**Exploring MLB’s 2020 Season with Altered Baseball Python Analysis**

A picture containing indoor

Description automatically generated

**Ricky Camilo**

**February 2022**

**Table of Contents**

1. Useful Baseball Statistics Terminology
2. What’s So Interesting About These Baseballs
3. How Will We Analyze and Munge This Data?
4. Analyzing Hitting Statistics From the 2018-2021 MLB Seasons
5. Analyzing Pitching Statistics From the 2018-2021 MLB Seasons
6. Analyzing Statcast Data From the 2018-2021 MLB Seasons
7. Predictive Models of 2020 MLB Hitting, Pitching and Statcast Statistics (Linear Regression)
8. Results and Summary

**Useful Baseball Statistics Terminology**

Sabermetrics - The empirical analysis of baseball statistics that measure in-game activity.

H - Hits

HR - Home runs

RBI - Runs batted in

WAR - Wins Above Replacement an all-encompassing statistic that assigns a numeric value to a player in comparison to “0” which would be considered a “league average player”

AVG - Batting average (average of how many times a hitter gets a hit per at bat)

SLG% - Slugging percentage (% of hits that are more than just a single)

ERA- Earned Run Average (How many runs a pitcher usually gives up per 9 innings of play)

K - Strikeouts

H/9 - Hits per 9 innings (How many hits a pitcher allows per 9 innings of play)

HR/9 - Home runs per 9 innings (How many homeruns a pitcher allows per 9 innings)

EV - Exit Velocity (how fast a baseball is traveling after it’s been hit)

**What’s So Interesting About These Baseballs?**

Two weeks before the start of the MLB’s 2021 Spring Training campaign. MLB Executive Reporter Mark Feinsand broke that an independent lab reported that official game used baseballs used in the 2020 season had less drag and flew one to two feet shorter on balls hit over 375 feet. The conclusion being inconsistencies in the height of the seams. Rawlings Official Game Used Baseballs are all hand sewn and have a deviation range of .530 to .570. Rawlings has since admitted to loosening the seams on the baseballs which is supposed to slightly reduce drag which in turn should increase overall hitting production. This is all after a groundbreaking discovery by Dr. Meredith Wills who found that the baseballs used in 2019 and the 2020 shortened season were inconsistent with the baseballs used in seasons past. What makes these studies and claims so interesting is that the 2020 season was cut short because of COVID as all these rumors of the ball being different started circling. There is a large gap in data that does not exist because the 2020 was hardly 2 months’ worth of games. We will analyze data from the 2018-2021 seasons and create a regression algorithm that takes in previous season data and predicts stats for a full 162 game 2020 season which we will compare to see how it stacks up to the “juiced” season of 2019 or “normal” season of 2018.

**How Will We Analyze and Munge This Data?**

**pybaseball** is a Python package for baseball data analysis. This package scrapes Baseball Reference, Baseball Savant, and FanGraphs so we don't have to. The package retrieves statcast data, pitching stats, batting stats, division standings/team records, awards data, and more. Data is available at the individual pitch level, as well as aggregated at the season level and over custom time periods. Baseball Savant is MLB’s clearinghouse for statcast data (statcast is a high-speed, high-accuracy, automated tool developed to analyze player movements and athletic abilities in Major League Baseball (MLB))

In this project I use a package called **pybaseball** to analyze original quantitative baseball statistics and sabermetric data to see if we notice a difference in offensive production from season to season. We will also analyze **statcast** data exported in .csv format from the Baseball Savant website to compare baseball exit velocity and launch angles per season. I also use **pandas** for our data frames, **seaborn** and **matplotlib** for visualization purposes, and **sklearn** for our linear regression model.

All the code used in this analysis can be found on my Github under the [“Exploring MLB’s 2020 Season with Altered Baseball”](https://github.com/rickycamilo9/Exploring-MLB-s-2020-Season-with-Altered-Baseball-) repository. I highly encourage running the code on google colab or your choice of IDE for the sake of clarity and space. I will condense or omit our python code outputs to data that is considered important to our analysis.

**Analyzing Hitting Statistics From the 2018-2021 MLB Seasons**

Lets begin by pip installing pybaseball and matplotlib (for visualization), importing pandas for our dataframes and setting our display options, importing seaborn (for visualization), importing sklearn for our linear regression model and importing pybaseball.

**%pip install pybaseball==2.2.1**

**%matplotlib inline**

**import pandas as pd;**

**pd.set\_option('display.max\_rows', None)**

**pd.set\_option('display.max\_columns', None)**

**pd.set\_option('display.width', None)**

**pd.set\_option('display.max\_colwidth', -1)**

**pd.set\_option('display.float\_format', lambda x: '%.5f' % x)**

**import seaborn as sns**

**import sklearn as sk**

**import pybaseball as pyb**

**from matplotlib import pyplot as plt**

**from pybaseball import cache**

**cache.enable()**

**import warnings; warnings.filterwarnings('ignore')**

Next, we will import our cumulative battings statistics from pybaseball and filter each DF up until the WAR column

**#importing cumulative team batting stats from pybaseball**

**from pybaseball import team\_batting as team\_batting**

**batting\_stats\_2018 = team\_batting(2018)**

**batting\_stats\_2019 = team\_batting(2019)**

**batting\_stats\_2020 = team\_batting(2020)**

**batting\_stats\_2021 = team\_batting(2021)**

**#modifying data frames for each season to only show columns up to WAR**

**batting\_stats\_2018 = batting\_stats\_2018.loc[:, :'WAR']**

**batting\_stats\_2019 = batting\_stats\_2019.loc[:, :'WAR']**

**batting\_stats\_2020 = batting\_stats\_2020.loc[:, :'WAR']**

**batting\_stats\_2021 = batting\_stats\_2021.loc[:, :'WAR']**

Now we will call our batting stats data frames we have created to extract the meaningful and most important hitting stats.

**batting\_stats\_2018.mean()**

teamIDfg 15.50000

Season 2018.00000

Age 27.96667

G 2386.33333

AB 5514.40000

PA 6171.30000

H 1367.26667

1B 877.40000

2B 275.46667

3B 28.23333

HR 186.16667

R 721.00000

RBI 686.86667

BB 522.86667

IBB 30.96667

SO 1373.56667

HBP 64.06667

SF 41.16667

SH 27.43333

GDP 115.23333

SB 82.46667

CS 31.93333

AVG 0.24783

GB 1790.16667

FB 1465.33333

LD 889.23333

IFFB 151.43333

Pitches 24039.70000

Balls 8717.60000

Strikes 15322.10000

IFH 121.40000

BU 64.43333

BUH 14.70000

BB% 0.08463

K% 0.22277

BB/K 0.38300

OBP 0.31797

SLG 0.40923

OPS 0.72723

ISO 0.16130

BABIP 0.29547

GB/FB 1.23067

LD% 0.21460

GB% 0.43207

FB% 0.35340

IFFB% 0.10327

HR/FB 0.12683

IFH% 0.06770

BUH% 0.23057

wOBA 0.31460

wRAA -1.76667

wRC 719.33333

Bat -25.98333

Fld 0.58000

Rep 184.64667

Pos 10.87333

RAR 184.64000

WAR 19.01333

dtype: float64

**batting\_stats\_2018.sum()**

teamIDfg 465

Season 60540

Team BOSNYYLADCLEOAKHOUWSNCOLTBRMILCHCATLTORCINSTLTEXLAAMINSEAPITPHIARINYMCHWKCRBALDETSDPSFGMIA

Age 839

G 71590

AB 165432

PA 185139

H 41018

1B 26322

2B 8264

3B 847

HR 5585

R 21630

RBI 20606

BB 15686

IBB 929

SO 41207

HBP 1922

SF 1235

SH 823

GDP 3457

SB 2474

CS 958

AVG 7.43500

GB 53705

FB 43960

LD 26677

IFFB 4543

Pitches 721191

Balls 261528

Strikes 459663

IFH 3642

BU 1933

BUH 441

BB% 2.53900

K% 6.68300

BB/K 11.49000

OBP 9.53900

SLG 12.27700

OPS 21.81700

ISO 4.83900

BABIP 8.86400

GB/FB 36.92000

LD% 6.43800

GB% 12.96200

FB% 10.60200

IFFB% 3.09800

HR/FB 3.80500

IFH% 2.03100

BUH% 6.91700

wOBA 9.43800

wRAA -53.00000

wRC 21580

Bat -779.50000

Fld 17.40000

Rep 5539.40000

Pos 326.20000

RAR 5539.20000

WAR 570.40000

dtype: object

The 2018 season was the last season in which the balls were reported “normal”. Here are the main hitting statistics for that season:

