Data Analysis Using Numpy/Pandas - Baseball Dataset

Key Points

- Correlation between Team Perfromace Metrics
- Characteristics of Highly Paid Players

Note: For ease of use the data used for anaylsis was from the year 2011 and above

In [1]:
```
#Common Library Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
#For better plotting
import seaborn as sns
#To print all graphs inline
%matplotlib inline
```

In [2]:
```
#To run all output blocks of the code in a cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [3]:
```
#Loading all Input Data
teamInfo = pd.read_csv("./DataSet/Teams.csv")
playerInfo = pd.read_csv("./DataSet/Master.csv")
salaryInfo = pd.read_csv("./DataSet/Salaries.csv")
seriesPostInfo = pd.read_csv("./DataSet/SeriesPost.csv")
battingInfo = pd.read_csv("./DataSet/Batting.csv")
battingPostInfo = pd.read_csv("./DataSet/BattingPost.csv")
pitchingInfo = pd.read_csv("./DataSet/Pitching.csv")
pitchingPostInfo = pd.read_csv("./DataSet/PitchingPost.csv")
AllstarInfo = pd.read_csv("./DataSet/AllstarFull.csv")
```

Statistics Used For Analysis

I am a newbie to the terms and stats used in baseball sport as I don't follow the sport very much. I have decided to stick to the basics. Inspired by the **Monyeball theory** I have used the following stats for pitchers and batter to perform my analysis.

The following metrics were used to caluclate the metrics below

- Rank: Position in final standings
- R: Runs scored
- RA: Opponents runs scored

- G: Games played
- W: Wins
- H: Hits by batters
- BB: Walks by batters
- HBP: Batters hit by pitch
- AB: At bats
- SF: Sacrifice flies
- HR: Homeruns by batters
- 2B: Doubles
- 3B: Triples

Batters:

The **Batting Average [BA]** is defined by the number of hits divided by at bats. It can be calculated using the formula below:

- BA = H/AB

On-base Percentage is a measure of how often a batter reaches base for any reason other than a fielding error, fielder's choice, dropped/uncaught third strike, fielder's obstruction, or catcher's interference. It can be calculated using the formula below:

- OBP = (H+BB+HBP)/(AB+BB+HBP+SF)

Slugging Percentage is a measure of the power of a hitter. It can ve calculated using the formula below:

- SLG = H+2B+(2*3B)+(3*HR)/AB

Pitchers:

Earned Run Average (ERA) - Already provided in the dataset.

Monyeball Experiment Analysis:

Based on the analysis, a good strategy for recruiting batters would focus on targeting undervalued players with high OBP and SLG. Although BA and OBP have a positive correlation, there were some players that have high OBP and SLG, and relatively small BA. These players were undervalued by the market, and were the target for recruitment.

Data Cleanup

The team ID values from the dataset **Salary** and **Teams** is not consistent. The team ID is cleaned up to make it consistent. The function **setTeamID()** function is used to do the cleanup.

In [4]:
```
def setTeamID(data):
        data['teamID'].replace('CHA', 'CHW',inplace=True)
        data['teamID'].replace('CHN', 'CHC',inplace=True)
        data['teamID'].replace('LAN', 'LAD',inplace=True)
```

```
        data['teamID'].replace('NYA', 'NYM',inplace=True)
        data['teamID'].replace('KCA', 'KCR',inplace=True)
        data['teamID'].replace('NYN', 'NYY',inplace=True)
        data['teamID'].replace('SDN', 'SDP',inplace=True)
        data['teamID'].replace('SFN', 'SFG',inplace=True)
        data['teamID'].replace('SLN', 'STL',inplace=True)
        data['teamID'].replace('TBA', 'TBR',inplace=True)
        data['teamID'].replace('WAS', 'WSN',inplace=True)
```

In [5]:

```
#Cleaning Up Data for Team dataset
setTeamID(teamInfo)
```

In [6]:

```
#Five Year Dataset
teamLastFiveYrs = teamInfo[teamInfo['yearID'] > 2011]
teamInfo2016 = teamInfo[teamInfo['yearID'] == 2016]
#Five Year Dataset
salaryInfo2016 = salaryInfo[salaryInfo['yearID'] == 2016]
salaryLastFiveYrs = salaryInfo[salaryInfo['yearID'] > 2011]
```

In [7]:

