Analyzing Sports Data in R

St. Louis R User Group

John Snyder

May 27th, 2020

Overview

Extensive data are being collected on athletes, teams, and opponents, and these data are being used in various ways

- For player evaluation and development
- To make strategic and tactical decisions
- To enhance the fan experience

This provides many with a rich collection of data which can be explored with almost limitless potential.

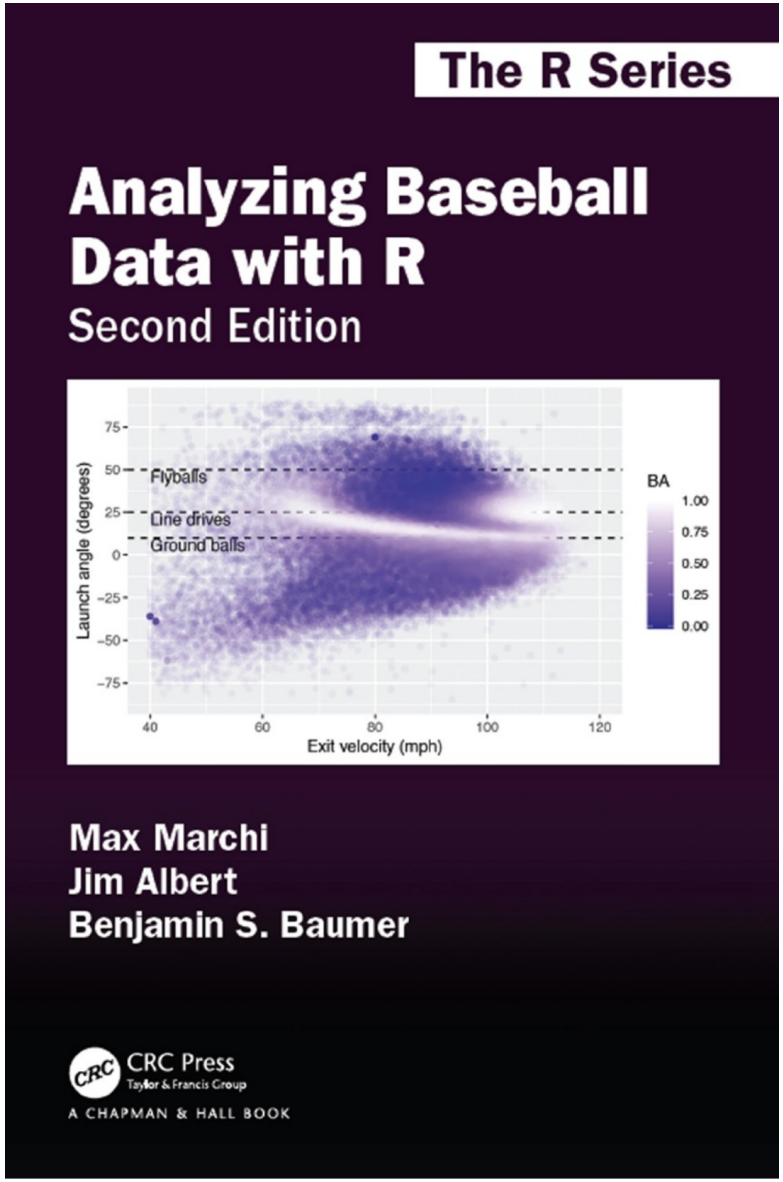
- Visualizations
- Advanced analysis

Here we focus on publicly available datasets.

- Disclaimer: I am not a huge sports fan, domain knowledge is thin.
- I am a big fan of data though.

Baseball - MLB Statcast System

- Statcast system
  - Fast and accurate tracking of movements on the field.
  - Uses arrays of cameras to pinpoint locations of players and balls.
  - All MLB teams have this technology in their stadiums.
  - Teams use data from this system to gain competitive advantage
  - Raw data is not public, but processed data is.
    - ★ baseballsavant.mlb.com
- We will focus on batted ball data here.
- See Chapter 12 of Marchi et. al.



Setup Libraries

```
library(tidyverse)
library(ggplot2)
library(parallel)

# devtools::install_github("BillPetti/baseballr")
library(baseballr)

# Modeling packages
library(mgcv)
library(xgboost)
#Output interpretation packages
library(pROC) # for AUC
library(caret)
library(pdp)
library(vip)
```

- The *baseballr* package provides functionality to scrape baseballsavant.mlb.com.
  - Returns a maximum of 40,000 rows.
  - Later we will see how to download all data.

Paul Goldschmidt 2019 Data

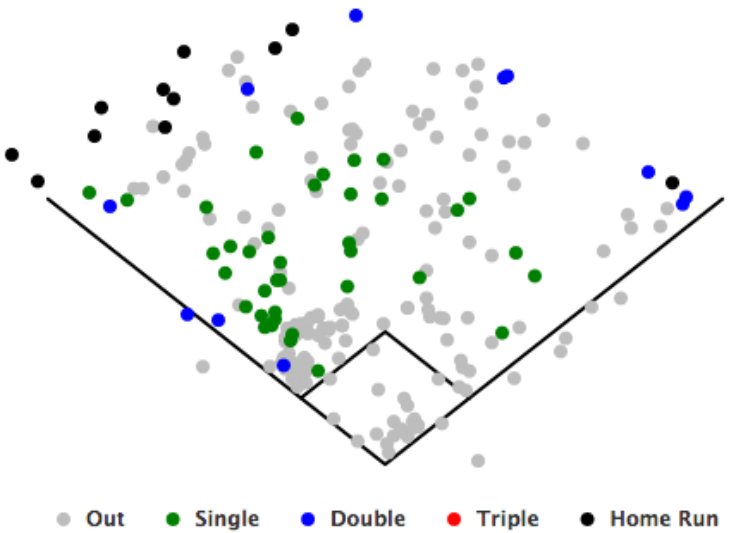
We can use *scrape_statcast_savant()* to download all Paul Goldschmidt's at bats for 2019 directly from Baseball Savant.

```
goldy_id <- (playerid_lookup(last_name = "Goldschmidt",
                             first_name = "Paul")$mlbam_id)
goldy <- scrape_statcast_savant(start_date = "2019-03-01",
                                end_date = "2019-11-30",
                                # single playerid only
                                playerid = goldy_id,
                                player_type = "batter")

goldy_hits <- goldy %>% filter(type == "X") # Only consider hits
goldy_hits %>%
  dplyr::select(events, stand, p_throws, hc_x, hc_y, launch_angle) %>%
  head(3)
```

```
## # A tibble: 3 x 6
##   events     stand p_throws  hc_x  hc_y launch_angle
##   <chr>      <chr> <chr>    <dbl> <dbl>        <dbl>
## 1 field_out R     L        149.   57.8         41.3
## 2 field_out R     R         78.9  74.6         44.7
## 3 single    R     R         72.4  92.5         14.4
```

Spray Chart: Visualization of Batted Balls

- Spray Chart
  - Visualization of where batted balls end up on the field.
  - Track every ball put in play for the entire season.
  - Highlights player tendencies.
  - Allow insights into outcomes based on ball position.

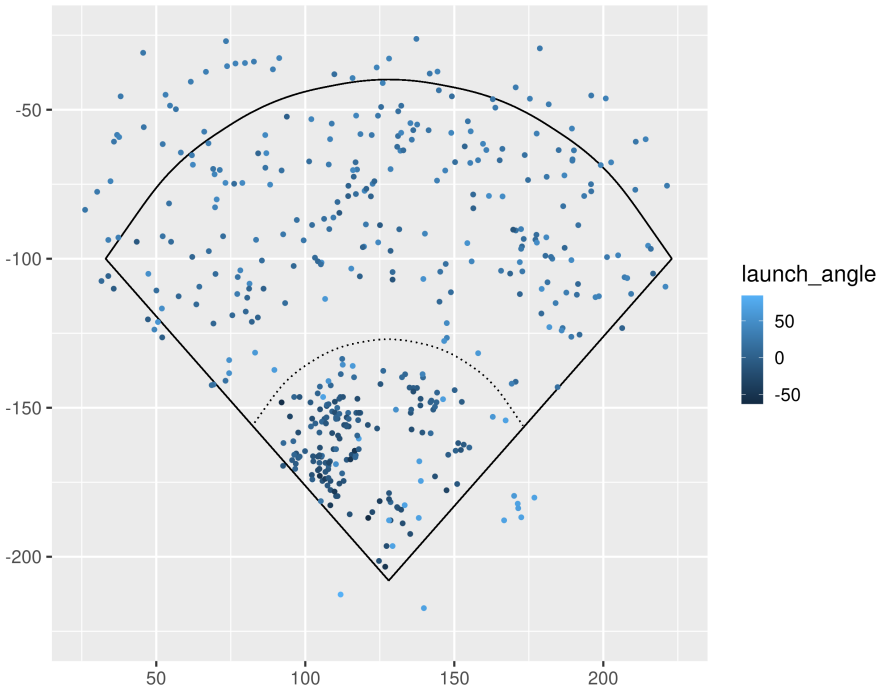



```
spray_chart <- function(...) {
  ggplot(...) +
    # add curves to draw the field
    geom_curve(x = 33, xend = 223,
               y = -100, yend = -100,
               curvature = -.65,
               color="black") +
    geom_segment(x = 128, xend = 33,
                 y = -208, yend = -100,
                 color="black") +
    geom_segment(x = 128, xend = 223,
                 y = -208, yend = -100,
                 color="black") +
    geom_curve(x = 83, xend = 173,
               y = -155, yend = -156,
               curvature = -.65,
               linetype = "dotted",
               color="black") +
    coord_fixed() +
    scale_x_continuous(NULL,
                       limits = c(25,225)) +
    scale_y_continuous(NULL,
                       limits = c(-225,-25))
}
```

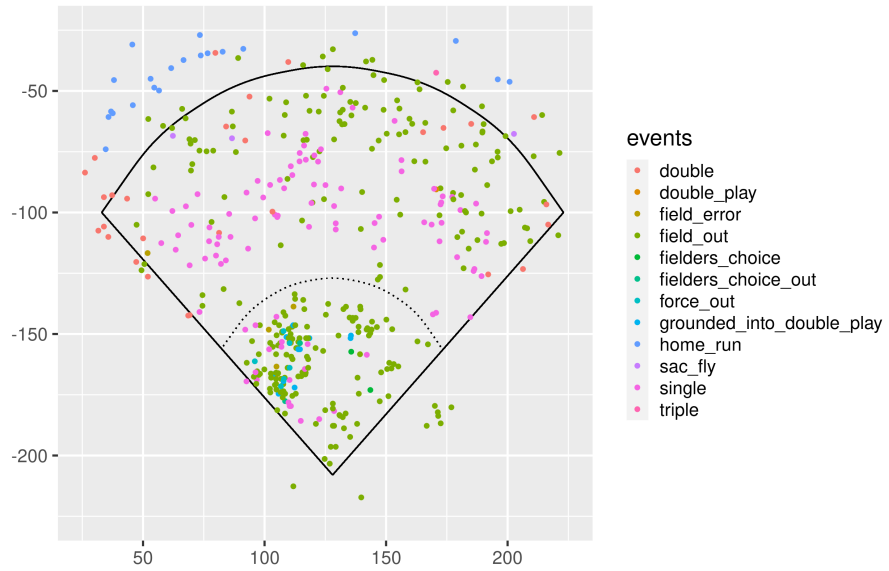
Paul Goldschmidt Spray Charts for 2019

These spray charts provide insights into hit tendencies

```
spray_chart(goldy_hits, aes(x = hc_x,
                            y = -hc_y,
                           color = launch_angle)) +
  geom_point() + theme_grey(base_size = 18)
```




```
spray_chart(goldy_hits, aes(x = hc_x,
                            y = -hc_y,
                           color = events)) +
  geom_point() + theme_grey(base_size = 18)
```