H= 41,018

HR= 5,585

RBI= 20,606

WAR= 570

AVG= .248

SLG% = .409

**batting\_stats\_2019.mean()**

Season 2019.00000

Age 27.86667

G 2389.46667

AB 5555.03333

PA 6217.20000

H 1401.30000

1B 864.90000

2B 284.36667

3B 26.16667

HR 225.86667

R 782.23333

RBI 749.03333

BB 529.83333

IBB 25.10000

SO 1427.43333

HBP 66.13333

SF 38.33333

SH 25.86667

GDP 115.43333

SB 76.00000

CS 27.73333

AVG 0.25217

GB 1772.26667

FB 1476.10000

LD 886.23333

IFFB 145.30000

Pitches 24415.90000

Balls 8857.03333

Strikes 15558.86667

IFH 115.13333

BU 56.60000

BUH 12.86667

BB% 0.08520

K% 0.22983

BB/K 0.37467

OBP 0.32243

SLG 0.43467

OPS 0.75723

ISO 0.18257

BABIP 0.29827

GB/FB 1.20900

LD% 0.21430

GB% 0.42873

FB% 0.35693

IFFB% 0.09823

HR/FB 0.15240

IFH% 0.06493

BUH% 0.23393

wOBA 0.31973

wRAA -1.97000

wRC 780.53333

Bat -27.58667

Fld 0.37333

Rep 195.53667

Pos 11.34667

RAR 195.55333

WAR 18.99000

dtype: float64

**batting\_stats\_2019.sum()**

teamIDfg 465

Season 60570

Team HOUMINNYYLADBOSWSNATLCHCCOLOAKNYMMILTBRARICLETEXLAAPHISTLSEAPITCINTORCHWBALSDPKCRSFGDETMIA

Age 836

G 71684

AB 166651

PA 186516

H 42039

1B 25947

2B 8531

3B 785

HR 6776

R 23467

RBI 22471

BB 15895

IBB 753

SO 42823

HBP 1984

SF 1150

SH 776

GDP 3463

SB 2280

CS 832

AVG 7.56500

GB 53168

FB 44283

LD 26587

IFFB 4359

Pitches 732477

Balls 265711

Strikes 466766

IFH 3454

BU 1698

BUH 386

BB% 2.55600

K% 6.89500

BB/K 11.24000

OBP 9.67300

SLG 13.04000

OPS 22.71700

ISO 5.47700

BABIP 8.94800

GB/FB 36.27000

LD% 6.42900

GB% 12.86200

FB% 10.70800

IFFB% 2.94700

HR/FB 4.57200

IFH% 1.94800

BUH% 7.01800

wOBA 9.59200

wRAA -59.10000

wRC 23416

Bat -827.60000

Fld 11.20000

Rep 5866.10000

Pos 340.40000

RAR 5866.60000

WAR 569.70000

dtype: object

The 2019 season was the first season of the juiced baseballs. These stats show an obvious increase in hitting production league wide.

H= 42,039

HR= 6,776

RBI = 22,471

WAR= 570

AVG= .252

SLG%= .435

**batting\_stats\_2020.mean()**

teamIDfg 15.50000

Season 2020.00000

Age 27.83333

G 890.70000

AB 1967.66667

PA 2216.86667

H 481.30000

1B 302.36667

2B 94.10000

3B 8.03333

HR 76.80000

R 278.13333

RBI 265.93333

BB 203.06667

IBB 6.73333

SO 519.53333

HBP 27.36667

SF 13.40000

SH 4.20000

GDP 41.23333

SB 29.50000

CS 9.73333

AVG 0.24423

GB 620.63333

FB 517.76667

LD 313.93333

IFFB 49.30000

Pitches 8786.36667

Balls 3254.93333

Strikes 5531.43333

IFH 43.30000

BU 13.33333

BUH 4.40000

BB% 0.09153

K% 0.23463

BB/K 0.39367

OBP 0.32163

SLG 0.41690

OPS 0.73850

ISO 0.17277

BABIP 0.29163

GB/FB 1.20700

LD% 0.21617

GB% 0.42750

FB% 0.35630

IFFB% 0.09533

HR/FB 0.14817

IFH% 0.06950

BUH% 0.34223

wOBA 0.31943

wRAA 0.01667

wRC 278.23333

Bat 0.35333

Fld 0.22000

Rep 72.19333

Pos -7.26000

RAR 72.19667

WAR 7.01667

dtype: float64

**batting\_stats\_2020.sum()**

teamIDfg 465

Season 60600

Team ATLLADNYMSDPNYYSFGPHICHWBOSWSNLAATORTBRBALMINOAKCINHOUCHCKCRMIACOLMILSTLARICLEDETSEATEXPIT

Age 835

G 26721

AB 59030

PA 66506

H 14439

1B 9071

2B 2823

3B 241

HR 2304

R 8344

RBI 7978

BB 6092

IBB 202

SO 15586

HBP 821

SF 402

SH 126

GDP 1237

SB 885

CS 292

AVG 7.32700

GB 18619

FB 15533

LD 9418

IFFB 1479

Pitches 263591

Balls 97648

Strikes 165943

IFH 1299

BU 400

BUH 132

BB% 2.74600

K% 7.03900

BB/K 11.81000

OBP 9.64900

SLG 12.50700

OPS 22.15500

ISO 5.18300

BABIP 8.74900

GB/FB 36.21000

LD% 6.48500

GB% 12.82500

FB% 10.68900

IFFB% 2.86000

HR/FB 4.44500

IFH% 2.08500

BUH% 10.26700

wOBA 9.58300

wRAA 0.50000

wRC 8347

Bat 10.60000

Fld 6.60000

Rep 2165.80000

Pos -217.80000

RAR 2165.90000

WAR 210.50000

dtype: object

These stats are for the COVID shortened 2020 season (60 games) which is a small sample size to test if we want to see if the baseballs were juiced or not.

H=14,439

HR= 2,304

RBI= 7,978

WAR= 210

AVG= .244

SLG% = .417

**batting\_stats\_2021.mean()**

teamIDfg 15.50000

Season 2021.00000

Age 28.20000

G 2387.36667

AB 5398.03333

PA 6060.56667

H 1316.13333

1B 833.53333

2B 262.10000

3B 22.36667

HR 198.13333

R 733.66667

RBI 699.76667

BB 526.46667

IBB 23.43333

SO 1404.83333

HBP 70.40000

SF 38.10000

SH 25.53333

GDP 110.93333

SB 73.76667

CS 23.70000

AVG 0.24377

GB 1717.00000

FB 1459.86667

LD 827.06667

IFFB 145.43333

Pitches 23661.83333

Balls 8545.63333

Strikes 15116.20000

IFH 117.83333

BU 52.70000

BUH 10.43333

BB% 0.08677

K% 0.23183

BB/K 0.37600

OBP 0.31690

SLG 0.41053

OPS 0.72753

ISO 0.16680

BABIP 0.29160

GB/FB 1.18433

LD% 0.20653

GB% 0.42907

FB% 0.36447

IFFB% 0.09953

HR/FB 0.13547

IFH% 0.06870

BUH% 0.20773

wOBA 0.31420

wRAA -1.96667

wRC 731.96667

Bat -23.72333

Fld 0.84000

Rep 189.40000

Pos 9.55000

RAR 189.40000

WAR 18.99333

dtype: float64

**batting\_stats\_2021.sum()**

teamIDfg 465

Season 60630

Team TORHOUBOSSFGCHWCINLADWSNATLTBRMINNYYCOLPHIOAKSDPSTLCHCMILLAANYMCLEDETBALKCRARISEAPITMIATEX

Age 846

G 71621

AB 161941

PA 181817

H 39484

1B 25006

2B 7863

3B 671

HR 5944

R 22010

RBI 20993

BB 15794

IBB 703

SO 42145

HBP 2112

SF 1143

SH 766

GDP 3328

SB 2213

CS 711

AVG 7.31300

GB 51510

FB 43796

LD 24812

IFFB 4363

Pitches 709855

Balls 256369

Strikes 453486

IFH 3535

BU 1581

BUH 313

BB% 2.60300

K% 6.95500

BB/K 11.28000

OBP 9.50700

SLG 12.31600

OPS 21.82600

ISO 5.00400

BABIP 8.74800

GB/FB 35.53000

LD% 6.19600

GB% 12.87200

FB% 10.93400

IFFB% 2.98600

HR/FB 4.06400

IFH% 2.06100

BUH% 6.23200

wOBA 9.42600

wRAA -59.00000

wRC 21959

Bat -711.70000

Fld 25.20000

Rep 5682.00000

Pos 286.50000

RAR 5682.00000

WAR 569.80000

dtype: object

These stats are for the 2021 season (last full regular season) and the balls were apparently altered to perform like the baseballs of 2018.

H= 39,484

HR=5944

RBI= 20,993

WAR= 570

AVG=.244

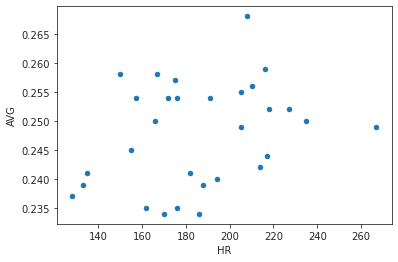
SLG%= .411

We will later create a linear regression algorithm that takes data from previous seasons and predicts what a full 2020 season would have been like. We will compare the stats for all 4 seasons (2018, 2019, 2020, 2021) and see if a hypothetical 2020 season would have performed closer to a regular season or the 2019 juiced ball season.

Lastly for this hitting section we will visualize a scatter plot for each data frame that shows us any correlation between homeruns and batting average.

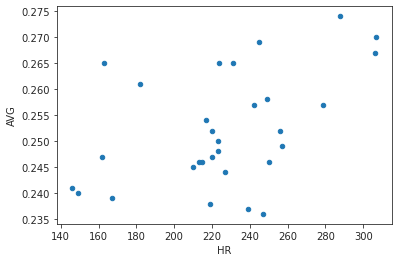
**import seaborn as sns; sns.set\_style('ticks');**

**batting\_stats\_2018.plot(x='HR', y='AVG', kind='scatter');**



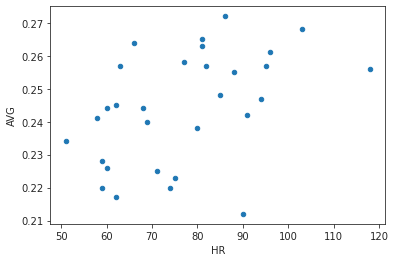
**import seaborn as sns; sns.set\_style('ticks');**

**batting\_stats\_2019.plot(x='HR', y='AVG', kind='scatter');**



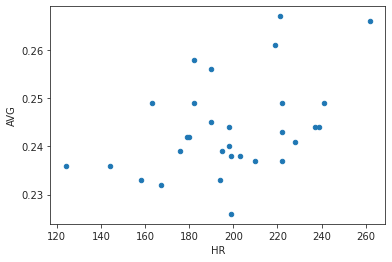
**import seaborn as sns; sns.set\_style('ticks');**

**batting\_stats\_2020.plot(x='HR', y='AVG', kind='scatter');**



**import seaborn as sns; sns.set\_style('ticks');**

**batting\_stats\_2021.plot(x='HR', y='AVG', kind='scatter');**



Looking at the relationship between AVG and HR shows us what we already knew to be a true. 2019 had a bump in hitting production where there were 4 teams with over 280 homeruns but only 1 in 2018 and 2021.