```
teamInfo2016 = teamInfo2016[['yearID','teamID','name','park','lgID','divID
','Rank', 'R', 'RA', 'G', 'W', 'H', 'BB', 'HBP', 'AB', 'SF', 'HR', '2B', '
3B', 'ERA']]
teamLastFiveYrs = teamLastFiveYrs[['yearID','teamID','name','park','lgID',
'divID','Rank', 'R', 'RA', 'G', 'W', 'H', 'BB', 'HBP', 'AB', 'SF', 'HR', '
2B', '3B', 'ERA']]
```

Metrics Calculation

The function **calulateMetrics()** will calculate the following metircs

- Batting Average
- On-Base Percentage
- Sluggish Percentage

In [8]:

```
def calulateMetrics(data):
    data['BA'] = data['H']/data['AB']
    data['OBP'] = (data['H'] + data['BB'] + data['HBP']) / (data['AB'] + d
ata['BB'] + data['HBP'] + data['SF'])
    data['SLG'] = (data['H'] + data['2B'] + (2*data['3B']) + (3*data['HR'
])) / data['AB']
    #Winning Percentage [WP]
    data['WP'] = data['W']/data['G']
    #On Base Plus Slugging [OPS] - On Base Percentage + Slugging Average
    data['OPS'] = data['OBP'] + data['SLG']
    return data
```

In [9]:

```
teamInfo2016 = calulateMetrics(teamInfo2016)
teamLastFiveYrs = calulateMetrics(teamLastFiveYrs)
```

In [10]:

```
def getDataByLeague(data, league):
    return data[data['lgID'] == league]
```

In [11]:
```
#Split the Team Info by League AL & NL
teamInfoAL2016 = getDataByLeague(teamInfo2016, 'AL')
teamInfoNL2016 = getDataByLeague(teamInfo2016, 'NL')
teamInfoALFiveYrs = getDataByLeague(teamLastFiveYrs, 'AL')
teamInfoNLFiveYrs = getDataByLeague(teamLastFiveYrs, 'NL')
```

Calculating Descriptive Statistics

In [12]:
```
#Calculating League Stats
teamInfoNL2016Stats = teamInfoNL2016[['R','BA','OBP','SLG','OPS','HR','ERA
']].describe()
teamInfoNL5YrsStats = teamInfoNLFiveYrs[['R','BA','OBP','SLG','OPS','HR','
ERA']].describe()
teamInfoAL2016Stats = teamInfoAL2016[['R','BA','OBP','SLG','OPS','HR','ERA
']].describe()
teamInfoAL5YrsStats = teamInfoALFiveYrs[['R','BA','OBP','SLG','OPS','HR','
ERA']].describe()
```

League Stats for the year 2016 and for the last five years

National League[NL] Stats

- Year 2016
- Last Five Years [2011 & Above]

In [13]:
```
#National League Stats
teamInfoNL2016Stats.iloc[1]
teamInfoNL5YrsStats.iloc[1]
```

Out[13]:
```
R      718.266667
BA       0.253744
OBP      0.321912
SLG      0.411938
OPS      0.733850
HR     177.133333
ERA      4.165333
Name: mean, dtype: float64
```

Out[13]:
```
R      671.434211
BA       0.252215
OBP      0.316415
SLG      0.395997
OPS      0.712412
HR     151.947368
ERA      3.883947
Name: mean, dtype: float64
```

American League[AL] Stats

- Year 2016

- Last Five Years [2011 & Above]

In [14]:
```
#American League Stats
teamInfoAL2016Stats.iloc[1]
teamInfoAL5YrsStats.iloc[1]
```

Out[14]:
```
R      731.333333
BA       0.256792
OBP      0.320978
SLG      0.422620
OPS      0.743598
HR     196.866667
ERA      4.203333
Name: mean, dtype: float64
```

Out[14]:
```
R      708.040541
BA       0.255227
OBP      0.318814
SLG      0.407943
OPS      0.726757
HR     172.324324
ERA      4.018378
Name: mean, dtype: float64
```

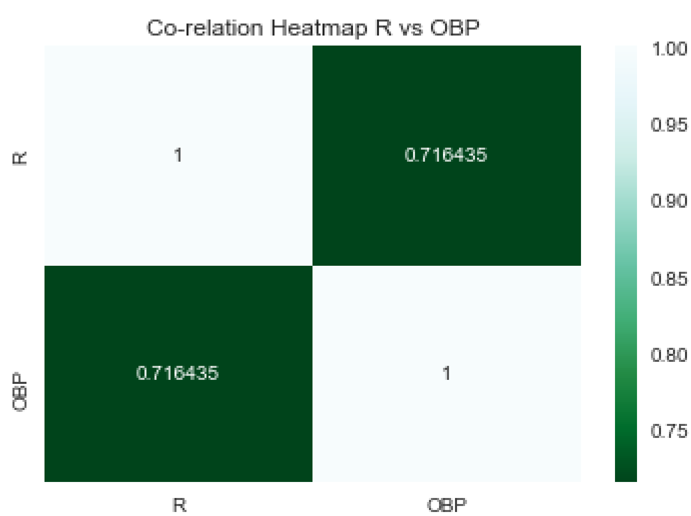
Calculating the Co-relation between Input Variables

- **Runs Scored Vs On-Base Percentage** [National League, 2016]

In [15]:
```
ax = plt.axes()
sns.heatmap(teamInfoNL2016[['R', 'OBP']].corr(), annot=True, fmt="g", cmap
='BuGn_r',ax = ax)
ax.set_title('Co-relation Heatmap R vs OBP')
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1199d12b0>

