Analysis on MLB Player Salaries

By Max Pargman and Alex Carter

Points of Interest

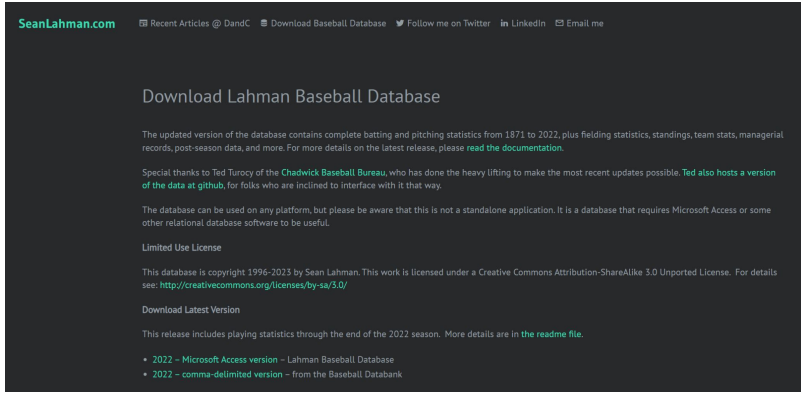
We chose the topic of analyzing MLB player salaries and what goes into valuing a player, as we both played baseball in highschool and are very interested into not only statistics but the economics behind the game. We felt by selecting the data bases of players statistics, profiles, and salaries we would have enough of a foundation to formulate insights into why certain players get paid these astronomical amounts.

Expectations

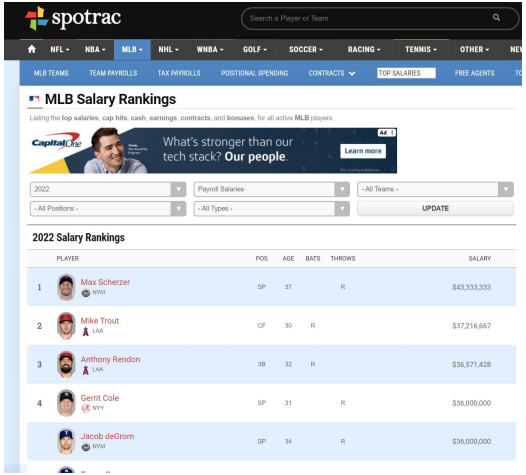
It is common knowledge that “The Ladies Dig The Longball”, and we think that MLB General Managers enjoy them too. We expect that Home Runs be the driving force behind the value of players salaries as the long ball is a major focal point of today’s game.

Finding our Datasets

- Downloaded: Sean Lahman Baseball Database http://seanlahman.com/download-baseball-database/
- Web Collection #1 Source: API https://api.sportsdata.io/v3/mlb/scores/json/Players?key=76ee08fa3709417c85ec4abc607fe42a
- Web Collection #2 Source: Web Scraping https://www.spotrac.com/mlb/rankings/2022/salary/




```
[
  - {
      PlayerID: 10000001,
      SportsDataID: "",
      Status: "15-Day Injured List",
      TeamID: 23,
      Team: "COL",
      Jersey: 45,
      PositionCategory: "P",
      Position: "SP",
      MLBAMID: 502624,
      FirstName: "Chase",
      LastName: "Anderson",
      BatHand: "R",
      ThrowHand: "R",
      Height: 73,
      Weight: 210,
      BirthDate: "1987-11-30T00:00:00",
      BirthCity: "Wichita Falls",
      BirthState: "TX",
      BirthCountry: "USA",
      HighSchool: null,
      College: "Oklahoma",
      ProDebut: "2014-05-11T00:00:00",
      Salary: null,
      PhotoUrl: "https://s3-us-west-2.amazonaws.com/static.fantasydata
      SportRadarPlayerID: "4bdcd59e-2cd5-4d1e-8058-588a72a1f8ae",
      RotoworldPlayerID: 7333,
      RotoWirePlayerID: 12419,
      FantasyAlarmPlayerID: 100318,
      StatsPlayerID: 501567,
      SportsDirectPlayerID: 106019,
      XmlTeamPlayerID: 4645,
```

