NBA_Lottery

Conor Nield

2023-10-23

NBA Draft Lottery

In the chunk below I will calculate the expected draft pick before the lottery, the chance of getting a lottery pick, and the difference between the draft lottery result and expected value.

```
import pandas as pd
df5 = pd.read_csv("/Users/conornield/Desktop/NBA_Lottery_3.csv")
#In this
def calculate_values(year, Rank, combinations, result, team, df5):
  #Sparse each year based on the number of lottery picks and number of lottery teams
    if year >= 2019:
        picks = 4
    elif year >= 1987:
        picks = 3
    else:
      #All teams from the 1985 and 1986 draft have the same expected value and lottery chance.
      return pd.Series([3.5, 1.0, 3.5 - result])


    if year == 1989:
        teams = 9
    elif year <= 1988:
        teams = 7
    elif year <= 1995:
        teams = 11
    elif year <= 2003:
        teams = 13
    else:
        teams = 14
    #chance of lottery pick
    lc = 0
    #expected pick position
    ev = 0
    #Values below will come into play for post lottery odds
    teams_above_0 = 0
    teams_above_1 = 0
    teams_above_2 = 0
    teams_above_3 = 0
    teams_above_4 = 0
    A = 0
    B = 0
    C = 0
```

```
D = 0

#Calculating pick 1 probability
#The 1996 - 1998 draft the Toronto Raptors and Vancouver Grizzlies were ineligible for the first pi
if (year > 1995 and year < 1999):
  if (team == 'TOR' or team == 'VAN'):
    prob_first = 0
  elif (year == 1996):
    prob_first = combinations/ (1000 - 407)
  elif (year == 1997):
    prob_first = combinations / (1000 - 273)
  elif (year == 1998):
    prob_first = combinations/ (1000 - 304)
else:
  prob_first = combinations / 1000
ev += prob_first
lc += prob_first
if (year > 1995 and year < 1999):
  other_teams_after_first = df5[(df5["Year"] == year) & (df5["Rank"] != Rank) & (df5["Team"] != 'TO
else:
  other_teams_after_first = df5[(df5["Year"] == year) & (df5["Rank"] != Rank)]
other_teams_after_first_2 = df5[(df5["Year"] == year) & (df5["Rank"] != Rank)]

# For the second pick

avg_comb_left_after_first = 1000 - sum((other_teams_after_first["Combinations"]**2) / (1000 - combi
prob_second = (combinations / avg_comb_left_after_first) * (1-prob_first)
ev += 2 * prob_second
lc += prob_second

# For the third pick
prob_third = 0
combo_removed = 0
for _, row_j in other_teams_after_first.iterrows():
  combo_j = row_j["Combinations"]
  Rank_j = row_j["Rank"]
  other_teams_after_j = other_teams_after_first_2[other_teams_after_first_2["Rank"] != Rank_j]
  p_j = combo_j / (1000 - combinations)
  for _, row_k in other_teams_after_j.iterrows():
    combo_k = row_k["Combinations"]
    p_k_given_j = combo_k / (1000 - combo_j - combinations)
    combo_removed += p_j * p_k_given_j * (combo_j + combo_k)
avg_comb_after_second = 1000 - combo_removed
prob_third += (combinations / (1000 - combo_removed)) * (1 - prob_second - prob_first)
ev += 3 * prob_third
lc += prob_third


# For the fourth pick, if applicable
prob_fourth = 0
combo_removed_third = 0
if (picks == 4):
  for _, row_j in other_teams_after_first.iterrows():
```

```
      combo_j = row_j["Combinations"]
      Rank_j = row_j["Rank"]
      other_teams_after_j = other_teams_after_first_2[other_teams_after_first_2["Rank"] != Rank_j]
      p_j = combo_j / (1000 - combinations)

      for _, row_k in other_teams_after_j.iterrows():
        combo_k = row_k["Combinations"]
        Rank_k = row_k["Rank"]
        p_k_given_j = combo_k / (1000 - combo_j - combinations)
        other_teams_after_k = other_teams_after_j[other_teams_after_j["Rank"] != Rank_k]

        for _, row_l in other_teams_after_k.iterrows():
