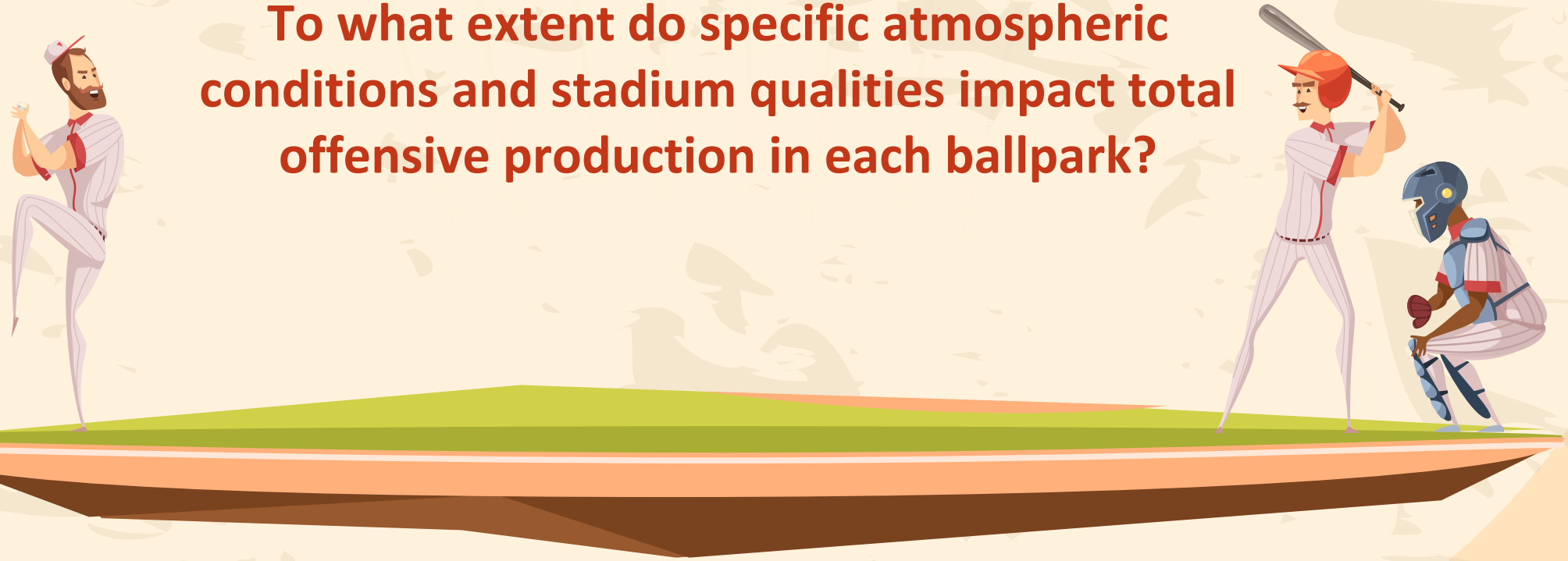
Exploring the Impact of Ballpark Factors and Atmospheric Conditions on Offensive Output of MLB teams

Andrea Contreras, Jay Irby, Sam Wright, Ben Lang

Research Question

To what extent do specific atmospheric conditions and stadium qualities impact total offensive production in each ballpark?



Introduction

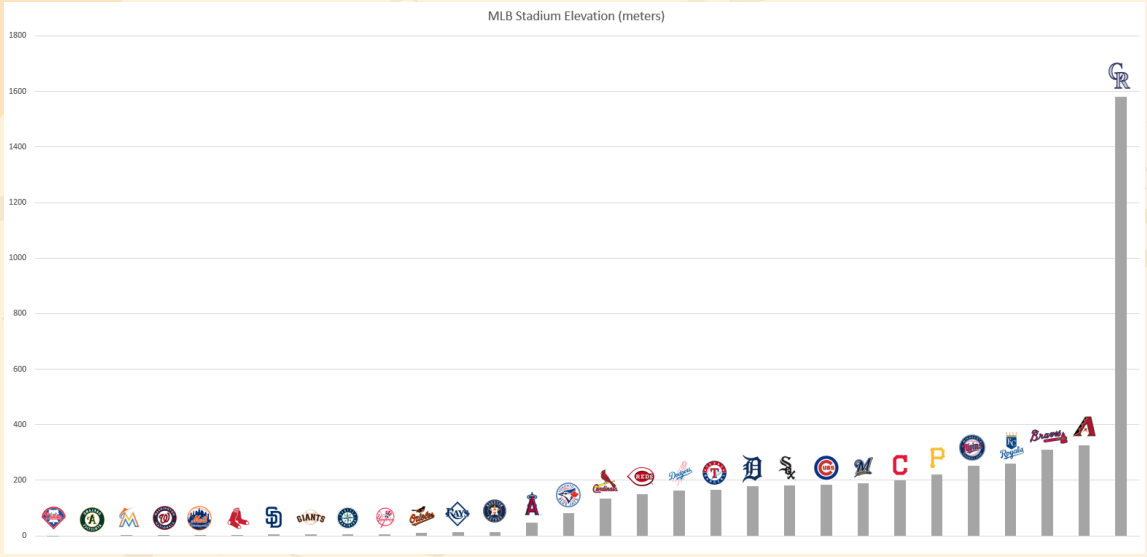
- Our goal: to determine why certain stadiums are more ‘hitter friendly’ than others by looking at specific conditions and impacting variables.
- Physics of Baseball
- Inverse relationship of **drag force** and **air density**.
- Atmospheric conditions and their relationship with offensive output.