**Analyzing Pitching Statistics From the 2018-2021 MLB Seasons**

This sections code will be very similar to how we extracted our hitting data. First, we will import our cumulative team pitching stats from pybaseball

**#importing cumulative team pitching stats from pybaseball**

**from pybaseball import team\_pitching**

**pitching\_stats\_2018 = pyb.team\_pitching(2018)**

**pitching\_stats\_2019 = pyb.team\_pitching(2019)**

**pitching\_stats\_2020 = pyb.team\_pitching(2020)**

**pitching\_stats\_2021 = pyb.team\_pitching(2021)**

**pitching\_stats\_2018 = pitching\_stats\_2018.loc[:, :'WAR']**

**pitching\_stats\_2019 = pitching\_stats\_2019.loc[:, :'WAR']**

**pitching\_stats\_2020 = pitching\_stats\_2020.loc[:, :'WAR']**

**pitching\_stats\_2021 = pitching\_stats\_2021.loc[:, :'WAR']**

**pitching\_stats\_2018.mean()**

teamIDfg 15.50000

Season 2018.00000

Age 27.90000

W 81.03333

L 81.03333

ERA 4.15300

G 706.53333

GS 162.06667

CG 1.40000

ShO 0.63333

SV 41.46667

BS 21.06667

IP 1449.44667

TBF 6171.30000

H 1367.26667

R 721.00000

ER 668.36667

HR 186.16667

BB 522.86667

IBB 30.96667

HBP 64.06667

WP 61.56667

BK 5.03333

SO 1373.56667

GB 1790.16667

FB 1465.33333

LD 889.23333

IFFB 151.43333

Balls 8717.60000

Strikes 15322.10000

Pitches 24039.70000

RS 721.00000

IFH 121.40000

BU 64.43333

BUH 17.96667

K/9 8.52533

BB/9 3.24667

K/BB 2.67833

H/9 8.49200

HR/9 1.15667

AVG 0.24470

WHIP 1.30367

BABIP 0.29320

LOB% 0.72920

FIP 4.15233

GB/FB 1.22833

LD% 0.21440

GB% 0.43207

FB% 0.35340

IFFB% 0.10327

HR/FB 0.12697

IFH% 0.06780

BUH% 0.29290

Starting 102.76000

Start-IP 866.34000

Relieving 31.72333

Relief-IP 575.31333

RAR 134.47000

WAR 14.33667

dtype: float64

**pitching\_stats\_2018.sum()**

teamIDfg 465

Season 60540

Team HOULADCHCARIMILTBRATLBOSCLENYYOAKSTLSFGPITWSNNYMSEAPHILAACOLSDPMINDETCINMIACHWTORTEXKCRBAL

Age 837

W 2431

L 2431

ERA 124.59000

G 21196

GS 4862

CG 42

ShO 19

SV 1244

BS 632

IP 43483.40000

TBF 185139

H 41018

R 21630

ER 20051

HR 5585

BB 15686

IBB 929

HBP 1922

WP 1847

BK 151

SO 41207

GB 53705

FB 43960

LD 26677

IFFB 4543

Balls 261528

Strikes 459663

Pitches 721191

RS 21630

IFH 3642

BU 1933

BUH 539

K/9 255.76000

BB/9 97.40000

K/BB 80.35000

H/9 254.76000

HR/9 34.70000

AVG 7.34100

WHIP 39.11000

BABIP 8.79600

LOB% 21.87600

FIP 124.57000

GB/FB 36.85000

LD% 6.43200

GB% 12.96200

FB% 10.60200

IFFB% 3.09800

HR/FB 3.80900

IFH% 2.03400

BUH% 8.78700

Starting 3082.80000

Start-IP 25990.20000

Relieving 951.70000

Relief-IP 17259.40000

RAR 4034.10000

WAR 430.10000

dtype: object

This was the last time season the balls were normal

K= 41,207

ERA= 4.15

H/9 = 8.49

HR/9 = 1.16

EV= 88.4

**pitching\_stats\_2019.mean()**

teamIDfg 15.50000

Season 2019.00000

Age 27.90000

W 80.96667

L 80.96667

ERA 4.50800

G 714.30000

GS 161.93333

CG 1.50000

ShO 0.86667

SV 39.33333

BS 22.90000

IP 1447.27333

TBF 6217.20000

H 1401.30000

R 782.23333

ER 724.60000

HR 225.86667

BB 529.83333

IBB 25.10000

HBP 66.13333

WP 59.60000

BK 5.10000

SO 1427.43333

GB 1772.26667

FB 1476.10000

LD 886.23333

IFFB 145.30000

Balls 8857.03333

Strikes 15558.86667

Pitches 24415.90000

RS 782.23333

IFH 115.13333

BU 56.60000

BUH 15.80000

K/9 8.87333

BB/9 3.29600

K/BB 2.73800

H/9 8.71600

HR/9 1.40600

AVG 0.24917

WHIP 1.33400

BABIP 0.29600

LOB% 0.72397

FIP 4.50733

GB/FB 1.20733

LD% 0.21423

GB% 0.42870

FB% 0.35700

IFFB% 0.09850

HR/FB 0.15293

IFH% 0.06497

BUH% 0.29550

Starting 112.16000

Start-IP 835.70333

Relieving 29.25667

Relief-IP 603.33333

RAR 141.41333

WAR 14.33000

dtype: float64

**pitching\_stats\_2019.sum()**

teamIDfg 465

Season 60570

Team LADHOUTBRCLESTLOAKCHCCINMINATLARINYMWSNNYYSFGMILPHISDPBOSMIATORCHWSEATEXLAAPITKCRDETCOLBAL

Age 837

W 2429

L 2429

ERA 135.24000

G 21429

GS 4858

CG 45

ShO 26

SV 1180

BS 687

IP 43418.20000

TBF 186516

H 42039

R 23467

ER 21738

HR 6776

BB 15895

IBB 753

HBP 1984

WP 1788

BK 153

SO 42823

GB 53168

FB 44283

LD 26587

IFFB 4359

Balls 265711

Strikes 466766

Pitches 732477

RS 23467

IFH 3454

BU 1698

BUH 474

K/9 266.20000

BB/9 98.88000

K/BB 82.14000

H/9 261.48000

HR/9 42.18000

AVG 7.47500

WHIP 40.02000

BABIP 8.88000

LOB% 21.71900

FIP 135.22000

GB/FB 36.22000

LD% 6.42700

GB% 12.86100

FB% 10.71000

IFFB% 2.95500

HR/FB 4.58800

IFH% 1.94900

BUH% 8.86500

Starting 3364.80000

Start-IP 25071.10000

Relieving 877.70000

Relief-IP 18100.00000

RAR 4242.40000

WAR 429.90000

dtype: object

These pitching stats for the 2019 season show a slight increase in strikeouts but an overwhelming jump by nearly half a run in ERA, an increase in Hits per 9, Homeruns per 9, and exit velocity

K= 42,823

ERA= 4.51

H/9 = 8.72

HR/9 = 1.41

EV= 88.7

**pitching\_stats\_2020.mean()**

teamIDfg 15.50000

Season 2020.00000

Age 27.80000

W 29.93333

L 29.93333

ERA 4.45667

G 265.30000

GS 59.86667

CG 0.96667

ShO 0.40000

SV 14.06667

BS 8.26667

IP 515.39667

TBF 2216.86667

H 481.30000

R 278.13333

ER 255.13333

HR 76.80000

BB 203.06667

IBB 6.73333

HBP 27.36667

WP 22.50000

BK 2.10000

SO 519.53333

GB 620.63333

FB 517.76667

LD 313.93333

IFFB 49.30000

Balls 3254.93333

Strikes 5531.43333

Pitches 8786.36667

RS 278.13333

IFH 43.30000

BU 13.33333

BUH 5.16667

K/9 9.06833

BB/9 3.54867

K/BB 2.62100

H/9 8.40333

HR/9 1.34100

AVG 0.24197

WHIP 1.32733

BABIP 0.29067

LOB% 0.71947

FIP 4.45667

GB/FB 1.20833

LD% 0.21603

GB% 0.42753

FB% 0.35650

IFFB% 0.09493

HR/FB 0.14877

IFH% 0.06970

BUH% 0.38187

Starting 38.42667

Start-IP 284.27667

Relieving 11.94000

Relief-IP 224.77333

RAR 50.35667

WAR 5.30000

dtype: float64

**pitching\_stats\_2020.sum()**

teamIDfg 465

Season 60600

Team LADCLETBRMINOAKCHWSDPCINSTLCHCMILKCRHOUNYYATLBALTORSFGPITARIMIANYMTEXSEALAAWSNPHIBOSCOLDET

Age 834

W 898

L 898

ERA 133.70000

G 7959

GS 1796

CG 29

ShO 12

SV 422

BS 248

IP 15461.90000

TBF 66506

H 14439

R 8344

ER 7654

HR 2304

BB 6092

IBB 202

HBP 821

WP 675

BK 63

SO 15586

GB 18619

FB 15533

LD 9418

IFFB 1479

Balls 97648

Strikes 165943

Pitches 263591

RS 8344

IFH 1299

BU 400

BUH 155

K/9 272.05000

BB/9 106.46000

K/BB 78.63000

H/9 252.10000

HR/9 40.23000

AVG 7.25900

WHIP 39.82000

BABIP 8.72000

LOB% 21.58400

FIP 133.70000

GB/FB 36.25000

LD% 6.48100

GB% 12.82600

FB% 10.69500

IFFB% 2.84800

HR/FB 4.46300

IFH% 2.09100

BUH% 11.45600

Starting 1152.80000

Start-IP 8528.30000

Relieving 358.20000

Relief-IP 6743.20000

RAR 1510.70000

WAR 159.00000

dtype: object

Since this is the season that was COVID shortened we will take this 60-game sample size with a grain of salt since it was barely 2 months’ worth of games and training camp was canceled so players were not in the usual routine they are used to. ERA was still closer to 2019s juiced ball season which makes total sense since it was a higher offensive production environment and pitchers gave up more runs.

K= 15,586

ERA= 4.46

H/9 = 8.40

HR/9 = 1.34

EV = 88.4

**pitching\_stats\_2021.mean()**

teamIDfg 15.50000

Season 2021.00000

Age 28.33333

W 80.96667

L 80.96667

ERA 4.27067

G 718.03333

GS 161.93333

CG 1.66667

ShO 0.96667

SV 39.70000

BS 24.96667

IP 1420.31333

TBF 6060.56667

H 1316.13333

R 733.66667

ER 673.30000

HR 198.13333

BB 526.46667

IBB 23.43333

HBP 70.40000

WP 62.06667

BK 5.16667

SO 1404.83333

GB 1717.00000

FB 1459.86667

LD 827.06667

IFFB 145.43333

Balls 8545.63333

Strikes 15116.20000

Pitches 23661.83333

RS 733.66667

IFH 117.83333

BU 52.70000

BUH 12.80000

K/9 8.89867

BB/9 3.33733

K/BB 2.70167

H/9 8.34267

HR/9 1.25667

AVG 0.24077

WHIP 1.29800

BABIP 0.28940

LOB% 0.72223

FIP 4.26800

GB/FB 1.18267

LD% 0.20653

GB% 0.42913

FB% 0.36440

IFFB% 0.09943

HR/FB 0.13557

IFH% 0.06860

BUH% 0.25160

Starting 103.67667

Start-IP 810.44000

Relieving 33.27000

Relief-IP 600.89000

RAR 136.94000

WAR 14.32333

dtype: float64

**pitching\_stats\_2021.sum()**

teamIDfg 465

Season 60630

Team LADSFGMILTBRCHWNYYHOUATLNYMTORMIASTLOAKSDPBOSSEADETCLEPHICINKCRLAATEXWSNCOLMINCHCPITARIBAL

Age 850

W 2429

L 2429

ERA 128.12000

G 21541

GS 4858

CG 50

ShO 29

SV 1191

BS 749

IP 42609.40000

TBF 181817

H 39484

R 22010

ER 20199

HR 5944

BB 15794

IBB 703

HBP 2112

WP 1862

BK 155

SO 42145

GB 51510

FB 43796

LD 24812

IFFB 4363

Balls 256369

Strikes 453486

Pitches 709855

RS 22010

IFH 3535

BU 1581

BUH 384

K/9 266.96000

BB/9 100.12000

K/BB 81.05000

H/9 250.28000

HR/9 37.70000

AVG 7.22300

WHIP 38.94000

BABIP 8.68200

LOB% 21.66700

FIP 128.04000

GB/FB 35.48000

LD% 6.19600

GB% 12.87400

FB% 10.93200

IFFB% 2.98300

HR/FB 4.06700

IFH% 2.05800

BUH% 7.54800

Starting 3110.30000

Start-IP 24313.20000

Relieving 998.10000

Relief-IP 18026.70000

RAR 4108.20000

WAR 429.70000

dtype: object

The season when MLB regulated the baseballs back to normal. We can clearly see that these statistics are closer to 2018s than 2019s.

K= 42,145

ERA= 4.27

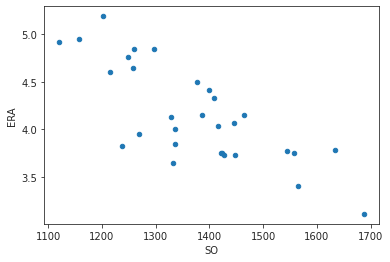
H/9 = 8.34

HR/9 = 1.26

EV= 88.7

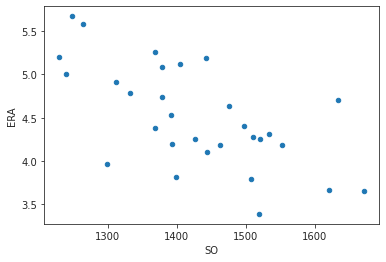
**import seaborn as sns; sns.set\_style('ticks');**

**pitching\_stats\_2018.plot(x='SO', y='ERA', kind='scatter');**



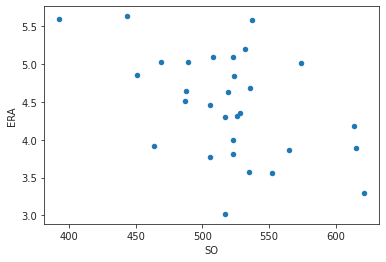
**import seaborn as sns; sns.set\_style('ticks');**

**pitching\_stats\_2019.plot(x='SO', y='ERA', kind='scatter');**



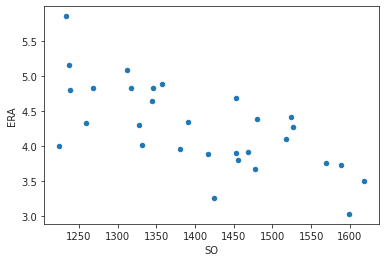
**import seaborn as sns; sns.set\_style('ticks');**

**pitching\_stats\_2020.plot(x='SO', y='ERA', kind='scatter');**



**import seaborn as sns; sns.set\_style('ticks');**

**pitching\_stats\_2021.plot(x='SO', y='ERA', kind='scatter');**



The above scatter plots simply visualize each season we are analyzing and compares its Strikeouts to ERA. We can see that there is a correlation between Strikeouts and ERA. I chose to compare these two pitching stats because even with the higher offensive climate in baseball. The game has a clear three true outcome approach where teams prefer an aggressive hitting strategy where often hitters either strike out, hit a home run, or walk. It is interesting to see the 2019 season have such a spike in ERA but also a spike in strike outs which seems counter intuitive but does follow the modern-day strategy many teams subscribe to (more on that later).

**Analyzing Statcast Data From the 2018-2021 MLB Seasons**

In this section we will analyze statcast statistics for the 2018, 2019, 2020, 2021 seasons. I pulled this data in .csv format from baseballsavant.com which is MLBs own site dedicated to providing player matchups, Statcast metrics, and advanced statistics in a simple and easy-to-view way. We will use pandas to read the csv and put each season into a dataframe.

We will analyze and compare the average hit and homerun distances as well as the max hit and max distance speeds of all 30 MLB teams.

**# exit velocity data averaged by team and season per df**

**import pandas as pd**

**hitting\_velo\_2018 = pd.read\_csv("hitting 2018 exit\_velocity.csv")**

**hitting\_velo\_2019 = pd.read\_csv("hitting 2019 exit\_velocity.csv")**

**hitting\_velo\_2020 = pd.read\_csv("hitting 2020 exit\_velocity.csv")**

**hitting\_velo\_2021 = pd.read\_csv("hitting 2021 exit\_velocity.csv")**

**hitting\_velo\_2018.mean()**

season 2018.00000

team\_id 128.70000

attempts 4209.46667

avg\_hit\_angle 12.34333

anglesweetspotpercent 33.44333

max\_hit\_speed 116.07333

avg\_hit\_speed 88.41000

fbld 92.33333

gb 85.95000

max\_distance 465.83333

avg\_distance 172.16667

avg\_hr\_distance 397.36667

ev95plus 1492.66667

ev95per-swing 13.33667

ev95percent 35.46000

barrels 281.00000

brl\_percent 6.68000

brl\_pa 4.55000

dtype: float64

Last season with normal balls:

Max Distance – 466

Max Speed – 116

Avg HR Distance - 397

**hitting\_velo\_2019.mean()**

season 2019.00000

team\_id 128.70000

attempts 4191.83333

avg\_hit\_angle 12.67667

anglesweetspotpercent 33.54000

max\_hit\_speed 116.05000

avg\_hit\_speed 88.72333

fbld 92.67000

gb 86.07667

max\_distance 473.50000

avg\_distance 175.63333

avg\_hr\_distance 400.20000

ev95plus 1529.03333

ev95per-swing 13.31667

ev95percent 36.47333

barrels 307.56667

brl\_percent 7.32333

brl\_pa 4.94000

dtype: float64

Juiced ball season:

Max Distance – 473.5

Max Speed – 116

Avg HR Distance - 400

**hitting\_velo\_2020.mean()**

season 2020.00000

team\_id 128.70000

attempts 1465.73333

avg\_hit\_angle 12.72333

anglesweetspotpercent 33.40333

max\_hit\_speed 114.85667

avg\_hit\_speed 88.43667

fbld 92.80333

gb 85.32000

max\_distance 458.73333

avg\_distance 168.53333

avg\_hr\_distance 400.80000

ev95plus 550.90000

ev95per-swing 13.65333

ev95percent 37.56000

barrels 111.76667

brl\_percent 7.60000

brl\_pa 5.04333

dtype: float64

COVID shortened season:

Max Distance – 458

Max Speed – 114

Avg HR Distance - 401

**hitting\_velo\_2021.mean()**

season 2021.00000

team\_id 128.70000

attempts 4056.86667

avg\_hit\_angle 12.54333

anglesweetspotpercent 33.47000

max\_hit\_speed 116.55000

avg\_hit\_speed 88.76000

fbld 92.97333

gb 85.73000

max\_distance 469.00000

avg\_distance 166.96667

avg\_hr\_distance 400.93333

ev95plus 1569.60000

ev95per-swing 14.07000

ev95percent 38.67667

barrels 321.26667

brl\_percent 7.92333

brl\_pa 5.29667

dtype: float64

Season when balls went back to normal:

Max Distance – 469

Max Speed – 116.5

Avg HR Distance - 401

When comparing the 2018-2021 seasons statcast data it doesn’t seem like there is anything that jumps out. The 2019 season did have a slight increase in average Max Distance but the Max Hit Speeds and Average Home Run Distances are either identical or negligible in their differences.

**Predictive Model of 2020 MLB statcast Statistics (Linear Regression)**

In this section we will build a linear regression algorithm that predicts what a full 162 game season would have produced as it relates to Statcast data. We realized in the last section that our data from the 2018, 2019, and 2021 seasons showed that there was not anything significantly different as it relates to in-game (statcast) data. Beginning in this section I strongly recommend following with the google colab notebook or GitHub repository open because most of the code and outputs are robust and are not easily displayed in this format.

**import seaborn as sns**

**import sklearn as sk**

**import pybaseball as pyb**

**from matplotlib import pyplot as plt**

**from pybaseball import cache**

**cache.enable()**

**import warnings; warnings.filterwarnings('ignore')**

We begin again by importing all our past libraries.

**# exit velocity data averaged by team and season per df**

**import pandas as pd**

**hitting\_velo\_2018 = pd.read\_csv("hitting 2018 exit\_velocity.csv")**

**hitting\_velo\_2019 = pd.read\_csv("hitting 2019 exit\_velocity.csv")**

**hitting\_velo\_2020 = pd.read\_csv("hitting 2020 exit\_velocity.csv")**

**hitting\_velo\_2021 = pd.read\_csv("hitting 2021 exit\_velocity.csv")**

**hitting\_velo\_2018 = hitting\_velo\_2018**

**hitting\_velo\_2019 = hitting\_velo\_2019**

**hitting\_velo\_2020 = hitting\_velo\_2020**

**hitting\_velo\_2021 = hitting\_velo\_2021**

Here we concatenate the 4 .csv files into 1 data frame

**total\_velo\_stats = pd.concat([hitting\_velo\_2018, hitting\_velo\_2019, hitting\_velo\_2020 , hitting\_velo\_2021], axis=0)**

Here we concatenate only the 2018, 2019, and 2020 .csv files so we have 1 big data frame.

**velo\_stats\_2018\_2019 = pd.concat([hitting\_velo\_2018, hitting\_velo\_2019,hitting\_velo\_2020], axis=0)**

**total\_velo\_stats\_copy = total\_velo\_stats.copy()**

**In this chuck we are filtering our total\_velo\_stats data frame since it naturally has a lot of columns we will not use.**

**total\_velo\_stats\_copy = total\_velo\_stats\_copy.loc[:, ['season','team','max\_hit\_speed','avg\_hit\_speed', 'max\_distance', 'avg\_hr\_distance']]**

**total\_velo\_stats\_copy['max\_hit\_speed\_2020'] = total\_velo\_stats\_copy.sort\_values(['season','team'], ascending=False).groupby('team')['max\_hit\_speed'].shift()**

**total\_velo\_stats\_copy['avg\_hit\_speed\_2020'] = total\_velo\_stats\_copy.sort\_values(['season','team'], ascending=False).groupby('team')['avg\_hit\_speed'].shift()**

**total\_velo\_stats\_copy['max\_distance\_2020'] = total\_velo\_stats\_copy.sort\_values(['season','team'], ascending=False).groupby('team')['max\_distance'].shift()**

**total\_velo\_stats\_copy['avg\_hr\_distance\_2020'] = total\_velo\_stats\_copy.sort\_values(['season','team'], ascending=False).groupby('team')['avg\_hr\_distance'].shift()**

**total\_velo\_stats\_copy = total\_velo\_stats\_copy.loc[total\_velo\_stats\_copy['max\_hit\_speed\_2020'].notnull()]**

**total\_velo\_stats\_copy = total\_velo\_stats\_copy.loc[total\_velo\_stats\_copy['avg\_hit\_speed\_2020'].notnull()]**

**total\_velo\_stats\_copy = total\_velo\_stats\_copy.loc[total\_velo\_stats\_copy['max\_distance\_2020'].notnull()]**

**total\_velo\_stats\_copy = total\_velo\_stats\_copy.loc[total\_velo\_stats\_copy['avg\_hr\_distance\_2020'].notnull()]**

**total\_velo\_stats\_copy**

Here we will find the correlation (if any) for each column/statistical metric in the dataframe

**total\_velo\_stats\_copy.corr()**

Splitting our data for x, y training and testing

**from sklearn.model\_selection import train\_test\_split**

**x = total\_velo\_stats\_copy[['max\_hit\_speed','avg\_hit\_speed', 'max\_distance', 'avg\_hr\_distance']].values**

**y= total\_velo\_stats\_copy[['max\_hit\_speed\_2020', 'avg\_hit\_speed\_2020', 'max\_distance\_2020', 'avg\_hr\_distance\_2020']].values**

**print('Original Data Shape - X: {0}, Y: {1}'.format(x.shape, y.shape))**

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=42)       #test size is 20% of original**

**print('Train Data Shape - X{0}, Y:{1}.'.format(x\_train.shape, y\_train.shape) )**

**print('Test Data Shape - X{0}, Y:{1}.'.format(x\_test.shape, y\_test.shape) )**

Original Data Shape - X: (90, 4), Y: (90, 4)

Train Data Shape - X(72, 4), Y:(72, 4).

Test Data Shape - X(18, 4), Y:(18, 4).

Taking our training data and fitting it into our regression model

**from sklearn.linear\_model import LinearRegression**

**from sklearn.metrics import mean\_absolute\_error**

**lr = LinearRegression()**

**lr.fit(x\_train, y\_train)**

LinearRegression()

**y\_pred = lr.predict(x\_train)**

**print ('Mean number of hits:', x\_train[:, 0].mean())**

**print ('Mean absolute error:', mean\_absolute\_error(y\_pred, y\_train))**

Mean number of hits: 115.70000000000002

Mean absolute error: 3.698991863153225

**velo\_stats\_2018\_2019\_copy = velo\_stats\_2018\_2019.copy()**

**velo\_stats\_2018\_2019\_copy = velo\_stats\_2018\_2019\_copy.loc[:,['season','team','max\_hit\_speed','avg\_hit\_speed', 'max\_distance', 'avg\_hr\_distance']]**

**velo\_stats\_2018\_2019\_copy['2019\_actual\_max\_hit\_speed'] = velo\_stats\_2018\_2019\_copy.sort\_values(['team', 'season'], ascending=False).groupby('team')['max\_hit\_speed'].shift()**

**velo\_stats\_2018\_2019\_copy['2019\_actual\_avg\_hit\_speed'] = velo\_stats\_2018\_2019\_copy.sort\_values(['team', 'season'], ascending=False).groupby('team')['avg\_hit\_speed'].shift()**

**velo\_stats\_2018\_2019\_copy['2019\_actual\_max\_distance'] = velo\_stats\_2018\_2019\_copy.sort\_values(['team', 'season'], ascending=False).groupby('team')['max\_distance'].shift()**

**velo\_stats\_2018\_2019\_copy['2019\_actual\_avg\_hr\_distance'] = velo\_stats\_2018\_2019\_copy.sort\_values(['team', 'season'], ascending=False).groupby('team')['avg\_hr\_distance'].shift()**

**velo\_stats\_2018\_2019\_copy = velo\_stats\_2018\_2019\_copy.loc[velo\_stats\_2018\_2019\_copy['2019\_actual\_max\_hit\_speed'].notnull()]**

**velo\_stats\_2018\_2019\_copy**

Assigning an x and y value to our dataframes.

**x = velo\_stats\_2018\_2019\_copy[['max\_hit\_speed','avg\_hit\_speed', 'max\_distance', 'avg\_hr\_distance' ]].values**

**y = velo\_stats\_2018\_2019\_copy[['2019\_actual\_max\_hit\_speed','2019\_actual\_avg\_hit\_speed', '2019\_actual\_max\_distance', '2019\_actual\_avg\_hr\_distance' ]].values**

**y\_pred = lr.predict(x)**

**print('Mean of Stats:', velo\_stats\_2018\_2019\_copy.mean())**

**print('Mean absolute error:', mean\_absolute\_error(y\_pred, y))**

Mean of Stats: season 2018.50000

max\_hit\_speed 116.06167

avg\_hit\_speed 88.56667

max\_distance 469.66667

avg\_hr\_distance 398.78333

2019\_actual\_max\_hit\_speed 115.45333

2019\_actual\_avg\_hit\_speed 88.58000

2019\_actual\_max\_distance 466.11667

2019\_actual\_avg\_hr\_distance 400.50000

dtype: float64

Mean absolute error: 4.025842800890172

Assigning our Y predict variable our sorted data frame.

**velo\_stats\_2018\_2019\_copy[['predicted\_max\_hit\_speed', 'predicted\_avg\_hit\_speed', 'predicted\_max\_distance', 'predicted\_avg\_hr\_distance', ]] = y\_pred**

**velo\_stats\_2018\_2019\_copy['season'] = 2019**

**#pitch\_2019\_copy = pitch\_2019\_copy.rename(columns={'2019\_actual\_g':'Actual\_G'}**

**#use sort\_values to find the top predicted hits**

**velo\_stats\_2018\_2019\_copy = velo\_stats\_2018\_2019\_copy.loc[:,['season','team','2019\_actual\_max\_hit\_speed', '2019\_actual\_avg\_hit\_speed', '2019\_actual\_max\_distance', '2019\_actual\_avg\_hr\_distance', 'predicted\_max\_hit\_speed' , 'predicted\_avg\_hit\_speed', 'predicted\_max\_distance', 'predicted\_avg\_hr\_distance',  ]]**

Filtering said data frame to show us the stats we are looking for.

**predicted\_2020\_stats = velo\_stats\_2018\_2019\_copy.loc[:,['team', 'predicted\_max\_hit\_speed', 'predicted\_avg\_hit\_speed', 'predicted\_max\_distance', 'predicted\_avg\_hr\_distance']]**

**predicted\_2020\_stats**

**predicted\_2020\_stats.mean()**

predicted\_max\_hit\_speed 116.06841

predicted\_avg\_hit\_speed 88.66975

predicted\_max\_distance 467.52805

predicted\_avg\_hr\_distance 400.17873

dtype: float64

Max Distance – 467

Max Hit Speed – 116

Average HR Distance – 400.2

Even with our algorithm predicting a full 2020 season using data from previous seasons. Our statcast numbers don’t say much. They are a very close to the other seasons and don’t really give us a lot of insight on whether or not a hypothetical 2020 season would have played out closer to the “normal” 2018 season or “juiced” 2019 season.

**Predictive Model of 2020 MLB Hitting Statistics (Linear Regression)**

In this section we will build a linear regression algorithm that predicts what a full 162 game season would have produced hitting numbers wise.

Creating 2 different data frames; one for cumulative team hitting statistics for the 2015 through 2019 seasons and one for 2018 through 2019.

**hits\_df = pyb.team\_batting(2015, 2019)**

**hits\_2019 = pyb.team\_batting(2018, 2019)**

Creating our df for predicting a full 2020 season (we filter only for the 'Season', 'Team', 'G', 'AB', 'H', 'HR', 'RBI', 'AVG', 'OBP', 'SLG', 'OPS', 'wRC', 'WAR' columns since its what will be useful in our analysis.)

**hits\_df\_copy = hits\_df.copy()**

**hits\_df\_copy = hits\_df\_copy.loc[:, ['Season', 'Team', 'G', 'AB', 'H', 'HR', 'RBI', 'AVG', 'OBP', 'SLG', 'OPS', 'wRC', 'WAR']]**

**hits\_df\_copy['G\_Next\_Year'] = hits\_df\_copy.sort\_values(['Season', 'Team'], ascending=False).groupby('Team')['G'].shift()**

**hits\_df\_copy['H\_Next\_Year'] = hits\_df\_copy.sort\_values(['Season', 'Team'], ascending=False).groupby('Team')['H'].shift()**

**hits\_df\_copy['AB\_Next\_Year'] = hits\_df\_copy.sort\_values(['Season', 'Team'], ascending=False).groupby('Team')['AB'].shift()**

**hits\_df\_copy['HR\_Next\_Year'] = hits\_df\_copy.sort\_values(['Season', 'Team'], ascending=False).groupby('Team')['HR'].shift()**

**hits\_df\_copy['RBI\_Next\_Year'] = hits\_df\_copy.sort\_values(['Season', 'Team'], ascending=False).groupby('Team')['RBI'].shift()**

**hits\_df\_copy['AVG\_Next\_Year'] = hits\_df\_copy.sort\_values(['Season', 'Team'], ascending=False).groupby('Team')['AVG'].shift()**

**hits\_df\_copy['SLG\_Next\_Year'] = hits\_df\_copy.sort\_values(['Season', 'Team'], ascending=False).groupby('Team')['SLG'].shift()**

**hits\_df\_copy['wRC\_Next\_Year'] = hits\_df\_copy.sort\_values(['Season', 'Team'], ascending=False).groupby('Team')['wRC'].shift()**

**hits\_df\_copy['WAR\_Next\_Year'] = hits\_df\_copy.sort\_values(['Season', 'Team'], ascending=False).groupby('Team')['WAR'].shift()**

**hits\_df\_copy = hits\_df\_copy.loc[hits\_df\_copy['G\_Next\_Year'].notnull()]**

**hits\_df\_copy = hits\_df\_copy.loc[hits\_df\_copy['H\_Next\_Year'].notnull()]**

**hits\_df\_copy = hits\_df\_copy.loc[hits\_df\_copy['AB\_Next\_Year'].notnull()]**

**hits\_df\_copy = hits\_df\_copy.loc[hits\_df\_copy['HR\_Next\_Year'].notnull()]**

**hits\_df\_copy = hits\_df\_copy.loc[hits\_df\_copy['RBI\_Next\_Year'].notnull()]**

**hits\_df\_copy = hits\_df\_copy.loc[hits\_df\_copy['AVG\_Next\_Year'].notnull()]**

**hits\_df\_copy = hits\_df\_copy.loc[hits\_df\_copy['SLG\_Next\_Year'].notnull()]**

**hits\_df\_copy = hits\_df\_copy.loc[hits\_df\_copy['wRC\_Next\_Year'].notnull()]**

**hits\_df\_copy = hits\_df\_copy.loc[hits\_df\_copy['WAR\_Next\_Year'].notnull()]**

**hits\_df\_copy**

Finding the correlation between the columns and seasons (if any)

**hits\_df\_copy.corr()**

Splitting our data for x, y training and testing

**from sklearn.model\_selection import train\_test\_split**

**X = hits\_df\_copy[['G', 'AB', 'H', 'HR', 'RBI', 'AVG', 'SLG', 'wRC', 'WAR' ]].values**

**y= hits\_df\_copy[['G\_Next\_Year', 'AB\_Next\_Year', 'H\_Next\_Year', 'HR\_Next\_Year', 'RBI\_Next\_Year', 'AVG\_Next\_Year', 'SLG\_Next\_Year', 'wRC\_Next\_Year', 'WAR\_Next\_Year' ]].values**

**print('Original Data Shape - X: {0}, Y: {1}'.format(X.shape, y.shape))**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)       #test size is 20% of original**

**print('Train Data Shape - X{0}, Y:{1}.'.format(X\_train.shape, y\_train.shape) )**

**print('Test Data Shape - X{0}, Y:{1}.'.format(X\_test.shape, y\_test.shape) )**

Taking our training data and fitting it into our regression model

**from sklearn.linear\_model import LinearRegression**

**from sklearn.metrics import mean\_absolute\_error**

**lr = LinearRegression()**

**lr.fit(X\_train, y\_train)**

LinearRegression()

**y\_pred = lr.predict(X\_train)**

**print ('Mean number of hits:', X\_train[:, 0].mean())**

**print ('Mean absolute error:', mean\_absolute\_error(y\_pred, y\_train))**

Mean number of hits: 2364.7916666666665

Mean absolute error: 29.59835920038931

Making a copy of our 2019 hits data frame and filtering for useful columns.

**hits\_2019\_copy = hits\_2019.copy()**

**hits\_2019\_copy = hits\_2019\_copy.loc[:,['Season', 'Team', 'G', 'AB', 'H', 'HR', 'RBI', 'AVG', 'OBP', 'SLG', 'OPS', 'wRC', 'WAR']]**

**hits\_2019\_copy['2019\_actual\_g'] = hits\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['G'].shift()**

**hits\_2019\_copy['2019\_actual\_ab'] = hits\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['AB'].shift()**

**hits\_2019\_copy['2019\_actual\_hits'] = hits\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['H'].shift()**

**hits\_2019\_copy['2019\_actual\_hr'] = hits\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['HR'].shift()**

**hits\_2019\_copy['2019\_actual\_rbi'] = hits\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['RBI'].shift()**

**hits\_2019\_copy['2019\_actual\_avg'] = hits\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['AVG'].shift()**

**hits\_2019\_copy['2019\_actual\_slg'] = hits\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['SLG'].shift()**

**hits\_2019\_copy['2019\_actual\_wRC'] = hits\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['wRC'].shift()**

**hits\_2019\_copy['2019\_actual\_WAR'] = hits\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['WAR'].shift()**

**hits\_2019\_copy = hits\_2019\_copy.loc[hits\_2019\_copy['2019\_actual\_hits'].notnull()]**

**hits\_2019\_copy**

Assigning values to our X and Y

**X = hits\_2019\_copy[['G', 'AB', 'H', 'HR', 'RBI', 'AVG', 'SLG', 'wRC', 'WAR', ]].values**

**y = hits\_2019\_copy[['2019\_actual\_g', '2019\_actual\_ab','2019\_actual\_hits', '2019\_actual\_hr', '2019\_actual\_rbi', '2019\_actual\_avg', '2019\_actual\_slg', '2019\_actual\_wRC', '2019\_actual\_WAR' ]].values**

