analysis

February 4, 2023

1 Import Dependencies

We begin by importing the necessary libraries.

[1]:
```
# Data analysis
import pandas as pd
pd.options.display.max_columns = None

# Data visualization
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
sns.set_theme(style='whitegrid', palette='pastel')

# Miscellaneous
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

2 Analysis

Then, we proceed with our analyses by reading in the Teams dataframe from the R Lahman package. Note that a separate R script was used to export the dataframe into a csv file.

[2]:
```
# Read data
df = pd.read_csv('data/teams_df.csv')
# Drop duplicate column
df.drop(columns=['Unnamed: 0'], inplace=True)
# Display dataframe
df
```

[2]:
```
      yearID lgID teamID franchID divID  Rank   G  Ghome   W   L DivWin  \
0       1871  NaN    BS1      BNA   NaN     3  31    NaN  20  10    NaN
1       1871  NaN    CH1      CNA   NaN     2  28    NaN  19   9    NaN
2       1871  NaN    CL1      CFC   NaN     8  29    NaN  10  19    NaN
3       1871  NaN    FW1      KEK   NaN     7  19    NaN   7  12    NaN
4       1871  NaN    NY2      NNA   NaN     5  33    NaN  16  17    NaN
…        …   …     …       …    …   …   …     …   …   …      …
```

2980	2021	NL	SLN	STL	C	2	162	81.0	90	72	N
2981	2021	AL	TBA	TBD	E	1	162	81.0	100	62	Y
2982	2021	AL	TEX	TEX	W	5	162	81.0	60	102	N
2983	2021	AL	TOR	TOR	E	4	162	80.0	91	71	N
2984	2021	NL	WAS	WSN	E	5	162	81.0	65	97	N

	WCWin	LgWin	WSWin	R	AB	H	X2B	X3B	HR	BB	SO	SB	\
0	NaN	N	NaN	401	1372	426	70	37	3	60.0	19.0	73.0	
1	NaN	N	NaN	302	1196	323	52	21	10	60.0	22.0	69.0	
2	NaN	N	NaN	249	1186	328	35	40	7	26.0	25.0	18.0	
3	NaN	N	NaN	137	746	178	19	8	2	33.0	9.0	16.0	
4	NaN	N	NaN	302	1404	403	43	21	1	33.0	15.0	46.0	
…	…	…	…	…	…	…	…	…	…	…	…	…	
2980	Y	N	N	706	5351	1303	261	22	198	478.0	1341.0	89.0	
2981	N	N	N	857	5507	1336	288	36	222	585.0	1542.0	88.0	
2982	N	N	N	625	5405	1254	225	24	167	433.0	1381.0	106.0	
2983	N	N	N	846	5476	1455	285	13	262	496.0	1218.0	81.0	
2984	N	N	N	724	5385	1388	272	20	182	573.0	1303.0	56.0	

	CS	HBP	SF	RA	ER	ERA	CG	SHO	SV	IPouts	HA	HRA	BBA	\
0	16.0	NaN	NaN	303	109	3.55	22	1	3	828	367	2	42	
1	21.0	NaN	NaN	241	77	2.76	25	0	1	753	308	6	28	
2	8.0	NaN	NaN	341	116	4.11	23	0	0	762	346	13	53	
3	4.0	NaN	NaN	243	97	5.17	19	1	0	507	261	5	21	
4	15.0	NaN	NaN	313	121	3.72	32	1	0	879	373	7	42	
…	…	…	…	…	…	…	..	…	..	…	…	…	…	
2980	22.0	86.0	44.0	672	626	3.98	3	15	50	4251	1234	152	608	
2981	42.0	72.0	41.0	651	593	3.67	1	13	42	4367	1264	184	436	
2982	29.0	58.0	31.0	815	758	4.79	0	3	31	4273	1402	232	513	
2983	20.0	51.0	35.0	663	610	3.91	1	14	34	4216	1257	209	473	
2984	26.0	84.0	31.0	820	743	4.80	1	8	36	4183	1364	247	548	

	SOA	E	DP	FP	name	\
0	23	243	24	0.834	Boston Red Stockings	
1	22	229	16	0.829	Chicago White Stockings	
2	34	234	15	0.818	Cleveland Forest Citys	
3	17	163	8	0.803	Fort Wayne Kekiongas	
4	22	235	14	0.840	New York Mutuals	
…	…	…	…	…	…	
2980	1225	84	137	0.986	St. Louis Cardinals	
2981	1478	80	130	0.986	Tampa Bay Rays	
2982	1239	83	146	0.986	Texas Rangers	
2983	1468	90	122	0.984	Toronto Blue Jays	
2984	1346	96	116	0.983	Washington Nationals	

	park	attendance	BPF	PPF	teamIDBR	\
0	South End Grounds I	NaN	103	98	BOS	