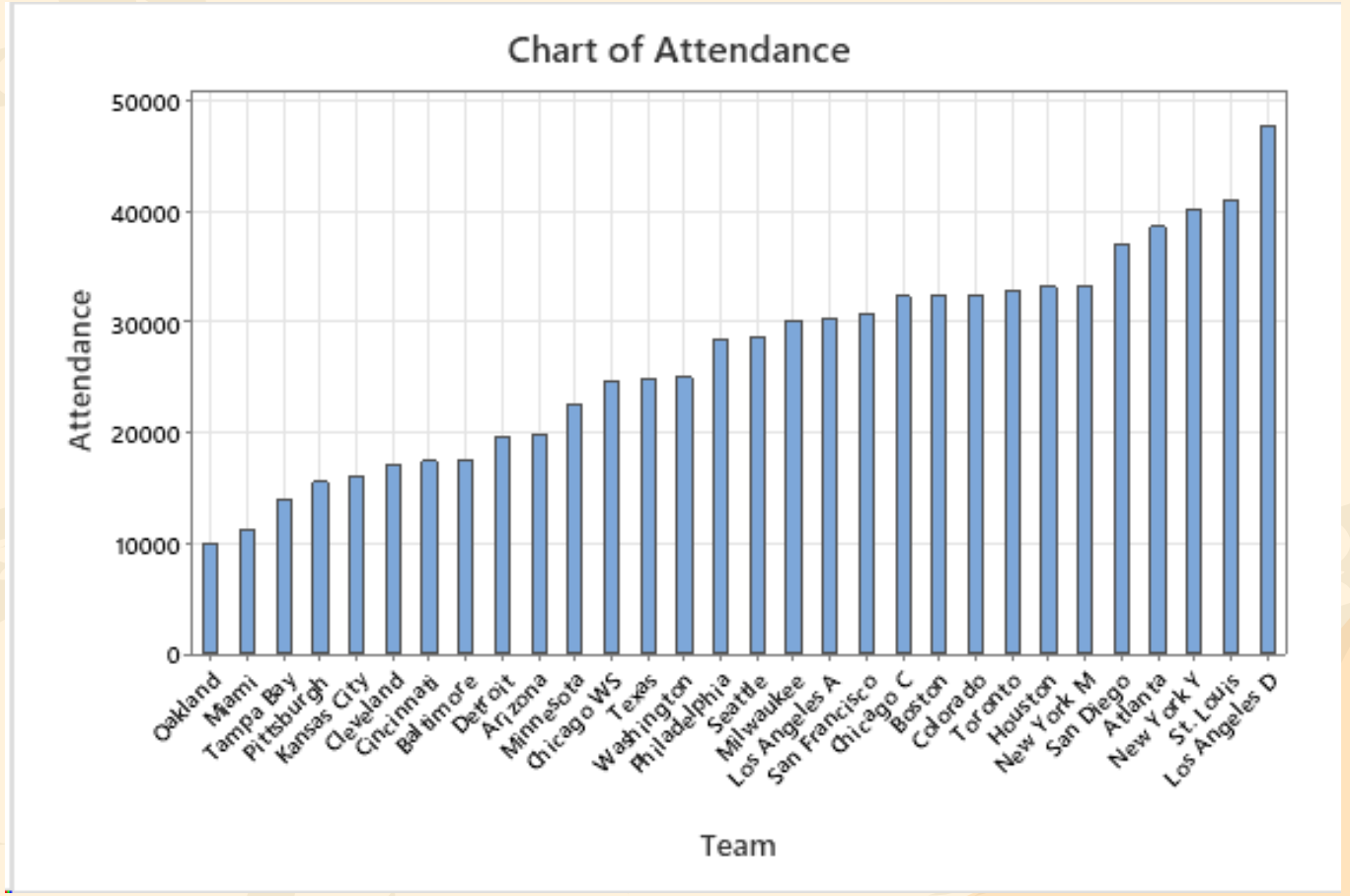
Discrepancies in the Results of Previous Literature

Atmospheric Conditions and Ballpark Qualities: Independent Variables

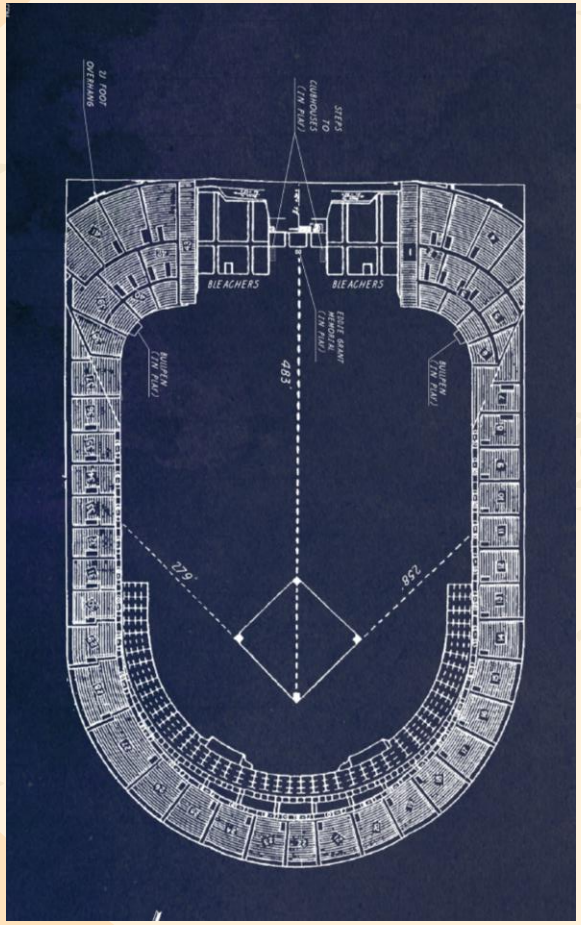
- Altitude (feet)
- Stadium Dimensions
  - Left Field length away from home plate (feet)
  - Center Field length away from home plate (feet)
  - Right Field length away from home plate (feet)
- Average Attendance (per game)
- Humidity (%)
- Stadium Type (indoor/outdoor)
- Average Temperature (Fahrenheit)