Data Cleaning: Downloaded

- Batter Stats
- .drop()
- .merge()

	playerID	yearID	teamID	lgID	G	AB	R	H	2B	3B	...	GIDP	nameFirst	nameLast	weight	height	bats	throws	debut	finalGame	nameFull
0	abbotje01	2000	CHA	AL	80	215	31	59	15	1	...	2.0	Jeff	Abbott	190.0	74.0	R	L	1997-06-10	2001-09-29	Jeff Abbott
1	abbotku01	2000	NYN	NL	79	157	22	34	7	1	...	2.0	Kurt	Abbott	180.0	71.0	R	R	1993-09-07	2001-04-13	Kurt Abbott
2	abbotpa01	2000	SEA	AL	35	5	1	2	1	0	...	0.0	Paul	Abbott	185.0	75.0	R	R	1990-08-21	2004-08-07	Paul Abbott
3	abreubo01	2000	PHI	NL	154	576	103	182	42	10	...	12.0	Bobby	Abreu	220.0	72.0	L	R	1996-09-01	2014-09-28	Bobby Abreu
4	aceveju01	2000	MIL	NL	62	1	1	0	0	0	...	0.0	Juan	Acevedo	245.0	74.0	R	R	1995-04-30	2003-08-05	Juan Acevedo
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32913	zimmebr01	2022	TOR	AL	77	76	11	8	4	0	...	0.0	Bradley	Zimmer	185.0	76.0	L	R	2017-05-16	2022-10-05	Bradley Zimmer
32914	zimmebr01	2022	PHI	NL	9	16	4	4	1	0	...	0.0	Bradley	Zimmer	185.0	76.0	L	R	2017-05-16	2022-10-05	Bradley Zimmer
32915	zimmebr01	2022	TOR	AL	23	13	3	1	0	0	...	0.0	Bradley	Zimmer	185.0	76.0	L	R	2017-05-16	2022-10-05	Bradley Zimmer
32916	zimmebr02	2022	BAL	AL	15	0	0	0	0	0	...	0.0	Bruce	Zimmermann	215.0	73.0	L	L	2020-09-17	2022-09-05	Bruce Zimmermann
32917	zuninmi01	2022	TBA	AL	36	115	7	17	3	0	...	2.0	Mike	Zunino	235.0	74.0	R	R	2013-06-12	2022-06-09	Mike Zunino

32918 rows × 30 columns

```python
def data_parser():

    ## reading downloaded csv files
    batting = pd.read_csv("Batting.csv",delimiter=",")
    people = pd.read_csv("People.csv", delimiter=",")

    people["nameFull"] = people["nameFirst"] + " " + people["nameLast"]

    ## dropping player records from years before 2000 for simplicity
    batting.drop(batting[batting.yearID < 2000].index,inplace=True)

    ## merging batting and people csvs to have one dataframe with both info
    batters_df = pd.merge(batting,people,how='left',on='playerID')

    ## dropping unnecessary columns and resetting indices
    batters_df.drop(["retroID","bbrefID",'birthYear', 'birthMonth', 'birthDay', 'birthCountry',
           'birthState', 'birthCity', 'deathYear', 'deathMonth', 'deathDay',
           'deathCountry', 'deathState', 'deathCity','stint','nameGiven'],inplace=True,axis=1)
    batters_df.reset_index(drop=True,inplace=True)

    ## making csv
    batters_df.to_csv("Batter Stats.csv",index=True)
```

```python
batter_stats.keys()
```

```
Index(['playerID', 'yearID', 'teamID', 'lgID', 'G', 'AB', 'R', 'H', '2B', '3B',
       'HR', 'RBI', 'SB', 'CS', 'BB', 'SO', 'IBB', 'HBP', 'SH', 'SF', 'GIDP',
       'nameFirst', 'nameLast', 'weight', 'height', 'bats', 'throws', 'debut',
       'finalGame', 'nameFull'],
      dtype='object')
```