```
1          Union Base-Ball Grounds         NaN  104  102       CHI
2     National Association Grounds         NaN   96  100       CLE
3                   Hamilton Field         NaN  101  107       KEK
4         Union Grounds (Brooklyn)         NaN   90   88       NYU
…                              …        …    …    …        …
2980             Busch Stadium III   2102530.0   92   92       STL
2981               Tropicana Field    761072.0   92   91       TBR
2982              Globe Life Field   2110258.0   99  101       TEX
2983                  Sahlen Field    805901.0  102  101       TOR
2984                Nationals Park   1465543.0   95   96       WSN

     teamIDlahman45 teamIDretro
0               BS1         BS1
1               CH1         CH1
2               CL1         CL1
3               FW1         FW1
4               NY2         NY2
…               …           …
2980            SLN         SLN
2981            TBA         TBA
2982            TEX         TEX
2983            TOR         TOR
2984            MON         WAS

[2985 rows x 48 columns]
```

We print a high-level overview of the data to better understand each column.

[3]:
```
# Print high-level info of df
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2985 entries, 0 to 2984
Data columns (total 48 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   yearID          2985 non-null   int64
 1   lgID            2935 non-null   object
 2   teamID          2985 non-null   object
 3   franchID        2985 non-null   object
 4   divID           1468 non-null   object
 5   Rank            2985 non-null   int64
 6   G               2985 non-null   int64
 7   Ghome           2586 non-null   float64
 8   W               2985 non-null   int64
 9   L               2985 non-null   int64
 10  DivWin          1440 non-null   object
 11  WCWin           804 non-null    object
```

