Data Aggregation and

Preprocessing and Tensor Setup

Data Preprocessing

Data cleaning, normalization and processing and addition of advanced statistics pulled from external sources.

batting_pre

March 9, 2020

[120]:
```
import math
import numpy as np
import pandas as pd

# We're going to be reassigning some columns, so we'll turn off this warning -␣
↪we know what we're doing!
pd.options.mode.chained_assignment = None  # default='warn'
```

[121]:
```
# This will be exported to a separate module
ids = pd.read_csv('../data/lahman/mlb_data/People.csv')
ids = ids[['playerID', 'retroID']]
id_dict = ids.set_index('playerID').to_dict()['retroID']

def get_retroid(id):
    return id_dict[id] if id_dict is not None else id
```

[122]:
```
df = pd.read_csv('../data/lahman/mlb_data/Batting.csv').sort_values('playerID')
```

[123]:
```
df['playerID'] = df['playerID'].apply(get_retroid)
```

[124]:
```
df.rename(columns={'playerID': 'retroID'}, inplace=True)
```

[125]:
```
df[df['retroID'] == None]
```

[125]:
```
Empty DataFrame
Columns: [retroID, yearID, stint, teamID, lgID, G, AB, R, H, 2B, 3B, HR, RBI,
SB, CS, BB, SO, IBB, HBP, SH, SF, GIDP]
Index: []

[0 rows x 22 columns]
```

Cleaning the Data - Missing Values

Print percentages of missing data in each column of the batting table

[126]:
```
100 * df.isnull().sum() / len(df)
```

[126]:
```
retroID     0.000000
yearID      0.000000
stint       0.000000
teamID      0.000000
lgID        0.000000
G           0.000000
AB          0.000000
R           0.000000
H           0.000000
2B          0.000000
3B          0.000000
HR          0.000000
RBI         0.000000
SB          0.000000
CS          8.221708
BB          0.000000
SO          0.000000
IBB        21.711883
HBP         0.000000
SH          0.000000
SF         21.090864
GIDP        9.839985
dtype: float64
```

Since this data is by season, it's likely that we have entries for a player for one season with no data in these fields but there is data for other seasons. Since we're taking aggregate sums for each player, we have two options: set these null values to zero so they don't add to the sum, or set them to the average for that player. We'll have to test the theory to see which is more viable.

We're going to start with IBB rather than CS, since it's a more significant chunk of the dataset.

Handling missing IBB data

[127]:
```
df[(df['IBB'].isnull())]
```

[127]:
```
        retroID  yearID  stint teamID lgID    G   AB   R    H  2B  ...  RBI  \
19269  aaroh101    1954      1    ML1   NL  122  468  58  131  27  ...   69
18684  abera101    1953      2    DET   AL   17   23   2    3   0  ...    2
16858  abera101    1950      1    CLE   AL    1    2   0    0   0  ...    0
19270  abera101    1954      1    DET   AL   32   39   3    5   0  ...    3
18683  abera101    1953      1    CLE   AL    6    0   0    0   0  ...    0
...         ...     ...    ...    ...  ...  ...  ...  ..  ...  ..  ...  ...
15128  zubeb101    1946      1    NYA   AL    3    2   0    0   0  ...    0
14447  zubeb101    1945      1    NYA   AL   21   42   1    7   0  ...    3
13868  zubeb101    1944      1    NYA   AL   22   31   1    4   0  ...    1
19843  zuveg101    1954      1    CIN   NL    2    2   1    1   0  ...    0
19844  zuveg101    1954      2    DET   AL   35   64   1    8   1  ...    3
```

```
       SB   CS  BB  SO  IBB  HBP  SH   SF  GIDP
19269   2  2.0  28  39  NaN    3   6  4.0  13.0
18684   0  0.0   1   6  NaN    0   1  NaN   0.0
16858   0  0.0   1   1  NaN    0   0  NaN   0.0
19270   0  0.0   2  17  NaN    0   3  1.0   1.0
18683   0  0.0   2   0  NaN    0   0  NaN   0.0
...    ..  ...  ..  ..  ...  ...  ..  ...   ...
15128   0  0.0   0   1  NaN    0   0  NaN   0.0
14447   0  0.0   1  13  NaN    0   2  NaN   1.0
13868   0  0.0   0  10  NaN    0   4  NaN   1.0
19843   0  0.0   0   1  NaN    0   0  0.0   0.0
19844   0  1.0   1  14  NaN    0   9  0.0   2.0

[19159 rows x 22 columns]
```

[128]:
```
df[(df['retroID'] == 'abera101')]
```

[128]:
```
        retroID  yearID  stint teamID lgID   G  AB  R  H  2B  ...  RBI  SB  \
19846  abera101    1955      1    DET   AL  39  17  0  1   0  ...    0   0
18684  abera101    1953      2    DET   AL  17  23  2  3   0  ...    2   0
16858  abera101    1950      1    CLE   AL   1   2  0  0   0  ...    0   0
19270  abera101    1954      1    DET   AL  32  39  3  5   0  ...    3   0
18683  abera101    1953      1    CLE   AL   6   0  0  0   0  ...    0   0
21123  abera101    1957      2    KC1   AL   3   1  0  1   0  ...    0   0
20501  abera101    1956      1    DET   AL  42  10  0  3   0  ...    0   0
21122  abera101    1957      1    DET   AL  28   8  0  1   0  ...    1   0

        CS  BB  SO  IBB  HBP  SH   SF  GIDP
19846  0.0   0   9  0.0    0   2  0.0   1.0
18684  0.0   1   6  NaN    0   1  NaN   0.0
16858  0.0   1   1  NaN    0   0  NaN   0.0
19270  0.0   2  17  NaN    0   3  1.0   1.0
18683  0.0   2   0  NaN    0   0  NaN   0.0
21123  0.0   0   0  0.0    0   0  0.0   0.0
20501  0.0   1   4  0.0    0   2  0.0   0.0
21122  0.0   1   4  0.0    0   0  0.0   0.0

[8 rows x 22 columns]
```

[129]:
```
df[(df['retroID'] == 'zubeb101')]
```

[129]:
```
        retroID  yearID  stint teamID lgID   G  AB  R  H  2B  ...  RBI  SB  \
9445   zubeb101    1936      1    CLE   AL   2   5  1  1   0  ...    0   0
10501  zubeb101    1938      1    CLE   AL  15   7  0  0   0  ...    0   0
11080  zubeb101    1939      1    CLE   AL  16   5  0  1   0  ...    0   0
11621  zubeb101    1940      1    CLE   AL  17   3  0  1   0  ...    0   0
12203  zubeb101    1941      1    WS1   AL  36  26  0  0   0  ...    0   0
```

```
12742  zubeb101    1942       1    WS1    AL  37  39  5  6    3  ...    3   0
13299  zubeb101    1943       1    NYA    AL  20  38  1  7    1  ...    2   0
15711  zubeb101    1947       1    BOS    AL  20  13  0  2    0  ...    0   0
15129  zubeb101    1946       2    BOS    AL  15  18  1  2    0  ...    2   0
15128  zubeb101    1946       1    NYA    AL   3   2  0  0    0  ...    0   0
14447  zubeb101    1945       1    NYA    AL  21  42  1  7    0  ...    3   0
13868  zubeb101    1944       1    NYA    AL  22  31  1  4    0  ...    1   0

        CS  BB  SO  IBB  HBP  SH  SF  GIDP
9445   0.0   0   1  NaN    0   0 NaN   NaN
10501  0.0   0   1  NaN    0   1 NaN   NaN
11080  0.0   0   2  NaN    0   0 NaN   0.0
11621  0.0   0   0  NaN    0   0 NaN   0.0
12203  0.0   1   8  NaN    0   2 NaN   0.0
12742  0.0   1   7  NaN    0   3 NaN   2.0
13299  0.0   4  14  NaN    0   5 NaN   2.0
15711  0.0   2   3  NaN    0   2 NaN   2.0
15129  0.0   1   6  NaN    0   1 NaN   0.0
15128  0.0   0   1  NaN    0   0 NaN   0.0
14447  0.0   1  13  NaN    0   2 NaN   1.0
13868  0.0   0  10  NaN    0   4 NaN   1.0

[12 rows x 22 columns]
```

First let's look at IBB, intentional bases on balls. It seems like most of the missing data is from early in the dataset - it could be that IBB was not recorded then, and/or not considered a trackable play?

[130]:
```
df[(df['IBB'].isnull())]['yearID'].max()
```

[130]:
```
1954
```

[131]:
```
df[(df['IBB'].isnull())]['yearID'].min()
```

[131]:
```
1919
```

[132]:
```
df[(df['IBB'].isnull())]
```

[132]:
```
        retroID  yearID  stint teamID lgID    G   AB   R    H  2B  ...  RBI  \
19269  aaroh101    1954      1    ML1   NL  122  468  58  131  27  ...   69
18684  abera101    1953      2    DET   AL   17   23   2    3   0  ...    2
16858  abera101    1950      1    CLE   AL    1    2   0    0   0  ...    0
19270  abera101    1954      1    DET   AL   32   39   3    5   0  ...    3
18683  abera101    1953      1    CLE   AL    6    0   0    0   0  ...    0
...         ...     ...    ...    ...  ...  ...  ...  ..  ...  ..  ...  ...
15128  zubeb101    1946      1    NYA   AL    3    2   0    0   0  ...    0
14447  zubeb101    1945      1    NYA   AL   21   42   1    7   0  ...    3
```

```
13868  zubeb101    1944      1    NYA   AL   22   31   1    4   0  ...    1
19843  zuveg101    1954      1    CIN   NL    2    2   1    1   0  ...    0
19844  zuveg101    1954      2    DET   AL   35   64   1    8   1  ...    3

       SB   CS  BB  SO  IBB  HBP  SH   SF  GIDP
19269   2  2.0  28  39  NaN    3   6  4.0  13.0
18684   0  0.0   1   6  NaN    0   1  NaN   0.0
16858   0  0.0   1   1  NaN    0   0  NaN   0.0
19270   0  0.0   2  17  NaN    0   3  1.0   1.0
18683   0  0.0   2   0  NaN    0   0  NaN   0.0
...    ..  ...  ..  ..  ...  ...  ..  ...   ...
15128   0  0.0   0   1  NaN    0   0  NaN   0.0
14447   0  0.0   1  13  NaN    0   2  NaN   1.0
13868   0  0.0   0  10  NaN    0   4  NaN   1.0
19843   0  0.0   0   1  NaN    0   0  0.0   0.0
19844   0  1.0   1  14  NaN    0   9  0.0   2.0

[19159 rows x 22 columns]
```

We have 19159 total rows where there is no data for IBB, and we know none of those rows goes past the year 1954…

[133]:
```python
df[(df['yearID'] < 1955)]
```

[133]:
```
        retroID  yearID  stint teamID lgID    G   AB   R    H  2B  ...  RBI  \
19269  aaroh101    1954      1    ML1   NL  122  468  58  131  27  ...   69
18684  abera101    1953      2    DET   AL   17   23   2    3   0  ...    2
16858  abera101    1950      1    CLE   AL    1    2   0    0   0  ...    0
19270  abera101    1954      1    DET   AL   32   39   3    5   0  ...    3
18683  abera101    1953      1    CLE   AL    6    0   0    0   0  ...    0
...         ...     ...    ...    ...  ...  ...  ...  ..  ...  ..  ...  ...
13868  zubeb101    1944      1    NYA   AL   22   31   1    4   0  ...    1
18682  zuveg101    1952      1    CLE   AL    2    0   1    0   0  ...    0
19843  zuveg101    1954      1    CIN   NL    2    2   1    1   0  ...    0
19844  zuveg101    1954      2    DET   AL   35   64   1    8   1  ...    3
18050  zuveg101    1951      1    CLE   AL   16    0   0    0   0  ...    0

       SB   CS  BB  SO  IBB  HBP  SH   SF  GIDP
19269   2  2.0  28  39  NaN    3   6  4.0  13.0
18684   0  0.0   1   6  NaN    0   1  NaN   0.0
16858   0  0.0   1   1  NaN    0   0  NaN   0.0
19270   0  0.0   2  17  NaN    0   3  1.0   1.0
18683   0  0.0   2   0  NaN    0   0  NaN   0.0
...    ..  ...  ..  ..  ...  ...  ..  ...   ...
13868   0  0.0   0  10  NaN    0   4  NaN   1.0
18682   0  0.0   0   0  0.0    0   0  0.0   0.0
19843   0  0.0   0   1  NaN    0   0  0.0   0.0
```

```
19844    0  1.0    1  14  NaN    0   9  0.0   2.0
18050    0  0.0    0   0  0.0    0   0  0.0   0.0

[19845 rows x 22 columns]
```

And we have 19845 total rows up to the year 1954. That means…

[134]:
```
19159 / 19845
```

[134]:
```
0.9654320987654321
```

Over 96% of the data before 1955 is missing IBB. I think this gives justification to just setting all of those NaNs to 0.

[135]:
```
df['IBB'].fillna(value=0, inplace=True)
```

[136]:
```
100 * df.isnull().sum() / len(df)
```

[136]:
```
retroID     0.000000
yearID      0.000000
stint       0.000000
teamID      0.000000
lgID        0.000000
G           0.000000
AB          0.000000
R           0.000000
H           0.000000
2B          0.000000
3B          0.000000
HR          0.000000
RBI         0.000000
SB          0.000000
CS          8.221708
BB          0.000000
SO          0.000000
IBB         0.000000
HBP         0.000000
SH          0.000000
SF         21.090864
GIDP        9.839985
dtype: float64
```

Our IBB issue is solved. Let's move on to SF (sacrifice flies). We'll check the years and rows again to see if we're justified in using the same method to eliminate nulls.

Handling missing SF data

[137]:
```
df[(df['SF'].isnull())]['yearID'].max()
```

[137]: 1953

[138]:
```
df[(df['SF'].isnull())]['yearID'].min()
```

[138]: 1919

[139]:
```
df[(df['SF'].isnull())].shape[0]
```

[139]: 18611

[140]:
```
df[(df['yearID'] < 1954)].shape[0]
```

[140]: 19269

[141]:
```
18611/19269
```

[141]: 0.965851886449738

Almost the same percentage, and one less year covered. I think we can fill those missing values with 0.

[142]:
```
df['SF'].fillna(value=0, inplace=True)
```

[143]:
```
100 * df.isnull().sum() / len(df)
```

[143]:
```
retroID    0.000000
yearID     0.000000
stint      0.000000
teamID     0.000000
lgID       0.000000
G          0.000000
AB         0.000000
R          0.000000
H          0.000000
2B         0.000000
3B         0.000000
HR         0.000000
RBI        0.000000
SB         0.000000
CS         8.221708
BB         0.000000
SO         0.000000
IBB        0.000000
HBP        0.000000
SH         0.000000
SF         0.000000
GIDP       9.839985
dtype: float64
```

Two more to go, let's move on to CS (caught stealing)

Handling missing CS data

[144]:
```
df[(df['CS'].isnull())]['yearID'].max()
```

[144]:
```
1950
```

[145]:
```
df[(df['CS'].isnull())]['yearID'].min()
```

[145]:
```
1919
```

[146]:
```
df[(df['CS'].isnull())]
```

[146]:
```
        retroID  yearID  stint teamID lgID    G   AB   R   H  2B  ...  RBI  \
14448  aberw101    1946      1    NY1   NL   15    8   0   0   0  ...    0
16285  aberc101    1949      1    CHN   NL    4    7   0   0   0  ...    0
15712  aberc101    1948      1    CHN   NL   12   32   1   6   1  ...    6
15131  aberc101    1947      1    CHN   NL   47  140  24  39   6  ...   20
16286  abrac101    1949      1    BRO   NL    8   24   6   2   1  ...    0
...         ...     ...    ...    ...  ...  ...  ...  ..  ..  ..  ...  ...
532    zitzb101    1919      1    PIT   NL   11   26   5   5   1  ...    2
4782   zitzb101    1927      1    CIN   NL   88  232  47  66  10  ...   24
5312   zitzb101    1928      1    CIN   NL  101  266  53  79   9  ...   33
533    zitzb101    1919      2    CIN   NL    2    1   0   0   0  ...    0
5842   zitzb101    1929      1    CIN   NL   47   84  18  19   3  ...    6

       SB  CS  BB  SO  IBB  HBP  SH   SF  GIDP
14448   0 NaN   0   4  0.0    0   0  0.0   0.0
16285   0 NaN   0   2  0.0    0   0  0.0   1.0
15712   0 NaN   5  10  0.0    0   0  0.0   0.0
15131   0 NaN  20  32  0.0    0   0  0.0   5.0
16286   1 NaN   7   6  0.0    0   0  0.0   1.0
...    ..  ..  ..  ..  ...  ...  ..  ...   ...
532     2 NaN   0   6  0.0    0   1  0.0   NaN
4782    9 NaN  20  18  0.0    4  17  0.0   NaN
5312   13 NaN  13  22  0.0    3  14  0.0   NaN
533     0 NaN   0   0  0.0    0   0  0.0   NaN
5842    4 NaN   9  10  0.0    1   2  0.0   NaN

[7255 rows x 22 columns]
```

[147]:
```
df[(df['retroID'] == 'zitzb101')]
```

[147]:
```
       retroID  yearID  stint teamID lgID    G   AB   R   H  2B  ...  RBI  SB  \
4241  zitzb101    1926      1    CIN   NL   53   94  21  23   2  ...    3   3
532   zitzb101    1919      1    PIT   NL   11   26   5   5   1  ...    2   2
3715  zitzb101    1925      1    CIN   NL  104  301  53  76  13  ...   21  11
```

```
4782  zitzb101   1927    1   CIN  NL   88  232  47  66  10  ...  24   9
5312  zitzb101   1928    1   CIN  NL  101  266  53  79   9  ...  33  13
533   zitzb101   1919    2   CIN  NL    2    1   0   0   0  ...   0   0
5842  zitzb101   1929    1   CIN  NL   47   84  18  19   3  ...   6   4

        CS  BB  SO  IBB  HBP  SH   SF  GIDP
4241   NaN   6   7  0.0    2   3  0.0   NaN
532    NaN   0   6  0.0    0   1  0.0   NaN
3715  11.0  35  22  0.0    6   2  0.0   NaN
4782   NaN  20  18  0.0    4  17  0.0   NaN
5312   NaN  13  22  0.0    3  14  0.0   NaN
533    NaN   0   0  0.0    0   0  0.0   NaN
5842   NaN   9  10  0.0    1   2  0.0   NaN

[7 rows x 22 columns]
```

[148]:
```
df[(df['CS'].isnull())].shape[0]
```

[148]: 7255

[149]:
```
df[(df['yearID'] < 1951)].shape[0]
```

[149]: 17435

[150]:
```
7255/17435
```

[150]: 0.4161170060223688

There isn't a great solution for this. If we drop all missing rows with NaN for CS, we're going to lose over 41% of the data prior to 1951. It doesn't encompass enough of the data to just fill in values like we did before, we can't drop rows, and we don't want to drop the column since it isn't missing any data after 1950. One idea, and this may be controversial, is to find the average ratio between SB (stolen bases) and CS and fill in with values based on that ratio.

First, we'll get all data without missing CS values

[151]:
```
df_temp = df[(df['CS'].notnull())]
# df_temp
```

[152]:
```
total_sb = df_temp['SB'].sum()
total_sb
```

[152]: 182622

[153]:
```
total_cs = df_temp['CS'].sum()
total_cs
```

[153]: 94186.0

[154]:
```
total_sb/total_cs
```

[154]:
```
1.9389505871360924
```

So on average, players are almost twice as likely to steal a base as they are to get caught. This is easy math that we're going to round to make it even easier. It's probably not the best method of solving this issue but at least we still have over 60 years of clean data!

[155]:
```
df[(df['CS'].isnull())]
```

[155]:
```
        retroID  yearID  stint teamID lgID    G   AB   R   H  2B  ...  RBI  \
14448  aberw101    1946      1    NY1   NL   15    8   0   0   0  ...    0
16285  aberc101    1949      1    CHN   NL    4    7   0   0   0  ...    0
15712  aberc101    1948      1    CHN   NL   12   32   1   6   1  ...    6
15131  aberc101    1947      1    CHN   NL   47  140  24  39   6  ...   20
16286  abrac101    1949      1    BRO   NL    8   24   6   2   1  ...    0
...         ...     ...    ...    ...  ...  ...  ...  ..  ..  ..  ...  ...
532    zitzb101    1919      1    PIT   NL   11   26   5   5   1  ...    2
4782   zitzb101    1927      1    CIN   NL   88  232  47  66  10  ...   24
5312   zitzb101    1928      1    CIN   NL  101  266  53  79   9  ...   33
533    zitzb101    1919      2    CIN   NL    2    1   0   0   0  ...    0
5842   zitzb101    1929      1    CIN   NL   47   84  18  19   3  ...    6

       SB  CS  BB  SO  IBB  HBP  SH   SF  GIDP
14448   0 NaN   0   4  0.0    0   0  0.0   0.0
16285   0 NaN   0   2  0.0    0   0  0.0   1.0
15712   0 NaN   5  10  0.0    0   0  0.0   0.0
15131   0 NaN  20  32  0.0    0   0  0.0   5.0
16286   1 NaN   7   6  0.0    0   0  0.0   1.0
...    ..  ..  ..  ..  ...  ...  ..  ...   ...
532     2 NaN   0   6  0.0    0   1  0.0   NaN
4782    9 NaN  20  18  0.0    4  17  0.0   NaN
5312   13 NaN  13  22  0.0    3  14  0.0   NaN
533     0 NaN   0   0  0.0    0   0  0.0   NaN
5842    4 NaN   9  10  0.0    1   2  0.0   NaN

[7255 rows x 22 columns]
```

[156]:
```
df[(df['CS']).isnull()].apply(lambda x: x['SB'] / 2, axis=1)
```

[156]:
```
14448    0.0
16285    0.0
15712    0.0
15131    0.0
16286    0.5
        ...
532      1.0
```

```
4782     4.5
5312     6.5
533      0.0
5842     2.0
Length: 7255, dtype: float64
```

[157]:
```
df[(df['CS']).isnull()].apply(lambda x: x['SB'] / 2, axis=1).value_counts()
```

[157]:
```
0.0     4494
0.5      794
1.0      438
1.5      303
2.0      257
2.5      165
3.0      139
3.5      131
4.0       91
4.5       81
5.0       51
5.5       50
6.5       38
6.0       36
7.5       28
7.0       22
8.0       21
9.0       19
8.5       16
9.5       11
11.5       8
10.0       8
10.5       8
11.0       7
13.0       6
14.0       5
12.0       5
14.5       3
18.5       3
12.5       2
17.5       2
13.5       2
16.5       2
16.0       2
20.0       1
15.0       1
15.5       1
24.0       1
18.0       1
```

```
17.0       1
21.5       1
dtype: int64
```

I don't love the max of 24, but overall these values look good and we definitely don't have many of the higher values. So we're going to apply this to our missing CS data

First I'm going to test it out on a copy

[158]:
```
df_temp = df[(df['CS']).isnull()]
```

[159]:
```
df_temp['CS'] = df_temp.apply(lambda x: x['SB'] / 2, axis=1)
```

[160]:
```
df_temp[(df_temp['retroID'] == 'zitzb101')]
```

[160]:
```
       retroID  yearID  stint teamID lgID    G   AB   R   H  2B  ...  RBI  SB  \
4241  zitzb101    1926      1    CIN   NL   53   94  21  23   2  ...    3   3
532   zitzb101    1919      1    PIT   NL   11   26   5   5   1  ...    2   2
4782  zitzb101    1927      1    CIN   NL   88  232  47  66  10  ...   24   9
5312  zitzb101    1928      1    CIN   NL  101  266  53  79   9  ...   33  13
533   zitzb101    1919      2    CIN   NL    2    1   0   0   0  ...    0   0
5842  zitzb101    1929      1    CIN   NL   47   84  18  19   3  ...    6   4

       CS  BB  SO  IBB  HBP  SH   SF  GIDP
4241  1.5   6   7  0.0    2   3  0.0   NaN
532   1.0   0   6  0.0    0   1  0.0   NaN
4782  4.5  20  18  0.0    4  17  0.0   NaN
5312  6.5  13  22  0.0    3  14  0.0   NaN
533   0.0   0   0  0.0    0   0  0.0   NaN
5842  2.0   9  10  0.0    1   2  0.0   NaN

 [6 rows x 22 columns]
```

We know from before that this guy had NaNs for his CS and now it's all filled in, so our plan worked. Let's do it for the actual data

I don't know how to reassign values to a subset of a DataFrame based on a predicate (or if it's possible), so we'll get a little hacky and apply a function with a conditional. Here's what I tried originally:

df[(df['CS']).isnull()]['CS'] = df.apply(lambda x: x['SB'] / 2, axis=1)

[161]:
```
def fill_cs(data):
    if math.isnan(data['CS']):
        return data['SB'] / 2
    else:
        return data['CS']
```

[162]:
```
df['CS'] = df.apply(lambda x: fill_cs(x), axis=1)
```

[163]:
```
df[(df['retroID'] == 'zitzb101')]
```

[163]:
```
       retroID  yearID  stint teamID lgID    G   AB   R   H  2B  ...  RBI  SB  \
4241  zitzb101    1926      1    CIN   NL   53   94  21  23   2  ...    3   3
532   zitzb101    1919      1    PIT   NL   11   26   5   5   1  ...    2   2
3715  zitzb101    1925      1    CIN   NL  104  301  53  76  13  ...   21  11
4782  zitzb101    1927      1    CIN   NL   88  232  47  66  10  ...   24   9
5312  zitzb101    1928      1    CIN   NL  101  266  53  79   9  ...   33  13
533   zitzb101    1919      2    CIN   NL    2    1   0   0   0  ...    0   0
5842  zitzb101    1929      1    CIN   NL   47   84  18  19   3  ...    6   4

        CS  BB  SO  IBB  HBP  SH   SF  GIDP
4241   1.5   6   7  0.0    2   3  0.0   NaN
532    1.0   0   6  0.0    0   1  0.0   NaN
3715  11.0  35  22  0.0    6   2  0.0   NaN
4782   4.5  20  18  0.0    4  17  0.0   NaN
5312   6.5  13  22  0.0    3  14  0.0   NaN
533    0.0   0   0  0.0    0   0  0.0   NaN
5842   2.0   9  10  0.0    1   2  0.0   NaN

[7 rows x 22 columns]
```

[164]:
```
100 * df.isnull().sum() / len(df)
```

[164]:
```
retroID    0.000000
yearID     0.000000
stint      0.000000
teamID     0.000000
lgID       0.000000
G          0.000000
AB         0.000000
R          0.000000
H          0.000000
2B         0.000000
3B         0.000000
HR         0.000000
RBI        0.000000
SB         0.000000
CS         0.000000
BB         0.000000
SO         0.000000
IBB        0.000000
HBP        0.000000
SH         0.000000
SF         0.000000
GIDP       9.839985
dtype: float64
```

Handling missing GIDP data

[165]:
```
df[(df['GIDP'].isnull())]
```

[165]:
```
       retroID  yearID  stint teamID lgID    G   AB   R   H  2B  ...  RBI  \
2082   abrag101    1923      1    CIN   NL    3    1   0   1   0  ...    0
534    acosj101    1920      1    WS1   AL   17   25   2   6   1  ...    1
1569   acosj101    1922      1    CHA   AL    5    5   0   1   0  ...    0
1049   acosj101    1921      1    WS1   AL   33   30   2   2   0  ...    0
6374   adaij102    1931      1    CHN   NL   18   76   9  21   3  ...    3
...         ...     ...    ...    ...  ...  ...  ...  ..  ..  ..  ...  ...
5312   zitzb101    1928      1    CIN   NL  101  266  53  79   9  ...   33
533    zitzb101    1919      2    CIN   NL    2    1   0   0   0  ...    0
5842   zitzb101    1929      1    CIN   NL   47   84  18  19   3  ...    6
9445   zubeb101    1936      1    CLE   AL    2    5   1   1   0  ...    0
10501  zubeb101    1938      1    CLE   AL   15    7   0   0   0  ...    0

       SB   CS  BB  SO  IBB  HBP  SH   SF  GIDP
2082    0  0.0   0   0  0.0    0   0  0.0   NaN
534     0  0.0   4   7  0.0    0   2  0.0   NaN
1569    0  0.0   1   1  0.0    0   0  0.0   NaN
1049    1  0.0   6  14  0.0    0   1  0.0   NaN
6374    1  0.5   1   8  0.0    0   2  0.0   NaN
...    ..  ...  ..  ..  ...  ...  ..  ...   ...
5312   13  6.5  13  22  0.0    3  14  0.0   NaN
533     0  0.0   0   0  0.0    0   0  0.0   NaN
5842    4  2.0   9  10  0.0    1   2  0.0   NaN
9445    0  0.0   0   1  0.0    0   0  0.0   NaN
10501   0  0.0   0   1  0.0    0   1  0.0   NaN

[8683 rows x 22 columns]
```

[166]:
```
df[(df['GIDP'].isnull())]['yearID'].max()
```

[166]: 1938

[167]:
```
df[(df['yearID'] < 1939)].shape[0]
```

[167]: 10502

[168]:
```
df[(df['GIDP'].isnull())].shape[0]
```

[168]: 8683

[169]:
```
8683/10502
```

[169]: 0.8267948962102457

Over 82% of records before 1939 are missing GIDP, but it doesn't extend beyond that. I think we can once again just fill the values in with 0

[170]:
```
df['GIDP'].fillna(value=0, inplace=True)
```

[171]:
```
100 * df.isnull().sum() / len(df)
```

[171]:
```
retroID    0.0
yearID     0.0
stint      0.0
teamID     0.0
lgID       0.0
G          0.0
AB         0.0
R          0.0
H          0.0
2B         0.0
3B         0.0
HR         0.0
RBI        0.0
SB         0.0
CS         0.0
BB         0.0
SO         0.0
IBB        0.0
HBP        0.0
SH         0.0
SF         0.0
GIDP       0.0
dtype: float64
```

We've handled all missing data in the batting database

Data Integration

Now we need to eliminate an columns that we don't want (if any) and convert the ones we keep to numerical values.

[172]:
```
df.head()
```

[172]:
```
         retroID  yearID  stint teamID lgID   G  AB  R  H  2B  ...  RBI  SB  \
79400  aardd001    2013      1    NYN   NL  43   0  0  0   0  ...    0   0
82244  aardd001    2015      1    ATL   NL  33   1  0  0   0  ...    0   0
69712  aardd001    2006      1    CHN   NL  45   2  0  0   0  ...    0   0
73859  aardd001    2009      1    SEA   AL  73   0  0  0   0  ...    0   0
71089  aardd001    2007      1    CHA   AL  25   0  0  0   0  ...    0   0

        CS  BB  SO  IBB  HBP  SH   SF  GIDP
79400  0.0   0   0  0.0    0   0  0.0   0.0
```

```
82244  0.0   0   1  0.0    0   0  0.0   0.0
69712  0.0   0   0  0.0    0   1  0.0   0.0
73859  0.0   0   0  0.0    0   0  0.0   0.0
71089  0.0   0   0  0.0    0   0  0.0   0.0

[5 rows x 22 columns]
```

[173]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 88242 entries, 79400 to 86706
Data columns (total 22 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   retroID  88242 non-null  object
 1   yearID   88242 non-null  int64
 2   stint    88242 non-null  int64
 3   teamID   88242 non-null  object
 4   lgID     88242 non-null  object
 5   G        88242 non-null  int64
 6   AB       88242 non-null  int64
 7   R        88242 non-null  int64
 8   H        88242 non-null  int64
 9   2B       88242 non-null  int64
 10  3B       88242 non-null  int64
 11  HR       88242 non-null  int64
 12  RBI      88242 non-null  int64
 13  SB       88242 non-null  int64
 14  CS       88242 non-null  float64
 15  BB       88242 non-null  int64
 16  SO       88242 non-null  int64
 17  IBB      88242 non-null  float64
 18  HBP      88242 non-null  int64
 19  SH       88242 non-null  int64
 20  SF       88242 non-null  float64
 21  GIDP     88242 non-null  float64
dtypes: float64(4), int64(15), object(3)
memory usage: 15.5+ MB
```

We will handle the metadata columns later and only worry about numerical columns for now

[174]:
```
df['lgID'].value_counts()
```

[174]:
```
NL    44129
AL    44113
Name: lgID, dtype: int64
```

[175]:
```
pd.get_dummies(df['lgID'], drop_first=True)
```

[175]:
```
       NL
79400   1
82244   1
69712   1
73859   0
71089   0
...    ..
20499   0
18050   0
83729   0
85212   0
86706   0

[88242 rows x 1 columns]
```

This one will be easy - there are only two leagues in the dataset, so we can just transform that into a single boolean column. Of course that column will be NL, the superior league.

[176]:
```
df['NL'] = pd.get_dummies(df['lgID'], drop_first=True)
df.drop(columns=['lgID'], inplace=True)
```

[177]:
```
df
```

[177]:
```
        retroID  yearID  stint teamID   G  AB  R  H  2B  3B  ...  SB   CS  BB  \
79400  aardd001    2013      1    NYN  43   0  0  0   0   0  ...   0  0.0   0
82244  aardd001    2015      1    ATL  33   1  0  0   0   0  ...   0  0.0   0
69712  aardd001    2006      1    CHN  45   2  0  0   0   0  ...   0  0.0   0
73859  aardd001    2009      1    SEA  73   0  0  0   0   0  ...   0  0.0   0
71089  aardd001    2007      1    CHA  25   0  0  0   0   0  ...   0  0.0   0
...         ...     ...    ...    ...  ..  .. .. ..  ..  ..  ...  ..  ...  ..
20499  zuveg101    1955      2    BAL  28  23  1  5   1   0  ...   0  0.0   1
18050  zuveg101    1951      1    CLE  16   0  0  0   0   0  ...   0  0.0   0
83729  zycht001    2015      1    SEA  13   0  0  0   0   0  ...   0  0.0   0
85212  zycht001    2016      1    SEA  12   0  0  0   0   0  ...   0  0.0   0
86706  zycht001    2017      1    SEA  45   0  0  0   0   0  ...   0  0.0   0

       SO  IBB  HBP  SH   SF  GIDP  NL
79400   0  0.0    0   0  0.0   0.0   1
82244   1  0.0    0   0  0.0   0.0   1
69712   0  0.0    0   1  0.0   0.0   1
73859   0  0.0    0   0  0.0   0.0   0
71089   0  0.0    0   0  0.0   0.0   0
...    ..  ...  ...  ..  ...   ...  ..
20499   5  0.0    0   1  0.0   1.0   0
18050   0  0.0    0   0  0.0   0.0   0
83729   0  0.0    0   0  0.0   0.0   0
85212   0  0.0    0   0  0.0   0.0   0
```

```
86706    0  0.0     0    0  0.0    0.0    0

[88242 rows x 22 columns]
```

Now we need to figure out how to handle the teamID column.

[178]:
```
df['teamID'].nunique()
```

[178]:
```
45
```

Since we have more than 30 team IDs, to keep things consistent I'm just going to map them to franchise ID.

[179]:
```
# This will be exported to a separate module
teams = pd.read_csv('../data/lahman/mlb_data/Teams.csv')
teams = teams[['teamID', 'franchID']]
team_dict = teams.set_index('teamID').to_dict()['franchID']

def get_team(team):
    return team_dict[team] if id_dict is not None else team
```

[180]:
```
df['teamID'] = df['teamID'].apply(get_team)
```

[181]:
```
df['teamID'].nunique()
```

[181]:
```
30
```

We're now all set with team IDs as strings

[182]:
```
df.head()
```

[182]:
```
        retroID  yearID  stint teamID   G  AB  R  H  2B  3B  ...  SB   CS  BB  \
79400  aardd001    2013      1    NYM  43   0  0  0   0   0  ...   0  0.0   0
82244  aardd001    2015      1    ATL  33   1  0  0   0   0  ...   0  0.0   0
69712  aardd001    2006      1    CHC  45   2  0  0   0   0  ...   0  0.0   0
73859  aardd001    2009      1    SEA  73   0  0  0   0   0  ...   0  0.0   0
71089  aardd001    2007      1    CHW  25   0  0  0   0   0  ...   0  0.0   0

       SO  IBB  HBP  SH   SF  GIDP  NL
79400   0  0.0    0   0  0.0   0.0   1
82244   1  0.0    0   0  0.0   0.0   1
69712   0  0.0    0   1  0.0   0.0   1
73859   0  0.0    0   0  0.0   0.0   0
71089   0  0.0    0   0  0.0   0.0   0

[5 rows x 22 columns]
```

[183]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 88242 entries, 79400 to 86706
Data columns (total 22 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   retroID  88242 non-null  object
 1   yearID   88242 non-null  int64
 2   stint    88242 non-null  int64
 3   teamID   88242 non-null  object
 4   G        88242 non-null  int64
 5   AB       88242 non-null  int64
 6   R        88242 non-null  int64
 7   H        88242 non-null  int64
 8   2B       88242 non-null  int64
 9   3B       88242 non-null  int64
 10  HR       88242 non-null  int64
 11  RBI      88242 non-null  int64
 12  SB       88242 non-null  int64
 13  CS       88242 non-null  float64
 14  BB       88242 non-null  int64
 15  SO       88242 non-null  int64
 16  IBB      88242 non-null  float64
 17  HBP      88242 non-null  int64
 18  SH       88242 non-null  int64
 19  SF       88242 non-null  float64
 20  GIDP     88242 non-null  float64
 21  NL       88242 non-null  uint8
dtypes: float64(4), int64(15), object(2), uint8(1)
memory usage: 14.9+ MB
```

[184]:
```python
df = df.sort_index()
```

[185]:
```python
df.head()
```

[185]:
```
    retroID  yearID  stint teamID    G   AB   R   H  2B  3B  ...  SB   CS  BB  \
0  adamb104    1919      1    PIT   34   92   2  17   2   1  ...   0  0.0   6
1  adamb106    1919      1    PHI   78  232  14  54   7   2  ...   4  2.0   6
2  adamw101    1919      1    OAK    1    2   0   0   0   0  ...   0  0.0   0
3  agnes101    1919      1    MIN   42   98   6  23   7   0  ...   1  0.5  10
4  ainse101    1919      1    DET  114  364  42  99  17  12  ...   9  4.5  45

   SO  IBB  HBP  SH   SF  GIDP  NL
0  13  0.0    0   3  0.0   0.0   1
1  27  0.0    0   3  0.0   0.0   1
2   1  0.0    0   0  0.0   0.0   0
3   8  0.0    1   9  0.0   0.0   0
4  30  0.0    1  12  0.0   0.0   0
```

```
[5 rows x 22 columns]
```

We need some sort of dictionary to associate a player's retroID with an index. The following steps care of that. This is so we can later associate the correct retroID with our data.