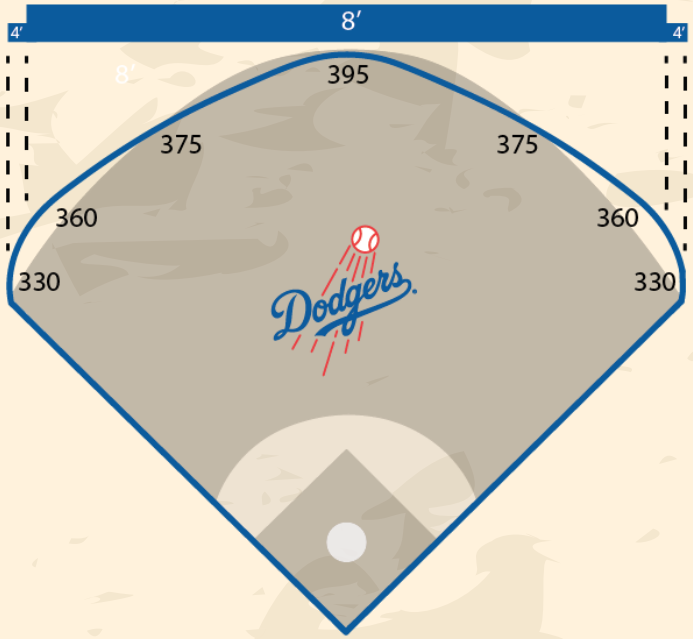
Other Effects

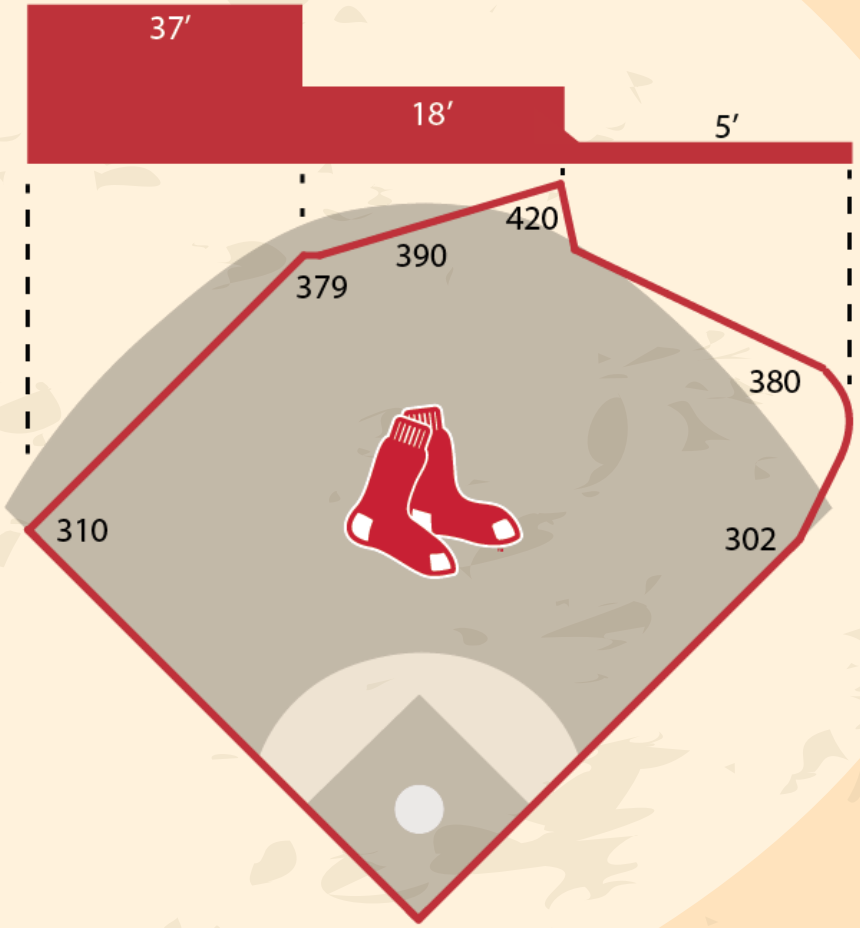




Different Stadium Sizes







Offensive Statistics: Dependent Variables

- Total Runs - R
- Home Runs – HR
- Weighted on-base percentage – wOBA

$$wOBA = \frac{.69\times uBB + .72\times HBP + .89\times 1B + 1.27\times 2B + 1.62\times 3B + 2.10\times HR}{AB + BB - IBB + SF + HBP}$$

- Ballpark factor – metric created by baseballsavant.com and other sites
  - How hitter friendly a ballpark is

