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	

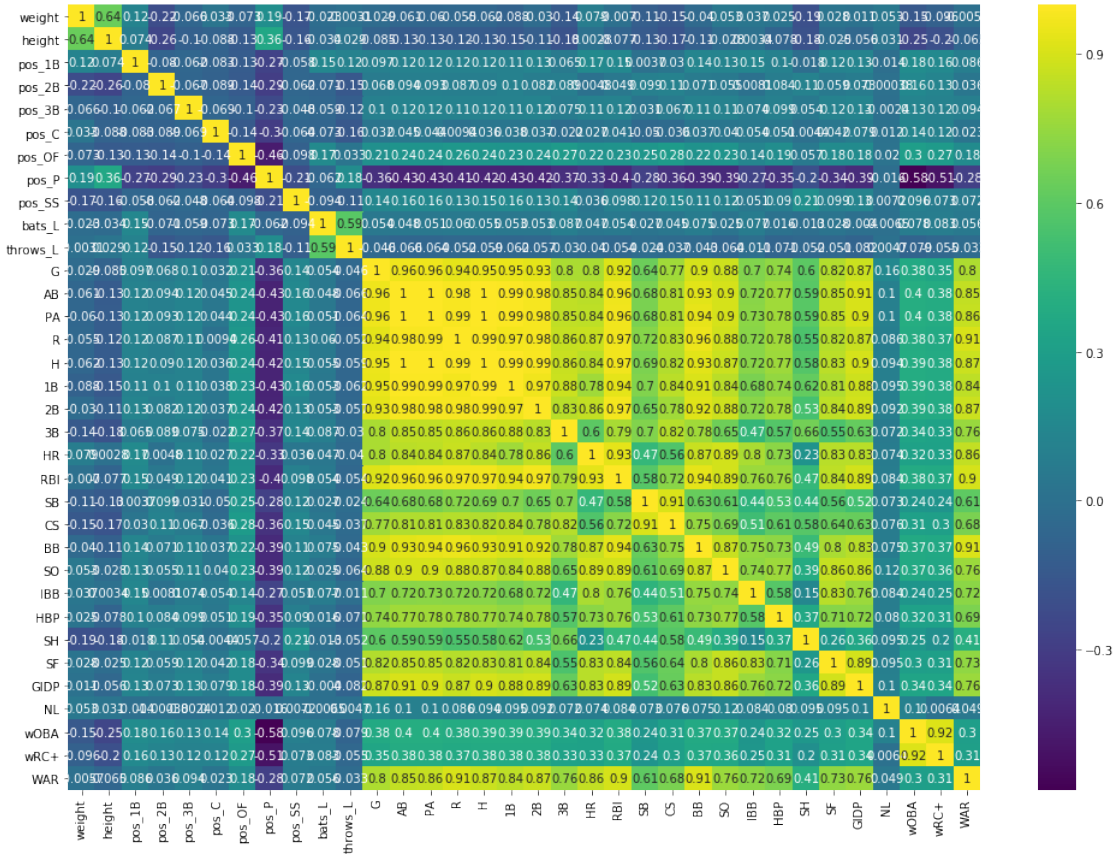
```
15290  0.487705    0.65       0       0       0      0       0      1       0
15291  0.397541    0.45       0       0       0      0       0      0       1
15292  0.467213    0.60       0       0       0      0       0      1       0

       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]:
```
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)
```

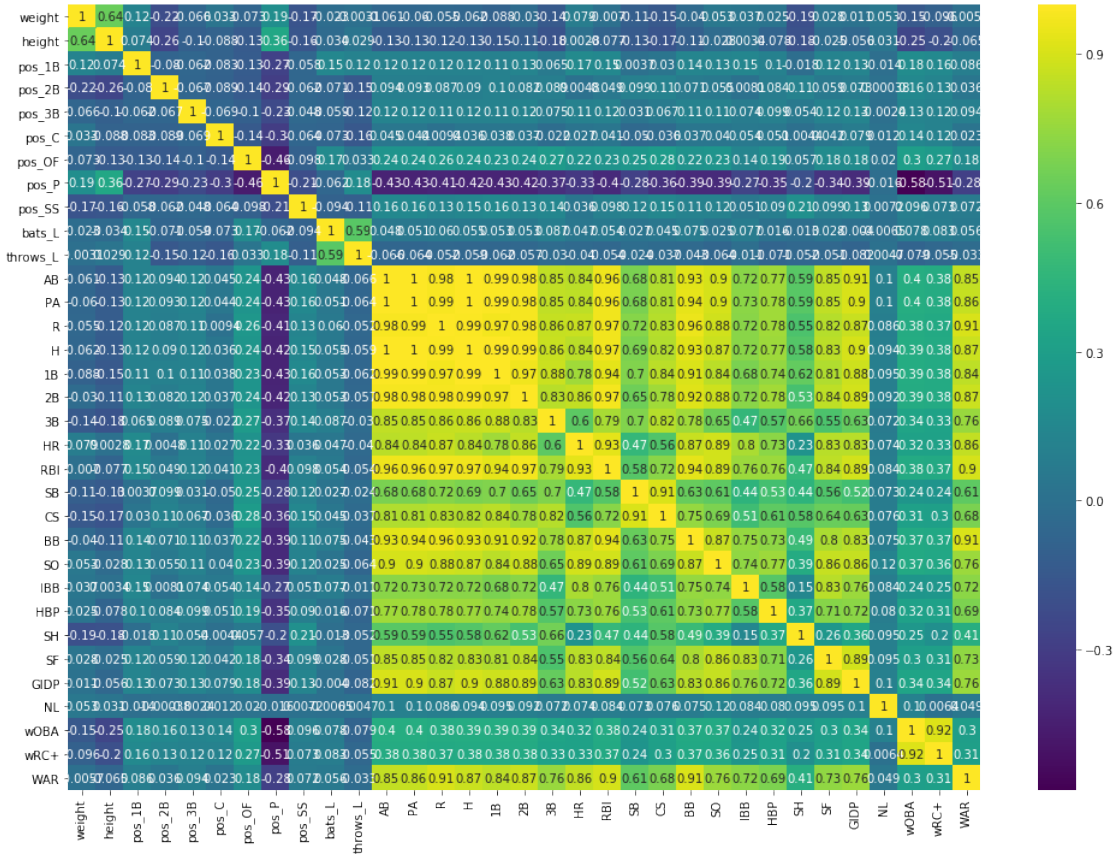


We see a high correlation between G (games) and AB/PA (at-bats/plate appearances). It makes sense that we can drop the G column.

[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)

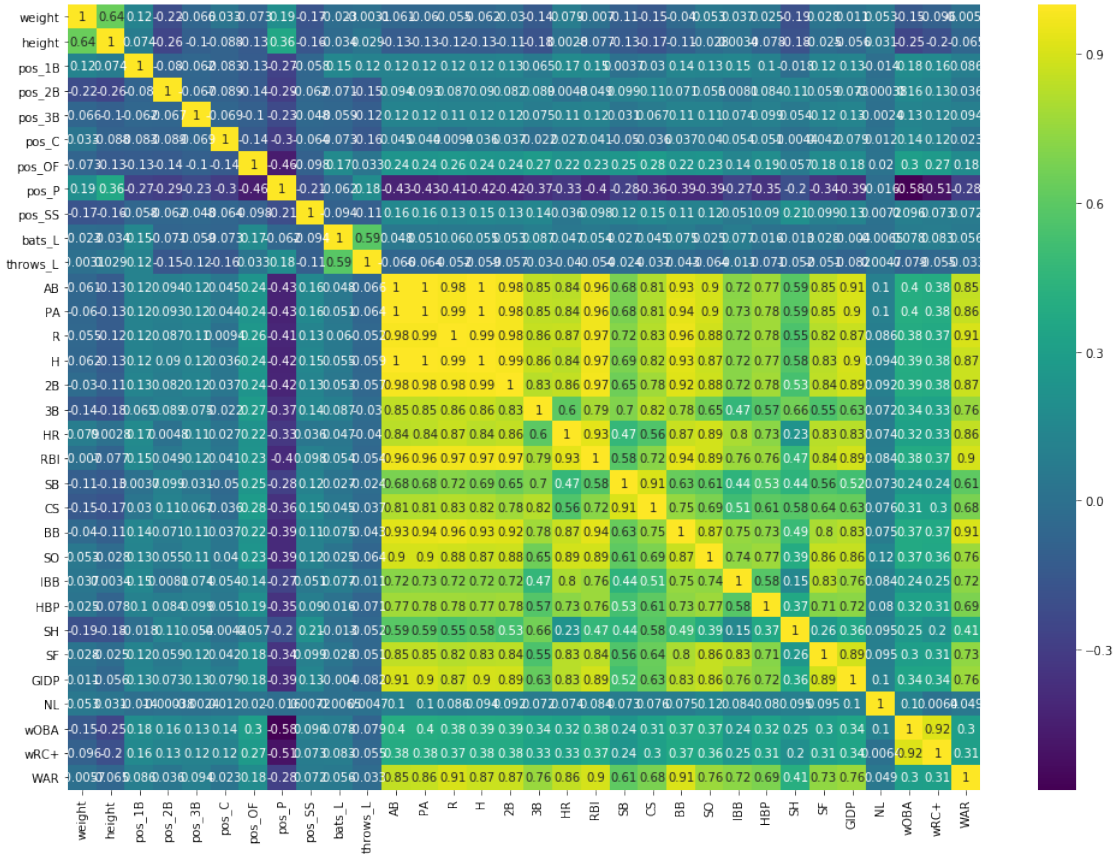


There's obviously a high correlation in H (hits) and 1B/2B/3B/HR (singles, doubles, triples and home runs). We added 1B as a column to help with statistics but it's unnecessary now - the relationship between hits and types of hits will be preserved in the model.