**y\_pred = lr.predict(X)**

**print('Mean of Stats:', hits\_2019\_copy.mean())**

**print('Mean absolute error:', mean\_absolute\_error(y\_pred, y))**

Mean of Stats: Season 2018.00000

G 2386.33333

AB 5514.40000

H 1367.26667

HR 186.16667

RBI 686.86667

AVG 0.24783

OBP 0.31797

SLG 0.40923

OPS 0.72723

wRC 719.33333

WAR 19.01333

2019\_actual\_g 2389.46667

2019\_actual\_ab 5555.03333

2019\_actual\_hits 1401.30000

2019\_actual\_hr 225.86667

2019\_actual\_rbi 749.03333

2019\_actual\_avg 0.25217

2019\_actual\_slg 0.43467

2019\_actual\_wRC 780.53333

2019\_actual\_WAR 18.99000

dtype: float64

Mean absolute error: 34.894989354342385

Making the new copy of the data frame = y-pred

**hits\_2019\_copy[['Predicted\_G', 'Predicted\_AB', 'Predicted\_H', 'Predicted\_HR', 'Predicted\_RBI', 'Predicted\_AVG', 'Predicted\_SLG', 'Predicted\_wRC', 'Predicted\_WAR']] = y\_pred**

**hits\_2019\_copy['Season'] = 2019**

**#hits\_2019\_copy = hits\_2019\_copy.rename(columns={'2019\_actual\_g':'Actual\_G'})**

**#use sort\_values to find the top predicted hits**

**hits\_2019\_copy = hits\_2019\_copy.loc[:,['Season','Team','2019\_actual\_g', '2019\_actual\_ab', '2019\_actual\_hits', '2019\_actual\_hr', '2019\_actual\_rbi' , '2019\_actual\_avg' , '2019\_actual\_slg', '2019\_actual\_wRC', '2019\_actual\_WAR', 'Predicted\_G', 'Predicted\_AB', 'Predicted\_H', 'Predicted\_HR', 'Predicted\_RBI', 'Predicted\_AVG', 'Predicted\_SLG', 'Predicted\_wRC', 'Predicted\_WAR']]**

**hits\_2019\_copy**

**predicted\_2020\_stats = hits\_2019\_copy.loc[:,['Team', 'Predicted\_G', 'Predicted\_AB', 'Predicted\_H', 'Predicted\_HR', 'Predicted\_RBI', 'Predicted\_AVG', 'Predicted\_SLG', 'Predicted\_wRC', 'Predicted\_WAR']]**

**predicted\_2020\_stats**

Finding the averages of all the columns in our data frame

**predicted\_2020\_stats.mean()**

Predicted\_G 2383.03539

Predicted\_AB 5533.80713

Predicted\_H 1391.50597

Predicted\_HR 210.97001

Predicted\_RBI 728.96278

Predicted\_AVG 0.25142

Predicted\_SLG 0.42629

Predicted\_wRC 758.35881

Predicted\_WAR 19.84809

dtype: float64

Finding the sum of all the columns in our data frame

**predicted\_2020\_stats.sum()**

Team BOSNYYLADCLEOAKHOUWSNCOLTBRMILCHCATLTORCINSTLTEXLAAMINSEAPITPHIARINYMCHWKCRBALDETSDPSFGMIA

Predicted\_G 71491.06157

Predicted\_AB 166014.21378

Predicted\_H 41745.17895

Predicted\_HR 6329.10020

Predicted\_RBI 21868.88340

Predicted\_AVG 7.54257

Predicted\_SLG 12.78855

Predicted\_wRC 22750.76427

Predicted\_WAR 595.44285

dtype: object

H= 41,745

HR= 6,329

RBI= 21,868

WAR= 595

AVG= .251

SLG%= .426

Our algorithm predicted that a full 2020 season would have produced higher offensive production than the “normal” 2018 season and on par with the “juiced” 2019 season.

**Predictive Model of 2020 MLB Pitching Statistics (Linear Regression)**

In this section we will build a linear regression algorithm that predicts what a full 162 game season would have produced pitching numbers wise.

Creating 2 different data frames; one for cumulative team pitching statistics for the 2015 through 2019 seasons and one for 2018 through 2019.

**pitch\_df = pyb.team\_pitching(2015, 2019)**

**pitch\_2019 = pyb.team\_pitching(2018, 2019)**

**pitch\_df**

Creating our df for predicting a full 2020 season

**pitch\_df\_copy = pitch\_df.copy()**

**pitch\_df\_copy = pitch\_df\_copy.loc[:, ['Season','Team', 'G', 'H', 'SO', 'ERA', 'H/9', 'HR/9', 'EV']]**

**pitch\_df\_copy['G\_Next\_Year'] = pitch\_df\_copy.sort\_values(['Season','Team'], ascending=False).groupby('Team')['G'].shift()**

**pitch\_df\_copy['H\_Next\_Year'] = pitch\_df\_copy.sort\_values(['Season','Team'], ascending=False).groupby('Team')['H'].shift()**

**pitch\_df\_copy['SO\_Next\_Year'] = pitch\_df\_copy.sort\_values(['Season','Team'], ascending=False).groupby('Team')['SO'].shift()**

**pitch\_df\_copy['ERA\_Next\_Year'] = pitch\_df\_copy.sort\_values(['Season','Team'], ascending=False).groupby('Team')['ERA'].shift()**

**pitch\_df\_copy['H/9\_Next\_Year'] = pitch\_df\_copy.sort\_values(['Season','Team'], ascending=False).groupby('Team')['H/9'].shift()**

**pitch\_df\_copy['HR/9\_Next\_Year'] = pitch\_df\_copy.sort\_values(['Season','Team'], ascending=False).groupby('Team')['HR/9'].shift()**

**pitch\_df\_copy['EV\_Next\_Year'] = pitch\_df\_copy.sort\_values(['Season','Team'], ascending=False).groupby('Team')['EV'].shift()**

**pitch\_df\_copy = pitch\_df\_copy.loc[pitch\_df\_copy['G\_Next\_Year'].notnull()]**

**pitch\_df\_copy = pitch\_df\_copy.loc[pitch\_df\_copy['H\_Next\_Year'].notnull()]**

**pitch\_df\_copy = pitch\_df\_copy.loc[pitch\_df\_copy['SO\_Next\_Year'].notnull()]**

**pitch\_df\_copy = pitch\_df\_copy.loc[pitch\_df\_copy['ERA\_Next\_Year'].notnull()]**

**pitch\_df\_copy = pitch\_df\_copy.loc[pitch\_df\_copy['H/9\_Next\_Year'].notnull()]**

**pitch\_df\_copy = pitch\_df\_copy.loc[pitch\_df\_copy['HR/9\_Next\_Year'].notnull()]**

**pitch\_df\_copy = pitch\_df\_copy.loc[pitch\_df\_copy['EV\_Next\_Year'].notnull()]**

**pitch\_df\_copy**

Finding the correlation between the columns and seasons in our dataframe

**pitch\_df\_copy.corr()**

Splitting our data for x, y training and testing

**from sklearn.model\_selection import train\_test\_split**

**x = pitch\_df\_copy[['G', 'H', 'SO', 'ERA', 'H/9', 'HR/9', 'EV']].values**

**y= pitch\_df\_copy[['G\_Next\_Year', 'H\_Next\_Year', 'SO\_Next\_Year', 'ERA\_Next\_Year', 'H/9\_Next\_Year','HR/9\_Next\_Year', 'EV\_Next\_Year']].values**

**print('Original Data Shape - X: {0}, Y: {1}'.format(x.shape, y.shape))**

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=42)       #test size is 20% of original**

**print('Train Data Shape - X{0}, Y:{1}.'.format(x\_train.shape, y\_train.shape) )**

**print('Test Data Shape - X{0}, Y:{1}.'.format(x\_test.shape, y\_test.shape) )**

Original Data Shape - X: (120, 7), Y: (120, 7)

Train Data Shape - X(96, 7), Y:(96, 7).

Test Data Shape - X(24, 7), Y:(24, 7).

Taking our training data and fitting it into our regression model

**from sklearn.linear\_model import LinearRegression**

**from sklearn.metrics import mean\_absolute\_error**

**lr = LinearRegression()**

**lr.fit(x\_train, y\_train)**

**y\_pred = lr.predict(x\_train)**

**print ('Mean number of hits:', x\_train[:, 0].mean())**

**print ('Mean absolute error:', mean\_absolute\_error(y\_pred, y\_train))**

Mean number of hits: 684.0104166666666

Mean absolute error: 22.399957928044778

Creating a copy of our pitch 2019 data frame and filtering for useful columns

**pitch\_2019\_copy = pitch\_2019.copy()**

**pitch\_2019\_copy = pitch\_2019\_copy.loc[:,['Season','Team', 'G', 'H', 'SO', 'ERA', 'H/9', 'HR/9', 'EV']]**

**pitch\_2019\_copy['2019\_actual\_g'] = pitch\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['G'].shift()**

**pitch\_2019\_copy['2019\_actual\_h'] = pitch\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['H'].shift()**

**pitch\_2019\_copy['2019\_actual\_so'] = pitch\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['SO'].shift()**

**pitch\_2019\_copy['2019\_actual\_era'] = pitch\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['ERA'].shift()**

**pitch\_2019\_copy['2019\_actual\_h/9'] = pitch\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['H/9'].shift()**

**pitch\_2019\_copy['2019\_actual\_hr/9'] = pitch\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['HR/9'].shift()**

**pitch\_2019\_copy['2019\_actual\_ev'] = pitch\_2019\_copy.sort\_values(['Team', 'Season'], ascending=False).groupby('Team')['EV'].shift()**

**pitch\_2019\_copy = pitch\_2019\_copy.loc[pitch\_2019\_copy['2019\_actual\_so'].notnull()]**

**pitch\_2019\_copy**

Assigning X and Y values

**x = pitch\_2019\_copy[['G', 'H', 'SO', 'ERA', 'H/9', 'HR/9','EV' ]].values**

**y = pitch\_2019\_copy[['2019\_actual\_g', '2019\_actual\_h','2019\_actual\_so', '2019\_actual\_era', '2019\_actual\_h/9', '2019\_actual\_hr/9','2019\_actual\_ev']].values**