Data Cleaning: Web Collection #1: API

```python
def web_parser1():

  ## getting api data
  api_url = 'https://api.sportsdata.io/v3/mlb/scores/json/Players?key=76ee08fa3709417c85ec4abc607fe42a'
  response = requests.get(api_url)
  data = response.json()

  ## filtering data: dropping minor league players, non rostered players and pitchers
  unfiltered_data = [player for player in data if player['Status'] != 'Minors' and player['Status'] != '40 Man Active']
  filtered_data = [player for player in unfiltered_data if player['PositionCategory'] != 'P']

  ## making dataframe with filtered data
  player_profiles_df = pd.DataFrame(filtered_data)

  ## dropping unnecessary columns
  columns_to_drop = ['SportsDataID', 'Salary', 'PhotoUrl', 'SportRadarPlayerID', 'RotoworldPlayerID', 'RotoWirePlayerID', 'FantasyAlarmPlayerID',
                     'StatsPlayerID', 'XmlTeamPlayerID', 'InjuryStatus', 'InjuryBodyPart', 'InjuryStartDate', 'InjuryNotes', 'FanDuelPlayerID', 'DraftKingsPlayerID',
                     'YahooPlayerID', 'UpcomingGameID', 'FanDuelName', 'DraftKingsName', 'YahooName', 'GlobalTeamID', 'FantasyDraftName', 'FantasyDraftPlayerID',
                     'UsaTodayPlayerID', 'UsaTodayHeadshotUrl', 'UsaTodayHeadshotNoBackgroundUrl', 'UsaTodayHeadshotUpdated', 'UsaTodayHeadshotNoBackgroundUpdated',
                     'SportsDirectPlayerID', 'MLBAMID']
  player_profiles_df = player_profiles_df.drop(columns_to_drop, axis=1)

  ## modifying columns and setting nulls
  player_profiles_df['ProDebut'] = player_profiles_df['ProDebut'].str.split('T').str[0]
  player_profiles_df['BirthDate'] = player_profiles_df['BirthDate'].str.split('T').str[0]
  player_profiles_df.replace('', np.nan, inplace=True)

  player_profiles_df["FullName"] = player_profiles_df["FirstName"] + " " + player_profiles_df["LastName"]

  player_profiles_df = pd.merge(player_profiles_df, batter_stats[["playerID","nameFull"]],
                                how='left',left_on='FullName', right_on="nameFull")
  player_profiles_df = player_profiles_df.drop_duplicates(subset="playerID")


  ## making csv
  player_profiles_df.to_csv('Player Profiles.csv', index=False)
```

```python
player_profiles.keys()
```

```
Index(['Status', 'TeamID', 'Team', 'Jersey', 'PositionCategory', 'Position',
       'FirstName', 'LastName', 'BatHand', 'ThrowHand', 'Height', 'Weight',
       'BirthDate', 'BirthCity', 'BirthState', 'BirthCountry', 'HighSchool',
       'College', 'ProDebut', 'Experience', 'FullName', 'playerID',
       'nameFull'],
      dtype='object')
```

PlayerID	Status	TeamID	Team	Jersey	PositionCategory	Position	FirstName	LastName	BatHand	ThrowHand	...	BirthCity	BirthState	BirthCountry	HighSchool	College	ProDebut
10000029	Active	14	ARI	13.0	IF	SS	Nick	Ahmed	R	R	...	Springfield	MA	USA	NaN	Connecticut	2014-06-29
10000030	10-Day Injured List	21	LAA	23.0	IF	2B	Brandon	Drury	R	R	...	Grants Pass	OR	USA	Grants Pass (OR)	None	2015-09-01
10000031	Active	31	STL	46.0	IF	1B	Paul	Goldschmidt	R	R	...	Wilmington	DE	USA	NaN	Texas State	2011-08-01
10000040	Active	1	LAD	6.0	OF	LF	David	Peralta	L	L	...	Valencia	NaN	Venezuela	NaN	None	2014-06-01
10000041	10-Day Injured List	13	SEA	8.0	OF	LF	AJ	Pollock	R	R	...	Hebron	CT	USA	NaN	Notre Dame	2012-04-18
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10012277	Active	31	STL	21.0	OF	RF	Lars	Nootbaar	L	R	...	El Segundo	CA	USA	NaN	Southern California	2021-06-22
10012310	Active	30	HOU	21.0	IF	C	Yainer	Diaz	R	R	...	Azua	NaN	Dominican Republic	NaN	None	2022-09-02
10012442	Active	10	CLE	17.0	OF	RF	Will	Brennan	L	L	...	Colorado Springs	CO	USA	NaN	Kansas State	2022-09-21
10013284	Active	23	COL	14.0	IF	SS	Ezequiel	Tovar	R	R	...	Maracay	NaN	Venezuela	NaN	None	2022-09-23
10013287	Active	5	KC	11.0	IF	3B	Maikel	Garcia	R	R	...	La Sabana	NaN	Venezuela	NaN	None	2022-07-15