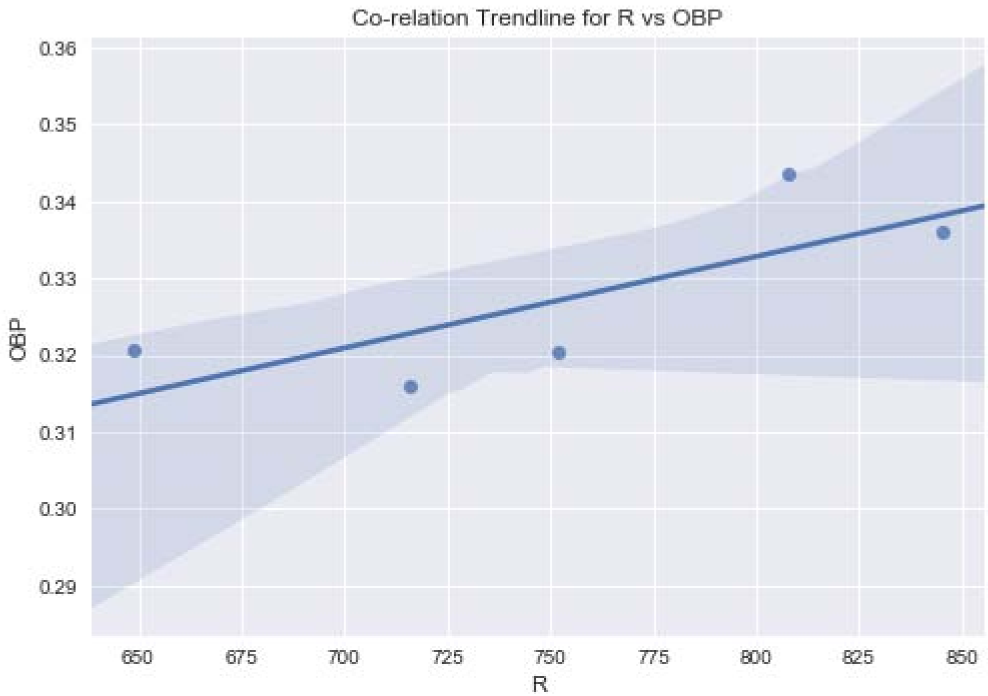
Out[15]: <matplotlib.text.Text at 0x1199265f8>



In [16]:
```
import seaborn as sns; sns.set(color_codes=True)
ax = sns.regplot(x=teamInfoNL2016['R'].head(), y=teamInfoNL2016['OBP'].he
```

```
ad())
ax.set_title('Co-relation Trendline for R vs OBP')
```

