# The Salary-Performance Relationship in the English Premier League

Last week we looked at the salary-performance relationship in the NBA. The rules governing the operation of the English Premier League are very different. Unlike the NBA, there is no salary cap, nor is there a draft system, roster limits, and the revenue sharing mechanisms used in the NBA and other North American major leagues are much more limited.

Another important difference, which is not unconnnected, is the system of promotion and relegation. This requires that the worst performing teams in the league (measured by league position) are automatically relegated the following season to play in the next tier down, to be replaced by the best performing teams from that lower tier. This system is the norm in the world of soccer, and is perhaps the main reason that teams do not agree to restraints such as salary caps. There is a high degree of revenue inequality in soccer, and richer clubs are unwilling to share with the poorer ones, for fear that this might cause them to be relegated.

This week we are going to follow the same procedure as we did for the NBA. We will look at the impact of salaries (relative to the average for the season) on team performance (measured this time by league position), and then see how the addition of potential omitted variables - (the lagged depedent variable and fixed effects) impact the estimates.

In [1]:
```
# As usual, we begin by loading the packages we will need

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
```

In [2]:
```
# Now we load the data

EPL=pd.read_excel("../../Data/Week 5/EPL pay and performance.xlsx")
```

We use .describe() to look at the summary statistics for the data. From this we can see that we have 380 observations, for teams running from 1997 to 2015 (19 seasons). Our two main variables of interest are win percentage and team salaries. We also include a dummy variable for whether the team had been promoted that season. We can also use .info() to summarize the dataframe.

In [3]:
```
EPL.describe()
```

Out[3]:

| | Season_ending | promoted_last_season | Position | Revenues | salaries |
|---|---|---|---|---|---|
| count | 380.000000 | 380.000000 | 380.000000 | 3.750000e+02 | 3.750000e+02 |
| mean | 2006.000000 | 0.150000 | 10.500000 | 8.431025e+07 | 5.183637e+07 |
| std | 5.484447 | 0.357542 | 5.773884 | 7.533617e+07 | 4.331005e+07 |
| min | 1997.000000 | 0.000000 | 1.000000 | 9.238238e+06 | 4.172024e+06 |
| 25% | 2001.000000 | 0.000000 | 5.750000 | 3.895834e+07 | 2.413900e+07 |
| 50% | 2006.000000 | 0.000000 | 10.500000 | 5.907200e+07 | 3.774400e+07 |
| 75% | 2011.000000 | 0.000000 | 15.250000 | 9.753050e+07 | 6.300007e+07 |
| max | 2015.000000 | 1.000000 | 20.000000 | 4.331640e+08 | 2.331060e+08 |

In [4]:
```
EPL.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 380 entries, 0 to 379
Data columns (total 6 columns):
Season_ending           380 non-null int64
Club                    380 non-null object
promoted_last_season    380 non-null int64
Position                380 non-null int64
Revenues                375 non-null float64
salaries                375 non-null float64
dtypes: float64(2), int64(3), object(1)
memory usage: 17.9+ KB
```

We use .groupby to sum salaries. Note the way we use .reset_index and .rename(columns ={}) to make the data manageable.

In [5]:
```
Sumsal = EPL.groupby(['Season_ending'])['salaries'].sum().reset_index().rename(columns={'salaries':'allsal'})
Sumsal
```

Out[5]:

| | Season_ending | allsal |
|---|---|---|
| 0 | 1997 | 2.195995e+08 |
| 1 | 1998 | 2.896696e+08 |
| 2 | 1999 | 3.900185e+08 |
| 3 | 2000 | 4.673185e+08 |
| 4 | 2001 | 5.622860e+08 |
| 5 | 2002 | 6.765034e+08 |
| 6 | 2003 | 7.477382e+08 |
| 7 | 2004 | 7.980298e+08 |
| 8 | 2005 | 7.836889e+08 |
| 9 | 2006 | 8.671860e+08 |
| 10 | 2007 | 9.506965e+08 |
| 11 | 2008 | 1.188491e+09 |
| 12 | 2009 | 1.248423e+09 |
| 13 | 2010 | 1.332551e+09 |
| 14 | 2011 | 1.583955e+09 |
| 15 | 2012 | 1.626853e+09 |
| 16 | 2013 | 1.782494e+09 |
| 17 | 2014 | 1.891789e+09 |
| 18 | 2015 | 2.031348e+09 |

For numbers that have many digits, Python will typically use scientific notation to represent them. If you would rather see the data in standard decimal format

you can use "pd.options.display.float_format = ". In the line below we show the salary data with no decimal places- '{:.0f}' - if you wanted the data to one decimal place you would write '{:.1f}' - this way you can format the data to as many decimal places as you prefer. Another option, if you would prefer to see the numbers in a format that is easier to read from the screen-in this case would be to divide allsal by 1,000,000.

In [6]:
```
pd.options.display.float_format = '{:.0f}'.format
Sumsal
```

Out[6]:

| | Season_ending | allsal |
|---|---|---|
| 0 | 1997 | 219599462 |
| 1 | 1998 | 289669601 |
| 2 | 1999 | 390018517 |
| 3 | 2000 | 467318483 |
| 4 | 2001 | 562286010 |
| 5 | 2002 | 676503369 |
| 6 | 2003 | 747738215 |
| 7 | 2004 | 798029773 |
| 8 | 2005 | 783688898 |
| 9 | 2006 | 867186039 |
| 10 | 2007 | 950696528 |
| 11 | 2008 | 1188491236 |
| 12 | 2009 | 1248422708 |
| 13 | 2010 | 1332551287 |
| 14 | 2011 | 1583955432 |
| 15 | 2012 | 1626852832 |
| 16 | 2013 | 1782493515 |
| 17 | 2014 | 1891788759 |
| 18 | 2015 | 2031348184 |

As with the NBA, the sharp upward trend in total salaries is clearly visible. allsal increased from £220 million in 1997 to £2031 million in 2015. In each season we want to compare team spending relative to the average of that season.