378 rows × 23 columns

- Filtering
- .drop()
- .merge()

Data Cleaning: Web Collection #2: Web Scraping

```python
def web_parser2():
    ## making dataframe
    salaries_df=pd.DataFrame()

    ## looping through pages on website for 2020-2022 data
    for n in range(20,23):
        ## gettting html to scrape
        url = f'https://www.spotrac.com/mlb/rankings/20{n}/salary/'
        data = {
            'ajax': 'true',
            'mobile': 'false'
        }
        soup = BeautifulSoup(requests.post(url, data=data).content, 'html.parser')
        table = soup.find("table").text

        ## pulling out names, salaries and years for each player
        t1 = table.strip()
        players = re.findall(r"[A-Z][a-zA-Z]{2,}\s[a-zA-Z]{2,}",t1)[2:]
        salaries = re.findall(r"\$([\d,]*)",t1)
        year = [int(f"20{n}") for i in range(len(players))]

        ## making rank for players because the data comes ordered
        rank = list(range(1,len(players)+1))

        ## adding data to dataframe
        data = list(zip(players,salaries,year,rank))
        if salaries_df.empty:
            salaries_df = pd.DataFrame(data,columns=["Player","Salary","Year","Rank"])
        else:
            df_new = pd.DataFrame(data,columns=["Player","Salary","Year","Rank"])
            salaries_df = pd.concat([salaries_df,df_new],ignore_index=True)

    ######## making player_profiles dataframe including pitchers to use for salaries dataframe
    api_url = 'https://api.sportsdata.io/v3/mlb/scores/json/Players?key=76ee08fa3709417c85ec4abc607fe42a'
    response = requests.get(api_url)
    data = response.json()
    unfiltered_data = [player for player in data if player['Status'] != 'Minors' and player['Status'] != '40 Man Active']
    player_profiles_with_pitchers = pd.DataFrame(unfiltered_data)
    player_profiles_with_pitchers["FullName"] = player_profiles_with_pitchers["FirstName"] + " " + player_profiles_with_pitchers["LastName"]
    #########################
```

```python
## adding positions to dataframe using the player_profiles dataframe
salaries_df = pd.merge(salaries_df,player_profiles_with_pitchers[["FullName","Position"]],
                       how='left',left_on='Player',right_on='FullName')
salaries_df = salaries_df.drop(columns=["FullName"],axis=1)

## filtering out pitchers, rows with NaN, and reindexing
salaries_df = salaries_df[salaries_df["Position"] != "SP"]
salaries_df = salaries_df[salaries_df["Position"] != "RP"]
salaries_df = salaries_df[salaries_df["Position"] != "CP"]
salaries_df = salaries_df.dropna(axis=0)
salaries_df.index = list(range(len(salaries_df)))

salaries_df["Salary"] = salaries_df["Salary"].str.replace(",","").astype(int)

salaries_df = pd.merge(salaries_df, batter_stats[["playerID", "nameFull","yearID"]],
                       how='left', left_on=['Player','Year'], right_on=['nameFull','yearID'])
## to csv
salaries_df.to_csv("Player Salaries 2020-2022.csv",index=True)
```

- BeautifulSoup
- Requests
- Regex
- .merge()