Out[16]: <matplotlib.text.Text at 0x11a2883c8>

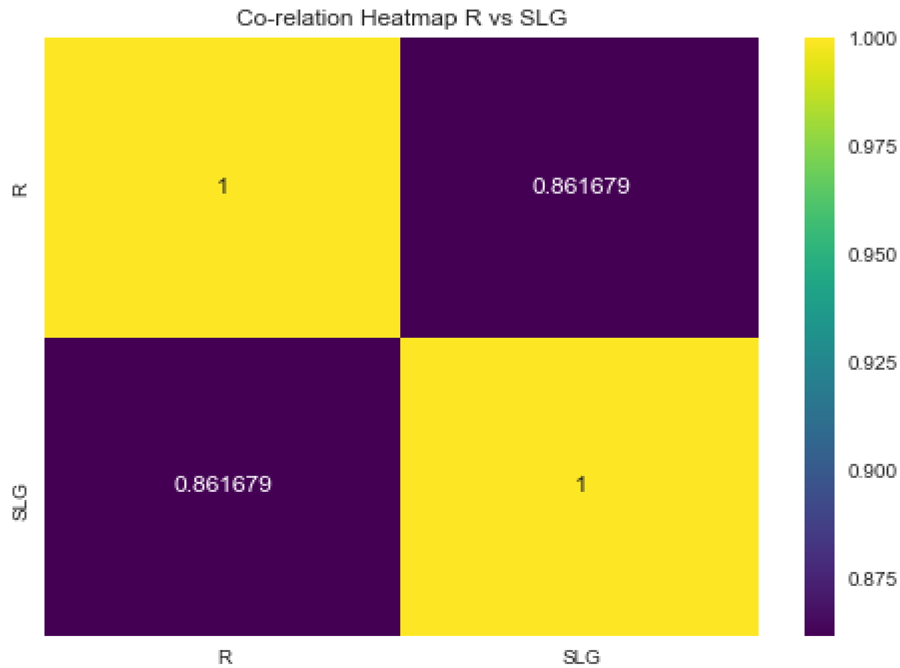


- **Runs Scored Vs Sluggish Percentage** [American League, 2016]

In [17]:
```
ax = plt.axes()
sns.heatmap(teamInfoAL2016[['R', 'SLG']].corr(), annot=True, fmt="g", cmap
='viridis')
ax.set_title('Co-relation Heatmap R vs SLG')
```

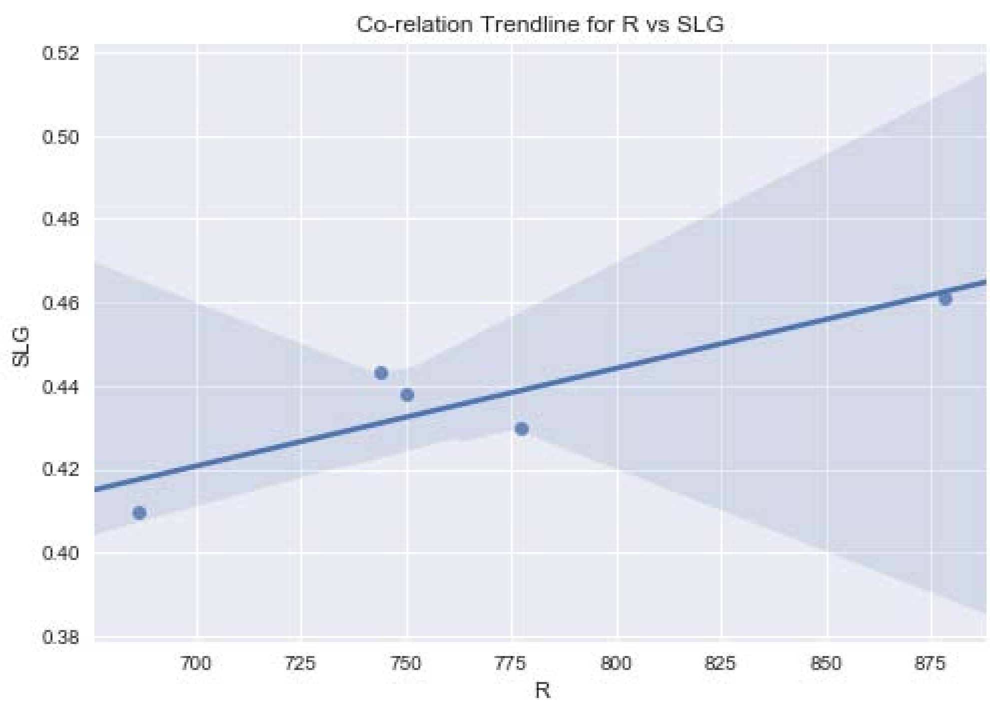
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x119642048>

