project2

March 20, 2022

0.1 Jay Desmarais CMSC320 Project 2: Moneyball

[1]:
```
import sqlite3
import pandas
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import re

pandas.options.display.max_rows = 200
```

0.1.1 Part 1: Wrangling

Problem 1

[2]:
```
sqlite_file = 'lahman2014.sqlite'
conn = sqlite3.connect(sqlite_file)

# Construct the pieces of the SQL query.
select_clause = "SELECT Teams.yearID, Teams.teamID, franchID, W, G, sum(salary)␣
↪as total_payroll, cast(W as float) * 100 / cast(G as float) as win_pct "
from_clause = "FROM Salaries, Teams "
where_clause = "WHERE (Salaries.yearID == Teams.yearID AND Salaries.teamID ==␣
↪Teams.teamID) "
groupby_clause = "GROUP BY Salaries.yearID, Salaries.teamID "

# Construct the completed SQL query for team efficiency.
efficiency_query = select_clause + from_clause + where_clause + groupby_clause

# Use pandas to execute the SQL query and read the results.
team_data = pandas.read_sql(efficiency_query, conn)

# Display the pandas dataframe.
team_data

# There is no missing data in the final table because an inner join was used␣
↪that checked for both table's content for matching yearIDs and teamIDs.
```

```
# This prevented any data to come through where one table contained an entry␣
↪and aother did not.
# If either of the individual tables was missing data on one team from one year␣
↪that the other table had, it was not joined to the new table.
# This will allow us to compare only the teams for which we have both the data␣
↪of their payroll and their winning data.
```

[2]:

```
     yearID teamID franchID   W    G  total_payroll    win_pct
0      1985    ATL      ATL  66  162     14807000.0  40.740741
1      1985    BAL      BAL  83  161     11560712.0  51.552795
2      1985    BOS      BOS  81  163     10897560.0  49.693252
3      1985    CAL      ANA  90  162     14427894.0  55.555556
4      1985    CHA      CHW  85  163      9846178.0  52.147239
..      …    …      …  ..  …            …        …
853    2014    SLN      STL  90  162    120693000.0  55.555556
854    2014    TBA      TBD  77  162     72689100.0  47.530864
855    2014    TEX      TEX  67  162    112255059.0  41.358025
856    2014    TOR      TOR  83  162    109920100.0  51.234568
857    2014    WAS      WSN  96  162    131983680.0  59.259259

[858 rows x 7 columns]
```

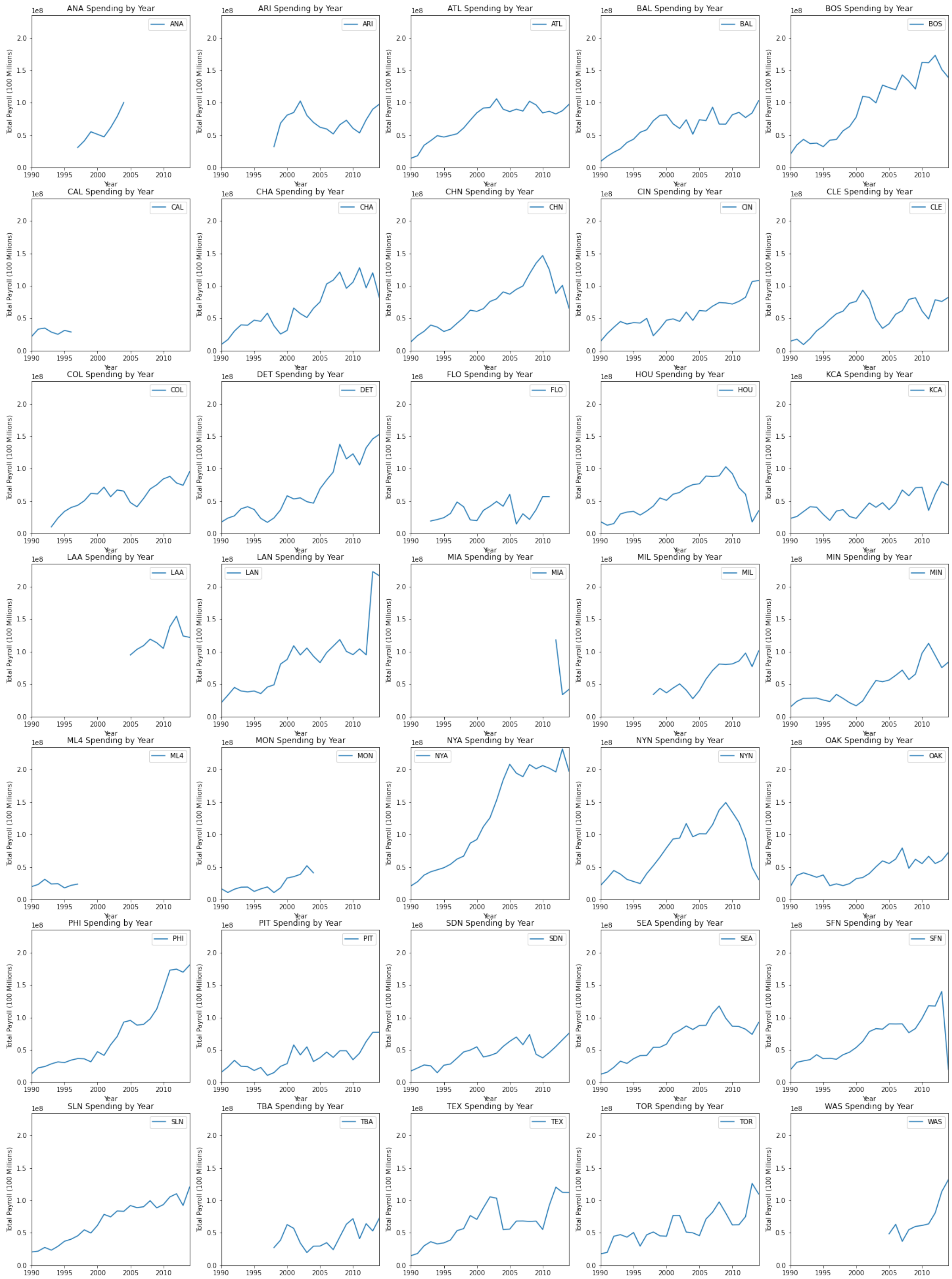
0.1.2 Part 2: Exploratory Data Analysis

Problem 2

[3]:

```
# Create the figure and subplots.
fig, ax = plt.subplots(7, 5, figsize=(25, 35))

# Set the payroll values to integers.
data = team_data.astype({'total_payroll': int}, errors='raise')

# Groupd the data by team.
grouped = team_data.groupby('teamID')

# Increment through each group and add its values to a subplot.
i = 0; j= 0
for key, group in grouped:
    group.plot(\
        ax=ax[j][i],\
        kind='line',\
        x='yearID',\
        y='total_payroll',\
        label=key,\
        xlim=(1990,2014),\
        ylim=(0, 235000000))
    ax[j][i].set_title('{} Spending by Year'.format(key))
    ax[j][i].set_xlabel('Year')
```

```python
    ax[j][i].set_ylabel('Total Payroll (100 Millions)')
    if i > 3:
        j += 1; i = 0
    else:
        i += 1

# I chose to seperate each team into a different subplot here and plot all the
 ↪values from year 1990 - 2014
# along with all payrolls (in hundreds of millions) from 0 to 2.35. This will
 ↪allow us to see the same graph
# and axis ticks and spread to easily compare all charts without it getting too
 ↪crowded or requiring the
# consideration of different axis values.
```


ANA Spending by Year
ARI Spending by Year
ATL Spending by Year
BAL Spending by Year
BOS Spending by Year
CAL Spending by Year
CHA Spending by Year
CHN Spending by Year
CIN Spending by Year
CLE Spending by Year
COL Spending by Year
DET Spending by Year
FLO Spending by Year
HOU Spending by Year
KCA Spending by Year
LAA Spending by Year
LAN Spending by Year
MIA Spending by Year
MIL Spending by Year
MIN Spending by Year
ML4 Spending by Year
MON Spending by Year
NYA Spending by Year
NYN Spending by Year
OAK Spending by Year
PHI Spending by Year
PIT Spending by Year
SDN Spending by Year
SEA Spending by Year
SFN Spending by Year
SLN Spending by Year
TBA Spending by Year
TEX Spending by Year
TOR Spending by Year
WAS Spending by Year
Year
Total Payroll (100 Millions)
1e8

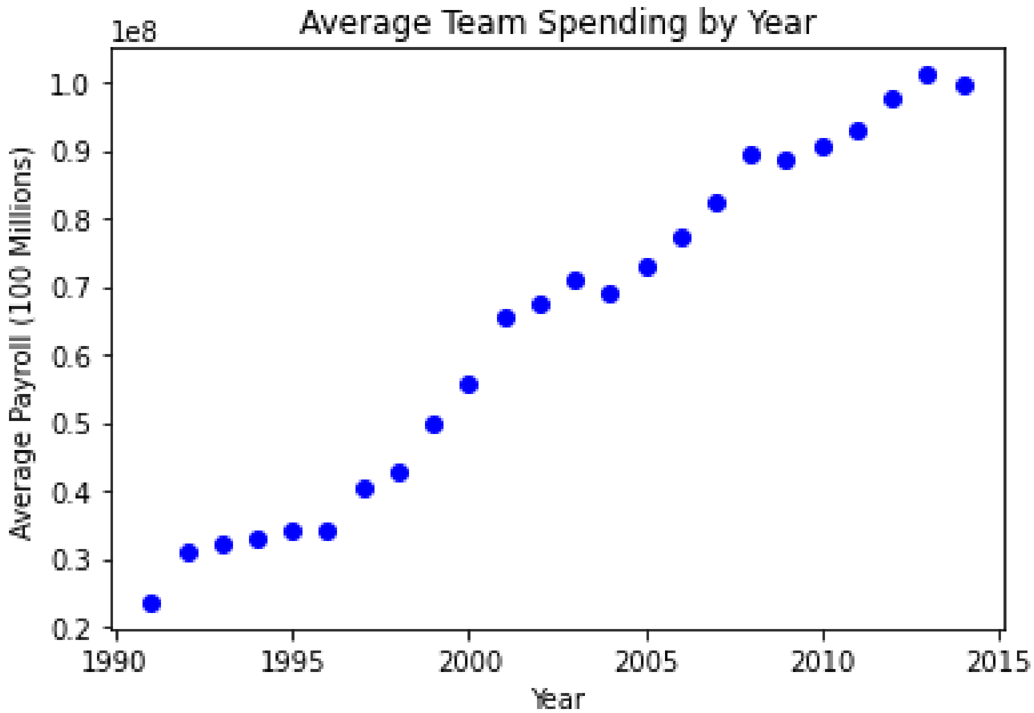
Question 1

1. In these plots, it is evident that there is a trend for payrolls to increase over time.
2. There is a medium amount of spread in this data as a majority of the data is concentrated around the 25 - 120 million range, but some values are at the larger extreme, closer to and even surpassing 200 million.
3. There are some outliers as some teams spend a significant more each year than the average team, skewing the data to the right.

Problem 3

[4]:
```
# Create the figure and subplot.
fig, ax = plt.subplots()

# Group the date by year.
year_avg_payroll = team_data.loc[team_data['yearID'] > 1990]
year_avg_payroll = year_avg_payroll.groupby('yearID').agg({'total_payroll':␣
↪'mean'})

# Plot the x and y values calculated.
x = year_avg_payroll.index.tolist()
y = year_avg_payroll['total_payroll'].values
ax.scatter(x, y, color='blue')

ax.set_title('Average Team Spending by Year')
ax.set_xlabel('Year')
ax.set_ylabel('Average Payroll (100 Millions)')

# This graph show the mean total payroll across all teams for each year from␣
↪1990 - 2014.
# This shows a general upwards trend when averaging all teams payrolls,␣
↪confirming statement 1 from question 1.
```

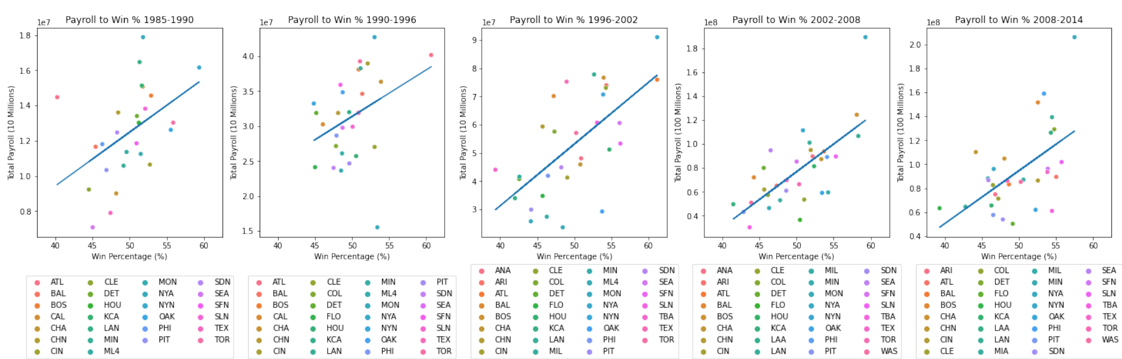
[4]:
```
Text(0, 0.5, 'Average Payroll (100 Millions)')
```



Problem 4

[5]:
```
# Initialize the figure and subplots.
fig, ax = plt.subplots(1, 5, figsize=(25, 5))

# Create 5 equal groupings of data by year.
team_data['bin'] =  pandas.cut(team_data['yearID'], 5, precision = 1)

# Group the data and find averages across bins.
grouped_efficiency = team_data.groupby(['bin', 'teamID']).agg({'total_payroll':
 ↪'mean', 'win_pct': 'mean'})
# Drop any data where a team has no averages for a bin and reset the indices.
grouped_efficiency = grouped_efficiency.dropna()
grouped_efficiency = grouped_efficiency.reset_index()

# Group the data by the bins for easy plotting.
grouped = grouped_efficiency.groupby('bin')

# Plot each grouping of years.
i = 0
for key, group in grouped:
    ax[i] = sns.scatterplot(\
        data=group,
        ax = ax[i],
```

```
        x='win_pct',
        y='total_payroll',
        hue='teamID'
    )

    # Grab arrays of the x any y values for later use.
    xaxis = group['win_pct'].tolist()
    yaxis = group['total_payroll'].tolist()

    # Plot a regression line for best fit for ease of interpretation.
    trend = np.polyfit(xaxis, yaxis, 1)
    trendpoly = np.poly1d(trend)
    ax[i].plot(xaxis, trendpoly(xaxis))

    # Rename and reformat plot labels, axis, and legend.
    years = re.findall(r'([0-9]{4})', str(key))
    ax[i].legend(loc='lower center', bbox_to_anchor=(.5, -.6), ncol=4)
    ax[i].set(xlim = (37.5, 62.5))
    ax[i].set_title('Payroll to Win % {}-{}'.format(years[0], years[1]))
    ax[i].set_xlabel('Win Percentage (%)')
    if group['total_payroll'].max() >= 100000000:
        ax[i].set_ylabel('Total Payroll (100 Millions)')
    else:
        ax[i].set_ylabel('Total Payroll (10 Millions)')
    i += 1

# The below graphs demonstrate each team's ability to turn their money into wins.
# This is done by plotting the win % in relation to the total payroll used.
# The graphs are split into years of 5 or 6 periods.
# This is done to get an idea of the relationship in question without needing to consider increased costs as the years progress.
# A regression line is also fitted to each plot for the ease of interpretation.
```



Question 2 Across these periods, team payrolls got significantly bigger. There could be many causes for this, but in the end, more money was spent each year. There are a few teams that stand out as spending more, namely NYA and BOS, which goes hand in hand with the movie Moneyball, where remarks are made about teams not being able to keep up with those higher spending teams, and this would appear to be true as the NYN rank at the upper end of win percentage for each period, but there are lots of teams that use their money much more efficiently, like OAK. OAK consistently lands in the upper middle of the pack in terms of win %, but their spending is on the lower end of the scale, which can be seen as they land significatly below the regression line 4 out of 5 periods.

0.1.3 Part 3: Data Transformations

Problem 5

[6]:
```
# Group the data by year.
team_years = team_data.groupby('yearID')

# For the data at each year, do the following:
for key, group in team_years:
    # Calculate the mean total_payroll.
    avg_payroll = group['total_payroll'].mean()
    # Calculate the standard deviation of the total_payroll.
    s = group['total_payroll'].std()
    # Calculate a standardized_payroll for the year and add it to the table.
    team_data.loc[team_data["yearID"] == key ,"standardized_payroll"] =␣
 ↪((team_data['total_payroll'] - avg_payroll) / s)

team_data
```

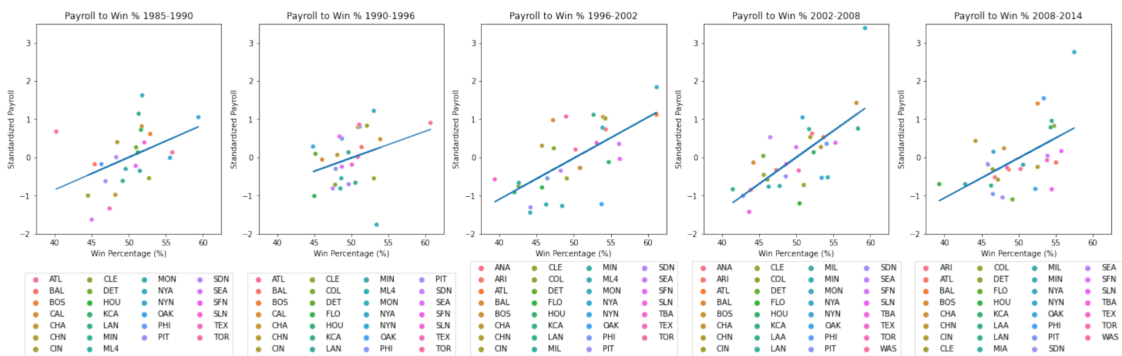
[6]:
```
     yearID teamID franchID   W    G  total_payroll    win_pct  \
0      1985    ATL      ATL  66  162     14807000.0  40.740741
1      1985    BAL      BAL  83  161     11560712.0  51.552795
2      1985    BOS      BOS  81  163     10897560.0  49.693252
3      1985    CAL      ANA  90  162     14427894.0  55.555556
4      1985    CHA      CHW  85  163      9846178.0  52.147239
..      …      …        …  ..  …            …          …
853    2014    SLN      STL  90  162    120693000.0  55.555556
854    2014    TBA      TBD  77  162     72689100.0  47.530864
855    2014    TEX      TEX  67  162    112255059.0  41.358025
856    2014    TOR      TOR  83  162    109920100.0  51.234568
857    2014    WAS      WSN  96  162    131983680.0  59.259259

                  bin  standardized_payroll
0    (1985.0, 1990.8]              1.914905
1    (1985.0, 1990.8]              0.601068
2    (1985.0, 1990.8]              0.332678
3    (1985.0, 1990.8]              1.761474
4    (1985.0, 1990.8]             -0.092838
```

```
..                  ...                 ...
853   (2008.2, 2014.0]             0.457126
854   (2008.2, 2014.0]            -0.593171
855   (2008.2, 2014.0]             0.272509
856   (2008.2, 2014.0]             0.221422
857   (2008.2, 2014.0]             0.704160

[858 rows x 9 columns]
```

Problem 6

[7]:
```python
# Initialize the figure and subplots.
fig, ax = plt.subplots(1, 5, figsize=(25, 5))

# Group the data and find averages across bins.
standardized_efficiency = team_data.groupby(['bin', 'teamID']).
 ↪agg({'standardized_payroll': 'mean', 'win_pct': 'mean'})
# Drop any data where a team has no averages for a bin and reset the indices.
standardized_efficiency = standardized_efficiency.dropna()
standardized_efficiency = standardized_efficiency.reset_index()

# Group the data by the bins for easy plotting.
grouped = standardized_efficiency.groupby('bin')

# Plot each grouping of years.
i = 0
for key, group in grouped:
    ax[i] = sns.scatterplot(\
        data=group,
        ax = ax[i],
        x='win_pct',
        y='standardized_payroll',
        hue='teamID'
    )

    # Grab arrays of the x any y values for later use.
    xaxis = group['win_pct'].tolist()
    yaxis = group['standardized_payroll'].tolist()

    # Plot a regression line for best fit for ease of interpretation.
    trend = np.polyfit(xaxis, yaxis, 1)
    trendpoly = np.poly1d(trend)
    ax[i].plot(xaxis, trendpoly(xaxis))

    # Rename and reformat plot labels, axis, and legend.
    years = re.findall(r'([0-9]{4})', str(key))
    ax[i].legend(loc='lower center', bbox_to_anchor=(.5, -.6), ncol=4)
```

```
    ax[i].set(xlim = (37.5, 62.5))
    ax[i].set(ylim = (-2, 3.5))
    ax[i].set_title('Payroll to Win % {}-{}'.format(years[0], years[1]))
    ax[i].set_xlabel('Win Percentage (%)')
    ax[i].set_ylabel('Standardized Payroll')
    i += 1

# These charts have a very similar purpose to those displayed as a part of␣
↪problem 4.
# Instead of having a total payroll on the y axis, these charts display a␣
↪standardized payroll.
# This helps to improve comparison of data year to year without needing to␣
↪consider varying price relations across years.
```



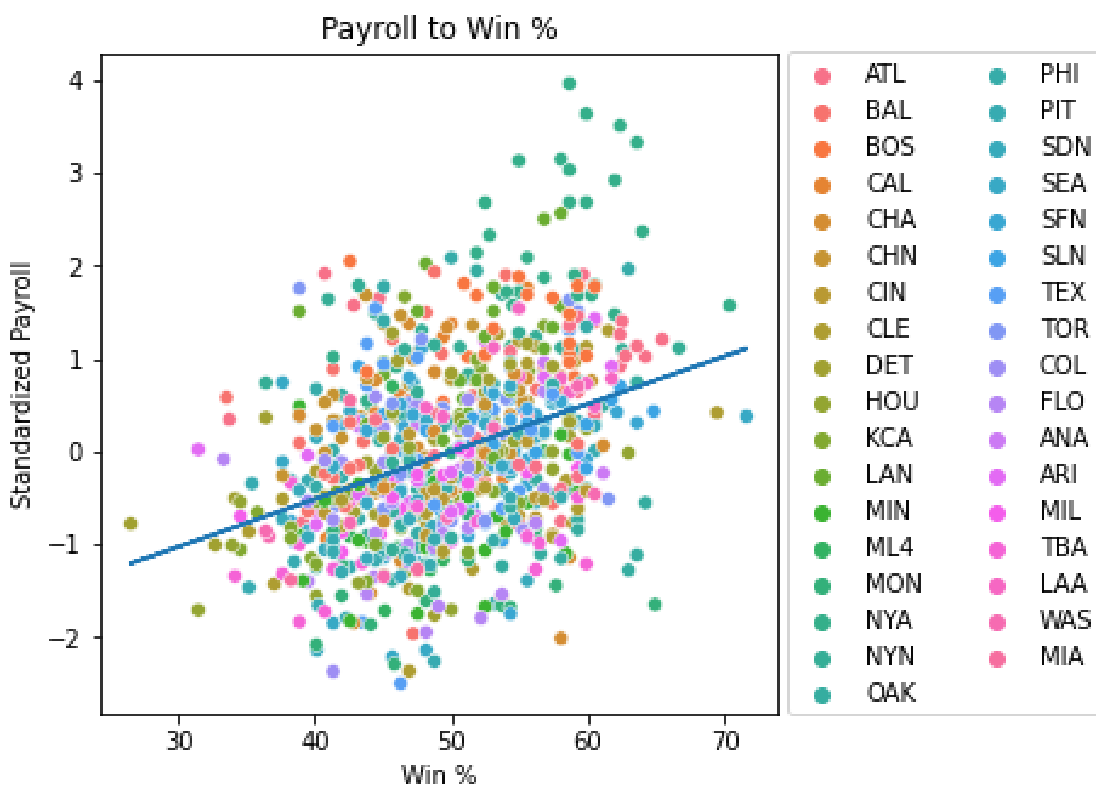
Question 3 The changes made in problem 5 are reflected in the difference between problems 4 and 6 by standardizing the variable. This makes it so the data is easier to compare to each other and the linear regression. By standardizing a variable, there is no longer a significant difference in variable values, and instead an observer can concentrate between two variables relationship to each other. Although the data in it's bare bones is not all that different, the plots now contain the same axis values and are, like previously mentioned, much easier to compare and analyze.

Problem 7

[8]:
```
# Initialize the figure and subplots.
fig, ax = plt.subplots(figsize=(5,5))

# Plot the x and y values calculated.
ax = sns.scatterplot(\
    data=team_data,
    x='win_pct',
    y='standardized_payroll',
    hue='teamID'
)
```

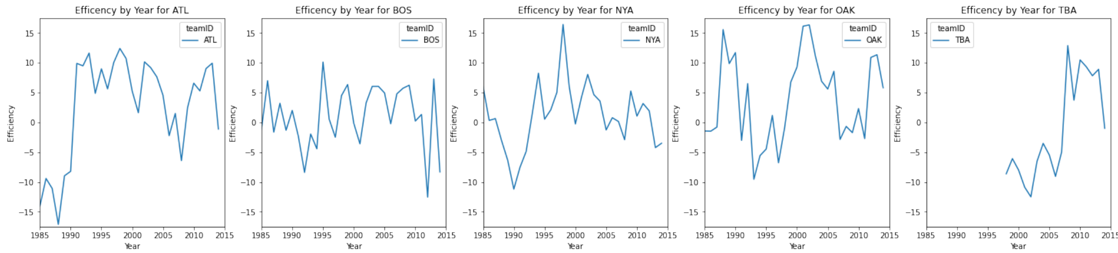
```
# Grab arrays of the x any y values for later use.
xaxis = team_data['win_pct'].tolist()
yaxis = team_data['standardized_payroll'].tolist()

# Plot a regression line for best fit for ease of interpretation.
trend = np.polyfit(xaxis, yaxis, 1)
trendpoly = np.poly1d(trend)
ax.plot(xaxis, trendpoly(xaxis))

# Rename and reformat plot labels, axis, and legend.
ax.set_title('Payroll to Win %')
ax.set_xlabel('Win %')
ax.set_ylabel('Standardized Payroll')
ax.legend(loc='center right', bbox_to_anchor=(1.5, .5), ncol=2)

# This chart shows an aggregate collection of all teams standardized payrolls␣
↪and the corresponding wins across all years.
# The regression line in this chart demonstrates that there is a correlation␣
↪between the amount a team spends and how much they win.
```

[8]: <matplotlib.legend.Legend at 0x7fa2de659cd0>



Problem 8

[10]:
```
# Initialize the figure and subplots.
fig, ax = plt.subplots(1, 5, figsize=(25, 5))

for key, group in team_years:
    team_data.loc[team_data["yearID"] == key ,"expected_win_pct"] =␣
 ↪((team_data['standardized_payroll'] * 2.5) + 50)
    team_data.loc[team_data["yearID"] == key ,"efficiency"] =␣
 ↪(team_data['win_pct'] - team_data['expected_win_pct'])

teams = ["OAK", "BOS", "NYA", "ATL", "TBA"]

select_efficiency = team_data.loc[team_data['teamID'].isin(teams)]
select_efficiency = select_efficiency.reset_index()

select_efficiency = select_efficiency.groupby('teamID')

i = 0
for key, group in select_efficiency:
    ax[i] = sns.lineplot(\
        data=group,
        ax = ax[i],
        x='yearID',
        y='efficiency',
        hue='teamID'
    )

    # Rename and reformat plot labels, axis, and legend.
    years = re.findall(r'([0-9]{4})', str(key))
    ax[i].set_title('Efficency by Year for {}'.format(key))
    ax[i].set_xlabel('Year')
    ax[i].set_ylabel('Efficiency')
    ax[i].set_ylim(-17.5, 17.5)
    ax[i].set_xlim(1985, 2015)
    i += 1

# These charts show the efficiency of a team, which loosely translates to their␣
 ↪ability to buy wins with as little money as possible.
# These graphs display a few teams over many years and can help to determine␣
 ↪how much a team improves their spending year to year.
```



Question 4 These charts are much different than looking at those from questions 2 and 3. Prreviously, we were looking at team win % compared to their payroll. Here, we user calculations to determine what a teams efficiency rating is based on those numbers, and look at their efficiency over time more directly. Similar conclusions can be drawn from each board, but the teams ability to turn money into wins is more simply put on the graphs directly above. Oakland's efficienccy during the moneyball period was very good, sitting at 15 between 2000 and 2005 after a huge jump from the negatives a few years prior. This shows that the team found a great way to put their money to better use and translate more directly to wins.