	Player	Salary	Year	Rank	Position	playerID	nameFull	yearID
0	Mike Trout	37766667	2020	1	CF	troutmi01	Mike Trout	2020.0
1	Nolan Arenado	35025000	2020	4	3B	arenano01	Nolan Arenado	2020.0
2	Manny Machado	32000000	2020	8	3B	machama01	Manny Machado	2020.0
3	Miguel Cabrera	30000000	2020	12	DH	cabremi01	Miguel Cabrera	2020.0
4	Jose Altuve	29000000	2020	14	2B	altuvjo01	Jose Altuve	2020.0
...	...	...	...	...	...	...	...	...
714	Corbin Carroll	700000	2022	975	LF	carroco02	Corbin Carroll	2022.0
715	Ildemaro Vargas	700000	2022	978	2B	vargail01	Ildemaro Vargas	2022.0
716	Ildemaro Vargas	700000	2022	978	2B	vargail01	Ildemaro Vargas	2022.0
717	Sean Bouchard	700000	2022	979	LF	bouchse01	Sean Bouchard	2022.0
718	Stone Garrett	700000	2022	985	LF	garrest01	Stone Garrett	2022.0

719 rows × 8 columns

Insight/Visualization #1: Correlations of Home Runs, Stolen Bases vs Salary

```python
def insight1():
    batter_stats = pd.read_csv("Batter Stats.csv",index_col=0)
    salaries = pd.read_csv("Player Salaries 2020-2022.csv",index_col=0)

    batter_stats_and_salaries = pd.merge(batter_stats, salaries[["playerID", "Salary","Year"]],
                                         how='inner',left_on=['playerID","yearID"],right_on=["playerID","Year"])

    hr_corr = batter_stats_and_salaries["HR"].corr(batter_stats_and_salaries["Salary"])
    sb_corr = batter_stats_and_salaries["SB"].corr(batter_stats_and_salaries["Salary"])
    print(f"Homerun vs Salary correlation: {round(hr_corr,4)} \nStolen Base vs Salary correlation: {round(sb_corr,4)}")


############ Function Call ############
insight1()
```

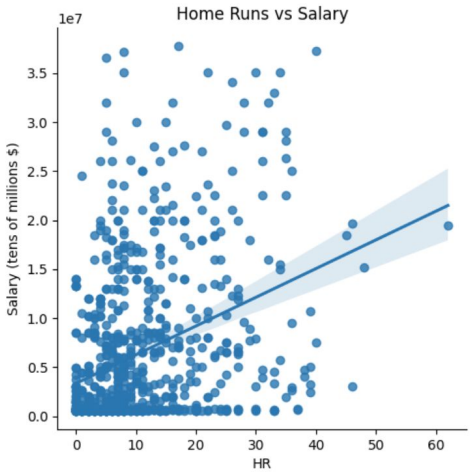
```
Homerun vs Salary correlation: 0.3523
Stolen Base vs Salary correlation: 0.0715
```

```python
def visual1():
    batter_stats = pd.read_csv("Batter Stats.csv",index_col=0)
    salaries = pd.read_csv("Player Salaries 2020-2022.csv",index_col=0)

    batter_stats_and_salaries = pd.merge(batter_stats, salaries[["playerID", "Salary","Year"]]
                                         how='inner',left_on=['playerID","yearID"],
                                         right_on=["playerID","Year"])

    sns.lmplot(x="HR",y="Salary",data=batter_stats_and_salaries)
    ax1=plt.gca()
    ax1.set_title("Home Runs vs Salary")
    ax1.set_ylabel("Salary (tens of millions $)")
    sns.lmplot(x="SB",y="Salary",data=batter_stats_and_salaries)
    ax2=plt.gca()
    ax2.set_title("Stolen Bases vs Salary")
    ax2.set_ylabel("Salary (tens of millions $)")
```