Out[17]: <matplotlib.text.Text at 0x119f61ba8>



In [18]:
```
import seaborn as sns; sns.set(color_codes=True)
ax = sns.regplot(x=teamInfoAL2016['R'].head(), y=teamInfoAL2016['SLG'].he
ad())
ax.set_title('Co-relation Trendline for R vs SLG')
```

Out[18]: <matplotlib.text.Text at 0x11a1d0cf8>



Result:

- Both **On-Base Percentage** and **Sluggish Percentage** have a **strong co-relation** with **Runs Scored**
- Although both the metrics have a strong co-relation **Sluggish Percentgae** is the

strongest of the both

League Player Salary and Batting Metrics

The following analysis will look into the salary cap and the perfromance metrics of the players in National and American League

Batting Metrics

In [19]:
```
#Cleaning Up Data for Team dataset
setTeamID(battingInfo)
#Five Year Dataset
battingFiveYrs = battingInfo[battingInfo['yearID'] > 2011]
batting2016 = battingInfo[battingInfo['yearID'] == 2016]
batting2016 = batting2016[['yearID', 'teamID', 'lgID', 'playerID','R','G',
'H', 'BB', 'HBP', 'AB', 'SF', 'HR', '2B', '3B']]
battingFiveYrs = battingFiveYrs[['yearID', 'teamID', 'lgID', 'playerID','R
','G', 'H', 'BB', 'HBP', 'AB', 'SF', 'HR', '2B', '3B']]
```

In [20]:
```
def calculateBattingMetrics(data):
    data['BA'] = data['H']/data['AB']
    data['OBP'] = (data['H'] + data['BB'] + data['HBP']) / (data['AB'] + d
ata['BB'] + data['HBP'] + data['SF'])
    data['SLG'] = (data['H'] + data['2B'] + (2*data['3B']) + (3*data['HR'
])) / data['AB']
    #HR Ratio [HRRation]
    data['HRRatio'] = data['HR']/data['AB']
    #On Base Plus Slugging [OPS] - On Base Percentage + Slugging Average
    data['OPS'] = data['OBP'] + data['SLG']
    return data
```

In [21]:
```
batting2016 = calculateBattingMetrics(batting2016).fillna(0)
battingFiveYrs = calculateBattingMetrics(battingFiveYrs).fillna(0)
```

In [22]:
```
def AddPlayerNames(data):
    return pd.merge(playerInfo[['nameFirst','nameLast','nameGiven','bats',
'throws','playerID']], data, on=['playerID'], how='inner')
```

In [23]:
```
def getTeamInfo(data):
    setTeamID(teamInfo)
    setTeamID(data)
    return pd.merge(teamLastFiveYrs[['name','teamID']], data, on=['teamID'
], how='inner')
```

In [24]:
```
SalariesBy2016 = salaryInfo2016.groupby(['playerID', 'yearID']).sum().rese
t_index()
SalariesBy5Yrs = salaryLastFiveYrs.groupby(['playerID', 'yearID']).sum().r
eset_index()
```

In [25]:
```
MergeBattingInfo2016 = pd.merge(batting2016, SalariesBy2016, on=['yearID',
```

```
'playerID'], how='inner').fillna(0)
MergeBattingInfo5Yrs = pd.merge(battingFiveYrs, SalariesBy5Yrs, on=['yearI
D', 'playerID'], how='inner').fillna(0)
ALTopPlayers2016 = getDataByLeague(MergeBattingInfo2016, 'AL')
ALTopPlayers5Yrs = getDataByLeague(MergeBattingInfo5Yrs, 'AL')
NLTopPlayers2016 = getDataByLeague(MergeBattingInfo2016, 'NL')
NLTopPlayers5Yrs = getDataByLeague(MergeBattingInfo5Yrs, 'NL')
NL2016withNames = AddPlayerNames(NLTopPlayers2016)
NL5YrswithNames = AddPlayerNames(NLTopPlayers5Yrs)
AL2016withNames = AddPlayerNames(ALTopPlayers2016)
AL5YrswithNames = AddPlayerNames(ALTopPlayers5Yrs)
NLTopPlayers = NL5YrswithNames[['nameFirst','nameLast','nameGiven','bats',
'throws','playerID','yearID','teamID','lgID','R','BA','OBP','SLG','OPS','
HR','salary']].sort_values('salary', ascending=False).head()
ALTopPlayers = AL5YrswithNames[['nameFirst','nameLast','nameGiven','bats',
'throws','playerID','yearID','teamID','lgID','R','BA','OBP','SLG','OPS','
HR','salary']].sort_values('salary', ascending=False).head()
#sort_values('salary', ascending=False)
```