Data Scraping

- Manually scraping data tables: fantasypros.com. Baseballsavant.com did not work
- Created a dictionary with key values corresponding to field and value with home runs
- Differences in field names/team names created discrepancies in data
- ballparkpal.com shut down
- BeautifulSoup data scraping

```
page = "https://www.fantasypros.com/mlb/park-factors.php" #scraping from website
html = requests.get(page)
soup = BeautifulSoup(html.text, 'html.parser')
table = soup.find("table", id="data-table")
rows = table.find_all('tr')
data = []
for row in rows: #pulling table from site
    cols = row.find_all('td')
    cols = [col.text.strip() for col in cols]
    data.append(cols)

# Print the extracted data
data = data[1:]
df = pd.DataFrame(data) #converting to data frame
df = df.iloc[::3, :]
df[[1,2,3,4,5]] = df[[1,2,3,4,5]].apply(pd.to_numeric)
df[[0]] = df[[0]].astype(str)
df['Average'] = df[[1,2,3,4,5]].mean(axis=1) #average ball park factor
df.rename(columns = {0:"field", 1:'runs', 2:'hr', 3:"1b", 4:"2b", 5:"3b"}, inplace = True)
df['Team Name'] = df['field'].str.extract(r"\((.*?)\)", expand=False)
sorted_df = df.sort_values(by=["Average"], ascending=False) #sort by averes
display(sorted_df)
```

```
df = pd.read_csv("datacapstone.csv") #importing manually scraped excel
df = pd.DataFrame(df)
df['Homeruns'] = df[["Homeruns 2021", "Homeruns 2022"]].mean(axis=1) #averaging home runs over 2021/2022
df[["Attendance"]] = df[["Attendance"]].apply(pd.to_numeric) #converting attendance to integers
df1 = df[["Team Name", "Stadium","Stadium Type", "Humidity (afternoons in march-oct)", "Left (feet)", "Right (feet)", "Center (feet)","Altitude", "WOBA", "Attendance", "Homerun
df1 = df1.sort_values(["Homeruns"], ascending=False) #sort in descending home run averages
display(df1[["Team Name", "Stadium", "Stadium Type", "Altitude", "Humidity (afternoons in march-oct)", "Left (feet)", "Right (feet)", "Center (feet)", "WOBA", "Altitude", "Atte
```

Data Massaging

Combined imported manually scraped data with the BeautifulSoup scraped data. Matched rows by team name column.

```
new_df = pd.merge(df1, sorted_df, on='Team Name') #merging two data frames by team
new_df = new_df[["Team Name", "Stadium", "Stadium Type", "Altitude", "Humidity (afternoons in march-oct)","Left (feet)", "Right (feet)", "Center (feet)", "Attendar
new_df.rename(columns = {"Humidity (afternoons in march-oct)": "Humidity", "Average":"Park Factor"}, inplace = True) #cleaning names
new_df = new_df.sort_values(by=["Homeruns"], ascending=False) #sort by home runs
display(new_df)
```

Statistical Analysis

Types

-Linear regression and multiple regression
-Matplotlib for visualization
-Mann-Whitney U t-test for sea level vs. non sea level comparison, and indoor vs. outdoor stadium comparison

Tables

We compiled data on Excel and made a table using Python listing characteristics of each stadium

	Team Name	Stadium	Stadium Type	Altitude	Humidity (afternoons in march-oct)	Left (feet)	Right (feet)	Center (feet)	WOBA	Altitude	Attendance	Homeruns
7	Cincinnati Reds	Great American Ball Park	Outdoor	550	56	327.0	325.0	404.0	112	550	17447	233.5
3	Baltimore Orioles	Camden Yards	Outdoor	20	52	384.0	318.0	400.0	95	20	17543	214.0
14	Los Angeles Dodgers	Dodger Stadium	Outdoor	340	56	330.0	330.0	395.0	107	340	47641	213.5
29	Washington Nationals	Nationals Park	Outdoor	7	52	336.0	335.0	402.0	99	7	25017	208.0
5	Chicago Cubs	Wrigley Field	Outdoor	595	56	355.0	353.0	400.0	99	595	32305	203.0
0	Los Angeles Angels	Angel Stadium	Outdoor	160	55	330.0	330.0	396.0	106	160	30339	202.0
9	Colorado Rockies	Coors Field	Outdoor	5280	39	347.0	350.0	415.0	113	5280	32467	201.0
16	Minnesota Twins	Target Field	Outdoor	815	55	339.0	328.0	411.0	99	815	22514	201.0
15	Milwaukee Brewers	Miller Park	Indoor	635	61	342.0	345.0	400.0	101	635	30155	200.0
2	Atlanta Braves	Truist Park	Outdoor	1050	54	335.0	325.0	400.0	103	1050	38641	200.0
27	Texas Rangers	Globe Life Field	Indoor	551	52	372.0	326.0	407.0	102	551	24831	196.0
12	Houston Astros	Minute Maid Park	Indoor	22	58	315.0	326.0	435.0	104	22	33197	194.0
20	Philadelphia Phillies	Citizens Bank Park	Outdoor	20	52	329.7	330.0	405.7	105	20	28459	193.5
24	Seattle Mariners	T-Mobile Park	Indoor	-2	57	331.0	326.0	401.0	93	-2	28590	193.0
6	Chicago White Sox	Guarenteed Rate Field	Outdoor	595	56	330.0	335.0	400.0	99	595	24704	191.5
4	Boston Red Sox	Fenway Park	Outdoor	21	57	310.0	302.0	420.0	108	21	32408	190.5
8	Cleveland Guardians	Progressive Field	Outdoor	660	58	325.0	325.0	410.0	95	660	17050	180.5
28	Toronto Blue Jays	Rogers Centre	Indoor	300	61	330.0	330.0	400.0	101	300	32763	179.0
19	Oakland Athletics	Oakland Coliseum	Outdoor	25	68	330.0	330.0	400.0	93	25	9973	178.0
22	San Diego Padres	Petco Park	Outdoor	20	64	329.3	327.3	402.0	90	20	36931	173.5
18	New York Yankees	Yankee Satdium	Outdoor	55	53	318.0	314.0	408.0	99	55	40207	168.0
26	Tampa Bay Rays	Tropicana Field	Indoor	15	58	315.0	322.0	404.0	99	15	13927	163.0
1	Arizona Diamondbacks	Chase Field	Indoor	1090	19	330.0	334.0	407.0	96	1090	19817	160.5
17	New York Mets	Citi Field	Outdoor	20	53	335.0	330.0	408.0	92	20	33308	159.5
25	St. Louis Cardinals	Busch Stadium	Outdoor	535	55	330.0	330.0	402.0	94	535	40994	158.0
11	Miami Marlins	Lonedepot Stadium	Indoor	10	61	344.0	345.0	407.0	103	10	11203	155.5
21	Pittsburgh Pirates	PNC Park	Outdoor	730	54	325.0	320.0	399.0	96	730	15524	153.5
23	San Francisco Giants	Oracle Park	Outdoor	0	68	339.0	309.0	399.0	102	0	30650	151.5
13	Kansas City Royals	Kauffman Stadium	Outdoor	750	58	330.0	330.0	403.3	98	750	15947	150.0
10	Detroit Tigers	Comerica Park	Outdoor	585	55	345.0	330.0	420.0	96	585	19634	138.5

```
X = new_df[["Stadium Type", "Altitude","Attendance", "Humidity", "Park Factor","Left (feet)", "Right (feet)", "Center (feet)"]] #independent vars
y = new_df[["Homeruns"]] #dependent
x = sm.add_constant(X)
x.reindex(y.index)
model = sm.OLS(y, x).fit() #creating regression model
print(model.summary())
```