```
 12  LgWin           2957 non-null   object
 13  WSWin           2628 non-null   object
 14  R               2985 non-null   int64
 15  AB              2985 non-null   int64
 16  H               2985 non-null   int64
 17  X2B             2985 non-null   int64
 18  X3B             2985 non-null   int64
 19  HR              2985 non-null   int64
 20  BB              2984 non-null   float64
 21  SO              2969 non-null   float64
 22  SB              2859 non-null   float64
 23  CS              2153 non-null   float64
 24  HBP             1827 non-null   float64
 25  SF              1444 non-null   float64
 26  RA              2985 non-null   int64
 27  ER              2985 non-null   int64
 28  ERA             2985 non-null   float64
 29  CG              2985 non-null   int64
 30  SHO             2985 non-null   int64
 31  SV              2985 non-null   int64
 32  IPouts          2985 non-null   int64
 33  HA              2985 non-null   int64
 34  HRA             2985 non-null   int64
 35  BBA             2985 non-null   int64
 36  SOA             2985 non-null   int64
 37  E               2985 non-null   int64
 38  DP              2985 non-null   int64
 39  FP              2985 non-null   float64
 40  name            2985 non-null   object
 41  park            2951 non-null   object
 42  attendance      2706 non-null   float64
 43  BPF             2985 non-null   int64
 44  PPF             2985 non-null   int64
 45  teamIDBR        2985 non-null   object
 46  teamIDlahman45  2985 non-null   object
 47  teamIDretro     2985 non-null   object
dtypes: float64(10), int64(25), object(13)
memory usage: 1.1+ MB
```

Notice, `yearID` and `lgID` seem to be features of interest, since they match the given conditions in the initial prompt. Also, `R` appears to be our target feature.

Let's better understand the `lgID` column by computing a normalized count of each distinct value.

[4]:
```python
# Perform a normalized count of the values in lgID, expressed as a percentage
df['lgID'].value_counts(normalize=True).apply(lambda x: f'{x * 100:.2f}%')
```

[4]:
```
NL    51.75%
AL    44.12%
AA     2.90%
FL     0.55%
UA     0.41%
PL     0.27%
Name: lgID, dtype: object
```

As expected, `NL` (a.k.a. National League) and `AL` (a.k.a. American League) make up the majority of entries.

Naturally, we perform the necessary aggregations by computing the mean number of runs as a function of `yearID` and `lgID`, filtering for entries between the years 1980 & 2019.

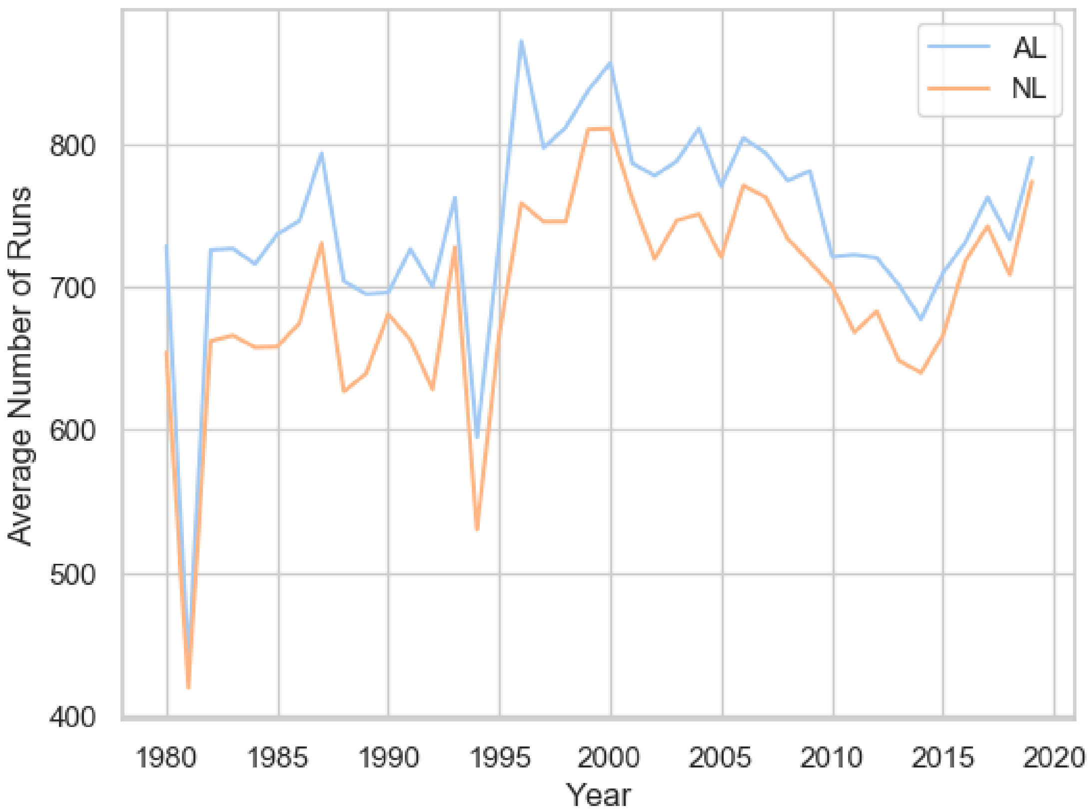
[5]:
```python
# Filter by year -> group by year and league -> compute mean of R
data = df[(df['yearID'] >= 1980) & (df['yearID'] <= 2019)][['yearID', 'lgID', 'R']].groupby(['yearID', 'lgID']).mean().reset_index()
# Drop columns where lgID is not AL or NL
data = data[data['lgID'].isin(['AL', 'NL'])]
# Display results
data
```

[5]:
```
    yearID lgID           R
0     1980   AL  728.642857
1     1980   NL  654.333333
2     1981   AL  436.571429
3     1981   NL  419.583333
4     1982   AL  725.928571
..     …  …         …
75    2017   NL  742.600000
76    2018   AL  733.266667
77    2018   NL  708.733333
78    2019   AL  790.600000
79    2019   NL  773.866667

[80 rows x 3 columns]
```

It remains to visualize the trends in the data via a line plot.

[6]:
```python
# Create line plot
sns.lineplot(x='yearID', y='R', hue='lgID', data=data)
# Customize axes labels
plt.xlabel('Year')
plt.ylabel('Average Number of Runs')
# Create legend
plt.legend(loc='best')
# Display plot
plt.show()
```


AL
NL
Average Number of Runs
400
500
600
700
800
1980
1985
1990
1995
2000
2005
2010
2015
2020
Year