In [26]:

```
AL5YrswAvg = AL5YrswithNames[['yearID','R','BA','OBP','SLG','OPS','HR']].
groupby(['yearID']).mean().reset_index()
AL5YrswAvg.columns = ['yearID','RAvg','BAAvg','OBPAvg','SLGAvg','OPSAvg','
HRAvg']
AL5Yrsw = pd.merge(AL5YrswithNames,AL5YrswAvg, on='yearID', how='inner')
AL5Yr = AL5Yrsw[['nameFirst','nameLast','playerID', 'teamID','yearID','lgI
D','R','RAvg','BA','BAAvg','OBP','OBPAvg','SLG','SLGAvg','OPSAvg','HR','H
RAvg', 'salary']]
AL5Yr.sort_values('salary', ascending=False).head()
```

Out[26]:

	nameFirst	nameLast	playerID	teamID	yearID	lgID	R	RAvg	BA	BAAvg
1638	David	Price	priceda01	BOS	2016	AL	0	21.789593	0.000000	0.13882
330	Alex	Rodriguez	rodrial01	NYM	2012	AL	74	21.025701	0.272138	0.14439
2130	Alex	Rodriguez	rodrial01	NYM	2013	AL	21	20.534368	0.243590	0.13599
1303	Justin	Verlander	verlaju01	DET	2015	AL	0	19.765591	0.000000	0.14043
1383	Miguel	Cabrera	cabremi01	DET	2016	AL	92	21.789593	0.315966	0.13882

In [27]:

```
NL5YrswAvg = NL5YrswithNames[['yearID','R','BA','OBP','SLG','OPS','HR']].
groupby(['yearID']).mean().reset_index()
NL5YrswAvg.columns = ['yearID','RAvg','BAAvg','OBPAvg','SLGAvg','OPSAvg','
HRAvg']
NL5Yrsw = pd.merge(NL5YrswithNames,NL5YrswAvg, on='yearID', how='inner')
NL5Yr = NL5Yrsw[['nameFirst','nameLast','playerID', 'teamID','yearID','lgI
D','R','RAvg','BA','BAAvg','OBP','OBPAvg','SLG','SLGAvg','OPSAvg','HR','H
RAvg', 'salary']]
NL5Yr.sort_values('salary', ascending=False).head()
```

Out[27]:

	nameFirst	nameLast	playerID	teamID	yearID	lgID	R	RAvg	BA	BAAvg
1577	Clayton	Kershaw	kershcl01	LAD	2016	NL	2	21.234513	0.173913	0.15365
2044	Clayton	Kershaw	kershcl01	LAD	2015	NL	2	19.169978	0.126761	0.15505
1520	Zack	Greinke	greinza01	ARI	2016	NL	4	21.234513	0.211538	0.15365

1434	Yoenis	Cespedes	cespeyo01	NYY	2016	NL	72	21.234513	0.279749	0.15365
1071	Zack	Greinke	greinza01	LAD	2014	NL	5	19.556054	0.200000	0.15663

Pitching Metrics

In [28]:
```
pitchInfo = pitchingInfo[pitchingInfo['yearID'] > 2011]
pitchInfo = pitchInfo[['yearID', 'teamID','lgID' ,'playerID', 'W', 'L', 'G
', 'BAOpp', 'ERA', 'R','CG', 'SHO','SV', 'H']].fillna(0)
```

In [29]:
```
setTeamID(pitchInfo)
pitchPlayer = pd.merge(pitchInfo, SalariesBy5Yrs, on=['yearID', 'playerID'
], how='inner').fillna(0)
ALTopPitchers = pitchPlayer[pitchPlayer['lgID']=='AL']
NLTopPitchers = pitchPlayer[pitchPlayer['lgID']=='NL']
NLTopPitcherswithNames = pd.merge(playerInfo[['nameFirst','nameLast','name
Given','playerID']], NLTopPitchers, on=['playerID'], how='inner').fillna(0
)
ALTopPitcherswithNames = pd.merge(playerInfo[['nameFirst','nameLast','name
Given','playerID']], ALTopPitchers, on=['playerID'], how='inner').fillna(0
)
NLTopPitcherswithNames.sort_values('salary', ascending=False).head()
ALTopPitcherswithNames.sort_values('salary', ascending=False).head()
```

