

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
df = pd.read_csv("/Users/conornield/Desktop/NBA_Draft_Lottery_4.csv")
#In this
def calculate_values(year, Rank, combinations, result, team, df):
    #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 = df[(df["Year"] == year) & (df["Rank"] != Rank) & (df["Team"] != 'TOR')]
else:
    other_teams_after_first = df[(df["Year"] == year) & (df["Rank"] != Rank)]
other_teams_after_first_2 = df[(df["Year"] == year) & (df["Rank"] != Rank)]

# For the second pick

avg_comb_left_after_first = 1000 - sum((other_teams_after_first["Combinations"]**2) / (1000 - combinations))
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 - prob_third)
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 = df[(df["Year"] == year) & (df["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_1 * (Rank + 3) + teams_above_0 * (Rank + 4)
else:
    ev += teams_above_3 * Rank + teams_above_2 * (Rank + 1) + teams_above_1 * (Rank + 2) + teams_above_0 * (Rank + 3)

return pd.Series([ev, lc, result - ev])
df[["Expected_Value", "Lottery_Chance", "Change"]] = df.apply(lambda x: calculate_values(x['Year'], x['Team'], x['Rank']), axis=1)
df.to_csv("/Users/conornield/Desktop/NBA_Draft_Lottery_Git.csv", index=False)

```

Total and average change by each team

```

total_change = df.groupby('Team')['Change'].sum().reset_index()
grouped_df = df.groupby('Team')['Change'].mean().reset_index()
sorted_df = grouped_df.sort_values(by='Change', ascending=False)

sorted_df.to_csv("/Users/conornield/Desktop/NBA_Draft_Lottery_Git_2.csv", index=False)

```