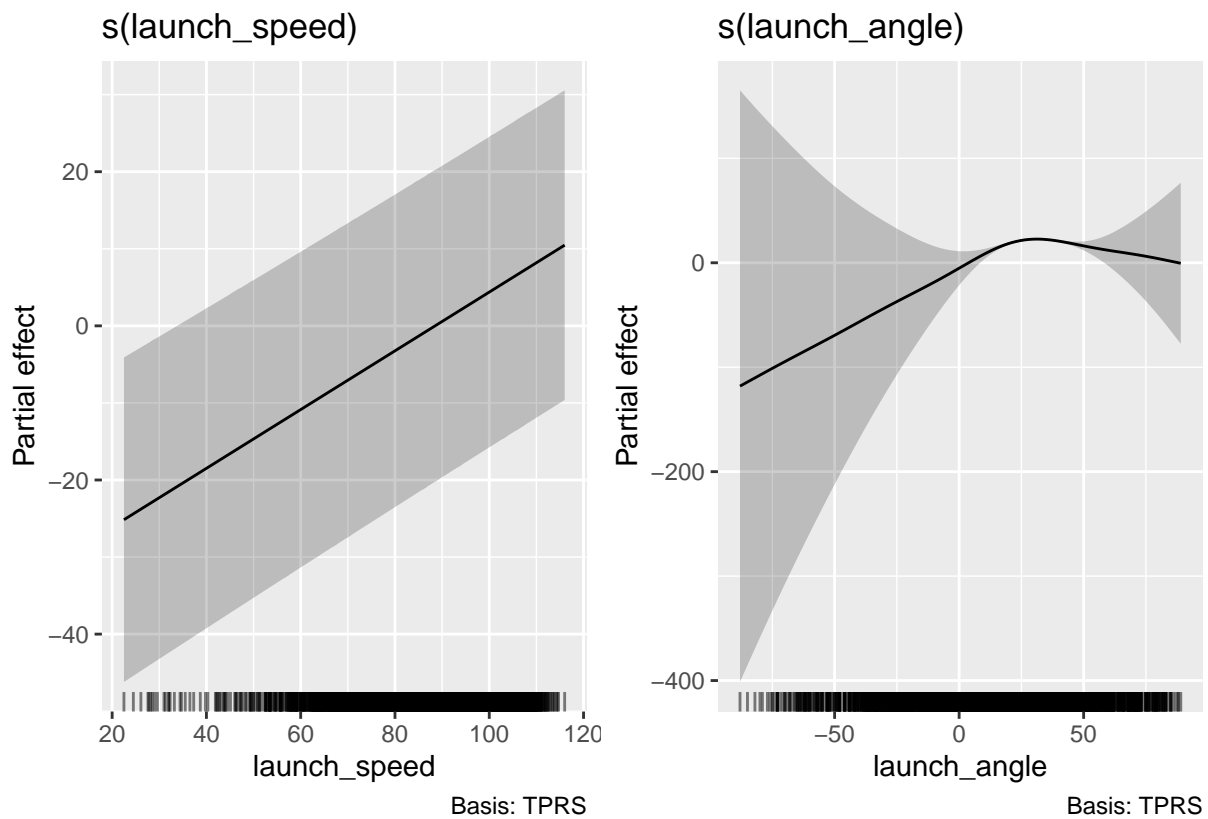
# REML allows for a more stable solution

summary(init_logit_gam)

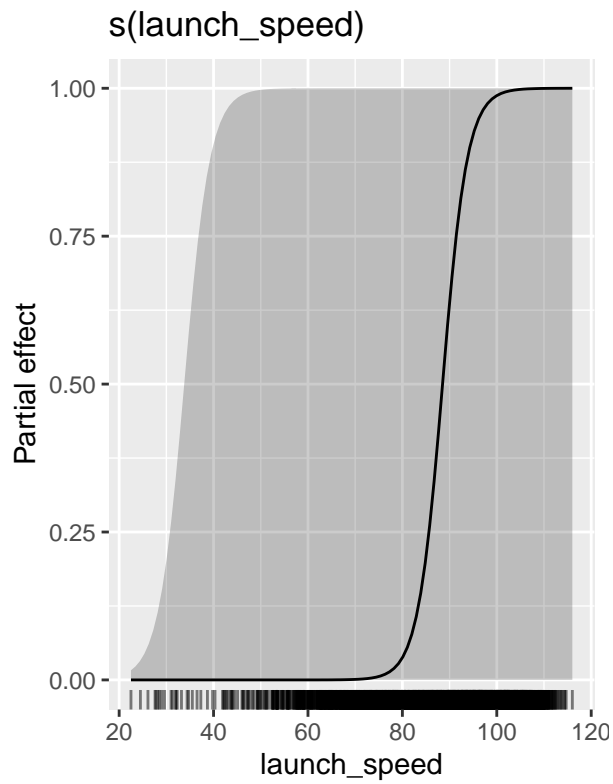
##
## Family: binomial
## Link function: logit
##
## Formula:
## is_hr ~ s(launch_speed) + s(launch_angle)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -26.96      10.31  -2.614  0.00895 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df Chi.sq p-value
```

```
## s(launch_speed) 1.000 1.000 151.5 <2e-16 ***
## s(launch_angle) 2.962 3.305 112.0 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) = 0.588 Deviance explained = 68.3%
## -REML = 231.49 Scale est. = 1 n = 3517
```

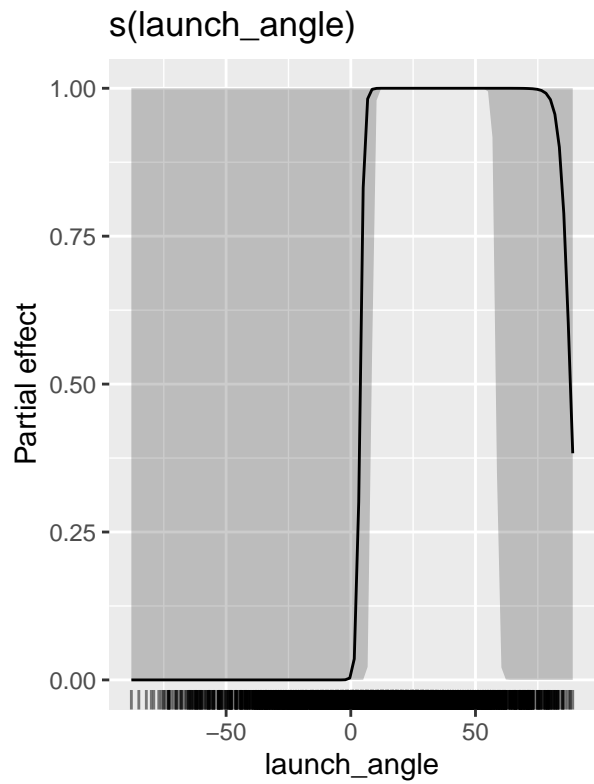
```
# displays the partial effect of each term in the model. Add up to the overall prediction
draw(init_logit_gam)
```