Out[29]:

	nameFirst	nameLast	nameGiven	playerID	yearID	teamID	lgID	W	L	G	BAOpp	ERA
542	Clayton	Kershaw	Clayton Edward	kershcl01	2016	LAD	NL	12	4	21	0.184	1.69
541	Clayton	Kershaw	Clayton Edward	kershcl01	2015	LAD	NL	16	7	33	0.194	2.13
405	Zack	Greinke	Donald Zachary	greinza01	2016	ARI	NL	13	7	26	0.262	4.37
403	Zack	Greinke	Donald Zachary	greinza01	2014	LAD	NL	17	8	32	0.247	2.71
404	Zack	Greinke	Donald Zachary	greinza01	2015	LAD	NL	19	3	32	0.187	1.66

Out[29]:

	nameFirst	nameLast	nameGiven	playerID	yearID	teamID	lgID	W	L	G	BAOpp	ER
786	David	Price	David Taylor	priceda01	2016	BOS	AL	17	9	35	0.258	3.[illegible]
1033	Justin	Verlander	Justin Brooks	verlaju01	2015	DET	AL	5	8	20	0.229	3.[illegible]
1034	Justin	Verlander	Justin Brooks	verlaju01	2016	DET	AL	16	9	34	0.207	3.[illegible]
421	Felix	Hernandez	Felix Abraham	hernafe02	2016	SEA	AL	11	8	25	0.239	3.[illegible]

420	Felix	Hernandez	Felix Abraham	hernafe02	2015	SEA	AL	18	9	31	0.240	3.5

In [30]:
```
NL5YrswAvg = NLTopPitcherswithNames[['yearID','ERA']].groupby(['yearID']).
mean().reset_index()
NL5YrswAvg.columns = ['yearID','ERAvg']
NL5Yrsw = pd.merge(NLTopPitcherswithNames,NL5YrswAvg, on='yearID', how='in
ner')
NL5Yr = NL5Yrsw[['nameFirst','nameLast','playerID', 'teamID','yearID','lgI
D','W', 'L', 'G', 'BAOpp', 'ERA','ERAvg', 'salary']]
NL5Yr.sort_values('salary', ascending=False).head()
```

Out[30]:

	nameFirst	nameLast	playerID	teamID	yearID	lgID	W	L	G	BAOpp	ERA	ERAvg
1007	Clayton	Kershaw	kershcl01	LAD	2016	NL	12	4	21	0.184	1.69	4.612265
801	Clayton	Kershaw	kershcl01	LAD	2015	NL	16	7	33	0.194	2.13	3.984934
979	Zack	Greinke	greinza01	ARI	2016	NL	13	7	26	0.262	4.37	4.612265
531	Zack	Greinke	greinza01	LAD	2014	NL	17	8	32	0.247	2.71	4.236771
1016	Jon	Lester	lestejo01	CHC	2016	NL	19	5	32	0.211	2.44	4.612265

In [31]:
```
AL5YrswAvg = ALTopPitcherswithNames[['yearID','ERA']].groupby(['yearID']).
mean().reset_index()
AL5YrswAvg.columns = ['yearID','ERAvg']
AL5Yrsw = pd.merge(ALTopPitcherswithNames,AL5YrswAvg, on='yearID', how='in
ner')
AL5Yr = AL5Yrsw[['nameFirst','nameLast','playerID', 'teamID','yearID','lgI
D','W', 'L', 'G', 'BAOpp', 'ERA','ERAvg', 'salary']]
AL5Yr.sort_values('salary', ascending=False).head()
```

Out[31]:

	nameFirst	nameLast	playerID	teamID	yearID	lgID	W	L	G	BAOpp	ERA	ERAvg
799	David	Price	priceda01	BOS	2016	AL	17	9	35	0.258	3.99	4.381379
854	Justin	Verlander	verlaju01	DET	2016	AL	16	9	34	0.207	3.04	4.381379
632	Justin	Verlander	verlaju01	DET	2015	AL	5	8	20	0.229	3.38	4.279750
727	Felix	Hernandez	hernafe02	SEA	2016	AL	11	8	25	0.239	3.82	4.381379
496	Felix	Hernandez	hernafe02	SEA	2015	AL	18	9	31	0.240	3.53	4.279750