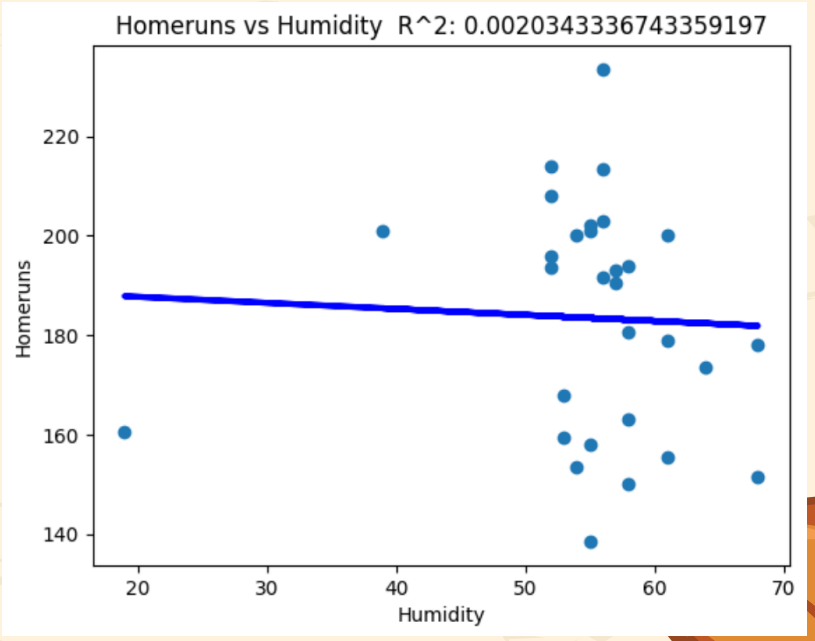
```
                            OLS Regression Results
==============================================================================
Dep. Variable:               Homeruns   R-squared:                       0.130
Model:                            OLS   Adj. R-squared:                 -0.202
Method:                 Least Squares   F-statistic:                    0.3914
Date:                Thu, 20 Apr 2023   Prob (F-statistic):              0.913
Time:                        14:24:16   Log-Likelihood:                -134.49
No. Observations:                  30   AIC:                             287.0
Df Residuals:                      21   BIC:                             299.6
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         74.8555    370.184      0.202      0.842    -694.984     844.695
Stadium Type  -1.8589     11.974     -0.155      0.878     -26.760      23.043
Altitude      -0.0001      0.008     -0.014      0.989      -0.016       0.016
Attendance     0.0006      0.001      1.172      0.254      -0.000       0.002
Humidity       0.0095      0.642      0.015      0.988      -1.326       1.345
Park Factor   16.0186     52.933      0.303      0.765     -94.061     126.098
Left (feet)    0.3206      0.339      0.947      0.355      -0.384       1.025
Right (feet)   0.1862      0.554      0.336      0.740      -0.965       1.337
Center (feet) -0.2276      0.677     -0.336      0.740      -1.635       1.180
==============================================================================
Omnibus:                        0.374   Durbin-Watson:                   0.352
Prob(Omnibus):                  0.829   Jarque-Bera (JB):                0.007
Skew:                           0.010   Prob(JB):                        0.996
Kurtosis:                       3.073   Cond. No.                     2.25e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.25e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Homeruns

Humidity Affecting Homeruns

```
x = new_df[["Humidity"]] #independent
y = new_df[["Homeruns"]] #dependent
model = LinearRegression() #running regression
model.fit(x, y)
r_squared = model.score(x, y)
regr = linear_model.LinearRegression() #regression line
regr.fit(x, y) #plotting
plt.plot(x,y, "o")
plt.plot(x, regr.predict(x), color='blue', linewidth=3) #plotting regression
plt.title(f"Homeruns vs Humidity  R^2: {r_squared}") #title
plt.xlabel("Humidity")
plt.ylabel("Homeruns")
x = sm.add_constant(x)
x.reindex(y.index)
model = sm.OLS(y, x).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               Homeruns   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                 -0.034
Method:                 Least Squares   F-statistic:                    0.05708
Date:                Fri, 21 Apr 2023   Prob (F-statistic):              0.813
Time:                        20:30:12   Log-Likelihood:                -136.55
No. Observations:                  30   AIC:                             277.1
Df Residuals:                      28   BIC:                             279.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        190.1540     28.393      6.697      0.000     131.993     248.315
Humidity      -0.1217      0.509     -0.239      0.813      -1.165       0.922
==============================================================================
Omnibus:                        1.165   Durbin-Watson:                   0.058
Prob(Omnibus):                  0.559   Jarque-Bera (JB):                0.896
Skew:                          -0.090   Prob(JB):                        0.639
Kurtosis:                       2.173   Cond. No.                         365.
==============================================================================
```



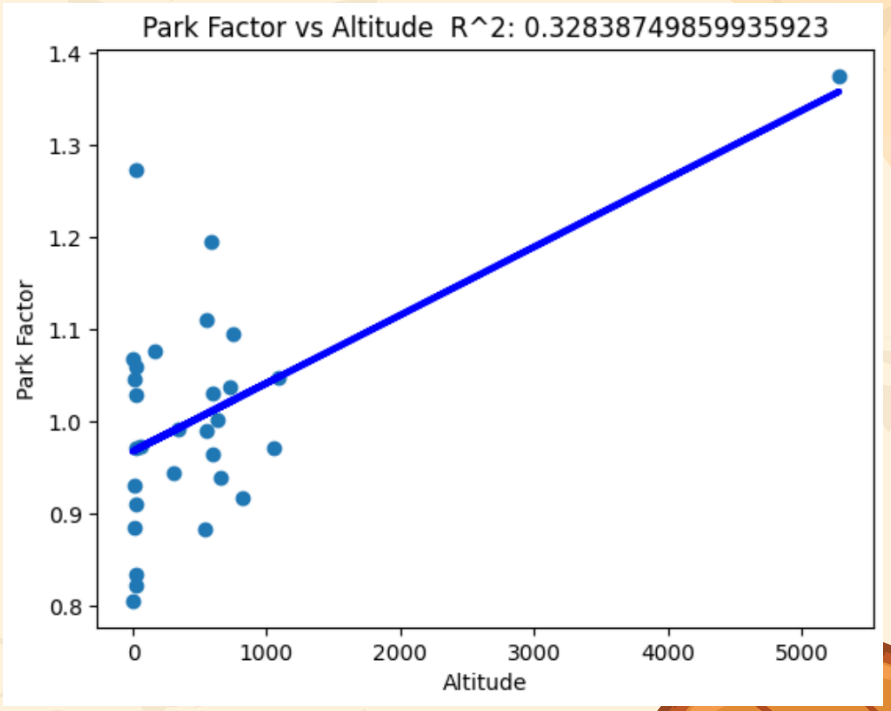
```
X = new_df[["Stadium Type", "Altitude","Attendance", "Humidity","Left (feet)", "Right (feet)", "Center (feet)"]] #independent
y = new_df[["Park Factor"]] #dependent
x = sm.add_constant(X)
x.reindex(y.index)
model = sm.OLS(y, x).fit() #regression
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            Park Factor   R-squared:                       0.478
Model:                            OLS   Adj. R-squared:                  0.312
Method:                 Least Squares   F-statistic:                     2.879
Date:                Thu, 20 Apr 2023   Prob (F-statistic):             0.0271
Time:                        14:25:07   Log-Likelihood:                 30.247
No. Observations:                  30   AIC:                            -44.49
Df Residuals:                      22   BIC:                            -33.28
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            -0.4315      1.488     -0.290      0.775      -3.518       2.655
Stadium Type     -0.0462      0.047     -0.980      0.338      -0.144       0.052
Altitude       7.135e-05   2.68e-05      2.666      0.014    1.58e-05       0.000
Attendance    -1.535e-06   2.06e-06     -0.745      0.464   -5.81e-06    2.74e-06
Humidity         -0.0003      0.003     -0.107      0.916      -0.006       0.005
Left (feet)       0.0003      0.001      0.210      0.836      -0.003       0.003
Right (feet)     -0.0016      0.002     -0.730      0.473      -0.006       0.003
Center (feet)     0.0047      0.003      1.850      0.078      -0.001       0.010
==============================================================================
Omnibus:                        0.679   Durbin-Watson:                   2.215
Prob(Omnibus):                  0.712   Jarque-Bera (JB):                0.681
Skew:                           0.019   Prob(JB):                        0.712
Kurtosis:                       2.263   Cond. No.                     2.25e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.25e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Park Factor

```python
x = new_df[["Altitude"]] #independent
y = new_df[["Park Factor"]] #dependent
model = LinearRegression() # running regression
model.fit(x, y)
r_squared = model.score(x, y)
regr = linear_model.LinearRegression() #getting regression line
regr.fit(x, y)
plt.plot(x,y, "o") #plotting
plt.plot(x, regr.predict(x), color='blue', linewidth=3) #plotting line
plt.title(f"Park Factor vs Altitude  R^2: {r_squared}")
plt.xlabel("Altitude")
plt.ylabel("Park Factor")
x = sm.add_constant(x)
x.reindex(y.index)
model = sm.OLS(y, x).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            Park Factor   R-squared:                       0.328
Model:                            OLS   Adj. R-squared:                  0.304
Method:                 Least Squares   F-statistic:                     13.69
Date:                Fri, 21 Apr 2023   Prob (F-statistic):           0.000933
Time:                        20:35:14   Log-Likelihood:                 26.463
No. Observations:                  30   AIC:                            -48.93
Df Residuals:                      28   BIC:                            -46.12
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.9679      0.022     44.935      0.000       0.924       1.012
Altitude    7.384e-05      2e-05      3.700      0.001     3.3e-05       0.000
==============================================================================
Omnibus:                        6.046   Durbin-Watson:                   1.787
Prob(Omnibus):                  0.049   Jarque-Bera (JB):                4.283
Skew:                           0.785   Prob(JB):                        0.117
Kurtosis:                       3.981   Cond. No.                     1.23e+03
==============================================================================
```

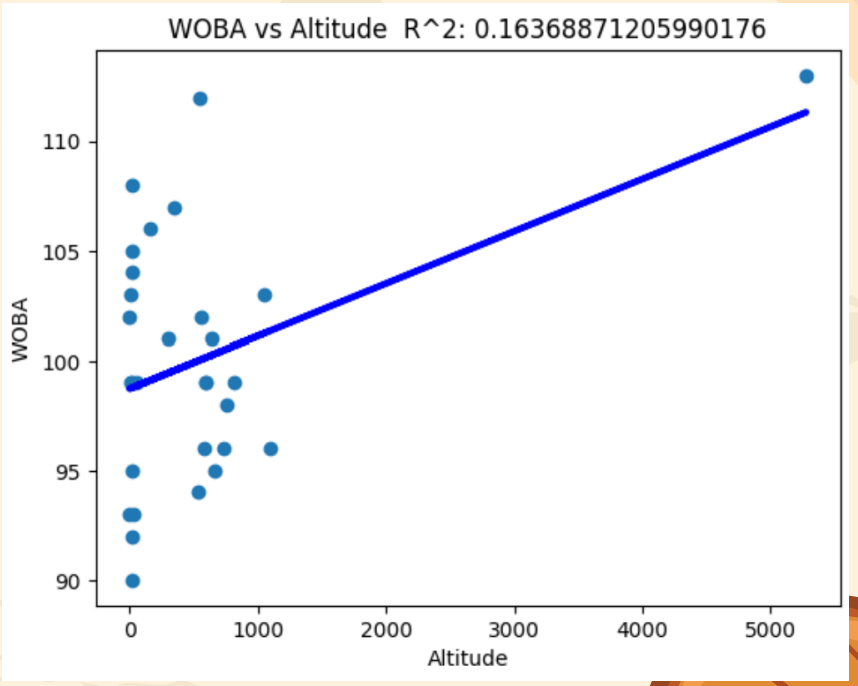


```
X = new_df[["Altitude","Attendance", "Humidity","Left (feet)", "Right (feet)", "Center (feet)"]] #independent
y = new_df[["WOBA"]] #dependent
x = sm.add_constant(X)
x.reindex(y.index)
model = sm.OLS(y, x).fit() #regression
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   WOBA   R-squared:                       0.237
Model:                            OLS   Adj. R-squared:                 -0.006
Method:                 Least Squares   F-statistic:                    0.9754
Date:                Thu, 20 Apr 2023   Prob (F-statistic):              0.473
Time:                        14:25:56   Log-Likelihood:                -90.136
No. Observations:                  30   AIC:                             196.3
Df Residuals:                      22   BIC:                             207.5
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          99.7945     82.294      1.213      0.238     -70.872     270.461
Stadium Type    1.2526      2.611      0.480      0.636      -4.162       6.667
Altitude        0.0030      0.001      2.057      0.052    -2.5e-05       0.006
Attendance   9.047e-05      0.000      0.794      0.436      -0.000       0.000
Humidity        0.0809      0.143      0.565      0.577      -0.216       0.377
Left (feet)    -0.0311      0.075     -0.413      0.684      -0.187       0.125
Right (feet)   -0.0608      0.122     -0.499      0.623      -0.313       0.192
Center (feet)   0.0538      0.140      0.383      0.705      -0.237       0.345
==============================================================================
Omnibus:                        1.261   Durbin-Watson:                   1.726
Prob(Omnibus):                  0.532   Jarque-Bera (JB):                0.680
Skew:                           0.367   Prob(JB):                        0.712
Kurtosis:                       3.070   Cond. No.                     2.25e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.25e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```python
x = new_df[["Altitude"]] #independent var
y = new_df[["WOBA"]] #dependent var
model = LinearRegression() #run regression
model.fit(x, y)
r_squared = model.score(x, y)
regr = linear_model.LinearRegression() #fit reg line
regr.fit(x, y)
plt.plot(x,y, "o") #plot values
plt.plot(x, regr.predict(x), color='blue', linewidth=3) #plot line
plt.title(f"WOBA vs Altitude  R^2: {r_squared}") #labels
plt.xlabel("Altitude")
plt.ylabel("WOBA")
x = sm.add_constant(x)
x.reindex(y.index)
model = sm.OLS(y, x).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   WOBA   R-squared:                       0.164
Model:                            OLS   Adj. R-squared:                  0.134
Method:                 Least Squares   F-statistic:                     5.480
Date:                Fri, 21 Apr 2023   Prob (F-statistic):             0.0266
Time:                        20:37:26   Log-Likelihood:                -91.509
No. Observations:                  30   AIC:                             187.0
Df Residuals:                      28   BIC:                             189.8
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         98.7386      1.099     89.835      0.000      96.487     100.990
Altitude       0.0024      0.001      2.341      0.027       0.000       0.004
==============================================================================
Omnibus:                        1.256   Durbin-Watson:                   1.955
Prob(Omnibus):                  0.534   Jarque-Bera (JB):                1.210
Skew:                           0.402   Prob(JB):                        0.546
Kurtosis:                       2.432   Cond. No.                     1.23e+03
==============================================================================
```