**y\_pred = lr.predict(x)**

**print('Mean of Stats:', pitch\_2019\_copy.mean())**

**print('Mean absolute error:', mean\_absolute\_error(y\_pred, y))**

Mean of Stats: Season 2018.00000

G 706.53333

H 1367.26667

SO 1373.56667

ERA 4.15300

H/9 8.49200

HR/9 1.15667

EV 88.40000

2019\_actual\_g 714.30000

2019\_actual\_h 1401.30000

2019\_actual\_so 1427.43333

2019\_actual\_era 4.50800

2019\_actual\_h/9 8.71600

2019\_actual\_hr/9 1.40600

2019\_actual\_ev 88.70667

dtype: float64

Mean absolute error: 22.711032823684942

Making the new copy of the data frame = y-pred

**pitch\_2019\_copy[['Predicted\_G', 'Predicted\_H', 'Predicted\_SO', 'Predicted\_ERA', 'Predicted\_H/9', 'Predicted\_HR/9','Predicted\_EV']] = y\_pred**

**pitch\_2019\_copy['Season'] = 2019**

**#pitch\_2019\_copy = pitch\_2019\_copy.rename(columns={'2019\_actual\_g':'Actual\_G'}**

**#use sort\_values to find the top predicted hits**

**pitch\_2019\_copy = pitch\_2019\_copy.loc[:,['Season','Team','2019\_actual\_g', '2019\_actual\_h', '2019\_actual\_so', '2019\_actual\_era', '2019\_actual\_h/9' , '2019\_actual\_hr/9', 'Predicted\_G', 'Predicted\_H', 'Predicted\_SO', 'Predicted\_ERA', 'Predicted\_H/9', 'Predicted\_HR/9','Predicted\_EV' ]]**

**pitch\_2019\_copy**

Filtering our data frame with predicted 2020 stats

**predicted\_2020\_stats = pitch\_2019\_copy.loc[:,['Team', 'Predicted\_G', 'Predicted\_H', 'Predicted\_SO', 'Predicted\_ERA', 'Predicted\_H/9', 'Predicted\_HR/9','Predicted\_EV']]**

**predicted\_2020\_stats**

Finding the averages of the columns of our new predicted 2020 pitching data frame

**predicted\_2020\_stats.mean()**

Predicted\_G 707.78371

Predicted\_H 1382.33205

Predicted\_SO 1401.91147

Predicted\_ERA 4.35885

Predicted\_H/9 8.60626

Predicted\_HR/9 1.30995

Predicted\_EV 88.40525

dtype: float64

Finding the sum of the columns of our new predicted 2020 pitching data frame

**predicted\_2020\_stats.sum()**

Team HOULADCHCARIMILTBRATLBOSCLENYYOAKSTLSFGPITWSNNYMSEAPHILAACOLSDPMINDETCINMIACHWTORTEXKCRBAL

Predicted\_G 21233.51125

Predicted\_H 41469.96162

Predicted\_SO 42057.34419

Predicted\_ERA 130.76561

Predicted\_H/9 258.18794

Predicted\_HR/9 39.29857

Predicted\_EV 2652.15757

dtype: object

K= 42,607

ERA= 4.26

H/9 = 8.5

HR/9 = 1.26

EV=88.4

The algorithm predicts that a full 2020 season would have been on par with the “juiced” season of 2019. The stats for this hypothetical season are closer to 2019s number than 2018 or 2021 where the ball was normal. The ERA for this predicted season is not as high as 2019 but still much higher than 2018. While strikeouts also went way up which would fall in line with the three true outcome approach that we mentioned earlier.

**Results and Summary:**

Hitting Data:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Season | H | HR | RBI | WAR | AVG | SLG% |
| 2018 | 41,018 | 5,585 | 20,606 | 570 | .248 | .409 |
| 2019 | 42,039 | 6,776 | 22,471 | 570 | .252 | .435 |
| 2020 (60 Games) | 14,439 | 2,304 | 7,978 | 210 | .244 | .417 |
| **2020 (Predicted)** | **41,745** | **6,329** | **21,868** | **595** | **.251** | **.426** |
| 2021 | 39,484 | 5944 | 20,993 | 570 | .244 | .411 |

Here we see that the 2019 season was the season with the highest hitting production. This season had the highest number of hits, home runs, RBI’s, tied for most WAR, highest batting average and slugging %. Our predicted 2020 season is the 2nd highest hitting production season trailing only 2019 in every category except for WAR where it surpassed 2019. Given the fact that we have proof of there being inconsistencies with the baseballs in our 60-game sample size of the 2020 season we can safely predict that a full 2020 season would have likely been similarly as hit producing as 2019.

We can also see that our 2020 season of 60 games was very hitting productive in just that 37% of a season played. In those 60 games hitters were able to get 34% of the total hits from 2019, 34% of total home runs and 35% of the total RBI’s. I think it is safe to conclude given our predictions and statements from independent scientist and MLB themselves that a full 2020 season would have been played with “juiced” or altered baseballs that would have allowed for a more hitting friendly atmosphere.

Pitching Data:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Season | K | ERA | H/9 | HR/9 | Exit Velocity |
| 2018 | 41,207 | 4.15 | 8.49 | 1.16 | 88.4 |
| 2019 | 42,823 | 4.51 | 8.72 | 1.41 | 88.7 |
| 2020 (60 Games) | 15,586 | 4.46 | 8.40 | 1.34 | 88.4 |
| **2020 (Predicted)** | **42,607** | **4.26** | **8.5** | **1.26** | **88.4** |
| 2021 | 42,145 | 4.27 | 8.34 | 1.26 | 88.7 |

2018 was the most pitcher friendly season of all the seasons we analyzed which further shows that a full 2020 season would have been played with “juiced” baseballs. 2019 saw a significant increase in strikeouts, ERA, H/9, and HR/9 but barely any difference in exit velocity.

Baseball can sometimes go through phases where the pace of play either favors hitters or pitchers. In this instance 2019 was favorable to hitters because of the change of baseball. This counterintuitively correlates with more strikeouts because hitters are trying to hit the ball harder knowing that there is a better chance for a home run since the balls are juiced. Basically, anytime there is a jump in home runs (which there was) expect a jump in strikeouts from hitters swinging out of their shoes more often.

Our 2020 60 game sample size also saw a decrease in pitching production which puts it on par with the 2019 and predicted 2020 season. 2020 had 36% of total strikeouts in just 37% of the games. This aligns with how 34% of the home runs in 2019 were hit in just 60 games in 2020.

There being a negligible difference in exit velocity is the most interesting find since I would have assumed there to be an increase there too with the baseballs being tailored to be more hitting friendly

Statcast Data:

|  |  |  |  |
| --- | --- | --- | --- |
| Season | Max Hit Distance | Max Hit Speed | Avg HR Distance |
| 2018 | 466 ft | 116 mph | 397.3 ft |
| 2019 | 473.5 ft | 116 mph | 400.2 ft |
| 2020 (60 Games) | 458 ft | 114 mph | 401 ft |
| **2020 (Predicted)** | **467 ft** | **116 mph** | **400.2 ft** |
| 2021 | 469 ft | 117 mph | 401 ft |

Oddly enough, when it comes to statcast data there is not much to compare as all the seasons including our predicted 2020 seem to be very close. The 2019 season was slightly higher in terms of the maximum distances of balls hit that season, maximum hit speed, and average home run distance which were 473.5 ft, 116 mph, and 400.2 ft respectively. Our predicted 2020 season was the 2nd highest with 467 ft, 116 mph, and 400.2 ft.

The actual 2020 60 game season was right on par with 2019 and 2020 (predicted) but so was 2021 which was supposed to be the season where the balls were “de-juiced”. This data shows us a small difference between 2019 and the seasons after that and 2018 when the balls were last normal. Given the fact that MLB was known to randomly change the baseballs at their own discretion it’s tough to not think about them choosing which games to juice the baseballs for and which games to use normal baseball’s for

Also considering there being barely a difference in exit velocity leads me to believe that there might be more to the juiced baseballs than just the seams being slightly taller or the balls being slightly lighter. All in all, this research leads me to make the claim that MLB would have allowed the 2020 season to play out with juiced baseballs like it did in 2019 and would ranked amongst one of the highest offense producing seasons in the history of professional baseball.

In conclusion, MLB surely has and continued to alter the baseballs at their leisure from 2019-2020. Our data all points to a full season of 2020 being like 2019 and not 2018. This begs the questioning of MLB’s reputation and whether they decide to switch the balls for certain games or even when certain players are playing. Given MLB’s dark history of gambling on games or steroid consumption it leaves a lot of open room for questions and debate

**References and Sources**

Apstein, S. (2021, February 9). *Did MLB use juiced balls in 2020?* Sports Illustrated. Retrieved February 8, 2022, from https://www.si.com/mlb/2021/02/09/mlb-ball-is-live-daily-cover

Brown, M. (2021, December 8). *Major League Baseball's integrity is at stake as study finds inconsistencies with the ball*. Forbes. Retrieved February 1, 2022, from https://www.forbes.com/sites/maurybrown/2021/12/06/major-league-baseballs-integrity-at-stake-due-to-inconsistencies-with-the-ball/?sh=ad797c7015c8

Feinsand, M. (2021, February 9). *MLB to alter baseballs for '21 season*. MLB.com. Retrieved February 1, 2022, from https://www.mlb.com/news/mlb-to-alter-baseballs-for-2021