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
import math

df5 = pd.read_csv("/Users/conornield/Desktop/NBA_Lottery_2.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([4, 1.0, 4 - result, 2, (4-result)/2])


    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
    var = 0
    sd = 0
    #Values below will come into play for post lottery odds
    move_back_0 = 0
    move_back_1 = 0
    move_back_2 = 0
    move_back_3 = 0
```

```
move_back_4 = 0
A = 0
B = 0
C = 0
D = 0
z = 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 better odds picked in the lottery
              if ((A + B + C + D) == 0):
                move_back_4 += (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):
                move_back_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) == 2):
                move_back_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):
                move_back_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) == 4):
                move_back_0 += (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):
              move_back_3 += (p_m * p_n_given_m * p_o_given_m_n * (1 - lc))
            elif ((A + B + C + D) == 1):
              move_back_2 += (p_m * p_n_given_m * p_o_given_m_n * (1 - lc))
            elif ((A + B + C + D) == 2):
              move_back_1 += (p_m * p_n_given_m * p_o_given_m_n * (1 - lc))
            elif ((A + B + C + D) == 3):
              move_back_0 += (p_m * p_n_given_m * p_o_given_m_n * (1 - lc))

    if (picks == 4):
      ev += move_back_0 * Rank + move_back_1 * (Rank + 1) + move_back_2 * (Rank + 2) + move_back_3 * (Ra
      var = ((1 - ev)**2 * prob_first) + ((2 - ev)**2 * prob_second) + ((3 - ev)**2 * prob_third) + ((4
      sd = math.sqrt(var)
      z = (ev - result) / sd

    else:
      ev += move_back_0 * Rank + move_back_1 * (Rank + 1) + move_back_2 * (Rank + 2) + move_back_3 * (Ra
      var = ((1 - ev)**2 * prob_first) + ((2 - ev)**2 * prob_second) + ((3 - ev)**2 * prob_third) + ((4
      sd = math.sqrt(var)
      z = (ev - result) / sd

    return pd.Series([ev, lc, ev - result, sd, z])
df5[["Expected_Value", "Lottery_Chance", "Difference", "Standard_deviation", "Z"]] = df5.apply(lambda x
df5.to_csv("/Users/conornield/Desktop/NBA_Draft_Git.csv", index=False)
```

Total and average change by each team

```
grouped_df5 = df5.groupby('Team')['Z'].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
## 24      SAS              0.647658            4.533606                  7
## 4   CHA/NOP              0.443604            8.872087                 20
## 14      LAL              0.438695            3.948259                  9
## 20      PHI              0.387681            6.978261                 18
## 1       BKN              0.316957            5.388269                 17
## 13      LAC              0.285256            7.131390                 25
## 11      HOU              0.226392            2.490312                 11
## 25  SEA/OKC              0.211969            2.967560                 14
## 19      ORL              0.196416            4.124731                 21
## 22      POR              0.193719            1.743468                  9
## 5       CHI              0.174908            2.448715                 14
## 28  VAN/MEM              0.082269            1.234042                 15
## 6       CLE              0.043773            0.831680                 19
## 12      IND             -0.015756           -0.204829                 13
## 3       CHA             -0.034526           -0.552422                 16
## 26      TOR             -0.036673           -0.586770                 16
## 16      MIL             -0.156277           -2.500434                 16
## 23      SAC             -0.185413           -5.006152                 27
## 18      NYK             -0.219602           -4.172439                 19
## 17      MIN             -0.231791           -5.331202                 23
## 10      GSW             -0.251245           -6.029883                 24
## 0       ATL             -0.251568           -3.521957                 14
## 21      PHX             -0.257872           -4.383830                 17
## 27      UTA             -0.268524           -2.416714                  9
## 29      WAW             -0.278606           -6.407932                 23
## 15      MIA             -0.296531           -3.261842                 11
## 9       DET             -0.342272           -5.818618                 17
## 2       BOS             -0.392603           -4.318629                 11
## 8       DEN             -0.424677           -5.945482                 14
## 7       DAL             -0.490388           -8.336597                 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_1.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_2'] = df8.groupby('Team')['Winning Percentage'].rolling(2, min_p
df8['average_future_winning_percentage_3'] = df8.groupby('Team')['Winning Percentage'].rolling(3, min_p
df8['average_future_winning_percentage_4'] = df8.groupby('Team')['Winning Percentage'].rolling(4, min_p
df8['average_future_winning_percentage_5'] = df8.groupby('Team')['Winning Percentage'].rolling(5, min_p
df8['average_future_winning_percentage_6'] = df8.groupby('Team')['Winning Percentage'].rolling(6, min_p
```

```
df8['average_future_winning_percentage_8'] = df8.groupby('Team')['Winning Percentage'].rolling(8, min_p
df8['average_future_winning_percentage_10'] = df8.groupby('Team')['Winning Percentage'].rolling(10, min
df8['average_future_winning_percentage_12'] = df8.groupby('Team')['Winning Percentage'].rolling(12, min
df8['average_future_playoff_success_2'] = df8.groupby('Team')['Playoff outcome'].rolling(2, min_periods
df8['average_future_playoff_success_3'] = df8.groupby('Team')['Playoff outcome'].rolling(3, min_periods
df8['average_future_playoff_success_4'] = df8.groupby('Team')['Playoff outcome'].rolling(4, min_periods
df8['average_future_playoff_success_5'] = df8.groupby('Team')['Playoff outcome'].rolling(5, min_periods
df8['average_future_playoff_success_6'] = df8.groupby('Team')['Playoff outcome'].rolling(6, min_periods
df8['average_future_playoff_success_8'] = df8.groupby('Team')['Playoff outcome'].rolling(8, min_periods
df8['average_future_playoff_success_10'] = df8.groupby('Team')['Playoff outcome'].rolling(10, min_perio
df8['average_future_playoff_success_12'] = df8.groupby('Team')['Playoff outcome'].rolling(12, min_perio
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_2'] = df9['average_future_winning_percentage_2'] - df9['previous_winni
df9['change_in_winning_percentage_3'] = df9['average_future_winning_percentage_3'] - df9['previous_winni
df9['change_in_winning_percentage_4'] = df9['average_future_winning_percentage_4'] - df9['previous_winni
df9['change_in_winning_percentage_5'] = df9['average_future_winning_percentage_5'] - df9['previous_winni
df9['change_in_winning_percentage_6'] = df9['average_future_winning_percentage_6'] - df9['previous_winni
df9['change_in_winning_percentage_8'] = df9['average_future_winning_percentage_8'] - df9['previous_winni
df9['change_in_winning_percentage_10'] = df9['average_future_winning_percentage_10'] - df9['previous_wi
df9['change_in_winning_percentage_12'] = df9['average_future_winning_percentage_12'] - df9['previous_wi
df9.to_csv('NBA_data.csv', index=False)
```