          combo_l = row_l["Combinations"]
          p_l_given_j_k = combo_l / (1000 - combo_j - combo_k - combinations)
          combo_removed_third += p_j * p_k_given_j * p_l_given_j_k * (combo_j + combo_k + combo_l)
    avg_comb_after_third = 1000 - combo_removed_third
    prob_fourth += (combinations / (1000 - combo_removed_third)) * (1 - prob_second - prob_first - pr
    ev += 4 * prob_fourth
    lc += prob_fourth

  #Post lottery odds
  for _, row_m in other_teams_after_first.iterrows():
    combo_m = row_m["Combinations"]
    Rank_m = row_m["Rank"]
    p_m = combo_m/(1000 - combinations)
    if (Rank_m < Rank):
      A = 1
    else:
      A = 0
    other_teams_after_m = other_teams_after_first_2[other_teams_after_first_2["Rank"] != Rank_m]
    other_teams_after_first = df5[(df5["Year"] == year) & (df5["Rank"] != Rank)]
    for _, row_n in other_teams_after_m.iterrows():
      combo_n = row_n["Combinations"]
      Rank_n = row_n["Rank"]
      p_n_given_m = combo_n / (1000 - combo_m - combinations)
      other_teams_after_n = other_teams_after_m[other_teams_after_m["Rank"] != Rank_n]
      if (Rank_n < Rank):
        B = 1
      else:
        B = 0

      for _, row_o in other_teams_after_n.iterrows():
        combo_o = row_o["Combinations"]
        Rank_o = row_o["Rank"]
        p_o_given_m_n = combo_o / (1000 - combo_n - combo_m - combinations)
        if (Rank_o < Rank):
          C = 1
        else:
          C = 0
        other_teams_after_o = other_teams_after_n[other_teams_after_n["Rank"] != Rank_o]
        if (picks == 4):
          for _, row_p in other_teams_after_o.iterrows():
            combo_p = row_p["Combinations"]
```

```
            Rank_p = row_p["Rank"]
            p_p_given_m_n_o = combo_p / (1000 - combo_m - combo_n - combo_o - combinations)
            if (Rank_p < Rank):
              D = 1
            else:
              D = 0
            #A + B + C + D is the number of teams with a lower rank picked in the lottery
            if ((A + B + C + D) == 0):
              teams_above_0 += (p_m * p_n_given_m * p_o_given_m_n * p_p_given_m_n_o * (1 - lc))
            elif ((A + B + C + D) == 1):
              teams_above_1 += (p_m * p_n_given_m * p_o_given_m_n * p_p_given_m_n_o * (1 - lc))
            elif ((A + B + C + D) == 2):
              teams_above_2 += (p_m * p_n_given_m * p_o_given_m_n * p_p_given_m_n_o * (1 - lc))
            elif ((A + B + C + D) == 3):
              teams_above_3 += (p_m * p_n_given_m * p_o_given_m_n * p_p_given_m_n_o * (1 - lc))
            elif ((A + B + C + D) == 4):
              teams_above_4 += (p_m * p_n_given_m * p_o_given_m_n * p_p_given_m_n_o * (1 - lc))
          else:
            if ((A + B + C + D) == 0):
              teams_above_0 += (p_m * p_n_given_m * p_o_given_m_n * (1 - lc))
            elif ((A + B + C + D) == 1):
              teams_above_1 += (p_m * p_n_given_m * p_o_given_m_n * (1 - lc))
            elif ((A + B + C + D) == 2):
              teams_above_2 += (p_m * p_n_given_m * p_o_given_m_n * (1 - lc))
            elif ((A + B + C + D) == 3):
              teams_above_3 += (p_m * p_n_given_m * p_o_given_m_n * (1 - lc))

    if (picks == 4):
      ev += teams_above_4 * Rank + teams_above_3 * (Rank + 1) + teams_above_2 * (Rank + 2) + teams_above
    else:
      ev += teams_above_3 * Rank + teams_above_2 * (Rank + 1) + teams_above_1 * (Rank + 2) + teams_above

    return pd.Series([ev, lc, ev - result])
df5[["Expected_Value", "Lottery_Chance", "Lottery_luck"]] = df5.apply(lambda x: calculate_values(x['Yea
df5.to_csv("/Users/conornield/Desktop/NBA_Draft_Git.csv", index=False)
```

Total and average change by each team

```
grouped_df5 = df5.groupby('Team')['Lottery_luck'].agg(['mean', 'sum', 'count']).reset_index()
grouped_df5.columns = ['Team', 'Average_Lottery_Luck', 'Total_Lottery_Luck', 'Lottery_Instances']
sorted_df5 = grouped_df5.sort_values(by='Average_Lottery_Luck', ascending=False)
sorted_df5.to_csv("/Users/conornield/Desktop/NBA_Draft_Lottery_Git_3.csv", index=False)
print(sorted_df5)
```

```
##        Team  Average_Lottery_Luck  Total_Lottery_Luck  Lottery_Instances
## 26      SAS              1.078674            7.550717                  7
## 13      LAL              0.889268            8.003410                  9
## 18  NOP/CHA              0.838539           16.770776                 20
## 22      PHI              0.626320           11.273766                 18
## 20  OKC/SEA              0.533227            7.465174                 14
## 24      POR              0.516883            4.651948                  9
## 1       BKN              0.478736            8.138517                 17
## 14  MEM/VAN              0.462595            6.938929                 15
```

```
## 12      LAC              0.412364           10.309112                25
## 10      HOU              0.388643            4.275074                11
## 4       CHI              0.293225            4.105153                14
## 21      ORL              0.272576            5.724090                21
## 3       CHA              0.051867            0.829869                16
## 5       CLE             -0.000098           -0.001859                19
## 27      TOR             -0.002182           -0.034911                16
## 11      IND             -0.082390           -1.071067                13
## 25      SAC             -0.293488           -7.924176                27
## 16      MIL             -0.340734           -5.451748                16
## 17      MIN             -0.452963          -10.418149                23
## 9       GSW             -0.477279          -11.454708                24
## 19      NYK             -0.479948           -9.119012                19
## 28      UTA             -0.492265           -4.430384                 9
## 23      PHX             -0.529690           -9.004738                17
## 15      MIA             -0.534193           -5.876126                11
## 0       ATL             -0.582787           -8.159016                14
## 29      WAS             -0.640307          -14.727064                23
## 8       DET             -0.717319          -12.194426                17
## 2       BOS             -0.733399           -8.067393                11
## 7       DEN             -0.738629          -10.340809                14
## 6       DAL             -1.099275          -18.687671                17
```

```
#Average_Lottery_luck - the average amount a team changed from expected pick number to result
#Total_lottery_luck - the total amount of change between expected draft pick and draft lottery result
#Lottery_instances - amount of times a team was in the lottery
```

Now I'm going to calculate if having draft lottery luck is coorelated with future success.

```
df8 = pd.read_csv("/Users/conornield/Desktop/NBA_season_record.csv")
df8['Year'] = df8['Year'].astype(str).str[:4]
df8['Year'] = df8['Year'].astype(int)
df8 = df8.sort_values(by=['Team', 'Year'])
df8['previous_winning_percentage'] = df8.groupby('Team')['Winning Percentage'].shift(1)
df8 = df8[::-1]  # Reverse the DataFrame
df8['average_future_winning_percentage'] = df8.groupby('Team')['Winning Percentage'].rolling(8, min_per
df8 = df8[::-1]  # Reverse back to original order
df8['average_future_playoff_success'] = df8.groupby('Team')['Playoff outcome'].rolling(8, min_periods=1
df8 = df8[::-1]  # Reverse back to original order
df8.to_csv('updated_nba_data.csv', index=False)
df8['Year'] = df8['Year'].astype(int)
```

Now I'm merging the two data frames to make one data frame

```
df9 = pd.merge(df5, df8, on=['Team', 'Year'])
df9['change_in_winning_percentage'] = df9['average_future_winning_percentage'] - df9['previous_winning_
df9.to_csv('NBA_data.csv', index=False)
```

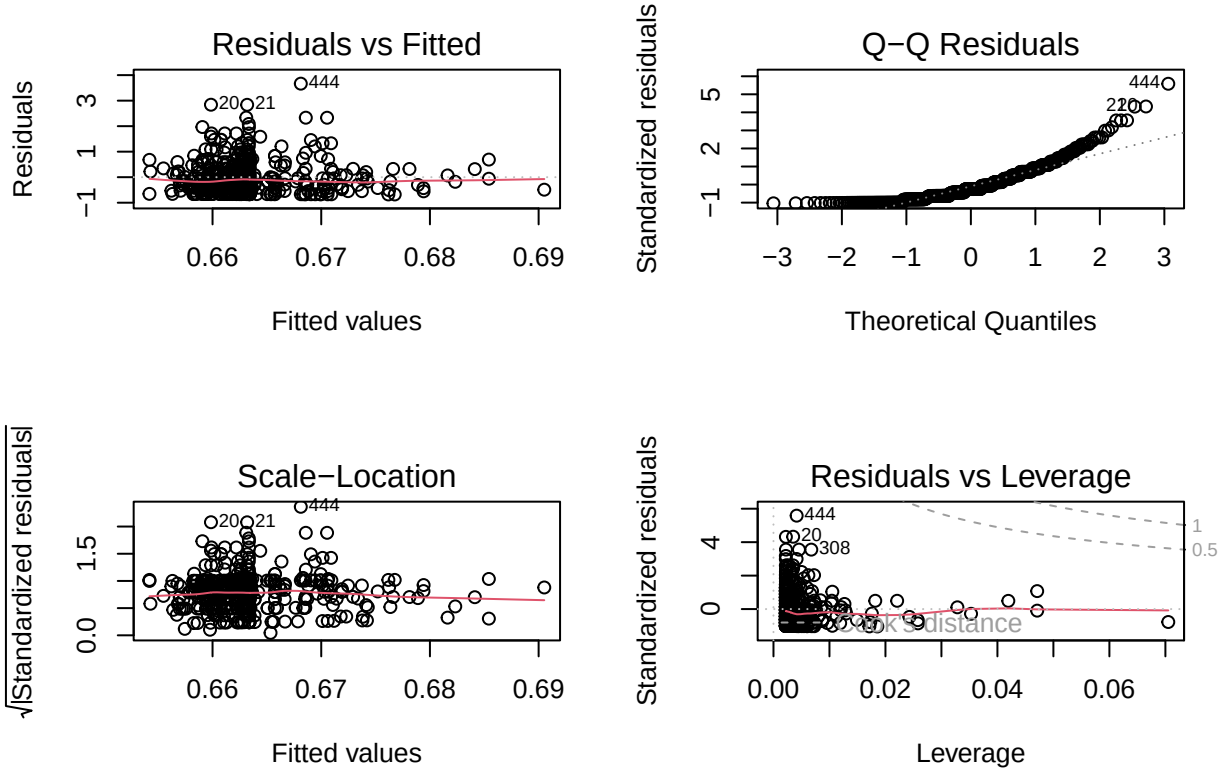
```
df10 <- read.table("NBA_data.csv", header = TRUE, sep=",")
model1 <- lm(average_future_playoff_success ~ Lottery_luck, data=df10)
model2 <- lm(change_in_winning_percentage ~ Lottery_luck, data=df10)
summary(model1)
```

```
##
## Call:
## lm(formula = average_future_playoff_success ~ Lottery_luck, data = df10)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6768 -0.4609 -0.1633  0.3366  3.6652
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.663943   0.030987  21.426   <2e-16 ***
## Lottery_luck 0.002785   0.017775   0.157    0.876
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6577 on 451 degrees of freedom
## Multiple R-squared:  5.442e-05,  Adjusted R-squared:  -0.002163
## F-statistic: 0.02454 on 1 and 451 DF,  p-value: 0.8756
```

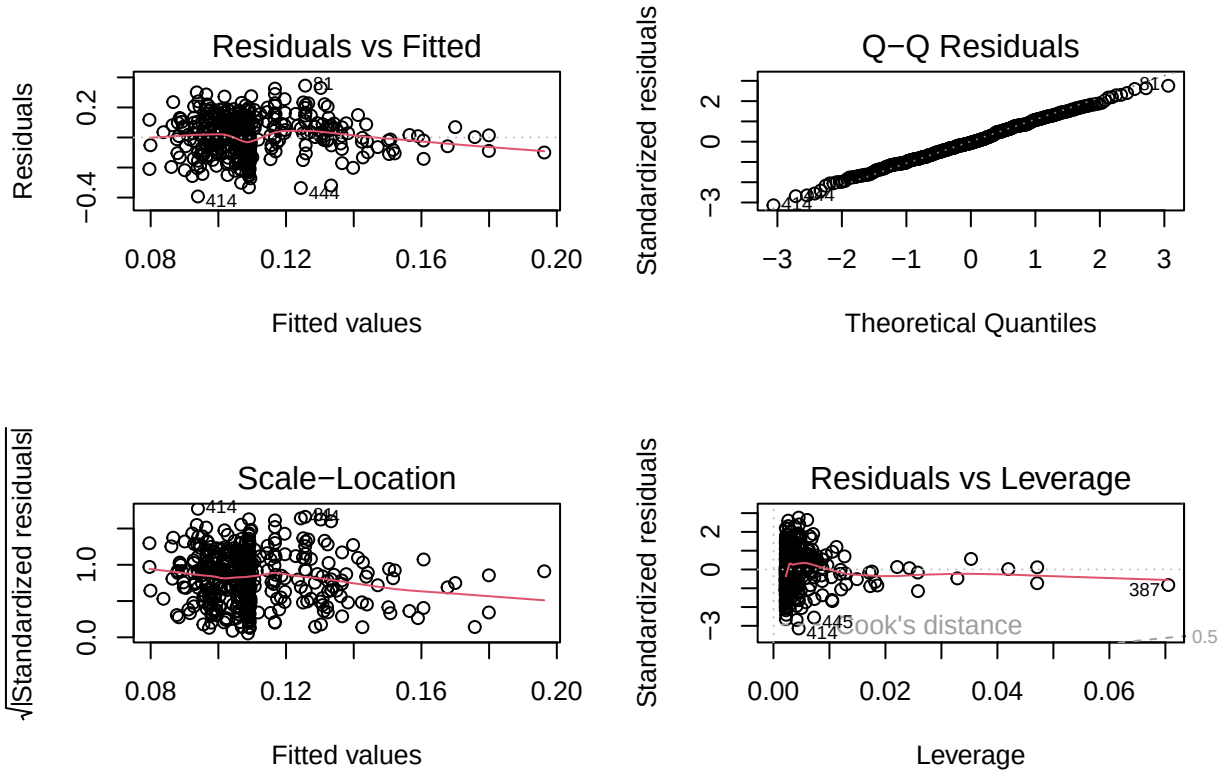
```
summary(model2)
```

```
##
## Call:
## lm(formula = change_in_winning_percentage ~ Lottery_luck, data = df10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39260 -0.08943 -0.00603  0.09299  0.34491
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.110947   0.005905   18.79  < 2e-16 ***
## Lottery_luck 0.008944   0.003387    2.64  0.00857 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1253 on 451 degrees of freedom
## Multiple R-squared:  0.01522,    Adjusted R-squared:  0.01304
## F-statistic: 6.971 on 1 and 451 DF,  p-value: 0.00857
```

```
par(mfrow = c(2,2))
plot(model1)
```


Residuals vs Fitted
Residuals
Fitted values
444
20
21
Q−Q Residuals
Standardized residuals
Theoretical Quantiles
Scale−Location
√|Standardized residuals|
Residuals vs Leverage
Leverage
Cook's distance
308
1
0.5

```
plot(model2)
```



Change in winning percentage is statistically coorelated with lottery luck. Playoff success is not. The biggest outlier is the Golden State Warriors 1995 draft lottery in which they were ranked 5th pre draft and ended up with the first pick. Even though the lottery luck was high that year, they ended up with around the same record the following 8 years. The relationship between playoff success and lottery luck does not have normally distributed errors, indicating that it is not a liner relationship.