- .corr()
- Seaborn
- lmplot
- Matplotlib



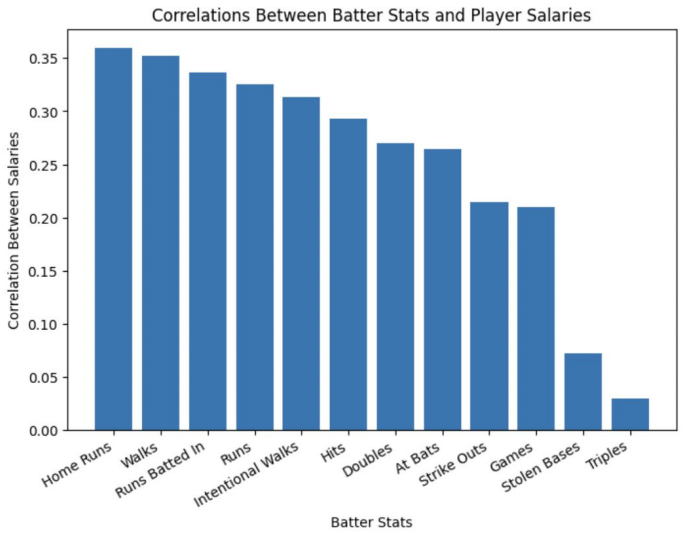
Insight #2: Correlations of All Stats vs Salary

```python
def insight2():
  batter_stats = pd.read_csv("Batter Stats.csv",index_col=0)
  salaries = pd.read_csv("Player Salaries 2020-2022.csv",index_col=0)

  batter_stats_and_salaries = pd.merge(batter_stats, salaries[["playerID", "Salary","Year"]],
                                       how='inner',left_on=['playerID','yearID']
                                       ,right_on=["playerID","Year"])

  correlations = batter_stats_and_salaries.corr()
  correlations = pd.DataFrame(correlations).reset_index()
  salaries_correlations = correlations[["index","Salary"]]
  salaries_correlations.columns = ["Stat","Salary"]

  salaries_correlations.drop([0,10,14,15,16,17,18,19,20,21],inplace=True)
  salaries_correlations.sort_values(by="Salary", ascending=False,inplace=True)
  salaries_correlations.index = list(range(1,13))
  salaries_correlations.index = salaries_correlations.index.rename("Rank")
  salaries_correlations["Stat"] = ["Home Runs", "Walks","Runs Batted In", "Runs",
                                   "Intentional Walks","Hits","Doubles","At Bats",
                                   "Strike Outs","Games","Stolen Bases","Triples"]
  salaries_correlations.to_csv("Salaries Correlations.csv",index=True)
```



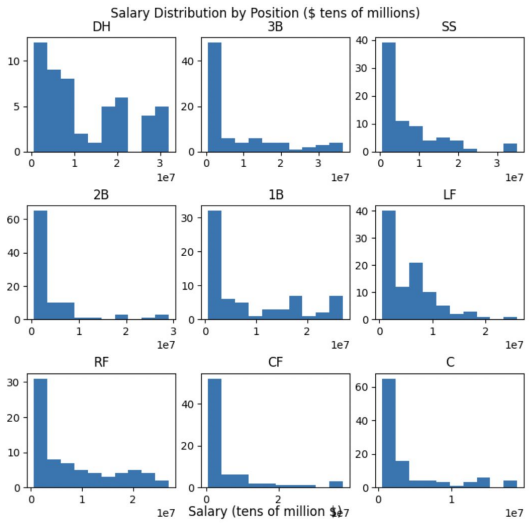
Rank	Stat	Salary
1	Home Runs	0.359445
2	Walks	0.352334
3	Runs Batted In	0.336125
4	Runs	0.325290
5	Intentional Walks	0.313066
6	Hits	0.293024
7	Doubles	0.269721
8	At Bats	0.264524
9	Strike Outs	0.214457
10	Games	0.210125
11	Stolen Bases	0.071550
12	Triples	0.029563

```python
def visual2():
    salaries_correlations = pd.read_csv("Salaries Correlations.csv",index_col=0)
    fig = plt.figure(figsize=(8,6))
    plt.bar(salaries_correlations["Stat"], salaries_correlations["Salary"])
    fig.autofmt_xdate()
    plt.title("Correlations Between Batter Stats and Player Salaries")
    plt.ylabel("Correlation Between Salaries")
    plt.xlabel("Batter Stats")
    plt.show()
```

- .corr()
- Matplotlib
- .bar()