Pulling All Available Statcast Data

```
scrape_dates <- data.frame(
  start_date = as.character(as.Date("2015-03-01") + (0:250)*7),
  end_date   = as.character(as.Date("2015-03-01") + (0:250)*7 + 6))
head(scrape_dates,3)
```

```
##   start_date   end_date
## 1 2015-03-01 2015-03-07
## 2 2015-03-08 2015-03-14
## 3 2015-03-15 2015-03-21
```

```
scrape_dates.list <- split(scrape_dates, seq(nrow(scrape_dates)))

cl <- makeForkCluster(detectCores())
parLapplyLB(cl, scrape_dates.list,
            function(week) {
              cur_file_path <- paste0("~/hitter_csv/",
                                      week$start_date,"--",week$end_date,".csv")
              dat <- baseballr::scrape_statcast_savant_batter_all(week$start_date,
                                                                  week$end_date)
              readr::write_csv(x = dat,path = cur_file_path)
            })
stopCluster(cl)

all_bound_csvs = plyr::ldply(dir("~/hitter_csv/", full.names = TRUE), read_csv)
write_csv(all_bound_csvs, "~/hitter_csv/batter_all.csv")
```

Data Reading and Formatting

```
# https://baseballsavant.mlb.com/csv-docs
batter <- read.csv("~/hitter_csv/batter_all.csv")
batter$game_date <- as.Date(batter$game_date)
batter$year  <- format(batter$game_date, "%Y")
batter$month <- format(batter$game_date,"%m")
batter$yrmo  <- format(batter$game_date,"%Y-%m")
batter$home_run <- ifelse(batter$events=="home_run","Yes","No") %>%
                    as.factor

batter_hit <- batter %>% filter(type=="X")
```

```
nrow(batter)
```

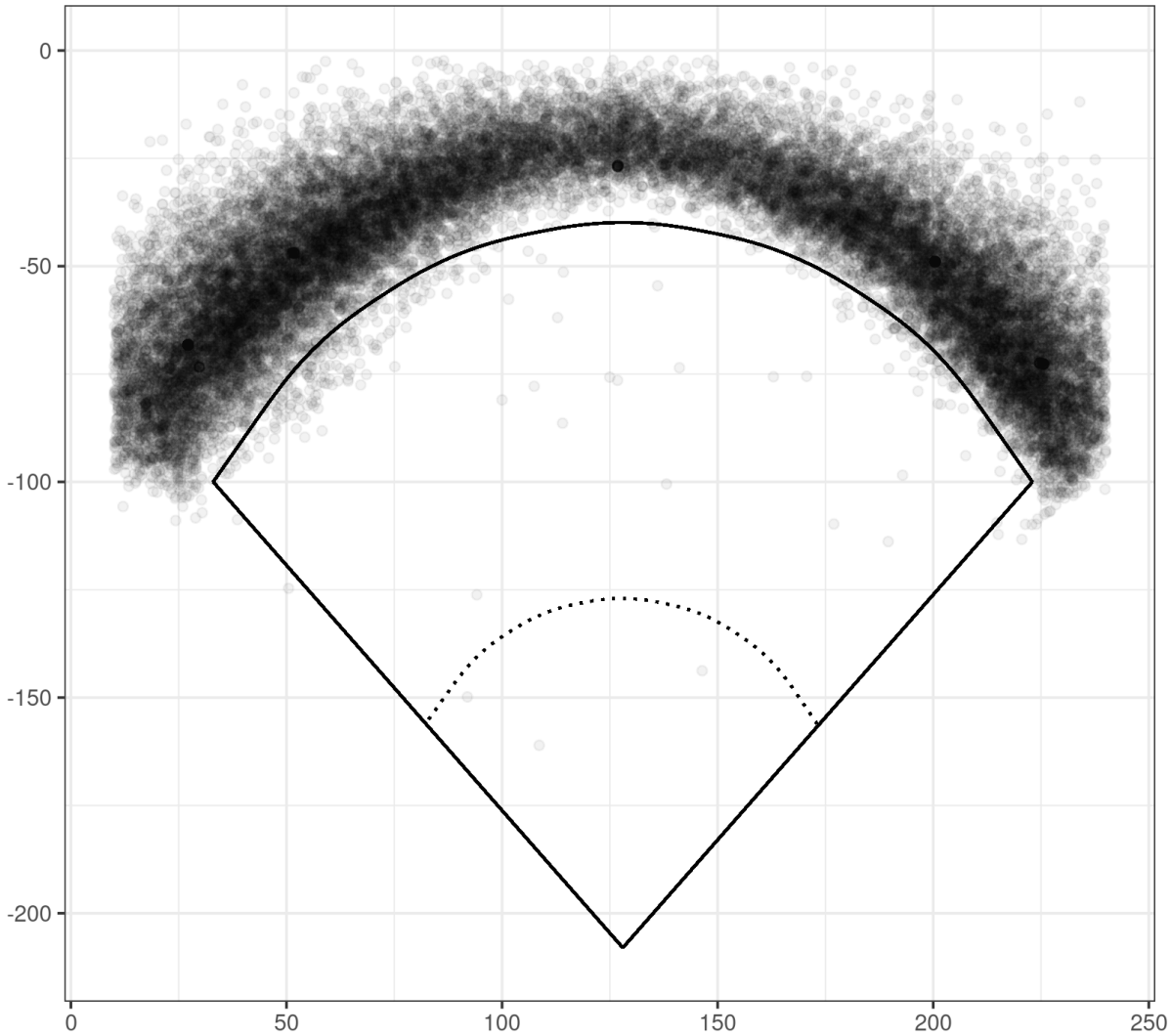
```
## [1] 3646118
```

The full dataset contains play by play data:

- Player information for both teams.
- Batted ball characteristics
- Pitch characteristics

Home Run Spray Chart

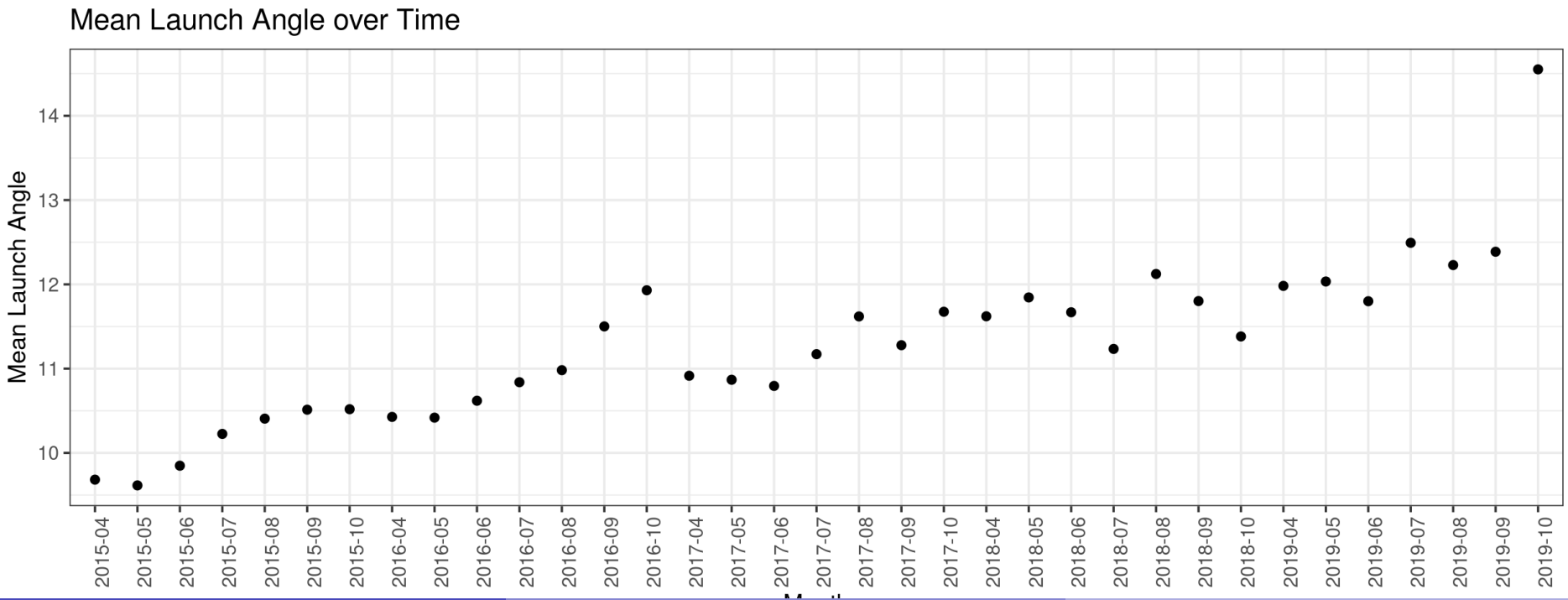
```
batter %>% filter(events == "home_run") %>%
spray_chart(aes(x = hc_x, y = -hc_y)) +
  geom_point(alpha = 0.05) + theme_bw() +
    scale_x_continuous(NULL, limits = c(10, 240)) + scale_y_continuous(NULL, limits = c(-210, 0))
```



Launch Angle Over Time

Player development has focused on increasing the angle of balls off the bat in recent years.

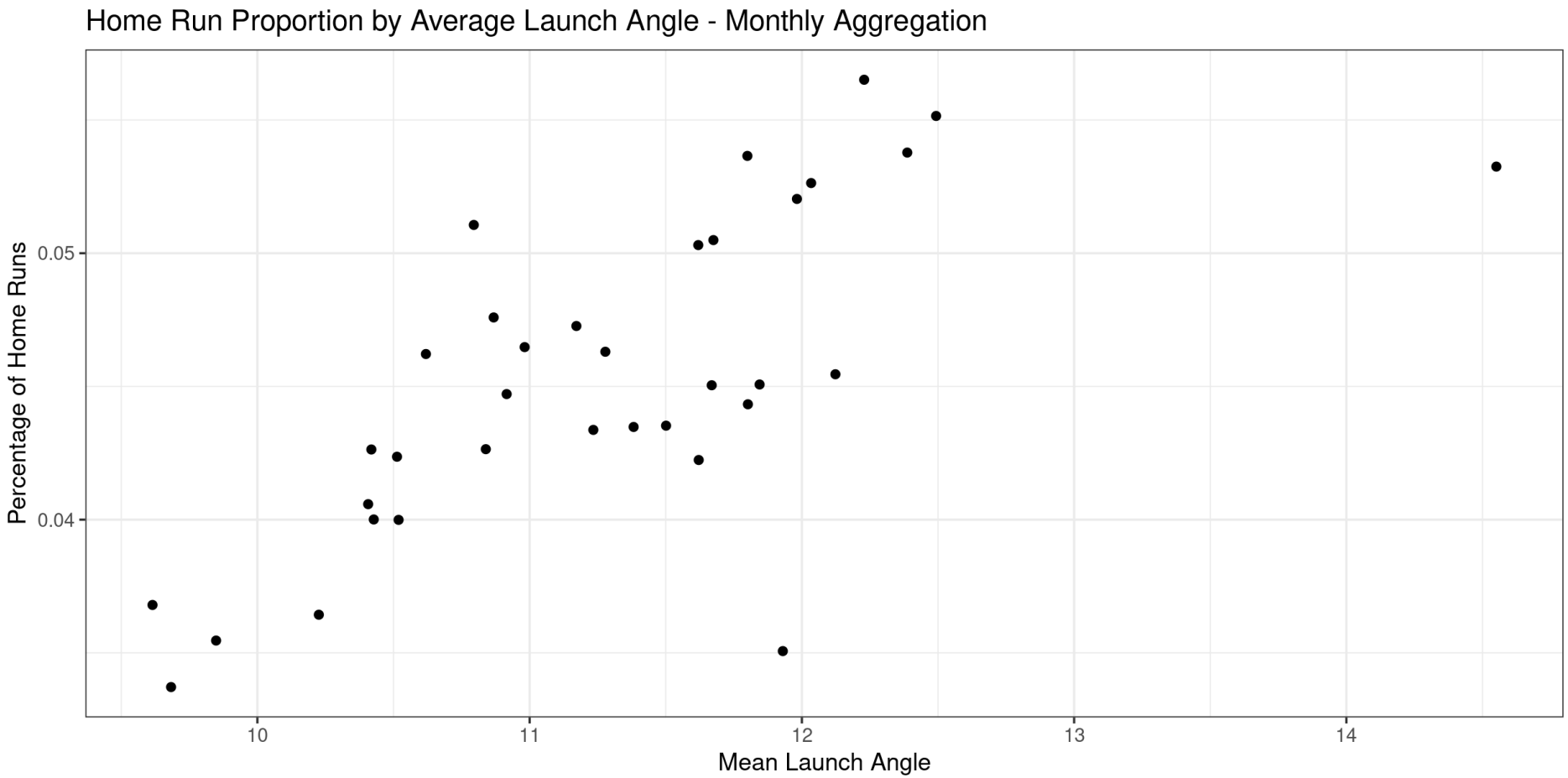
```
HR <- batter_hit %>%
  filter(!(month %in% c("03","11"))) %>% # Remove March and November
  group_by(yrmo) %>%
  summarise(m_la = mean(launch_angle,na.rm = T),
            prop_HR = sum(home_run == "Yes")/n())

ggplot(HR,aes(x=yrmo)) + geom_point(aes(y=m_la)) +
  ggtitle("Mean Launch Angle over Time") + xlab("Month") + ylab("Mean Launch Angle") +
  theme_bw() + theme(axis.text.x = element_text(angle = 90))
```