[186]:
```
df.reset_index(inplace=True)
```

[187]:
```
metadata_column_labels = ['index', 'yearID', 'stint', 'teamID']
```

[188]:
```
metadata = df[metadata_column_labels].set_index(df['retroID']).reset_index()
```

[189]:
```
metadata.head()
```

[189]:

	retroID	index	yearID	stint	teamID
0	adamb104	0	1919	1	PIT
1	adamb106	1	1919	1	PHI
2	adamw101	2	1919	1	OAK
3	agnes101	3	1919	1	MIN
4	ainse101	4	1919	1	DET

The metadata table will eventually be expanded with information from Players.csv to hold all relevant player information that isn't used for the neural network.

[190]:
```
indexer = metadata.drop_duplicates('retroID').set_index('index').T.
↪to_dict('retroID')[0]
```

[191]:
```
df = df.drop(columns=metadata_column_labels)
```

[192]:
```
df.head()
```

[192]:

	retroID	G	AB	R	H	2B	3B	HR	RBI	SB	CS	BB	SO	IBB	HBP	SH	\
0	adamb104	34	92	2	17	2	1	0	4	0	0.0	6	13	0.0	0	3	
1	adamb106	78	232	14	54	7	2	1	17	4	2.0	6	27	0.0	0	3	
2	adamw101	1	2	0	0	0	0	0	0	0	0.0	0	1	0.0	0	0	
3	agnes101	42	98	6	23	7	0	0	10	1	0.5	10	8	0.0	1	9	
4	ainse101	114	364	42	99	17	12	3	32	9	4.5	45	30	0.0	1	12	

	SF	GIDP	NL
0	0.0	0.0	1
1	0.0	0.0	1
2	0.0	0.0	0
3	0.0	0.0	0
4	0.0	0.0	0

Now that the metadata is gone, we just have the ID and the numerical batting information. We can group by the ID and just sum every other column to get player career totals.

[193]:
```
df = df.groupby('retroID').sum().reset_index()
```

[194]:
```
df
```

[194]:
```
      retroID     G     AB     R     H   2B  3B   HR   RBI   SB    CS    BB  \
0     aardd001   331      4     0     0    0   0    0     0    0   0.0     0
1     aaroh101  3298  12364  2174  3771  624  98  755  2297  240  73.0  1402
2     aarot101   437    944   102   216   42   6   13    94    9   8.0    86
3     aased001   448      5     0     0    0   0    0     0    0   0.0     0
4     abada001    15     21     1     2    0   0    0     0    0   1.0     4
...        ...   ...    ...   ...   ...  ...  ..  ...   ...  ...   ...   ...
15187 zupcb001   319    795    99   199   47   4    7    80    7   5.0    57
15188 zupof101    16     18     3     3    1   0    0     0    0   0.0     2
15189 zuveg101   266    142     5    21    2   1    0     7    0   1.0     9
15190 zuvep001   209    491    41   109   17   2    2    20    2   0.0    34
15191 zycht001    70      0     0     0    0   0    0     0    0   0.0     0

         SO    IBB  HBP  SH     SF   GIDP  NL
0         2    0.0    0   1    0.0    0.0   4
1      1383  293.0   32  21  121.0  328.0  21
2       145    3.0    0   9    6.0   36.0   7
3         3    0.0    0   0    0.0    0.0   2
4         5    0.0    0   0    0.0    1.0   1
...     ...    ...  ...  ..    ...    ...  ..
15187   137    3.0    6  20    8.0   15.0   0
15188     6    0.0    0   0    0.0    0.0   0
15189    39    0.0    0  16    0.0    3.0   1
15190    50    1.0    2  18    0.0    8.0   4
15191     0    0.0    0   0    0.0    0.0   0

[15192 rows x 19 columns]
```

Since we summed everything, we just need to change the NL column back. We can divide each value by itself to get either 1 or 0 like we had before.

[195]:
```
df['NL'] = np.where(df['NL'] > 0, 1, 0)
```

[196]:
```
tensor = df.drop(columns=['retroID'])
```

[197]:
```
tensor
```

[197]:
```
         G     AB     R     H   2B  3B   HR   RBI   SB    CS    BB    SO  \
0      331      4     0     0    0   0    0     0    0   0.0     0     2
1     3298  12364  2174  3771  624  98  755  2297  240  73.0  1402  1383
2      437    944   102   216   42   6   13    94    9   8.0    86   145
3      448      5     0     0    0   0    0     0    0   0.0     0     3
4       15     21     1     2    0   0    0     0    0   1.0     4     5
```

```
...      ...    ...    ...    ...   ...  ..  ...    ...   ...   ...    ...    ...
15187    319    795     99    199    47   4    7     80     7   5.0     57    137
15188     16     18      3      3     1   0    0      0     0   0.0      2      6
15189    266    142      5     21     2   1    0      7     0   1.0      9     39
15190    209    491     41    109    17   2    2     20     2   0.0     34     50
15191     70      0      0      0     0   0    0      0     0   0.0      0      0

          IBB  HBP  SH     SF   GIDP  NL
0         0.0    0   1    0.0    0.0   1
1       293.0   32  21  121.0  328.0   1
2         3.0    0   9    6.0   36.0   1
3         0.0    0   0    0.0    0.0   1
4         0.0    0   0    0.0    1.0   1
...       ...  ...  ..    ...    ...  ..
15187     3.0    6  20    8.0   15.0   0
15188     0.0    0   0    0.0    0.0   0
15189     0.0    0  16    0.0    3.0   1
15190     1.0    2  18    0.0    8.0   1
15191     0.0    0   0    0.0    0.0   0

[15192 rows x 18 columns]
```

[198]:
```
tensor.to_csv('../output/tensor.csv')
metadata.to_csv('../output/metadata.csv')
```

We now have a tensor with only relevant information, an indexing dictionary to get the player for each row, and a (soon to be expanded) metadata table to get more information on each player.

fielding_pre

March 9, 2020

[47]:
```
import numpy as np
import pandas as pd
pd.options.mode.chained_assignment = None  # default='warn'
```

[48]:
```
df = pd.read_csv('../data/lahman/mlb_data/Fielding.csv').sort_values('playerID')
```

[49]:
```
# This will be exported to a separate module
ids = pd.read_csv('../data/lahman/mlb_data/People.csv')
ids = ids[['playerID', 'retroID']]
id_dict = ids.set_index('playerID').to_dict()['retroID']

def get_retroid(id):
    return id_dict[id] if id_dict is not None else id
```

[50]:
```
df['playerID'] = df['playerID'].apply(get_retroid)
df.rename(columns={'playerID': 'retroID'}, inplace=True)
```

Exploration

[51]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 112837 entries, 85308 to 106797
Data columns (total 18 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   retroID  112837 non-null  object
 1   yearID   112837 non-null  int64
 2   stint    112837 non-null  int64
 3   teamID   112837 non-null  object
 4   lgID     112837 non-null  object
 5   POS      112837 non-null  object
 6   G        112837 non-null  int64
 7   GS       89431 non-null   float64
 8   InnOuts  89431 non-null   float64
 9   PO       112837 non-null  int64
 10  A        112837 non-null  int64
 11  E        112836 non-null  float64
```

```
 12  DP        112837 non-null  int64
 13  PB        8538 non-null    float64
 14  WP        1169 non-null    float64
 15  SB        6389 non-null    float64
 16  CS        6389 non-null    float64
 17  ZR        1169 non-null    float64
dtypes: float64(8), int64(6), object(4)
memory usage: 16.4+ MB
```

[52]:
```
df.shape
```

[52]:
```
(112837, 18)
```

[53]:
```
df.columns
```

[53]:
```
Index(['retroID', 'yearID', 'stint', 'teamID', 'lgID', 'POS', 'G', 'GS',
       'InnOuts', 'PO', 'A', 'E', 'DP', 'PB', 'WP', 'SB', 'CS', 'ZR'],
      dtype='object')
```

We want to get rid of columns which already exist in the Batting DataFrame (with which we will be merging this)

[54]:
```
columns_to_drop = ['stint', 'teamID', 'lgID', 'G']
```

[55]:
```
df.drop(columns=columns_to_drop, inplace=True)
```

[56]:
```
df.head()
```

[56]:

	retroID	yearID	POS	GS	InnOuts	PO	A	E	DP	PB	WP	SB	CS	ZR
85308	aardd001	2004	P	0.0	32.0	0	0	0.0	0	NaN	NaN	NaN	NaN	NaN
101187	aardd001	2013	P	0.0	119.0	1	5	0.0	0	NaN	NaN	NaN	NaN	NaN
99344	aardd001	2012	P	0.0	3.0	0	0	0.0	0	NaN	NaN	NaN	NaN	NaN
95793	aardd001	2010	P	0.0	149.0	2	3	1.0	0	NaN	NaN	NaN	NaN	NaN
104866	aardd001	2015	P	0.0	92.0	0	1	1.0	0	NaN	NaN	NaN	NaN	NaN

Cleaning and Preprocessing

We see a lot of NaNs in the last 5 columns. According to the Lahman readme, these are:

- PB - Passed Balls (by catchers)
- WP - Wild Pitches (by catchers)
- SB - Opponent Stolen Bases (by catchers)
- CS - Opponents Caught Stealing (by catchers)
- ZR - Zone Rating

It looks like the data demands that we treat catchers separately from other position players. This intuitively makes sense from what we know about baseball, and it saves us from getting rid of a lot of data. First, though, let's look at how much of that data is missing if we JUST look at catchers.

[57]:
```
df_catchers = df[df['POS'] == 'C']
```

[58]:
```
# Get missing data in the catchers category as a percentage
100 * df_catchers.isnull().sum() / len(df)
```

[58]:
```
retroID     0.000000
yearID      0.000000
POS         0.000000
GS          1.901858
InnOuts     1.901858
PO          0.000000
A           0.000000
E           0.000000
DP          0.000000
PB          0.000000
WP          6.530659
SB          1.904517
CS          1.904517
ZR          6.530659
dtype: float64
```

Most of the percentages are negligable, but we can take a look at WP and ZR and see if the missing data is from early years.

[59]:
```
early_catchers = df_catchers[df_catchers['yearID'] < 1955]
```

[60]:
```
100 * early_catchers.isnull().sum() / len(df)
```

[60]:
```
retroID     0.000000
yearID      0.000000
POS         0.000000
GS          1.901858
InnOuts     1.901858
PO          0.000000
A           0.000000
E           0.000000
DP          0.000000
PB          0.000000
WP          1.901858
SB          1.901858
CS          1.901858
ZR          1.901858
dtype: float64
```

Definitely not the case. Let's try to narrow down where the issue is.

[61]:
```
post1985_catchers = df_catchers[df_catchers['yearID'] > 1985]
```

[62]:
```
100 * post1985_catchers.isnull().sum() / len(df)
```

[62]:
```
retroID     0.000000
yearID      0.000000
POS         0.000000
GS          0.000000
InnOuts     0.000000
PO          0.000000
A           0.000000
E           0.000000
DP          0.000000
PB          0.000000
WP          3.265773
SB          0.000000
CS          0.000000
ZR          3.265773
dtype: float64
```

[63]:
```
df_1955_to_1986_catchers = df_catchers[(df_catchers['yearID'] >= 1955) &␣
 ↪(df_catchers['yearID'] <= 1985)]
```

[64]:
```
100 * df_1955_to_1986_catchers.isnull().sum() / len(df)
```

[64]:
```
retroID     0.000000
yearID      0.000000
POS         0.000000
GS          0.000000
InnOuts     0.000000
PO          0.000000
A           0.000000
E           0.000000
DP          0.000000
PB          0.000000
WP          1.363028
SB          0.002659
CS          0.002659
ZR          1.363028
dtype: float64
```

[65]:
```
pre_1930_catchers = df_catchers[df_catchers['yearID'] < 1930]
```

[66]:
```
100 * pre_1930_catchers.isnull().sum() / len(df)
```

[66]:
```
retroID     0.000000
yearID      0.000000
POS         0.000000
GS          0.591118
```

```
InnOuts    0.591118
PO         0.000000
A          0.000000
E          0.000000
DP         0.000000
PB         0.000000
WP         0.591118
SB         0.591118
CS         0.591118
ZR         0.591118
dtype: float64
```

We see that the issue is mainly in the very early years, and we are fine with dropping that information by just filling it in as we did in the Batters table.

So with that, we are fine with filling all NA values with 0.

[67]:
```
df_catchers['GS'].fillna(value=0, inplace=True)
df_catchers['InnOuts'].fillna(value=0, inplace=True)
df_catchers['WP'].fillna(value=0, inplace=True)
df_catchers['SB'].fillna(value=0, inplace=True)
df_catchers['CS'].fillna(value=0, inplace=True)
df_catchers['ZR'].fillna(value=0, inplace=True)
```

[68]:
```
df['GS'].fillna(value=0, inplace=True)
df['InnOuts'].fillna(value=0, inplace=True)
#We can just drop the catcher-related columns from the original dataframe, as we␣
↪will also drop all catcher rows
catcher_columns = ['PB', 'WP', 'SB', 'CS', 'ZR']
df.drop(columns=catcher_columns, inplace=True)
```

Now drop all catcher rows so we have two separate dataframes, and get rid of the yearID column which we're done with and will be useless after aggregation.

[69]:
```
df = df[df['POS'] != 'C']
```

[70]:
```
df.drop(columns=['yearID'], inplace=True)
df_catchers.drop(columns=['yearID'], inplace=True)
```

[71]:
```
df.shape
```

[71]: (104299, 8)

[72]:
```
df_catchers.shape
```

[72]: (8538, 13)

[73]:
```
100 * df.isnull().sum() / len(df)
```

[73]:
```
retroID    0.000000
POS        0.000000
GS         0.000000
InnOuts    0.000000
PO         0.000000
A          0.000000
E          0.000959
DP         0.000000
dtype: float64
```

Now we just see a little bit of information missing from Errors, so we can fill that with 0s no problem.

[74]:
```
df['E'].fillna(value=0, inplace=True)
```

[75]:
```
100 * df.isnull().sum() / len(df)
```

[75]:
```
retroID    0.0
POS        0.0
GS         0.0
InnOuts    0.0
PO         0.0
A          0.0
E          0.0
DP         0.0
dtype: float64
```

[76]:
```
100 * df_catchers.isnull().sum() / len(df)
```

[76]:
```
retroID    0.0
POS        0.0
GS         0.0
InnOuts    0.0
PO         0.0
A          0.0
E          0.0
DP         0.0
PB         0.0
WP         0.0
SB         0.0
CS         0.0
ZR         0.0
dtype: float64
```

Aggregation

Now we just need to aggregate all stats to get total career numbers for each player.

[77]:
```
df = df.groupby('retroID').sum().reset_index()
```

[78]:
```
df_catchers = df_catchers.groupby('retroID').sum().reset_index()
```

[79]:
```
df
```

[79]:
```
        retroID      GS   InnOuts    PO    A      E   DP
0      aardd001     0.0    1011.0    11   29    3.0    2
1      aaroh101  2977.0   78414.0  7436  429  144.0  218
2      aarot101   206.0    6472.0  1317  113   22.0  124
3      aased001    91.0    3328.0    67  135   13.0   10
4      abada001     4.0     138.0    37    1    1.0    3
...         ...     ...       ...   ...  ...    ...  ...
14222  zumaj001     0.0     629.0     7   14    2.0    1
14223  zupcb001   198.0    5842.0   483   22   12.0    5
14224  zuveg101    31.0    1847.0    45  145    7.0   10
14225  zuvep001   136.0    3844.0   267  415   23.0   84
14226  zycht001     1.0     218.0     1    6    1.0    0

[14227 rows x 7 columns]
```

[80]:
```
df_catchers
```

[80]:
```
       retroID     GS  InnOuts    PO    A     E  DP    PB    WP     SB    CS  \
0     adamb105    1.0     27.0     6    0   0.0   0   0.0   0.0    1.0   0.0
1     adamb106    0.0      0.0   249   90  12.0  15   7.0   0.0    0.0   0.0
2     adamd101    3.0     78.0     9    2   0.0   0   1.0   0.0    0.0   0.0
3     adled101   65.0   1840.0   453   26   4.0   2   8.0  19.0   37.0  16.0
4     afent001   20.0    613.0   123    5   1.0   3   6.0   0.0   17.0   3.0
...        ...    ...      ...   ...  ...   ...  ..   ...   ...    ...   ...
1524  zimmd101   27.0    744.0   150   18   6.0   1   5.0  12.0   10.0  10.0
1525  zimmj101  298.0   8560.0  2131  150  21.0  26  19.0  84.0  110.0  80.0
1526  zinta001    0.0      3.0     2    0   0.0   0   0.0   0.0    0.0   0.0
1527  zunim001  535.0  14489.0  4356  264  21.0  22  39.0   0.0  248.0  98.0
1528  zupof101    1.0    114.0    31    1   2.0   0   1.0   1.0    2.0   1.0

       ZR
0     0.0
1     0.0
2     0.0
3     0.0
4     0.0
...   ...
1524  3.0
1525  4.0
1526  0.0
1527  0.0
```

```
1528  0.0

[1529 rows x 12 columns]
```

[ ]:

```

```

add_advanced_batting

April 29, 2020

[76]:
```python
import pandas as pd
import matplotlib.pyplot as plt
```

[30]:
```python
df = pd.read_csv('../core/output/batters.csv')
df_adv = pd.read_csv('../core/output/advanced_batting.csv')
```

Adding Advanced Stats

We will use a combination of wOBA, wRC+ and WAR as our overall rating - our Y value.

[43]:
```python
df_adv.sort_values('retroID')
```

[43]:
```
         retroID   wOBA   wRC+    WAR
9203    aardd001  0.000 -100.0   -0.1
3       aaroh101  0.403  153.0  136.3
13920   aarot101  0.282   76.0   -1.7
9158    aased001  0.000 -100.0   -0.1
11841   abada001  0.184    0.0   -0.4
...          ...    ...    ...    ...
13227   zupcb001  0.293   74.0   -0.9
10487   zupof101  0.225   37.0   -0.2
11591   zuveg101  0.179    0.0   -0.3
14134   zuvep001  0.254   52.0   -2.2
6759    zycht001  0.000    NaN    0.0

[14399 rows x 4 columns]
```

[42]:
```python
df
```

[42]:
```
         retroID    weight  height  debutYear  finalYear  pos_1B  pos_2B  \
0       aardd001  0.569672    0.60       2004       2015       0       0
1       aaroh101  0.426230    0.45       1954       1976       0       0
2       aarot101  0.467213    0.60       1962       1971       1       0
3       aased001  0.467213    0.60       1977       1990       0       0
4       abada001  0.442623    0.50       2001       2006       1       0
...          ...       ...     ...        ...        ...     ...     ...
15288   zupcb001  0.590164    0.65       1991       1994       0       0
15289   zupof101  0.434426    0.40       1957       1961       0       0
```

```
15290  zuveg101  0.487705    0.65        1951        1959        0        0
15291  zuvep001  0.397541    0.45        1982        1991        0        0
15292  zycht001  0.467213    0.60        2015        2017        0        0

       pos_3B  pos_C  pos_OF  ...   SB    CS    BB    SO  IBB  HBP  SH   SF  \
0           0      0       0  ...    0   0.0     0     2    0    0   1    0
1           0      0       1  ...  240  73.0  1402  1383  293   32  21  121
2           0      0       0  ...    9   8.0    86   145    3    0   9    6
3           0      0       0  ...    0   0.0     0     3    0    0   0    0
4           0      0       0  ...    0   1.0     4     5    0    0   0    0
...       ...    ...     ...  ...  ...   ...   ...   ...  ...  ...  ..  ...
15288       0      0       1  ...    7   5.0    57   137    3    6  20    8
15289       0      1       0  ...    0   0.0     2     6    0    0   0    0
15290       0      0       0  ...    0   1.0     9    39    0    0  16    0
15291       0      0       0  ...    2   0.0    34    50    1    2  18    0
15292       0      0       0  ...    0   0.0     0     0    0    0   0    0

       GIDP  NL
0         0   1
1       328   1
2        36   1
3         0   1
4         1   1
...     ...  ..
15288    15   0
15289     0   0
15290     3   1
15291     8   1
15292     0   0

[15293 rows x 36 columns]
```

[35]:
```
df.shape
```

[35]:
```
(15293, 36)
```

[36]:
```
df_adv.shape
```

[36]:
```
(14399, 4)
```

[44]:
```
df = df.merge(df_adv, how='left')
```

[46]:
```
df['wOBA'].fillna(0, inplace=True)
df['wRC+'].fillna(0, inplace=True)
df['WAR'].fillna(0, inplace=True)
```

[47]:
```
df
```

```
[47]:         retroID    weight  height  debutYear  finalYear  pos_1B  pos_2B  \
0      aardd001  0.569672    0.60       2004       2015       0       0
1      aaroh101  0.426230    0.45       1954       1976       0       0
2      aarot101  0.467213    0.60       1962       1971       1       0
3      aased001  0.467213    0.60       1977       1990       0       0
4      abada001  0.442623    0.50       2001       2006       1       0
...         ...       ...     ...        ...        ...     ...     ...
15288  zupcb001  0.590164    0.65       1991       1994       0       0
15289  zupof101  0.434426    0.40       1957       1961       0       0
15290  zuveg101  0.487705    0.65       1951       1959       0       0
15291  zuvep001  0.397541    0.45       1982       1991       0       0
15292  zycht001  0.467213    0.60       2015       2017       0       0

       pos_3B  pos_C  pos_OF  ...    SO  IBB  HBP  SH   SF  GIDP  NL   wOBA  \
0           0      0       0  ...     2    0    0   1    0     0   1  0.000
1           0      0       1  ...  1383  293   32  21  121   328   1  0.403
2           0      0       0  ...   145    3    0   9    6    36   1  0.282
3           0      0       0  ...     3    0    0   0    0     0   1  0.000
4           0      0       0  ...     5    0    0   0    0     1   1  0.184
...       ...    ...     ...  ...   ...  ...  ...  ..  ...   ...  ..    ...
15288       0      0       1  ...   137    3    6  20    8    15   0  0.293
15289       0      1       0  ...     6    0    0   0    0     0   0  0.225
15290       0      0       0  ...    39    0    0  16    0     3   1  0.179
15291       0      0       0  ...    50    1    2  18    0     8   1  0.254
15292       0      0       0  ...     0    0    0   0    0     0   0  0.000

        wRC+    WAR
0     -100.0   -0.1
1      153.0  136.3
2       76.0   -1.7
3     -100.0   -0.1
4        0.0   -0.4
...      ...    ...
15288   74.0   -0.9
15289   37.0   -0.2
15290    0.0   -0.3
15291   52.0   -2.2
15292    0.0    0.0

[15293 rows x 39 columns]
```

For now, we're just going to take the mean of the three most accepted advanced statistics, giving them equal importance. This will lead to a model that favors offense over defense, as WAR is the only stat that takes defense into account, but that's fine.

```
[50]: df['Batting'] = df[['wOBA', 'wRC+', 'WAR']].mean(axis=1).round(3)
```

```
[51]: df['Batting']
```

```
[51]: 0       -33.367
      1        96.568
      2        24.861
      3       -33.367
      4        -0.072
                ...
      15288    24.464
      15289    12.342
      15290    -0.040
      15291    16.685
      15292     0.000
      Name: Rating, Length: 15293, dtype: float64
```

```
[81]: df['Batting'].mean()
```

```
[81]: 11.660071993722617
```

```
[82]: df['Batting'].min()
```

```
[82]: -33.5
```

```
[83]: df['Batting'].max()
```

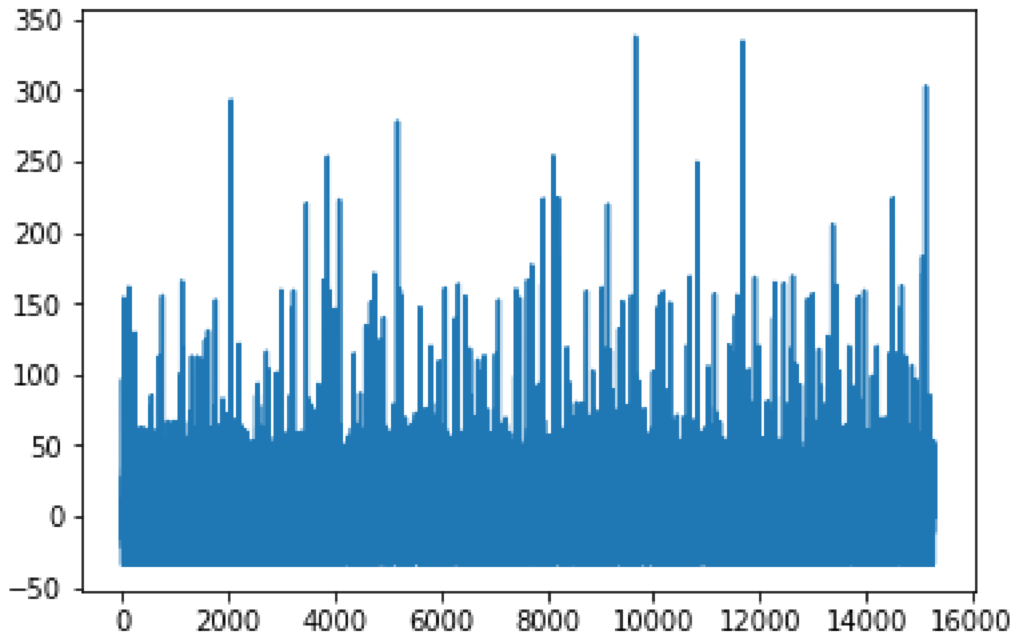
```
[83]: 337.916
```

Normalization

The Batting stat now has a very wide range which seems to trend more toward the lower end. We need to normalize the statistic so that our sigmoid output will be able to accurately predict it. For this reason, we'll use min-max normalization to get a range [0, 1].

```
[77]: plt.plot(df['Batting'])
```

```
[77]: [<matplotlib.lines.Line2D at 0x12a2a5190>]
```

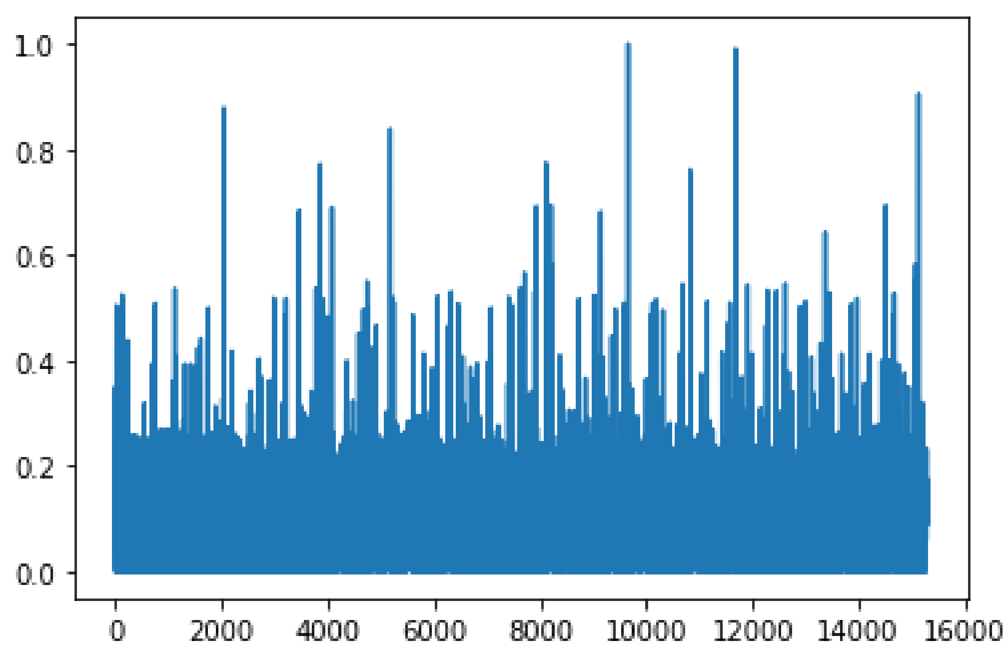



```
[84]: from sklearn.preprocessing import MinMaxScaler
```

```
[85]: scaler = MinMaxScaler()
```

```
[86]: plt.plot(scaler.fit_transform(df[['Batting']]))
```

```
[86]: [<matplotlib.lines.Line2D at 0x121311650>]
```



[88]:
```python
df['Batting'] = scaler.fit_transform(df[['Batting']])
```

[89]:
```python
df
```

[89]:
```
        retroID    weight  height  debutYear  finalYear  pos_1B  pos_2B  \
0      aardd001  0.569672    0.60       2004       2015       0       0
1      aaroh101  0.426230    0.45       1954       1976       0       0
2      aarot101  0.467213    0.60       1962       1971       1       0
3      aased001  0.467213    0.60       1977       1990       0       0
4      abada001  0.442623    0.50       2001       2006       1       0
...         ...       ...     ...        ...        ...     ...     ...
15288  zupcb001  0.590164    0.65       1991       1994       0       0
15289  zupof101  0.434426    0.40       1957       1961       0       0
15290  zuveg101  0.487705    0.65       1951       1959       0       0
15291  zuvep001  0.397541    0.45       1982       1991       0       0
15292  zycht001  0.467213    0.60       2015       2017       0       0

       pos_3B  pos_C  pos_OF  ...  IBB  HBP  SH   SF  GIDP  NL   wOBA   wRC+  \
0           0      0       0  ...    0    0   1    0     0   1  0.000 -100.0
1           0      0       1  ...  293   32  21  121   328   1  0.403  153.0
2           0      0       0  ...    3    0   9    6    36   1  0.282   76.0
3           0      0       0  ...    0    0   0    0     0   1  0.000 -100.0
4           0      0       0  ...    0    0   0    0     1   1  0.184    0.0
...       ...    ...     ...  ...  ...  ...  ..  ...   ...  ..    ...    ...
15288       0      0       1  ...    3    6  20    8    15   0  0.293   74.0
15289       0      1       0  ...    0    0   0    0     0   0  0.225   37.0
15290       0      0       0  ...    0    0  16    0     3   1  0.179    0.0
15291       0      0       0  ...    1    2  18    0     8   1  0.254   52.0
15292       0      0       0  ...    0    0   0    0     0   0  0.000    0.0

         WAR    Rating
0       -0.1  0.000358
1      136.3  0.350195
2       -1.7  0.157131
3       -0.1  0.000358
4       -0.4  0.090002
...      ...       ...
15288   -0.9  0.156062
15289   -0.2  0.123425
15290   -0.3  0.090088
15291   -2.2  0.135118
15292    0.0  0.090195

[15293 rows x 40 columns]
```

We now have the Y value that our NN should attempt to predict. We'll keep wOBA, wRC+ and WAR as columns at this point so we can decide later if they need to come out.

[ ]:

```

```

pitching_pre

April 28, 2020

[211]:
```python
import pandas as pd
import numpy as np
pd.options.mode.chained_assignment = None  # default='warn'
```

[212]:
```python
df = pd.read_csv('../data/lahman/mlb_data/Pitching.csv')
```

[213]:
```python
# This will be exported to a separate module
ids = pd.read_csv('../data/lahman/mlb_data/People.csv')
ids = ids[['playerID', 'retroID']]
id_dict = ids.set_index('playerID').to_dict()['retroID']

def get_retroid(id):
    return id_dict[id] if id_dict is not None else id
```

[214]:
```python
df['playerID'] = df['playerID'].apply(get_retroid)
df.rename(columns={'playerID': 'retroID'}, inplace=True)
```

Exploration

[215]:
```python
df.head()
```

[215]:
```
    retroID  yearID  stint teamID lgID   W   L   G  GS  CG  ...  IBB  WP  HBP  \
0  adamb104    1919      1    PIT   NL  17  10  34  29  23  ...  NaN   2    3
1  adamw101    1919      1    PHA   AL   0   0   1   0   0  ...  NaN   0    1
2  alexg102    1919      1    CHN   NL  16  11  30  27  20  ...  NaN   1    0
3  altrn101    1919      1    WS1   AL   0   0   1   0   0  ...  NaN   0    0
4  amesr101    1919      1    SLN   NL   3   5  23   7   1  ...  NaN   3    1

   BK     BFP  GF   R  SH  SF  GIDP
0   0  1017.0   5  66 NaN NaN   NaN
1   0    21.0   1   2 NaN NaN   NaN
2   0   906.0   3  51 NaN NaN   NaN
3   0     4.0   0   4 NaN NaN   NaN
4   0   314.0  10  44 NaN NaN   NaN

[5 rows x 30 columns]
```

[216]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40372 entries, 0 to 40371
Data columns (total 30 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   retroID  40372 non-null  object
 1   yearID   40372 non-null  int64
 2   stint    40372 non-null  int64
 3   teamID   40372 non-null  object
 4   lgID     40372 non-null  object
 5   W        40372 non-null  int64
 6   L        40372 non-null  int64
 7   G        40372 non-null  int64
 8   GS       40372 non-null  int64
 9   CG       40372 non-null  int64
 10  SHO      40372 non-null  int64
 11  SV       40372 non-null  int64
 12  IPouts   40372 non-null  int64
 13  H        40372 non-null  int64
 14  ER       40372 non-null  int64
 15  HR       40372 non-null  int64
 16  BB       40372 non-null  int64
 17  SO       40372 non-null  int64
 18  BAOpp    40360 non-null  float64
 19  ERA      40298 non-null  float64
 20  IBB      32121 non-null  float64
 21  WP       40372 non-null  int64
 22  HBP      40372 non-null  int64
 23  BK       40372 non-null  int64
 24  BFP      40369 non-null  float64
 25  GF       40372 non-null  int64
 26  R        40372 non-null  int64
 27  SH       27512 non-null  float64
 28  SF       27512 non-null  float64
 29  GIDP     26381 non-null  float64
dtypes: float64(7), int64(20), object(3)
memory usage: 9.2+ MB
```

[217]:
```
df.columns
```

[217]:
```
Index(['retroID', 'yearID', 'stint', 'teamID', 'lgID', 'W', 'L', 'G', 'GS',
       'CG', 'SHO', 'SV', 'IPouts', 'H', 'ER', 'HR', 'BB', 'SO', 'BAOpp',
       'ERA', 'IBB', 'WP', 'HBP', 'BK', 'BFP', 'GF', 'R', 'SH', 'SF', 'GIDP'],
      dtype='object')
```

[218]:
```
columns_to_drop = ['stint', 'teamID', 'lgID']
```

[219]:
```
df.drop(columns=columns_to_drop, inplace=True)
```

[220]:
```
df.shape
```

[220]:
```
(40372, 27)
```

Cleaning and Preprocessing

[221]:
```
100 * df.isnull().sum() / len(df)
```

[221]:
```
retroID     0.000000
yearID      0.000000
W           0.000000
L           0.000000
G           0.000000
GS          0.000000
CG          0.000000
SHO         0.000000
SV          0.000000
IPouts      0.000000
H           0.000000
ER          0.000000
HR          0.000000
BB          0.000000
SO          0.000000
BAOpp       0.029724
ERA         0.183295
IBB        20.437432
WP          0.000000
HBP         0.000000
BK          0.000000
BFP         0.007431
GF          0.000000
R           0.000000
SH         31.853760
SF         31.853760
GIDP       34.655207
dtype: float64
```

[222]:
```
df_early = df[df['yearID'] <= 1930]
```

[223]:
```
100 * df_early.isnull().sum() / len(df)
```

[223]:
```
retroID    0.000000
yearID     0.000000
```

```
W          0.000000
L          0.000000
G          0.000000
GS         0.000000
CG         0.000000
SHO        0.000000
SV         0.000000
IPouts     0.000000
H          0.000000
ER         0.000000
HR         0.000000
BB         0.000000
SO         0.000000
BAOpp      0.002477
ERA        0.042108
IBB        6.442584
WP         0.000000
HBP        0.000000
BK         0.000000
BFP        0.007431
GF         0.000000
R          0.000000
SH         6.442584
SF         6.442584
GIDP       6.442584
dtype: float64
```

[224]:
```
df_modern = df[df['yearID'] >= 1980]
```

[225]:
```
100 * df_modern.isnull().sum() / len(df)
```

[225]:
```
retroID    0.000000
yearID     0.000000
W          0.000000
L          0.000000
G          0.000000
GS         0.000000
CG         0.000000
SHO        0.000000
SV         0.000000
IPouts     0.000000
H          0.000000
ER         0.000000
HR         0.000000
BB         0.000000
SO         0.000000
BAOpp      0.019816
```

```
ERA         0.056970
IBB         0.000000
WP          0.000000
HBP         0.000000
BK          0.000000
BFP         0.000000
GF          0.000000
R           0.000000
SH          0.000000
SF          0.000000
GIDP        0.000000
dtype: float64
```

Luckily the more modern data is barely missing any information.

[226]:
```
df_mid = df[(df['yearID'] > 1935) & (df['yearID'] < 1975)]
```

[227]:
```
100 * df_mid.isnull().sum() / len(df)
```

[227]:
```
retroID       0.000000
yearID        0.000000
W             0.000000
L             0.000000
G             0.000000
GS            0.000000
CG            0.000000
SHO           0.000000
SV            0.000000
IPouts        0.000000
H             0.000000
ER            0.000000
HR            0.000000
BB            0.000000
SO            0.000000
BAOpp         0.007431
ERA           0.066878
IBB          11.428713
WP            0.000000
HBP           0.000000
BK            0.000000
BFP           0.000000
GF            0.000000
R             0.000000
SH           22.845041
SF           22.845041
GIDP         25.646488
dtype: float64
```

We see that much of the lost data comes within this 40-year span. I think that given what the major missing information is - intentional bases on balls, sacrifice hits, sacrifice flies and grounded into double play - and the fact that these statistics are not often used as primary indicators of a pitcher's ability, coupled with the fact that it's mostly localized within less than half of our time frame, I can be forgiven for just filling these values as 0.

[228]:
```
df['IBB'].fillna(0, inplace=True)
df['SH'].fillna(0, inplace=True)
df['SF'].fillna(0, inplace=True)
df['GIDP'].fillna(0, inplace=True)
```

[229]:
```
100 * df.isnull().sum() / len(df)
```

[229]:
```
retroID     0.000000
yearID      0.000000
W           0.000000
L           0.000000
G           0.000000
GS          0.000000
CG          0.000000
SHO         0.000000
SV          0.000000
IPouts      0.000000
H           0.000000
ER          0.000000
HR          0.000000
BB          0.000000
SO          0.000000
BAOpp       0.029724
ERA         0.183295
IBB         0.000000
WP          0.000000
HBP         0.000000
BK          0.000000
BFP         0.007431
GF          0.000000
R           0.000000
SH          0.000000
SF          0.000000
GIDP        0.000000
dtype: float64
```

We're left with three fields that having missing data: opponents' batting average, earned run average and batters faced by pitcher. We'll have to do some data exploration on these because I don't want to just fill them with 0s.

Missing Values: BAOpp

[230]:
```python
df_baopp_missing = df[df['BAOpp'].isnull()]
```

[231]:
```python
df_baopp_missing.shape
```

[231]:
```
(12, 27)
```

[232]:
```python
df_baopp_missing.sort_values('retroID')
```

[232]:
```
        retroID  yearID  W  L  G  GS  CG  SHO  SV  IPouts  ...  IBB  WP  HBP  \
14000  apodb101    1973  0  0  1   0   0    0   0       0  ...  0.0   0    0
19114  arrof001    1986  0  0  1   0   0    0   0       0  ...  0.0   0    0
39581  brotr001    2018  0  0  1   0   0    0   0       0  ...  0.0   0    0
36447  dunnj001    2014  0  0  1   0   0    0   0       2  ...  0.0   2    0
38848  eschj001    2017  0  0  1   0   0    0   0       0  ...  0.0   0    0
3709   fordw103    1936  0  0  1   0   0    0   0       0  ...  0.0   0    0
297    glasn101    1920  0  0  1   0   0    0   0       7  ...  0.0   0    1
26791  halts001    2000  0  0  1   0   0    0   0       0  ...  0.0   0    0
27036  radis001    2000  0  0  1   0   0    0   0       0  ...  0.0   0    0
36189  tolls002    2013  0  0  1   0   0    0   0       0  ...  0.0   0    0
36208  villb002    2013  0  0  1   0   0    0   0       0  ...  0.0   0    0
13621  younl101    1971  0  0  1   0   0    0   0       0  ...  0.0   0    0

       BK   BFP  GF  R   SH   SF  GIDP
14000   0   2.0   0  1  0.0  0.0   0.0
19114   0   3.0   0  0  0.0  0.0   0.0
39581   0   2.0   0  1  0.0  0.0   0.0
36447   0   2.0   0  0  0.0  1.0   0.0
38848   0   2.0   0  0  0.0  0.0   0.0
3709    0   3.0   0  2  0.0  0.0   0.0
297     0  12.0   0  4  0.0  0.0   0.0
26791   0   1.0   0  0  0.0  0.0   0.0
27036   0   1.0   0  0  0.0  0.0   0.0
36189   0   2.0   0  0  0.0  0.0   0.0
36208   0   1.0   1  0  0.0  0.0   0.0
13621   0   0.0   0  0  0.0  0.0   0.0

[12 rows x 27 columns]
```

Nobody appears in this table more than once. We'll hope that we can get career numbers for them and fill with the average. For anyone who only appears once I'll go with the league average, and I'll add a standard deviation since they probably we're exactly middle-of-the-road. Probably not the best way but it's only a few datapoints.

[233]:
```python
baopp_checks = df[df['retroID'].isin(df_baopp_missing['retroID'])].
↪sort_values('retroID')
```

[234]:
```python
baopp_checks['retroID'].value_counts()
```

[234]:
```
radis001    11
arrof001     9
brotr001     7
tolls002     5
apodb101     5
villb002     4
dunnj001     2
eschj001     2
halts001     2
younl101     1
glasn101     1
fordw103     1
Name: retroID, dtype: int64
```

We only have three data points with one year of appearances. We'll fill those with the league average.

[235]:
```python
val_counts = baopp_checks['retroID'].value_counts()
```

[236]:
```python
from itertools import compress
# Get list of retroIDs of players who only have one year appearance
```

[237]:
```python
one_time_players = list(compress(val_counts.index, val_counts.eq(1)))
```

[238]:
```python
df[df['retroID'].isin(one_time_players)]
```

[238]:
```
        retroID  yearID  W  L  G  GS  CG  SHO  SV  IPouts  ...  IBB  WP  HBP  \
297    glasn101    1920  0  0  1   0   0    0   0       7  ...  0.0   0    1
3709   fordw103    1936  0  0  1   0   0    0   0       0  ...  0.0   0    0
13621  younl101    1971  0  0  1   0   0    0   0       0  ...  0.0   0    0

       BK   BFP  GF  R   SH   SF  GIDP
297     0  12.0   0  4  0.0  0.0   0.0
3709    0   3.0   0  2  0.0  0.0   0.0
13621   0   0.0   0  0  0.0  0.0   0.0

[3 rows x 27 columns]
```

[239]:
```python
df['BAOpp'].mean()
```

[239]:
```
0.27445659068384537
```

[240]:
```python
df['BAOpp'].std()
```

[240]:
```
0.07751058079199835
```

[241]:
```python
filled_baopp = df['BAOpp'].mean() + df['BAOpp'].std()
```

[242]:
```
filled_baopp
```

[242]:
```
0.3519671714758437
```

[243]:
```
df.loc[df['retroID'].isin(one_time_players), ['BAOpp']] = filled_baopp
```

[244]:
```
df[df['retroID'].isin(one_time_players)]['BAOpp']
```

[244]:
```
297      0.351967
3709     0.351967
13621    0.351967
Name: BAOpp, dtype: float64
```

[245]:
```
df_baopp_missing = df[df['BAOpp'].isnull()].sort_values('retroID')
```

[246]:
```
df_baopp_missing
```

[246]:
```
        retroID  yearID  W  L  G  GS  CG  SHO  SV  IPouts  ...  IBB  WP  HBP  \
14000  apodb101    1973  0  0  1   0   0    0   0       0  ...  0.0   0    0
19114  arrof001    1986  0  0  1   0   0    0   0       0  ...  0.0   0    0
39581  brotr001    2018  0  0  1   0   0    0   0       0  ...  0.0   0    0
36447  dunnj001    2014  0  0  1   0   0    0   0       2  ...  0.0   2    0
38848  eschj001    2017  0  0  1   0   0    0   0       0  ...  0.0   0    0
26791  halts001    2000  0  0  1   0   0    0   0       0  ...  0.0   0    0
27036  radis001    2000  0  0  1   0   0    0   0       0  ...  0.0   0    0
36189  tolls002    2013  0  0  1   0   0    0   0       0  ...  0.0   0    0
36208  villb002    2013  0  0  1   0   0    0   0       0  ...  0.0   0    0

       BK  BFP  GF  R   SH   SF  GIDP
14000   0  2.0   0  1  0.0  0.0   0.0
19114   0  3.0   0  0  0.0  0.0   0.0
39581   0  2.0   0  1  0.0  0.0   0.0
36447   0  2.0   0  0  0.0  1.0   0.0
38848   0  2.0   0  0  0.0  0.0   0.0
26791   0  1.0   0  0  0.0  0.0   0.0
27036   0  1.0   0  0  0.0  0.0   0.0
36189   0  2.0   0  0  0.0  0.0   0.0
36208   0  1.0   1  0  0.0  0.0   0.0

[9 rows x 27 columns]
```

Now we just have to worry about the players with at least two years of appearances.

[247]:
```
baopp_checks = df[df['retroID'].isin(df_baopp_missing['retroID'])].
 ↪sort_values('retroID')
```

[248]:
```
baopp_checks['retroID'].value_counts()
```

[248]:
```
radis001    11
arrof001     9
brotr001     7
tolls002     5
apodb101     5
villb002     4
dunnj001     2
eschj001     2
halts001     2
Name: retroID, dtype: int64
```

[249]:
```
one_time_players = list(val_counts.index)
```

[250]:
```
one_time_players
```

[250]:
```
['radis001',
 'arrof001',
 'brotr001',
 'tolls002',
 'apodb101',
 'villb002',
 'dunnj001',
 'eschj001',
 'halts001',
 'younl101',
 'glasn101',
 'fordw103']
```

[251]:
```
df[df['retroID'] == 'radis001']['BAOpp']
```

[251]:
```
21355    0.241
21865    0.206
22335    0.243
22871    0.268
23957    0.309
24557    0.264
25145    0.236
25740    0.272
26377    0.270
27036      NaN
27708    0.400
Name: BAOpp, dtype: float64
```

[252]:
```
df[df['retroID'] == 'radis001']['BAOpp'].mean().round(4)
```

[252]:
```
0.2709
```

This looks good, so we'll iterate through and assign each player's missing BAOpp as his career mean for that state.

[253]:
```
df['BAOpp'] = df.groupby("retroID")['BAOpp'].transform(lambda baopp: baopp.
↪fillna(baopp.mean()))
```

[254]:
```
100 * df.isnull().sum() / len(df)
```

[254]:
```
retroID    0.000000
yearID     0.000000
W          0.000000
L          0.000000
G          0.000000
GS         0.000000
CG         0.000000
SHO        0.000000
SV         0.000000
IPouts     0.000000
H          0.000000
ER         0.000000
HR         0.000000
BB         0.000000
SO         0.000000
BAOpp      0.000000
ERA        0.183295
IBB        0.000000
WP         0.000000
HBP        0.000000
BK         0.000000
BFP        0.007431
GF         0.000000
R          0.000000
SH         0.000000
SF         0.000000
GIDP       0.000000
dtype: float64
```

[255]:
```
df.head()
```

[255]:
```
    retroID  yearID   W   L   G  GS  CG  SHO  SV  IPouts  ...  IBB  WP  HBP  \
0  adamb104    1919  17  10  34  29  23    6   1     790  ...  0.0   2    3
1  adamw101    1919   0   0   1   0   0    0   0      14  ...  0.0   0    1
2  alexg102    1919  16  11  30  27  20    9   1     705  ...  0.0   1    0
3  altrn101    1919   0   0   1   0   0    0   0       0  ...  0.0   0    0
4  amesr101    1919   3   5  23   7   1    0   1     210  ...  0.0   3    1

   BK     BFP  GF   R   SH   SF  GIDP
0   0  1017.0   5  66  0.0  0.0   0.0
1   0    21.0   1   2  0.0  0.0   0.0
```

```
2   0   906.0   3  51  0.0  0.0   0.0
3   0     4.0   0   4  0.0  0.0   0.0
4   0   314.0  10  44  0.0  0.0   0.0

[5 rows x 27 columns]
```

Missing Values: ERA

[256]:
```
df_era_missing = df[df['ERA'].isnull()].sort_values('retroID')
```

[257]:
```
df_era_missing
```

[257]:
```
        retroID  yearID  W  L  G  GS  CG  SHO  SV  IPouts  ...  IBB  WP  HBP  \
3      altrn101    1919  0  0  1   0   0    0   0       0  ...  0.0   0    0
20491  alvaw001    1989  0  1  1   1   0    0   0       0  ...  0.0   0    0
14000  apodb101    1973  0  0  1   0   0    0   0       0  ...  0.0   0    0
19114  arrof001    1986  0  0  1   0   0    0   0       0  ...  0.0   0    0
663    bents101    1922  0  0  1   0   0    0   0       0  ...  0.0   0    0
...         ...     ... .. .. ..  ..  ..  ...  ..     ...  ...  ...  ..  ...
40336  weisz001    2018  0  0  1   0   0    0   0       0  ...  0.0   0    0
6277   willa103    1946  0  0  1   0   0    0   0       0  ...  0.0   0    0
10476  willt102    1962  0  0  1   0   0    0   0       0  ...  0.0   0    0
18235  wortr101    1983  0  0  1   0   0    0   0       0  ...  0.0   0    0
13621  younl101    1971  0  0  1   0   0    0   0       0  ...  0.0   0    0

       BK  BFP  GF  R   SH   SF  GIDP
3       0  4.0   0  4  0.0  0.0   0.0
20491   0  5.0   0  3  0.0  0.0   0.0
14000   0  2.0   0  1  0.0  0.0   0.0
19114   0  3.0   0  0  0.0  0.0   0.0
663     0  2.0   0  0  0.0  0.0   0.0
...    ..  ...  .. ..  ...  ...   ...
40336   0  4.0   0  4  0.0  0.0   0.0
6277    0  2.0   0  0  0.0  0.0   0.0
10476   0  3.0   0  1  0.0  0.0   0.0
18235   0  4.0   0  1  0.0  0.0   0.0
13621   0  0.0   0  0  0.0  0.0   0.0

[74 rows x 27 columns]
```

[258]:
```
df_era_missing['retroID'].nunique()
```

[258]:
```
74
```

74 rows and 74 unique IDs means that each of these players is only missing the ERA stat for one year. We'll first see, like with BAOpp, if they played other years.

[259]:
```
era_checks = df[df['retroID'].isin(df_era_missing['retroID'])].
    ↪sort_values('retroID')
```

[260]:
```
val_counts = era_checks['retroID'].value_counts()
```

[261]:
```
one_time_players = list(compress(val_counts.index, val_counts.eq(1)))
```

[262]:
```
one_time_players
```

[262]:
```
['russr102',
 'moorb104',
 'palam101',
 'musis101',
 'garda103',
 'weisz001',
 'koenw101',
 'bents101',
 'fordw103',
 'schej101',
 'brucf101',
 'davav101',
 'hamad101',
 'walkm101',
 'sundg101',
 'younl101',
 'browj102']
```

[263]:
```
df['ERA'].mean()
```

[263]:
```
5.165393567919002
```

[264]:
```
df['ERA'].std()
```

[264]:
```
5.2791159271962815
```

Intuitively, 5.17 is a bit of a high ERA. Though the stat can grow infinitely in theory and low numbers are very difficult, I don't want to assign 10 to the missing values. It's just too much. I'll just do mean + std/2.

[265]:
```
filled_era = df['ERA'].mean() + (df['ERA'].std())/2
```

[266]:
```
df.loc[df['retroID'].isin(one_time_players), ['ERA']] = filled_era
df[df['retroID'].isin(one_time_players)]['ERA']
```

[266]:
```
101     7.804952
173     7.804952
663     7.804952
```

```
722      7.804952
931      7.804952
1447     7.804952
1755     7.804952
2154     7.804952
3709     7.804952
4485     7.804952
4513     7.804952
7678     7.804952
8773     7.804952
9903     7.804952
12532    7.804952
13621    7.804952
40336    7.804952
Name: ERA, dtype: float64
```

[267]:
```
df_era_missing = df[df['ERA'].isnull()].sort_values('retroID')
```

We'll continue to follow the same method as for BAOpp with the rest of the missing values.

[268]:
```
df_era_missing.shape
```

[268]:
```
(57, 27)
```

[269]:
```
era_checks = df[df['retroID'].isin(df_era_missing['retroID'])].
 ↪sort_values('retroID')
era_checks['retroID'].value_counts()
```

[269]:
```
hillr001    16
choar001    16
mclic101    15
alvaw001    15
farme101    14
medid101    14
kosld101    13
coopm101    13
radis001    11
burkb102    10
owchb001    10
milna101    10
arrof001     9
chent101     9
harvb001     9
perim001     9
deanp101     9
ray-j101     9
pennb001     7
```

```
navaj101     7
brotr001     7
jonen001     7
painp101     6
willt102     6
moorc101     6
tolls002     5
luebs101     5
reina102     5
scarm101     5
mccud001     5
apodb101     5
blake101     4
villb002     4
kreur101     4
wortr101     4
tankd001     3
pitls101     3
stufp101     3
sabee001     3
geard101     3
kochm001     3
vaugp101     3
roeto101     2
urdal001     2
willa103     2
green002     2
dibup101     2
uhl-b101     2
wardd101     2
kella101     2
eschj001     2
kammb101     2
jeant101     2
engej101     2
smitd105     2
halts001     2
altrn101     2
Name: retroID, dtype: int64
```

[270]:
```
df['ERA'] = df.groupby("retroID")['ERA'].transform(lambda era: era.fillna(era.
 ↪mean()))
```

[271]:
```
100 * df.isnull().sum() / len(df)
```

[271]:
```
retroID     0.000000
yearID      0.000000
```

```
W          0.000000
L          0.000000
G          0.000000
GS         0.000000
CG         0.000000
SHO        0.000000
SV         0.000000
IPouts     0.000000
H          0.000000
ER         0.000000
HR         0.000000
BB         0.000000
SO         0.000000
BAOpp      0.000000
ERA        0.000000
IBB        0.000000
WP         0.000000
HBP        0.000000
BK         0.000000
BFP        0.007431
GF         0.000000
R          0.000000
SH         0.000000
SF         0.000000
GIDP       0.000000
dtype: float64
```

Missing Values: BFP

[272]:
```
df_bfp_missing = df[df['BFP'].isnull()].sort_values('retroID')
```

[273]:
```
df_bfp_missing
```

[273]:
```
       retroID  yearID  W  L   G  GS  CG  SHO  SV  IPouts  ...  IBB  WP  HBP  \
709   fourj101    1922  0  0   1   0   0    0   0       3  ...  0.0   0    0
1171  jamel101    1924  0  0   1   0   0    0   0       3  ...  0.0   0    0
802   pierb103    1922  3  9  29  12   7    1   0     364  ...  0.0   4    6

      BK  BFP  GF   R   SH   SF  GIDP
709    0  NaN   1   0  0.0  0.0   0.0
1171   0  NaN   1   2  0.0  0.0   0.0
802    0  NaN  10  77  0.0  0.0   0.0

[3 rows x 27 columns]
```

These amounts are negligable. Rather than take averages or set to 0, I'm going to get a little clever. A pitcher intuitively faces hitters until he gets an out, and comes out of the game if he can't get one. There's a lot of work I could do to get a good approximation, but since I'm only filling 3 rows out of over 40,000 I'm just going to set the missing values to (IPouts - G). This gives us the number of outs a pitcher earned minus 1 for each game he appeared in (presumably this 1 represents the final batter, whom the pitcher did not get out).

[274]:
```
df['BFP'].fillna(df['IPouts'] - df['G'], inplace=True)
```

[275]:
```
100 * df.isnull().sum() / len(df)
```

[275]:
```
retroID     0.0
yearID      0.0
W           0.0
L           0.0
G           0.0
GS          0.0
CG          0.0
SHO         0.0
SV          0.0
IPouts      0.0
H           0.0
ER          0.0
HR          0.0
BB          0.0
SO          0.0
BAOpp       0.0
ERA         0.0
IBB         0.0
WP          0.0
HBP         0.0
BK          0.0
BFP         0.0
GF          0.0
R           0.0
SH          0.0
SF          0.0
GIDP        0.0
dtype: float64
```

Data Aggregation

Now we can group by retroID, but we need to be more careful than we were with fielding, catching and batting. Some of these stats are averages and some are sum totals, so when we group by we need to handle them differently. We'll split them into two dataframes and do a join. The splitting step will require some intuitive knowledge about baseball statistics. But before we do all of this, we can now get rid of the yearID column.

[276]:
```
df.drop(columns=['yearID'], inplace=True)
```

[277]:
```
df.columns
```

[277]:
```
Index(['retroID', 'W', 'L', 'G', 'GS', 'CG', 'SHO', 'SV', 'IPouts', 'H', 'ER',
       'HR', 'BB', 'SO', 'BAOpp', 'ERA', 'IBB', 'WP', 'HBP', 'BK', 'BFP', 'GF',
       'R', 'SH', 'SF', 'GIDP'],
      dtype='object')
```

[278]:
```
average_stats = ['BAOpp', 'ERA']
```

It's only two columns that are averages, and we could probably do without ERA since it's a function of batters faced and runs allowed. We'll keep it since it's such a fundamental statistic in the sport and we have to split anyway for BAOpp, which is a very important one to keep track of.

[279]:
```
df_avgs = df[['retroID', 'BAOpp', 'ERA']]
```

[280]:
```
df_avgs.head()
```

[280]:

	retroID	BAOpp	ERA
0	adamb104	0.22	1.98
1	adamw101	0.38	3.86
2	alexg102	0.21	1.72
3	altrn101	1.00	0.00
4	amesr101	0.31	4.89

[281]:
```
df_sums = df.drop(columns=average_stats)
```

[282]:
```
df_sums.head()
```

[282]:
```
    retroID   W   L   G  GS  CG  SHO  SV  IPouts    H  ...  IBB  WP  HBP  BK  \
0  adamb104  17  10  34  29  23    6   1     790  213  ...  0.0   2    3   0
1  adamw101   0   0   1   0   0    0   0      14    7  ...  0.0   0    1   0
2  alexg102  16  11  30  27  20    9   1     705  180  ...  0.0   1    0   0
3  altrn101   0   0   1   0   0    0   0       0    4  ...  0.0   0    0   0
4  amesr101   3   5  23   7   1    0   1     210   88  ...  0.0   3    1   0

      BFP  GF   R   SH   SF  GIDP
0  1017.0   5  66  0.0  0.0   0.0
1    21.0   1   2  0.0  0.0   0.0
2   906.0   3  51  0.0  0.0   0.0
3     4.0   0   4  0.0  0.0   0.0
4   314.0  10  44  0.0  0.0   0.0

[5 rows x 24 columns]
```

[283]:
```
df_avgs.shape
```

[283]:
```
(40372, 3)
```

[284]:
```
df_sums.shape
```

[284]: (40372, 24)

[289]:
```
df_avgs = df_avgs.groupby('retroID').mean().round(4).reset_index()
```

[292]:
```
df_avgs.shape
```

[292]: (7835, 3)

[288]:
```
df_sums = df_sums.groupby('retroID').sum().reset_index()
```

[293]:
```
df_sums.shape
```

[293]: (7835, 24)

[291]:
```
pd.merge(df_avgs, df_sums, on='retroID')
```

[291]:
```
       retroID   BAOpp     ERA   W    L    G   GS  CG  SHO  SV  ...   IBB  WP  \
0     aardd001  0.2574  5.1944  16   18  331    0   0    0  69  ...  22.0  12
1     aased001  0.2508  3.4931  66   60  448   91  22    5  82  ...  45.0  22
2     abadf001  0.2501  4.0733   8   27  363    6   0    0   2  ...  10.0   9
3     abbog001  0.2786  4.3317  62   83  248  206  37    5   0  ...  28.0  18
4     abboj001  0.2804  4.4964  87  108  263  254  31    6   0  ...  30.0  53
...        ...     ...     ...  ..  ...  ...  ...  ..  ...  ..  ...   ...  ..
7830  zolds101  0.2700  3.6890  43   53  250   93  30    5   8  ...   0.0   8
7831  zubeb101  0.2717  5.3617  43   42  224   65  23    3   6  ...   0.0  28
7832  zumaj001  0.2286  3.4420  13   12  171    0   0    0   5  ...  11.0  16
7833  zuveg101  0.2760  4.1280  32   36  265   31   9    2  40  ...  29.0  10
7834  zycht001  0.2183  2.8000   7    3   70    1   0    0   1  ...   5.0   2

      HBP  BK     BFP   GF    R    SH    SF   GIDP
0      16   1  1475.0  141  169  17.0  11.0   21.0
1       7   3  4730.0  235  503  50.0  34.0  106.0
2      12   2  1350.0   96  137   7.0  12.0   22.0
3      32   5  5508.0   13  707  60.0  39.0  111.0
4      32  11  7211.0    5  880  70.0  47.0  200.0
...   ...  ..     ...  ...  ...   ...   ...    ...
7830    3   4  3946.0   78  423   0.0   0.0    0.0
7831    4   1  3476.0   90  418   0.0   0.0    0.0
7832    4   0   911.0   35   80   6.0  10.0   10.0
7833   27   1  2746.0  139  296   0.0   0.0    0.0
7834    8   1   309.0   14   24   1.0   3.0    6.0

[7835 rows x 26 columns]
```

This gives us the appropriate amount of rows and columns, so the merge worked. We'll send this as our final output.