Insight/Visualization #3: Salaries by Position

Rank	Position	Average Salary
1	DH	$12,769,874.31
2	1B	$8,341,460.58
3	3B	$8,214,912.85
4	RF	$7,738,627.84
5	SS	$7,027,504.62
6	CF	$5,458,135.31
7	LF	$5,333,010.08
8	2B	$4,082,042.55
9	C	$3,709,248.03




```python
def insight3():
    salaries_df = pd.read_csv("Player Salaries 2020-2022.csv", index_col=0)
    average_salary_by_position = salaries_df.groupby("Position")["Salary"].mean()
    df_average_salary_by_position = average_salary_by_position.to_frame(name="Average Salary")
    df_average_salary_by_position.reset_index(inplace=True)
    df_average_salary_by_position.sort_values(by="Average Salary", ascending=False, inplace=True)
    df_average_salary_by_position["Average Salary"] = df_average_salary_by_position["Average Salary"].map(lambda x: "${:,.2f}".format(x))
    df_average_salary_by_position.index = np.arange(1, len(df_average_salary_by_position) + 1)
    df_average_salary_by_position.index = df_average_salary_by_position.index.rename("Rank")
    df_average_salary_by_position.to_csv("Average Salary by Position.csv",index=True)
```

```python
def visual3():
  pos = ["DH","3B","SS","2B","1B","LF","RF","CF","C"]
  fig = plt.figure(figsize=(7,7))
  fig.suptitle("Salary Distribution by Position ($ tens of millions)")
  for i in range(len(pos)):

    plt.subplot(3,3,i+1)
    plt.hist(salaries[salaries["Position"] == pos[i]]["Salary"])
    plt.title(pos[i])

  fig.tight_layout(pad=0.5)
  fig.supxlabel("Salary (tens of million $)")
  plt.show()
```

Insight #4: Multiple Regression with sets of stats

```
def insight4():
  batter_stats_and_salaries = pd.merge(batter_stats, salaries[["playerID", "Salary","Year"]], how='inner',left_on=['playerID','yearID'],right_on=["playerID","Year"])
  stats = ["HR","BB","RBI","IBB","H","2B","AB","SO","G","SB","3B"]

  data = []
  for i in stats:
    row = []
    for j in stats:
      if i == j:
        row.append(np.nan)
      else:
        X = batter_stats_and_salaries[[i,j]]
        y = batter_stats_and_salaries["Salary"]

        ols = linear_model.LinearRegression()
        regr = ols.fit(X,y)
        r2 = regr.score(X,y)
        row.append(r2)
    data.append(row)
  mlr_df = pd.DataFrame(data,index=stats,columns=stats)
  n = 0
  row_max = ""
  column_max = ""
  for row in mlr_df:

    for column in mlr_df:
      x = mlr_df.loc[row][column]
      if x > n and np.isnan(x) == False:
        row_max = row
        column_max = column
        n = x

  print(f"{row_max} and {column_max} have the greatest correlation to salary at {round(n,4)} out of all possible pairs of analyzed stats.")
```

Output: BB and G have the greatest correlation to salary at 0.1494 out of all possible pairs of analyzed stats.