```
draw(init_logit_gam, fun = plogis)
```



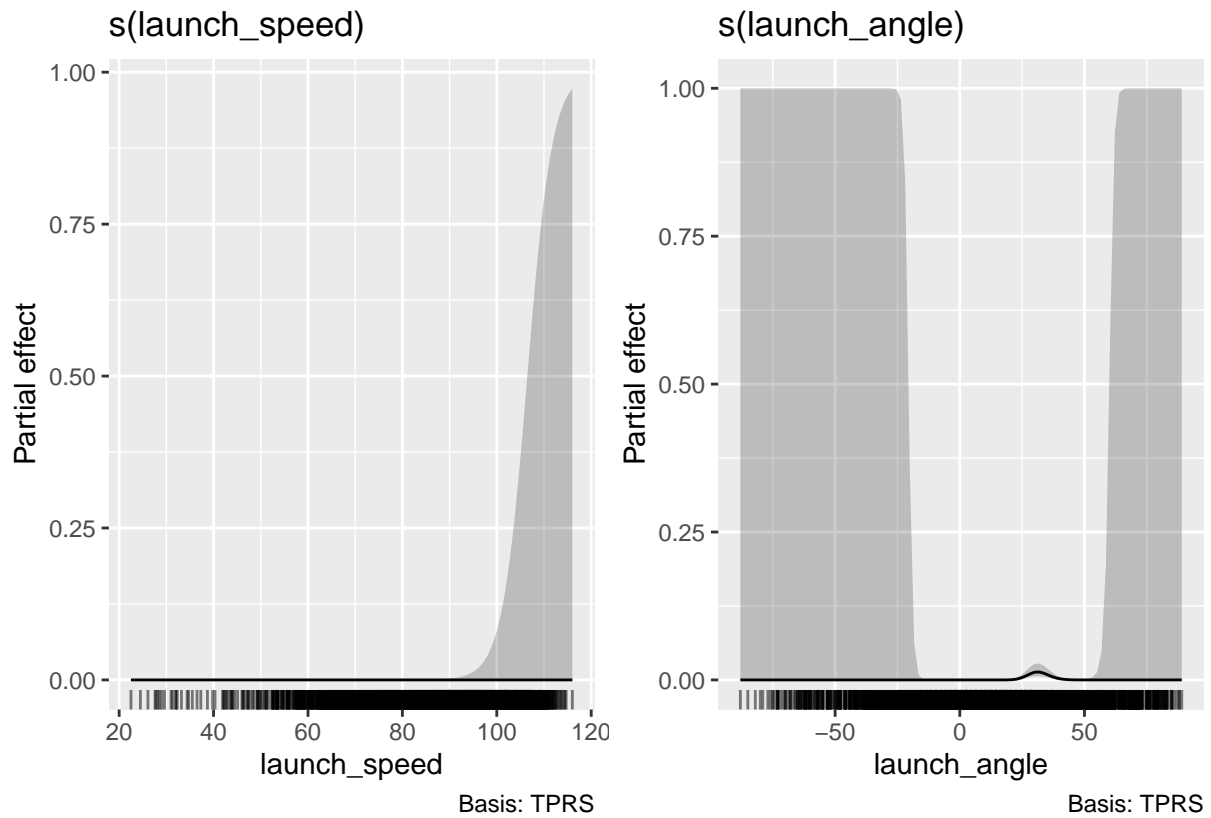
Basis: TPRS



Basis: TPRS

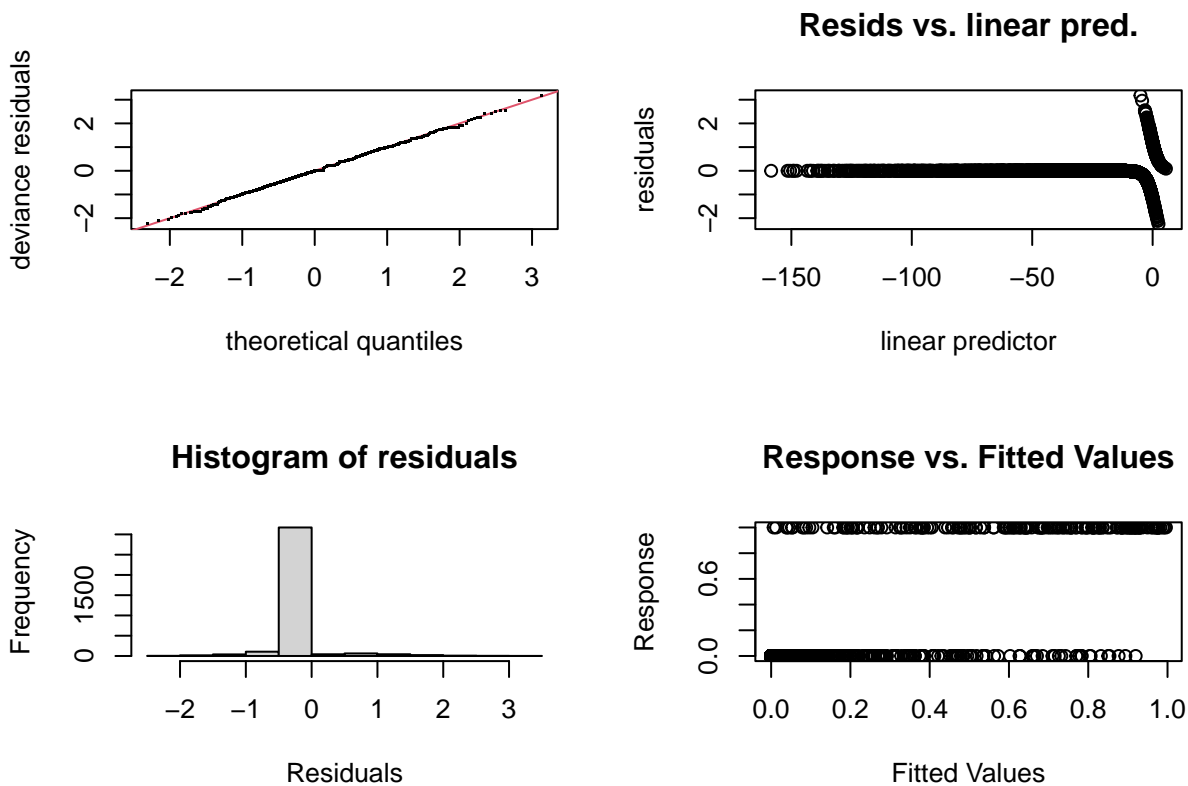
*#centered on average value of 0.5 because it's the partial effect without the intercept*