[294]:
```
df = pd.merge(df_avgs, df_sums, on='retroID')
```

[296]:
```
df.shape
```

[296]:
```
(7835, 26)
```

We're ready to export the resulting table by saving to a csv.

add_advanced_pitching_stats

April 29, 2020

[53]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

[54]:
```
df = pd.read_csv('../core/output/pitchers.csv')
df_adv = pd.read_csv('../core/output/advanced_pitching.csv')
```

Adding Advanced Stats

[55]:
```
df_adv.sort_values('retroID')
```

[55]:
```
       retroID        IP   K/9  BB/9  HR/9  BABIP  LOB%   ERA   FIP   WAR
2866  aardd001  0.062360  9.08  4.89  1.09  0.285  74.5  4.27  4.45   1.1
841   aased001  0.205233  5.20  3.71  0.72  0.282  73.4  3.80  3.85  11.7
3237  abadf001  0.061102  7.62  3.16  1.14  0.281  77.7  3.67  4.24   0.6
949   abbog001  0.237967  3.39  2.46  1.13  0.278  69.3  4.39  4.46  10.2
394   abboj001  0.309765  4.77  3.33  0.83  0.295  70.0  4.25  4.25  22.7
...        ...       ...   ...   ...   ...    ...   ...   ...   ...   ...
1030  zolds101  0.171925  2.00  2.91  0.52  0.267  70.7  3.54  3.80   9.3
1934  zubeb101  0.145445  4.39  5.36  0.40  0.283  69.0  4.28  3.96   3.3
2098  zumaj001  0.038711  9.01  4.89  0.77  0.267  78.7  3.00  3.94   2.7
2399  zuveg101  0.118817  3.12  2.84  0.78  0.270  73.2  3.54  3.93   1.9
2808  zycht001  0.013360  9.91  4.21  0.37  0.293  79.1  2.72  3.22   1.1

[8025 rows x 10 columns]
```

[56]:
```
df
```

[56]:
```
       retroID  BAOpp     ERA  CG  SHO  IPouts     H   ER   HR   BB  ...  WP  \
0     aardd001  0.2574  5.1944   0    0    1011   296  160   41  183  ...  12
1     aased001  0.2508  3.4931  22    5    3328  1085  468   89  457  ...  22
2     abadf001  0.2447  4.0810   0    0     992   309  135   42  116  ...  10
3     abbog001  0.2786  4.3317  37    5    3858  1405  627  162  352  ...  18
4     abboj001  0.2804  4.4964  31    6    5022  1779  791  154  620  ...  53
...        ...     ...     ...  ..  ...     ...   ...  ...  ...  ...  ...  ..
8020  zolds101  0.2700  3.6890  30    5    2788   956  366   54  301  ...   8
8021  zubeb101  0.2717  5.3617  23    3    2358   767  374   35  468  ...  28
8022  zumaj001  0.2286  3.4420   0    0     629   169   71   18  114  ...  16
```

```
8023  zuveg101  0.2760  4.1280   9    2    1927   660  253   56  203  ...   10
8024  zycht001  0.2183  2.8000   0    0     218    57   22    3   34  ...    2

      HBP  BK   BFP   GF    R  SH  SF  GIDP        K%
0      16   1  1475  141  169  17  11    21  0.230508
1       7   3  4730  235  503  50  34   106  0.135518
2      12   2  1399   97  143   7  12    25  0.200143
3      32   5  5508   13  707  60  39   111  0.087872
4      32  11  7211    5  880  70  47   200  0.123145
...   ...  ..   ...  ...  ...  ..  ..   ...       ...
8020    3   4  3946   78  423   0   0     0  0.052458
8021    4   1  3476   90  418   0   0     0  0.110184
8022    4   0   911   35   80   6  10    10  0.230516
8023   27   1  2746  139  296   0   0     0  0.081209
8024    8   1   309   14   24   1   3     6  0.258900

[8025 rows x 22 columns]
```

[57]:
```
df = df.drop(columns=['ERA'])
```

[58]:
```
df = df.merge(df_adv, on='retroID' ,how='left')
```

[59]:
```
100 * df.isnull().sum() / len(df)
```

[59]:
```
retroID    0.0
BAOpp      0.0
CG         0.0
SHO        0.0
IPouts     0.0
H          0.0
ER         0.0
HR         0.0
BB         0.0
SO         0.0
IBB        0.0
WP         0.0
HBP        0.0
BK         0.0
BFP        0.0
GF         0.0
R          0.0
SH         0.0
SF         0.0
GIDP       0.0
K%         0.0
IP         0.0
K/9        0.0
```

```
BB/9       0.0
HR/9       0.0
BABIP      0.0
LOB%       0.0
ERA        0.0
FIP        0.0
WAR        0.0
dtype: float64
```

[60]:
```
df['Pitching'] = df[['K%', 'ERA', 'FIP', 'WAR']].mean(axis=1).round(3)
```

[61]:
```
df['Pitching']
```

[61]:
```
0       2.513
1       4.871
2       2.178
3       4.784
4       7.831
        ...
8020    4.173
8021    2.913
8022    2.468
8023    2.363
8024    1.825
Name: Pitching, Length: 8025, dtype: float64
```

Finalizing the new Pitching statistic

[64]:
```
df['Pitching'].mean()
```

[64]:
```
3.627380436137072
```

[65]:
```
df['Pitching'].min()
```

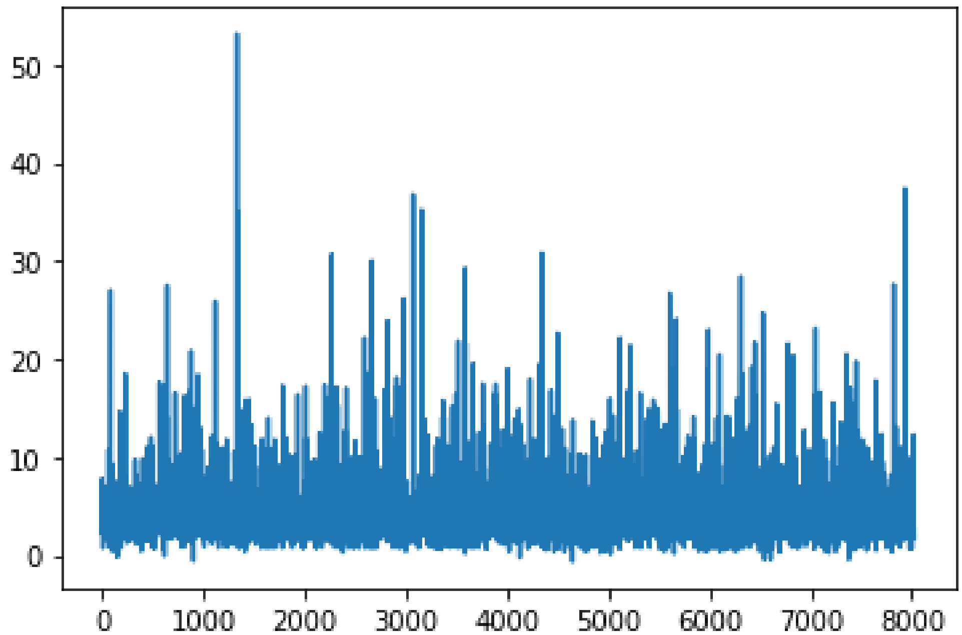
[65]:
```
-0.654
```

[66]:
```
df['Pitching'].max()
```

[66]:
```
53.138
```

[63]:
```
plt.plot(df['Pitching'])
```

[63]:
```
[<matplotlib.lines.Line2D at 0x127615250>]
```



[67]:
```
df[df['Pitching'] == df['Pitching'].max()]
```

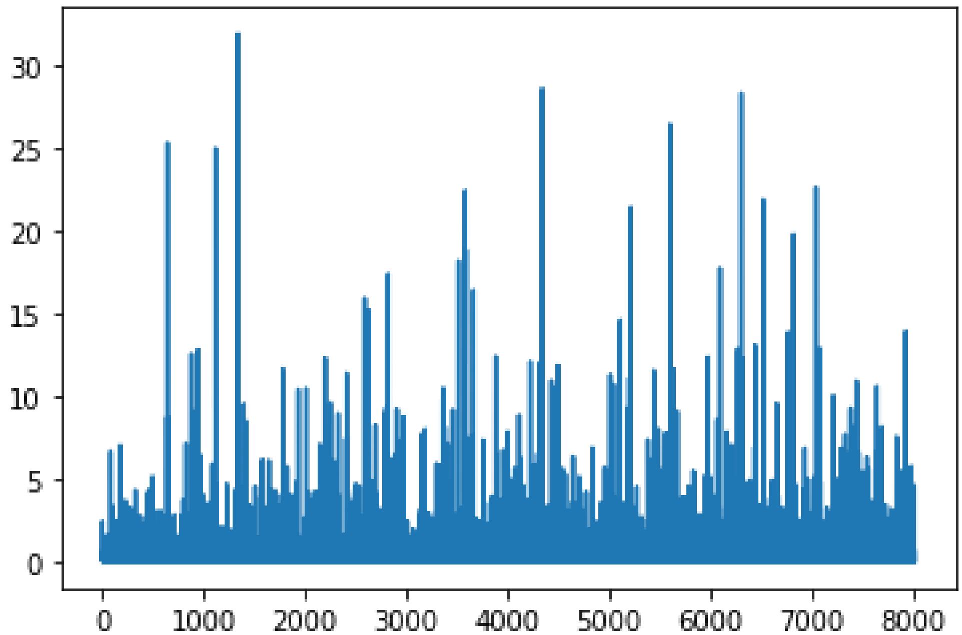
[67]:
```
       retroID  BAOpp  CG  SHO  IPouts  H  ER  HR  BB  SO  ...        IP  \
1333  cleaj101   0.83   0    0       1  5   7   0   3   1  ...  0.000019

       K/9  BB/9  HR/9  BABIP  LOB%    ERA    FIP  WAR  Pitching
1333  27.0  81.0   0.0    1.0  12.5  189.0  23.54 -0.1    53.138

[1 rows x 31 columns]
```

This immediately demonstrates an issue with our Pitching stat - a player with very few appearances, but who did well in those appearances, will be skewed too high. We need to also take innings pitched into account.

[91]:
```
df['Pitching'] = df[['K%', 'ERA', 'FIP', 'WAR']].mean(axis=1).round(3) * df['IP']
```

[92]:
```
plt.plot(df['Pitching'])
```

[92]:
```
[<matplotlib.lines.Line2D at 0x12793b850>]
```




```
[93]: df[df['Pitching'] == df['Pitching'].max()]
```

```
[93]:        retroID   BAOpp   CG  SHO  IPouts     H    ER   HR    BB    SO  ...  \
1341  clemr001  0.2308  118   46   14750  4185  1707  363  1580  4672  ...

            IP   K/9  BB/9  HR/9  BABIP  LOB%   ERA   FIP    WAR   Pitching
1341  0.909717  8.55  2.89  0.66  0.284  74.6  3.12  3.09  133.7  31.871935

[1 rows x 31 columns]
```

This looks like it worked, but let's explore deeper.

```
[94]: df['Pitching'].mean()
```

```
[94]: 0.49352977097414313
```

```
[95]: df['Pitching'].min()
```

```
[95]: -5.0887375e-05
```

```
[96]: df['Pitching'].max()
```

```
[96]: 31.871935094999998
```

```
[97]: df.sort_values('Pitching').tail(10)
```

```
[97]:      retroID   BAOpp   CG  SHO  IPouts     H    ER   HR    BB    SO  ...  \
5219  niekp001  0.2570  245   45   16213  5044  2012  482  1809  3342  ...
6529  seavt001  0.2285  231   61   14348  3971  1521  380  1390  3640  ...
3588  johnr005  0.2252  100   37   12406  3346  1513  411  1497  4875  ...
7049  suttd001  0.2376  178   58   15847  4692  1914  472  1343  3574  ...
1121  carls001  0.2546  254   55   15652  4672  1864  414  1833  4136  ...
645   blylb001  0.2487  242   60   14910  4632  1830  430  1322  3701  ...
5607  perrg101  0.2540  303   53   16051  4938  1846  399  1379  3534  ...
6311  ryann001  0.2088  222   61   16158  3923  1911  321  2795  5714  ...
4347  maddg002  0.2553  109   35   15025  4726  1756  353   999  3371  ...
1341  clemr001  0.2308  118   46   14750  4185  1707  363  1580  4672  ...

            IP    K/9  BB/9  HR/9  BABIP  LOB%   ERA   FIP    WAR   Pitching
5219  1.000000   5.57  3.01  0.80  0.270  73.6  3.35  3.62   78.5  21.404000
6529  0.884921   6.85  2.62  0.72  0.259  76.7  2.86  3.04   92.7  21.854894
3588  0.765178  10.61  3.26  0.89  0.291  74.7  3.29  3.19  110.4  22.412829
7049  0.977406   6.09  2.29  0.80  0.261  73.6  3.26  3.24   85.9  22.618152
1121  0.965397   7.13  3.16  0.71  0.279  74.1  3.22  3.15   96.9  24.969993
645   0.919672   6.70  2.39  0.78  0.282  74.1  3.31  3.19  103.3  25.286382
5607  0.990008   5.94  2.32  0.67  0.275  73.3  3.11  3.06  100.5  26.441134
6311  0.996651   9.55  4.67  0.54  0.265  73.1  3.19  2.97  107.2  28.307878
4347  0.926722   6.06  1.80  0.63  0.281  72.3  3.16  3.26  116.7  28.562499
1341  0.909717   8.55  2.89  0.66  0.284  74.6  3.12  3.09  133.7  31.871935

[10 rows x 31 columns]
```

```
[98]: df[df['retroID'] == 'kersc001']
```

```
[98]:      retroID   BAOpp  CG  SHO  IPouts     H   ER   HR   BB    SO  ...  \
3761  kersc001  0.2105  25   15    6824  1715  617  173  577  2464  ...

            IP   K/9  BB/9  HR/9  BABIP  LOB%   ERA   FIP   WAR  Pitching
3761  0.420829  9.75  2.28  0.68   0.27  79.4  2.44  2.74  64.5  7.359878

[1 rows x 31 columns]
```

This looks good, but we intuitively see a problem with the Pitching stat. ERA and FIP are part of the average, but a low ERA/FIP is better than a high one. We need to subtract them rather than add. With this change, I'm going to see how the stat looks without taking IP into account.

```
[132]: df['-ERA'] = 0 - df['ERA']
df['-FIP'] = 0 - df['FIP']
```

```
[133]: df
```

```
[133]:      retroID   BAOpp  CG  SHO  IPouts    H   ER  HR   BB   SO  ...  BB/9  \
0     aardd001  0.2574   0    0    1011  296  160  41  183  340  ...  4.89
```

```
1     aased001  0.2508  22    5    3328  1085  468   89  457  641  ...  3.71
2     abadf001  0.2447   0    0     992   309  135   42  116  280  ...  3.16
3     abbog001  0.2786  37    5    3858  1405  627  162  352  484  ...  2.46
4     abboj001  0.2804  31    6    5022  1779  791  154  620  888  ...  3.33
...        ...     ...  ..  ...     ...   ...  ...  ...  ...  ...  ...   ...
8020  zolds101  0.2700  30    5    2788   956  366   54  301  207  ...  2.91
8021  zubeb101  0.2717  23    3    2358   767  374   35  468  383  ...  5.36
8022  zumaj001  0.2286   0    0     629   169   71   18  114  210  ...  4.89
8023  zuveg101  0.2760   9    2    1927   660  253   56  203  223  ...  2.84
8024  zycht001  0.2183   0    0     218    57   22    3   34   80  ...  4.21

      HR/9  BABIP  LOB%   ERA   FIP   WAR  Pitching  -ERA  -FIP
0     1.09  0.285  74.5  4.27  4.45   1.1    -1.847 -4.27 -4.45
1     0.72  0.282  73.4  3.80  3.85  11.7     1.046 -3.80 -3.85
2     1.14  0.281  77.7  3.67  4.24   0.6    -1.777 -3.67 -4.24
3     1.13  0.278  69.3  4.39  4.46  10.2     0.359 -4.39 -4.46
4     0.83  0.295  70.0  4.25  4.25  22.7     3.581 -4.25 -4.25
...    ...    ...   ...   ...   ...   ...       ...   ...   ...
8020  0.52  0.267  70.7  3.54  3.80   9.3     0.503 -3.54 -3.80
8021  0.40  0.283  69.0  4.28  3.96   3.3    -1.207 -4.28 -3.96
8022  0.77  0.267  78.7  3.00  3.94   2.7    -1.002 -3.00 -3.94
8023  0.78  0.270  73.2  3.54  3.93   1.9    -1.372 -3.54 -3.93
8024  0.37  0.293  79.1  2.72  3.22   1.1    -1.145 -2.72 -3.22

[8025 rows x 33 columns]
```

[134]:
```python
df['Pitching'] = df[['K%', '-ERA', '-FIP', 'WAR']].mean(axis=1).round(3)
```

[135]:
```python
df.sort_values('Pitching').tail(10)
```

[135]:
```
       retroID   BAOpp   CG  SHO  IPouts     H    ER   HR    BB    SO  ...  \
7049  suttd001  0.2376  178   58   15847  4692  1914  472  1343  3574  ...
2824  grovl101  0.2535  298   35   11822  3849  1339  162  1187  2266  ...
6529  seavt001  0.2285  231   61   14348  3971  1521  380  1390  3640  ...
1121  carls001  0.2546  254   55   15652  4672  1864  414  1833  4136  ...
5607  perrg101  0.2540  303   53   16051  4938  1846  399  1379  3534  ...
645   blylb001  0.2487  242   60   14910  4632  1830  430  1322  3701  ...
6311  ryann001  0.2088  222   61   16158  3923  1911  321  2795  5714  ...
3588  johnr005  0.2252  100   37   12406  3346  1513  411  1497  4875  ...
4347  maddg002  0.2553  109   35   15025  4726  1756  353   999  3371  ...
1341  clemr001  0.2308  118   46   14750  4185  1707  363  1580  4672  ...

      BB/9  HR/9  BABIP  LOB%   ERA   FIP   WAR  Pitching  -ERA  -FIP
7049  2.29  0.80  0.261  73.6  3.26  3.24  85.9    19.891 -3.26 -3.24
2824  2.71  0.37  0.284  71.8  3.06  3.36  88.8    20.629 -3.06 -3.36
6529  2.62  0.72  0.259  76.7  2.86  3.04  92.7    21.747 -2.86 -3.04
1121  3.16  0.71  0.279  74.1  3.22  3.15  96.9    22.680 -3.22 -3.15
```

```
5607  2.32  0.67  0.275  73.3  3.11  3.06  100.5    23.623 -3.11 -3.06
645   2.39  0.78  0.282  74.1  3.31  3.19  103.3    24.245 -3.31 -3.19
6311  4.67  0.54  0.265  73.1  3.19  2.97  107.2    25.323 -3.19 -2.97
3588  3.26  0.89  0.291  74.7  3.29  3.19  110.4    26.051 -3.29 -3.19
4347  1.80  0.63  0.281  72.3  3.16  3.26  116.7    27.611 -3.16 -3.26
1341  2.89  0.66  0.284  74.6  3.12  3.09  133.7    31.930 -3.12 -3.09

[10 rows x 33 columns]
```

[136]:
```
df[df['retroID'] == 'kersc001']
```

[136]:
```
       retroID   BAOpp  CG  SHO  IPouts     H   ER   HR   BB    SO  ...  BB/9  \
3761  kersc001  0.2105  25   15    6824  1715  617  173  577  2464  ...  2.28

      HR/9  BABIP  LOB%   ERA   FIP   WAR  Pitching  -ERA  -FIP
3761  0.68   0.27  79.4  2.44  2.74  64.5    14.899 -2.44 -2.74

[1 rows x 33 columns]
```

[137]:
```
df[df['retroID'] == 'johnr005']
```

[137]:
```
       retroID   BAOpp   CG  SHO  IPouts     H    ER   HR    BB    SO  ...  \
3588  johnr005  0.2252  100   37   12406  3346  1513  411  1497  4875  ...

      BB/9  HR/9  BABIP  LOB%   ERA   FIP    WAR  Pitching  -ERA  -FIP
3588  3.26  0.89  0.291  74.7  3.29  3.19  110.4    26.051 -3.29 -3.19

[1 rows x 33 columns]
```

[138]:
```
df[df['retroID'] == 'bumgm001']
```

[138]:
```
      retroID   BAOpp  CG  SHO  IPouts     H   ER   HR   BB    SO  ...  BB/9  \
942  bumgm001  0.2358  15    6    5538  1622  642  192  428  1794  ...  2.09

     HR/9  BABIP  LOB%   ERA   FIP   WAR  Pitching  -ERA  -FIP
942  0.94  0.284  76.4  3.13  3.32  31.3     6.272 -3.13 -3.32

[1 rows x 33 columns]
```

[139]:
```
df[df['retroID'] == 'mahop001']
```

[139]:
```
       retroID   BAOpp  CG  SHO  IPouts    H   ER   HR   BB   SO  ...  BB/9  \
4377  mahop001  0.2751   0    0    2127  738  431  116  392  452  ...  4.98

      HR/9  BABIP  LOB%   ERA   FIP  WAR  Pitching  -ERA  -FIP
4377  1.47  0.284  69.5  5.47  5.62 -3.0    -3.487 -5.47 -5.62
```

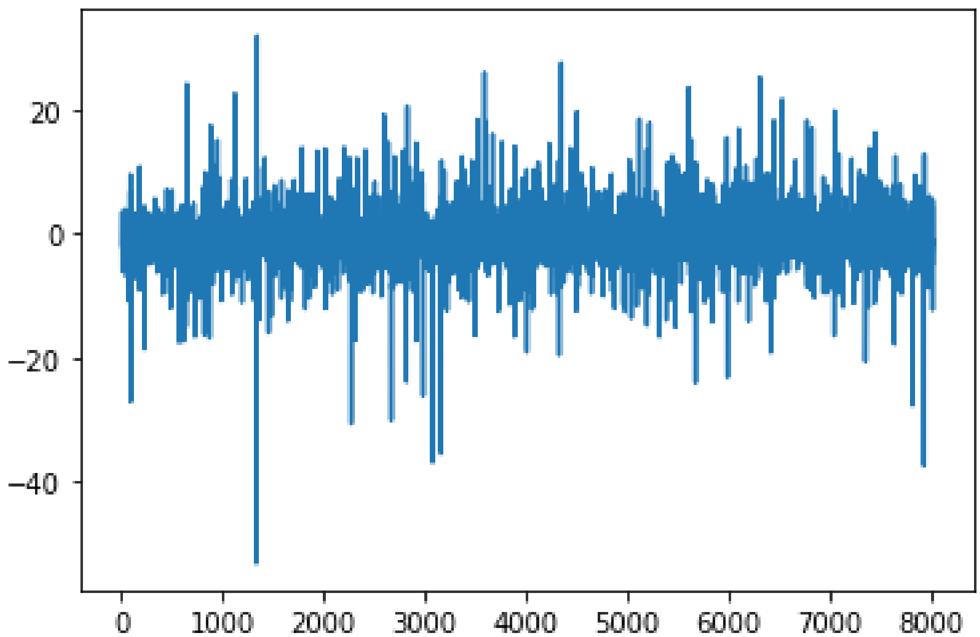
```
[1 rows x 33 columns]
```

[140]:
```python
df[df['retroID'] == 'coleg001']
```

[140]:
```
      retroID   BAOpp  CG  SHO  IPouts     H   ER   HR   BB    SO  ...  BB/9  \
1383  coleg001  0.2381   2    1    3585  1034  427  115  315  1336  ...  2.37

      HR/9  BABIP  LOB%   ERA   FIP   WAR  Pitching   -ERA  -FIP
1383  0.87  0.303  75.6  3.22  3.06  28.8     5.699  -3.22 -3.06

[1 rows x 33 columns]
```

[141]:
```python
plt.plot(df['Pitching'])
```

[141]:
```
[<matplotlib.lines.Line2D at 0x12a1537d0>]
```



[131]:
```python
df.sort_values('Pitching').head(10)
```

[131]:
```
      retroID  BAOpp  CG  SHO  IPouts  H  ER  HR  BB  SO  ...   BB/9  HR/9  \
1333  cleaj101   0.83   0    0       1  5   7   0   3   1  ...   81.0   0.0
7932  wurmf101   0.50   0    0       1  1   4   0   5   1  ...  135.0   0.0
3074  heart001   0.75   0    0       1  3   4   0   4   0  ...  108.0   0.0
3159  herne001   0.50   0    0       1  1   3   1   2   0  ...   54.0  27.0
2272  fishf101   0.66   0    0       1  2   4   0   2   1  ...   54.0   0.0
2666  gomec002   0.00   0    0       1  0   3   0   4   0  ...  108.0   0.0
7819  wilst104   0.00   0    0       1  0   3   0   2   0  ...   54.0   0.0
```

```
88     alexm001    0.50   0     0        2  1    5    1    4    0  ...    54.0   13.5
2981   harll101    0.50   0     0        2  2    5    1    4    1  ...    54.0   13.5
5674   pickr001    0.60   0     0        2  3    6    0    4    2  ...    54.0    0.0

      BABIP  LOB%    ERA    FIP  WAR  Pitching   -ERA   -FIP
1333   1.00  12.5  189.0  23.54 -0.1   -53.132 -189.0 -23.54
7932   1.00  33.3  108.0  41.54 -0.1   -37.374 -108.0 -41.54
3074   0.75  28.6  108.0  39.21 -0.1   -36.828 -108.0 -39.21
3159   0.00   0.0   81.0  60.16 -0.3   -35.365  -81.0 -60.16
2272   1.00   0.0  108.0  14.60  0.0   -30.600 -108.0 -14.60
2666   0.00  25.0   81.0  39.16 -0.1   -30.065  -81.0 -39.16
7819   0.00   0.0   81.0  29.57 -0.1   -27.667  -81.0 -29.57
88     0.00   0.0   67.5  40.67 -0.1   -27.068  -67.5 -40.67
2981   0.50  21.7   67.5  37.08 -0.1   -26.139  -67.5 -37.08
5674   1.00  14.3   81.0  15.14  0.0   -23.979  -81.0 -15.14

[10 rows x 33 columns]
```

I'm happy with this. The new Pitching stat somewhat reflects its component parts but doesn't immediately align with WAR. We don't just want to recreate WAR so that's a good thing.

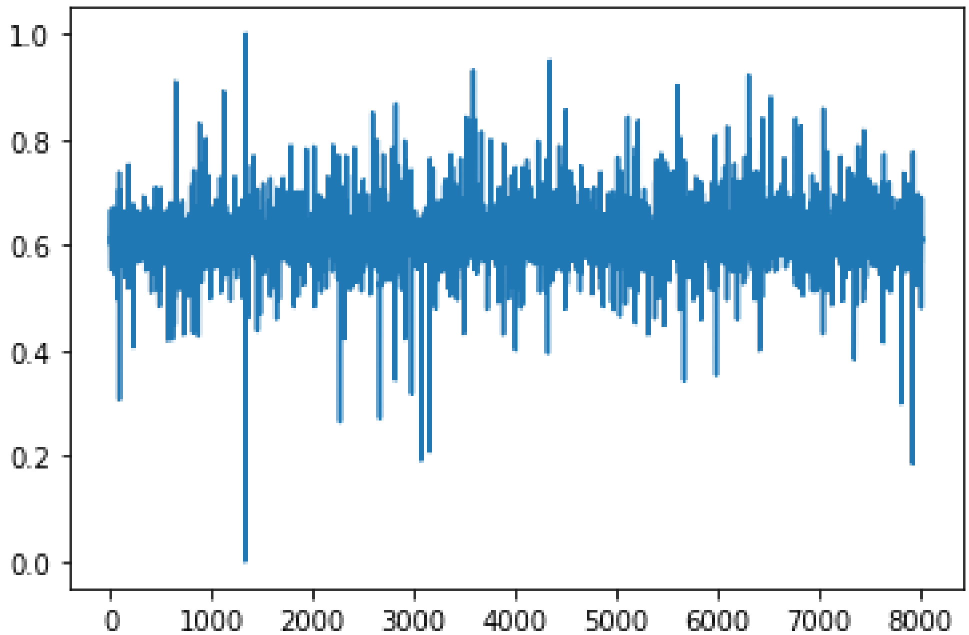
Normalization

[142]:
```
from sklearn.preprocessing import MinMaxScaler
```

[143]:
```
scaler = MinMaxScaler()
```

[144]:
```
plt.plot(scaler.fit_transform(df[['Pitching']]))
```

[144]:
```
[<matplotlib.lines.Line2D at 0x1337d3f50>]
```



[145]:
```
df['Pitching'] = scaler.fit_transform(df[['Pitching']])
```

[147]:
```
df.sort_values('Pitching').head(10)
```

[147]:
```
       retroID  BAOpp  CG  SHO  IPouts  H  ER  HR  BB  SO  ...   BB/9  HR/9  \
1333  cleaj101   0.83   0    0       1  5   7   0   3   1  ...   81.0   0.0
7932  wurmf101   0.50   0    0       1  1   4   0   5   1  ...  135.0   0.0
3074  heart001   0.75   0    0       1  3   4   0   4   0  ...  108.0   0.0
3159  herne001   0.50   0    0       1  1   3   1   2   0  ...   54.0  27.0
2272  fishf101   0.66   0    0       1  2   4   0   2   1  ...   54.0   0.0
2666  gomec002   0.00   0    0       1  0   3   0   4   0  ...  108.0   0.0
7819  wilst104   0.00   0    0       1  0   3   0   2   0  ...   54.0   0.0
88    alexm001   0.50   0    0       2  1   5   1   4   0  ...   54.0  13.5
2981  harll101   0.50   0    0       2  2   5   1   4   1  ...   54.0  13.5
5674  pickr001   0.60   0    0       2  3   6   0   4   2  ...   54.0   0.0

      BABIP  LOB%    ERA    FIP  WAR  Pitching   -ERA   -FIP
1333   1.00  12.5  189.0  23.54 -0.1  0.000000 -189.0 -23.54
7932   1.00  33.3  108.0  41.54 -0.1  0.185253 -108.0 -41.54
3074   0.75  28.6  108.0  39.21 -0.1  0.191672 -108.0 -39.21
3159   0.00   0.0   81.0  60.16 -0.3  0.208871  -81.0 -60.16
2272   1.00   0.0  108.0  14.60  0.0  0.264889 -108.0 -14.60
2666   0.00  25.0   81.0  39.16 -0.1  0.271179  -81.0 -39.16
7819   0.00   0.0   81.0  29.57 -0.1  0.299370  -81.0 -29.57
88     0.00   0.0   67.5  40.67 -0.1  0.306412  -67.5 -40.67
2981   0.50  21.7   67.5  37.08 -0.1  0.317333  -67.5 -37.08
```

```
5674   1.00  14.3   81.0  15.14  0.0  0.342726  -81.0 -15.14

[10 rows x 33 columns]
```

[148]:

```
df.sort_values('Pitching').tail(10)
```

[148]:

```
       retroID   BAOpp   CG  SHO  IPouts     H    ER   HR    BB    SO  ...  \
7049  suttd001  0.2376  178   58   15847  4692  1914  472  1343  3574  ...
2824  grovl101  0.2535  298   35   11822  3849  1339  162  1187  2266  ...
6529  seavt001  0.2285  231   61   14348  3971  1521  380  1390  3640  ...
1121  carls001  0.2546  254   55   15652  4672  1864  414  1833  4136  ...
5607  perrg101  0.2540  303   53   16051  4938  1846  399  1379  3534  ...
645   blylb001  0.2487  242   60   14910  4632  1830  430  1322  3701  ...
6311  ryann001  0.2088  222   61   16158  3923  1911  321  2795  5714  ...
3588  johnr005  0.2252  100   37   12406  3346  1513  411  1497  4875  ...
4347  maddg002  0.2553  109   35   15025  4726  1756  353   999  3371  ...
1341  clemr001  0.2308  118   46   14750  4185  1707  363  1580  4672  ...

      BB/9  HR/9  BABIP  LOB%   ERA   FIP    WAR  Pitching  -ERA  -FIP
7049  2.29  0.80  0.261  73.6  3.26  3.24   85.9  0.858468 -3.26 -3.24
2824  2.71  0.37  0.284  71.8  3.06  3.36   88.8  0.867144 -3.06 -3.36
6529  2.62  0.72  0.259  76.7  2.86  3.04   92.7  0.880287 -2.86 -3.04
1121  3.16  0.71  0.279  74.1  3.22  3.15   96.9  0.891256 -3.22 -3.15
5607  2.32  0.67  0.275  73.3  3.11  3.06  100.5  0.902342 -3.11 -3.06
645   2.39  0.78  0.282  74.1  3.31  3.19  103.3  0.909654 -3.31 -3.19
6311  4.67  0.54  0.265  73.1  3.19  2.97  107.2  0.922327 -3.19 -2.97
3588  3.26  0.89  0.291  74.7  3.29  3.19  110.4  0.930886 -3.29 -3.19
4347  1.80  0.63  0.281  72.3  3.16  3.26  116.7  0.949225 -3.16 -3.26
1341  2.89  0.66  0.284  74.6  3.12  3.09  133.7  1.000000 -3.12 -3.09

[10 rows x 33 columns]
```

Finally, we should get rid of the -ERA and -FIP columns.

[149]:

```
df = df.drop(columns=['-ERA', '-FIP'])
```

[150]:

```
df
```

[150]:

```
       retroID   BAOpp  CG  SHO  IPouts     H   ER   HR   BB   SO  ...  \
0     aardd001  0.2574   0    0    1011   296  160   41  183  340  ...
1     aased001  0.2508  22    5    3328  1085  468   89  457  641  ...
2     abadf001  0.2447   0    0     992   309  135   42  116  280  ...
3     abbog001  0.2786  37    5    3858  1405  627  162  352  484  ...
4     abboj001  0.2804  31    6    5022  1779  791  154  620  888  ...
...        ...     ...  ..  ...     ...   ...  ...  ...  ...  ...  ...
8020  zolds101  0.2700  30    5    2788   956  366   54  301  207  ...
8021  zubeb101  0.2717  23    3    2358   767  374   35  468  383  ...
```

```
8022  zumaj001  0.2286   0    0     629   169   71   18  114  210  ...
8023  zuveg101  0.2760   9    2    1927   660  253   56  203  223  ...
8024  zycht001  0.2183   0    0     218    57   22    3   34   80  ...

            IP   K/9  BB/9  HR/9  BABIP  LOB%   ERA   FIP   WAR  Pitching
0     0.062360  9.08  4.89  1.09  0.285  74.5  4.27  4.45   1.1  0.602913
1     0.205233  5.20  3.71  0.72  0.282  73.4  3.80  3.85  11.7  0.636924
2     0.061102  7.62  3.16  1.14  0.281  77.7  3.67  4.24   0.6  0.603736
3     0.237967  3.39  2.46  1.13  0.278  69.3  4.39  4.46  10.2  0.628847
4     0.309765  4.77  3.33  0.83  0.295  70.0  4.25  4.25  22.7  0.666725
...        ...   ...   ...   ...    ...   ...   ...   ...   ...       ...
8020  0.171925  2.00  2.91  0.52  0.267  70.7  3.54  3.80   9.3  0.630540
8021  0.145445  4.39  5.36  0.40  0.283  69.0  4.28  3.96   3.3  0.610437
8022  0.038711  9.01  4.89  0.77  0.267  78.7  3.00  3.94   2.7  0.612847
8023  0.118817  3.12  2.84  0.78  0.270  73.2  3.54  3.93   1.9  0.608497
8024  0.013360  9.91  4.21  0.37  0.293  79.1  2.72  3.22   1.1  0.611166

[8025 rows x 31 columns]
```

```
[ ]:
```

teams_pre

March 9, 2020

[4]:
```
import math
import numpy as np
import pandas as pd
```

[5]:
```
df = pd.read_csv('../data/lahman/mlb_data/Teams.csv')
```

[6]:
```
df.columns
```

[6]:
```
Index(['yearID', 'lgID', 'teamID', 'franchID', 'divID', 'Rank', 'G', 'Ghome',
       'W', 'L', 'DivWin', 'WCWin', 'LgWin', 'WSWin', 'R', 'AB', 'H', '2B',
       '3B', 'HR', 'BB', 'SO', 'SB', 'CS', 'HBP', 'SF', 'RA', 'ER', 'ERA',
       'CG', 'SHO', 'SV', 'IPouts', 'HA', 'HRA', 'BBA', 'SOA', 'E', 'DP', 'FP',
       'name', 'park', 'attendance', 'BPF', 'PPF', 'teamIDBR',
       'teamIDlahman45', 'teamIDretro'],
      dtype='object')
```

[7]:
```
df.head()
```

[7]:
```
   yearID lgID teamID franchID divID  Rank    G  Ghome   W   L  ...   DP  \
0    1919   AL    BOS      BOS   NaN     6  138     66  66  71  ...  118
1    1919   NL    BRO      LAD   NaN     5  141     70  69  71  ...   84
2    1919   NL    BSN      ATL   NaN     6  140     68  57  82  ...  111
3    1919   AL    CHA      CHW   NaN     1  140     70  88  52  ...  116
4    1919   NL    CHN      CHC   NaN     3  140     71  75  65  ...   87

      FP               name          park  attendance  BPF  PPF  teamIDBR  \
0  0.975     Boston Red Sox  Fenway Park I      417291   94   94       BOS
1  0.963    Brooklyn Robins   Ebbets Field      360721  103  103       BRO
2  0.966      Boston Braves   Braves Field      167401   95   98       BSN
3  0.969  Chicago White Sox  Comiskey Park      627186  100   99       CHW
4  0.969       Chicago Cubs  Wrigley Field      424430  100   99       CHC

  teamIDlahman45 teamIDretro
0            BOS         BOS
1            BRO         BRO
2            BSN         BSN
3            CHA         CHA
```

```
4              CHN           CHN

[5 rows x 48 columns]
```

[8]:
```
df = df.drop(columns=['teamIDlahman45', 'teamIDBR'])
```

The first step is to ensure we're only using one ID per team. It would be best to just use Retrosheet's values, so our first step is to see where teamID differs from teamIDretro. Once we come up with a way to fix these differences, we'll want to write it as a script that we can use elsewhere - for example, in the batting table where we're using the regular teamID values.

[9]:
```
df[(df['teamID'] != df['teamIDretro'])][['yearID', 'teamID', 'teamIDretro',␣
↪'name']]
```

[9]:
```
      yearID teamID teamIDretro                name
551     1953    ML1         MLN    Milwaukee Braves
568     1954    ML1         MLN    Milwaukee Braves
585     1955    ML1         MLN    Milwaukee Braves
601     1956    ML1         MLN    Milwaukee Braves
617     1957    ML1         MLN    Milwaukee Braves
633     1958    ML1         MLN    Milwaukee Braves
649     1959    ML1         MLN    Milwaukee Braves
665     1960    ML1         MLN    Milwaukee Braves
683     1961    ML1         MLN    Milwaukee Braves
702     1962    ML1         MLN    Milwaukee Braves
722     1963    ML1         MLN    Milwaukee Braves
742     1964    ML1         MLN    Milwaukee Braves
762     1965    ML1         MLN    Milwaukee Braves
867     1970    ML4         MIL   Milwaukee Brewers
891     1971    ML4         MIL   Milwaukee Brewers
915     1972    ML4         MIL   Milwaukee Brewers
939     1973    ML4         MIL   Milwaukee Brewers
963     1974    ML4         MIL   Milwaukee Brewers
987     1975    ML4         MIL   Milwaukee Brewers
1011    1976    ML4         MIL   Milwaukee Brewers
1035    1977    ML4         MIL   Milwaukee Brewers
1061    1978    ML4         MIL   Milwaukee Brewers
1087    1979    ML4         MIL   Milwaukee Brewers
1113    1980    ML4         MIL   Milwaukee Brewers
1139    1981    ML4         MIL   Milwaukee Brewers
1165    1982    ML4         MIL   Milwaukee Brewers
1191    1983    ML4         MIL   Milwaukee Brewers
1217    1984    ML4         MIL   Milwaukee Brewers
1243    1985    ML4         MIL   Milwaukee Brewers
1269    1986    ML4         MIL   Milwaukee Brewers
1295    1987    ML4         MIL   Milwaukee Brewers
1321    1988    ML4         MIL   Milwaukee Brewers
```

```
1347  1989  ML4     MIL            Milwaukee Brewers
1373  1990  ML4     MIL            Milwaukee Brewers
1399  1991  ML4     MIL            Milwaukee Brewers
1425  1992  ML4     MIL            Milwaukee Brewers
1453  1993  ML4     MIL            Milwaukee Brewers
1481  1994  ML4     MIL            Milwaukee Brewers
1509  1995  ML4     MIL            Milwaukee Brewers
1537  1996  ML4     MIL            Milwaukee Brewers
1565  1997  ML4     MIL            Milwaukee Brewers
1801  2005  LAA     ANA  Los Angeles Angels of Anaheim
1831  2006  LAA     ANA  Los Angeles Angels of Anaheim
1861  2007  LAA     ANA  Los Angeles Angels of Anaheim
1891  2008  LAA     ANA  Los Angeles Angels of Anaheim
1921  2009  LAA     ANA  Los Angeles Angels of Anaheim
1951  2010  LAA     ANA  Los Angeles Angels of Anaheim
1981  2011  LAA     ANA  Los Angeles Angels of Anaheim
2010  2012  LAA     ANA  Los Angeles Angels of Anaheim
2040  2013  LAA     ANA  Los Angeles Angels of Anaheim
2070  2014  LAA     ANA  Los Angeles Angels of Anaheim
2100  2015  LAA     ANA  Los Angeles Angels of Anaheim
2130  2016  LAA     ANA  Los Angeles Angels of Anaheim
2160  2017  LAA     ANA  Los Angeles Angels of Anaheim
2190  2018  LAA     ANA  Los Angeles Angels of Anaheim
```

So clearly we have three teams where the IDs differ. We need to ask a few questions though:

Do they differ on those teams every time? We can’t just take that for granted.

[10]:
```
df[df['franchID'] == 'ANA']['teamID'].value_counts()
```

[10]:
```
CAL    32
LAA    18
ANA     8
Name: teamID, dtype: int64
```

[11]:
```
df[(df['teamID'] != df['teamIDretro'])][['teamID', 'teamIDretro', 'name']].
 ↪shape[0]
```

[11]: 55

[12]:
```
df[(df['teamID'] == 'ML1')].shape[0] + df[(df['teamID'] == 'ML4')].shape[0] +␣
 ↪df[(df['teamID'] == 'LAA')].shape[0]
```

[12]: 59

Unfortunately we have a disparity of 4, so we need to find out where that is.

[13]:
```
df[(df['teamID'] == 'ML1') & (df['teamID'] == df['teamIDretro'])]
```

[13]:
```
Empty DataFrame
Columns: [yearID, lgID, teamID, franchID, divID, Rank, G, Ghome, W, L, DivWin,
WCWin, LgWin, WSWin, R, AB, H, 2B, 3B, HR, BB, SO, SB, CS, HBP, SF, RA, ER, ERA,
CG, SHO, SV, IPouts, HA, HRA, BBA, SOA, E, DP, FP, name, park, attendance, BPF,
PPF, teamIDretro]
Index: []

[0 rows x 46 columns]
```

[14]:
```python
df[(df['teamID'] == 'ML4') & (df['teamID'] == df['teamIDretro'])]
```

[14]:
```
Empty DataFrame
Columns: [yearID, lgID, teamID, franchID, divID, Rank, G, Ghome, W, L, DivWin,
WCWin, LgWin, WSWin, R, AB, H, 2B, 3B, HR, BB, SO, SB, CS, HBP, SF, RA, ER, ERA,
CG, SHO, SV, IPouts, HA, HRA, BBA, SOA, E, DP, FP, name, park, attendance, BPF,
PPF, teamIDretro]
Index: []

[0 rows x 46 columns]
```

[15]:
```python
df[(df['teamID'] == 'LAA') & (df['teamID'] == df['teamIDretro'])]
```

[15]:
```
     yearID lgID teamID franchID divID  Rank    G  Ghome   W   L  ...  SOA  \
680    1961   AL    LAA      ANA   NaN     8  162     82  70  91  ...  973
699    1962   AL    LAA      ANA   NaN     3  162     81  86  76  ...  858
719    1963   AL    LAA      ANA   NaN     9  161     81  70  91  ...  889
739    1964   AL    LAA      ANA   NaN     5  162     81  82  80  ...  965

       E   DP     FP                 name                park  attendance  BPF  \
680  192  154  0.969  Los Angeles Angels  Wrigley Field (LA)      603510  111
699  175  153  0.972  Los Angeles Angels      Dodger Stadium     1144063   97
719  163  155  0.974  Los Angeles Angels      Dodger Stadium      821015   94
739  138  168  0.978  Los Angeles Angels      Dodger Stadium      760439   90

     PPF  teamIDretro
680  112          LAA
699   97          LAA
719   94          LAA
739   90          LAA

[4 rows x 46 columns]
```

[16]:
```python
df[(df['teamID'] == 'LAA') & (df['teamID'] != df['teamIDretro'])]
```

[16]:
```
      yearID lgID teamID franchID divID  Rank    G  Ghome   W   L  ...   SOA  \
1801    2005   AL    LAA      ANA     W     1  162     81  95  67  ...  1126
1831    2006   AL    LAA      ANA     W     2  162     81  89  73  ...  1164
```

```
1861    2007   AL    LAA      ANA      W      1  162      81   94  68  ...  1156
1891    2008   AL    LAA      ANA      W      1  162      81  100  62  ...  1106
1921    2009   AL    LAA      ANA      W      1  162      81   97  65  ...  1062
1951    2010   AL    LAA      ANA      W      3  162      81   80  82  ...  1130
1981    2011   AL    LAA      ANA      W      2  162      81   86  76  ...  1058
2010    2012   AL    LAA      ANA      W      3  162      81   89  73  ...  1157
2040    2013   AL    LAA      ANA      W      3  162      81   78  84  ...  1200
2070    2014   AL    LAA      ANA      W      1  162      81   98  64  ...  1342
2100    2015   AL    LAA      ANA      W      3  162      81   85  77  ...  1221
2130    2016   AL    LAA      ANA      W      4  162      81   74  88  ...  1136
2160    2017   AL    LAA      ANA      W      2  162      81   80  82  ...  1312
2190    2018   AL    LAA      ANA      W      4  162      81   80  82  ...  1386

        E   DP     FP                           name  \
1801   87  139  0.986  Los Angeles Angels of Anaheim
1831  124  154  0.979  Los Angeles Angels of Anaheim
1861  101  154  0.983  Los Angeles Angels of Anaheim
1891   91  159  0.985  Los Angeles Angels of Anaheim
1921   85  174  0.986  Los Angeles Angels of Anaheim
1951  113  116  0.981  Los Angeles Angels of Anaheim
1981   93  157  0.985  Los Angeles Angels of Anaheim
2010   98  141  0.984  Los Angeles Angels of Anaheim
2040  112  135  0.981  Los Angeles Angels of Anaheim
2070   83  127  0.986  Los Angeles Angels of Anaheim
2100   93  108  0.984  Los Angeles Angels of Anaheim
2130   97  148  0.983  Los Angeles Angels of Anaheim
2160   80  135  0.986  Los Angeles Angels of Anaheim
2190   76  173  0.987  Los Angeles Angels of Anaheim

                          park  attendance  BPF  PPF  teamIDretro
1801             Angel Stadium     3404686   98   97          ANA
1831             Angel Stadium     3406790  100  100          ANA
1861             Angel Stadium     3365632  101  100          ANA
1891             Angel Stadium     3336747  103  102          ANA
1921             Angel Stadium     3240386   99   98          ANA
1951             Angel Stadium     3250816   98   98          ANA
1981             Angel Stadium     3166321   93   93          ANA
2010  Angel Stadium of Anaheim     3061770   92   92          ANA
2040  Angel Stadium of Anaheim     3019505   94   94          ANA
2070  Angel Stadium of Anaheim     3095935   96   95          ANA
2100  Angel Stadium of Anaheim     3012765   94   95          ANA
2130  Angel Stadium of Anaheim     3016142   95   95          ANA
2160  Angel Stadium of Anaheim     3019585   96   96          ANA
2190  Angel Stadium of Anaheim     3020216   97   97          ANA

[14 rows x 46 columns]
```

[17]:
```
df['franchID'].unique()
```

[17]:
```
array(['BOS', 'LAD', 'ATL', 'CHW', 'CHC', 'CIN', 'CLE', 'DET', 'SFG',
       'NYY', 'OAK', 'PHI', 'PIT', 'BAL', 'STL', 'MIN', 'ANA', 'TEX',
       'HOU', 'NYM', 'KCR', 'WSN', 'SDP', 'MIL', 'SEA', 'TOR', 'COL',
       'FLA', 'ARI', 'TBD'], dtype=object)
```

[18]:
```
df[(df['franchID'].isnull())]
```

[18]:
```
Empty DataFrame
Columns: [yearID, lgID, teamID, franchID, divID, Rank, G, Ghome, W, L, DivWin,
WCWin, LgWin, WSWin, R, AB, H, 2B, 3B, HR, BB, SO, SB, CS, HBP, SF, RA, ER, ERA,
CG, SHO, SV, IPouts, HA, HRA, BBA, SOA, E, DP, FP, name, park, attendance, BPF,
PPF, teamIDretro]
Index: []

[0 rows x 46 columns]
```

[19]:
```
df['franchID'].nunique()
```

[19]:
```
30
```

It looks like it will be easiest to just use the franchise ID - they stay consistent throughout and there are only ever 30 max. We'll need a way to map to these values from an external script so we can use it in other files.

Building Tables and Tensors

Aggregating preprocessed data and compiling it into tables that are ready to be read into the models as tensors

build_tables

March 28, 2020

[88]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Building the Tables

We've done the major preprocessing in other scripts, and now it's time to get our final tables together for fielders, catchers and pitchers with all appropriate stats.

[89]:
```
df_bat = pd.read_csv('../core/output/batting_pre.csv')
df_field = pd.read_csv('../core/output/fielding_pre.csv')
df_catch = pd.read_csv('../core/output/catching_pre.csv')
df_pitch = pd.read_csv('../core/output/pitching_pre.csv')
df_meta = pd.read_csv('../core/output/metadata.csv')
```

[90]:
```
df_meta.head()
```

[90]:

	retroID	POS	birthYear	bats	throws	weight	height	debutYear	finalYear
0	aardd001	P	1981.0	R	R	215.0	75.0	2004	2015
1	aaroh101	OF	1934.0	R	R	180.0	72.0	1954	1976
2	aarot101	1B	1939.0	R	R	190.0	75.0	1962	1971
3	aased001	P	1954.0	R	R	190.0	75.0	1977	1990
4	abada001	1B	1972.0	L	L	184.0	73.0	2001	2006

Making Metadata Usable

We are interested in all of these fields, so we want to convert POS, bats and throws to numbers and use dummy variables. Note that we won't be using birthYear as-is, but rather subtracting it from current year to get a player's age for a season. This won't matter for the player career stats tensor so we can drop it here.

[91]:
```
df_meta.drop(columns=['birthYear'], inplace=True)
```

[92]:
```
df_meta_pos = pd.get_dummies(df_meta['POS'], prefix='pos')
df_meta_bats = pd.get_dummies(df_meta['bats'], drop_first=True, prefix='bats')
df_meta_throws = pd.get_dummies(df_meta['throws'], prefix='throws')
```

[93]:
```
dropped_meta_cols = ['POS', 'bats', 'throws']
df_meta.drop(columns=dropped_meta_cols, inplace=True)
```

```
df_meta.head()
```

[93]:
```
    retroID  weight  height  debutYear  finalYear
0  aardd001   215.0    75.0       2004       2015
1  aaroh101   180.0    72.0       1954       1976
2  aarot101   190.0    75.0       1962       1971
3  aased001   190.0    75.0       1977       1990
4  abada001   184.0    73.0       2001       2006
```

[94]:
```
df_meta = df_meta.join([df_meta_pos, df_meta_bats, df_meta_throws])
df_meta
```

[94]:
```
        retroID  weight  height  debutYear  finalYear  pos_1B  pos_2B  pos_3B  \
0      aardd001   215.0    75.0       2004       2015       0       0       0
1      aaroh101   180.0    72.0       1954       1976       0       0       0
2      aarot101   190.0    75.0       1962       1971       1       0       0
3      aased001   190.0    75.0       1977       1990       0       0       0
4      abada001   184.0    73.0       2001       2006       1       0       0
...         ...     ...     ...        ...        ...     ...     ...     ...
15026  zupcb001   220.0    76.0       1991       1994       0       0       0
15027  zupof101   182.0    71.0       1957       1961       0       0       0
15028  zuveg101   195.0    76.0       1951       1959       0       0       0
15029  zuvep001   173.0    72.0       1982       1991       0       0       0
15030  zycht001   190.0    75.0       2015       2017       0       0       0

       pos_C  pos_OF  pos_P  pos_SS  bats_L  bats_R  throws_L  throws_R  \
0          0       0      1       0       0       1         0         1
1          0       1      0       0       0       1         0         1
2          0       0      0       0       0       1         0         1
3          0       0      1       0       0       1         0         1
4          0       0      0       0       1       0         1         0
...      ...     ...    ...     ...     ...     ...       ...       ...
15026      0       1      0       0       0       1         0         1
15027      1       0      0       0       1       0         0         1
15028      0       0      1       0       0       1         0         1
15029      0       0      0       1       0       1         0         1
15030      0       0      1       0       0       1         0         1

       throws_S
0             0
1             0
2             0
3             0
4             0
...         ...
15026         0
15027         0
```

```
15028          0
15029          0
15030          0

[15031 rows x 17 columns]
```

I didn't drop_first on the 'throws_' columns because I want to get rid of 'throws_S' instead of 'throws_L'

[95]:
```
df_meta.drop(columns=['throws_S'], inplace=True)
```

We want to use weight and height but we can normalize them

[96]:
```
from sklearn.preprocessing import MinMaxScaler
```

[97]:
```
scaler = MinMaxScaler()
```

[98]:
```
df_meta[['weight', 'height']] = scaler.fit_transform(df_meta[['weight',␣
↪'height']])
df_meta
```

[98]:
```
        retroID    weight  height  debutYear  finalYear  pos_1B  pos_2B  \
0      aardd001  0.569672    0.60       2004       2015       0       0
1      aaroh101  0.426230    0.45       1954       1976       0       0
2      aarot101  0.467213    0.60       1962       1971       1       0
3      aased001  0.467213    0.60       1977       1990       0       0
4      abada001  0.442623    0.50       2001       2006       1       0
...         ...       ...     ...        ...        ...     ...     ...
15026  zupcb001  0.590164    0.65       1991       1994       0       0
15027  zupof101  0.434426    0.40       1957       1961       0       0
15028  zuveg101  0.487705    0.65       1951       1959       0       0
15029  zuvep001  0.397541    0.45       1982       1991       0       0
15030  zycht001  0.467213    0.60       2015       2017       0       0

       pos_3B  pos_C  pos_OF  pos_P  pos_SS  bats_L  bats_R  throws_L  \
0           0      0       0      1       0       0       1         0
1           0      0       1      0       0       0       1         0
2           0      0       0      0       0       0       1         0
3           0      0       0      1       0       0       1         0
4           0      0       0      0       0       1       0         1
...       ...    ...     ...    ...     ...     ...     ...       ...
15026       0      0       1      0       0       0       1         0
15027       0      1       0      0       0       1       0         0
15028       0      0       0      1       0       0       1         0
15029       0      0       0      0       1       0       1         0
15030       0      0       0      1       0       0       1         0
```

```
       throws_R
0             1
1             1
2             1
3             1
4             0
...         ...
15026         1
15027         1
15028         1
15029         1
15030         1

[15031 rows x 16 columns]
```

The metadata is now ready to go into the final tensor.

Combining Batting Data

[99]:
```
df_bat
```

[99]:
```
        retroID     G     AB     R     H   2B  3B   HR   RBI   SB    CS    BB  \
0      aardd001   331      4     0     0    0   0    0     0    0   0.0     0
1      aaroh101  3298  12364  2174  3771  624  98  755  2297  240  73.0  1402
2      aarot101   437    944   102   216   42   6   13    94    9   8.0    86
3      aased001   448      5     0     0    0   0    0     0    0   0.0     0
4      abada001    15     21     1     2    0   0    0     0    0   1.0     4
...         ...   ...    ...   ...   ...  ...  ..  ...   ...  ...   ...   ...
15187  zupcb001   319    795    99   199   47   4    7    80    7   5.0    57
15188  zupof101    16     18     3     3    1   0    0     0    0   0.0     2
15189  zuveg101   266    142     5    21    2   1    0     7    0   1.0     9
15190  zuvep001   209    491    41   109   17   2    2    20    2   0.0    34
15191  zycht001    70      0     0     0    0   0    0     0    0   0.0     0

         SO  IBB  HBP  SH   SF  GIDP  NL
0         2    0    0   1    0     0   1
1      1383  293   32  21  121   328   1
2       145    3    0   9    6    36   1
3         3    0    0   0    0     0   1
4         5    0    0   0    0     1   1
...     ...  ...  ...  ..  ...   ...  ..
15187   137    3    6  20    8    15   0
15188     6    0    0   0    0     0   0
15189    39    0    0  16    0     3   1
15190    50    1    2  18    0     8   1
15191     0    0    0   0    0     0   0

[15192 rows x 19 columns]
```

[100]:
```
df = pd.merge(df_meta, df_bat, how='inner', on=['retroID'])
```

[187]:
```
df.shape
```

[187]:
```
(15031, 34)
```

[101]:
```
df.head(10)
```

[101]:
```
    retroID    weight  height  debutYear  finalYear  pos_1B  pos_2B  pos_3B  \
0  aardd001  0.569672    0.60       2004       2015       0       0       0
1  aaroh101  0.426230    0.45       1954       1976       0       0       0
2  aarot101  0.467213    0.60       1962       1971       1       0       0
3  aased001  0.467213    0.60       1977       1990       0       0       0
4  abada001  0.442623    0.50       2001       2006       1       0       0
5  abadf001  0.590164    0.50       2010       2017       0       0       0
6  abbog001  0.508197    0.75       1973       1984       0       0       0
7  abboj001  0.508197    0.60       1989       1999       0       0       0
8  abboj002  0.467213    0.55       1997       2001       0       0       0
9  abbok001  0.508197    0.65       1991       1996       0       0       0

   pos_C  pos_OF  ...   SB    CS    BB    SO  IBB  HBP  SH   SF  GIDP  NL
0      0       0  ...    0   0.0     0     2    0    0   1    0     0   1
1      0       1  ...  240  73.0  1402  1383  293   32  21  121   328   1
2      0       0  ...    9   8.0    86   145    3    0   9    6    36   1
3      0       0  ...    0   0.0     0     3    0    0   0    0     0   1
4      0       0  ...    0   1.0     4     5    0    0   0    0     1   1
5      0       0  ...    0   0.0     0     5    0    0   0    0     1   1
6      0       0  ...    0   0.0     0     0    0    0   0    0     0   0
7      0       0  ...    0   0.0     0    10    0    0   3    0     0   1
8      0       1  ...    6   5.0    38    91    2    3   5    7    12   1
9      0       0  ...    0   0.0     1    19    0    0   6    0     0   1

[10 rows x 34 columns]
```

We noticed that df_bat and df_meta don't have the same number of rows, so we want to find out what's going on there.

[102]:
```
df_bat[~df_bat['retroID'].isin(df_meta['retroID'])]
```

[102]:
```
        retroID  G  AB  R  H  2B  3B  HR  RBI  SB   CS  BB  SO  IBB  HBP  SH  \
121    albeb101  6  18  1  5   1   0   0    0   0  0.0   0   2    0    0   0
358    aragj101  1   0  0  0   0   0   0    0   0  0.0   0   0    0    0   0
445    atkil101  1   1  1  0   0   0   0    0   0  0.0   0   0    0    0   0
611    banij001  1   1  0  1   0   0   0    0   0  0.0   0   0    0    0   0
633    barbr101  1   1  0  0   0   0   0    0   0  0.0   0   0    0    0   0
...         ... ..  .. .. ..  ..  ..  ..  ...  ..  ...  ..  ..  ...  ...  ..
14543  westj101  1   1  0  0   0   0   0    0   0  0.0   0   1    0    0   0
```

```
14730  willh101  10   9  0  2   0   0   0    0   0  0.0   1   4    0    0   0
14811  wilsi101   1   1  0  0   0   0   0    0   0  0.0   0   0    0    0   0
15009  wrigr002   1   3  0  0   0   0   0    0   0  0.0   0   1    0    0   0
15050  yeabb101   3   0  0  0   0   0   0    0   0  0.0   1   0    0    0   0

       SF  GIDP  NL
121     0     0   0
358     0     0   1
445     0     0   0
611     0     0   1
633     0     0   0
...    ..   ...  ..
14543   0     0   1
14730   0     0   1
14811   0     0   0
15009   0     1   0
15050   0     0   1

[161 rows x 19 columns]
```

[103]:
```
df_bat[~df_bat['retroID'].isin(df_meta['retroID'])]['G'].max()
```

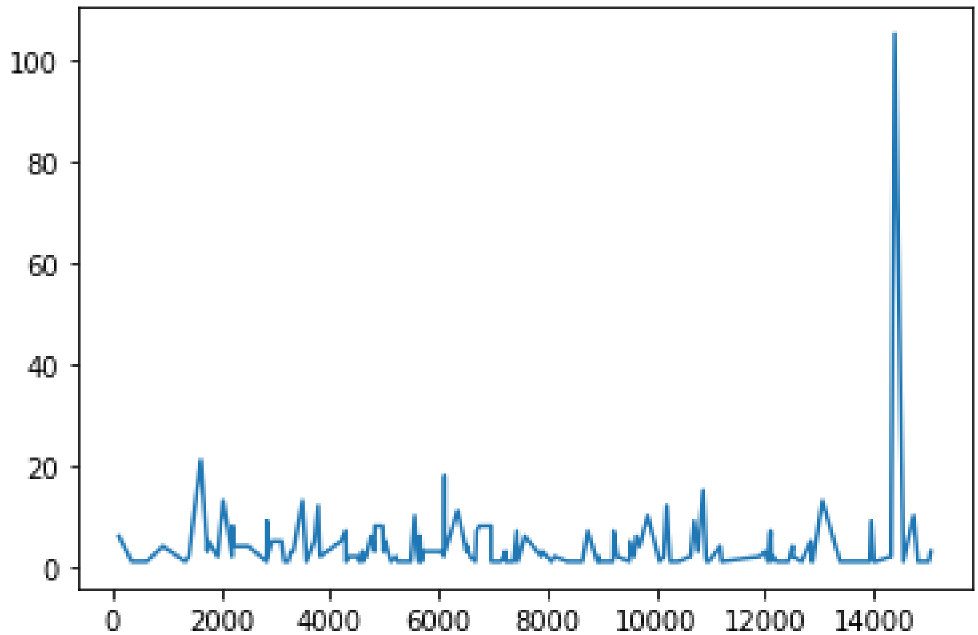
[103]: 105

[104]:
```
df_bat[~df_bat['retroID'].isin(df_meta['retroID'])]['G'].mean()
```

[104]: 4.105590062111801

[105]:
```
plt.plot(df_bat[~df_bat['retroID'].isin(df_meta['retroID'])]['G'])
```

[105]: [<matplotlib.lines.Line2D at 0x11ffb5bd0>]



For the most part, we're talking about players who have played under 20 total games. We can easily drop these data points and not really affect the overall result.

Combining the Tensors

Catchers

[106]:
```
df_catch.shape[0] + df_field.shape[0] + df_pitch.shape[0]
```

[106]:
```
23591
```

[107]:
```
df_catch
```

[107]:
```
       retroID   GS  InnOuts    PO    A   E  DP  PB  WP   SB  CS  ZR
0     adamb105    1       27     6    0   0   0   0   0    1   0   0
1     adamb106    0        0   249   90  12  15   7   0    0   0   0
2     adamd101    3       78     9    2   0   0   1   0    0   0   0
3     adled101   65     1840   453   26   4   2   8  19   37  16   0
4     afent001   20      613   123    5   1   3   6   0   17   3   0
...        ...  ...      ...   ...  ...  ..  ..  ..  ..  ...  ..  ..
1524  zimmd101   27      744   150   18   6   1   5  12   10  10   3
1525  zimmj101  298     8560  2131  150  21  26  19  84  110  80   4
1526  zinta001    0        3     2    0   0   0   0   0    0   0   0
1527  zunim001  535    14489  4356  264  21  22  39   0  248  98   0
1528  zupof101    1      114    31    1   2   0   1   1    2   1   0

[1529 rows x 12 columns]
```

[108]:
```
np.intersect1d(df_catch.columns, df.columns)
```

[108]:
```
array(['CS', 'SB', 'retroID'], dtype=object)
```

The 'caught stealing' and 'stolen bases' stats appear both offensively and defensively (CS/SB against) for catchers. We need to keep them separate when merging the metadata and we can do so by just adding a prefix to the defensive stats.

[109]:
```
df_catch.rename(columns={'CS': 'CS_A', 'SB': 'SB_A'}, inplace=True)
```

[110]:
```
catchers = pd.merge(df_catch, df, how='inner', on=['retroID'])
```

[111]:
```
catchers
```

[111]:
```
       retroID   GS  InnOuts    PO    A   E  DP  PB  WP  SB_A  ...  SB    CS  \
0     adamb105    1       27     6    0   0   0   0   0     1  ...   0   0.0
1     adamb106    0        0   249   90  12  15   7   0     0  ...   4   2.0
2     adamd101    3       78     9    2   0   0   1   0     0  ...   0   0.0
3     adled101   65     1840   453   26   4   2   8  19    37  ...   0   0.0
4     afent001   20      613   123    5   1   3   6   0    17  ...   0   0.0
...        ...  ...      ...   ...  ...  ..  ..  ..  ..   ...  ...  ..   ...
1524  zimmd101   27      744   150   18   6   1   5  12    10  ...  45  25.0
1525  zimmj101  298     8560  2131  150  21  26  19  84   110  ...   1   2.0
1526  zinta001    0        3     2    0   0   0   0   0     0  ...   0   0.0
1527  zunim001  535    14489  4356  264  21  22  39   0   248  ...   2   4.0
1528  zupof101    1      114    31    1   2   0   1   1     2  ...   0   0.0

       BB   SO  IBB  HBP  SH  SF  GIDP  NL
0       0    5    0    0   0   0     0   0
1       6   27    0    0   3   0     0   1
2       1    3    0    0   0   0     1   0
3      18   80    5    2   2   1     9   1
4       5   32    0    0   1   1     1   1
...   ...  ...  ...  ...  ..  ..   ...  ..
1524  246  678   27   13  37  14    99   1
1525   78  154   12   11  31   4    38   1
1526    5   34    0    0   0   1     0   1
1527  138  714    1   45   8  11    38   0
1528    2    6    0    0   0   0     0   0

[1529 rows x 45 columns]
```

[112]:
```
catchers.columns
```

[112]:
```
Index(['retroID', 'GS', 'InnOuts', 'PO', 'A', 'E', 'DP', 'PB', 'WP', 'SB_A',
       'CS_A', 'ZR', 'weight', 'height', 'debutYear', 'finalYear', 'pos_1B',
       'pos_2B', 'pos_3B', 'pos_C', 'pos_OF', 'pos_P', 'pos_SS', 'bats_L',
```

```
       'bats_R', 'throws_L', 'throws_R', 'G', 'AB', 'R', 'H', '2B', '3B', 'HR',
       'RBI', 'SB', 'CS', 'BB', 'SO', 'IBB', 'HBP', 'SH', 'SF', 'GIDP', 'NL'],
      dtype='object')
```

There's no reason to waste columns on position for the catchers.

[113]:
```
catchers.drop(columns=['pos_1B', 'pos_2B', 'pos_3B', 'pos_C',
       'pos_OF', 'pos_P', 'pos_SS'], inplace=True)
```

[114]:
```
catchers.columns
```

[114]:
```
Index(['retroID', 'GS', 'InnOuts', 'PO', 'A', 'E', 'DP', 'PB', 'WP', 'SB_A',
       'CS_A', 'ZR', 'weight', 'height', 'debutYear', 'finalYear', 'bats_L',
       'bats_R', 'throws_L', 'throws_R', 'G', 'AB', 'R', 'H', '2B', '3B', 'HR',
       'RBI', 'SB', 'CS', 'BB', 'SO', 'IBB', 'HBP', 'SH', 'SF', 'GIDP', 'NL'],
      dtype='object')
```

Pitchers

[147]:
```
np.intersect1d(df_pitch.columns, df.columns)
```

[147]:
```
array(['G', 'retroID'], dtype=object)
```

We have quite a few common columns for pitching data and metadata. We'll do what we did for catching and just add 'A' to the end (for 'against'). Since there are quite a few, we'll define a conversion dictionary ahead of time. Before we do that, we see that we can drop the 'G' (games) column as it should be the same between the tables. We can also drop the position information.

[148]:
```
batting_data_for_pitchers = df.drop(columns=['G', 'pos_1B', 'pos_2B',
                    'pos_3B', 'pos_C', 'pos_OF', 'pos_P', 'pos_SS'])
```

[149]:
```
pitching_data_conversion_dict = {
    'BB': 'BB_A',
    'GIDP': 'GIDP_A',
    'H': 'H_A',
    'HBP': 'HBP_A',
    'HR': 'HR_A',
    'IBB': 'IBB_A',
    'R': 'R_A',
    'SF': 'SF_A',
    'SH': 'SH_A',
    'SO': 'SO_A'
}
```

[150]:
```
df_pitch.rename(columns=pitching_data_conversion_dict, inplace=True)
```

[151]:
```
pitchers = pd.merge(df_pitch, batting_data_for_pitchers, how='inner',␣
 ↪on=['retroID'])
```

[152]:
```
pitchers
```

[152]:
```
       retroID   BAOpp     ERA   W    L    G   GS  CG  SHO  SV  ...  SB   CS  \
0     aardd001  0.2574  5.1944  16   18  331    0   0    0  69  ...   0  0.0
1     aased001  0.2508  3.4931  66   60  448   91  22    5  82  ...   0  0.0
2     abadf001  0.2501  4.0733   8   27  363    6   0    0   2  ...   0  0.0
3     abbog001  0.2786  4.3317  62   83  248  206  37    5   0  ...   0  0.0
4     abboj001  0.2804  4.4964  87  108  263  254  31    6   0  ...   0  0.0
...        ...     ...     ...  ..  ...  ...  ...  ..  ...  ..  ...  ..  ...
7830  zolds101  0.2700  3.6890  43   53  250   93  30    5   8  ...   1  0.0
7831  zubeb101  0.2717  5.3617  43   42  224   65  23    3   6  ...   0  0.0
7832  zumaj001  0.2286  3.4420  13   12  171    0   0    0   5  ...   0  0.0
7833  zuveg101  0.2760  4.1280  32   36  265   31   9    2  40  ...   0  1.0
7834  zycht001  0.2183  2.8000   7    3   70    1   0    0   1  ...   0  0.0

      BB  SO  IBB  HBP  SH  SF  GIDP  NL
0      0   2    0    0   1   0     0   1
1      0   3    0    0   0   0     0   1
2      0   5    0    0   0   0     1   1
3      0   0    0    0   0   0     0   0
4      0  10    0    0   3   0     0   1
...   ..  ..  ...  ...  ..  ..   ...  ..
7830  10  52    0    1   9   0     4   0
7831  10  66    0    0  20   0     8   0
7832   0   0    0    0   0   0     0   0
7833   9  39    0    0  16   0     3   1
7834   0   0    0    0   0   0     0   0

[7835 rows x 51 columns]
```

Fielders

[32]:
```
np.intersect1d(df_field.columns, df.columns)
```

[32]:
```
array(['retroID'], dtype=object)
```

We don't need to worry about common columns between the general fielding stats and metadata.