[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)

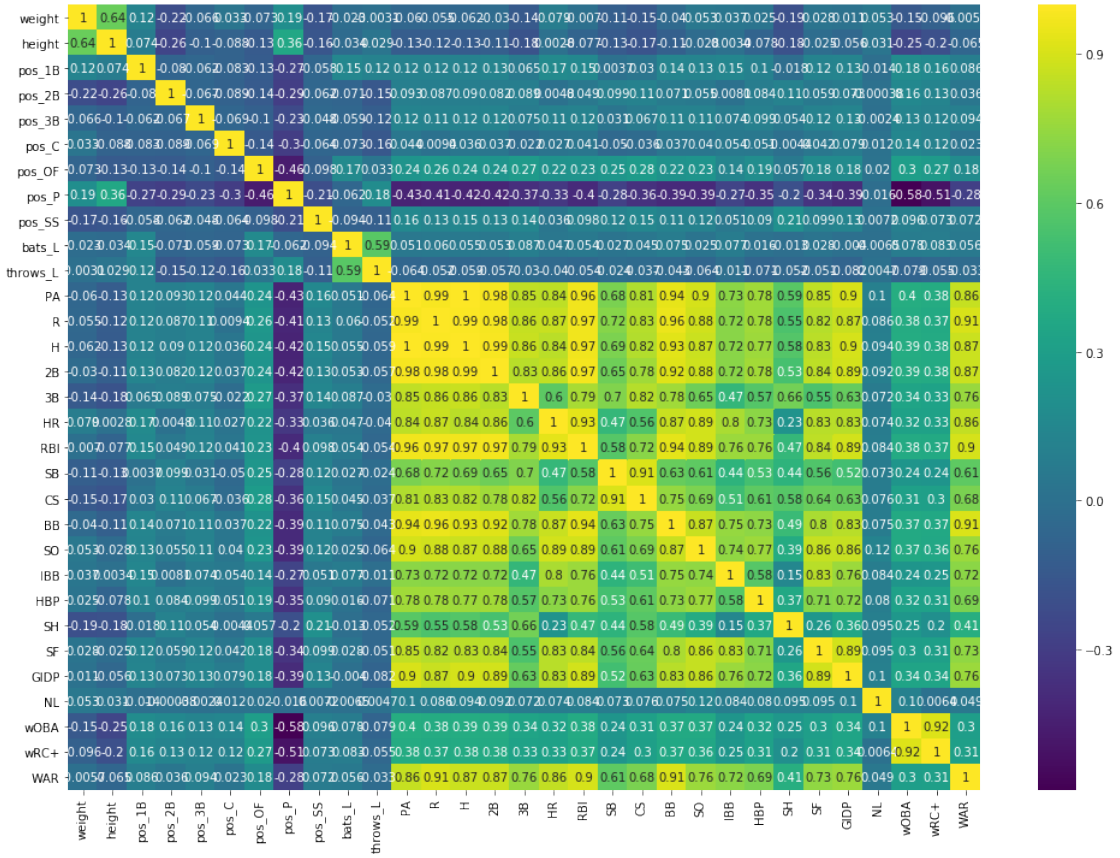


We have total correlation between AB (at-bats) and PA (plate-appearances). This makes sense, because PA is just AB with some other situations added in. PA is more robust and is better related to overall output (since it includes sacrifices, hits-by-pitch and walks) so we'll keep PA and get rid of AB.

[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)



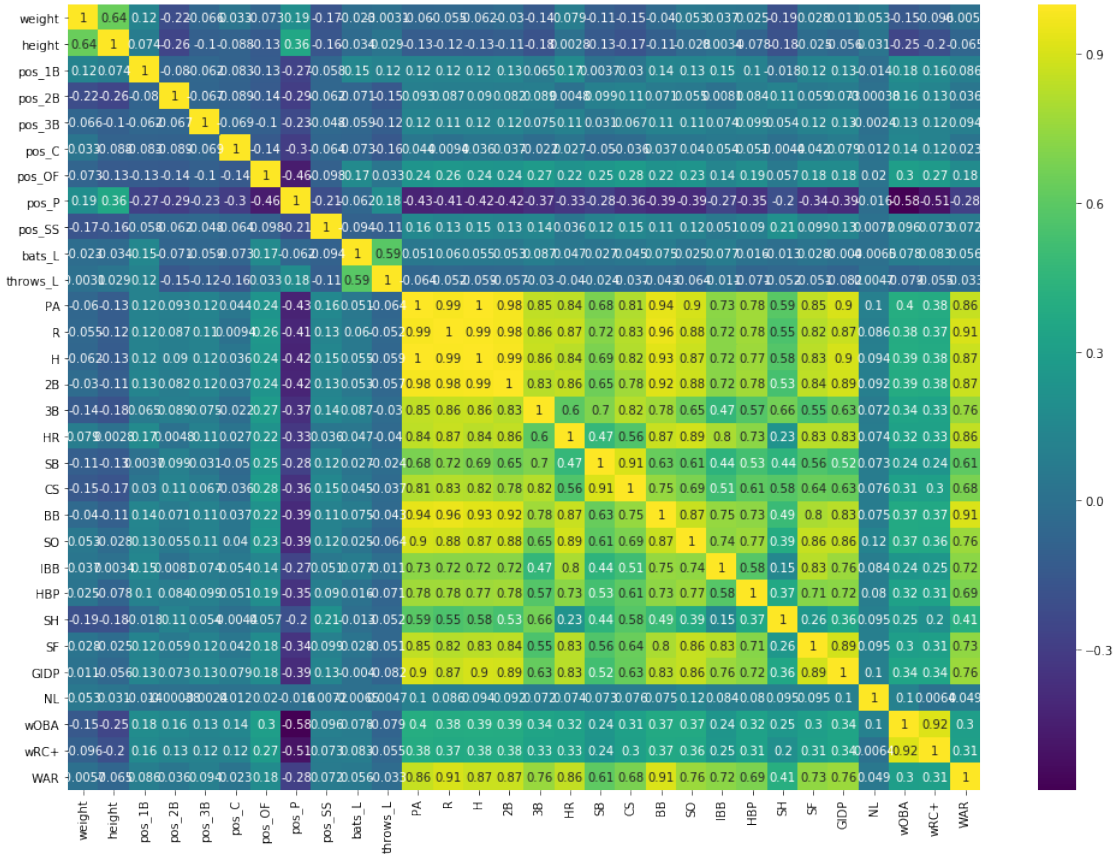
With the high correlation along the PA line, it may seem that we don't need them either. My reasoning for keeping them is a baseball-related one: we have most of the stats that make up a plate appearance, but we're missing the 'flied out' stat. This is an important one, as flyouts are a huge part of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

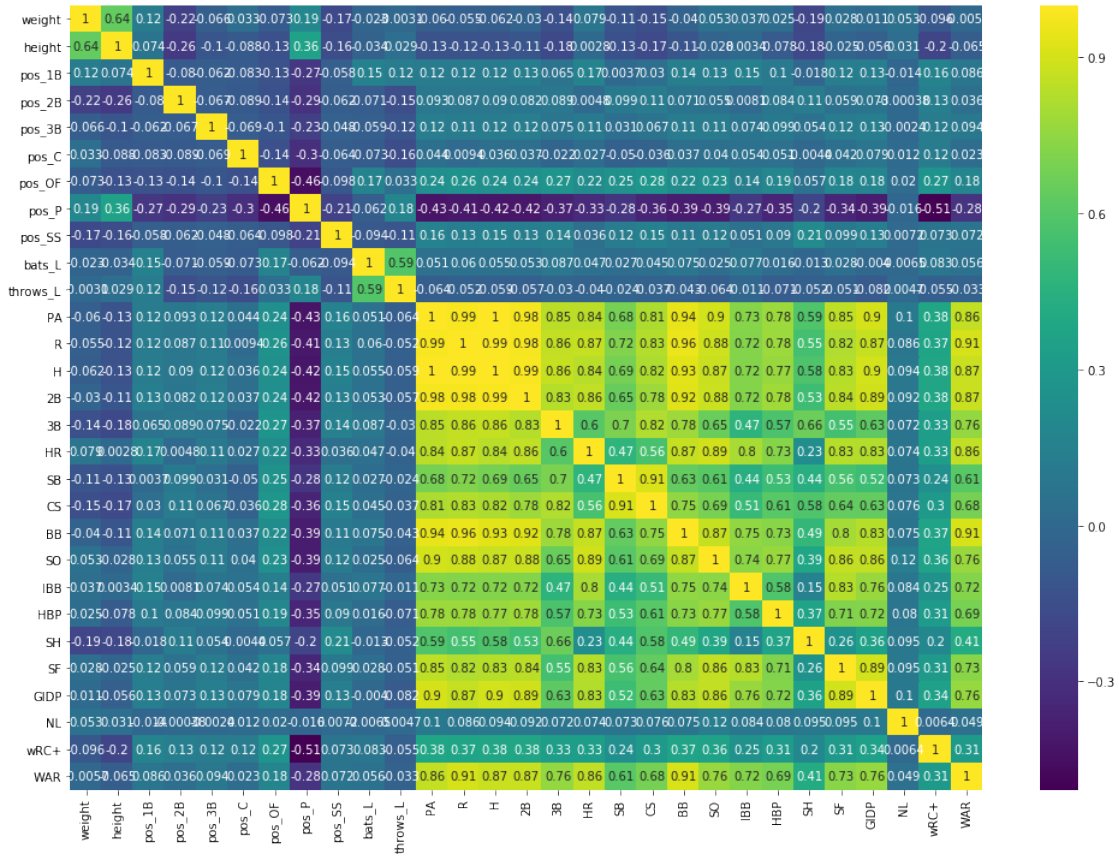
- They are both stats that essentially only consider offensive output, so do we really need two?
- wRC+ incorporates wRAA, which is built from wOBA, so the information is a bit repeated.
- We want to minimize the model's use of secondary/advanced stats which are extrapolated from the primary stats we 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]:
```
# We'll add the 'Batting' column back in to save our tensor
df.insert(loc=len(df.columns), column='Batting', value=y)
```

[29]:
```
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]
```