Series Winners

In [32]:
```
seriesFiveYrs = seriesPostInfo[seriesPostInfo['yearID'] > 2005]
series2016 = seriesPostInfo[seriesPostInfo['yearID'] == 2016]
seriesFiveYrs=seriesFiveYrs.rename(columns = {'teamIDwinner':'teamID'})
setTeamID(seriesFiveYrs)
seriesFiveYrs=seriesFiveYrs.rename(columns = {'teamID':'teamIDwinner'})
seriesFiveYrs=seriesFiveYrs.rename(columns = {'teamIDloser':'teamID'})
setTeamID(seriesFiveYrs)
seriesFiveYrs=seriesFiveYrs.rename(columns = {'teamID':'teamIDloser'})
```

```
WSwinners = seriesFiveYrs[seriesFiveYrs['round'] == 'WS']
WSwinners[WSwinners['lgIDwinner'] == 'NL'].head()
WSwinners[WSwinners['lgIDwinner'] == 'AL'].head()
```

Out[32]:

	yearID	round	teamIDwinner	lgIDwinner	teamIDloser	lgIDloser	wins	losses	ties
235	2006	WS	STL	NL	DET	AL	4	1	0
249	2008	WS	PHI	NL	TBR	AL	4	1	0
263	2010	WS	SFG	NL	TEX	AL	4	1	0
270	2011	WS	STL	NL	TEX	AL	4	3	0
279	2012	WS	SFG	NL	DET	AL	4	0	0

Out[32]:

	yearID	round	teamIDwinner	lgIDwinner	teamIDloser	lgIDloser	wins	losses	ties
242	2007	WS	BOS	AL	COL	NL	4	0	0
256	2009	WS	NYM	AL	PHI	NL	4	2	0
288	2013	WS	BOS	AL	STL	NL	4	3	0
306	2015	WS	KCR	AL	NYY	NL	4	1	0

Analysis

The **highly paid** players have the following features

- The top players in both NL and AL are all ptichers
- The perfromance of the players are way above the league average
- Team having higly paid players have made to the post season

Conclusion

We have successfully analyzed the data from the given dataset and performed analysis using Pandas and Numpy. We have chosen the **Baseball dataset** as our sample as we were not familiar with the game and wanted to take a shot at the data to understand the corresponding metrics used for the same.

As part of the analysis we wanted to find out the following

- Factors that affect a teams performance
- Top performers in each category [Batting/Pitching] for the last five years and their characteristics

For simplicity we have used the following metrics for the hypothesis

- Batters
  - Batting Average
  - On-Base Percentage
  - Sluggish Percentage
- Pitcher

    - Earned Runs Average

Based on the analysis the above mentioned factors have a high co-relation with the winning percentgae and they factor in teams performance. Also based on the last five year data the top five highly paid players were all pitchers. Though the america league is more inclined to the pitchers it is worth noting that both the leagues has the top highly paid as players as pitchers.

Note: The analysis is only done for the regular season. All the metrics used for the anaylsis holds good for the regular season.

Post-season metrics are not taken into consideration. After looking at the last five year world series winners it looks like its not always true for the top spending teams and the top paid players to win the world series. We would have to take a look at the post-season metorcs to do further analysis.

Limitations:

The dataset do have some limitations. Our few observations below

- Individual Match Statistics
- More info on the Pitcher - Pitch Types, Speed etc.
- Stadium Size, weather details ect.

Further Anlysis

Other metrics we were interested in and would like to deep dive into are as follows

- Effect on Pitch Type vs Wins
- Pitch Type mostly used in Post-Season
- Team Spending Vs Post-Season Wins
- Weather Vs Attendance and other factors affecting the same

References:

- https://www.pinstripealley.com/2010/12/23/1892608/baseball-statistics-and-acronyms-explained
- https://pandas.pydata.org/pandas-docs/stable/visualization.html
- https://www.baseball-reference.com/leagues/MLB/2012.shtml
- https://en.wikipedia.org/wiki/Major_League_Baseball_postseason

In [ ]:

```

```