```
df10 <- read.table("NBA_data.csv", header = TRUE, sep=",")
model2_RS <- lm(change_in_winning_percentage_2 ~ Z, data=df10)
model3_RS <- lm(change_in_winning_percentage_3 ~ Z, data=df10)
model4_RS <- lm(change_in_winning_percentage_4 ~ Z, data=df10)
model5_RS <- lm(change_in_winning_percentage_5 ~ Z, data=df10)
model6_RS <- lm(change_in_winning_percentage_6 ~ Z, data=df10)
model8_RS <- lm(change_in_winning_percentage_8 ~ Z, data=df10)
model10_RS <- lm(change_in_winning_percentage_10 ~ Z, data=df10)
model12_RS <- lm(change_in_winning_percentage_12 ~ Z, data=df10)
summary(model2_RS)
```

```
##
## Call:
## lm(formula = change_in_winning_percentage_2 ~ Z, data = df10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32184 -0.08773  0.00144  0.08299  0.44904
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.067627   0.006408  10.553  < 2e-16 ***
## Z           0.019229   0.007072   2.719  0.00683 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1284 on 401 degrees of freedom
## Multiple R-squared:  0.0181, Adjusted R-squared:  0.01566
## F-statistic: 7.394 on 1 and 401 DF,  p-value: 0.00683
```

```
summary(model3_RS)
```

```
##
## Call:
## lm(formula = change_in_winning_percentage_3 ~ Z, data = df10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39048 -0.09036  0.00321  0.09070  0.38177
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.080758   0.006484  12.455  < 2e-16 ***
## Z           0.021005   0.007156   2.935  0.00352 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1299 on 401 degrees of freedom
## Multiple R-squared:  0.02104,    Adjusted R-squared:  0.01859
## F-statistic: 8.617 on 1 and 401 DF,  p-value: 0.003523
```

```
summary(model4_RS)
```

```
##
## Call:
## lm(formula = change_in_winning_percentage_4 ~ Z, data = df10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42152 -0.09103 -0.00158  0.09361  0.36335
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.090228   0.006512  13.856  < 2e-16 ***
## Z           0.021811   0.007186   3.035  0.00256 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1304 on 401 degrees of freedom
## Multiple R-squared:  0.02246,    Adjusted R-squared:  0.02002
## F-statistic: 9.212 on 1 and 401 DF,  p-value: 0.002561
```

```
summary(model5_RS)
```

```
##
## Call:
```

```
## lm(formula = change_in_winning_percentage_5 ~ Z, data = df10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38796 -0.08925  0.00303  0.09327  0.37103
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.097100   0.006520  14.894  < 2e-16 ***
## Z           0.018754   0.007195   2.607  0.00949 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1306 on 401 degrees of freedom
## Multiple R-squared:  0.01666,    Adjusted R-squared:  0.01421
## F-statistic: 6.794 on 1 and 401 DF,  p-value: 0.009485
```

```
summary(model6_RS)
```

```
##
## Call:
## lm(formula = change_in_winning_percentage_6 ~ Z, data = df10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38674 -0.09280 -0.00775  0.09305  0.36644
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.102449   0.006509  15.741   <2e-16 ***
## Z           0.016653   0.007183   2.319   0.0209 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1304 on 401 degrees of freedom
## Multiple R-squared:  0.01323,    Adjusted R-squared:  0.01077
## F-statistic: 5.375 on 1 and 401 DF,  p-value: 0.02092
```

```
summary(model8_RS)
```

```
##
## Call:
## lm(formula = change_in_winning_percentage_8 ~ Z, data = df10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39799 -0.08736 -0.00910  0.09636  0.34087
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.110043   0.006347  17.337   <2e-16 ***
## Z           0.014372   0.007005   2.052   0.0409 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1272 on 401 degrees of freedom
## Multiple R-squared:  0.01039,    Adjusted R-squared:  0.00792
## F-statistic: 4.209 on 1 and 401 DF,  p-value: 0.04085
```

```
summary(model10_RS)
```

```
##
## Call:
## lm(formula = change_in_winning_percentage_10 ~ Z, data = df10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37578 -0.08255 -0.00278  0.08805  0.34064
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.114495   0.006238   18.35   <2e-16 ***
## Z           0.011289   0.006884    1.64    0.102
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.125 on 401 degrees of freedom
## Multiple R-squared:  0.00666,    Adjusted R-squared:  0.004183
## F-statistic: 2.689 on 1 and 401 DF,  p-value: 0.1018
```

```
summary(model12_RS)
```

```
##
## Call:
## lm(formula = change_in_winning_percentage_12 ~ Z, data = df10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42084 -0.08291 -0.00482  0.08493  0.33866
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.118026   0.006167  19.138   <2e-16 ***
## Z           0.010150   0.006806   1.491    0.137
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1235 on 401 degrees of freedom
## Multiple R-squared:  0.005516,   Adjusted R-squared:  0.003036
## F-statistic: 2.224 on 1 and 401 DF,  p-value: 0.1366
```

```
model2_P <- lm(average_future_playoff_success_2 ~ Z, data=df10)
model3_P <- lm(average_future_playoff_success_3 ~ Z, data=df10)
model4_P <- lm(average_future_playoff_success_4 ~ Z, data=df10)
model5_P <- lm(average_future_playoff_success_5 ~ Z, data=df10)
```

```r
model6_P <- lm(average_future_playoff_success_6 ~ Z, data=df10)
model8_P <- lm(average_future_playoff_success_8 ~ Z, data=df10)
model10_P <- lm(average_future_playoff_success_10 ~ Z, data=df10)
model12_P <- lm(average_future_playoff_success_12 ~ Z, data=df10)
summary(model2_P)
```

```
##
## Call:
## lm(formula = average_future_playoff_success_2 ~ Z, data = df10)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8953 -0.5635 -0.4577  0.4277  3.4959
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.58566    0.04196  13.958   <2e-16 ***
## Z            0.10562    0.04631   2.281   0.0231 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8405 on 401 degrees of freedom
## Multiple R-squared:  0.01281,    Adjusted R-squared:  0.01035
## F-statistic: 5.203 on 1 and 401 DF,  p-value: 0.02307
```

```r
summary(model3_P)
```

```
##
## Call:
## lm(formula = average_future_playoff_success_3 ~ Z, data = df10)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0609 -0.6289 -0.3006  0.3455  3.7089
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.68177    0.04174  16.332  < 2e-16 ***
## Z            0.12933    0.04607   2.807  0.00524 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8362 on 401 degrees of freedom
## Multiple R-squared:  0.01927,    Adjusted R-squared:  0.01683
## F-statistic: 7.881 on 1 and 401 DF,  p-value: 0.005241
```

```r
summary(model4_P)
```

```
##
## Call:
## lm(formula = average_future_playoff_success_4 ~ Z, data = df10)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1142 -0.6457 -0.2264  0.3543  3.8134
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.74283    0.04104  18.099  < 2e-16 ***
## Z            0.12669    0.04529   2.797  0.00541 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8222 on 401 degrees of freedom
## Multiple R-squared:  0.01914,    Adjusted R-squared:  0.01669
## F-statistic: 7.823 on 1 and 401 DF,  p-value: 0.005407
```

```
summary(model5_P)
```

```
##
## Call:
## lm(formula = average_future_playoff_success_5 ~ Z, data = df10)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0522 -0.5804 -0.2827  0.4196  3.2522
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.79453    0.03995  19.889   <2e-16 ***
## Z            0.10521    0.04409   2.386   0.0175 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8003 on 401 degrees of freedom
## Multiple R-squared:  0.014,  Adjusted R-squared:  0.01154
## F-statistic: 5.695 on 1 and 401 DF,  p-value: 0.01748
```

```
summary(model6_P)
```

```
##
## Call:
## lm(formula = average_future_playoff_success_6 ~ Z, data = df10)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9923 -0.6088 -0.2480  0.4499  2.8690
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.83344    0.03949  21.106   <2e-16 ***
## Z            0.08072    0.04358   1.852   0.0647 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.791 on 401 degrees of freedom
## Multiple R-squared:  0.008484,   Adjusted R-squared:  0.006011
## F-statistic: 3.431 on 1 and 401 DF,  p-value: 0.06471
```

```
summary(model8_P)
```

```
##
## Call:
## lm(formula = average_future_playoff_success_8 ~ Z, data = df10)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9751 -0.5877 -0.1908  0.4115  3.1480
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.87465    0.03750  23.322   <2e-16 ***
## Z            0.05104    0.04139   1.233    0.218
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7512 on 401 degrees of freedom
## Multiple R-squared:  0.003778,   Adjusted R-squared:  0.001293
## F-statistic: 1.521 on 1 and 401 DF,  p-value: 0.2183
```

```
summary(model10_P)
```

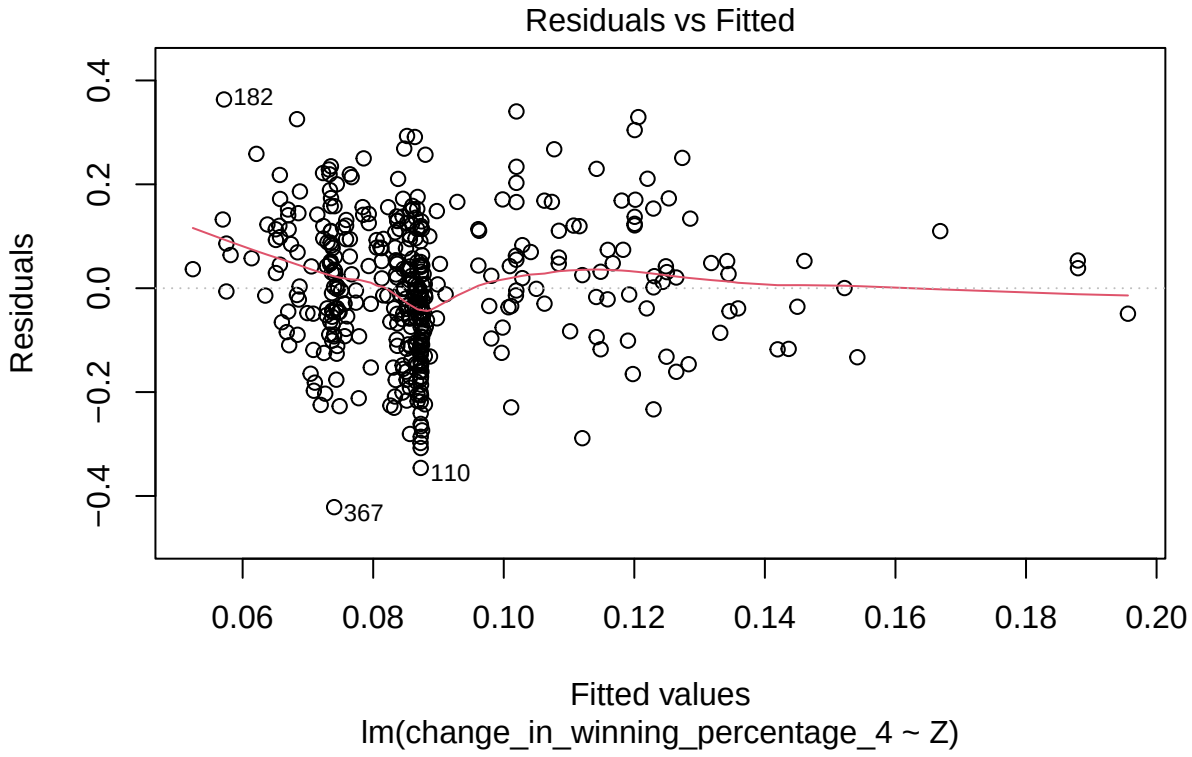
```
##
## Call:
## lm(formula = average_future_playoff_success_10 ~ Z, data = df10)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.96995 -0.57667 -0.09447  0.40167  2.81273
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.90248    0.03576  25.234   <2e-16 ***
## Z            0.03428    0.03947   0.868    0.386
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7164 on 401 degrees of freedom
## Multiple R-squared:  0.001877,   Adjusted R-squared:  -0.0006117
## F-statistic: 0.7542 on 1 and 401 DF,  p-value: 0.3857
```

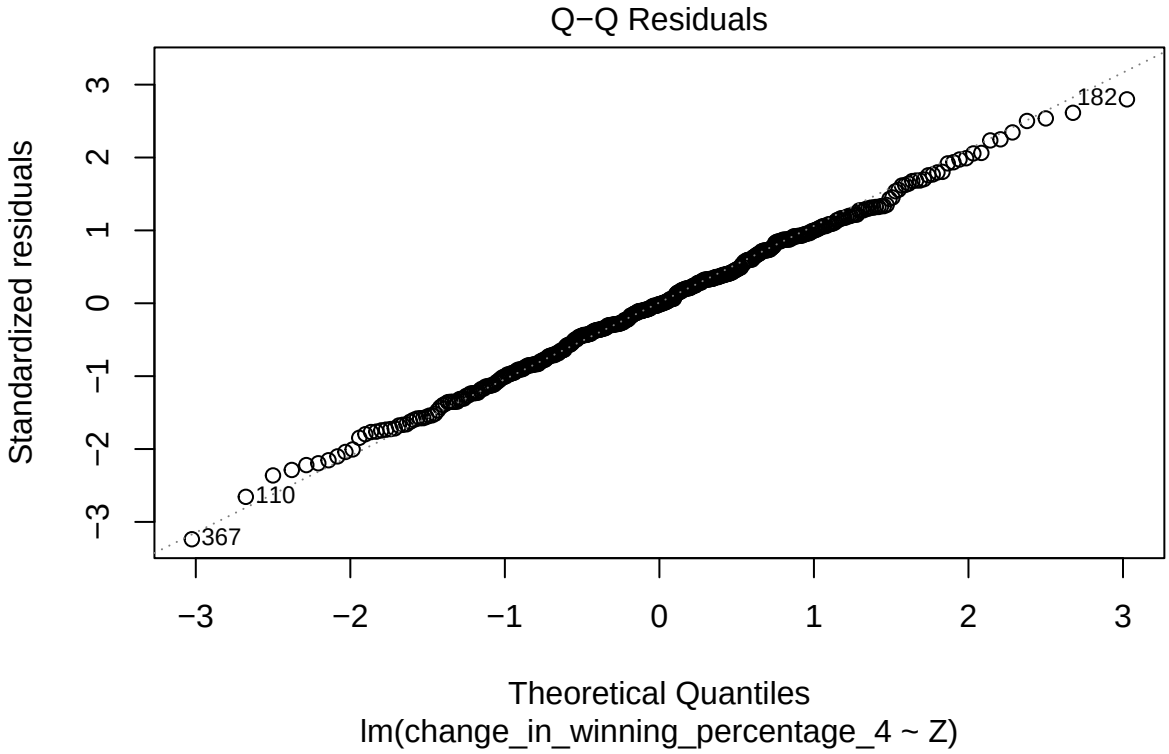
```
summary(model12_P)
```

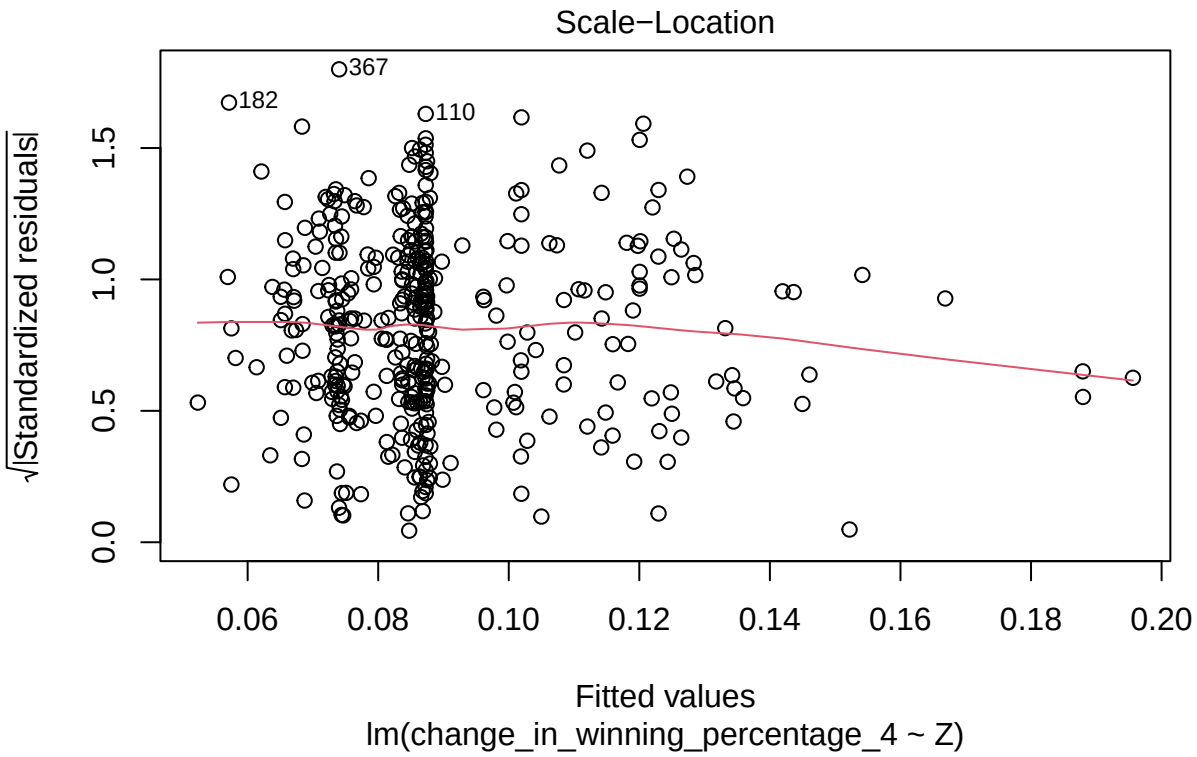
```
##
## Call:
## lm(formula = average_future_playoff_success_12 ~ Z, data = df10)
##
```

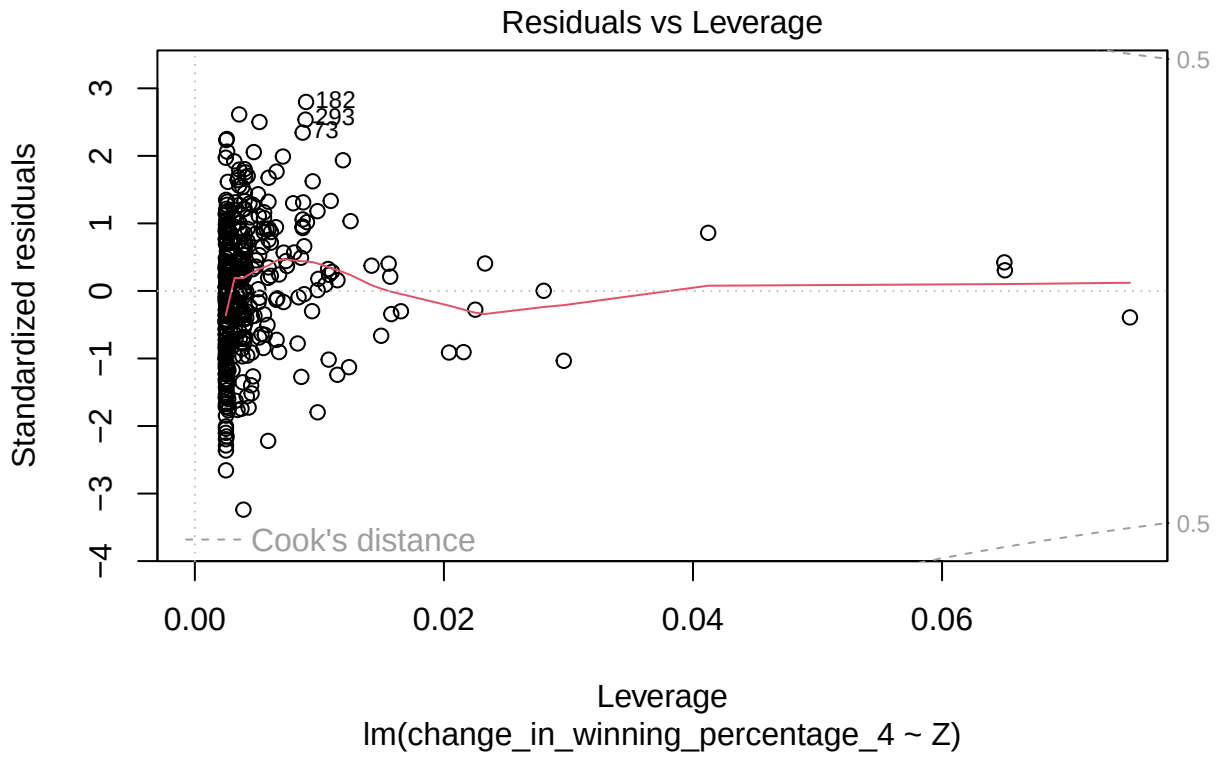
```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9781 -0.5801 -0.0691  0.3305  2.5924
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.92607    0.03441  26.916   <2e-16 ***
## Z            0.03138    0.03797   0.826    0.409
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6892 on 401 degrees of freedom
## Multiple R-squared:  0.0017, Adjusted R-squared:  -0.0007892
## F-statistic: 0.683 on 1 and 401 DF,  p-value: 0.4091
```

```
plot(model4_RS)
```

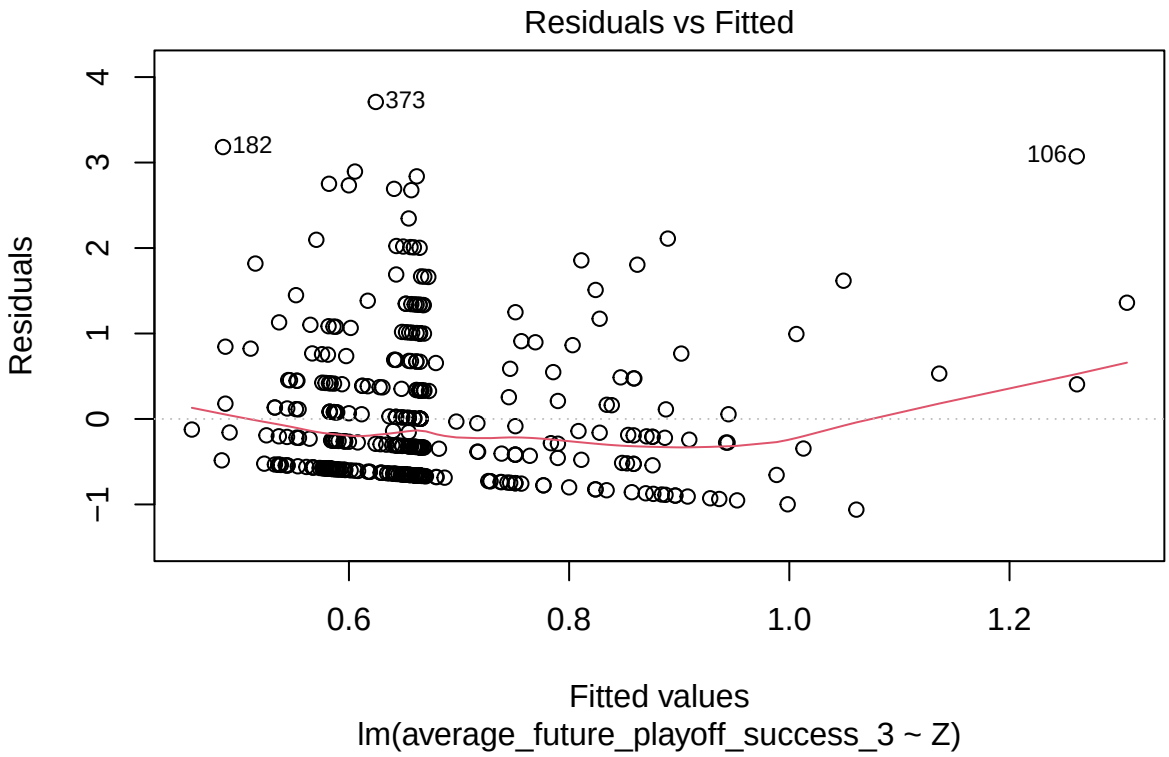


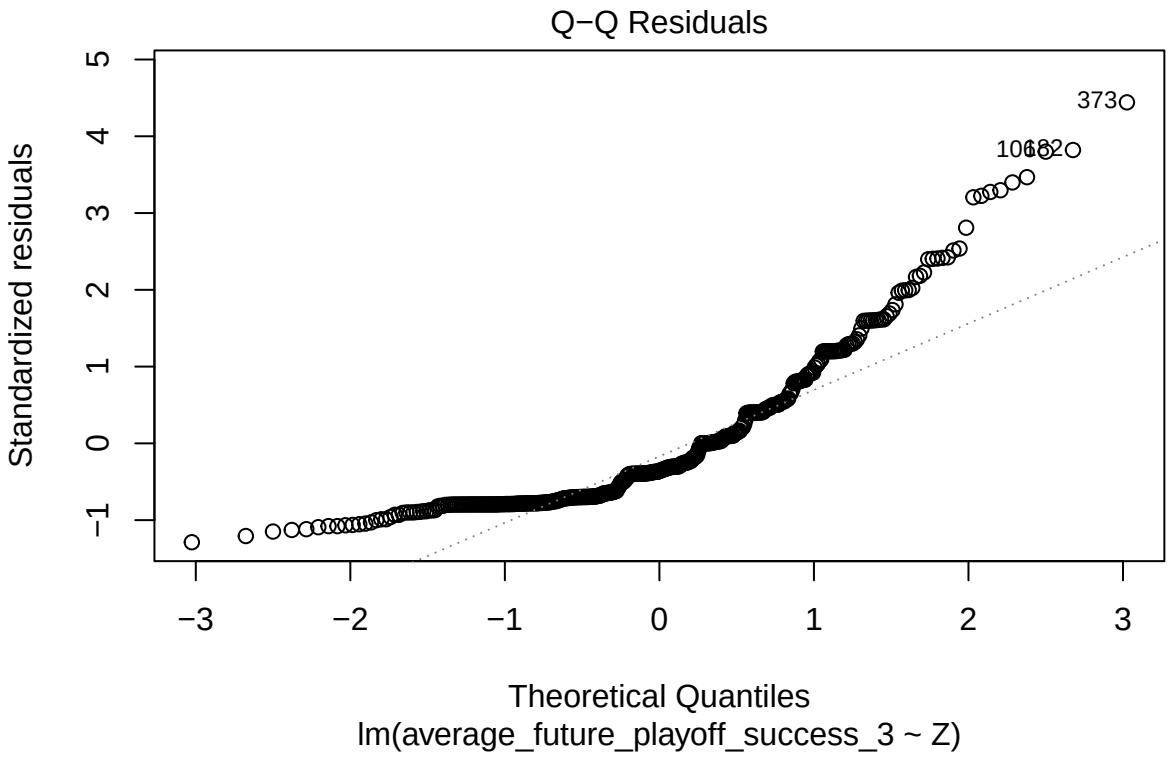

Q−Q Residuals
Standardized residuals
−3
−2
−1
0
1
2
3
182
110
367
Theoretical Quantiles
lm(change_in_winning_percentage_4 ~ Z)

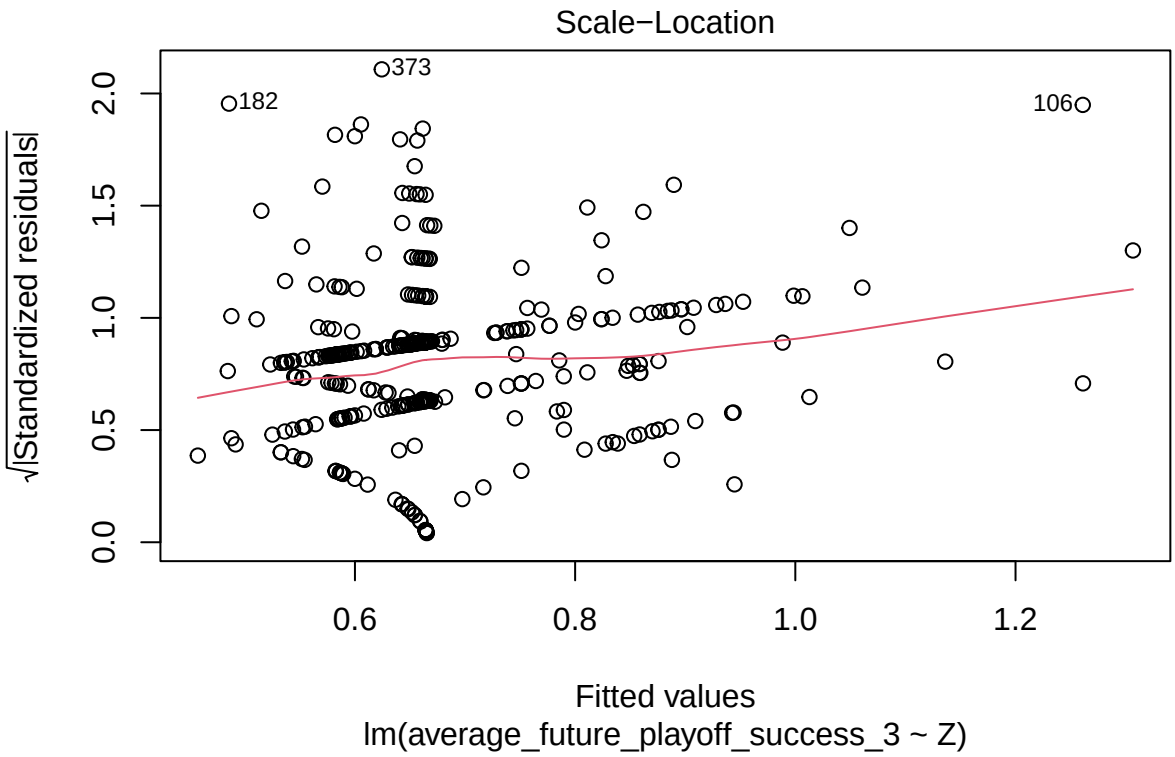

Scale−Location
367
182
110
√|Standardized residuals|
0.0
0.5
1.0
1.5
0.06
0.08
0.10
0.12
0.14
0.16
0.18
0.20
Fitted values
lm(change_in_winning_percentage_4 ~ Z)

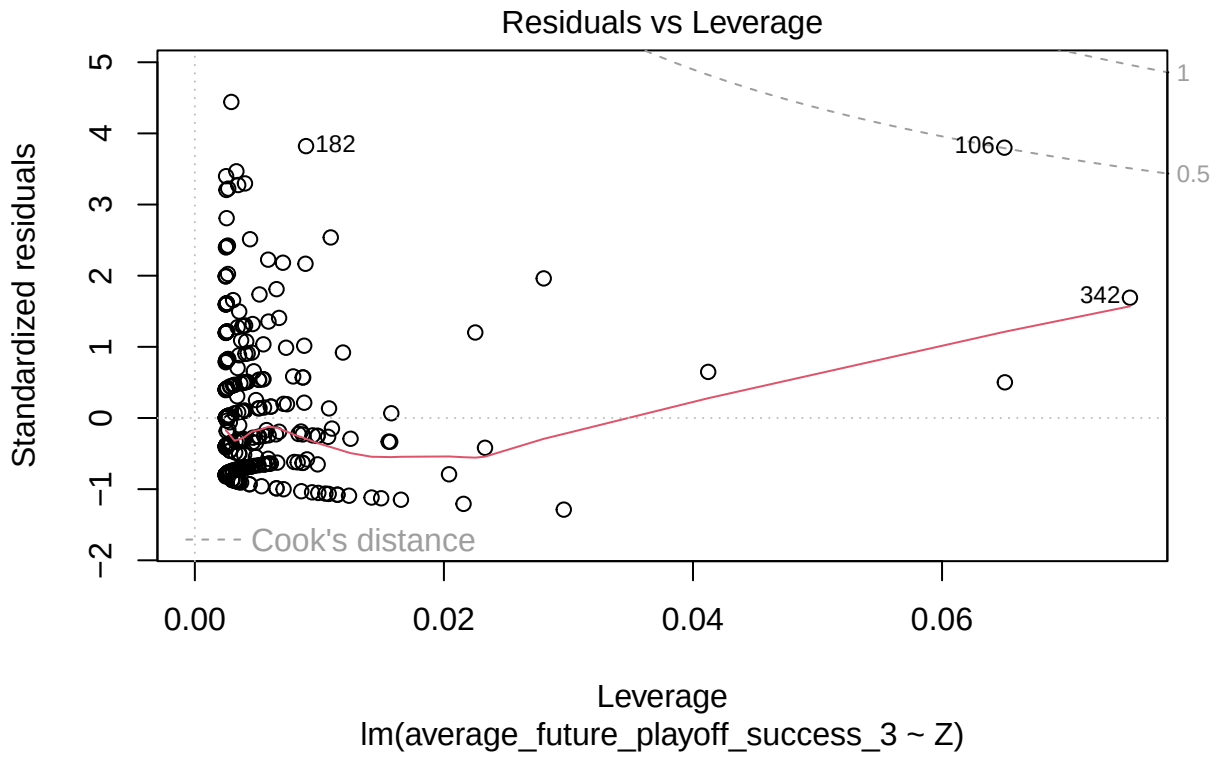