Launch Angle's Effect on Home Runs

There seems to be a relationship between launch angle and home runs.

```
ggplot(HR,aes(x=m_la,y=prop_HR)) + geom_point() +
  ggtitle("Home Run Proportion by Average Launch Angle - Monthly Aggregation") +
  xlab("Mean Launch Angle") + ylab("Percentage of Home Runs") +
  theme_bw()
```



wOBA as a Function of Hitting Characteristics

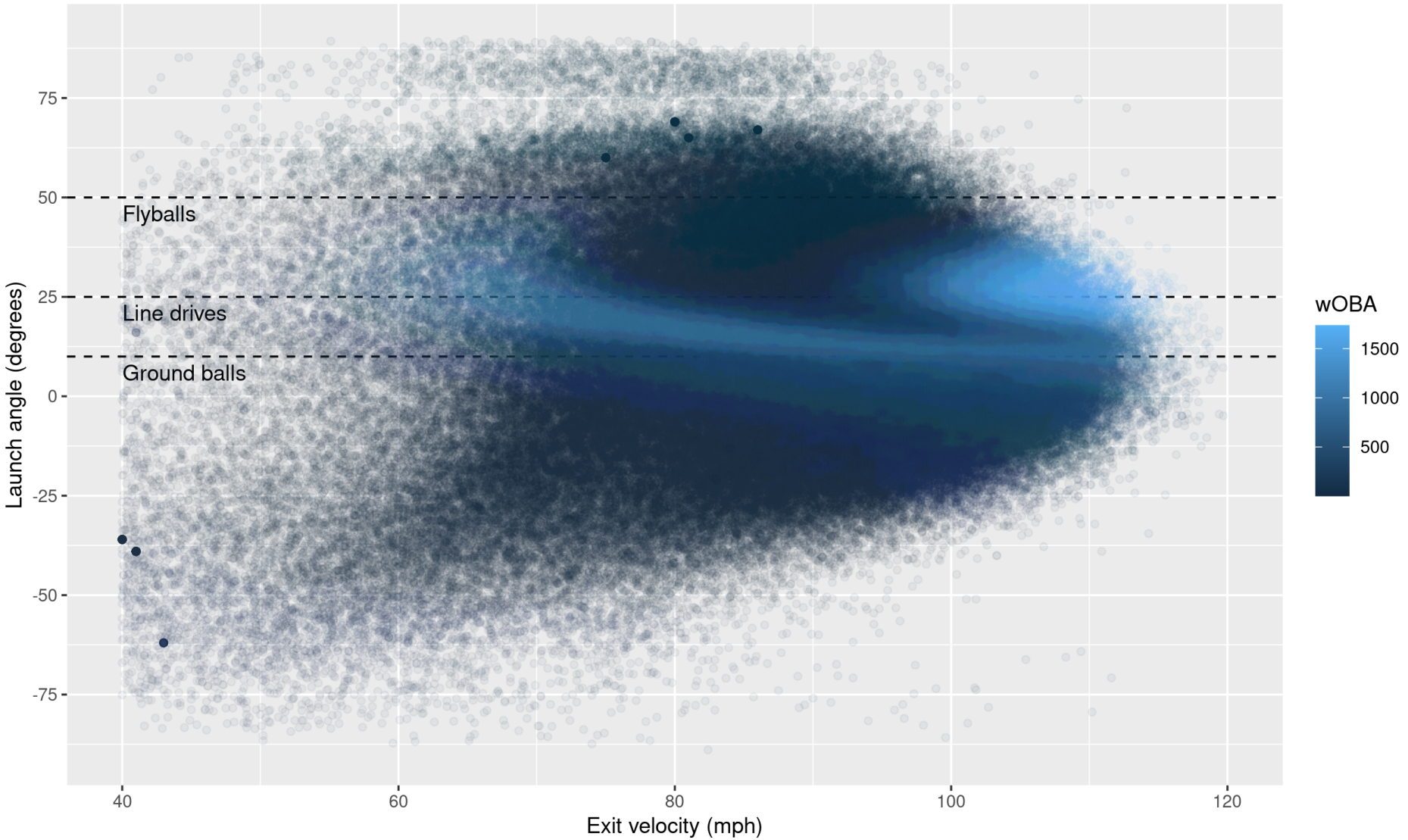
wOBA is designed to measure a player's overall offensive contributions per plate appearance.

- Factors in the results of hits.

```
guidelines <- tibble(
  launch_angle = c(10, 25, 50),
  launch_speed = 40,
  label = c("Ground balls", "Line drives", "Flyballs"))

wOBA_plot <- batter_hit %>%
  # for speed
  sample_n(nrow(.) / 2) %>%
  ggplot(aes(x = launch_speed, y = launch_angle,
             color = estimated_woba_using_speedangle)) +
  geom_hline(data = guidelines, aes(yintercept = launch_angle),
             color = "black", linetype = 2) +
  geom_text(data = guidelines,
            aes(label = label, y = launch_angle - 4),
            color = "black", hjust = "left") +
  geom_point(alpha = 0.05) +
  scale_x_continuous("Exit velocity (mph)", limits = c(40, 120)) +
  scale_y_continuous("Launch angle (degrees)", breaks = seq(-75, 75, 25)) +
  guides(color = guide_colorbar(title = "wOBA"))
```

wOBA as a Function of Hitting Characteristics

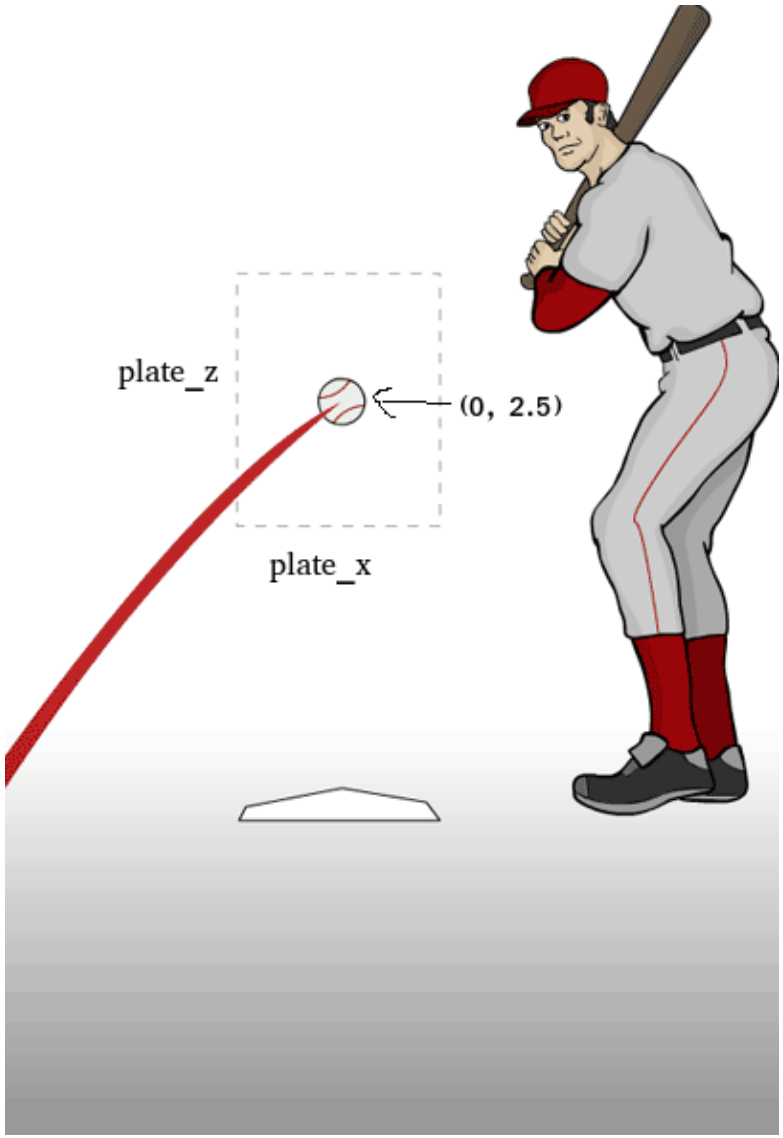
```
wOBA_plot
```



Modeling Strike Probability Based on Ball Position

```
batter_nohit <- batter %>%
  # Balls or Strikes in 2016 only
  filter(type %in% c("B","S"), year == "2015") %>%
  # Create response variable
  mutate(IsCalledStrike = ifelse(type == "S",1,0))

cl <- makeForkCluster(4)
mod <- bam(IsCalledStrike ~ s(plate_x, plate_z, by = stand),
           data = batter_nohit,
           family = "binomial", cluster = cl)
stopCluster(cl)
```

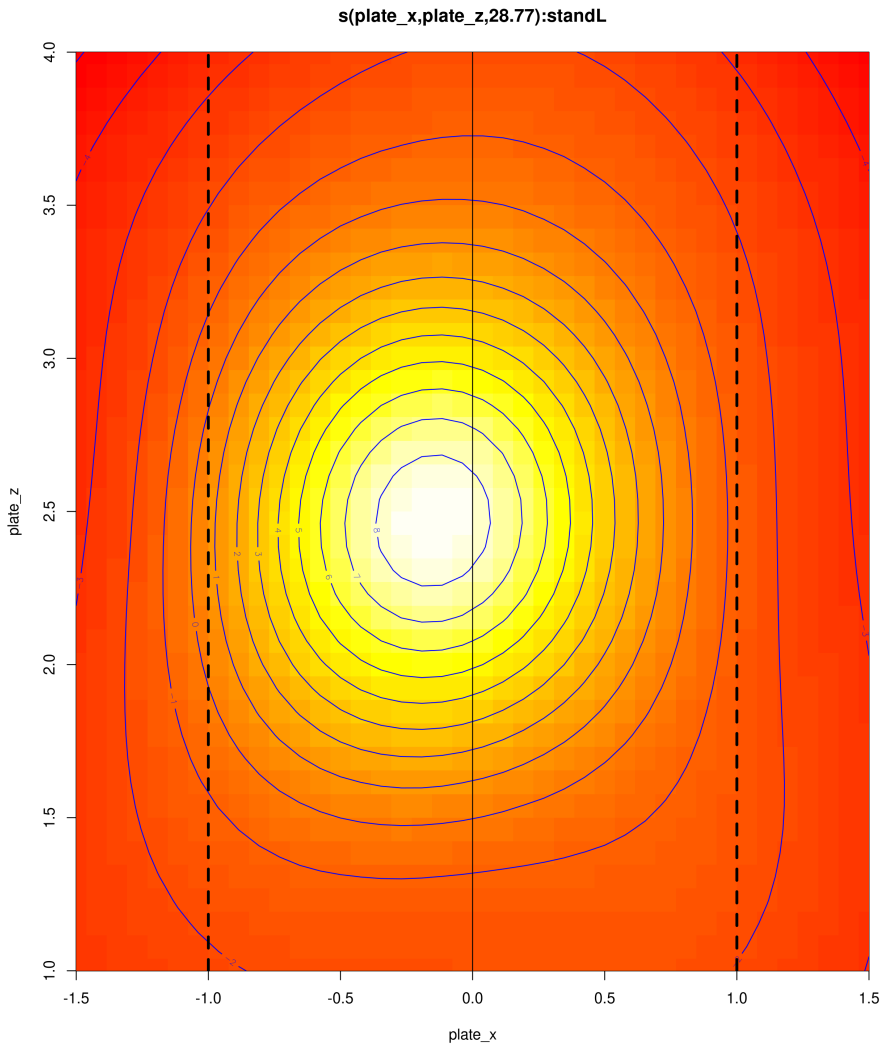
- This fits a *generalized additive model*.
  - Fits smooth functions of covariates.
  - **binomial** family indicates we are predicting probability of a two class problem.
  - Typically *gam()* is used, *bam()* is for large datasets.

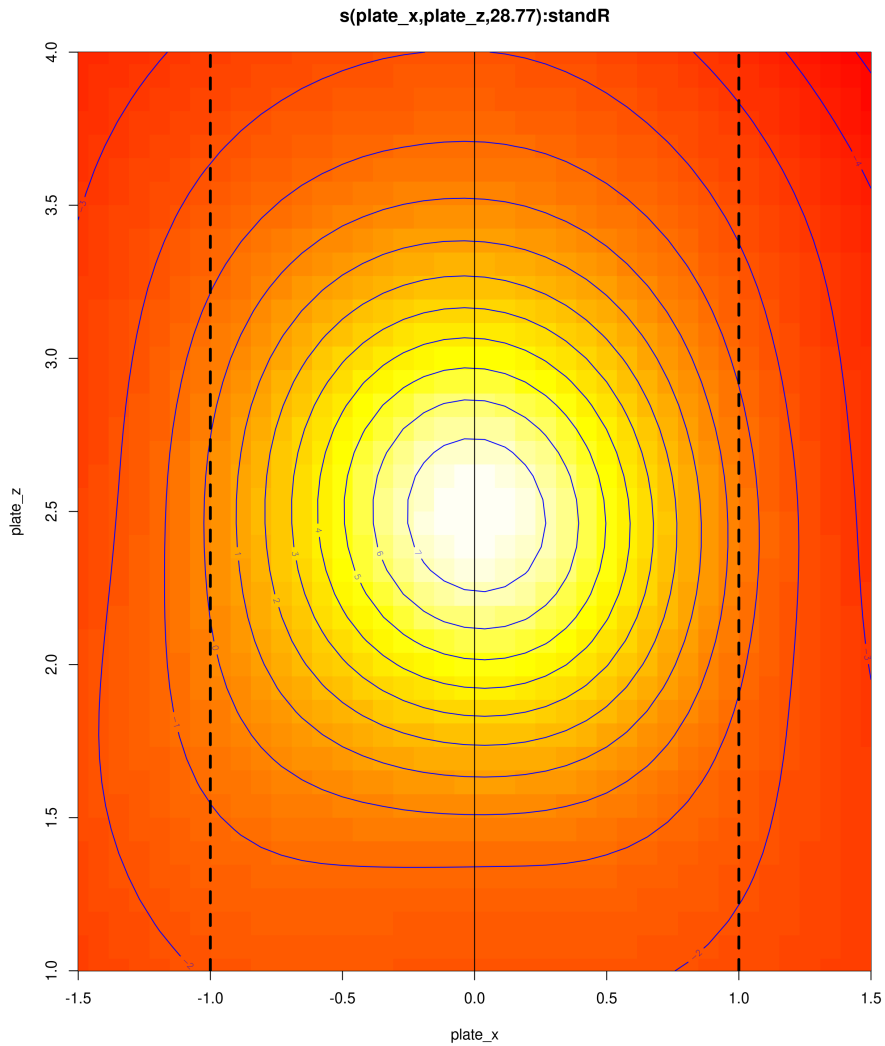


Strike Probability Based on Ball Position: 2015

```
plot(mod, scheme = 2,
     xlim=c(-1.5,1.5), ylim=c(1,4),
     select=1)
abline(v=c(0,-1,1), lty=c(1,2,2), lwd = c(1,3,3))
```

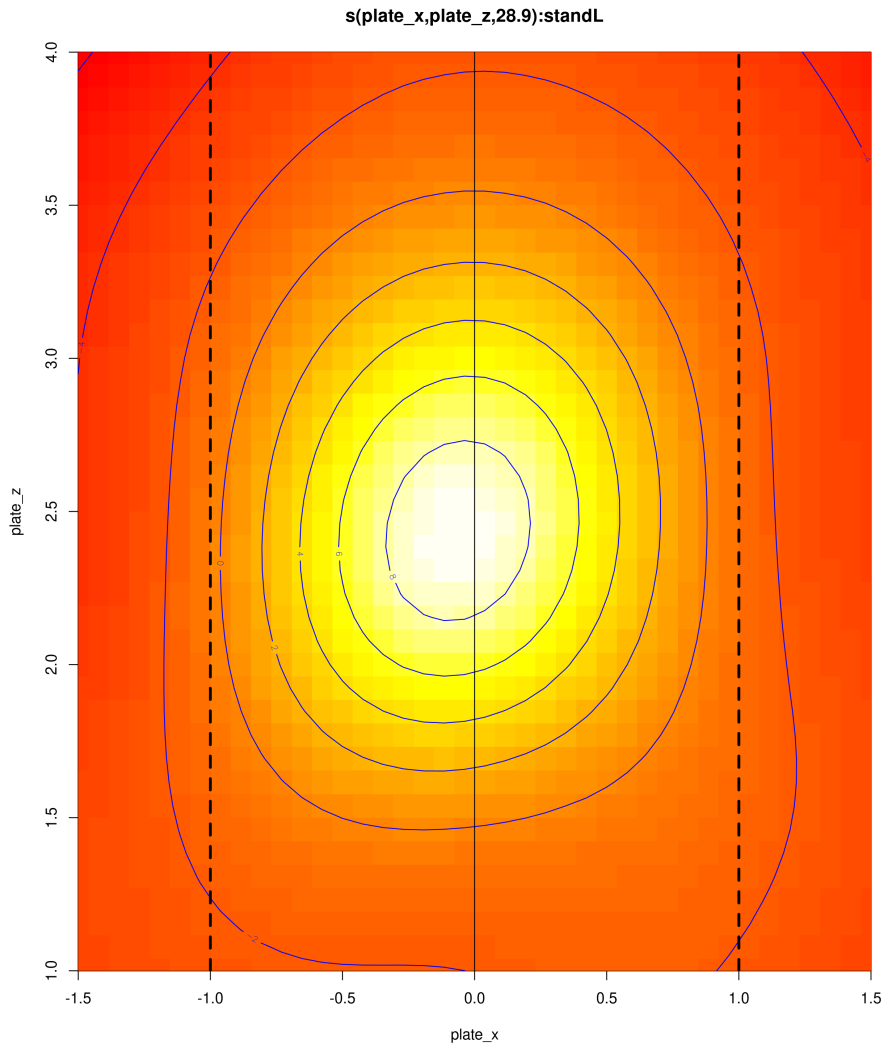
```
plot(mod, scheme = 2,
     xlim=c(-1.5,1.5), ylim=c(1,4),
     select=2)
abline(v=c(0,-1,1), lty=c(1,2,2), lwd = c(1,3,3))
```



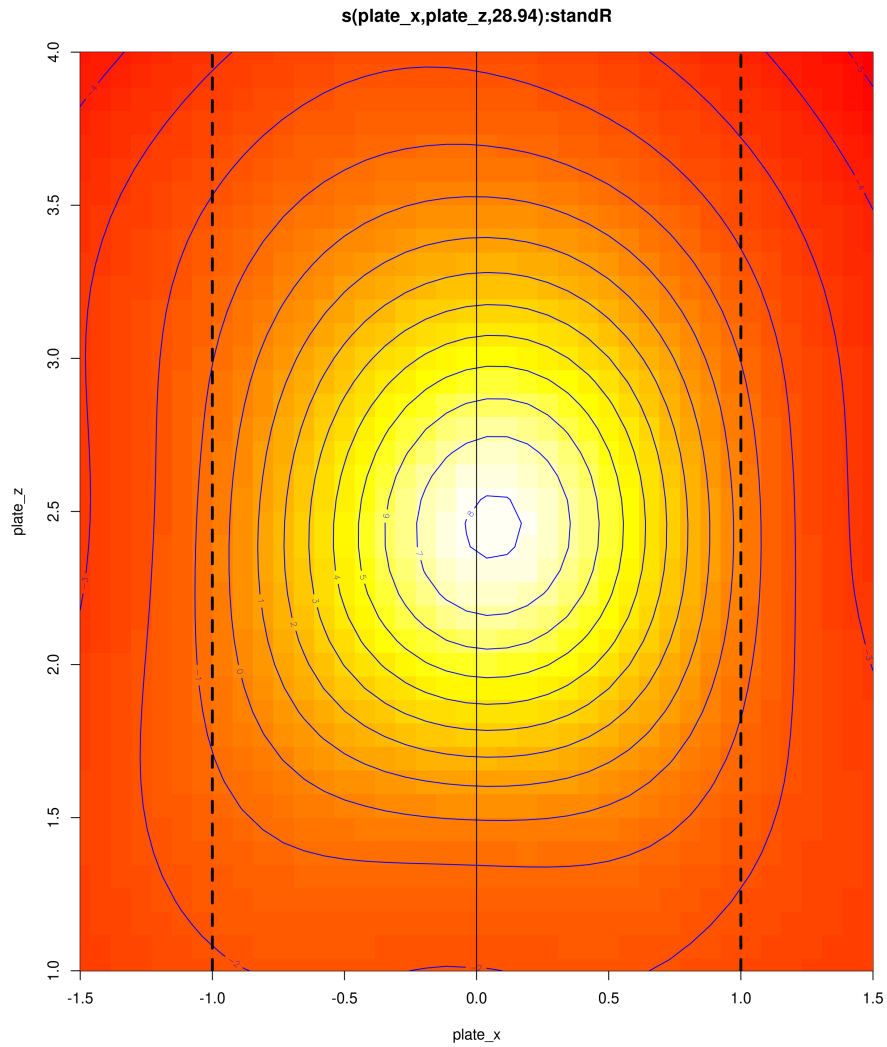


Strike Probability Based on Ball Position: 2019

```
plot(mod_2019, scheme = 2,
     xlim=c(-1.5,1.5), ylim=c(1,4),
     select=1)
abline(v=c(0,-1,1), lty=c(1,2,2), lwd = c(1,3,3))
```




```
plot(mod_2019, scheme = 2,
     xlim=c(-1.5,1.5), ylim=c(1,4),
     select=2)
abline(v=c(0,-1,1), lty=c(1,2,2), lwd = c(1,3,3))
```



Modeling Home Run Probability

Let's try to build a model to predict the probability of a home run as a function of:

- Batter's median launch angle.
- 75th percentile of batter's launch speeds.
- Various pitch characteristics.

```
batter_hit_summ <- batter_hit %>%
  group_by(batter) %>%
  summarise(N_Hits = n(),
            N_HR = sum(home_run == "Yes"),
            ExitSpeed75 = quantile(launch_speed,.75,na.rm=T),
            MedLaunchAngle = median(launch_angle))

batter_HR <- batter_hit %>%
  left_join(batter_hit_summ) %>%
  filter(N_Hits>300, N_HR > 50) %>%
  dplyr::select(home_run, ExitSpeed75, MedLaunchAngle, # batter properties
                plate_x,plate_z,
                # pitch velocity measurments
                vx0, vy0, vz0, ax, ay, az,
                release_spin_rate, release_extension,
                release_pos_x,release_pos_y,release_pos_z)
```

Modeling HR Probability: Rare Outcome

```
table(batter_hit$home_run)
```

```
##
##     No    Yes
## 618278  29417
```

Home runs only occur in roughly 5% of balls batted into play.

- Machine learning algorithms seek to achieve the best loss
  - With a rare event, the best loss will predict all plays *not* be a home run

```
n = nrow(batter_hit)
train.index = sample(n, 0.95*n)
train.data = as.matrix(batter_hit[train.index,])
train.label = label[train.index]
test.data = as.matrix(batter_hit[-train.index,])
test.label = label[-train.index]
```

We can penalize the xgboost optimization more for misclassifying home runs by specifying a *weight* variable.

```
weights <- rep(1,nrow(train.data))
weights[train.label==1] <- 20*weights[train.label==1]

# xgboost() takes input xgb.Matrix objects as input
xgb.train = xgb.DMatrix(data=train.data,label=train.label,
                        weight = weights)
xgb.test = xgb.DMatrix(data=test.data,label=test.label)
```

Modeling Home Run Probability: xgboost

- XGBoost is an extremely popular tree based approach to machine learning
  - Highly scale-able.
  - Innovative tree based approach.
  - Uses the Hessian in optimization
  - STL RUG - Mar '19

```
params <- list(
  booster = "gbtree",
  eta = 0.1,
  max_depth = 5,
  gamma = 3,
  objective = "binary:logistic",
  eval_metric = "auc")

# Train the XGBoost classifer
xgb.fit <- xgb.train(
  params = params,
  data = xgb.train,
  nrounds = 1000,
  early_stopping_rounds = 10,
  watchlist=list(val1 = xgb.train,
                 val2 = xgb.test),
  verbose = 1, print_every_n = 5)
```

Modeling HR Probability: Results

```
# Predict outcomes with the test data
xgb_prob = predict(xgb.fit,test.data,reshape=T)
xg_prediction = ifelse(xgb_prob>.5,"Yes","No")
test_label = levels(home_run)[test.label+1]
```

```
confusionMatrix(as.factor(xg_prediction), as.factor(test_label))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  8020  195
##        Yes 7045  691
##
##                Accuracy : 0.5461
##                  95% CI : (0.5383, 0.5539)
##     No Information Rate : 0.9445
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0673
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.53236
##             Specificity : 0.77991
##          Pos Pred Value : 0.97626
##          Neg Pred Value : 0.08932
##              Prevalence : 0.94445
##          Detection Rate : 0.50279
##    Detection Prevalence : 0.51501
##       Balanced Accuracy : 0.65613
##
##        'Positive' Class : No
##
```

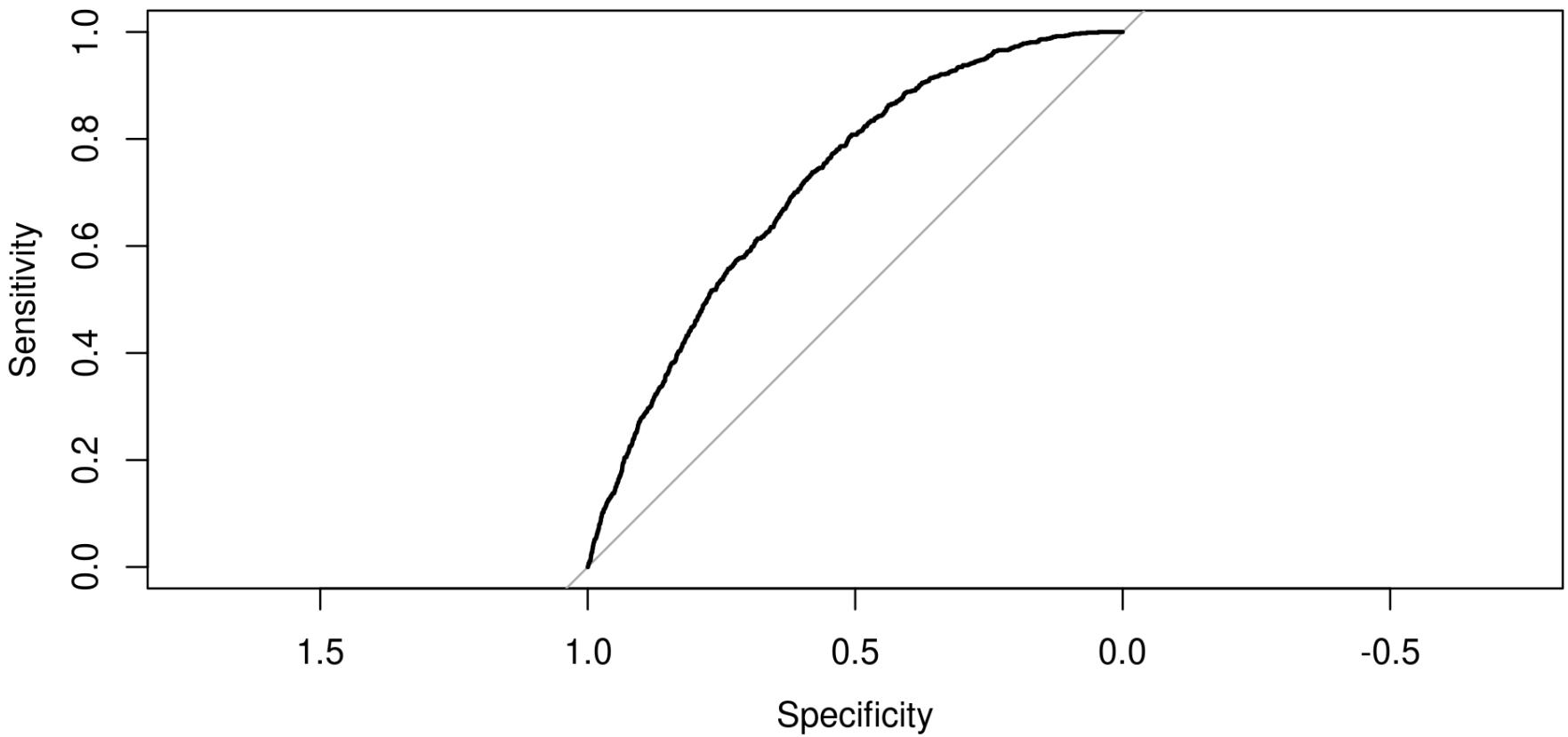
Modeling HR Probability: Results (ROC/AUC)

ROC curve's tell us how well a classification model balances errors.

```
auc(roc_test)
```

```
## Area under the curve: 0.7155
```

```
plot(roc_test)
```



Modeling HR Probability: Partial Dependence Plots

We can extract marginal effects from black box models with *partial dependence plots*.

- Partial Dependence Plots
  - STL RUG - Dec '19
- Visualize the effect of variables on predicted values.
- Friedman 2001
  - Popular, commonly integrated into software.

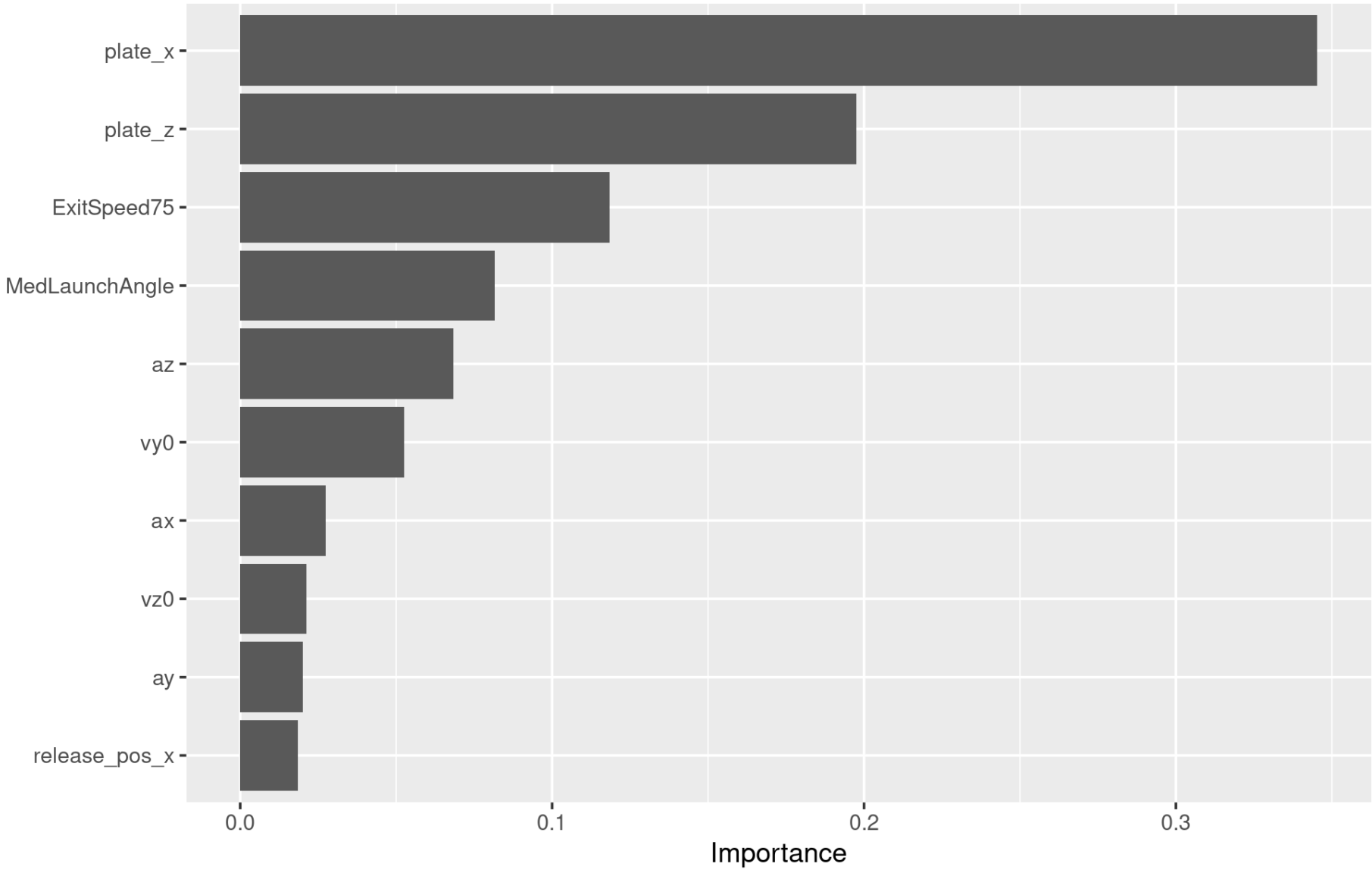
Idea:

1. Pick one or two variables.
2. Average out the effect of all other variables for a given value of our selected value.

```
p_vip <- vip(xgb.fit, num_features = 10)

p_exitspeed <- partial(xgb.fit,
                       pred.var = c("ExitSpeed75"),
                       plot = TRUE,
                       plot.engine = "ggplot2",
                       train = train.data)
p_launch <- partial(xgb.fit,pred.var=c("MedLaunchAngle")
                    plot = TRUE,
                    plot.engine = "ggplot2",
                    train = train.data)
p_platex <- partial(xgb.fit, pred.var = c("plate_x"),
                    plot = TRUE,
                    plot.engine = "ggplot2",
                    train = train.data)
p_platez <- partial(xgb.fit, pred.var = c("plate_z"),
                    plot = TRUE,
                    plot.engine = "ggplot2",
                    train = train.data)
p_az <- partial(xgb.fit, pred.var = c("az"),
                plot = TRUE,
                plot.engine = "ggplot2",
                train = train.data)
p_vy <- partial(xgb.fit, pred.var = c("vy0"),
                plot = TRUE,
                plot.engine = "ggplot2",
                train = train.data)
```

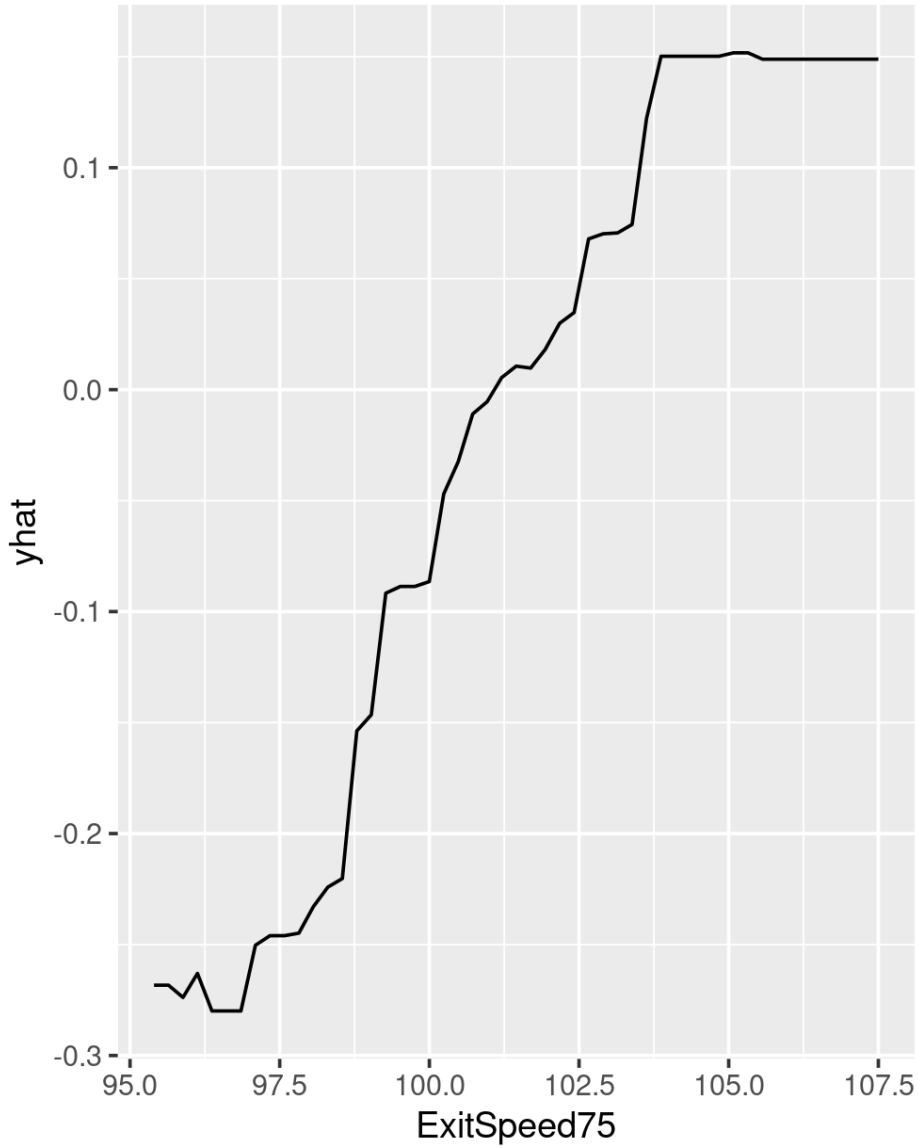
Modeling HR Probability: Partial Dependence Plots

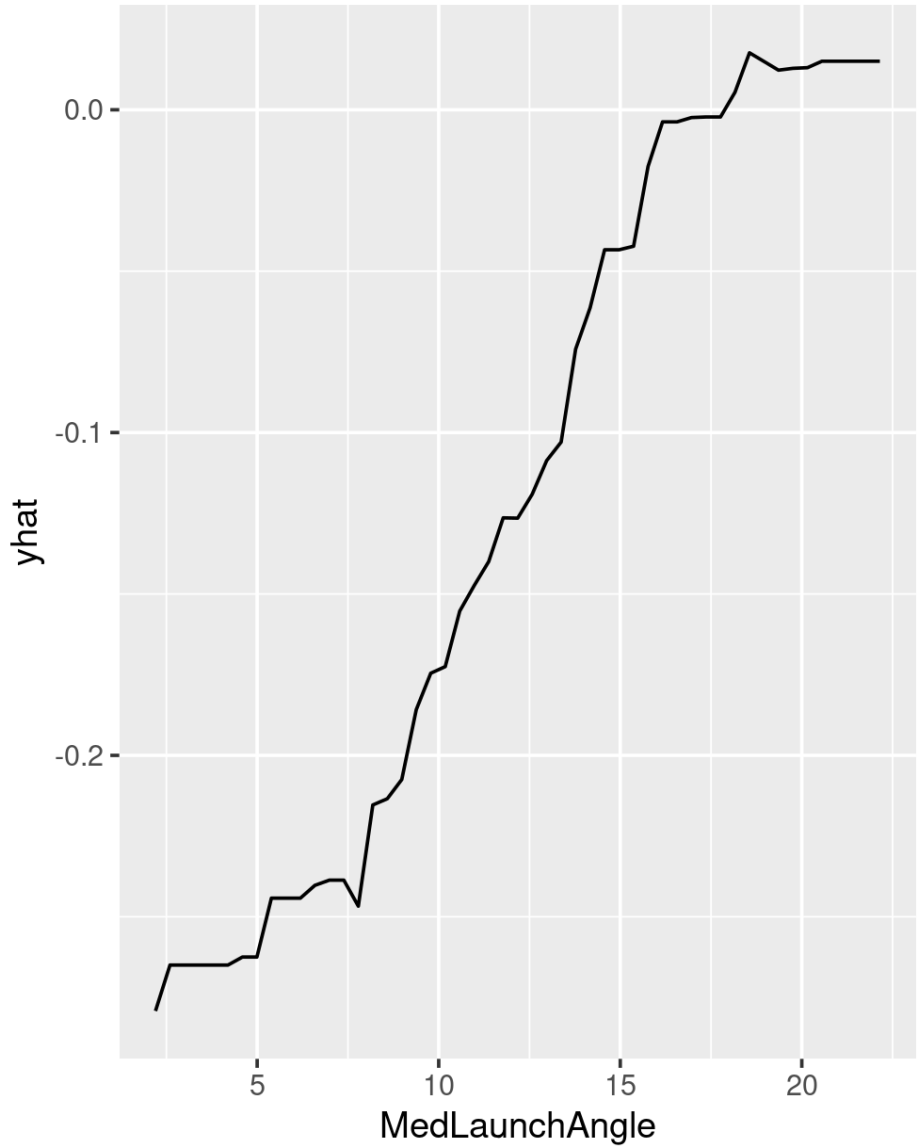
```
print(p_vip)
```



Modeling HR Probability: Partial Dependence Plots

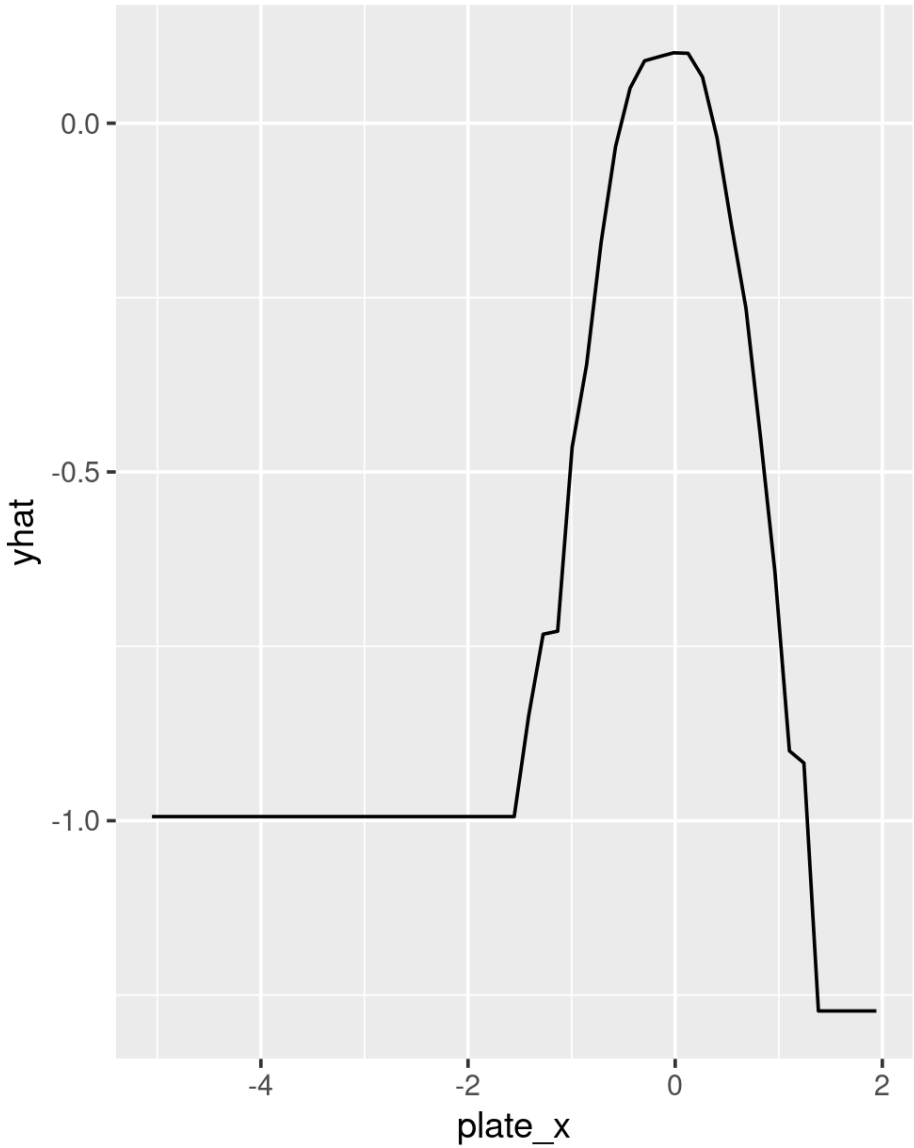
```
grid.arrange(p_exitspeed, p_launch, nrow = 1)
```

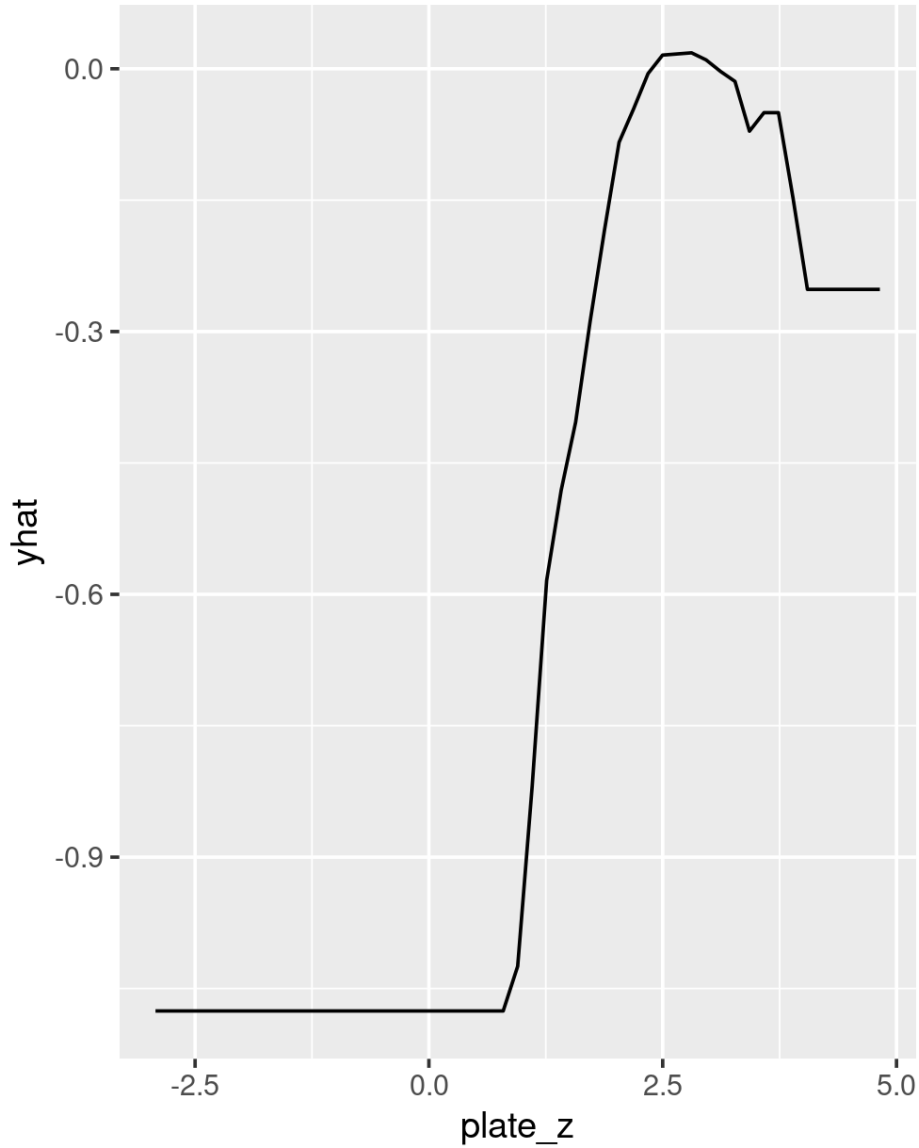




Modeling HR Probability: Partial Dependence Plots

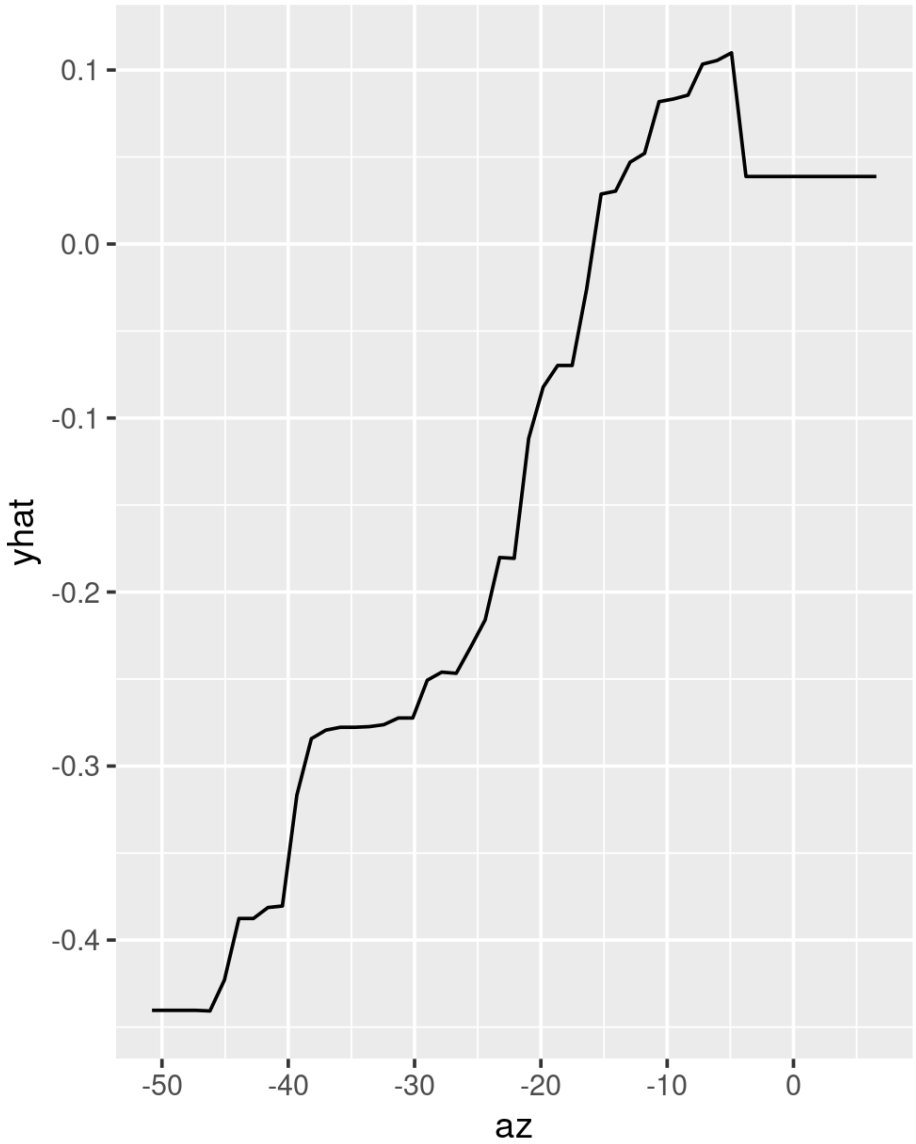
```
grid.arrange(p_platex, p_platez, nrow = 1)
```

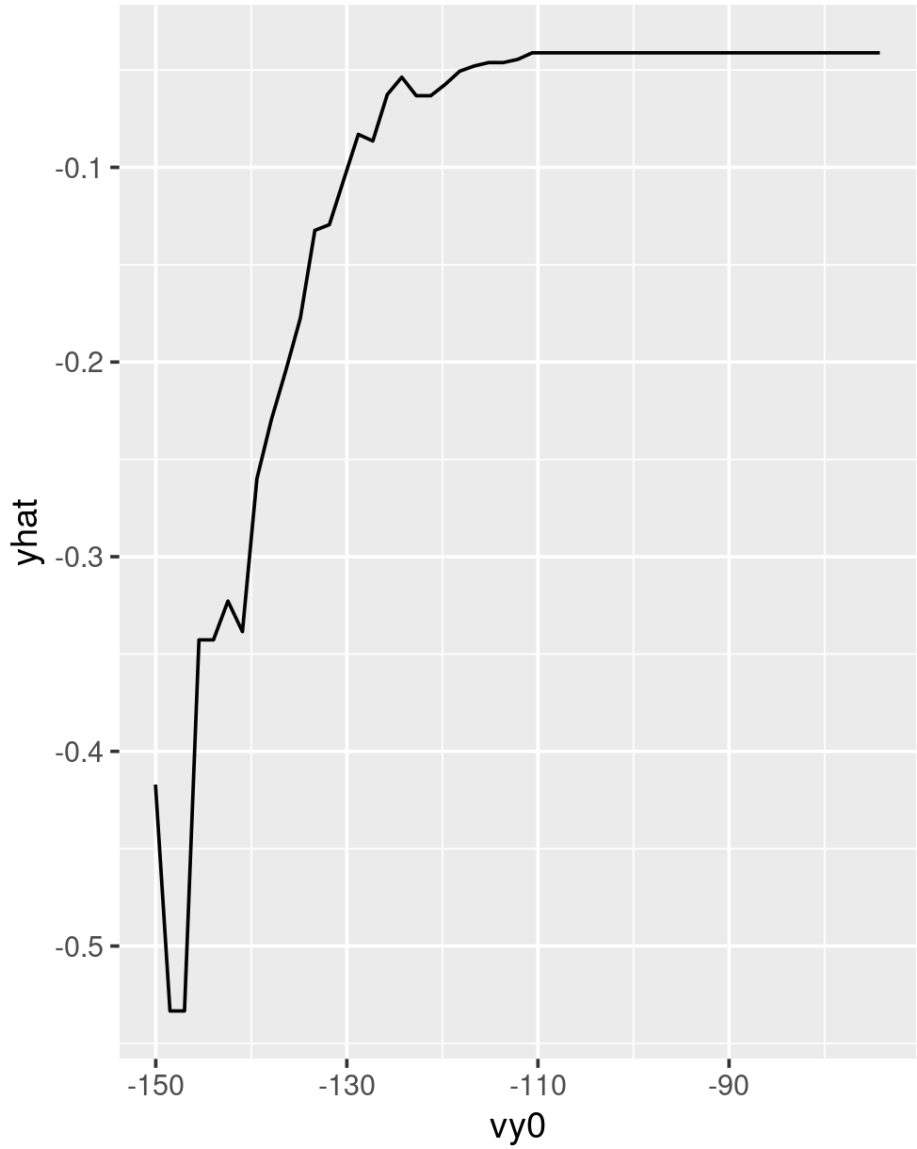




Modeling HR Probability: Partial Dependence Plots

```
grid.arrange(p_az, p_vy, nrow = 1)
```





Further Ideas on how to use Statcast Data

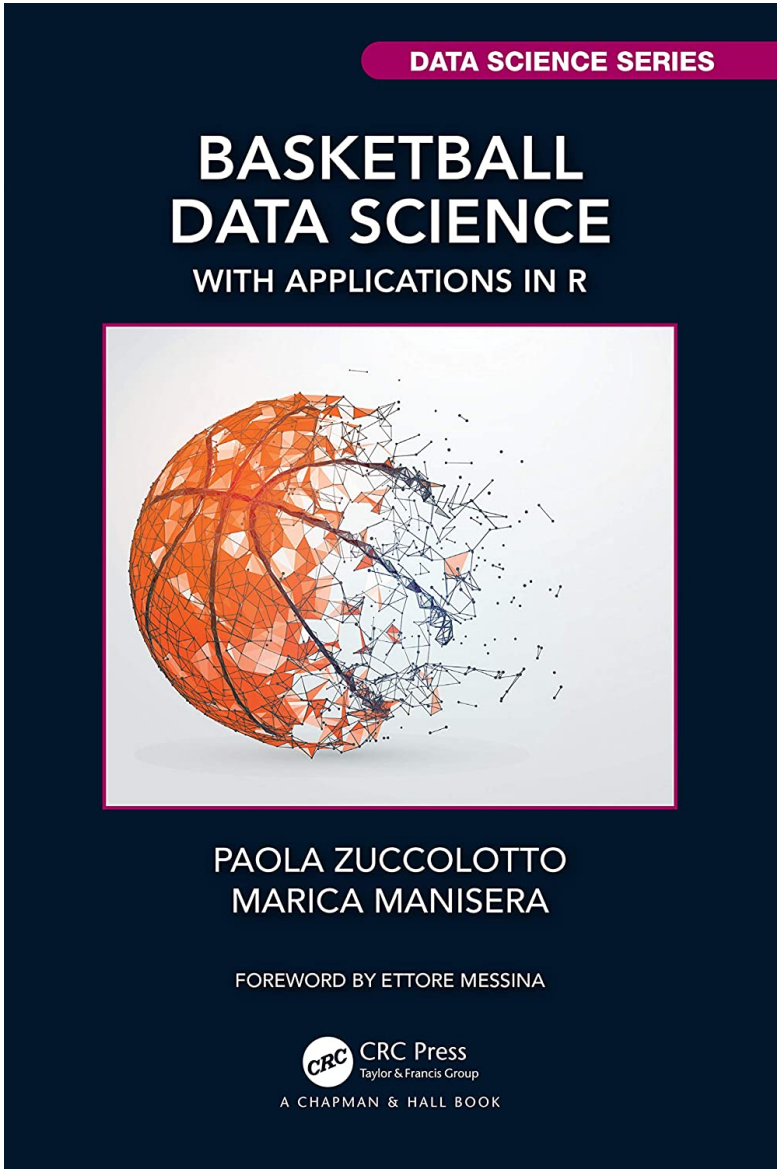
This data has almost limitless potential

- Home run prediction
  - Refine the home run model to incorporate other/engineered features.
  - Model distance as a proxy for home run likelihood.
  - Use mixed effects models with *lmer()* to remove variation before running a ML model.
- Strike zone variance with different counts.
  - Strike zone widens on 3-0 and 2-0 counts, shrinks on 0-2 counts.
- Take individual players in fielding positions into account.
- Factor weather into predictions

Basketball

```
# devtools::install_github("sndmrc/BasketballAnalyzeR")
library(BasketballAnalyzeR)
```

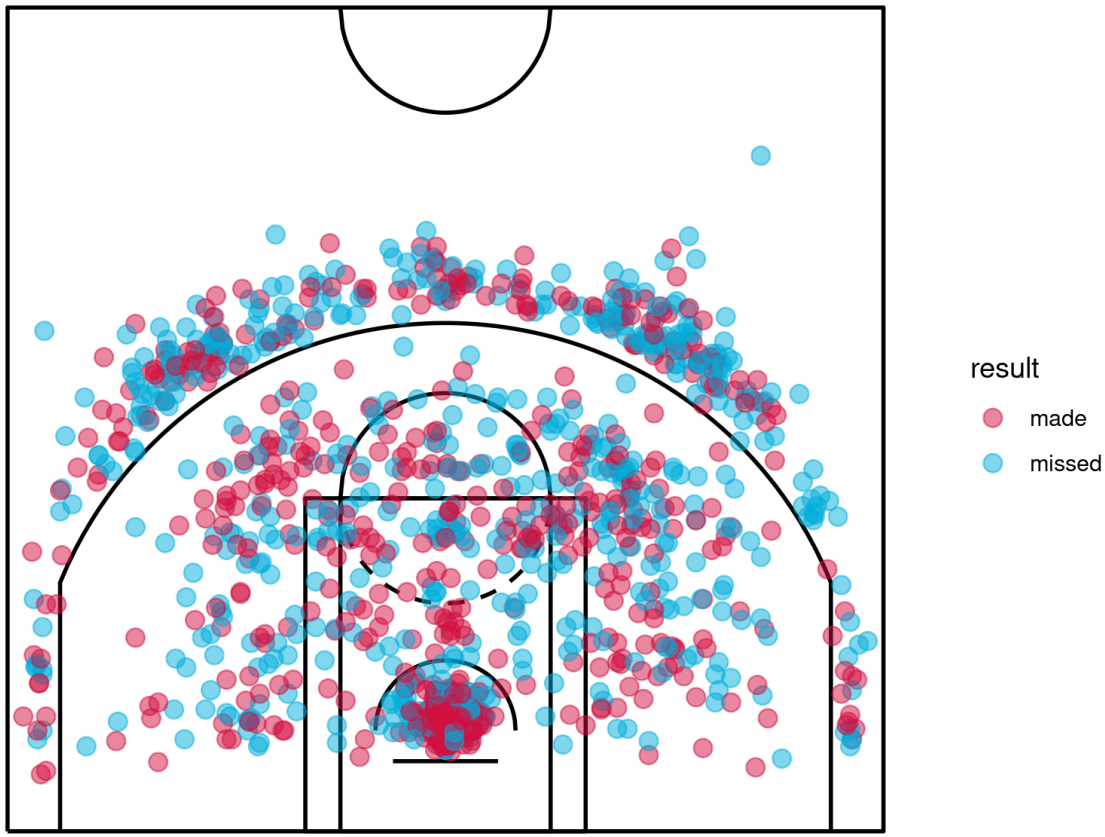
- Published January 2020
- Many high level functions to explore
  - Plotting
  - Pulling Data
- Website has all R code
  - bdsports.unibs.it



High Level Plot Generation

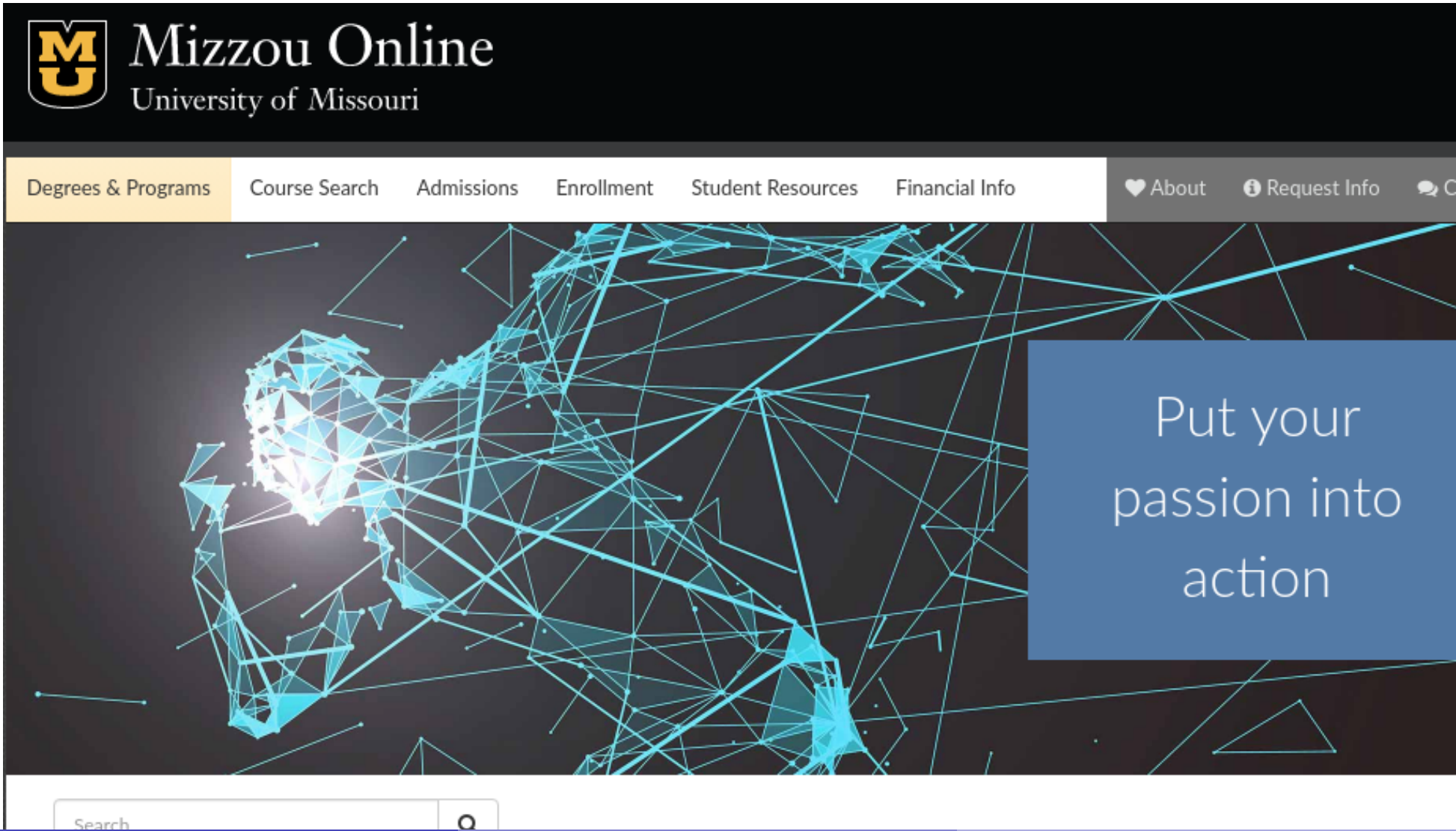
The package has built in datasets and functions to visualize them

```
PbP <- PbPmanipulation(PbP.BDB)
subdata <- subset(PbP, player=="Kevin Durant")
subdata$xx <- subdata$original_x/10
subdata$yy <- subdata$original_y/10-41.75
shotchart(data=subdata, x="xx", y="yy", scatter=TRUE, z="result")
```

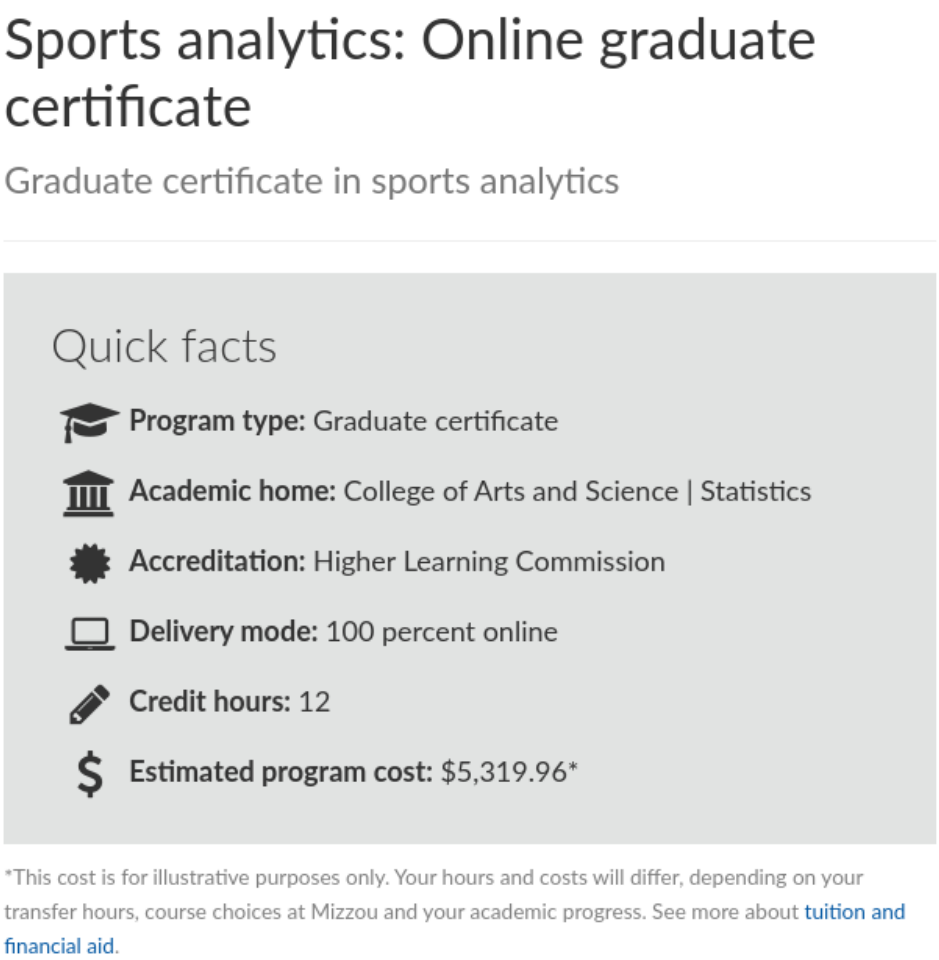


Sports Certificate

University of Missouri - Columbia



Sports Certificate



Sports analytics: Online graduate certificate

Graduate certificate in sports analytics

Quick facts

Program type: Graduate certificate

Academic home: College of Arts and Science | Statistics

Accreditation: Higher Learning Commission

Delivery mode: 100 percent online

Credit hours: 12

Estimated program cost: $5,319.96*

*This cost is for illustrative purposes only. Your hours and costs will differ, depending on your transfer hours, course choices at Mizzou and your academic progress. See more about tuition and financial aid.

Graduate and undergraduate level certifications

- Link: Undergraduate
- Link: Graduate

4 courses in total

- 2 Statistical/machine learning focused
- 2 Sports analytics focused