Mann-Whitney U T-test: Indoor vs. Outdoor Ballparks

- Converted scraped data to CSV file
- Utilized 0 and 1 values to indicate outdoor/indoor
- Used for loops to create 2 lists (group1 and group2), outdoor and indoor WOBA values
- Used scipy mannwhitneyu() function
- Yielded a very high p-value at 0.76
- Fail to reject the null hypothesis, no statistical significance between indoor/outdoor stadiums in terms of WOBA

```python
import pandas as pd
import scipy.stats as stats

# Import scraped data file
data = pd.read_csv('datacapstone.csv')

# Create x and y variables for 'for loop', x indicating stadium type (independent) and y indicating WOBA (dependent)
x = data['Stadium Type']
y = data['WOBA']

# Initialize variables as empty lists
index1 = [] # index1 represents all 'zeros' or outdoor stadiums
index2 = [] # index2 represents all 'ones' or indoor stadiums

# Iterate through x to indicate which indexes are either indoor/outdoor
for i in range(len(x)):
    if x[i] == 0:
        index1.append(i)
    elif x[i] == 1:
        index2.append(i)

# Initialize group1 and group2 variables as empty lists
group1 = [] # group1 contains outdoor WOBA values
group2 = [] # group2 contains indoor WOBA values

# Iterate through index1 and index2, appending WOBA values to each respective list.
for j in index1:
    group1.append(y[j])

for i in index2:
    group2.append(y[i])

# Use mannwhitneyu() with group1 and group2 as parameters
statistic, p_value = stats.mannwhitneyu(group1, group2)

# Print the test statistic and p-value
print(f'Mann-Whitney U statistic: {statistic}')
print(f'p-value: {p_value}')

# Set significance level (alpha)
alpha = 0.05

# Check for statistical significance
if p_value < alpha:
    print("Reject the null hypothesis: There is a statistically significant difference between the two groups.")
else:
    print("Fail to reject the null hypothesis: There is no statistically significant difference between the two groups.")
```

```
Mann-Whitney U statistic: 81.0
p-value: 0.7593462474990416
Fail to reject the null hypothesis: There is no statistically significant difference between the two groups.
```

Mann-Whitney U: Stadiums as Sea Level vs. Stadiums not at Sea Level

- Similar methodology to prior slide, except used if statements to index stadiums
- Sea Level: anything under 200 feet elevation (mostly inland vs. coastal)
- P-value is 0.428 for ‘Total Runs’, which is not significant at all three regular p values (0.1, 0.05, 0.01).

```
#Mann-Whitney U t-test- sea level vs non sea level stadiums: Total Runs
data = pd.read_csv('datacapstone.csv') #import scraped csv file

x = data['Altitude'] #create x variable to extract altitude values for each stadium
y = data['Total Runs 2022'] #create y variable to extract total run values for each stadium

#initialize index1 and index2 as empty lists
index1 = [] #for non sea level
index2 = [] #for sea level

#iterate through x to append indexes that either represent sea level or non sea level stadiums
for i in range(len(x)):
  if x[i] >= 200: #greater than or equal to 200 feet altitude is non sea level
    index1.append(i)
  elif x[i] < 200: #less than 200 feet altitude is sea level
    index2.append(i)


#initialize group1 and group2 variables as empty lists
group1 = [] #group1 contains non sea level total run values
group2 = [] #group2 contains sea level total run values

for j in index1: #iterate through index1, appending total run values to group1 list
  group1.append(y[j])

for i in index2: #iterate through index2, appending total run values to group2 list
  group2.append(y[i])

#use mannwhitneyu() with group1 and group2 as parameters
statistic, p_value = stats.mannwhitneyu(group1, group2)
print(f'Mann-Whitney U statistic: {statistic}') #print test statistic
print(f'p-value: {p_value}') #print p-value
```

```
Mann-Whitney U statistic: 131.5
p-value: 0.4278683341581393
```

Future Implications

Probability Analysis

In the event that atmospheric factors or field size produced any sort of notable difference in offensive output, generate a probability analysis using Python package scipy.stats.

Further Research Questions

-Are there specific types of pitches (e.g. fastballs, breaking balls, etc.) that are more effective in certain atmospheric conditions, and how does this affect offensive performance?

-How do different playing surfaces, such as natural grass, artificial turf, and dirt infields, affect the ability of a baseball team to produce runs and effective offensive output?

While there is little to medium correlation for the relationship between all of our measured independent/dependent variables, there are likely outside factors that we were unable to discover that affect offensive output. We cannot completely and accurately measure a correlation without a controlled environment.

Conclusion

References

- https://scholars.carroll.edu/bitstream/handle/20.500.12647/3435/1998_EatonS_THS_000306.pdf?sequence=1&isAllowed=y
- https://www.researchgate.net/publication/228668381_Effects_of_altitude_and_atmospheric_conditions_on_the_flight_of_a_baseball
- https://sabr.org/journal/article/high-altitude-offense-an-empirical-examination-of-the-relationship-between-runs-scored-and-stadium-elevation/.
- https://www.proquest.com/docview/89070489?pq-origsite=gscholar&fromopenview=true
- https://www.skybrary.aero/articles/density-altitude#:~:text=Density%20and%20pressure%2Ftemperature&text=As%20pressure%20increases%2C%20with%20temperature,%C2%B0C%20increase%20in%20temperature.
- https://www.proquest.com/scholarly-journals/park-elevation-long-ball-flight-major-league/docview/89070489/se-2
- https://www.ncei.noaa.gov/pub/data/ccd-data/relhum20.dat
- https://www.ncei.noaa.gov/data/global-summary-of-the-day/access/
- https://www.fantasypros.com/mlb/park-factors.php