```
draw(init_logit_gam, fun = plogis, constant = coef(init_logit_gam)[1])
```



*# intercept reflects relatively rare occurrence of HRs!*

*# Use gam.check() to see if we need more basis functions based on an approximate test*  
 gam.check(init\_logit\_gam)



```
##
## Method: REML   Optimizer: outer newton
## full convergence after 11 iterations.
## Gradient range [-5.632542e-05,-2.964163e-06]
## (score 231.4864 & scale 1).
## Hessian positive definite, eigenvalue range [5.631851e-05,0.8679399].
## Model rank = 19 / 19
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##           k'   edf k-index p-value
## s(launch_speed) 9.00 1.00   1.05   1.00
## s(launch_angle) 9.00 2.96   0.97   0.08 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
batted_ball_data <- battered_ball_data %>%
  mutate(init_gam_hr_prob = as.numeric(predict(init_logit_gam,
    newdata = battered_ball_data, type = "response")),
    init_gam_hr_class = as.numeric(init_gam_hr_prob >= 0.5))

battered_ball_data %>%
  group_by(is_train) %>%
  summarize(correct = mean(is_hr == init_gam_hr_class))
```

```
## # A tibble: 2 x 2
##   is_train correct
##   <int>   <dbl>
## 1       0  0.977
```



```
## 2      1    0.972
```

## What about the linear model?

```
init_linear_logit <- glm(is_hr ~ launch_speed + launch_angle,  
  data = filter(batted_ball_data, is_train == 1), family = binomial)  
  
batted_ball_data <- batted_ball_data %>%  
  mutate(init_glm_hr_prob = predict(init_linear_logit,  
    newdata = batted_ball_data, type = "response"),  
    init_glm_hr_class = as.numeric(init_glm_hr_prob >= 0.5))  
  
batted_ball_data %>%  
  group_by(is_train) %>%  
  summarize(correct = mean(is_hr == init_glm_hr_class))
```

```
## # A tibble: 2 x 2  
##   is_train correct  
##   <int>   <dbl>  
## 1      0    0.960  
## 2      1    0.951
```

- there are very few situations in reality where linear regressions perform better than an additive model using smooth functions – especially since smooth functions can just capture the linear model