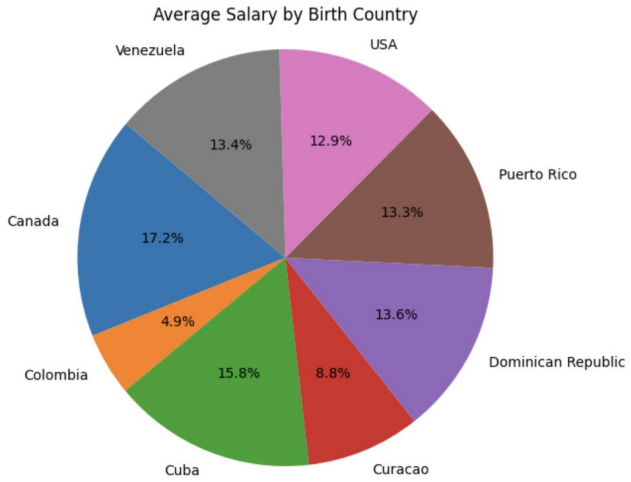
- Sci-kit learn
- MLR

Insight #5: Salaries by Nationality

```python
def insight5():
  player_profiles = pd.read_csv("Player Profiles.csv", delimiter=",",index_col=0)
  salaries = pd.read_csv("Player Salaries 2020-2022.csv",index_col=0)
  profile_and_salaries = pd.merge(player_profiles[["BirthCountry","playerID"]], salaries[["Salary","playerID"]], how='inner',left_on="playerID",right_on="playerID")

  players_per_country = profile_and_salaries["BirthCountry"].value_counts()
  countries_with_5_or_more_players = players_per_country[players_per_country >= 5].index
  filtered_average_salary_df = profile_and_salaries[profile_and_salaries["BirthCountry"].isin(countries_with_5_or_more_players)]


  avg_salary_per_country = filtered_average_salary_df.groupby("BirthCountry")["Salary"].mean()
  average_salary_nationality = pd.DataFrame(avg_salary_per_country).reset_index()
  average_salary_nationality.columns = ["BirthCountry", "AverageSalary"]
  average_salary_nationalit = average_salary_nationality.sort_values(by="AverageSalary", ascending=False)
  average_salary_nationality["AverageSalary"] = average_salary_nationality["AverageSalary"].map("{:,.2f}".format)
  average_salary_nationality = average_salary_nationality.reset_index(drop=True)
  average_salary_nationality = average_salary_nationality.head(10).reset_index(drop=True)
  average_salary_nationality.to_csv("average_salary_nationality.csv", index=False, sep=",")
```



BirthCountry	AverageSalary
Canada	8,804,633.33
Colombia	2,525,516.67
Cuba	8,031,622.33
Curacao	4,510,000.00
Dominican Republic	6,696,720.52
Puerto Rico	6,640,080.44
USA	6,502,744.30
Venezuela	6,480,252.59

```python
def visual4():
  average_salary_nationality = pd.read_csv("average_salary_nationality.csv")
  average_salary_nationality["AverageSalary"] = average_salary_nationality["AverageSalary"].str.replace(',', '').astype(float)
  plt.figure(figsize=(10, 6))
  plt.pie(average_salary_nationality["AverageSalary"], labels=average_salary_nationality["BirthCountry"], autopct='%1.1f%%', startangle=140)
  plt.axis('equal')
  plt.title("Average Salary by Birth Country of Top 100 Earners")
  plt.show()
```

Summary File: Aggregates on Salaries by Position

```python
def summary1():
    salaries_df = pd.read_csv("Player Salaries 2020-2022.csv", index_col=0)

    salary_by_position_summary = salaries_df.groupby("Position").agg({"Salary":['mean','min','max','sum']})
    salary_by_position_summary.columns=["Mean","Min","Max","Sum"]

    salary_by_position_summary.reset_index(inplace=True)
    salary_by_position_summary.sort_values(by="Mean", ascending=False, inplace=True)

    salary_by_position_summary.index = np.arange(1, len(salary_by_position_summary) + 1)
    salary_by_position_summary.index = salary_by_position_summary.index.rename("Rank")

    all_mean = salary_by_position_summary["Mean"].mean()
    all_min = salary_by_position_summary["Min"].min()
    all_max = salary_by_position_summary["Max"].max()
    all_sum = salary_by_position_summary["Sum"].sum()
    salary_by_position_summary.loc[10] = ["All",all_mean,all_min,all_max,all_sum]

    salary_by_position_summary.to_csv("Summary File: Salary Data by Position.csv",index=True)
```

- .groupby()
- .to_csv()

Rank	Position	Mean	Min	Max	Sum
1	DH	1.276987e+07	620000	32000000	664033464
2	1B	8.341461e+06	569500	27000000	558877859
3	3B	8.214913e+06	563500	36571428	673622854
4	RF	7.738628e+06	566025	27000000	564919832
5	SS	7.027505e+06	570000	35100000	534090351
6	CF	5.458135e+06	563500	37766667	403902013
7	LF	5.333010e+06	563500	26000000	506635958
8	2B	4.082043e+06	566000	29000000	383712000
9	C	3.709248e+06	563500	19000000	393180291
10	All	6.963868e+06	563500	37766667	4682974622

Conclusion

- Finding datasets
- Getting data from the web with BeautifulSoup and Regex
- Manipulating data with Pandas
- Visualizing data with Matplotlib and Seaborn
- Using pd.merge() and .groupby()
- The sci-kit learn module