To do this we now merge the aggregate salaries back in to the main dataframe and then divide the team's salary bill in each year by allsal in that year.

In [7]:
```
EPL = pd.merge(EPL, Sumsal, on=['Season_ending'], how='left')
display(EPL)
```

| | Season_ending | Club | promoted_last_season | Position | Revenues | salaries | allsal |
|---|---|---|---|---|---|---|---|
| 0 | 1997 | Arsenal | 0 | 3 | 27158007 | 15279000 | 219599462 |
| 1 | 1997 | Aston Villa | 0 | 5 | 22079000 | 10070000 | 219599462 |
| 2 | 1997 | Blackburn Rovers | 0 | 13 | 14302220 | 14336629 | 219599462 |
| 3 | 1997 | Chelsea | 0 | 6 | 23729000 | 14873000 | 219599462 |
| 4 | 1997 | Coventry City | 0 | 17 | 12264825 | 8396261 | 219599462 |
| 5 | 1997 | Derby County | 1 | 12 | 10737571 | 6406557 | 219599462 |
| 6 | 1997 | Everton | 0 | 15 | 18881961 | 10933090 | 219599462 |
| 7 | 1997 | Leeds United | 0 | 11 | 22249000 | 12312000 | 219599462 |
| 8 | 1997 | Leicester City | 1 | 9 | 17320000 | 8914000 | 219599462 |
| 9 | 1997 | Liverpool | 0 | 4 | 39394000 | 14559000 | 219599462 |
| 10 | 1997 | Manchester United | 0 | 1 | 87939000 | 21847000 | 219599462 |
| 11 | 1997 | Middlesbrough | 0 | 19 | 22502000 | 11332000 | 219599462 |
| 12 | 1997 | Newcastle United | 0 | 2 | 39874000 | 16277000 | 219599462 |
| 13 | 1997 | Nottingham Forest | 0 | 20 | 14435000 | 9640000 | 219599462 |
| 14 | 1997 | Sheffield Wednesday | 0 | 7 | 14335000 | 7571000 | 219599462 |
| 15 | 1997 | Southampton | 0 | 16 | 9238238 | 4776538 | 219599462 |
| 16 | 1997 | Sunderland | 1 | 18 | 13415000 | 5703000 | 219599462 |
| 17 | 1997 | Tottenham Hotspur | 0 | 10 | 27874000 | 12057000 | 219599462 |
| 18 | 1997 | West Ham United | 0 | 14 | 15256000 | 8298000 | 219599462 |
| 19 | 1997 | Wimbledon | 0 | 8 | 10684711 | 6018387 | 219599462 |
| 20 | 1998 | Arsenal | 0 | 1 | 40391000 | 21882000 | 289669601 |
| 21 | 1998 | Aston Villa | 0 | 7 | 31769000 | 12388000 | 289669601 |
| 22 | 1998 | Barnsley | 1 | 19 | nan | 4172024 | 289669601 |
| 23 | 1998 | Blackburn Rovers | 0 | 6 | 19355956 | 19034682 | 289669601 |
| 24 | 1998 | Bolton Wanderers | 1 | 18 | 15711326 | 8953766 | 289669601 |
| 25 | 1998 | Chelsea | 0 | 4 | 88307136 | 26982223 | 289669601 |
| 26 | 1998 | Coventry City | 0 | 11 | 17399739 | 10423348 | 289669601 |
| 27 | 1998 | Crystal Palace | 1 | 20 | nan | nan | 289669601 |
| 28 | 1998 | Derby County | 0 | 9 | 20058636 | 11776215 | 289669601 |
| 29 | 1998 | Everton | 0 | 17 | 22664943 | 13845059 | 289669601 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 350 | 2014 | Manchester United | 0 | 7 | 433164000 | 214677000 | 1891788759 |
| 351 | 2014 | Newcastle United | 0 | 10 | 129745000 | 78297000 | 1891788759 |
| 352 | 2014 | Norwich City | 0 | 18 | 94345000 | 54059000 | 1891788759 |
| 353 | 2014 | Southampton | 0 | 8 | 106098759 | 62951136 | 1891788759 |
| 354 | 2014 | Stoke City | 0 | 9 | 98318000 | 60557000 | 1891788759 |
| 355 | 2014 | Sunderland | 0 | 14 | 104383000 | 59549000 | 1891788759 |
| 356 | 2014 | Swansea City | 0 | 12 | 98691997 | 63231880 | 1891788759 |
| 357 | 2014 | Tottenham Hotspur | 0 | 6 | 180541000 | 101608000 | 1891788759 |
| 358 | 2014 | West Bromwich Albion | 0 | 17 | 86769000 | 65470000 | 1891788759 |
| 359 | 2014 | West Ham United | 0 | 13 | 114852000 | 63880000 | 1891788759 |
| 360 | 2015 | Arsenal | 0 | 3 | 344524000 | 192213000 | 2031348184 |
| 361 | 2015 | Aston Villa | 0 | 17 | 115692000 | 83777000 | 2031348184 |
| 362 | 2015 | Burnley | 1 | 19 | 78770000 | 29395000 | 2031348184 |
| 363 | 2015 | Chelsea | 0 | 1 | 319456000 | 217067000 | 2031348184 |
| 364 | 2015 | Crystal Palace | 0 | 10 | 102935970 | 68028216 | 2031348184 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **365** | 2015 | Everton | 0 | 11 | 125572000 | 77515000 | 2031348184 |
| **366** | 2015 | Hull City | 0 | 18 | 146209803 | 55611076 | 2031348184 |
| **367** | 2015 | Leicester City | 1 | 14 | 104437000 | 57438000 | 2031348184 |
| **368** | 2015 | Liverpool | 0 | 6 | 297947000 | 166085000 | 2031348184 |
| **369** | 2015 | Manchester City | 0 | 2 | 351766000 | 193821000 | 2031348184 |
| **370** | 2015 | Manchester United | 0 | 4 | 395178000 | 202421000 | 2031348184 |
| **371** | 2015 | Newcastle United | 0 | 15 | 128830000 | 65087000 | 2031348184 |
| **372** | 2015 | Queens Park Rangers | 1 | 20 | 85875000 | 72914000 | 2031348184 |
| **373** | 2015 | Southampton | 0 | 7 | 113735437 | 80401330 | 2031348184 |
| **374** | 2015 | Stoke City | 0 | 9 | 99626000 | 66580000 | 2031348184 |
| **375** | 2015 | Sunderland | 0 | 16 | 101087000 | 77106000 | 2031348184 |
| **376** | 2015 | Swansea City | 0 | 8 | 103928360 | 82540562 | 2031348184 |
| **377** | 2015 | Tottenham Hotspur | 0 | 5 | 196377000 | 100832000 | 2031348184 |
| **378** | 2015 | West Bromwich Albion | 0 | 13 | 96269000 | 69801000 | 2031348184 |
| **379** | 2015 | West Ham United | 0 | 12 | 120747000 | 72715000 | 2031348184 |

380 rows × 7 columns

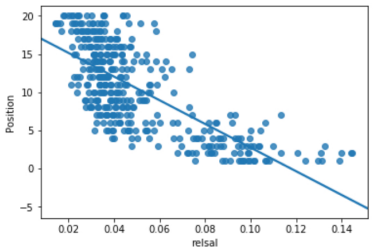
In [8]:
```
# Here we create the variable 'relsal' for the EPL

EPL['relsal']= EPL['salaries']/EPL['allsal']
```

Before running a regression, we use sns.reglot() to look at the relationship between salaries and win percentage on a chart.

In [9]:
```
sns.regplot(x="relsal", y="Position", data = EPL, ci=False)
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbcc5343780>

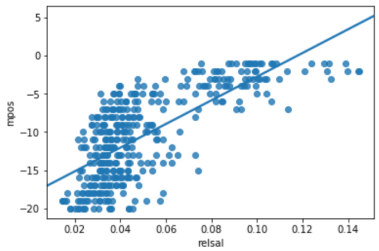



The chart shows that there is a negative relationship between league position and relsal. This is because a lower numerical value of league position means a better performance (e.g. 9 is better than 10 and 1 is better than 2). Higher wage spending relative to other teams generates a higher league position. To avoid confusion, we can reverse the relationship, so that higher spending (on the x axis) leads to a higher position on the y axis. We do this simply by defining 'mpos" as 'position' multiplied by -1. This changes nothing about the underlying logic of the relationship.

In [10]:
```
EPL['mpos'] = -EPL['Position']
```

In [11]:
```
sns.regplot(x="relsal", y="mpos", data = EPL, ci=False)
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbcb31f4f98>

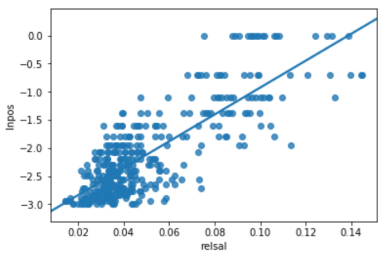



One thing you might notice about the data is that there appears to be a certain amount of curvature, with many dots (each dot represents a single team in a single year) located around the lower values on the x axis, and a smaller number of clubs strung out with high values on the x and y axes. This is a common feature of many types of data. In our regression, we estimate a linear relationship. Hence, it is better if we can first linearize our data, which we can often achieve by taking logarithms. We do that next.

In [12]:
```
EPL['lnpos']= -np.log(EPL['Position'])
```

In [13]:
```
sns.regplot(x="relsal", y="lnpos", data = EPL, ci=False)
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbcb31e39b0>




We now run the simple regression of league position on salaries:

In [14]:
```
possal1_lm = smf.ols(formula = 'lnpos ~ relsal ', data=EPL).fit()
print(possal1_lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  lnpos   R-squared:                       0.657
```

```
Model:                            OLS   Adj. R-squared:                  0.656
Method:                 Least Squares   F-statistic:                     715.6
Date:                Thu, 29 Jul 2021   Prob (F-statistic):           9.02e-89
Time:                        19:27:15   Log-Likelihood:                -243.89
No. Observations:                 375   AIC:                             491.8
Df Residuals:                     373   BIC:                             499.6
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -3.3181      0.051    -64.725      0.000      -3.419      -3.217
relsal        23.9149      0.894     26.750      0.000      22.157      25.673
==============================================================================
Omnibus:                        6.722   Durbin-Watson:                   2.203
Prob(Omnibus):                  0.035   Jarque-Bera (JB):                6.636
Skew:                           0.323   Prob(JB):                       0.0362
Kurtosis:                       3.080   Cond. No.                         37.3
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

As with the NBA data, the relsal variable is statistically significant. However, it is also noticeable that its impact is much larger, in that it accounts for much more of the variation in league position - the R-squared is 0.657, meaning that almost two thirds of the variation can be explained by salaries alone (recall the figure was 17% for the NBA).

Why is that relsal is so much more powerful in terms of explaining the variation in player salaries for the EPL than it was for the NBA? The answer lies in what was discussed at the beginning of this notebook - there are fewer restrictions on the operation of the market, there is much greater inequality between the teams, and this reveals itself in the fact that salaries are a much better explanatory variable for team performance.

We now consider other factors, to see if omitted variable bias might have caused us to under- or over- estimate the impact of player salaries.

The first factor we consider is one that is specific to the promotion and relegation system. During the period in question, three teams were promoted to the EPL in each year (replacing three teams that had been relegated). Do promoted teams start with a disadvantage relative to other teams? We can test for this by adding a dummy variable which is equal to one if the team in question was promoted to the EPL in that season, and otherwise equals zero. We run the regression again with the promotion dummy variable included:

## Self Test

Try running the regression again using mpos instead of lnpos as the y variable. What differences do you see when comparing the two regressions?

In [15]:
```
possal2_lm = smf.ols(formula = 'lnpos ~ relsal + promoted_last_season', data=EPL).fit()
print(possal2_lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  lnpos   R-squared:                       0.657
Model:                            OLS   Adj. R-squared:                  0.656
Method:                 Least Squares   F-statistic:                     356.8
Date:                Thu, 29 Jul 2021   Prob (F-statistic):           3.03e-87
Time:                        19:27:15   Log-Likelihood:                -243.89
No. Observations:                 375   AIC:                             493.8
Df Residuals:                     372   BIC:                             505.6
Df Model:                           2
Covariance Type:            nonrobust
============================================================================================
                               coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
Intercept                -3.3185      0.058    -56.829      0.000      -3.433      -3.204
relsal                   23.9198      0.957     24.988      0.000      22.037      25.802
promoted_last_season      0.0011      0.073      0.014      0.988      -0.142       0.144
==============================================================================
Omnibus:                        6.728   Durbin-Watson:                   2.203
Prob(Omnibus):                  0.035   Jarque-Bera (JB):                6.643
Skew:                           0.324   Prob(JB):                       0.0361
Kurtosis:                       3.080   Cond. No.                         40.4
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

The coefficient on promotion is statistically insignificant. This might come as a surprise - it might seem obvious that promoted teams are at a disadvantage, but there are other factors at play. If a promoted team spends money on players, then they appear to be in same position as everyone else. However, promoted teams may not have as much cash to spend, and hence they do experience a disadvantage, but that is channeled entirely through the effect of relsal. Promoted teams are often smaller than the established teams, but they often enjoy a boost in popularity from fans who are excited by the team's improved status. There can be positive and negative factors associated with promotion, and these can cancel each other out.

Note that the addition of the promotion variable hardly changed the estimated coefficient of relsal.

Given that the promotion effect is statistically insignificant, we now drop it from the regression analysis.

We now consider, as we did with the NBA, the impact of lagged dependent variable- league position in the previous season. As before, we do this by first sorting the df by teams and by season, and then use .shift(1) to create the lag of league position.

In [16]:
```
EPL.sort_values(by=['Club','Season_ending'], ascending=True)
EPL
```

Out[16]:

| | Season_ending | Club | promoted_last_season | Position | Revenues | salaries | allsal | relsal | mpos | lnpos |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1997 | Arsenal | 0 | 3 | 27158007 | 15279000 | 219599462 | 0 | -3 | -1 |
| **1** | 1997 | Aston Villa | 0 | 5 | 22079000 | 10070000 | 219599462 | 0 | -5 | -2 |
| **2** | 1997 | Blackburn Rovers | 0 | 13 | 14302220 | 14336629 | 219599462 | 0 | -13 | -3 |
| **3** | 1997 | Chelsea | 0 | 6 | 23729000 | 14873000 | 219599462 | 0 | -6 | -2 |
| **4** | 1997 | Coventry City | 0 | 17 | 12264825 | 8396261 | 219599462 | 0 | -17 | -3 |
| **5** | 1997 | Derby County | 1 | 12 | 10737571 | 6406557 | 219599462 | 0 | -12 | -2 |
| **6** | 1997 | Everton | 0 | 15 | 18881961 | 10933090 | 219599462 | 0 | -15 | -3 |
| **7** | 1997 | Leeds United | 0 | 11 | 22249000 | 12312000 | 219599462 | 0 | -11 | -2 |
| **8** | 1997 | Leicester City | 1 | 9 | 17320000 | 8914000 | 219599462 | 0 | -9 | -2 |
| **9** | 1997 | Liverpool | 0 | 4 | 39394000 | 14559000 | 219599462 | 0 | -4 | -1 |
| **10** | 1997 | Manchester United | 0 | 1 | 87939000 | 21847000 | 219599462 | 0 | -1 | -0 |

In [17]:
```
pd.set_option('display.max_rows', 400)
EPL
```

Out[17]:

| | Season_ending | Club | promoted_last_season | Position | Revenues | salaries | allsal | relsal | mpos | lnpos |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1997 | Arsenal | 0 | 3 | 27158007 | 15279000 | 219599462 | 0 | -3 | -1 |
| **1** | 1997 | Aston Villa | 0 | 5 | 22079000 | 10070000 | 219599462 | 0 | -5 | -2 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **2** | 1997 | Blackburn Rovers | 0 | 13 | 14302220 | 14336629 | 219599462 | 0 | -13 | -3 |
| **3** | 1997 | Chelsea | 0 | 6 | 23729000 | 14873000 | 219599462 | 0 | -6 | -2 |
| **4** | 1997 | Coventry City | 0 | 17 | 12264825 | 8396261 | 219599462 | 0 | -17 | -3 |
| **5** | 1997 | Derby County | 1 | 12 | 10737571 | 6406557 | 219599462 | 0 | -12 | -2 |
| **6** | 1997 | Everton | 0 | 15 | 18881961 | 10933090 | 219599462 | 0 | -15 | -3 |
| **7** | 1997 | Leeds United | 0 | 11 | 22249000 | 12312000 | 219599462 | 0 | -11 | -2 |
| **8** | 1997 | Leicester City | 1 | 9 | 17320000 | 8914000 | 219599462 | 0 | -9 | -2 |
| **9** | 1997 | Liverpool | 0 | 4 | 39394000 | 14559000 | 219599462 | 0 | -4 | -1 |
| **10** | 1997 | Manchester United | 0 | 1 | 87939000 | 21847000 | 219599462 | 0 | -1 | -0 |

In [18]:
```
EPL['lnpos_lag'] = EPL.groupby('Club')['lnpos'].shift(1)
EPL
```

Out[18]:

| | Season_ending | Club | promoted_last_season | Position | Revenues | salaries | allsal | relsal | mpos | lnpos | lnpos_lag |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1997 | Arsenal | 0 | 3 | 27158007 | 15279000 | 219599462 | 0 | -3 | -1 | nan |
| **1** | 1997 | Aston Villa | 0 | 5 | 22079000 | 10070000 | 219599462 | 0 | -5 | -2 | nan |
| **2** | 1997 | Blackburn Rovers | 0 | 13 | 14302220 | 14336629 | 219599462 | 0 | -13 | -3 | nan |
| **3** | 1997 | Chelsea | 0 | 6 | 23729000 | 14873000 | 219599462 | 0 | -6 | -2 | nan |
| **4** | 1997 | Coventry City | 0 | 17 | 12264825 | 8396261 | 219599462 | 0 | -17 | -3 | nan |
| **5** | 1997 | Derby County | 1 | 12 | 10737571 | 6406557 | 219599462 | 0 | -12 | -2 | nan |
| **6** | 1997 | Everton | 0 | 15 | 18881961 | 10933090 | 219599462 | 0 | -15 | -3 | nan |
| **7** | 1997 | Leeds United | 0 | 11 | 22249000 | 12312000 | 219599462 | 0 | -11 | -2 | nan |
| **8** | 1997 | Leicester City | 1 | 9 | 17320000 | 8914000 | 219599462 | 0 | -9 | -2 | nan |
| **9** | 1997 | Liverpool | 0 | 4 | 39394000 | 14559000 | 219599462 | 0 | -4 | -1 | nan |
| **10** | 1997 | Manchester United | 0 | 1 | 87939000 | 21847000 | 219599462 | 0 | -1 | -0 | nan |

If you scroll through the df you will see that, as with the NBA data, we have missing values (NaN) for the first season (1997), since the values for the previous season are not in the data. But also you will see that there are missing values for some teams in other seasons. These are for clubs which were promoted in that season, and hence they had no lagged value for their EPL position.

This means we will lose some observations when we run the regressions. It's always worse to have fewer observations, but on the other hand it's always better to include potential omitted variables. There is a trade-off here between reducing the size of our dataset and including all relevant variables. Here the problem is not too serious, since we lose 42 observations and still have 333 in our dataset, whereas we expect that omitting the lagged dependent variable would lead to significant bias in the estimate of relsal.

In [19]:
```
possal3_lm = smf.ols(formula = 'lnpos ~lnpos_lag + relsal', data=EPL).fit()
print(possal3_lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  lnpos   R-squared:                       0.708
Model:                            OLS   Adj. R-squared:                  0.706
Method:                 Least Squares   F-statistic:                     400.0
Date:                Thu, 29 Jul 2021   Prob (F-statistic):           6.22e-89
Time:                        19:27:17   Log-Likelihood:                -192.18
No. Observations:                 333   AIC:                             390.4
Df Residuals:                     330   BIC:                             401.8
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -2.0502      0.176    -11.633      0.000      -2.397      -1.703
lnpos_lag      0.3837      0.051      7.480      0.000       0.283       0.485
relsal        14.7409      1.488      9.908      0.000      11.814      17.668
==============================================================================
Omnibus:                        2.795   Durbin-Watson:                   2.125
Prob(Omnibus):                  0.247   Jarque-Bera (JB):                2.972
Skew:                           0.049   Prob(JB):                        0.226
Kurtosis:                       3.452   Cond. No.                         151.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

As expected, the lagged dependent variable has a a large and statistically significant effect on league position. As far as our estimate of relsal is concerned, we can see that our estimate has fallen from 23.9 to 14.7, suggesting that the omission of the lagged dependent variable led to a significant upward bias in our estimate of relsal.

Finally, as we did with the NBA, we consider the possible effects of heterogeneity by adding fixed effects into our regression, recall that we do this simply by adding C(Club) to our regression formula:

In [20]:
```
possal4_lm = smf.ols(formula = "lnpos ~ lnpos_lag + relsal +C(Club)", data=EPL).fit()
print(possal4_lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  lnpos   R-squared:                       0.746
Model:                            OLS   Adj. R-squared:                  0.710
Method:                 Least Squares   F-statistic:                     20.86
Date:                Thu, 29 Jul 2021   Prob (F-statistic):           8.34e-65
Time:                        19:27:18   Log-Likelihood:                -168.86
No. Observations:                 333   AIC:                             421.7
Df Residuals:                     291   BIC:                             581.7
Df Model:                          41
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                       -1.5948      0.255     -6.242      0.000      -2.098      -1.092
C(Club)[T.Aston Villa]          -0.5967      0.171     -3.488      0.001      -0.933      -0.260
C(Club)[T.Barnsley]          -6.508e-15   1.45e-15     -4.474      0.000   -9.37e-15   -3.65e-15
C(Club)[T.Birmingham City]      -0.6998      0.241     -2.908      0.004      -1.173      -0.226
C(Club)[T.Blackburn Rovers]     -0.6537      0.188     -3.485      0.001      -1.023      -0.285
C(Club)[T.Blackpool]         -2.602e-15   8.02e-16     -3.244      0.001   -4.18e-15   -1.02e-15
C(Club)[T.Bolton Wanderers]     -0.6032      0.204     -2.951      0.003      -1.005      -0.201
C(Club)[T.Bradford City]        -0.9260      0.466     -1.988      0.048      -1.843      -0.009
C(Club)[T.Burnley]              -0.7863      0.470     -1.672      0.096      -1.712       0.139
C(Club)[T.Cardiff City]       6.56e-15   2.13e-15      3.073      0.002    2.36e-15    1.08e-14
C(Club)[T.Charlton Athletic]    -0.6193      0.225     -2.750      0.006      -1.062      -0.176
C(Club)[T.Chelsea]              -0.3218      0.162     -1.990      0.048      -0.640      -0.003
C(Club)[T.Coventry City]        -0.7954      0.267     -2.980      0.003      -1.321      -0.270
C(Club)[T.Crystal Palace]       -0.5410      0.301     -1.797      0.073      -1.133       0.051
C(Club)[T.Derby County]         -0.7780      0.236     -3.301      0.001      -1.242      -0.314
C(Club)[T.Everton]              -0.4743      0.175     -2.713      0.007      -0.818      -0.130
C(Club)[T.Fulham]               -0.6840      0.195     -3.510      0.001      -1.067      -0.300
C(Club)[T.Hull City]            -0.8727      0.301     -2.901      0.004      -1.465      -0.281
```

```
C(Club)[T.Ipswich Town]              -1.2856      0.455     -2.827      0.005      -2.181      -0.391
C(Club)[T.Leeds United]              -0.4783      0.199     -2.402      0.017      -0.870      -0.086
C(Club)[T.Leicester City]            -0.6517      0.233     -2.799      0.005      -1.110      -0.194
C(Club)[T.Liverpool]                 -0.3678      0.146     -2.523      0.012      -0.655      -0.081
C(Club)[T.Manchester City]           -0.4192      0.163     -2.567      0.011      -0.741      -0.098
C(Club)[T.Manchester United]          0.0882      0.148      0.596      0.552      -0.203       0.380
C(Club)[T.Middlesbrough]             -0.6861      0.198     -3.464      0.001      -1.076      -0.296
C(Club)[T.Newcastle United]          -0.6826      0.168     -4.064      0.000      -1.013      -0.352
C(Club)[T.Norwich City]              -0.6359      0.300     -2.120      0.035      -1.226      -0.045
C(Club)[T.Nottingham Forest]         -0.9847      0.462     -2.129      0.034      -1.895      -0.075
C(Club)[T.Portsmouth]                -0.6271      0.265     -2.366      0.019      -1.149      -0.105
C(Club)[T.Queens Park Rangers]       -1.1099      0.343     -3.238      0.001      -1.785      -0.435
C(Club)[T.Reading]                   -0.9875      0.347     -2.844      0.005      -1.671      -0.304
C(Club)[T.Sheffield United]        6.731e-17   2.53e-16      0.266      0.791   -4.31e-16    5.66e-16
C(Club)[T.Sheffield Wednesday]       -0.9284      0.292     -3.184      0.002      -1.502      -0.354
C(Club)[T.Southampton]               -0.5853      0.205     -2.859      0.005      -0.988      -0.182
C(Club)[T.Stoke City]                -0.5699      0.236     -2.412      0.017      -1.035      -0.105
C(Club)[T.Sunderland]                -0.7656      0.195     -3.922      0.000      -1.150      -0.381
C(Club)[T.Swansea City]              -0.4404      0.293     -1.505      0.133      -1.016       0.135
C(Club)[T.Tottenham Hotspur]         -0.4626      0.165     -2.808      0.005      -0.787      -0.138
C(Club)[T.Watford]                   -0.8499      0.469     -1.813      0.071      -1.772       0.073
C(Club)[T.West Bromwich Albion]      -0.6778      0.225     -3.014      0.003      -1.120      -0.235
C(Club)[T.West Ham United]           -0.6668      0.183     -3.652      0.000      -1.026      -0.307
C(Club)[T.Wigan Athletic]            -0.7748      0.233     -3.332      0.001      -1.232      -0.317
C(Club)[T.Wimbledon]                 -0.9101      0.296     -3.080      0.002      -1.492      -0.328
C(Club)[T.Wolverhampton Wanderers]   -0.7955      0.305     -2.610      0.010      -1.395      -0.196
lnpos_lag                             0.2498      0.058      4.301      0.000       0.136       0.364
relsal                               10.9757      2.477      4.431      0.000       6.101      15.851
==============================================================================
Omnibus:                        3.243   Durbin-Watson:                   2.199
Prob(Omnibus):                  0.198   Jarque-Bera (JB):                3.685
Skew:                           0.034   Prob(JB):                        0.158
Kurtosis:                       3.511   Cond. No.                     2.99e+17
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.15e-32. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Remember that our main interest is in the effect of relsal. Adding the fixed effects has reduced the value of the coefficient a little further - to 10.98 - again suggesting that our original specification suffered from omitted variable bias, which biased our estimate of relsal upwards.

Before asking what the value of the coefficient of relsal means for league position, we should stop to consider the fixed effects. As can be seen from the regression output, almost all are negative and statistically significant. The reason for this is that when estimating the fixed effects there must always be a reference group- so that the fixed effect measures performance relative to the reference group. By default Python uses the first name on the list as the reference group, and since our clubs are listed alphabetically, the reference group is Arsenal. Now, over the period 1997-2015 Arsenal was one of the most consistently successful teams, which explains why most of the coefficients are negative. Most teams were performing worse than Arsenal, even after taking account of wage spending via relsal.

In this case, it might make more sense to evaluate the fixed effects relative to a mid-table team. We can choose the reference group, but first let's list the average league performance of the teams, to see which club would be a good candidate for the reference group. We use .groupby() to calculate average leagues position by club:

In [21]:
```
Avpos = EPL.groupby(['Club'])['Position'].mean()
Avpos
```

Out[21]:
```
Club
Arsenal                    3
Aston Villa               10
Barnsley                  19
Birmingham City           14
Blackburn Rovers          12
Blackpool                 19
Bolton Wanderers          13
Bradford City             18
Burnley                   18
Cardiff City              20
Charlton Athletic         13
Chelsea                    3
Coventry City             15
Crystal Palace            15
Derby County              14
Everton                   10
Fulham                    12
Hull City                 18
Ipswich Town              12
Leeds United               8
Leicester City            13
Liverpool                  4
Manchester City            8
Manchester United          2
Middlesbrough             13
Newcastle United          10
Norwich City              15
Nottingham Forest         20
Portsmouth                14
Queens Park Rangers       19
Reading                   15
Sheffield United          18
Sheffield Wednesday       14
Southampton               12
Stoke City                12
Sunderland                14
Swansea City              10
Tottenham Hotspur          8
Watford                   20
West Bromwich Albion      15
West Ham United           12
Wigan Athletic            15
Wimbledon                 14
Wolverhampton Wanderers   18
Name: Position, dtype: float64
```

"Mid-table" means an average league position of 10 or 11. There are a few we could choose from, but one of the most consistent over the period was Everton, so we use them.

To specify the reference group we expand the C() statement to define the "treatment" group reference. Hence we write C(Club, Treatment('Everton')):

In [22]:
```
possal5_lm = smf.ols(formula = "lnpos ~ lnpos_lag + relsal +C(Club, Treatment('Everton'))", data=EPL).fit()
print(possal5_lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  lnpos   R-squared:                       0.746
Model:                            OLS   Adj. R-squared:                  0.710
Method:                 Least Squares   F-statistic:                     20.86
Date:                Thu, 29 Jul 2021   Prob (F-statistic):           8.34e-65
Time:                        19:27:21   Log-Likelihood:                -168.86
```

```
No. Observations:                333   AIC:                             421.7
Df Residuals:                    291   BIC:                             581.7
Df Model:                         41
Covariance Type:            nonrobust
================================================================================================================
                                                   coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------------------------
Intercept                                       -2.0691      0.218     -9.481      0.000      -2.499      -1.640
C(Club, Treatment('Everton'))[T.Arsenal]         0.4743      0.175      2.713      0.007       0.130       0.818
C(Club, Treatment('Everton'))[T.Aston Villa]    -0.1225      0.144     -0.853      0.394      -0.405       0.160
C(Club, Treatment('Everton'))[T.Barnsley]     4.901e-15   2.77e-15      1.767      0.078   -5.58e-16    1.04e-14
C(Club, Treatment('Everton'))[T.Birmingham City] -0.2255      0.205     -1.101      0.272      -0.628       0.178
C(Club, Treatment('Everton'))[T.Blackburn Rovers] -0.1794      0.157     -1.144      0.254      -0.488       0.129
C(Club, Treatment('Everton'))[T.Blackpool]    7.78e-15   2.09e-15      3.722      0.000    3.67e-15    1.19e-14
C(Club, Treatment('Everton'))[T.Bolton Wanderers] -0.1289      0.166     -0.777      0.438      -0.455       0.198
C(Club, Treatment('Everton'))[T.Bradford City] -0.4517      0.444     -1.017      0.310      -1.326       0.423
C(Club, Treatment('Everton'))[T.Burnley]        -0.3121      0.446     -0.700      0.485      -1.190       0.566
C(Club, Treatment('Everton'))[T.Cardiff City] -2.514e-16   9.99e-16     -0.252      0.802   -2.22e-15    1.71e-15
C(Club, Treatment('Everton'))[T.Charlton Athletic] -0.1450      0.192     -0.754      0.452      -0.524       0.234
C(Club, Treatment('Everton'))[T.Chelsea]         0.1525      0.215      0.709      0.479      -0.271       0.576
C(Club, Treatment('Everton'))[T.Coventry City]  -0.3211      0.239     -1.344      0.180      -0.791       0.149
C(Club, Treatment('Everton'))[T.Crystal Palace] -0.0667      0.271     -0.247      0.805      -0.599       0.466
C(Club, Treatment('Everton'))[T.Derby County]   -0.3037      0.204     -1.490      0.137      -0.705       0.097
C(Club, Treatment('Everton'))[T.Fulham]         -0.2097      0.161     -1.304      0.193      -0.526       0.107
C(Club, Treatment('Everton'))[T.Hull City]      -0.3984      0.271     -1.471      0.142      -0.931       0.135
C(Club, Treatment('Everton'))[T.Ipswich Town]   -0.8113      0.443     -1.830      0.068      -1.684       0.061
C(Club, Treatment('Everton'))[T.Leeds United]   -0.0041      0.197     -0.021      0.984      -0.393       0.384
C(Club, Treatment('Everton'))[T.Leicester City] -0.1774      0.203     -0.873      0.383      -0.577       0.223
C(Club, Treatment('Everton'))[T.Liverpool]       0.1065      0.170      0.627      0.531      -0.228       0.441
C(Club, Treatment('Everton'))[T.Manchester City] 0.0550      0.176      0.312      0.755      -0.292       0.402
C(Club, Treatment('Everton'))[T.Manchester United] 0.5625      0.198      2.835      0.005       0.172       0.953
C(Club, Treatment('Everton'))[T.Middlesbrough]  -0.2118      0.165     -1.282      0.201      -0.537       0.113
C(Club, Treatment('Everton'))[T.Newcastle United] -0.2083      0.147     -1.417      0.157      -0.497       0.081
C(Club, Treatment('Everton'))[T.Norwich City]   -0.1617      0.270     -0.598      0.550      -0.694       0.370
C(Club, Treatment('Everton'))[T.Nottingham Forest] -0.5104      0.444     -1.150      0.251      -1.384       0.363
C(Club, Treatment('Everton'))[T.Portsmouth]     -0.1528      0.239     -0.641      0.522      -0.622       0.317
C(Club, Treatment('Everton'))[T.Queens Park Rangers] -0.6357      0.323     -1.968      0.050      -1.271       0.000
C(Club, Treatment('Everton'))[T.Reading]        -0.5132      0.322     -1.593      0.112      -1.147       0.121
C(Club, Treatment('Everton'))[T.Sheffield United] 4.708e-17   1.05e-16      0.447      0.656    -1.6e-16    2.55e-16
C(Club, Treatment('Everton'))[T.Sheffield Wednesday] -0.4541      0.268     -1.692      0.092      -0.982       0.074
C(Club, Treatment('Everton'))[T.Southampton]    -0.1110      0.166     -0.669      0.504      -0.438       0.216
C(Club, Treatment('Everton'))[T.Stoke City]     -0.0956      0.204     -0.469      0.639      -0.497       0.305
C(Club, Treatment('Everton'))[T.Sunderland]     -0.2913      0.158     -1.842      0.066      -0.603       0.020
C(Club, Treatment('Everton'))[T.Swansea City]    0.0339      0.269      0.126      0.900      -0.495       0.563
C(Club, Treatment('Everton'))[T.Tottenham Hotspur] 0.0117      0.145      0.081      0.936      -0.273       0.297
C(Club, Treatment('Everton'))[T.Watford]        -0.3756      0.445     -0.843      0.400      -1.252       0.501
C(Club, Treatment('Everton'))[T.West Bromwich Albion] -0.2035      0.185     -1.098      0.273      -0.568       0.161
C(Club, Treatment('Everton'))[T.West Ham United] -0.1925      0.151     -1.275      0.203      -0.490       0.105
C(Club, Treatment('Everton'))[T.Wigan Athletic] -0.3005      0.194     -1.548      0.123      -0.683       0.081
C(Club, Treatment('Everton'))[T.Wimbledon]      -0.4358      0.269     -1.619      0.106      -0.966       0.094
C(Club, Treatment('Everton'))[T.Wolverhampton Wanderers] -0.3212      0.272     -1.182      0.238      -0.856       0.214
lnpos_lag                                        0.2498      0.058      4.301      0.000       0.136       0.364
relsal                                          10.9757      2.477      4.431      0.000       6.101      15.851
==============================================================================
Omnibus:                        3.243   Durbin-Watson:                   2.199
Prob(Omnibus):                  0.198   Jarque-Bera (JB):                3.685
Skew:                           0.034   Prob(JB):                        0.158
Kurtosis:                       3.511   Cond. No.                     1.41e+17
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 9.72e-32. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

We now see that only four clubs have statistically significant coefficients. Two of these are Manchester United and Arsenal, the two dominant clubs over the period. This implies that these clubs, which spent more money than the others on players, still managed to extract better than average performance from these players. This fact is likely related to the two iconic managers of these clubs, Sir Alex Ferguson and Arsene Wenger.

Notice that changing the reference group does not change the coefficient on relsal or on the lagged dependent variable. The R-squared of the regression, or any other diagnostic statistic. The only thing that changes are the coefficients of the fixed effects themselves, and also the coefficient of the constant.

## Self Test

Calculate the fixed effects using Sunderland as the reference team. What changes do you see in the estimates?

Finally, we consider how changes in relsal affect league positions, given our estimated coefficient of just under 11. Ignoring the fixed effects and the lagged dependent variable, minus the log of league position can be expressed as a function of the constant plus the relsal coefficient times the value of relsal, i.e. -lnpos = -2.1 + 11 relsal. Because we have expressed league position as a logarithm, the impact on league position will differ for different values of relsal. From the charts above we can see that relsal varies roughly between 0.02 (2%) and 0.14 (14%).

Let's consider three values of relsal: .02, .07 and .14. What league positions are implied by these values? To convert -lnpos back into position we have to multiply by -1 and then take the exponent. To take an exponent using numpy you just type np.exp() with the expression in parentheses. If we do that to the right hand side of the equation then we have our answer.

In [23]:
```
print(np.exp(2.1- 11*.02))
print(np.exp(2.1- 11*.08))
print(np.exp(2.1- 11*.14))
```

```
6.553504862191149
3.387187733621335
1.7506725002961012
```

It is not surprising to see that the highest spending level implies a very high league position - somewhere between first (1) and second (2). It is more suprising to see that a level of spending somewhere around the mean (0.08) implies a position between 3rd and 4th, while even lowest spending (.02) implies a league position between 6th and 7th. The explanation for this lies with the role of the lagged dependent variable. This tends to emphasize the role of past performance in contributing to current performance. Teams that are able to spend consistently can more easily achieve a high league position than teams which attempt to do so by a short term infusion of spending.

## Self Test

Calculate the expected position of (a) Arsenal and (b) West Ham United, using the same relsal values as above (i.e. when relsal is .02, .08 and .14) but now including the fixed effects for the two clubs.

# Conclusion

While we have repeated the analysis that we conducted for the NBA almost exactly, our results have been quite different, reflecting the different organizational structure of the soccer in England (and in other soccer leagues outside North America). The main result of our analysis is that salary spending varies much more than it does in the NBA, and has a much larger impact on outcomes, even after we allow for possible omitted variables and heterogeneity. Next week we will look at Major League Baseball (MLB).

In [ ]: