EPL

24 July 2020

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.

```
# As usual, we begin by loading the packages we will need

library("readxl",quietly = TRUE)
library("tidyverse",quietly = TRUE)
```

```
# Now we load the data

EPL= read_excel("EPL pay and performance.xlsx")
```

We use summary() 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 str() to summarize the dataframe.

```
EPL %>% summary()
```

```
##  Season_ending       Club            promoted_last_season    Position
##  Min.   :1997   Length:380         Min.   :0.00          Min.   : 1.00
##  1st Qu.:2001   Class :character   1st Qu.:0.00          1st Qu.: 5.75
```

```
##  Median :2006   Mode  :character   Median :0.00         Median :10.50
##  Mean   :2006                       Mean   :0.15         Mean   :10.50
##  3rd Qu.:2011                       3rd Qu.:0.00         3rd Qu.:15.25
##  Max.   :2015                       Max.   :1.00         Max.   :20.00
##
##     Revenues            salaries
##  Min.   :  9238238   Min.   :  4172024
##  1st Qu.: 38958342   1st Qu.: 24139000
##  Median : 59072000   Median : 37744000
##  Mean   : 84310247   Mean   : 51836370
##  3rd Qu.: 97530500   3rd Qu.: 63000068
##  Max.   :433164000   Max.   :233106000
##  NA's   :5           NA's   :5
```

```
EPL %>% str()
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    380 obs. of  6 variables:
##  $ Season_ending       : num  1997 1997 1997 1997 1997 ...
##  $ Club                : chr  "Arsenal" "Aston Villa" "Blackburn Rovers" "Chelsea" ..
##  $ promoted_last_season: num  0 0 0 0 0 1 0 0 1 0 ...
##  $ Position            : num  3 5 13 6 17 12 15 11 9 4 ...
##  $ Revenues            : num  27158007 22079000 14302220 23729000 12264825 ...
##  $ salaries            : num  15279000 10070000 14336629 14873000 8396261 ...
```

We use group_by() to sum salaries.

```
Sumsal <- EPL %>%
  group_by(Season_ending)%>%
   dplyr::summarise(salaries = sum(salaries))%>%rename(allsal = salaries)
Sumsal
```

```
## # A tibble: 19 x 2
##    Season_ending     allsal
##            <dbl>      <dbl>
##  1          1997  219599462
##  2          1998         NA
##  3          1999  390018517
##  4          2000         NA
##  5          2001  562286010
##  6          2002         NA
##  7          2003  747738215
##  8          2004  798029773
##  9          2005  783688898
## 10          2006  867186039
## 11          2007  950696528
## 12          2008 1188491236
## 13          2009         NA
```

```
## 14          2010         NA
## 15          2011 1583955432
## 16          2012 1626852832
## 17          2013 1782493515
## 18          2014 1891788759
## 19          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.

```
EPL <- left_join(EPL, Sumsal, by="Season_ending")
head(EPL)
```

```
## # A tibble: 6 x 7
##   Season_ending Club    promoted_last_se~ Position Revenues salaries allsal
##           <dbl> <chr>               <dbl>    <dbl>    <dbl>    <dbl>  <dbl>
## 1          1997 Arsenal                 0        3 27158007 15279000 2.20e8
## 2          1997 Aston ~                 0        5 22079000 10070000 2.20e8
## 3          1997 Blackb~                 0       13 14302220 14336629 2.20e8
## 4          1997 Chelsea                 0        6 23729000 14873000 2.20e8
## 5          1997 Covent~                 0       17 12264825  8396261 2.20e8
## 6          1997 Derby ~                 1       12 10737571  6406557 2.20e8
```

```
tail(EPL)
```

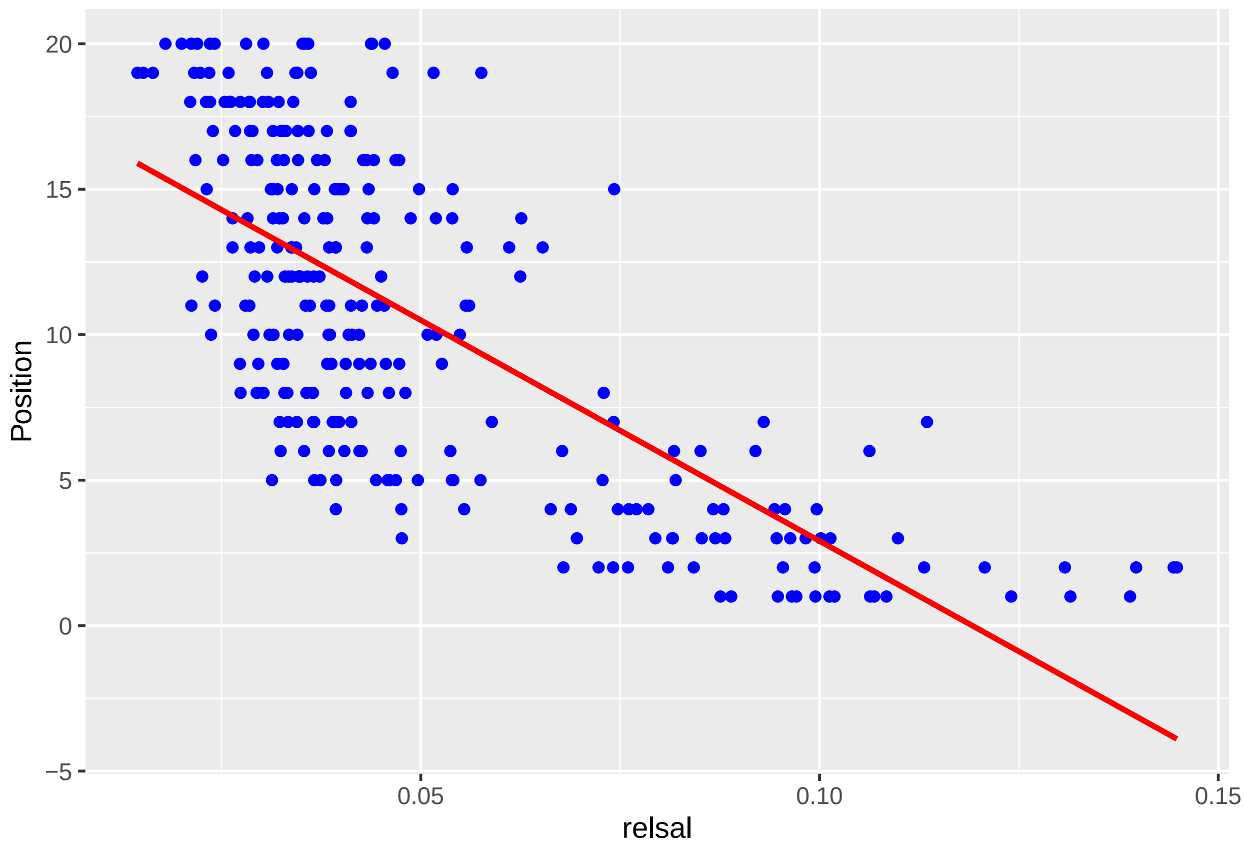
```
## # A tibble: 6 x 7
##   Season_ending Club    promoted_last_se~ Position Revenues salaries allsal
##           <dbl> <chr>               <dbl>    <dbl>    <dbl>    <dbl>  <dbl>
## 1          2015 Stoke ~                 0        9   9.96e7   6.66e7 2.03e9
## 2          2015 Sunder~                 0       16   1.01e8   7.71e7 2.03e9
## 3          2015 Swanse~                 0        8   1.04e8   8.25e7 2.03e9
## 4          2015 Totten~                 0        5   1.96e8   1.01e8 2.03e9
## 5          2015 West B~                 0       13   9.63e7   6.98e7 2.03e9
## 6          2015 West H~                 0       12   1.21e8   7.27e7 2.03e9
```

```
# Here we create the variable 'relsal' for the EPL

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

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

```
# Having prepared the data, we are now ready to examine it. First,
# we generate and xy plot use the ggplot2  package.
# This illustrates nicely the close correlation between win percentage
```

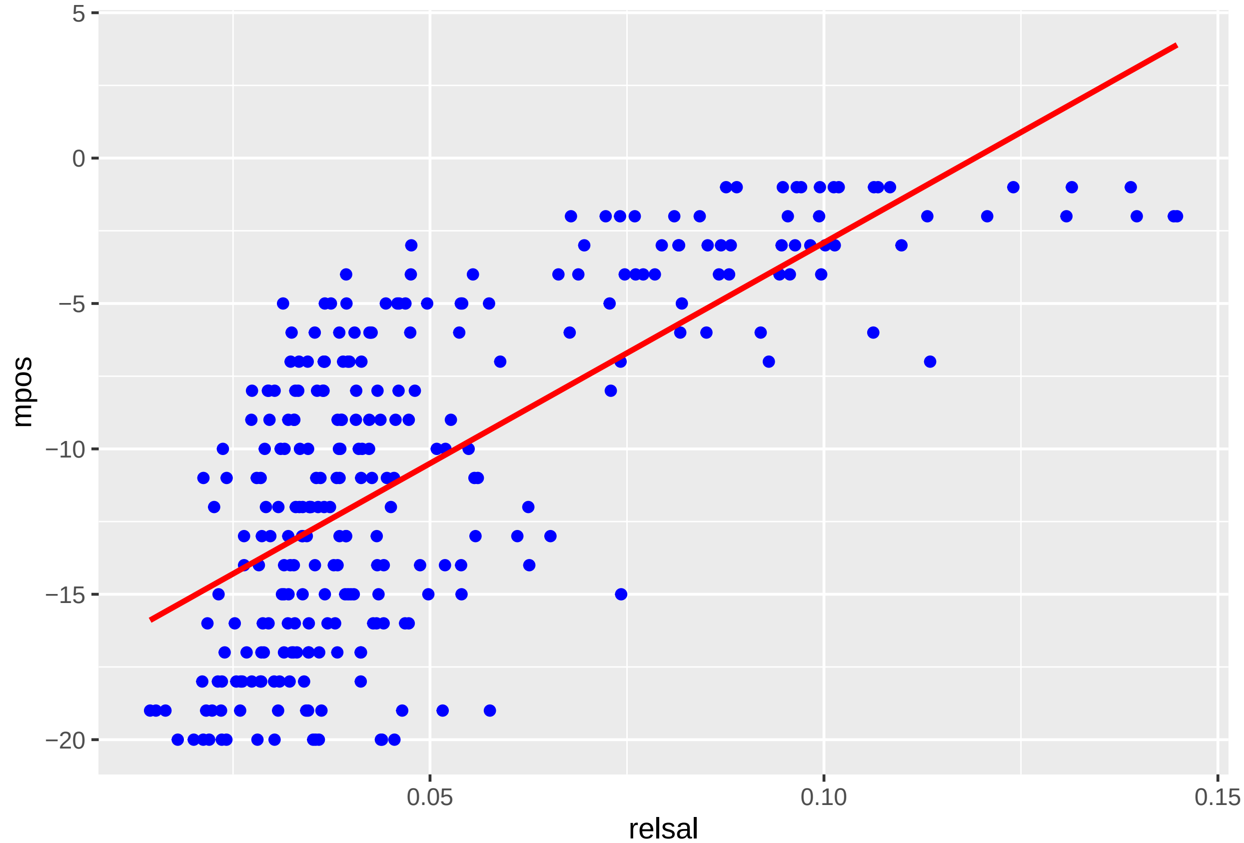
```r
# and the Pythagorean Expectation.

ggplot(data = EPL,aes(x = relsal,y = Position )) + geom_point(color='blue')+
  geom_smooth(method = "lm", se = FALSE,color = "red")
```



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.

```r
EPL[,'mpos'] = -EPL[,'Position']
```