[136]:
```
fielders = pd.merge(df_field, df, how='inner', on=['retroID'])
```

[137]:
```
fielders = fielders[~fielders['retroID'].isin(pitchers['retroID'])]
```

[138]:
```
fielders
```

[138]:
```
         retroID    GS  InnOuts    PO    A    E   DP    weight  height  \
1       aaroh101  2977    78414  7436  429  144  218  0.426230    0.45
2       aarot101   206     6472  1317  113   22  124  0.467213    0.60
4       abada001     4      138    37    1    1    3  0.442623    0.50
```

```
8      abboj002  140   3688  299     2   8    0  0.467213    0.55
10     abbok002  504  13474  938  1262  79  275  0.426230    0.40
...         ...  ...    ...  ...   ...  ..  ...       ...     ...
14218  zoske001    8    404   16    42   2    8  0.405738    0.45
14220  zubej001   26    702  167    12   2   11  0.467213    0.50
14221  zulej001   36   1019  296    15   5   20  0.631148    0.75
14223  zupcb001  198   5842  483    22  12    5  0.590164    0.65
14225  zuvep001  136   3844  267   415  23   84  0.397541    0.45

       debutYear  ...   SB    CS    BB    SO  IBB  HBP  SH   SF  GIDP  NL
1           1954  ...  240  73.0  1402  1383  293   32  21  121   328   1
2           1962  ...    9   8.0    86   145    3    0   9    6    36   1
4           2001  ...    0   1.0     4     5    0    0   0    0     1   1
8           1997  ...    6   5.0    38    91    2    3   5    7    12   1
10          1993  ...   22  11.0   133   571   11   17  21   12    37   1
...          ...  ...  ...   ...   ...   ...  ...  ...  ..  ...   ...  ..
14218       1991  ...    0   0.0     1    13    0    0   1    1     1   1
14220       1996  ...    1   0.0    12    20    1    1   1    1     4   1
14221       2000  ...    0   2.0    10    51    1    6   0    1     5   1
14223       1991  ...    7   5.0    57   137    3    6  20    8    15   0
14225       1982  ...    2   0.0    34    50    1    2  18    0     8   1

[6392 rows x 40 columns]
```

[140]:
```
catchers
```

[140]:
```
       retroID   GS  InnOuts    PO    A   E  DP  PB  WP  SB_A  ...  SB    CS  \
0     adamb105    1       27     6    0   0   0   0   0     1  ...   0   0.0
1     adamb106    0        0   249   90  12  15   7   0     0  ...   4   2.0
2     adamd101    3       78     9    2   0   0   1   0     0  ...   0   0.0
3     adled101   65     1840   453   26   4   2   8  19    37  ...   0   0.0
4     afent001   20      613   123    5   1   3   6   0    17  ...   0   0.0
...        ...  ...      ...   ...  ...  ..  ..  ..  ..   ...  ...  ..   ...
1524  zimmd101   27      744   150   18   6   1   5  12    10  ...  45  25.0
1525  zimmj101  298     8560  2131  150  21  26  19  84   110  ...   1   2.0
1526  zinta001    0        3     2    0   0   0   0   0     0  ...   0   0.0
1527  zunim001  535    14489  4356  264  21  22  39   0   248  ...   2   4.0
1528  zupof101    1      114    31    1   2   0   1   1     2  ...   0   0.0

       BB   SO  IBB  HBP  SH  SF  GIDP  NL
0       0    5    0    0   0   0     0   0
1       6   27    0    0   3   0     0   1
2       1    3    0    0   0   0     1   0
3      18   80    5    2   2   1     9   1
4       5   32    0    0   1   1     1   1
...   ...  ...  ...  ...  ..  ..   ...  ..
1524  246  678   27   13  37  14    99   1
```

```
1525    78  154    12    11  31    4     38   1
1526     5   34     0     0   0    1      0   1
1527   138  714     1    45   8   11     38   0
1528     2    6     0     0   0    0      0   0

[1529 rows x 38 columns]
```

[169]:
```python
fielders[fielders['retroID'].isin(catchers['retroID'])]
```

[169]:
```
        retroID   GS  InnOuts     PO     A    E    DP    weight  height  \
54     adamb105    2       54     20     1    0     1  0.508197    0.55
55     adamb106    0        0      0     0    0     0  0.446721    0.50
108    ainse101    0        0     11     0    0     0  0.426230    0.40
144    alexg101   22      500     83     6    3     4  0.487705    0.55
151    alfaj002    1       31      8     2    0     1  0.610656    0.55
...         ...  ...      ...    ...   ...  ...   ...       ...     ...
14111  yorkr101    0        0  11425  1030  136  1077  0.545082    0.50
14139  younj001  843    23486   1679   756  115   113  0.426230    0.45
14183  zaung001    0       33      3     2    0     1  0.385246    0.35
14199  zimmd101  813    21993   1491  2204  150   417  0.364754    0.30
14210  zinta001    5      198     65     5    1     4  0.508197    0.55

       debutYear  ...  SB    CS   BB   SO  IBB  HBP  SH  SF  GIDP  NL
54          1977  ...   0   0.0    0    5    0    0   0   0     0   0
55          1910  ...   4   2.0    6   27    0    0   3   0     0   1
108         1910  ...  20  11.5  125  122    0    3  44   0     0   1
144         1975  ...   8  12.0  154  381   12    5   4  19    34   1
151         2016  ...   3   0.0   22  179    8   18   0   1     4   1
...          ...  ...  ..   ...  ...  ...  ...  ...  ..  ..   ...  ..
14111       1934  ...  38  26.0  792  867    0   12  25   0   155   0
14139       1976  ...  60  55.0  332  589   30   36  18  33    80   1
14183       1995  ...  23  19.0  479  544   30   29  14  31    87   1
14199       1954  ...  45  25.0  246  678   27   13  37  14    99   1
14210       2002  ...   0   0.0    5   34    0    0   0   1     0   1

[655 rows x 40 columns]
```

We have some catchers that are also in the fielders table.

[168]:
```python
fielders[(fielders['retroID'].isin(catchers['retroID']) & fielders['pos_C'] ==␣
 ↪1)]
```

[168]:
```
        retroID  GS  InnOuts   PO  A  E  DP    weight  height  debutYear  \
108    ainse101   0        0   11  0  0   0  0.426230    0.40       1910
144    alexg101  22      500   83  6  3   4  0.487705    0.55       1975
151    alfaj002   1       31    8  2  0   1  0.610656    0.55       2016
156    allaa001   0       33   15  0  0   2  0.590164    0.70       1986
```

```
169    alleg001    1        54     4    4    0     2  0.446721    0.40        1979
...         ...  ...       ...   ...   ..   ..   ...       ...     ...         ...
13914  wingi101    0         0     7    2    1     0  0.344262    0.35        1911
13956  wockj001  249      6120  1429   90   17   133  0.467213    0.45        1974
14066  wronr001    0         6     2    0    0     0  0.446721    0.50        1988
14074  wyneb001    0         6     0    0    0     0  0.467213    0.50        1976
14183  zaung001    0        33     3    2    0     1  0.385246    0.35        1995

        ...  SB    CS   BB   SO  IBB  HBP  SH  SF  GIDP  NL
108     ...  20  11.5  125  122    0    3  44   0     0   1
144     ...   8  12.0  154  381   12    5   4  19    34   1
151     ...   3   0.0   22  179    8   18   0   1     4   1
156     ...  23  18.0   87  223    4    9  35  16    27   1
169     ...   3   7.0  130  192    3    5  15  11    35   0
...     ...  ..   ...  ...  ...  ...  ...  ..  ..   ...  ..
13914   ...  17  16.0  121   84    0    4  36   0     0   1
13956   ...   5  11.0  277  278   14    7   5  12    52   1
14066   ...   1   0.0    5   41    2    1   2   2     3   1
14074   ...  10  13.0  626  428   41   17  58  36   119   0
14183   ...  23  19.0  479  544   30   29  14  31    87   1

[430 rows x 40 columns]
```

[170]:
```
to_inspect = fielders[(fielders['retroID'].isin(catchers['retroID']) &␣
 ↪fielders['pos_C'] == 1)]['retroID']
```

[171]:
```
catchers[catchers['retroID'].isin(to_inspect)]
```

[171]:
```
        retroID    GS  InnOuts    PO    A   E  DP  PB  WP  SB_A  ...  SB    CS  \
6      ainse101     0        0  1528  361  72  31  34   0     0  ...  20  11.5
7      alexg101   205     5481  1008  100  35   8  24   0   212  ...   8  12.0
8      alfaj002   130     3430  1135   71  13   9  14   0    72  ...   3   0.0
10     allaa001   453    11965  2395  208  52  24  41   0   292  ...  23  18.0
11     alleg001   342     8959  1762  146  32  24  27   0   233  ...   3   7.0
...         ...   ...      ...   ...  ...  ..  ..  ..  ..   ...  ...  ..   ...
1498   wingi101     0        0  1716  536  79  53  36   0     0  ...  17  16.0
1502   wockj001   216     5957  1212  119  39  17  27   0   188  ...   5  11.0
1508   wronr001    47     1360   296   32   8   3   4   0    25  ...   1   0.0
1509   wyneb001  1164    31563  6281  583  75  88  61   0   708  ...  10  13.0
1521   zaung001   910    24700  6134  418  88  59  51   0   651  ...  23  19.0

        BB   SO  IBB  HBP  SH  SF  GIDP  NL
6      125  122    0    3  44   0     0   1
7      154  381   12    5   4  19    34   1
8       22  179    8   18   0   1     4   1
10      87  223    4    9  35  16    27   1
11     130  192    3    5  15  11    35   0
```

```
...    ...   ...   ...   ...  ..   ..    ...  ..
1498   121    84     0     4  36    0      0   1
1502   277   278    14     7   5   12     52   1
1508     5    41     2     1   2    2      3   1
1509   626   428    41    17  58   36    119   0
1521   479   544    30    29  14   31     87   1

[430 rows x 38 columns]
```

It looks like the information in the catchers table is a better indicator of the player's career.

[175]:
```
fielders[fielders['retroID'] == 'alexg101'][['debutYear', 'finalYear']]
```

[175]:
```
     debutYear  finalYear
144       1975       1981
```

[174]:
```
catchers[catchers['retroID'] == 'alexg101'][['debutYear', 'finalYear']]
```

[174]:
```
   debutYear  finalYear
7       1975       1981
```

Thee years line up. So for any catcher who appears in the fielders table with his position as catcher, we're going to drop him from the fielders table and only use the catchers information. We'll keep catchers in the fielders table if they're in a different position.

[176]:
```
fielders[fielders['pos_C'] == 1]
```

[176]:
```
        retroID   GS  InnOuts    PO   A   E   DP    weight  height  debutYear  \
108    ainse101    0        0    11   0   0    0  0.426230    0.40       1910
144    alexg101   22      500    83   6   3    4  0.487705    0.55       1975
151    alfaj002    1       31     8   2   0    1  0.610656    0.55       2016
156    allaa001    0       33    15   0   0    2  0.590164    0.70       1986
169    alleg001    1       54     4   4   0    2  0.446721    0.40       1979
...         ...  ...      ...   ...  ..  ..  ...       ...     ...        ...
13914  wingi101    0        0     7   2   1    0  0.344262    0.35       1911
13956  wockj001  249     6120  1429  90  17  133  0.467213    0.45       1974
14066  wronr001    0        6     2   0   0    0  0.446721    0.50       1988
14074  wyneb001    0        6     0   0   0    0  0.467213    0.50       1976
14183  zaung001    0       33     3   2   0    1  0.385246    0.35       1995

       ...  SB    CS   BB   SO  IBB  HBP  SH  SF  GIDP  NL
108    ...  20  11.5  125  122    0    3  44   0     0   1
144    ...   8  12.0  154  381   12    5   4  19    34   1
151    ...   3   0.0   22  179    8   18   0   1     4   1
156    ...  23  18.0   87  223    4    9  35  16    27   1
169    ...   3   7.0  130  192    3    5  15  11    35   0
...    ...  ..   ...  ...  ...  ...  ...  ..  ..   ...  ..
13914  ...  17  16.0  121   84    0    4  36   0     0   1
```

```
13956  ...   5  11.0  277  278   14    7   5  12    52   1
14066  ...   1   0.0    5   41    2    1   2   2     3   1
14074  ...  10  13.0  626  428   41   17  58  36   119   0
14183  ...  23  19.0  479  544   30   29  14  31    87   1

[430 rows x 40 columns]
```

All fielders with a position of 'C' are in the catchers table, so we don't have to worry about leaving any by filtering with the following predicate.

[184]:
```
fielders = fielders[~(fielders['retroID'].isin(catchers['retroID']) &␣
↪fielders['pos_C'] == 1)]
```

[185]:
```
fielders.shape
```

[185]:
```
(5962, 40)
```

batting_build_tensor

April 30, 2020

Building the Batters Tensor

I've separated this from the model itself so that we could visualize and work through the data, but in the actual script this will just be a short few lines at the beginning of the model file.

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the Data

[2]:
```
df = pd.read_csv('../core/output/batters.csv')
```

[3]:
```
indexer = df.reset_index()[['index', 'retroID']].to_dict()['retroID']
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15293 entries, 0 to 15292
Data columns (total 38 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   retroID    15293 non-null  object
 1   weight     15285 non-null  float64
 2   height     15287 non-null  float64
 3   debutYear  15293 non-null  int64
 4   finalYear  15293 non-null  int64
 5   pos_1B     15293 non-null  int64
 6   pos_2B     15293 non-null  int64
 7   pos_3B     15293 non-null  int64
 8   pos_C      15293 non-null  int64
 9   pos_OF     15293 non-null  int64
 10  pos_P      15293 non-null  int64
 11  pos_SS     15293 non-null  int64
 12  bats_L     15293 non-null  int64
 13  throws_L   15293 non-null  int64
 14  G          15293 non-null  int64
 15  AB         15293 non-null  int64
```

```
 16  PA         15293 non-null  int64
 17  R          15293 non-null  int64
 18  H          15293 non-null  int64
 19  1B         15293 non-null  int64
 20  2B         15293 non-null  int64
 21  3B         15293 non-null  int64
 22  HR         15293 non-null  int64
 23  RBI        15293 non-null  int64
 24  SB         15293 non-null  int64
 25  CS         15293 non-null  float64
 26  BB         15293 non-null  int64
 27  SO         15293 non-null  int64
 28  IBB        15293 non-null  int64
 29  HBP        15293 non-null  int64
 30  SH         15293 non-null  int64
 31  SF         15293 non-null  int64
 32  GIDP       15293 non-null  int64
 33  NL         15293 non-null  int64
 34  wOBA       15293 non-null  float64
 35  wRC+       15293 non-null  int64
 36  WAR        15293 non-null  float64
 37  Batting    15293 non-null  float64
dtypes: float64(6), int64(31), object(1)
memory usage: 4.4+ MB
```

[5]:
```
y = df['Batting'].values
```

[6]:
```
y
```

[6]:
```
array([0.00035809, 0.350195  , 0.157131  , ..., 0.0900877 , 0.135118  ,
       0.0901954 ])
```

[7]:
```
to_drop = ['retroID', 'debutYear', 'finalYear', 'Batting']
```

[8]:
```
df.drop(columns=to_drop, inplace=True)
```

[9]:
```
df
```

[9]:
```
         weight  height  pos_1B  pos_2B  pos_3B  pos_C  pos_OF  pos_P  pos_SS  \
0      0.569672    0.60       0       0       0      0       0      1       0
1      0.426230    0.45       0       0       0      0       1      0       0
2      0.467213    0.60       1       0       0      0       0      0       0
3      0.467213    0.60       0       0       0      0       0      1       0
4      0.442623    0.50       1       0       0      0       0      0       0
...         ...     ...     ...     ...     ...    ...     ...    ...     ...
15288  0.590164    0.65       0       0       0      0       1      0       0
15289  0.434426    0.40       0       0       0      1       0      0       0
```

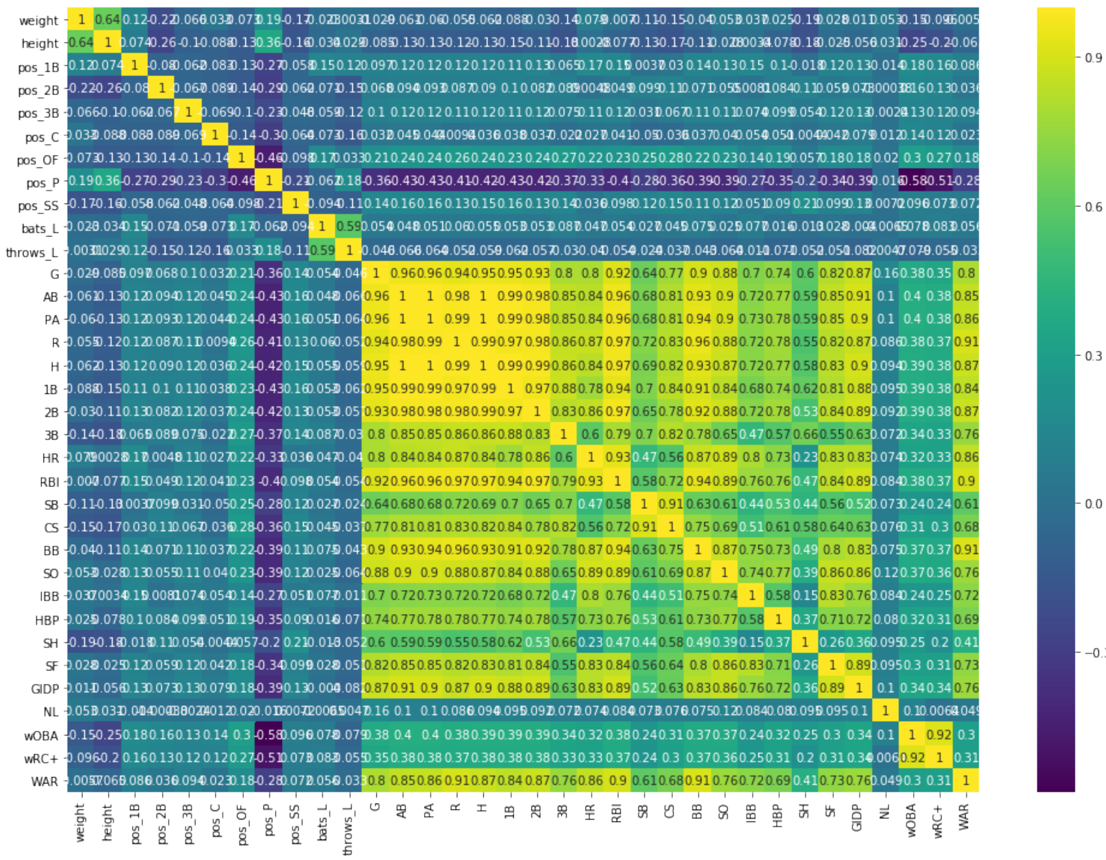
```
15290  0.487705   0.65        0        0        0       0        0       1        0
15291  0.397541   0.45        0        0        0       0        0       0        1
15292  0.467213   0.60        0        0        0       0        0       1        0

       bats_L  ...    SO  IBB  HBP  SH   SF  GIDP  NL   wOBA  wRC+    WAR
0           0  ...     2    0    0   1    0     0   1  0.000  -100   -0.1
1           0  ...  1383  293   32  21  121   328   1  0.403   153  136.3
2           0  ...   145    3    0   9    6    36   1  0.282    76   -1.7
3           0  ...     3    0    0   0    0     0   1  0.000  -100   -0.1
4           1  ...     5    0    0   0    0     1   1  0.184     0   -0.4
...       ...  ...   ...  ...  ...  ..  ...   ...  ..    ...   ...    ...
15288       0  ...   137    3    6  20    8    15   0  0.293    74   -0.9
15289       1  ...     6    0    0   0    0     0   0  0.225    37   -0.2
15290       0  ...    39    0    0  16    0     3   1  0.179     0   -0.3
15291       0  ...    50    1    2  18    0     8   1  0.254    52   -2.2
15292       0  ...     0    0    0   0    0     0   0  0.000     0    0.0

[15293 rows x 34 columns]
```

We now have a sort of proto-tensor, but maybe we can do some data manipulation to make the resulting model more efficient.

Observing Data Information

[10]:
```python
plt.figure(figsize=(17,12))
ax = sns.heatmap(df.corr(), annot=True, cmap='viridis')
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

[10]:
```
(34.0, 0.0)
```

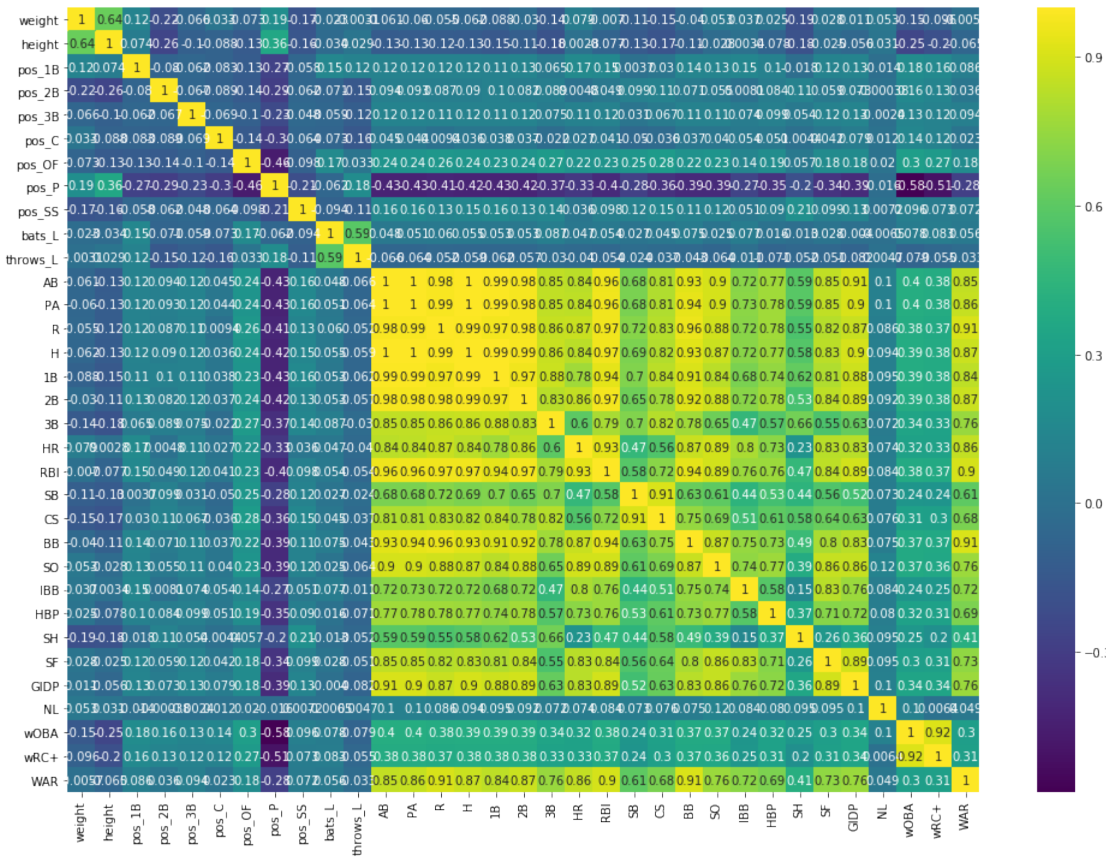


We see a high correlation between G (games) and AB/PA (at-bats/plate appearances). It makes sense that we can drop the G column.

```
[11]: df.drop(columns=['G'], inplace=True)
```

```
[12]: plt.figure(figsize=(17,12))
      ax = sns.heatmap(df.corr(), annot=True, cmap='viridis')
      bottom, top = ax.get_ylim()
      ax.set_ylim(bottom + 0.5, top - 0.5)
```

```
[12]: (33.0, 0.0)
```

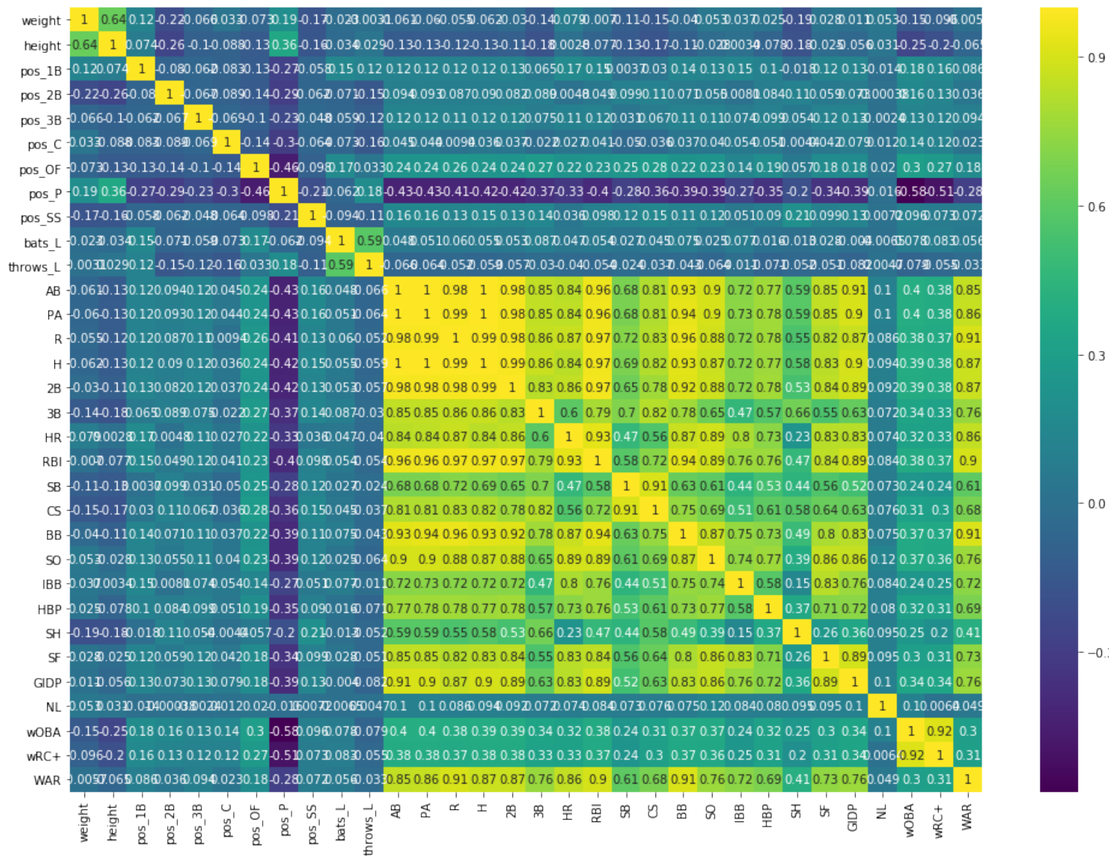


There's obviously a high correlation in H (hits) and 1B/2B/3B/HR (singles, doubles, triples and home runs). We added 1B as a column to help with statistics but it's unnecessary now - the relationship between hits and types of hits will be preserved in the model.

[13]:
```
df.drop(columns=['1B'], inplace=True)
```

[14]:
```
plt.figure(figsize=(17,12))
ax = sns.heatmap(df.corr(), annot=True, cmap='viridis')
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

[14]: (32.0, 0.0)

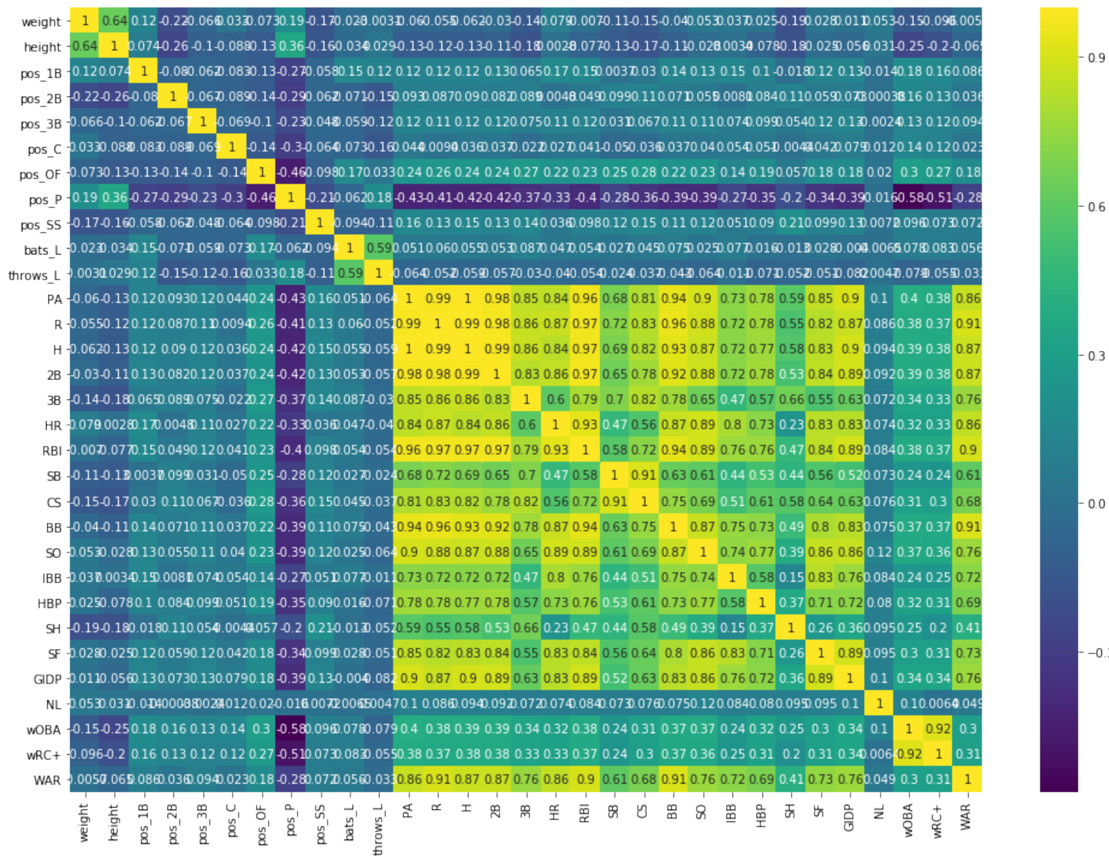


We have total correlation between AB (at-bats) and PA (plate-appearances). This makes sense, because PA is just AB with some other situations added in. PA is more robust and is better related to overall output (since it includes sacrifices, hits-by-pitch and walks) so we'll keep PA and get rid of AB.

[15]:
```
df.drop(columns=['AB'], inplace=True)
```

[16]:
```
plt.figure(figsize=(17,12))
ax = sns.heatmap(df.corr(), annot=True, cmap='viridis')
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

[16]: (31.0, 0.0)



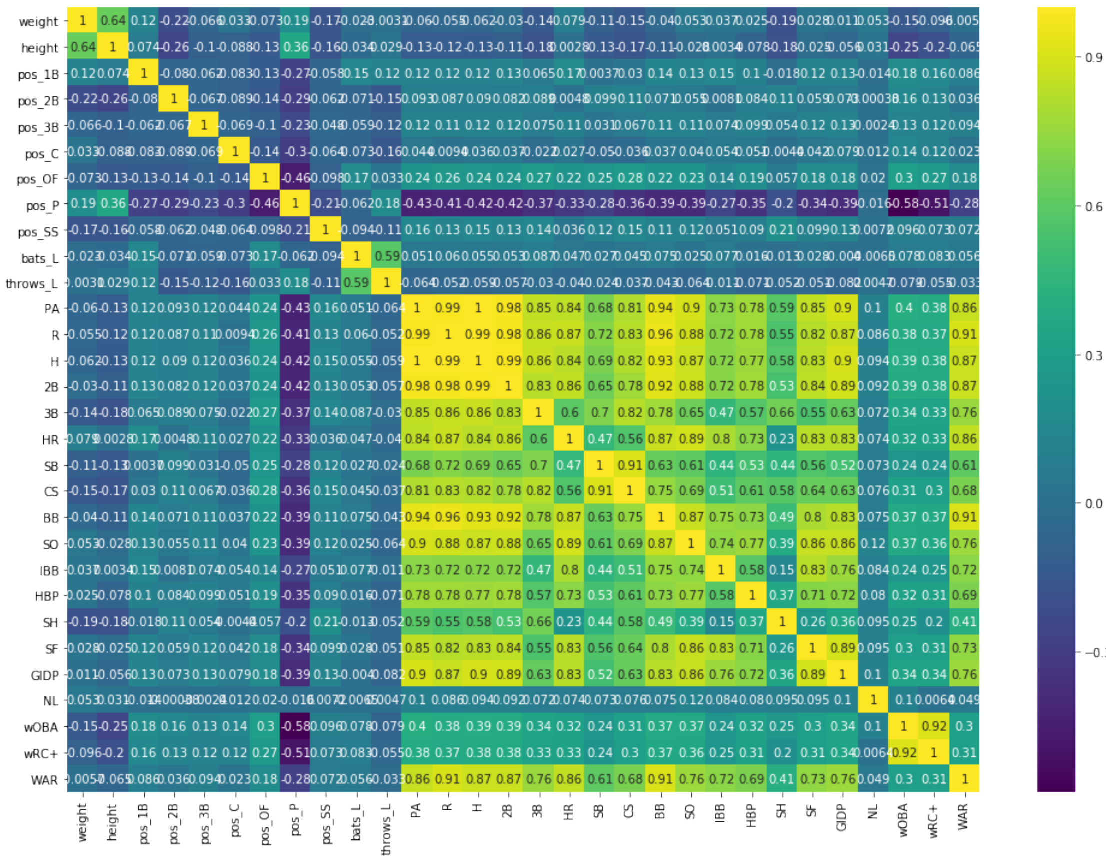
With the high correlation along the PA line, it may seem that we don't need them either. My reasoning for keeping them is a baseball-related one: we have most of the stats that make up a plate appearance, but we're missing the 'flied out' stat. This is an important one, as flyouts are a huge part of producing outs. Because of this, I'm going to keep PA as a stat.

One thing that I think we could drop is the RBI (runs batted in) stat. We don't see it appearing in many advanced stats, primarily wOBA and OPS+ with which we are concerned, and we intuitively see that it's encompasses factors beyond the pure output of the hitter. It could be said that the RBI stat tracks a hitter's ability to hit "under pressure", but that's the kind of soft feature we're not going to consider. I think we get more important information from hits, doubles, triples, home runs and even runs than we do from RBIs. We also see extremely high correlation between RBI and R/H/2B/HR, which further supports the decision to remove RBI.

[21]:
```
df.drop(columns=['RBI'], inplace=True)
```

[22]:
```
plt.figure(figsize=(17,12))
ax = sns.heatmap(df.corr(), annot=True, cmap='viridis')
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

[22]: (30.0, 0.0)



We still see a hot spot around PA/R/H/2B, but these are all important stats that we're not going to remove.

One thing I think we SHOULD consider is that we have both wOBA and wRC+. There are a few issues with this:

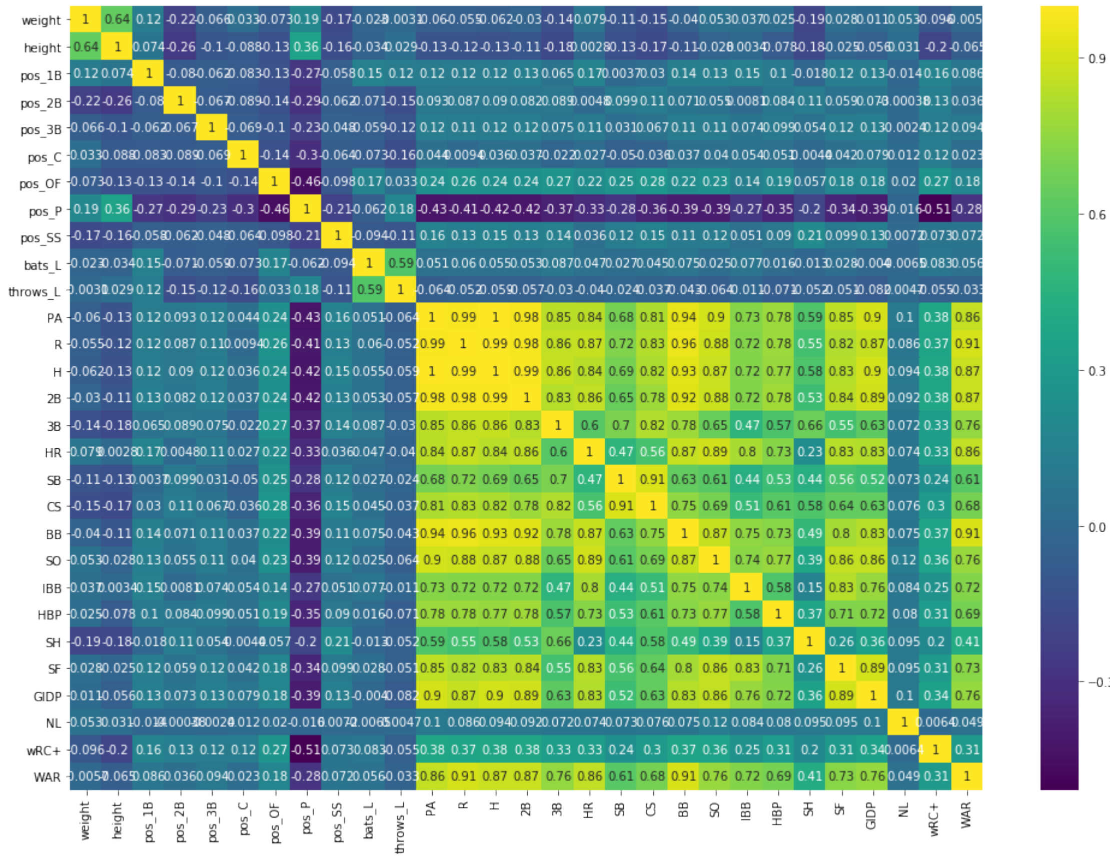
- They are both stats that essentially only consider offensive output, so do we really need two?
- wRC+ incorporates wRAA, which is built from wOBA, so the information is a bit repeated.
- We want to minimize the model's use of secondary/advanced stats which are extrapolated from the primary stats we have. However, wRC+ and WAR both normalize to league trends and thus offer a nice aggregation of data that might not be intrinsically found by the model.

So I think we can go ahead and get rid of wOBA.

[23]:
```
df.drop(columns=['wOBA'], inplace=True)
```

[24]:
```
plt.figure(figsize=(17,12))
ax = sns.heatmap(df.corr(), annot=True, cmap='viridis')
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

[24]: (29.0, 0.0)



I said before that I don't want to drop R/H/2B/3B, so ignoring that area I think we now have a good looking model that's ready to run.

[25]:
```
df
```

[25]:
```
         weight  height  pos_1B  pos_2B  pos_3B  pos_C  pos_OF  pos_P  pos_SS  \
0      0.569672    0.60       0       0       0      0       0      1       0
1      0.426230    0.45       0       0       0      0       1      0       0
2      0.467213    0.60       1       0       0      0       0      0       0
3      0.467213    0.60       0       0       0      0       0      1       0
4      0.442623    0.50       1       0       0      0       0      0       0
...         ...     ...     ...     ...     ...    ...     ...    ...     ...
15288  0.590164    0.65       0       0       0      0       1      0       0
15289  0.434426    0.40       0       0       0      1       0      0       0
15290  0.487705    0.65       0       0       0      0       0      1       0
15291  0.397541    0.45       0       0       0      0       0      0       1
15292  0.467213    0.60       0       0       0      0       0      1       0
```

```
       bats_L  ...    BB    SO  IBB  HBP  SH   SF  GIDP  NL  wRC+    WAR
0           0  ...     0     2    0    0   1    0     0   1  -100   -0.1
1           0  ...  1402  1383  293   32  21  121   328   1   153  136.3
2           0  ...    86   145    3    0   9    6    36   1    76   -1.7
3           0  ...     0     3    0    0   0    0     0   1  -100   -0.1
4           1  ...     4     5    0    0   0    0     1   1     0   -0.4
...       ...  ...   ...   ...  ...  ...  ..  ...   ...  ..   ...    ...
15288       0  ...    57   137    3    6  20    8    15   0    74   -0.9
15289       1  ...     2     6    0    0   0    0     0   0    37   -0.2
15290       0  ...     9    39    0    0  16    0     3   1     0   -0.3
15291       0  ...    34    50    1    2  18    0     8   1    52   -2.2
15292       0  ...     0     0    0    0   0    0     0   0     0    0.0

[15293 rows x 29 columns]
```

[28]:
```python
# We'll add the 'Batting' column back in to save our tensor
df.insert(loc=len(df.columns), column='Batting', value=y)
```

[29]:
```python
df
```

[29]:
```
         weight  height  pos_1B  pos_2B  pos_3B  pos_C  pos_OF  pos_P  pos_SS  \
0      0.569672    0.60       0       0       0      0       0      1       0
1      0.426230    0.45       0       0       0      0       1      0       0
2      0.467213    0.60       1       0       0      0       0      0       0
3      0.467213    0.60       0       0       0      0       0      1       0
4      0.442623    0.50       1       0       0      0       0      0       0
...         ...     ...     ...     ...     ...    ...     ...    ...     ...
15288  0.590164    0.65       0       0       0      0       1      0       0
15289  0.434426    0.40       0       0       0      1       0      0       0
15290  0.487705    0.65       0       0       0      0       0      1       0
15291  0.397541    0.45       0       0       0      0       0      0       1
15292  0.467213    0.60       0       0       0      0       0      1       0

       bats_L  ...    SO  IBB  HBP  SH   SF  GIDP  NL  wRC+    WAR   Batting
0           0  ...     2    0    0   1    0     0   1  -100   -0.1  0.000358
1           0  ...  1383  293   32  21  121   328   1   153  136.3  0.350195
2           0  ...   145    3    0   9    6    36   1    76   -1.7  0.157131
3           0  ...     3    0    0   0    0     0   1  -100   -0.1  0.000358
4           1  ...     5    0    0   0    0     1   1     0   -0.4  0.090001
...       ...  ...   ...  ...  ...  ..  ...   ...  ..   ...    ...       ...
15288       0  ...   137    3    6  20    8    15   0    74   -0.9  0.156062
15289       1  ...     6    0    0   0    0     0   0    37   -0.2  0.123425
15290       0  ...    39    0    0  16    0     3   1     0   -0.3  0.090088
15291       0  ...    50    1    2  18    0     8   1    52   -2.2  0.135118
15292       0  ...     0    0    0   0    0     0   0     0    0.0  0.090195

[15293 rows x 30 columns]
```

pitching_build_tensor

April 30, 2020

Building the Batters Tensor

I've separated this from the model itself so that we could visualize and work through the data, but in the actual script this will just be a short few lines at the beginning of the model file.

[14]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the Data

[33]:
```
df = pd.read_csv('../core/output/pitchers.csv')
```

[34]:
```
indexer = df.reset_index()[['index', 'retroID']].to_dict()['retroID']
```

[35]:
```
y = df['Pitching'].values
```

[36]:
```
y
```

[36]:
```
array([0.602913, 0.636924, 0.603736, ..., 0.612847, 0.608497, 0.611166])
```

[37]:
```
df.columns
```

[37]:
```
Index(['retroID', 'BAOpp', 'CG', 'SHO', 'IPouts', 'H', 'ER', 'HR', 'BB', 'SO',
       'IBB', 'WP', 'HBP', 'BK', 'BFP', 'GF', 'R', 'SH', 'SF', 'GIDP', 'K%',
       'IP', 'K/9', 'BB/9', 'HR/9', 'BABIP', 'LOB%', 'ERA', 'FIP', 'WAR',
       'Pitching'],
      dtype='object')
```

[38]:
```
to_drop = ['retroID', 'Pitching']
```

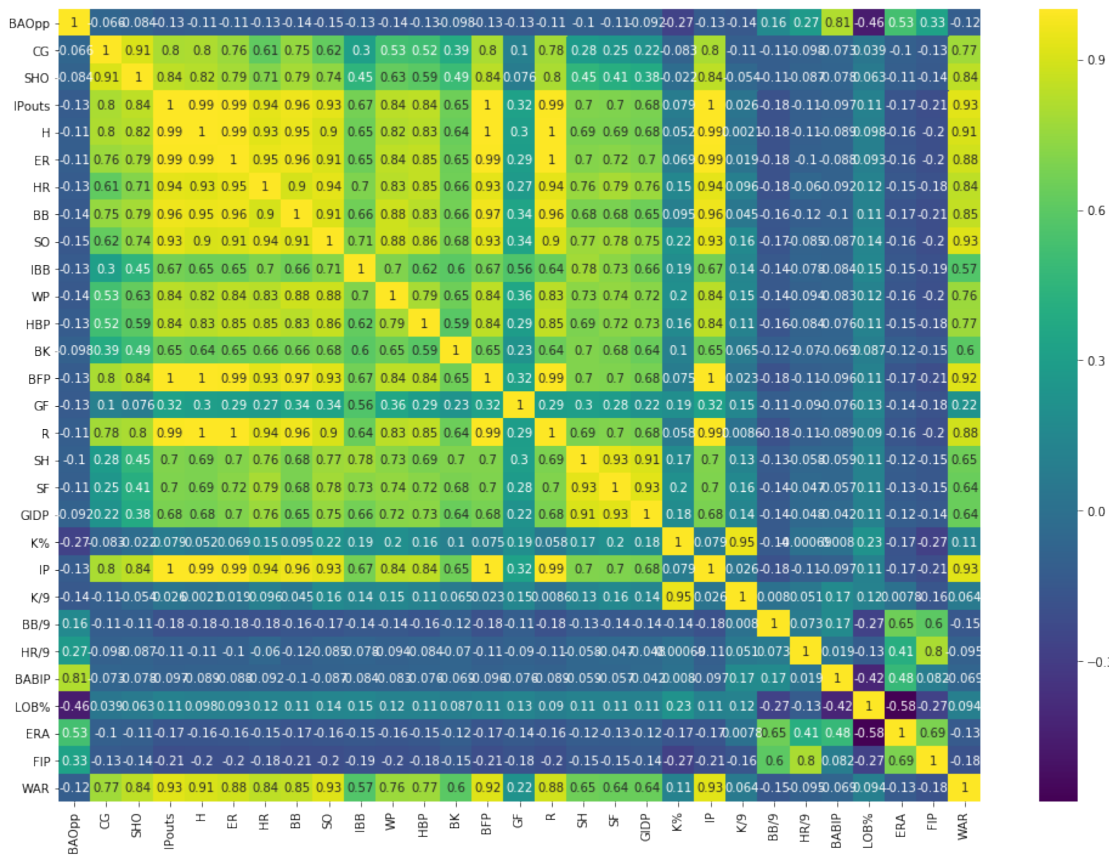
[39]:
```
df = df.drop(columns=to_drop)
```

Observing Data Information

[40]:
```
plt.figure(figsize=(17,12))
ax = sns.heatmap(df.corr(), annot=True, cmap='viridis')
bottom, top = ax.get_ylim()
```

```
ax.set_ylim(bottom + 0.5, top - 0.5)
```

[40]: (29.0, 0.0)

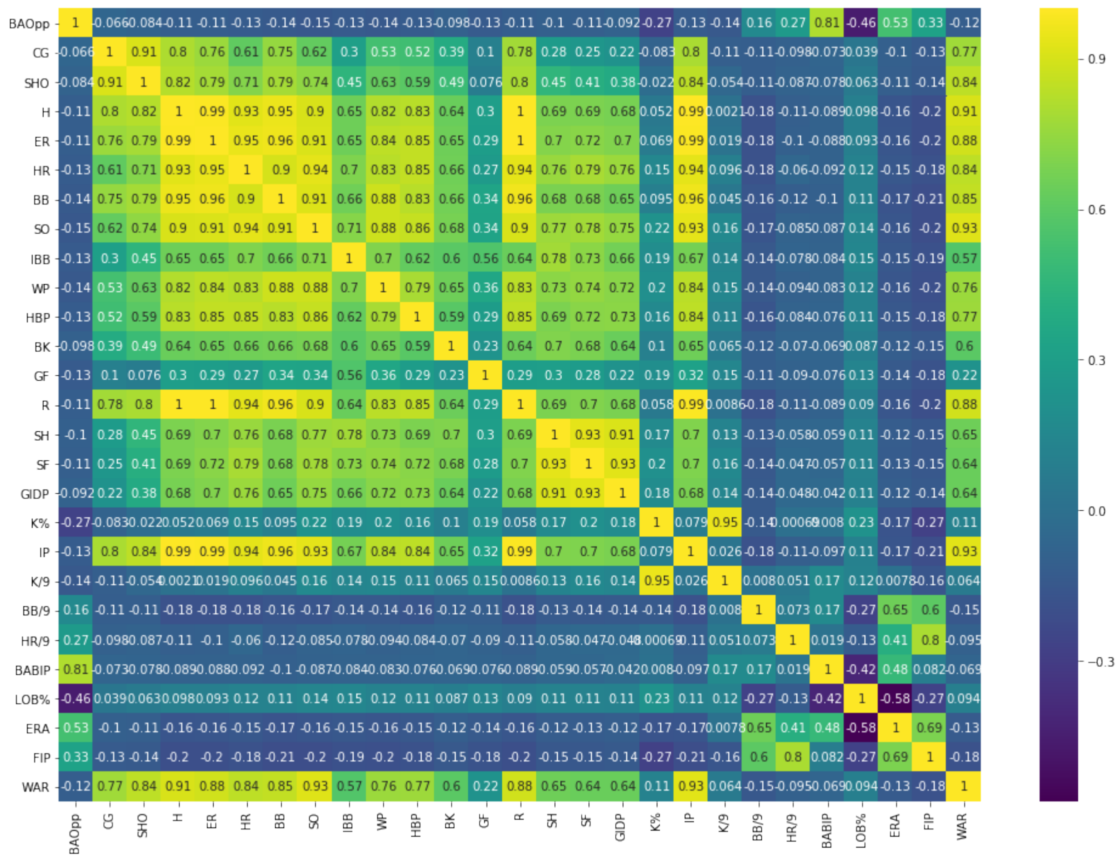


We see a lot of correlations with both IPouts and BFP, so we'll drop those.

[41]:
```
df = df.drop(columns=['IPouts', 'BFP'])
```

[42]:
```
plt.figure(figsize=(17,12))
ax = sns.heatmap(df.corr(), annot=True, cmap='viridis')
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

[42]: (27.0, 0.0)

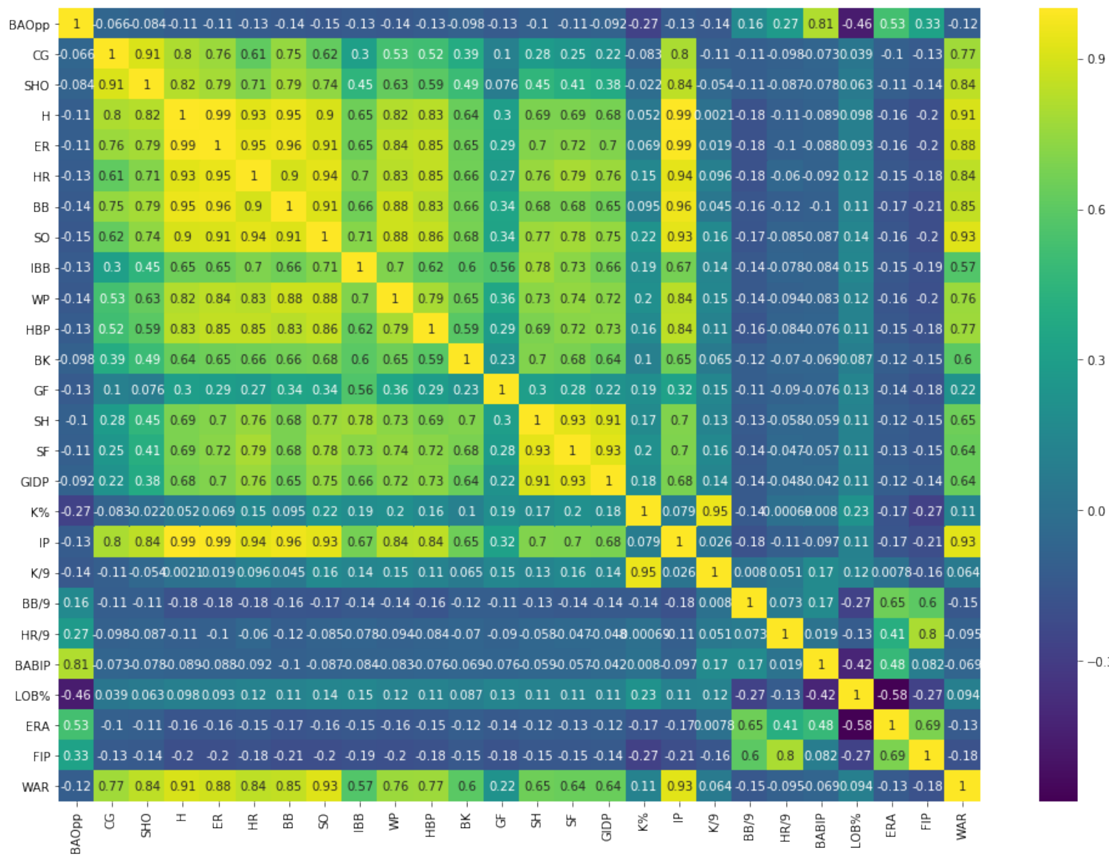


I think R (runs) is sufficiently covered by ER (earned runs) and HR (home runs), so I'll drop it too.

[43]:
```
df = df.drop(columns=['R'])
```

[44]:
```
plt.figure(figsize=(17,12))
ax = sns.heatmap(df.corr(), annot=True, cmap='viridis')
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

[44]: (26.0, 0.0)



I'm happy with this version of the tensor.

[45]:
```python
# We'll add the 'Pitching' column back in to save our tensor
df.insert(loc=len(df.columns), column='Pitching', value=y)
```

[46]:
```python
df
```

[46]:
```
      BAOpp  CG  SHO     H   ER   HR   BB   SO  IBB  WP  ...        IP   K/9  \
0    0.2574   0    0   296  160   41  183  340   22  12  ...  0.062360  9.08
1    0.2508  22    5  1085  468   89  457  641   45  22  ...  0.205233  5.20
2    0.2447   0    0   309  135   42  116  280   10  10  ...  0.061102  7.62
3    0.2786  37    5  1405  627  162  352  484   28  18  ...  0.237967  3.39
4    0.2804  31    6  1779  791  154  620  888   30  53  ...  0.309765  4.77
...     ...  ..  ...   ...  ...  ...  ...  ...  ...  ..  ...       ...   ...
8020 0.2700  30    5   956  366   54  301  207    0   8  ...  0.171925  2.00
8021 0.2717  23    3   767  374   35  468  383    0  28  ...  0.145445  4.39
8022 0.2286   0    0   169   71   18  114  210   11  16  ...  0.038711  9.01
8023 0.2760   9    2   660  253   56  203  223   29  10  ...  0.118817  3.12
8024 0.2183   0    0    57   22    3   34   80    5   2  ...  0.013360  9.91
```

```
      BB/9  HR/9  BABIP  LOB%   ERA   FIP   WAR  Pitching
0     4.89  1.09  0.285  74.5  4.27  4.45   1.1  0.602913
1     3.71  0.72  0.282  73.4  3.80  3.85  11.7  0.636924
2     3.16  1.14  0.281  77.7  3.67  4.24   0.6  0.603736
3     2.46  1.13  0.278  69.3  4.39  4.46  10.2  0.628847
4     3.33  0.83  0.295  70.0  4.25  4.25  22.7  0.666725
...    ...   ...    ...   ...   ...   ...   ...       ...
8020  2.91  0.52  0.267  70.7  3.54  3.80   9.3  0.630540
8021  5.36  0.40  0.283  69.0  4.28  3.96   3.3  0.610437
8022  4.89  0.77  0.267  78.7  3.00  3.94   2.7  0.612847
8023  2.84  0.78  0.270  73.2  3.54  3.93   1.9  0.608497
8024  4.21  0.37  0.293  79.1  2.72  3.22   1.1  0.611166

[8025 rows x 27 columns]
```

[ ]:

Game Log Processing

Taking Retrosheet game log CSV files and converting them into appropriate tensors for the predictive models

convert_gamelogs

May 7, 2020

Processing Game Logs

The information used here was obtained free of charge from and is copyrighted by Retrosheet. Interested parties may contact Retrosheet at “www.retrosheet.org”.

We’re going to use Retrosheet game logs as input data to our predictive model. The first thing we need to do is process them to fit our needs.

[135]:
```
import pandas as pd
import os
```

[156]:
```
df = pd.read_csv('../core/data/lahman/mlb_data/Teams.csv')
df = df[['teamID', 'franchID']]
team_dict = df.set_index('teamID').to_dict()['franchID']
team_dict['MLN'] = 'ATL'


def get_team(team):
    return team_dict[team] if team_dict[team] is not None else team
```

These are the columns of the Retrosheet game logs. This metadata was obtained here: https://www.retrosheet.org/gamelogs/glfields.txt

[137]:
```
columns = [
    'date',
    'game_number',
    'day_of_week',
    'visit_team',
    'visit_league',
    'visit_game_number',
    'home_team',
    'home_league',
    'home_game_number',
    'visit_score',
    'home_score',
    'game_length_outs',
    'day_night',
    'completion_info',
    'forfeit_info',
```

```
    'protest_info',
    'park_id',
    'attendance',
    'time_minutes',
    'visit_line_score',
    'home_line_score',
    'visit_ab',
    'visit_h',
    'visit_2b',
    'visit_3b',
    'visit_hr',
    'visit_rbi',
    'visit_sh',
    'visit_sf',
    'visit_hbp',
    'visit_bb',
    'visit_ibb',
    'visit_k',
    'visit_sb',
    'visit_cs',
    'visit_gidp',
    'visit_ci',
    'visit_lob',
    'visit_pitchers_used',
    'visit_individual_er',
    'visit_team_er',
    'visit_wp',
    'visit_bk',
    'visit_po',
    'visit_assists',
    'visit_e',
    'visit_passed_balls',
    'visit_double_plays',
    'visit_triple_plays',
    'home_ab',
    'home_h',
    'home_2b',
    'home_3b',
    'home_hr',
    'home_rbi',
    'home_sh',
    'home_sf',
    'home_hbp',
    'home_bb',
    'home_ibb',
    'home_k',
    'home_sb',
```

```
    'home_cs',
    'home_gidp',
    'home_ci',
    'home_lob',
    'home_pitchers_used',
    'home_individual_er',
    'home_team_er',
    'home_wp',
    'home_bk',
    'home_po',
    'home_assists',
    'home_e',
    'home_passed_balls',
    'home_double_plays',
    'home_triple_plays',
    'hp_ump_id',
    'hp_ump_name',
    '1b_ump_id',
    '1b_ump_name',
    '2b_ump_id',
    '2b_ump_name',
    '3b_ump_id',
    '3b_ump_name',
    'lf_ump_id',
    'lf_ump_name',
    'rf_ump_id',
    'rf_ump_name',
    'visit_manager_id',
    'visit_manager_name',
    'home_manager_id',
    'home_manager_name',
    'winning_pitcher_id',
    'winning_pitcher_name',
    'losing_pitcher_id',
    'losing_pitcher_name',
    'saving_pitcher_id',
    'saving_pitcher_name',
    'winning_rbi_batter_id',
    'winning_rbi_batter_name',
    'visit_sp_id',
    'visit_sp_name',
    'home_sp_id',
    'home_sp_name',
    'visit_player_1_id',
    'visit_player_1_name',
    'visit_player_1_pos',
    'visit_player_2_id',
```

```
    'visit_player_2_name',
    'visit_player_2_pos',
    'visit_player_3_id',
    'visit_player_3_name',
    'visit_player_3_pos',
    'visit_player_4_id',
    'visit_player_4_name',
    'visit_player_4_pos',
    'visit_player_5_id',
    'visit_player_5_name',
    'visit_player_5_pos',
    'visit_player_6_id',
    'visit_player_6_name',
    'visit_player_6_pos',
    'visit_player_7_id',
    'visit_player_7_name',
    'visit_player_7_pos',
    'visit_player_8_id',
    'visit_player_8_name',
    'visit_player_8_pos',
    'visit_player_9_id',
    'visit_player_9_name',
    'visit_player_9_pos',
    'home_player_1_id',
    'home_player_1_name',
    'home_player_1_pos',
    'home_player_2_id',
    'home_player_2_name',
    'home_player_2_pos',
    'home_player_3_id',
    'home_player_3_name',
    'home_player_3_pos',
    'home_player_4_id',
    'home_player_4_name',
    'home_player_4_pos',
    'home_player_5_id',
    'home_player_5_name',
    'home_player_5_pos',
    'home_player_6_id',
    'home_player_6_name',
    'home_player_6_pos',
    'home_player_7_id',
    'home_player_7_name',
    'home_player_7_pos',
    'home_player_8_id',
    'home_player_8_name',
    'home_player_8_pos',
```

```
    'home_player_9_id',
    'home_player_9_name',
    'home_player_9_pos',
    'additional_info',
    'acquisition_info'
]
```

The script is broken up here, then I later explore what I need to do to process the data. At the end I combine that all into one loop.

[12]:
```
for year in range(1919, 2020):
    file_path = '../core/data/retrosheet/gamelogs/GL{}'.format(year)
    df = pd.read_csv(file_path + '.TXT', delimiter = ',', header = 0, names =␣
 ↪columns)
    if os.path.exists(file_path + '.TXT'):
        os.remove(file_path + '.TXT')
    df.to_csv(file_path + '.csv')
```

[109]:
```
df = pd.read_csv('../core/data/retrosheet/gamelogs/GL2015.csv')
```

We don't want every column, so we'll specify exactly which ones to use

[110]:
```
df = df[[
        'date',
        'visit_team',
        'home_team',
        'visit_score',
        'home_score',
        'game_length_outs',
        'day_night',
        'park_id',
        'visit_manager_id',
        'home_manager_id',
        'visit_sp_id',
        'home_sp_id',
        'visit_player_1_id',
        'visit_player_2_id',
        'visit_player_3_id',
        'visit_player_4_id',
        'visit_player_5_id',
        'visit_player_6_id',
        'visit_player_7_id',
        'visit_player_8_id',
        'visit_player_9_id',
        'home_player_1_id',
        'home_player_2_id',
        'home_player_3_id',
```

```
        'home_player_4_id',
        'home_player_5_id',
        'home_player_6_id',
        'home_player_7_id',
        'home_player_8_id',
        'home_player_9_id'
    ]]
```

[111]:
```
df
```

[111]:
```
          date visit_team home_team  visit_score  home_score  \
0     20150406        MIN       DET            0           4
1     20150406        CLE       HOU            0           2
2     20150406        CHA       KCA            1          10
3     20150406        TOR       NYA            6           1
4     20150406        TEX       OAK            0           8
...        ...        ...       ...          ...         ...
2423  20151004        CHN       MIL            3           1
2424  20151004        WAS       NYN            0           1
2425  20151004        MIA       PHI            2           7
2426  20151004        CIN       PIT            0           4
2427  20151004        COL       SFN            7           3

      game_length_outs day_night park_id visit_manager_id home_manager_id  \
0                   51         D   DET05         molip001        ausmb001
1                   51         N   HOU03         frant001        hinca001
2                   51         D   KAN06         ventr001        yoste001
3                   54         D   NYC21         gibbj001        giraj001
4                   51         N   OAK01         banij001        melvb001
...                ...       ...     ...              ...             ...
2423                54         D   MIL06         maddj801        counc001
2424                51         D   NYC20         willm003        collt801
2425                51         D   PHI13         jennd801        mackp101
2426                51         D   PIT08         pricb801        hurdc001
2427                54         D   SFO03         weisw001        bochb002

      ... visit_player_9_id home_player_1_id home_player_2_id  \
0     ...          schaj002         davir003         kinsi001
1     ...          ramij003         altuj001         sprig001
2     ...          johnm006         escoa003         mousm001
3     ...          travd001         ellsj001         gardb001
4     ...          odorr001         gentc001         fulds001
...   ...               ...              ...              ...
2423  ...          hared001         genns001         petes002
2424  ...          roart001         granc001         wrigd002
2425  ...          conla001         galvf001         altha001
2426  ...          smitj004         polag001         harrj002
```

```
2427  ...           bergc001          pagaa001          tomlk001

     home_player_3_id home_player_4_id home_player_5_id home_player_6_id  \
0            cabrm001         martv001         martj006         cespy001
1            valbl001         gatte001         cartc002         castj006
2            cainl001         hosme001         morak001         gorda001
3            beltc001         teixm001         mccab002         headc001
4            zobrb001         butlb003         davii001         lawrb002
...               ...              ...              ...              ...
2423         linda001         davik003         santd002         pereh001
2424         murpd006         cespy001         dudal001         darnt001
2425         franm004         ruf-d001         franj004         blana001
2426         mccua001         walkn001         marts002         alvap001
2427         duffm002         poseb001         parkj002         willm008

     home_player_7_id home_player_8_id home_player_9_id
0            castn001         avila001         iglej001
1            lowrj001         rasmc001         marij002
2            riosa002         peres002         infao001
3            rodra001         drews001         gregd001
4            vogts001         semim001         sogae001
...               ...              ...              ...
2423         seguj002         maldm001         lopej004
2424         confm001         tejar001         degrj001
2425         krate001         ruppc001         buchd001
2426         cervf001         mercj002         happj001
2427         noonn001         willj005         cainm001

[2428 rows x 33 columns]
```

[112]:
```
df['date'] = df['date'].astype(str)
```

'date' isn't very useful, so we'll export it to three separate columns.

[113]:
```
df['year'] = df['date'].str[0:4].astype(int)
df['month'] = df['date'].str[4:6].astype(int)
df['day'] = df['date'].str[6:8].astype(int)
```

[114]:
```
df
```

[114]:
```
          date visit_team home_team  visit_score  home_score  \
0     20150406        MIN       DET            0           4
1     20150406        CLE       HOU            0           2
2     20150406        CHA       KCA            1          10
3     20150406        TOR       NYA            6           1
4     20150406        TEX       OAK            0           8
...        ...        ...       ...          ...         ...
```

```
2423  20151004        CHN       MIL           3           1
2424  20151004        WAS       NYN           0           1
2425  20151004        MIA       PHI           2           7
2426  20151004        CIN       PIT           0           4
2427  20151004        COL       SFN           7           3

      game_length_outs day_night park_id visit_manager_id home_manager_id  \
0                   51         D   DET05        molip001        ausmb001
1                   51         N   HOU03        frant001        hinca001
2                   51         D   KAN06        ventr001        yoste001
3                   54         D   NYC21        gibbj001        giraj001
4                   51         N   OAK01        banij001        melvb001
...                ...       ...     ...             ...             ...
2423                54         D   MIL06        maddj801        counc001
2424                51         D   NYC20        willm003        collt801
2425                51         D   PHI13        jennd801        mackp101
2426                51         D   PIT08        pricb801        hurdc001
2427                54         D   SFO03        weisw001        bochb002

      ... home_player_3_id home_player_4_id home_player_5_id home_player_6_id  \
0     ...         cabrm001         martv001         martj006         cespy001
1     ...         valbl001         gatte001         cartc002         castj006
2     ...         cainl001         hosme001         morak001         gorda001
3     ...         beltc001         teixm001         mccab002         headc001
4     ...         zobrb001         butlb003         davii001         lawrb002
...   ...              ...              ...              ...              ...
2423  ...         linda001         davik003         santd002         pereh001
2424  ...         murpd006         cespy001         dudal001         darnt001
2425  ...         franm004         ruf-d001         franj004         blana001
2426  ...         mccua001         walkn001         marts002         alvap001
2427  ...         duffm002         poseb001         parkj002         willm008

     home_player_7_id home_player_8_id home_player_9_id  year month day
0            castn001         avila001         iglej001  2015     4   6
1            lowrj001         rasmc001         marij002  2015     4   6
2            riosa002         peres002         infao001  2015     4   6
3            rodra001         drews001         gregd001  2015     4   6
4            vogts001         semim001         sogae001  2015     4   6
...               ...              ...              ...   ...   ...  ..
2423         seguj002         maldm001         lopej004  2015    10   4
2424         confm001         tejar001         degrj001  2015    10   4
2425         krate001         ruppc001         buchd001  2015    10   4
2426         cervf001         mercj002         happj001  2015    10   4
2427         noonn001         willj005         cainm001  2015    10   4

[2428 rows x 36 columns]
```

We aren't going to use every column in the final model, but we want to make sure that the ones we will are in the proper format.

[115]:
```
night_game = pd.get_dummies(df['day_night'], drop_first=True)
```

[116]:
```
night_game
```

[116]:
```
      N
0     0
1     1
2     0
3     0
4     1
...  ..
2423  0
2424  0
2425  0
2426  0
2427  0

[2428 rows x 1 columns]
```

[117]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2428 entries, 0 to 2427
Data columns (total 36 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   date                2428 non-null   object
 1   visit_team          2428 non-null   object
 2   home_team           2428 non-null   object
 3   visit_score         2428 non-null   int64
 4   home_score          2428 non-null   int64
 5   game_length_outs    2428 non-null   int64
 6   day_night           2428 non-null   object
 7   park_id             2428 non-null   object
 8   visit_manager_id    2428 non-null   object
 9   home_manager_id     2428 non-null   object
 10  winning_pitcher_id  2428 non-null   object
 11  losing_pitcher_id   2428 non-null   object
 12  saving_pitcher_id   1291 non-null   object
 13  visit_sp_id         2428 non-null   object
 14  home_sp_id          2428 non-null   object
 15  visit_player_1_id   2428 non-null   object
 16  visit_player_2_id   2428 non-null   object
 17  visit_player_3_id   2428 non-null   object
 18  visit_player_4_id   2428 non-null   object
```

```
 19  visit_player_5_id   2428 non-null   object
 20  visit_player_6_id   2428 non-null   object
 21  visit_player_7_id   2428 non-null   object
 22  visit_player_8_id   2428 non-null   object
 23  visit_player_9_id   2428 non-null   object
 24  home_player_1_id    2428 non-null   object
 25  home_player_2_id    2428 non-null   object
 26  home_player_3_id    2428 non-null   object
 27  home_player_4_id    2428 non-null   object
 28  home_player_5_id    2428 non-null   object
 29  home_player_6_id    2428 non-null   object
 30  home_player_7_id    2428 non-null   object
 31  home_player_8_id    2428 non-null   object
 32  home_player_9_id    2428 non-null   object
 33  year                2428 non-null   int64
 34  month               2428 non-null   int64
 35  day                 2428 non-null   int64
dtypes: int64(6), object(30)
memory usage: 683.0+ KB
```

[118]:
```
df.insert(loc=6, column='night_game', value=night_game)
```

[119]:
```
df
```

[119]:
```
          date visit_team home_team  visit_score  home_score  \
0     20150406        MIN       DET            0           4
1     20150406        CLE       HOU            0           2
2     20150406        CHA       KCA            1          10
3     20150406        TOR       NYA            6           1
4     20150406        TEX       OAK            0           8
...        ...        ...       ...          ...         ...
2423  20151004        CHN       MIL            3           1
2424  20151004        WAS       NYN            0           1
2425  20151004        MIA       PHI            2           7
2426  20151004        CIN       PIT            0           4
2427  20151004        COL       SFN            7           3

      game_length_outs  night_game day_night park_id visit_manager_id  ...  \
0                   51           0         D   DET05         molip001  ...
1                   51           1         N   HOU03         frant001  ...
2                   51           0         D   KAN06         ventr001  ...
3                   54           0         D   NYC21         gibbj001  ...
4                   51           1         N   OAK01         banij001  ...
...                ...         ...       ...     ...              ...  ...
2423                54           0         D   MIL06         maddj801  ...
2424                51           0         D   NYC20         willm003  ...
2425                51           0         D   PHI13         jennd801  ...
```

```
2426                  51            0         D    PIT08         pricb801  ...
2427                  54            0         D    SFO03         weisw001  ...

     home_player_3_id home_player_4_id home_player_5_id home_player_6_id  \
0            cabrm001         martv001         martj006         cespy001
1            valbl001         gatte001         cartc002         castj006
2            cainl001         hosme001         morak001         gorda001
3            beltc001         teixm001         mccab002         headc001
4            zobrb001         butlb003         davii001         lawrb002
...               ...              ...              ...              ...
2423         linda001         davik003         santd002         pereh001
2424         murpd006         cespy001         dudal001         darnt001
2425         franm004         ruf-d001         franj004         blana001
2426         mccua001         walkn001         marts002         alvap001
2427         duffm002         poseb001         parkj002         willm008

     home_player_7_id home_player_8_id home_player_9_id  year month day
0            castn001         avila001         iglej001  2015     4   6
1            lowrj001         rasmc001         marij002  2015     4   6
2            riosa002         peres002         infao001  2015     4   6
3            rodra001         drews001         gregd001  2015     4   6
4            vogts001         semim001         sogae001  2015     4   6
...               ...              ...              ...   ...   ...  ..
2423         seguj002         maldm001         lopej004  2015    10   4
2424         confm001         tejar001         degrj001  2015    10   4
2425         krate001         ruppc001         buchd001  2015    10   4
2426         cervf001         mercj002         happj001  2015    10   4
2427         noonn001         willj005         cainm001  2015    10   4

[2428 rows x 37 columns]
```

[120]:
```
to_drop = ['date', 'day_night']
```

[121]:
```
df = df.drop(columns=to_drop)
```

[122]:
```
df
```

[122]:
```
     visit_team home_team  visit_score  home_score  game_length_outs  \
0           MIN       DET            0           4                51
1           CLE       HOU            0           2                51
2           CHA       KCA            1          10                51
3           TOR       NYA            6           1                54
4           TEX       OAK            0           8                51
...         ...       ...          ...         ...               ...
2423        CHN       MIL            3           1                54
2424        WAS       NYN            0           1                51
2425        MIA       PHI            2           7                51
```

```
2426        CIN       PIT            0           4                 51
2427        COL       SFN            7           3                 54

     night_game park_id visit_manager_id home_manager_id winning_pitcher_id  \
0             0   DET05        molip001        ausmb001           pricd001
1             1   HOU03        frant001        hinca001           keucd001
2             0   KAN06        ventr001        yoste001           venty001
3             0   NYC21        gibbj001        giraj001           hutcd001
4             1   OAK01        banij001        melvb001           grays001
...         ...     ...             ...             ...                ...
2423          0   MIL06        maddj801        counc001           hared001
2424          0   NYC20        willm003        collt801           clipt001
2425          0   PHI13        jennd801        mackp101           garcl005
2426          0   PIT08        pricb801        hurdc001           happj001
2427          0   SFO03        weisw001        bochb002           brotr001

     ... home_player_3_id home_player_4_id home_player_5_id home_player_6_id  \
0    ...         cabrm001         martv001         martj006         cespy001
1    ...         valbl001         gatte001         cartc002         castj006
2    ...         cainl001         hosme001         morak001         gorda001
3    ...         beltc001         teixm001         mccab002         headc001
4    ...         zobrb001         butlb003         davii001         lawrb002
...  ...              ...              ...              ...              ...
2423 ...         linda001         davik003         santd002         pereh001
2424 ...         murpd006         cespy001         dudal001         darnt001
2425 ...         franm004         ruf-d001         franj004         blana001
2426 ...         mccua001         walkn001         marts002         alvap001
2427 ...         duffm002         poseb001         parkj002         willm008

     home_player_7_id home_player_8_id home_player_9_id  year month day
0            castn001         avila001         iglej001  2015     4   6
1            lowrj001         rasmc001         marij002  2015     4   6
2            riosa002         peres002         infao001  2015     4   6
3            rodra001         drews001         gregd001  2015     4   6
4            vogts001         semim001         sogae001  2015     4   6
...               ...              ...              ...   ...   ...  ..
2423         seguj002         maldm001         lopej004  2015    10   4
2424         confm001         tejar001         degrj001  2015    10   4
2425         krate001         ruppc001         buchd001  2015    10   4
2426         cervf001         mercj002         happj001  2015    10   4
2427         noonn001         willj005         cainm001  2015    10   4

[2428 rows x 35 columns]
```

[123]:
```
df['visit_team'] = df['visit_team'].apply(get_team)
```

[124]:
```
df['home_team'] = df['home_team'].apply(get_team)
```

Early games only took place during the day, so we need to handle the effects of using one-hot encoding when dropping first with those.

