

# Is the MLB Entering an Era of Speed Over Power?

Ivy Maynard – STAT 228 Final Project



# My Research

## Inspiration:

- I've observed an increased focus in sports media on fast players and stolen base records.
- Ronald Acuna Jr., Bobby Whitt Jr., and Jarren Duran

## Question:

- Do fast baserunners get into the heads of opposing teams, causing an increased chance of winning?
- Ultimately, should teams change their priorities from power hitters to fast baserunners?

## Analysis:

- I decided to look into whether or not a stolen base or a triple in the first inning increases a teams chance of winning.

# My Data

## Retrosheet:

- A nonprofit baseball statistics organization that created a collection of all public play-by-play and game data for every MLB+ game dating back to the 1800s
- R package "Retrosheet" allows you to import and scrape data directly from the website to R

Visitors		Home	Date	Score	Temperature	Wind	Precipitation	None	Drizzle
Tigers		Orioles	June 22, 1983	0-2	87°	0-10	None	None	None
Out In	Home Players	Defensive Pos In Out	Field	Dry	Wet	Soaked	Sky	Sunny	Cloudy
1	Bumgarner	8	10	19	28	37	46		
2	Quince	9	11	20	29	38	47		
3	Reese	10	12	21	30	39	48		
4	Lombardi	11	13	22	31	40	49		
5	Reese	12	14	23	32	41	50		
6	Murray	13	15	24	33	42	51		
7	Reese	14	16	25	34	43	52		
8	Reese	15	17	26	35	44	53		
9	Reese	16	18	27	36	45	54		
10	Reese	17	19	28	37	46	55		
11	Reese	18	20	29	38	47	56		
12	Reese	19	21	30	39	48	57		
13	Reese	20	22	31	40	49	58		
14	Reese	21	23	32	41	50	59		
15	Reese	22	24	33	42	51	60		
16	Reese	23	25	34	43	52	61		
17	Reese	24	26	35	44	53	62		
18	Reese	25	27	36	45	54	63		
19	Reese	26	28	37	46	55	64		
20	Reese	27	29	38	47	56	65		
21	Reese	28	30	39	48	57	66		
22	Reese	29	31	40	49	58	67		
23	Reese	30	32	41	50	59	68		
24	Reese	31	33	42	51	60	69		
25	Reese	32	34	43	52	61	70		
26	Reese	33	35	44	53	62	71		
27	Reese	34	36	45	54	63	72		
28	Reese	35	37	46	55	64	73		
29	Reese	36	38	47	56	65	74		
30	Reese	37	39	48	57	66	75		
31	Reese	38	40	49	58	67	76		
32	Reese	39	41	50	59	68	77		
33	Reese	40	42	51	60	69	78		
34	Reese	41	43	52	61	70	79		
35	Reese	42	44	53	62	71	80		
36	Reese	43	45	54	63	72	81		
37	Reese	44	46	55	64	73	82		
38	Reese	45	47	56	65	74	83		
39	Reese	46	48	57	66	75	84		
40	Reese	47	49	58	67	76	85		
41	Reese	48	50	59	68	77	86		
42	Reese	49	51	60	69	78	87		
43	Reese	50	52	61	70	79	88		
44	Reese	51	53	62	71	80	89		
45	Reese	52	54	63	72	81	90		
46	Reese	53	55	64	73	82	91		
47	Reese	54	56	65	74	83	92		
48	Reese	55	57	66	75	84	93		
49	Reese	56	58	67	76	85	94		
50	Reese	57	59	68	77	86	95		
51	Reese	58	60	69	78	87	96		
52	Reese	59	61	70	79	88	97		
53	Reese	60	62	71	80	89	98		
54	Reese	61	63	72	81	90	99		
55	Reese	62	64	73	82	91	100		
56	Reese	63	65	74	83	92	101		
57	Reese	64	66	75	84	93	102		
58	Reese	65	67	76	85	94	103		
59	Reese	66	68	77	86	95	104		
60	Reese	67	69	78	87	96	105		
61	Reese	68	70	79	88	97	106		
62	Reese	69	71	80	89	98	107		
63	Reese	70	72	81	90	99	108		
64	Reese	71	73	82	91	100	109		
65	Reese	72	74	83	92	101	110		
66	Reese	73	75	84	93	102	111		
67	Reese	74	76	85	94	103	112		
68	Reese	75	77	86	95	104	113		
69	Reese	76	78	87	96	105	114		
70	Reese	77	79	88	97	106	115		
71	Reese	78	80	89	98	107	116		
72	Reese	79	81	90	99	108	117		
73	Reese	80	82	91	100	109	118		
74	Reese	81	83	92	101	110	119		
75	Reese	82	84	93	102	111	120		
76	Reese	83	85	94	103	112	121		
77	Reese	84	86	95	104	113	122		
78	Reese	85	87	96	105	114	123		
79	Reese	86	88	97	106	115	124		
80	Reese	87	89	98	107	116	125		
81	Reese	88	90	99	108	117	126		
82	Reese	89	91	100	109	118	127		
83	Reese	90	92	101	110	119	128		
84	Reese	91	93	102	111	120	129		
85	Reese	92	94	103	112	121	130		
86	Reese	93	95	104	113	122	131		
87	Reese	94	96	105	114	123	132		
88	Reese	95	97	106	115	124	133		
89	Reese	96	98	107	116	125	134		
90	Reese	97	99	108	117	126	135		
91	Reese	98	100	109	118	127	136		
92	Reese	99	101	110	119	128	137		
93	Reese	100	102	111	120	129	138		
94	Reese	101	103	112	121	130	139		
95	Reese	102	104	113	122	131	140		
96	Reese	103	105	114	123	132	141		
97	Reese	104	106	115	124	133	142		
98	Reese	105	107	116	125	134	143		
99	Reese	106	108	117	126	135	144		
100	Reese	107	109	118	127	136	145		

# Data Wrangling

## Steps:

- Used function `get_my_retrosheet` to scrape play-by-play data for every MLB game in 2024
- Filtered my data to only include the first inning of each game
- Created columns to identify stolen bases and triples (1 for true and 0 for false)
- Grouped and summarized to find total T and SB in the inning
- Used `get_retrosheet` to find game data for each game of 2024
- Created column to determine win status using `ifelse` statements
- Used `pivot_longer` to create one row for each team involved in the game
- Joined data frames using game ID and selected relevant columns

GAMEID	team	SB	T	Win
ANA202404050	0	0	0	W
ANA202404050	1	0	0	L
ANA202404060	0	0	0	L
ANA202404060	1	0	0	W
ANA202404070	0	0	0	W
ANA202404070	1	0	0	L
ANA202404080	0	0	0	L
ANA202404080	1	1	1	W
ANA202404090	0	0	0	W
ANA202404090	1	0	0	L
ANA202404100	0	0	0	W
ANA202404100	1	0	0	L
ANA202404220	0	1	0	W
ANA202404220	1	0	0	L
ANA202404230	0	0	0	L
ANA202404230	1	0	0	W

1 to 16 of 4,858 entries, 5 total columns

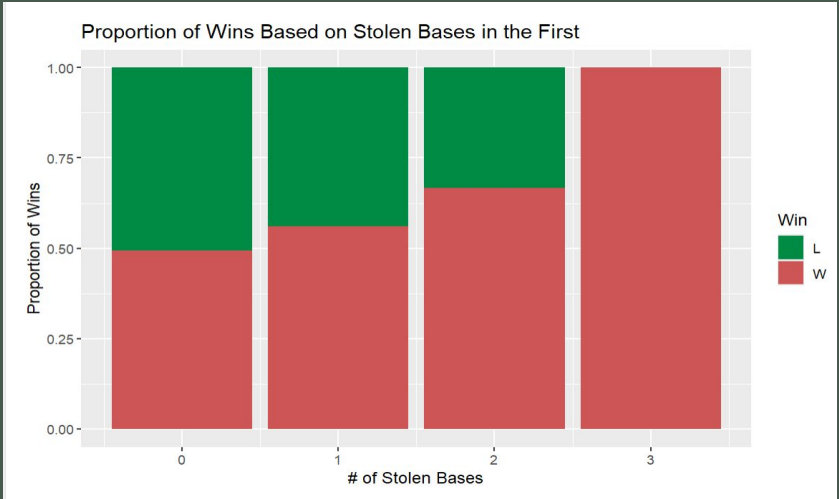
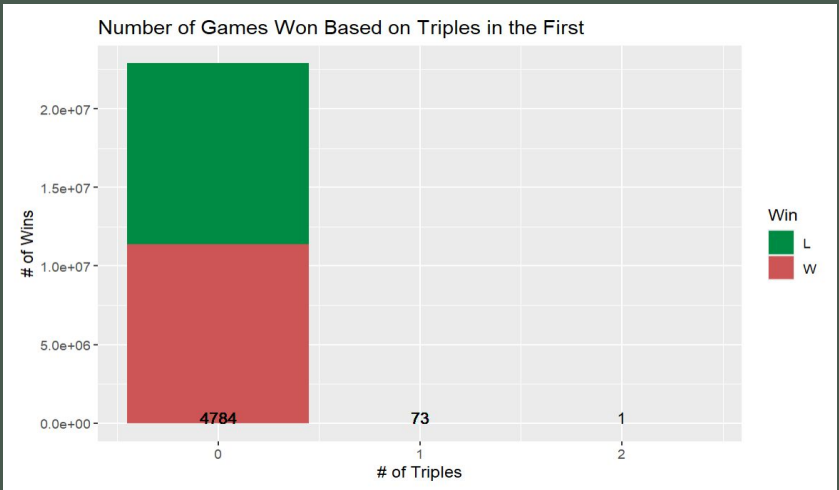
# Visualization

A bar plot showing games in 2024 that were wins vs losses based on # of SB in the 1st

- Stacked vs filled bar charts show the effects of disparities in count
- Many more games without a SB or T in the first

A bar plot showing games in 2024 that were wins vs losses based on # of T in the 1st

- Stacked vs filled
- Only 1 game had 2 triples in the 1st which leads to unexpected filled bar chart



# Modeling

## Steps:

- Split data into train (80%) and test(20%)
- Fit a logistic regression model
- Evaluate my model
  - Find the predicted probability of wins for test and training data
  - Create a confusion matrix
  - Determine F1 score
- Cross validate with data from 2023
  - Repeat steps

Call:

```
glm(formula = Win ~ SB + T + SB:T, family = "binomial", data = data_train)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-0.02586	0.03359	-0.770	0.4414
SB	0.21682	0.10464	2.072	0.0383 *
T	0.23858	0.29562	0.807	0.4196
SB:T	1.76769	1.09916	1.608	0.1078

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Logistic Model:

$$\text{Win} = 0.217(\text{SB}) + 0.239(\text{T}) + 1.768(\text{SB:T}) - 0.026$$

## F1 Scores:

2024 Training Data: 0.177

2023 Training Data: 0.171

2024 Test Data: 0.200

2023 Test Data: 0.161

## Confusion Matrix and Statistics

	Reference	
Prediction	L	W
L	1783	1739
W	161	205

# Conclusion

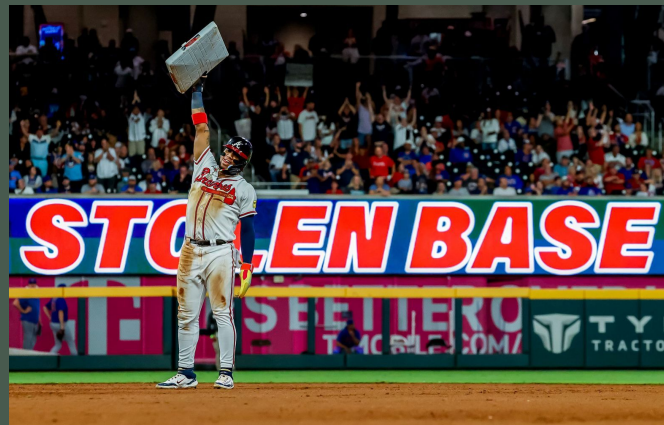
## What does the model tell us?

- F1 score is on a scale of 0-1 with 1 being the most accurate and 0 being the least
- The low F1 values shows us that our model is not much better at predicting wins than a random model
- The P value of SB is below the level of significance of .05 while T is not
- Suggest that there is a correlation between wins and SB in the first inning but no correlation between wins and T in the first inning

## Where can we go from here?

Lots of follow-up analysis can be done to see if we can find a better way to predict runs:

- Look at SB and T in all 9 innings
- Remove triples from the model
- Look at other speed metrics like sprint speed, 90 ft/s, doubles on shallow hits, etc.



## What does this say about the MLB?

This model does not provide concrete evidence to suggest that hitting more triples in the first inning leads to an increased win probability, but it does suggest that stolen bases can impact the outcome. Therefore, teams may benefit from acquiring players who steal more bases as well as training the top of their batting order to steal early in the game. More analysis would need to be done.

# Citations and References

## Retrosheet:

- Data: <https://www.retrosheet.org/>
- Function:  
<https://cran.r-project.org/web/packages/retrosheet/index.html>

## Get\_my\_retrosheet function:

- Author: Jim Albert
- GitHub:  
<https://gist.github.com/bayesball/e7d56e5edf31d7a71c6bf40cc1dfe743>
- <https://baseballwithr.wordpress.com/2024/03/27/retrosheet-package-and-comparing-count-rates/>

## Packages:

- tidyverse
- retrosheet
- openintro
- caret
- pROC
- rpart
- rpart.plot