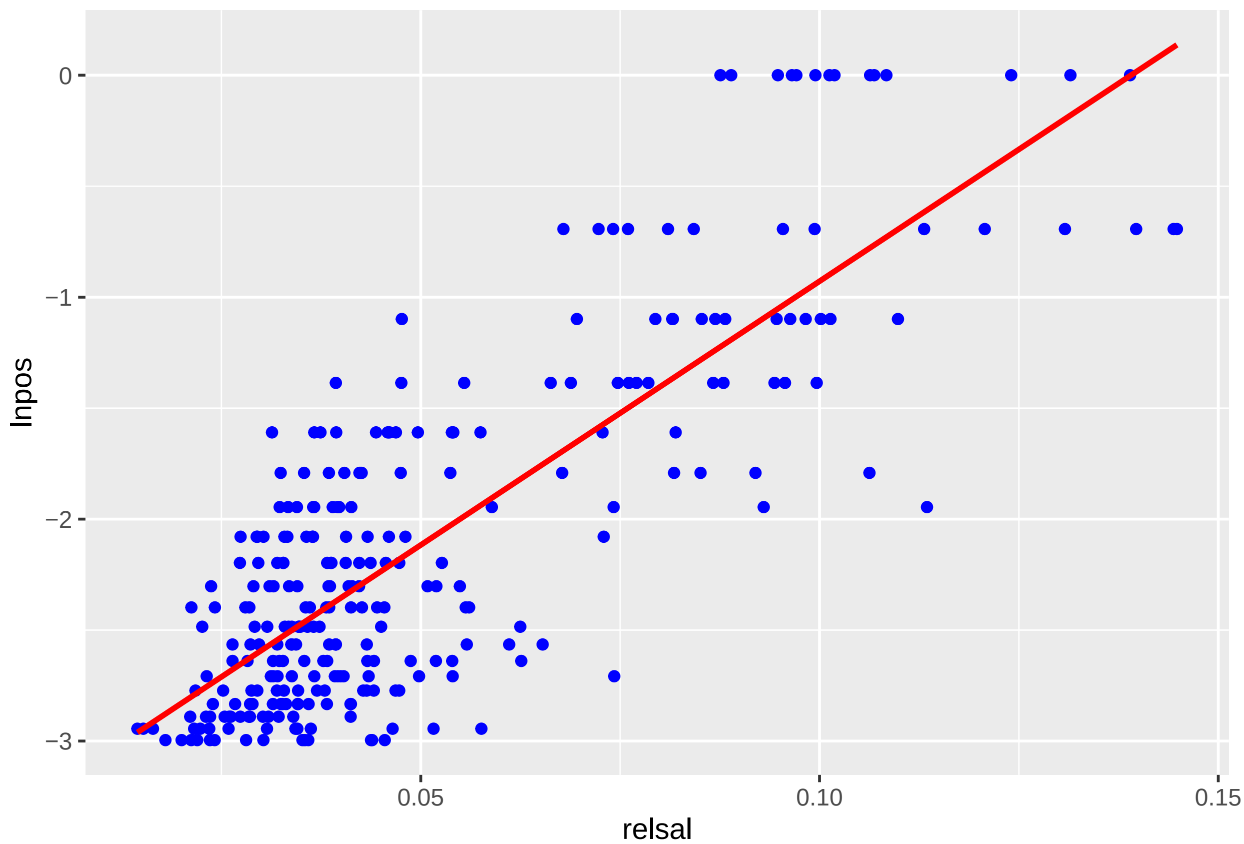
```r
ggplot(data = EPL,aes(x = relsal,y = mpos )) + geom_point(color='blue')+
  geom_smooth(method = "lm", se = FALSE,color = "red")
```



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.

```
EPL[,'lnpos']= -log(EPL[,'Position'])
```

```
ggplot(data = EPL,aes(x = relsal,y = lnpos )) + geom_point(color='blue')+
  geom_smooth(method = "lm", se = FALSE, color = "red")
```



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

```
possal1_lm = lm(formula = 'lnpos ~ relsal', data = EPL)
possal1_lm %>% summary()
```

```
##
## Call:
## lm(formula = "lnpos ~ relsal", data = EPL)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.33783 -0.30352 -0.05557  0.32939  1.22368
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.30514    0.05791  -57.07   <2e-16 ***
## relsal      23.76725    1.01813   23.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4621 on 278 degrees of freedom
##   (100 observations deleted due to missingness)
```

```
## Multiple R-squared:  0.6622, Adjusted R-squared:  0.661
## F-statistic: 544.9 on 1 and 278 DF,  p-value: < 2.2e-16
```

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?

```
possal2_lm = lm(formula = 'lnpos ~ relsal + promoted_last_season', data = EPL)
possal2_lm %>% summary()
```

```
##
## Call:
## lm(formula = "lnpos ~ relsal + promoted_last_season", data = EPL)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3374 -0.3034 -0.0563  0.3297  1.2238
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -3.303877   0.065947  -50.10   <2e-16 ***
## relsal               23.751932   1.088135   21.83   <2e-16 ***
## promoted_last_season -0.003339   0.082649   -0.04    0.968
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.4629 on 277 degrees of freedom
##   (100 observations deleted due to missingness)
## Multiple R-squared:  0.6622, Adjusted R-squared:  0.6598
## F-statistic: 271.5 on 2 and 277 DF,  p-value: < 2.2e-16
```

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.

```r
EPL <- EPL %>% arrange(Club,Season_ending)
head(EPL)
```

```
## # A tibble: 6 x 10
##   Season_ending Club  promoted_last_s~ Position Revenues salaries  allsal
##           <dbl> <chr>            <dbl>    <dbl>    <dbl>    <dbl>   <dbl>
## 1          1997 Arse~                0        3 27158007 15279000  2.20e8
## 2          1998 Arse~                0        1 40391000 21882000 NA
## 3          1999 Arse~                0        2 48623000 26478000  3.90e8
## 4          2000 Arse~                0        2 61260000 33970000 NA
## 5          2001 Arse~                0        2 62911000 40651000  5.62e8
## 6          2002 Arse~                0        1 90967000 61453000 NA
## # ... with 3 more variables: relsal <dbl>, mpos <dbl>, lnpos <dbl>
```

```r
tail(EPL)
```

```
## # A tibble: 6 x 10
##   Season_ending Club  promoted_last_s~ Position Revenues salaries  allsal
##           <dbl> <chr>            <dbl>    <dbl>    <dbl>    <dbl>   <dbl>
## 1          1999 Wimb~                0       16 14733185 11508510  3.90e8
## 2          2000 Wimb~                0       18 14552747 15770522 NA
## 3          2004 Wolv~                1       20 37980318 19278845  7.98e8
## 4          2010 Wolv~                1       15 60643790 29800808 NA
```

```
## 5          2011 Wolv~                    0        17 64401000 37915000  1.58e9
## 6          2012 Wolv~                    0        20 60646000 38339000  1.63e9
## # ... with 3 more variables: relsal <dbl>, mpos <dbl>, lnpos <dbl>
```

```
EPL <- EPL %>%
       group_by(Club)%>%
       mutate(lnpos_lag = dplyr::lag(lnpos))%>%
       ungroup()
head(EPL)
```

```
## # A tibble: 6 x 11
##   Season_ending Club  promoted_last_s~ Position Revenues salaries  allsal
##           <dbl> <chr>            <dbl>    <dbl>    <dbl>    <dbl>   <dbl>
## 1          1997 Arse~                0        3 27158007 15279000  2.20e8
## 2          1998 Arse~                0        1 40391000 21882000 NA
## 3          1999 Arse~                0        2 48623000 26478000  3.90e8
## 4          2000 Arse~                0        2 61260000 33970000 NA
## 5          2001 Arse~                0        2 62911000 40651000  5.62e8
## 6          2002 Arse~                0        1 90967000 61453000 NA
## # ... with 4 more variables: relsal <dbl>, mpos <dbl>, lnpos <dbl>,
## #   lnpos_lag <dbl>
```

```
tail(EPL)
```

```
## # A tibble: 6 x 11
##   Season_ending Club  promoted_last_s~ Position Revenues salaries  allsal
##           <dbl> <chr>            <dbl>    <dbl>    <dbl>    <dbl>   <dbl>
## 1          1999 Wimb~                0       16 14733185 11508510  3.90e8
## 2          2000 Wimb~                0       18 14552747 15770522 NA
## 3          2004 Wolv~                1       20 37980318 19278845  7.98e8
## 4          2010 Wolv~                1       15 60643790 29800808 NA
## 5          2011 Wolv~                0       17 64401000 37915000  1.58e9
## 6          2012 Wolv~                0       20 60646000 38339000  1.63e9
## # ... with 4 more variables: relsal <dbl>, mpos <dbl>, lnpos <dbl>,
## #   lnpos_lag <dbl>
```

If you scroll through the df you will see that, as with the NBA data, we have missing values (NA) 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.

```
possal3_lm = lm(formula = 'lnpos ~lnpos_lag + relsal', data = EPL)
possal3_lm %>% summary()
```

```
##
## Call:
## lm(formula = "lnpos ~lnpos_lag + relsal", data = EPL)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.56527 -0.27988 -0.03658  0.28612  1.17343
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.01219    0.20693  -9.724  < 2e-16 ***
## lnpos_lag    0.39296    0.06077   6.466 5.51e-10 ***
## relsal      14.37764    1.72710   8.325 6.26e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4254 on 242 degrees of freedom
##   (135 observations deleted due to missingness)
## Multiple R-squared:  0.7196, Adjusted R-squared:  0.7173
## F-statistic: 310.5 on 2 and 242 DF,  p-value: < 2.2e-16
```

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 with "lm" package:

```
possal4_lm <- lm(lnpos ~ lnpos_lag + relsal +
                   factor(Club),data = EPL)

possal4_lm %>% summary()
```

```
##
## Call:
## lm(formula = lnpos ~ lnpos_lag + relsal + factor(Club), data = EPL)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7302 -0.2517  0.0000  0.2841  0.9946
##
```

```
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                           -1.6927     0.2995  -5.651 5.31e-08
## lnpos_lag                              0.2819     0.0686   4.109 5.76e-05
## relsal                                11.0162     2.8482   3.868 0.000148
## factor(Club)Aston Villa               -0.5086     0.2045  -2.487 0.013680
## factor(Club)Birmingham City           -0.6294     0.2701  -2.331 0.020746
## factor(Club)Blackburn Rovers          -0.5323     0.2224  -2.393 0.017602
## factor(Club)Bolton Wanderers          -0.3853     0.2368  -1.627 0.105291
## factor(Club)Bradford City             -0.7382     0.4791  -1.541 0.124883
## factor(Club)Burnley                   -0.5964     0.4845  -1.231 0.219746
## factor(Club)Charlton Athletic         -0.4071     0.2525  -1.612 0.108422
## factor(Club)Chelsea                   -0.1476     0.1899  -0.777 0.438001
## factor(Club)Coventry City             -0.7989     0.3554  -2.248 0.025669
## factor(Club)Crystal Palace            -0.3558     0.3176  -1.120 0.263948
## factor(Club)Derby County              -0.5414     0.3125  -1.733 0.084638
## factor(Club)Everton                   -0.2310     0.2053  -1.125 0.261858
## factor(Club)Fulham                    -0.5601     0.2214  -2.529 0.012186
## factor(Club)Hull City                 -0.6423     0.3634  -1.767 0.078653
## factor(Club)Leeds United              -0.5942     0.2559  -2.322 0.021229
## factor(Club)Leicester City            -0.5652     0.2822  -2.003 0.046502
## factor(Club)Liverpool                 -0.2836     0.1728  -1.641 0.102375
## factor(Club)Manchester City           -0.2169     0.1842  -1.177 0.240387
## factor(Club)Manchester United          0.2269     0.1740   1.304 0.193798
## factor(Club)Middlesbrough             -0.4562     0.2308  -1.977 0.049438
## factor(Club)Newcastle United          -0.4690     0.1976  -2.373 0.018575
## factor(Club)Norwich City              -0.4555     0.3164  -1.440 0.151404
## factor(Club)Nottingham Forest         -0.7921     0.4749  -1.668 0.096897
## factor(Club)Portsmouth                -0.4477     0.2811  -1.593 0.112785
## factor(Club)Queens Park Rangers       -0.9203     0.3567  -2.580 0.010582
## factor(Club)Reading                   -0.8111     0.3618  -2.242 0.026053
## factor(Club)Sheffield Wednesday       -0.3931     0.4704  -0.836 0.404333
## factor(Club)Southampton               -0.3982     0.2372  -1.678 0.094842
## factor(Club)Stoke City                -0.3978     0.2665  -1.493 0.137054
## factor(Club)Sunderland                -0.5944     0.2321  -2.561 0.011152
## factor(Club)Swansea City              -0.2682     0.3074  -0.873 0.383843
## factor(Club)Tottenham Hotspur         -0.2782     0.1934  -1.438 0.151847
## factor(Club)Watford                   -0.6567     0.4826  -1.361 0.175089
## factor(Club)West Bromwich Albion      -0.4602     0.2504  -1.838 0.067513
## factor(Club)West Ham United           -0.5322     0.2197  -2.422 0.016318
## factor(Club)Wigan Athletic            -0.6185     0.2745  -2.253 0.025293
## factor(Club)Wimbledon                 -0.6416     0.4727  -1.357 0.176181
## factor(Club)Wolverhampton Wanderers   -0.7025     0.3661  -1.919 0.056395
##
## (Intercept)                          ***
```

```
## lnpos_lag                           ***
## relsal                              ***
## factor(Club)Aston Villa             *
## factor(Club)Birmingham City         *
## factor(Club)Blackburn Rovers        *
## factor(Club)Bolton Wanderers
## factor(Club)Bradford City
## factor(Club)Burnley
## factor(Club)Charlton Athletic
## factor(Club)Chelsea
## factor(Club)Coventry City           *
## factor(Club)Crystal Palace
## factor(Club)Derby County            .
## factor(Club)Everton
## factor(Club)Fulham                  *
## factor(Club)Hull City               .
## factor(Club)Leeds United            *
## factor(Club)Leicester City          *
## factor(Club)Liverpool
## factor(Club)Manchester City
## factor(Club)Manchester United
## factor(Club)Middlesbrough           *
## factor(Club)Newcastle United        *
## factor(Club)Norwich City
## factor(Club)Nottingham Forest       .
## factor(Club)Portsmouth
## factor(Club)Queens Park Rangers     *
## factor(Club)Reading                 *
## factor(Club)Sheffield Wednesday
## factor(Club)Southampton             .
## factor(Club)Stoke City
## factor(Club)Sunderland              *
## factor(Club)Swansea City
## factor(Club)Tottenham Hotspur
## factor(Club)Watford
## factor(Club)West Bromwich Albion    .
## factor(Club)West Ham United         *
## factor(Club)Wigan Athletic          *
## factor(Club)Wimbledon
## factor(Club)Wolverhampton Wanderers .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4321 on 204 degrees of freedom
##   (135 observations deleted due to missingness)
```

```
## Multiple R-squared:  0.7561, Adjusted R-squared:  0.7083
## F-statistic: 15.81 on 40 and 204 DF,  p-value: < 2.2e-16
```

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 group_by() to calculate average leagues position by club:

```
Avpos <- EPL %>% group_by(Club)%>%
              summarise(Position = mean(Position))
Avpos
```

```
## # A tibble: 44 x 2
##    Club              Position
##    <chr>                <dbl>
##  1 Arsenal               2.74
##  2 Aston Villa           9.95
##  3 Barnsley             19
##  4 Birmingham City      14.1
##  5 Blackburn Rovers     11.9
##  6 Blackpool            19
##  7 Bolton Wanderers     12.9
##  8 Bradford City        18.5
##  9 Burnley              18.5
## 10 Cardiff City         20
## # ... with 34 more rows
```

"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.

```
possal5_lm <- lm(lnpos ~ lnpos_lag + relsal +
                 factor(Club), data = EPL)

possal5_lm %>% summary()
```

```
##
## Call:
## lm(formula = lnpos ~ lnpos_lag + relsal + factor(Club), data = EPL)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7302 -0.2517  0.0000  0.2841  0.9946
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        -1.6927     0.2995  -5.651 5.31e-08
## lnpos_lag                           0.2819     0.0686   4.109 5.76e-05
## relsal                             11.0162     2.8482   3.868 0.000148
## factor(Club)Aston Villa            -0.5086     0.2045  -2.487 0.013680
## factor(Club)Birmingham City        -0.6294     0.2701  -2.331 0.020746
## factor(Club)Blackburn Rovers       -0.5323     0.2224  -2.393 0.017602
## factor(Club)Bolton Wanderers       -0.3853     0.2368  -1.627 0.105291
## factor(Club)Bradford City          -0.7382     0.4791  -1.541 0.124883
## factor(Club)Burnley                -0.5964     0.4845  -1.231 0.219746
## factor(Club)Charlton Athletic      -0.4071     0.2525  -1.612 0.108422
## factor(Club)Chelsea                -0.1476     0.1899  -0.777 0.438001
## factor(Club)Coventry City          -0.7989     0.3554  -2.248 0.025669
## factor(Club)Crystal Palace         -0.3558     0.3176  -1.120 0.263948
## factor(Club)Derby County           -0.5414     0.3125  -1.733 0.084638
## factor(Club)Everton                -0.2310     0.2053  -1.125 0.261858
## factor(Club)Fulham                 -0.5601     0.2214  -2.529 0.012186
## factor(Club)Hull City              -0.6423     0.3634  -1.767 0.078653
## factor(Club)Leeds United           -0.5942     0.2559  -2.322 0.021229
## factor(Club)Leicester City         -0.5652     0.2822  -2.003 0.046502
## factor(Club)Liverpool              -0.2836     0.1728  -1.641 0.102375
## factor(Club)Manchester City        -0.2169     0.1842  -1.177 0.240387
## factor(Club)Manchester United       0.2269     0.1740   1.304 0.193798
## factor(Club)Middlesbrough          -0.4562     0.2308  -1.977 0.049438
## factor(Club)Newcastle United       -0.4690     0.1976  -2.373 0.018575
## factor(Club)Norwich City           -0.4555     0.3164  -1.440 0.151404
## factor(Club)Nottingham Forest      -0.7921     0.4749  -1.668 0.096897
## factor(Club)Portsmouth             -0.4477     0.2811  -1.593 0.112785
## factor(Club)Queens Park Rangers    -0.9203     0.3567  -2.580 0.010582
## factor(Club)Reading                -0.8111     0.3618  -2.242 0.026053
## factor(Club)Sheffield Wednesday    -0.3931     0.4704  -0.836 0.404333
## factor(Club)Southampton            -0.3982     0.2372  -1.678 0.094842
## factor(Club)Stoke City             -0.3978     0.2665  -1.493 0.137054
## factor(Club)Sunderland             -0.5944     0.2321  -2.561 0.011152
## factor(Club)Swansea City           -0.2682     0.3074  -0.873 0.383843
## factor(Club)Tottenham Hotspur      -0.2782     0.1934  -1.438 0.151847
```

```
## factor(Club)Watford                    -0.6567     0.4826  -1.361 0.175089
## factor(Club)West Bromwich Albion       -0.4602     0.2504  -1.838 0.067513
## factor(Club)West Ham United            -0.5322     0.2197  -2.422 0.016318
## factor(Club)Wigan Athletic             -0.6185     0.2745  -2.253 0.025293
## factor(Club)Wimbledon                  -0.6416     0.4727  -1.357 0.176181
## factor(Club)Wolverhampton Wanderers   -0.7025     0.3661  -1.919 0.056395
##
## (Intercept)                          ***
## lnpos_lag                            ***
## relsal                               ***
## factor(Club)Aston Villa              *
## factor(Club)Birmingham City          *
## factor(Club)Blackburn Rovers         *
## factor(Club)Bolton Wanderers
## factor(Club)Bradford City
## factor(Club)Burnley
## factor(Club)Charlton Athletic
## factor(Club)Chelsea
## factor(Club)Coventry City            *
## factor(Club)Crystal Palace
## factor(Club)Derby County             .
## factor(Club)Everton
## factor(Club)Fulham                   *
## factor(Club)Hull City                .
## factor(Club)Leeds United             *
## factor(Club)Leicester City           *
## factor(Club)Liverpool
## factor(Club)Manchester City
## factor(Club)Manchester United
## factor(Club)Middlesbrough            *
## factor(Club)Newcastle United         *
## factor(Club)Norwich City
## factor(Club)Nottingham Forest        .
## factor(Club)Portsmouth
## factor(Club)Queens Park Rangers      *
## factor(Club)Reading                  *
## factor(Club)Sheffield Wednesday
## factor(Club)Southampton              .
## factor(Club)Stoke City
## factor(Club)Sunderland               *
## factor(Club)Swansea City
## factor(Club)Tottenham Hotspur
## factor(Club)Watford
## factor(Club)West Bromwich Albion     .
## factor(Club)West Ham United          *
```

```
## factor(Club)Wigan Athletic          *
## factor(Club)Wimbledon
## factor(Club)Wolverhampton Wanderers .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4321 on 204 degrees of freedom
##   (135 observations deleted due to missingness)
## Multiple R-squared:  0.7561, Adjusted R-squared:  0.7083
## F-statistic: 15.81 on 40 and 204 DF,  p-value: < 2.2e-16
```

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.

```
print(exp(2.1- 11*.02))
```

```
## [1] 6.553505
```

```
print(exp(2.1- 11*.08))
```

```
## [1] 3.387188
```

```r
print(exp(2.1- 11*.14))
```

```
## [1] 1.750673
```

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).