[147]:
```
file_path = '../core/data/retrosheet/gamelogs/GL{}'.format(1919)
df = pd.read_csv(file_path + '.TXT', delimiter = ',', header = 0, names =␣
 ↪columns)
```

[148]:
```
df['day_night'].nunique()
```

[148]:
```
1
```

[149]:
```
pd.get_dummies(df['day_night'])
```

[149]:
```
      D
0     1
1     1
2     1
3     1
4     1
...  ..
1112  1
1113  1
1114  1
1115  1
1116  1

[1117 rows x 1 columns]
```

Final Script

[172]:
```
for year in range(1919, 2020):
    file_path = '../core/data/retrosheet/gamelogs/GL{}'.format(year)
    df = pd.read_csv(file_path + '.TXT', delimiter = ',', header = 0, names =␣
 ↪columns)
    df = df[[
        'date',
        'visit_team',
        'home_team',
        'visit_score',
        'home_score',
        'game_length_outs',
        'day_night',
        'park_id',
        'visit_manager_id',
        'home_manager_id',
        'winning_pitcher_id',
        'losing_pitcher_id',
        'saving_pitcher_id',
```

```
        'visit_sp_id',
        'home_sp_id',
        'visit_player_1_id',
        'visit_player_2_id',
        'visit_player_3_id',
        'visit_player_4_id',
        'visit_player_5_id',
        'visit_player_6_id',
        'visit_player_7_id',
        'visit_player_8_id',
        'visit_player_9_id',
        'home_player_1_id',
        'home_player_2_id',
        'home_player_3_id',
        'home_player_4_id',
        'home_player_5_id',
        'home_player_6_id',
        'home_player_7_id',
        'home_player_8_id',
        'home_player_9_id',
    ]]
    df['date'] = df['date'].astype(str)
    df['year'] = df['date'].str[0:4].astype(int)
    df['month'] = df['date'].str[4:6].astype(int)
    df['day'] = df['date'].str[6:8].astype(int)
    night_game = pd.get_dummies(df['day_night'], drop_first=(df['day_night'].
↪nunique() > 1))
    df.insert(loc=6, column='night_game', value=night_game)
    df = df.drop(columns=['date', 'day_night'])
    df['visit_team'] = df['visit_team'].apply(get_team)
    df['home_team'] = df['home_team'].apply(get_team)
    if os.path.exists(file_path + '.TXT'):
        os.remove(file_path + '.TXT')
    df.to_csv(file_path + '.csv', index=False)
```

[170]:
```
df = pd.read_csv('../core/data/retrosheet/gamelogs/GL2015.csv')
```

[173]:
```
df
```

[173]:
```
     visit_team home_team  visit_score  home_score  game_length_outs  \
0           MIN       DET            0           4                51
1           CLE       HOU            0           2                51
2           CHW       KCR            1          10                51
3           TOR       NYY            6           1                54
4           TEX       OAK            0           8                51
...         ...       ...          ...         ...               ...
2423        CHC       MIL            3           1                54
```

```
2424        WSN      NYM           0           1                 51
2425        FLA      PHI           2           7                 51
2426        CIN      PIT           0           4                 51
2427        COL      SFG           7           3                 54

      night_game park_id visit_manager_id home_manager_id winning_pitcher_id  \
0              0   DET05         molip001        ausmb001           pricd001
1              1   HOU03         frant001        hinca001           keucd001
2              0   KAN06         ventr001        yoste001           venty001
3              0   NYC21         gibbj001        giraj001           hutcd001
4              1   OAK01         banij001        melvb001           grays001
...          ...     ...              ...             ...                ...
2423           0   MIL06         maddj801        counc001           hared001
2424           0   NYC20         willm003        collt801           clipt001
2425           0   PHI13         jennd801        mackp101           garcl005
2426           0   PIT08         pricb801        hurdc001           happj001
2427           0   SFO03         weisw001        bochb002           brotr001

      ... home_player_3_id home_player_4_id home_player_5_id home_player_6_id  \
0     ...         cabrm001         martv001         martj006         cespy001
1     ...         valbl001         gatte001         cartc002         castj006
2     ...         cainl001         hosme001         morak001         gorda001
3     ...         beltc001         teixm001         mccab002         headc001
4     ...         zobrb001         butlb003         davii001         lawrb002
...   ...              ...              ...              ...              ...
2423  ...         linda001         davik003         santd002         pereh001
2424  ...         murpd006         cespy001         dudal001         darnt001
2425  ...         franm004         ruf-d001         franj004         blana001
2426  ...         mccua001         walkn001         marts002         alvap001
2427  ...         duffm002         poseb001         parkj002         willm008

     home_player_7_id home_player_8_id home_player_9_id  year month day
0            castn001         avila001         iglej001  2015     4   6
1            lowrj001         rasmc001         marij002  2015     4   6
2            riosa002         peres002         infao001  2015     4   6
3            rodra001         drews001         gregd001  2015     4   6
4            vogts001         semim001         sogae001  2015     4   6
...               ...              ...              ...   ...   ...  ..
2423         seguj002         maldm001         lopej004  2015    10   4
2424         confm001         tejar001         degrj001  2015    10   4
2425         krate001         ruppc001         buchd001  2015    10   4
2426         cervf001         mercj002         happj001  2015    10   4
2427         noonn001         willj005         cainm001  2015    10   4

[2428 rows x 35 columns]
```

When we do the actual script, we don't want the column names hardcoded into it. So I've pasted those to .csv files but I need to process them a bit.

[252]:
```
gla = pd.read_csv('../core/data/retrosheet/rs_gl_cols_all.csv', header=None)
gl = pd.read_csv('../core/data/retrosheet/rs_gl_cols.csv', header=None)
```

[254]:
```
gl
```

[254]:
```
                                 0
0                           'date',
1                     'visit_team',
2                      'home_team',
3                    'visit_score',
4                     'home_score',
5               'game_length_outs',
6                      'day_night',
7                        'park_id',
8               'visit_manager_id',
9                'home_manager_id',
10            'winning_pitcher_id',
11             'losing_pitcher_id',
12             'saving_pitcher_id',
13                   'visit_sp_id',
14                    'home_sp_id',
15             'visit_player_1_id',
16             'visit_player_2_id',
17             'visit_player_3_id',
18             'visit_player_4_id',
19             'visit_player_5_id',
20             'visit_player_6_id',
21             'visit_player_7_id',
22             'visit_player_8_id',
23             'visit_player_9_id',
24              'home_player_1_id',
25              'home_player_2_id',
26              'home_player_3_id',
27              'home_player_4_id',
28              'home_player_5_id',
29              'home_player_6_id',
30              'home_player_7_id',
31              'home_player_8_id',
32               'home_player_9_id'
```

We need to get rid of whitespace, commas and quotation marks.

[189]:
```
gla.iloc[156][0].replace(',', '')
```

[189]:
```
"    'home_player_9_id'"
```

[190]:
```python
from functools import reduce
```

[241]:
```python
def trim_cell(cell):
    replacements = {' ': '', "'": '', ',': ''}
    string = cell[0]
    return reduce(lambda a, kv: a.replace(*kv), replacements.items(), string)
```

[200]:
```python
print(trim_cell(gla.iloc[156]))
```

```
home_player_9_id
```

[255]:
```python
gla = gla.apply(trim_cell, axis=1)
```

[256]:
```python
type(gla)
```

[256]:
```
pandas.core.series.Series
```

[257]:
```python
gl = gl.apply(trim_cell, axis=1)
```

[258]:
```python
gla.to_csv('../core/data/retrosheet/rs_gl_cols_all.csv', header=None)
gl.to_csv('../core/data/retrosheet/rs_gl_cols.csv', header=None)
```

[259]:
```python
gla = pd.read_csv('../core/data/retrosheet/rs_gl_cols_all.csv', header=None)
gl = pd.read_csv('../core/data/retrosheet/rs_gl_cols.csv', header=None)
```

[260]:
```python
gl
```

[260]:

	0	1
0	0	date
1	1	visit_team
2	2	home_team
3	3	visit_score
4	4	home_score
5	5	game_length_outs
6	6	day_night
7	7	park_id
8	8	visit_manager_id
9	9	home_manager_id
10	10	winning_pitcher_id
11	11	losing_pitcher_id
12	12	saving_pitcher_id
13	13	visit_sp_id
14	14	home_sp_id
15	15	visit_player_1_id
16	16	visit_player_2_id
17	17	visit_player_3_id
18	18	visit_player_4_id

```
19  19   visit_player_5_id
20  20   visit_player_6_id
21  21   visit_player_7_id
22  22   visit_player_8_id
23  23   visit_player_9_id
24  24    home_player_1_id
25  25    home_player_2_id
26  26    home_player_3_id
27  27    home_player_4_id
28  28    home_player_5_id
29  29    home_player_6_id
30  30    home_player_7_id
31  31    home_player_8_id
32  32    home_player_9_id
```

[261]:
```
gla[1].tolist()
```

[261]:
```
['date',
 'game_number',
 'day_of_week',
 'visit_team',
 'visit_league',
 'visit_game_number',
 'home_team',
 'home_league',
 'home_game_number',
 'visit_score',
 'home_score',
 'game_length_outs',
 'day_night',
 'completion_info',
 'forfeit_info',
 'protest_info',
 'park_id',
 'attendance',
 'time_minutes',
 'visit_line_score',
 'home_line_score',
 'visit_ab',
 'visit_h',
 'visit_2b',
 'visit_3b',
 'visit_hr',
 'visit_rbi',
 'visit_sh',
 'visit_sf',
 'visit_hbp',
```

```
'visit_bb',
'visit_ibb',
'visit_k',
'visit_sb',
'visit_cs',
'visit_gidp',
'visit_ci',
'visit_lob',
'visit_pitchers_used',
'visit_individual_er',
'visit_team_er',
'visit_wp',
'visit_bk',
'visit_po',
'visit_assists',
'visit_e',
'visit_passed_balls',
'visit_double_plays',
'visit_triple_plays',
'home_ab',
'home_h',
'home_2b',
'home_3b',
'home_hr',
'home_rbi',
'home_sh',
'home_sf',
'home_hbp',
'home_bb',
'home_ibb',
'home_k',
'home_sb',
'home_cs',
'home_gidp',
'home_ci',
'home_lob',
'home_pitchers_used',
'home_individual_er',
'home_team_er',
'home_wp',
'home_bk',
'home_po',
'home_assists',
'home_e',
'home_passed_balls',
'home_double_plays',
'home_triple_plays',
```

```
'hp_ump_id',
'hp_ump_name',
'1b_ump_id',
'1b_ump_name',
'2b_ump_id',
'2b_ump_name',
'3b_ump_id',
'3b_ump_name',
'lf_ump_id',
'lf_ump_name',
'rf_ump_id',
'rf_ump_name',
'visit_manager_id',
'visit_manager_name',
'home_manager_id',
'home_manager_name',
'winning_pitcher_id',
'winning_pitcher_name',
'losing_pitcher_id',
'losing_pitcher_name',
'saving_pitcher_id',
'saving_pitcher_name',
'winning_rbi_batter_id',
'winning_rbi_batter_name',
'visit_sp_id',
'visit_sp_name',
'home_sp_id',
'home_sp_name',
'visit_player_1_id',
'visit_player_1_name',
'visit_player_1_pos',
'visit_player_2_id',
'visit_player_2_name',
'visit_player_2_pos',
'visit_player_3_id',
'visit_player_3_name',
'visit_player_3_pos',
'visit_player_4_id',
'visit_player_4_name',
'visit_player_4_pos',
'visit_player_5_id',
'visit_player_5_name',
'visit_player_5_pos',
'visit_player_6_id',
'visit_player_6_name',
'visit_player_6_pos',
'visit_player_7_id',
```

```
 'visit_player_7_name',
 'visit_player_7_pos',
 'visit_player_8_id',
 'visit_player_8_name',
 'visit_player_8_pos',
 'visit_player_9_id',
 'visit_player_9_name',
 'visit_player_9_pos',
 'home_player_1_id',
 'home_player_1_name',
 'home_player_1_pos',
 'home_player_2_id',
 'home_player_2_name',
 'home_player_2_pos',
 'home_player_3_id',
 'home_player_3_name',
 'home_player_3_pos',
 'home_player_4_id',
 'home_player_4_name',
 'home_player_4_pos',
 'home_player_5_id',
 'home_player_5_name',
 'home_player_5_pos',
 'home_player_6_id',
 'home_player_6_name',
 'home_player_6_pos',
 'home_player_7_id',
 'home_player_7_name',
 'home_player_7_pos',
 'home_player_8_id',
 'home_player_8_name',
 'home_player_8_pos',
 'home_player_9_id',
 'home_player_9_name',
 'home_player_9_pos',
 'additional_info',
 'acquisition_info']
```

[247]:
```
gl
```

[247]:

	0
0	visit_team
1	home_team
2	visit_score
3	home_score
4	game_length_outs
5	day_night

```
6                park_id
7       visit_manager_id
8        home_manager_id
9    winning_pitcher_id
10    losing_pitcher_id
11    saving_pitcher_id
12          visit_sp_id
13           home_sp_id
14    visit_player_1_id
15    visit_player_2_id
16    visit_player_3_id
17    visit_player_4_id
18    visit_player_5_id
19    visit_player_6_id
20    visit_player_7_id
21    visit_player_8_id
22    visit_player_9_id
23     home_player_1_id
24     home_player_2_id
25     home_player_3_id
26     home_player_4_id
27     home_player_5_id
28     home_player_6_id
29     home_player_7_id
30     home_player_8_id
31     home_player_9_id
```

[ ]:

```

```

create_gamelog_tensors

May 10, 2020

[13]:
```
import os
import pandas as pd
import numpy as np
from tensorflow.keras.models import load_model
from joblib import load
pd.options.mode.chained_assignment = None  # default='warn'
```

[14]:
```
bat = load_model('../core/models/model_batting.h5')
pitch = load_model('../core/models/model_pitching.h5')
bat_scaler = load('../core/models/batting_scaler.save')
pitch_scaler = load('../core/models/pitching_scaler.save')
```

[15]:
```
gl = pd.read_csv('../core/data/retrosheet/gamelogs/GL2015.csv')
```

[16]:
```
gl
```

[16]:
```
     visit_team home_team  visit_score  home_score  game_length_outs  \
0           MIN       DET            0           4                51
1           CLE       HOU            0           2                51
2           CHW       KCR            1          10                51
3           TOR       NYY            6           1                54
4           TEX       OAK            0           8                51
...         ...       ...          ...         ...               ...
2423        CHC       MIL            3           1                54
2424        WSN       NYM            0           1                51
2425        FLA       PHI            2           7                51
2426        CIN       PIT            0           4                51
2427        COL       SFG            7           3                54

      night_game park_id visit_manager_id home_manager_id visit_sp_id  ...  \
0              0   DET05        molip001        ausmb001     hughp001  ...
1              1   HOU03        frant001        hinca001     klubc001  ...
2              0   KAN06        ventr001        yoste001     samaj001  ...
3              0   NYC21        gibbj001        giraj001     hutcd001  ...
4              1   OAK01        banij001        melvb001     gally001  ...
...          ...     ...             ...             ...          ...  ...
2423           0   MIL06        maddj801        counc001     hared001  ...
```

```
2424            0   NYC20         willm003        collt801     roart001   ...
2425            0   PHI13         jennd801        mackp101     conla001   ...
2426            0   PIT08         pricb801        hurdc001     smitj004   ...
2427            0   SF003         weisw001        bochb002     bergc001   ...

     home_player_4_id home_player_5_id home_player_6_id home_player_7_id  \
0            martv001         martj006         cespy001         castn001
1            gatte001         cartc002         castj006         lowrj001
2            hosme001         morak001         gorda001         riosa002
3            teixm001         mccab002         headc001         rodra001
4            butlb003         davii001         lawrb002         vogts001
...               ...              ...              ...              ...
2423         davik003         santd002         pereh001         seguj002
2424         cespy001         dudal001         darnt001         confm001
2425         ruf-d001         franj004         blana001         krate001
2426         walkn001         marts002         alvap001         cervf001
2427         poseb001         parkj002         willm008         noonn001

     home_player_8_id home_player_9_id  year month day home_win
0            avila001         iglej001  2015     4   6        1
1            rasmc001         marij002  2015     4   6        1
2            peres002         infao001  2015     4   6        1
3            drews001         gregd001  2015     4   6        0
4            semim001         sogae001  2015     4   6        1
...               ...              ...   ...   ...  ..      ...
2423         maldm001         lopej004  2015    10   4        0
2424         tejar001         degrj001  2015    10   4        1
2425         ruppc001         buchd001  2015    10   4        1
2426         mercj002         happj001  2015    10   4        1
2427         willj005         cainm001  2015    10   4        0

[2428 rows x 33 columns]
```

[17]:
```
columns = {
    'batting': [],
    'pitching': []
}
```

[18]:
```
batters = pd.read_csv('../core/output/batters.csv')
batter_years = pd.read_csv('../core/output/batting.csv')
batters_not_counted = list(batter_years[~batter_years['retroID']
                                        .isin(batters['retroID'])]['retroID'].
 ↪values)
pitchers = pd.read_csv('../core/output/pitchers.csv')
pitcher_years = pd.read_csv('../core/output/pitching.csv')
bat_scaler = load('../core/models/batting_scaler.save')
pitch_scaler = load('../core/models/pitching_scaler.save')
```

```
scalers = {
    'batting': bat_scaler,
    'pitching': pitch_scaler
}
career_features = {
    'batting': [
        'G', 'AB', 'PA', 'R', 'H', '1B', '2B', '3B',
        'HR', 'RBI', 'SB', 'CS', 'BB', 'SO', 'IBB',
        'HBP', 'SH', 'SF', 'GIDP'
    ],
    'pitching': [
        'CG', 'SHO', 'H', 'ER', 'HR', 'BB', 'SO',
        'BAOpp', 'ERA', 'IBB', 'WP', 'HBP', 'BK',
        'BFP', 'GF', 'R', 'SH', 'SF', 'GIDP'
    ]
}
unwanted_features = {
    'batting': ['retroID', 'G', 'AB', '1B', 'RBI', 'wOBA', 'Batting'],
    'pitching': ['IPouts', 'BFP', 'R', 'Pitching']
}
players = {
    'batting': {
        'players': batters,
        'years': batter_years
    },
    'pitching': {
        'players': pitchers,
        'years': pitcher_years
    }
}
```

[19]:
```
def to_tensor_input(scaler, player, label):
    scalers[label] = scaler
    return scaler.transform(player.values.reshape(-1, player.shape[0]))[0]


def convert_single_player(retro_id, year, player_type_label):
    scaler = scalers[player_type_label]
    if retro_id in batters_not_counted:
        return np.zeros(shape=(1, 30))
    player_table = players[player_type_label]['players']
    player_so_far_table = players[player_type_label]['years']
    player = player_table[player_table['retroID'] == retro_id]
    player_so_far = player_so_far_table[(player_so_far_table['retroID'] ==␣
 ↪retro_id)
                                        & (player_so_far_table['yearID'] <=␣
 ↪year)]
```

```
    if not player.size | player_so_far.size:
        print('Handled: {}'.format(retro_id))
        return np.zeros(shape=(1, 30))
    player_so_far = player_so_far.groupby('retroID').sum()
    features = career_features[player_type_label]
    try:
        for column in player[features]:
            player.iloc[0][column] = player_so_far.iloc[0][column]
    except:
        print(retro_id)
    player_columns_to_drop = unwanted_features[player_type_label]
    player = player.drop(columns=player_columns_to_drop)
    if not len(list(columns[player_type_label])):
        columns[player_type_label] = player.columns
    return to_tensor_input(scaler, player.T, player_type_label)


def get_batter_as_tensor_input(batter, year):
    scaler = scalers['batting']
    player = batters[batters['retroID'] == batter]
    player_so_far = batter_years[(batter_years['retroID'] == batter)
                                 & (batter_years['yearID'] <= year)]
    player_so_far = player_so_far.groupby('retroID').sum()
    features = ['G', 'AB', 'PA', 'R', 'H', '1B', '2B', '3B',
                'HR', 'RBI', 'SB', 'CS', 'BB', 'SO', 'IBB',
                'HBP', 'SH', 'SF', 'GIDP']
    for column in player[features]:
        player.iloc[0][column] = player_so_far.iloc[0][column]
    player_columns_to_drop = ['retroID', 'wOBA', 'Batting']
    player = player.drop(columns=player_columns_to_drop)
    return to_tensor_input(scaler, player, 'batting')
```

[20]:
```
convert_single_player('bettm001', 2015, 'batting')
```

[20]:
```
array([0.42623   , 0.3       , 0.        , 0.        , 0.        ,
       0.        , 1.        , 0.        , 0.        , 0.        ,
       0.        , 0.16129032, 0.2288002 , 0.2671024 , 0.22673872,
       0.30697051, 0.13612565, 0.1824147 , 0.08961593, 0.07462687,
       0.14503518, 0.17866769, 0.03633721, 0.06666667, 0.01486989,
       0.25      , 0.10379747, 0.        , 0.21133094, 0.26473988])
```

[21]:
```
gl.iloc[43]
```

[21]:
```
visit_team               SFG
home_team                SDP
visit_score                1
home_score                 0
```

```
game_length_outs            72
night_game                   0
park_id                  SAN02
visit_manager_id      bochb002
home_manager_id       blacb001
visit_sp_id           hudst001
home_sp_id            kenni001
visit_player_1_id     aokin001
visit_player_2_id     panij002
visit_player_3_id     pagaa001
visit_player_4_id     poseb001
visit_player_5_id     crawb001
visit_player_6_id     mcgec001
visit_player_7_id     blang001
visit_player_8_id     ariaj001
visit_player_9_id     hudst001
home_player_1_id      myerw001
home_player_2_id      norrd001
home_player_3_id      kempm001
home_player_4_id      uptoj001
home_player_5_id      middw001
home_player_6_id      alony001
home_player_7_id      gyorj001
home_player_8_id      amara001
home_player_9_id      kenni001
year                      2015
month                        4
day                          9
home_win                     0
Name: 43, dtype: object
```

[22]:
```
v1 = gl.iloc[0]['visit_player_1_id']
```

[23]:
```
v1
```

[23]:
```
'santd001'
```

[24]:
```
visit_id = []
home_id = []
```

[25]:
```
for i in range(1, 10):
    visit_id.append(gl.iloc[43]['visit_player_{}_id'.format(i)])
    home_id.append(gl.iloc[43]['home_player_{}_id'.format(i)])
```

[26]:
```
visit_id
```

[26]:
```
['aokin001',
 'panij002',
 'pagaa001',
 'poseb001',
 'crawb001',
 'mcgec001',
 'blang001',
 'ariaj001',
 'hudst001']
```

[27]:
```
gl.iloc[0]['year']
```

[27]:
```
2015
```

[28]:
```
visit = []
home = []
year = gl.iloc[43]['year']
for index in range(0, 9):
    vrid = visit_id[index]
#     vpos = 'pitching' if vrid == gl.iloc[0]['visit_sp_id'] else 'batting'
    vplayer = convert_single_player(vrid, year, 'batting')
    visit.append(vplayer)
    hrid = home_id[index]
#     hpos = 'pitching' if hrid == gl.iloc[0]['home_sp_id'] else 'batting'
    hplayer = convert_single_player(hrid, year, 'batting')
    home.append(hplayer)
```

[29]:
```
visit[0].shape
```

[29]:
```
(30,)
```

[30]:
```
home[0].shape
```

[30]:
```
(30,)
```

[31]:
```
gl.columns
```

[31]:
```
Index(['visit_team', 'home_team', 'visit_score', 'home_score',
       'game_length_outs', 'night_game', 'park_id', 'visit_manager_id',
       'home_manager_id', 'visit_sp_id', 'home_sp_id', 'visit_player_1_id',
       'visit_player_2_id', 'visit_player_3_id', 'visit_player_4_id',
       'visit_player_5_id', 'visit_player_6_id', 'visit_player_7_id',
       'visit_player_8_id', 'visit_player_9_id', 'home_player_1_id',
       'home_player_2_id', 'home_player_3_id', 'home_player_4_id',
       'home_player_5_id', 'home_player_6_id', 'home_player_7_id',
       'home_player_8_id', 'home_player_9_id', 'year', 'month', 'day',
       'home_win'],
```

```
      dtype='object')
```

[32]:
```python
batters = visit + home
```

[33]:
```python
dfb = pd.DataFrame(batters, columns=columns['batting'])
```

[34]:
```python
dfb
```

[34]:
```
      weight  height  pos_1B  pos_2B  pos_3B  pos_C  pos_OF  pos_P  pos_SS  \
0   0.426230    0.30     0.0     0.0     0.0    0.0     1.0    0.0     0.0
1   0.508197    0.50     0.0     1.0     0.0    0.0     0.0    0.0     0.0
2   0.508197    0.55     0.0     0.0     0.0    0.0     1.0    0.0     0.0
3   0.549180    0.50     0.0     0.0     0.0    1.0     0.0    0.0     0.0
4   0.618852    0.55     0.0     0.0     0.0    0.0     0.0    0.0     1.0
5   0.590164    0.50     0.0     0.0     1.0    0.0     0.0    0.0     0.0
6   0.454918    0.35     0.0     0.0     0.0    0.0     1.0    0.0     0.0
7   0.446721    0.50     0.0     1.0     0.0    0.0     0.0    0.0     0.0
8   0.405738    0.50     0.0     0.0     0.0    0.0     0.0    1.0     0.0
9   0.528689    0.60     1.0     0.0     0.0    0.0     0.0    0.0     0.0
10  0.651639    0.45     0.0     0.0     0.0    1.0     0.0    0.0     0.0
11  0.610656    0.65     0.0     0.0     0.0    0.0     1.0    0.0     0.0
12  0.569672    0.50     0.0     0.0     0.0    0.0     1.0    0.0     0.0
13  0.590164    0.60     0.0     0.0     1.0    0.0     0.0    0.0     0.0
14  0.631148    0.50     1.0     0.0     0.0    0.0     0.0    0.0     0.0
15  0.569672    0.35     0.0     1.0     0.0    0.0     0.0    0.0     0.0
16  0.344262    0.15     0.0     1.0     0.0    0.0     0.0    0.0     0.0
17  0.528689    0.45     0.0     0.0     0.0    0.0     0.0    1.0     0.0

    bats_L  ...        BB        SO       IBB       HBP        SH        SF  \
0      1.0  ...  0.091478  0.099345  0.004360  0.168421  0.111524  0.117188
1      1.0  ...  0.085614  0.097420  0.020349  0.073684  0.048327  0.203125
2      0.0  ...  0.124707  0.245668  0.026163  0.014035  0.078067  0.265625
3      0.0  ...  0.189210  0.244128  0.090116  0.147368  0.003717  0.398438
4      1.0  ...  0.155590  0.360801  0.091570  0.129825  0.022305  0.367188
5      0.0  ...  0.097342  0.199076  0.020349  0.028070  0.000000  0.242188
6      1.0  ...  0.141908  0.254524  0.030523  0.052632  0.096654  0.117188
7      0.0  ...  0.014464  0.057759  0.010174  0.028070  0.044610  0.070312
8      0.0  ...  0.010164  0.073161  0.000000  0.007018  0.249071  0.015625
9      0.0  ...  0.122361  0.322680  0.020349  0.045614  0.003717  0.156250
10     0.0  ...  0.076231  0.208317  0.014535  0.066667  0.007435  0.093750
11     0.0  ...  0.196638  0.616095  0.098837  0.091228  0.003717  0.562500
12     0.0  ...  0.284988  0.692337  0.071221  0.235088  0.011152  0.429688
13     0.0  ...  0.025020  0.125144  0.005814  0.031579  0.003717  0.078125
14     1.0  ...  0.143081  0.249519  0.042151  0.056140  0.003717  0.218750
15     0.0  ...  0.091087  0.243358  0.007267  0.059649  0.000000  0.171875
16     1.0  ...  0.042611  0.113593  0.018895  0.014035  0.085502  0.117188
17     0.0  ...  0.012901  0.059299  0.000000  0.003509  0.156134  0.015625
```

```
        GIDP   NL      wRC+       WAR
0   0.124051  1.0  0.184353  0.105202
1   0.129114  1.0  0.177158  0.103468
2   0.136709  1.0  0.181655  0.157803
3   0.374684  1.0  0.205036  0.354335
4   0.235443  1.0  0.173561  0.172254
5   0.291139  1.0  0.171763  0.072832
6   0.088608  1.0  0.173561  0.101734
7   0.060759  1.0  0.158273  0.051445
8   0.030380  1.0  0.095324  0.058382
9   0.179747  1.0  0.186151  0.104624
10  0.106329  1.0  0.171763  0.105202
11  0.453165  1.0  0.198741  0.206358
12  0.313924  1.0  0.197842  0.262428
13  0.081013  1.0  0.158273  0.055491
14  0.237975  1.0  0.181655  0.080925
15  0.184810  1.0  0.179856  0.100000
16  0.081013  1.0  0.147482  0.036416
17  0.007595  1.0  0.097122  0.055491

[18 rows x 30 columns]
```

[35]:
```
btensor = [dfb, gl.iloc[43]['home_win']]
```

[36]:
```
# btensor
```

[37]:
```
gl.shape
```

[37]: (2428, 33)

[38]:
```
gl.shape[0]
```

[38]: 2428

[39]:
```
players['batting']['players']['retroID'].str.contains('aardd001').sum() == 1
```

[39]: True

Modular script to handle all gamelogs

[40]:
```
cols = list(columns['batting'].values) + ['Result']
for year in range(1919, 2020):
    df = pd.DataFrame()
    print('{}'.format(year))
#     gl = pd.read_csv('../core/data/retrosheet/gamelogs/GL{}.csv'.format(year))
#     for index in range(0, gl.shape[0]):
```

```
#         visit_id = []
#         home_id = []
#         for i in range(1, 10):
#             visit_id.append(gl.iloc[index]['visit_player_{}_id'.format(i)])
#             home_id.append(gl.iloc[index]['home_player_{}_id'.format(i)])
#         visit = []
#         home = []
#         for i in range(0, 9):
#             vrid = visit_id[i]
#             vplayer = convert_single_player(vrid, year, 'batting')
#             visit.append(vplayer)
#             hrid = home_id[i]
#             hplayer = convert_single_player(hrid, year, 'batting')
#             home.append(hplayer)
#         batters = list(np.append(np.array(visit + home).flatten(), gl.
↪iloc[index]['home_win']))
#         try:
#             bat_df = pd.DataFrame(batters)
#         except:
#             print('{0}\n{1}'.format(vrid))
#         df = df.append(bat_df.T)

#     if not os.path.exists('../core/tensors/games/'):
#         os.mkdir('../core/tensors/games/')
#     df.to_csv('../core/tensors/games/{0}.csv'.format(str(year)), index=False,␣
↪header=None)
```

1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938

```
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
```

[ ]: