STAT 528 - Advanced Regression Analysis II

Multinomial response regression (part I)

Daniel J. Eck
Department of Statistics
University of Illinois

Last time

- Overdispersion
- Gala data analysis
- Negative Binomial regression
- Zero Inflated Count Models

Learning Objectives Today

- Multinomial regression via baseline-category logistic model
- Data analysis

This slide deck will be a little bit different. It will only contain the data analysis. The lecture will be largely on the blackboard.

Baseball example

In the game of baseball it is important to understand how the characteristics of a player's swing translate to positive outcomes on the baseball field. We want to estimate the probabilities of different baseball outcomes given quality of swing and hitting tendency variables (obtained by STATCAST).

The outcomes of interest are:

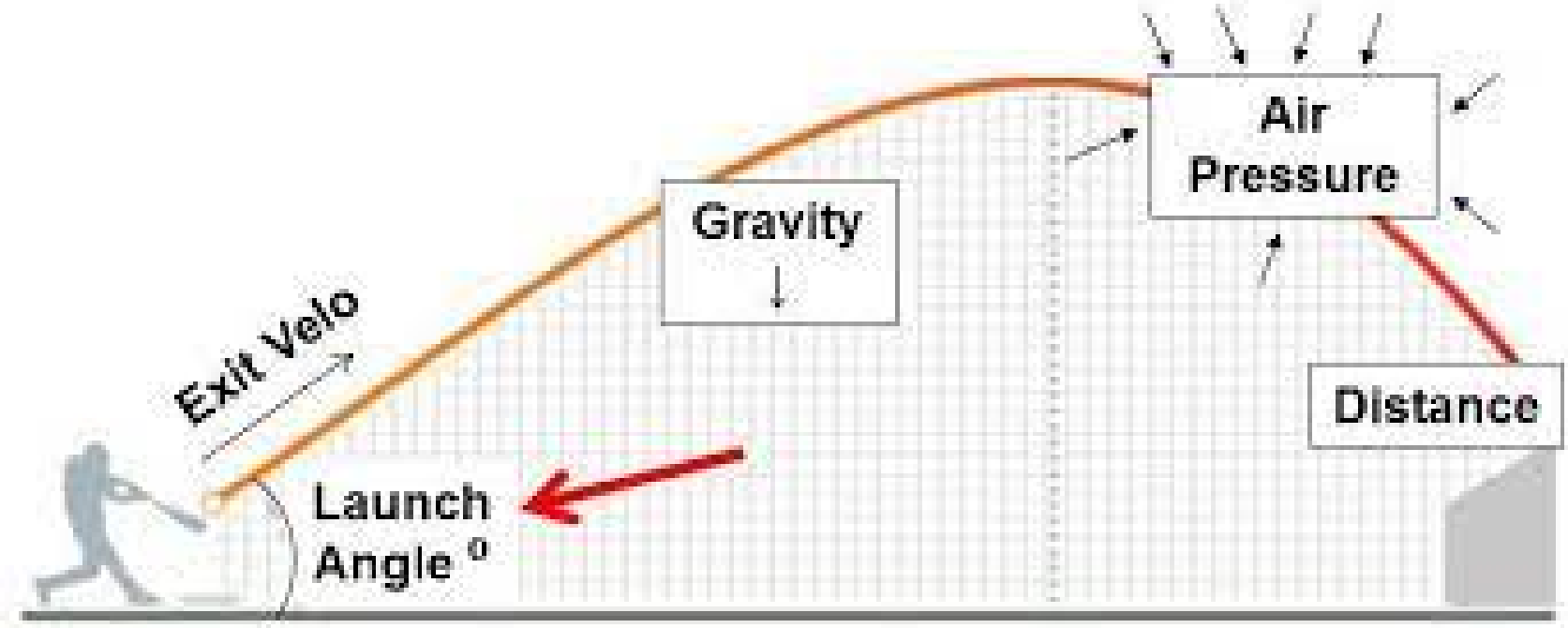
- single (1 base)
- double (2 bases)
- triple (3 bases)
- home runs (4 bases)
- and outs (0 bases; taken as baseline)

The more bases the better, 4 bases $=$ 1 run.

The team with the most runs wins the game. An out (0 bases) is a bad outcome for a batter, an out means the batter ended their time at the plate without reaching base.

We want to estimate the probabilities of singles, doubles, triples, home runs, and outs as a function of swing and batting tendency variables are:

- exit velocity (launch speed off the bat)
- launch angle (angle off the bat into the air)
- spray angle (where on the field the ball is going)


Air
Pressure
Gravity
Exit Velo
Distance
Launch
Angle °

We will use the baseline-category logistic model for this example.
We encoded events so that outs is the baseline category.

We will build models using AIC, LRT, and domain knowledge.

We now load in the relevant software and data set and display the first 10 rows of the data set

```
library(VGAM)  # has model-fitting functions
bball <- read.csv("bball.csv")
bball$events <- as.factor(bball$events)
dim(bball)
```

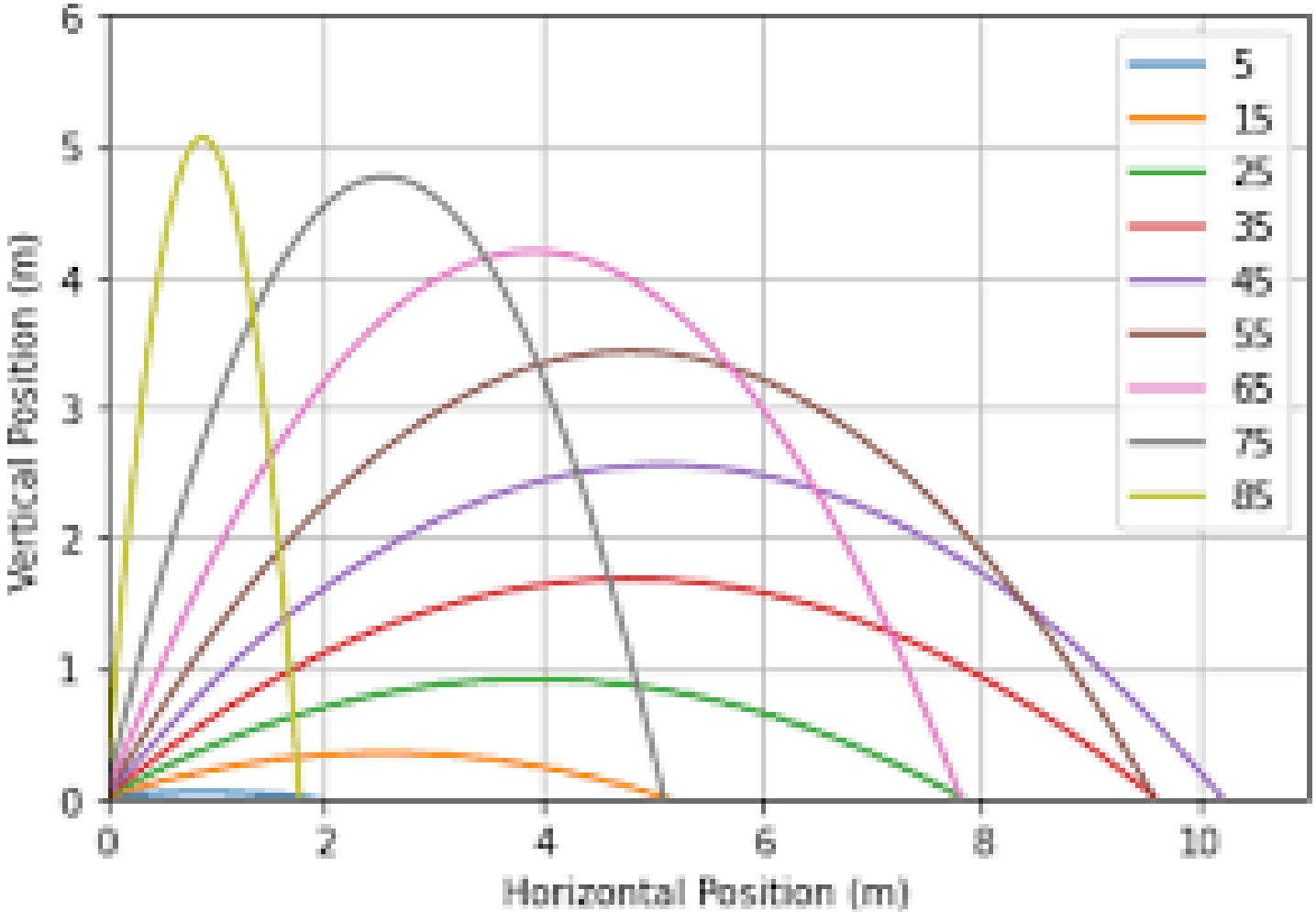
```
## [1] 50000     6
```

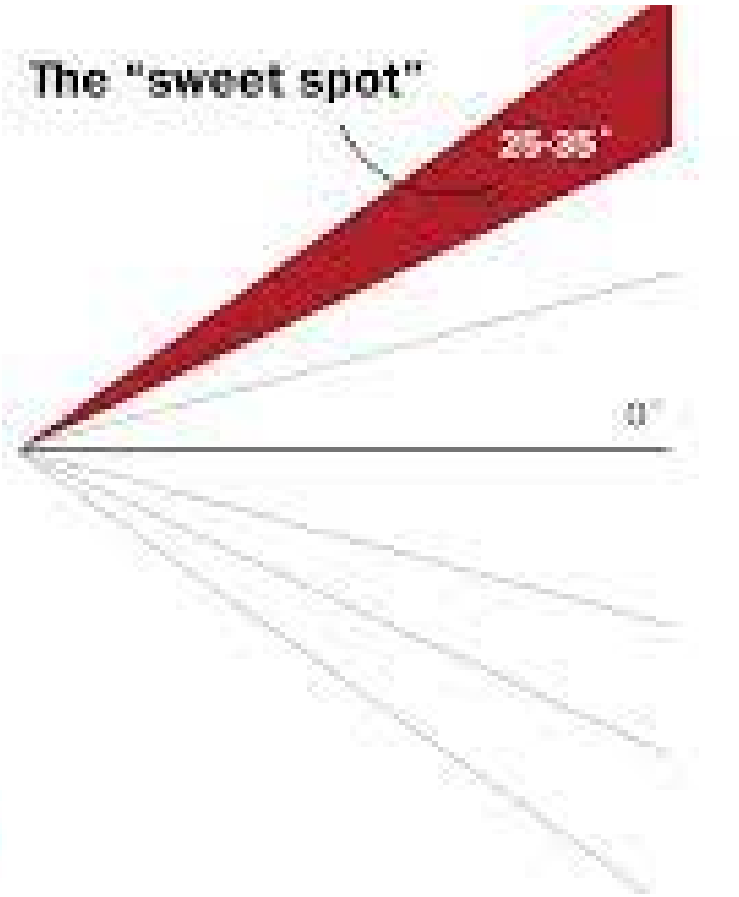
```
head(bball, 10)
```

```
##    pitcher batter events launch_speed launch_angle spray_angle
## 1   623167 493329    out         82.3          -41   2.4599306
## 2   592761 573131     b2         95.9           22 -12.0705545
## 3   429722 592206     b2        101.0           13 -45.5422457
## 4   571760 621005    out         61.9           63  12.9786747
## 5   519008 663757    out         93.2            5 -40.1840254
## 6   456501 672695    out         95.3           26  -9.4373687
## 7   605182 521692    out         74.9           35  -3.1754445
## 8   502624 444482    out         57.9          -28  -0.5256346
## 9   596133 595777    out         96.8          -23  45.8961936
## 10  579328 545358     b4        101.9           28  -0.9735690
```

We will fit two baseline category logistic models (nested) using a `vglm` from the `VGAM` package.

One of these models contains a square term for launch angle, why?


Vertical Position (m)
Horizontal Position (m)
0
1
2
3
4
5
6
0
2
4
6
8
10
5
15
25
35
45
55
65
75
85


The "sweet spot"
25-35°
0°

The model with a quadratic term for launch angle fits the data better.

```
mod1 <- vglm(events ~ launch_speed + launch_angle + spray_angle,
             family=multinomial, data=bball)
system.time(mod2 <- vglm(events ~ launch_speed + launch_angle + spray_angle +
               I(launch_angle^2),
             family=multinomial, data=bball))
```

```
##    user  system elapsed
##   2.637   0.311   2.956
```

```
AIC(mod2)
```

```
## [1] 70914.6
```

```
AIC(mod1)
```

```
## [1] 83119.98
```

```
pchisq(deviance(mod2), df.residual(mod2), lower = FALSE)
```

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

Model fitting is slow. Let's try scaling the predictors. Scaling does not change the model but it may speed up the optimization.

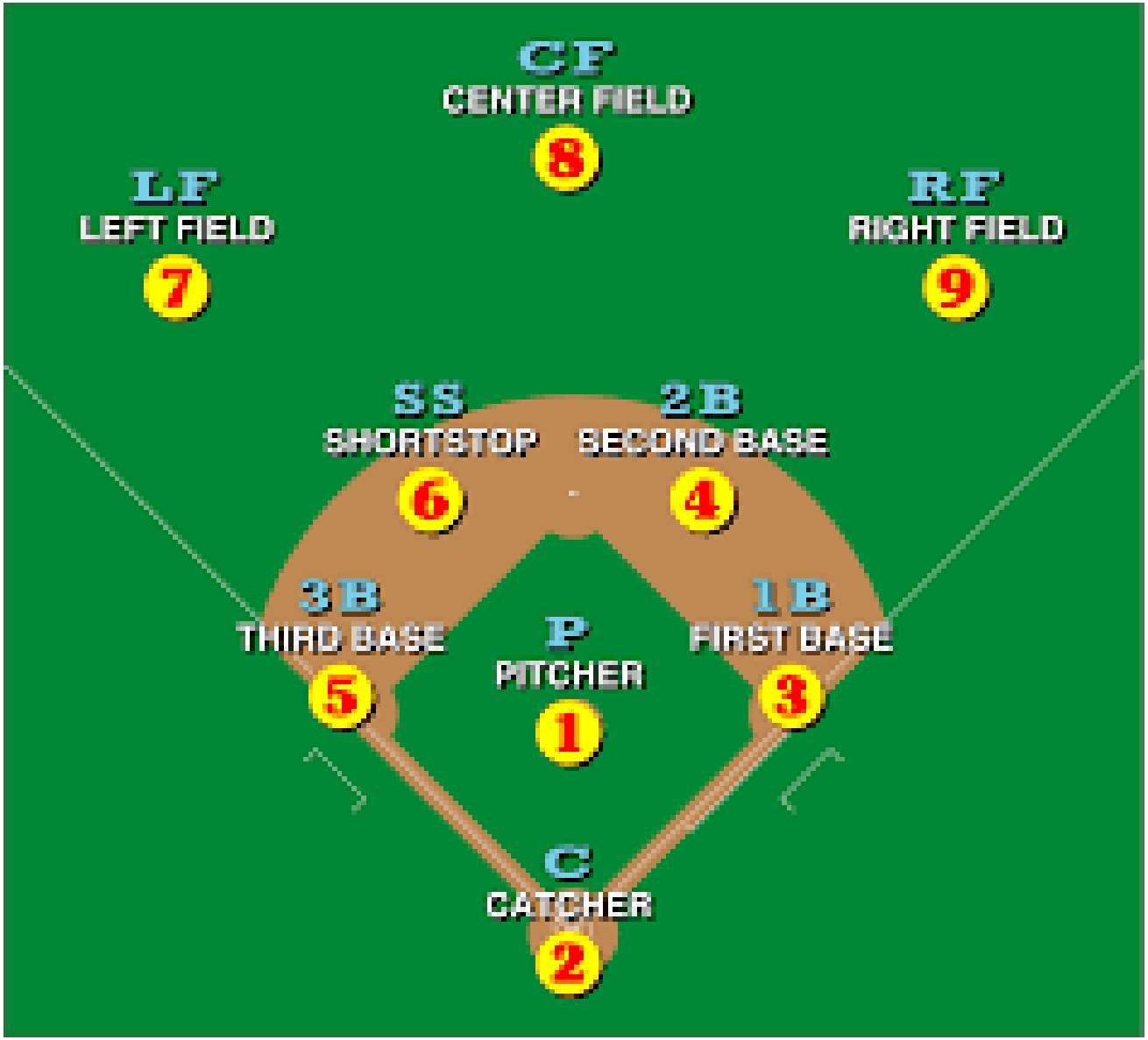
```
bball_scale <- bball
bball_scale[, 4:6] <- scale(bball[, 4:6])

system.time(mod2_scale <- vglm(events ~ launch_speed + launch_angle +
    spray_angle +  I(launch_angle^2),
    family=multinomial, data=bball_scale))
```

```
##    user  system elapsed
##   2.867   0.267   3.135
```

```
round(AIC(mod2), 4) == round(AIC(mod2_scale), 4)
```

```
## [1] TRUE
```

What type of polynomial should we consider for spray angle?


CF
CENTER FIELD
8
LF
LEFT FIELD
7
RF
RIGHT FIELD
9
SS
SHORTSTOP
6
2B
SECOND BASE
4
3B
THIRD BASE
5
P
PITCHER
1
1B
FIRST BASE
3
C
CATCHER
2

How about a 6th order polynomial?

```
system.time(mod3 <- vglm(events ~ launch_speed + launch_angle + spray_angle +
              I(launch_angle^2) + I(spray_angle^2) + I(spray_angle^3) +
              I(spray_angle^4) + I(spray_angle^5) + I(spray_angle^6),
            family=multinomial, data=bball))
```

```
##    user  system elapsed
##   8.895   0.820   9.742
```

```
system.time(mod3_scale <- vglm(events ~ launch_speed + launch_angle + spray_angle +
              I(launch_angle^2) + I(spray_angle^2) + I(spray_angle^3) +
              I(spray_angle^4) + I(spray_angle^5) + I(spray_angle^6),
            family=multinomial, data=bball_scale))
```

```
##    user  system elapsed
##   8.149   0.646   8.801
```

Scaling the predictors did not dramatically improve computational time.

Our 6th order polynomial model for spray angle fits the data better than the model with a single linear term.

```
AIC(mod3); AIC(mod2)
```

```
## [1] 66678.96
```

```
## [1] 70914.6
```

```
pchisq(deviance(mod3), df.residual(mod3), lower = FALSE)
```

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

We could try larger polynomial models, but numerical precision becomes problematic. Scaling did not alleviate this issue.

We now consider some interaction terms.

```
system.time(mod4 <- vglm(events ~ launch_speed + launch_angle + spray_angle +
                I(launch_angle^2) +
                I(spray_angle^2) + I(spray_angle^3) + I(spray_angle^4) +
                I(spray_angle^5) + I(spray_angle^6) +
                I(spray_angle*launch_angle) + I(spray_angle*launch_speed) +
                I(launch_angle*launch_speed),
              family=multinomial, data=bball))
```

```
##    user  system elapsed
##   9.368   0.673  10.055
```

```
AIC(mod3); AIC(mod4)
```

```
## [1] 66678.96
```

```
## [1] 66325.19
```

```
pchisq(deviance(mod4), df.residual(mod4), lower = FALSE)
```

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

Let's now examine what we can do with this flexible final model.

We plot the log $\left(\frac{\hat{\pi}_j(x)}{\hat{\pi}_J(x)}\right)$ as a function of spray angle for each hit outcome where J corresponds to an out.

We will set launch angle and exit velocity to their median values.

```
summary(bball$launch_angle)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -89.00   -6.00   12.00   12.29   30.00   89.00
```

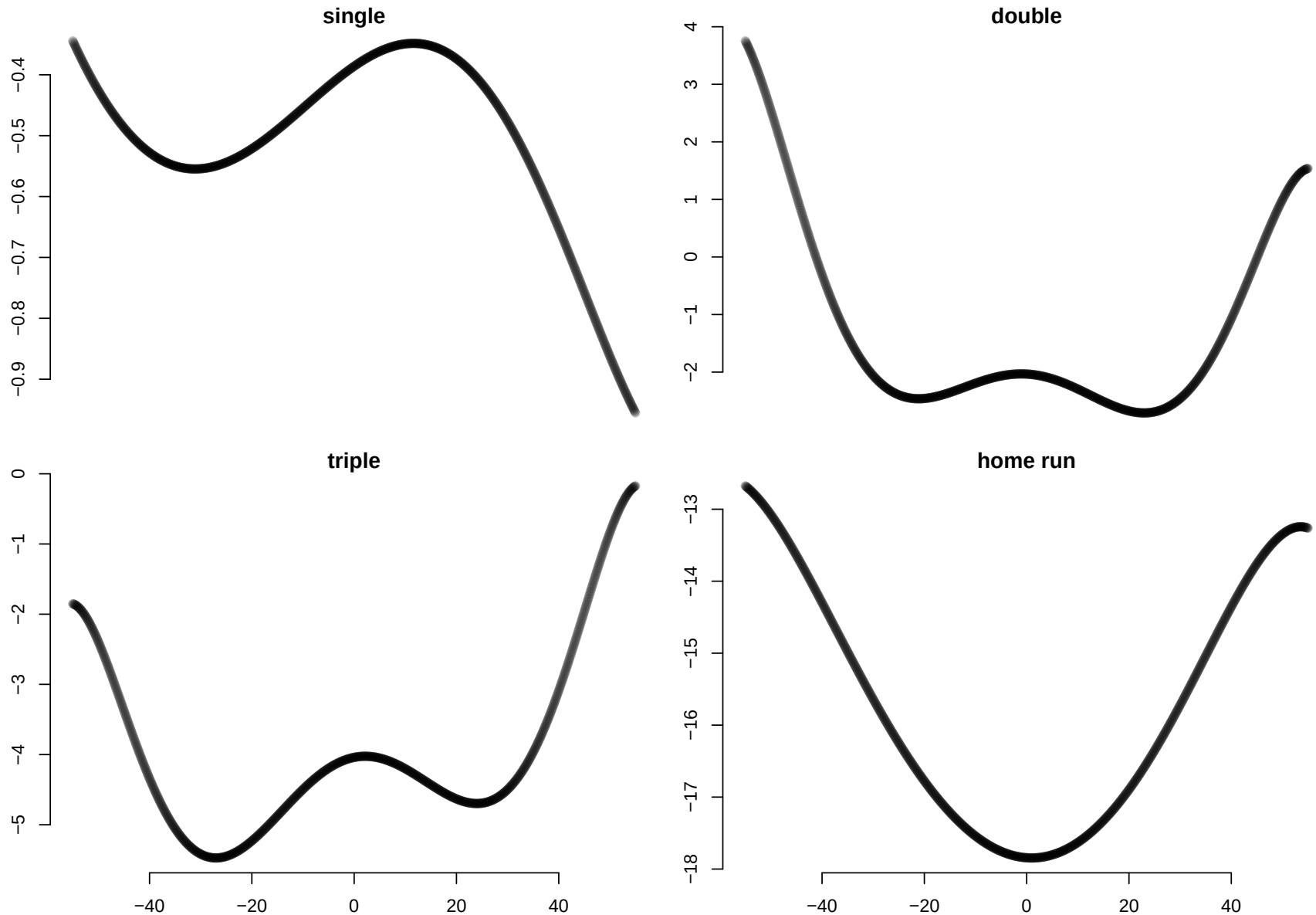
```
summary(bball$launch_speed)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.40   80.00   90.30   88.07   98.80  121.30
```

```
## obtain predictions
new_data = data.frame(spray_angle = seq(-55,55, by = 0.1),
           launch_speed = 90.30,
           launch_angle = 12)
pred <- predict(mod4, newdata = new_data)
head(pred, 3)
```

```
##   log(mu[,1]/mu[,5]) log(mu[,2]/mu[,5]) log(mu[,3]/mu[,5]) log(mu[,4]/mu[,5])
## 1         -0.3442213           3.755072          -1.852540          -12.67547
## 2         -0.3460507           3.737278          -1.854847          -12.68112
## 3         -0.3478719           3.719141          -1.857601          -12.68688
```

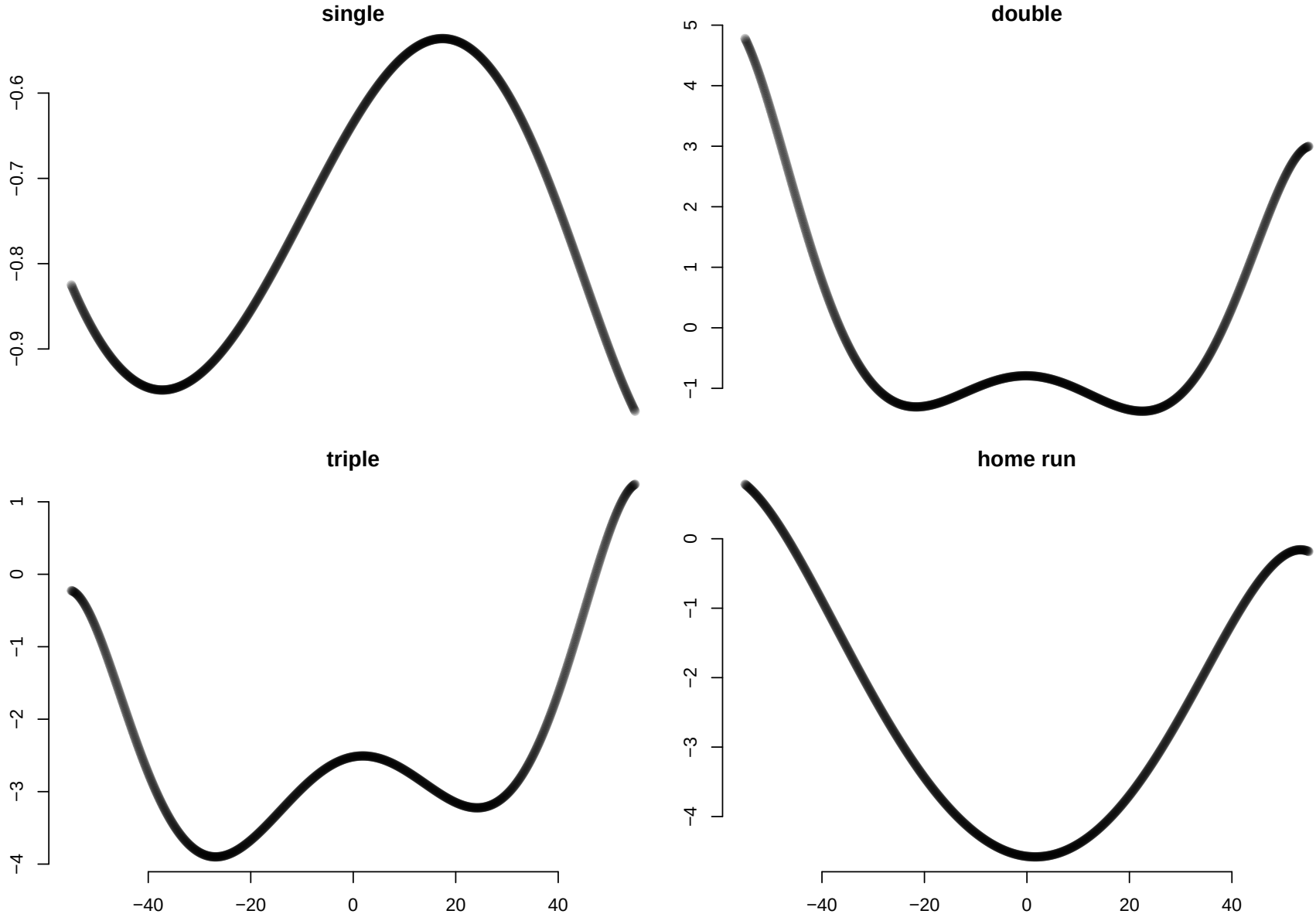
```
pred2 <- predict(mod4, newdata = new_data, tyoe = "response")
```


single
−0.4
−0.5
−0.6
−0.7
−0.8
−0.9
double
4
3
2
1
0
−1
−2
triple
0
−1
−2
−3
−4
−5
−40
−20
0
20
40
home run
−13
−14
−15
−16
−17
−18
−40
−20
0
20
40

Let's consider these predicted values at a new combination of launch angle and exit velocity.

```
## obtain predictions
new_data = data.frame(spray_angle = seq(-55,55, by = 0.1),
           launch_speed = 100,
           launch_angle = 20)
pred <- predict(mod4, newdata = new_data)
head(pred, 3)
```

```
##   log(mu[,1]/mu[,5]) log(mu[,2]/mu[,5]) log(mu[,3]/mu[,5]) log(mu[,4]/mu[,5])
## 1         -0.8249607           4.777891         -0.2259619          0.7844824
## 2         -0.8263690           4.760494         -0.2284624          0.7784865
## 3         -0.8277692           4.742754         -0.2314093          0.7723810
```


single
−0.6
−0.7
−0.8
−0.9
double
5
4
3
2
1
0
−1
triple
1
0
−1
−2
−3
−4
−40
−20
0
20
40
home run
0
−1
−2
−3
−4
−40
−20
0
20
40

Going deeper: SEAM methodology

We have developed a tool for estimating batted-ball distributions for individual batter-pitcher matchups. SEAM is shorthand for Synthetic Estimated Average Matchup. Here is the app:

https://seam.stat.illinois.edu/

Authors:

- Julia Wapner (currently at Baltimore Orioles)
- Charles Young (currently at Houston Astros)
- David Dalpiaz
- Daniel J Eck

What is SEAM?

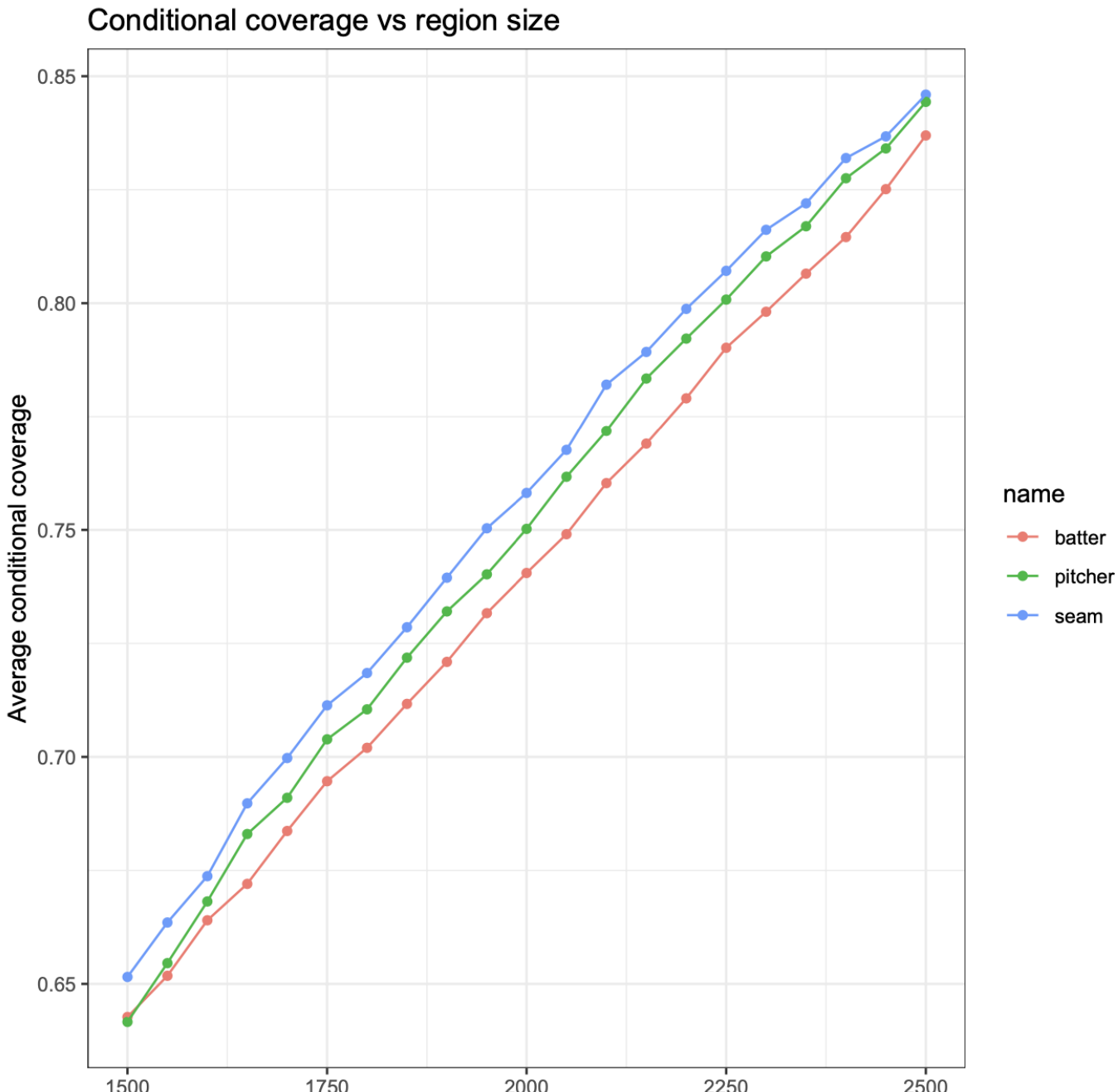
SEAM posits a nonparametric regression model between 2-dimensional batted-ball coordinates and the same STATCAST variables that we considered in this lecture.

Matchup data is sparse. So we borrow data from "similar" matchups to help estimate the batted-ball distribution under study.

Some methodology choices are subjective. However, SEAM performs well.

For more details, see here:
https://github.com/ecklab/seam-manuscript/blob/main/seam.pdf

Why SEAM?


Conditional coverage vs region size
Average conditional coverage
0.85
0.80
0.75
0.70
0.65
1500
1750
2000
2250
2500
name
batter
pitcher
seam