```
plot(model3_P)
```

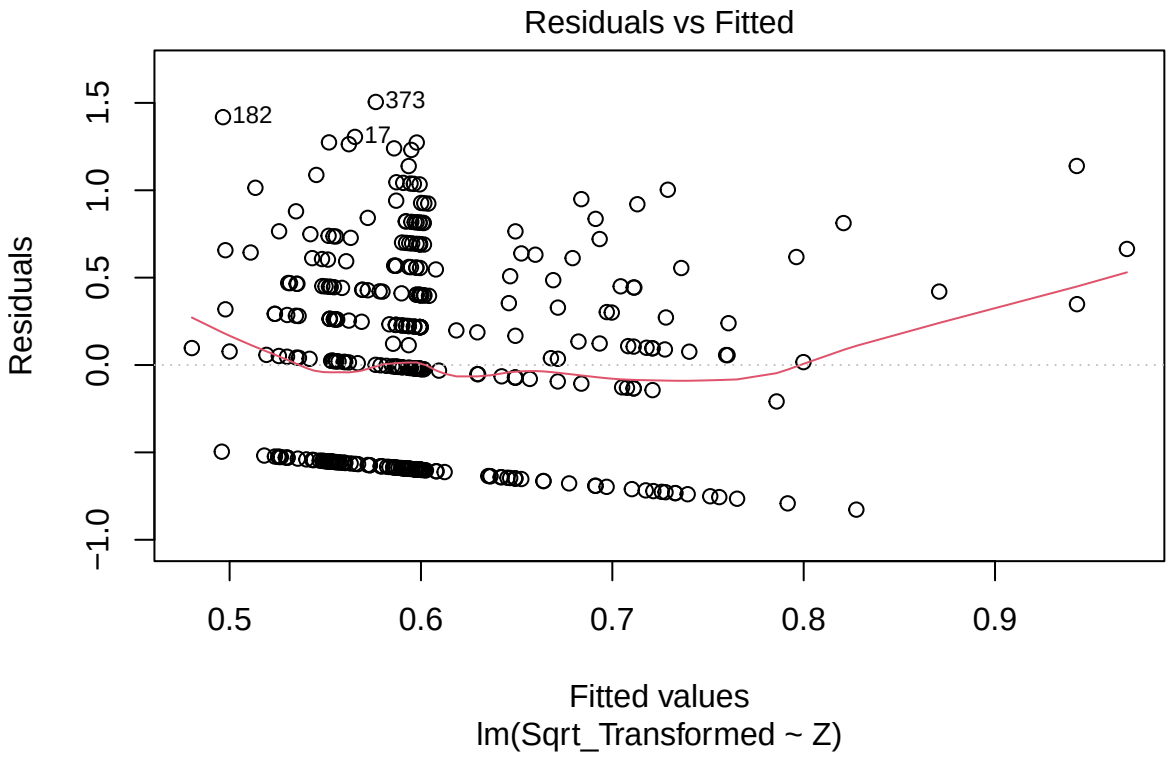

Residuals vs Fitted
Residuals
Fitted values
lm(average_future_playoff_success_3 ~ Z)
373
182
106
4
3
2
1
0
−1
0.6
0.8
1.0
1.2

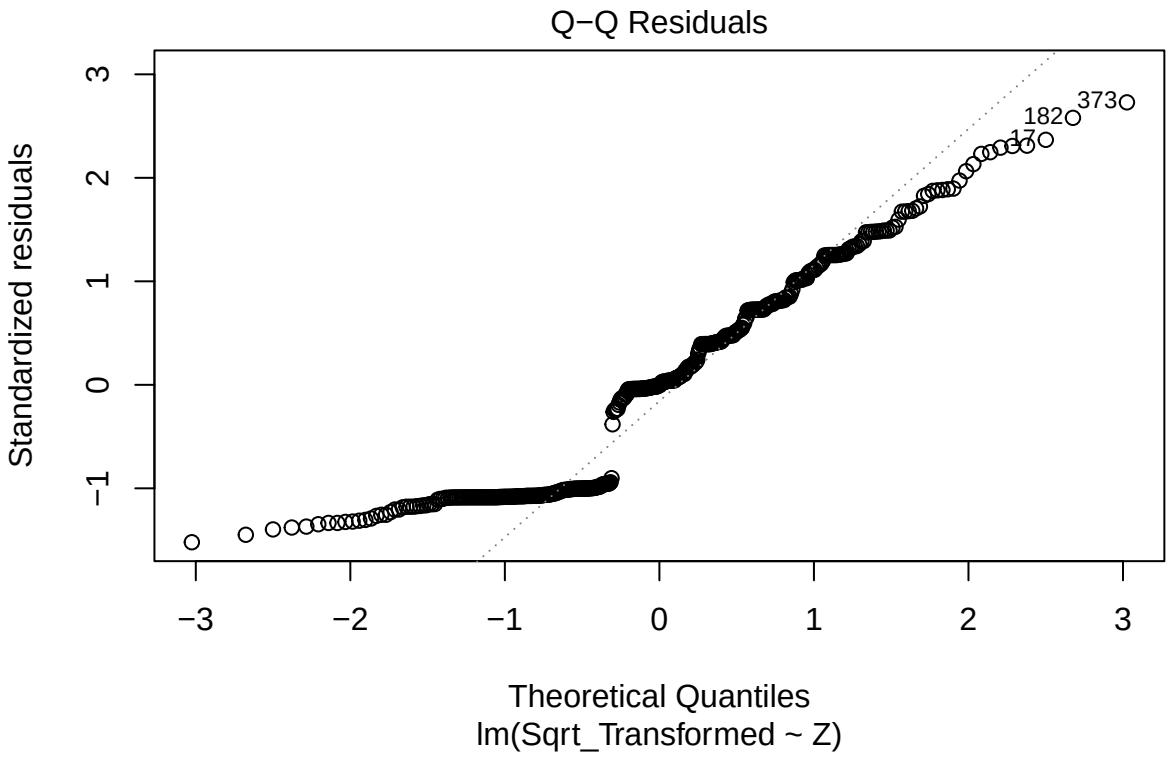

Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(average_future_playoff_success_3 ~ Z)
373

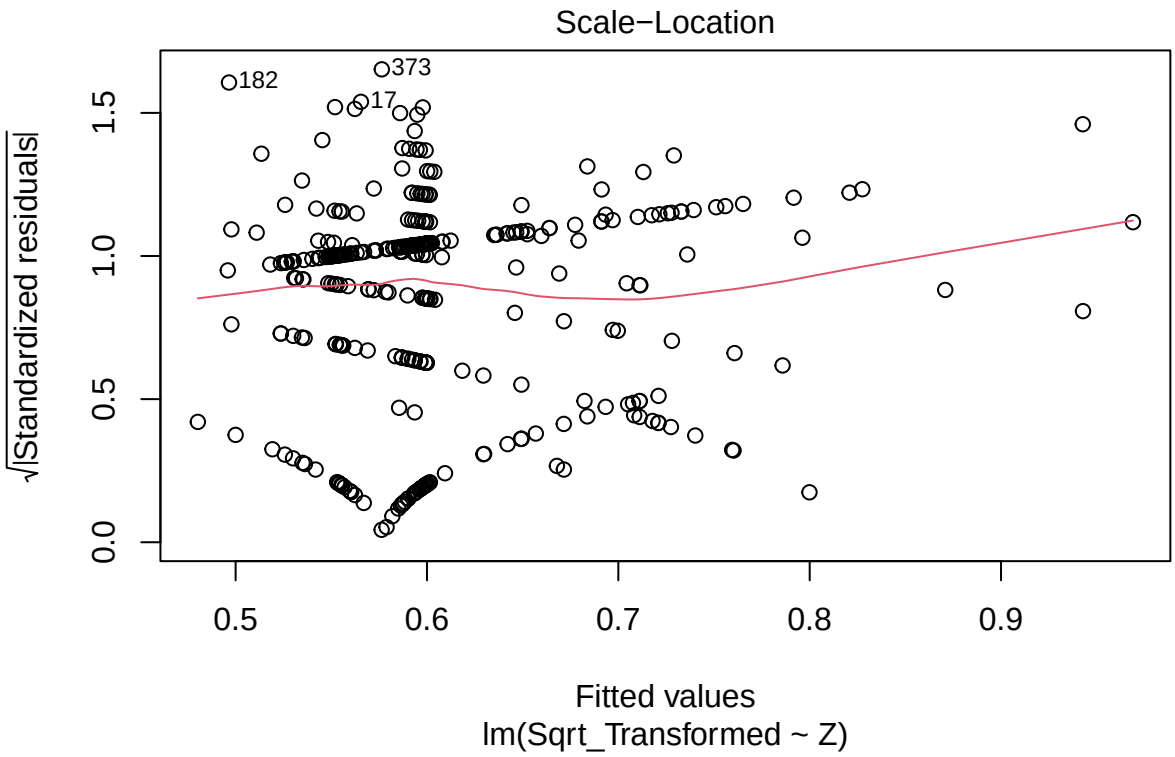

Scale−Location
373
182
106
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
0.6
0.8
1.0
1.2
Fitted values
lm(average_future_playoff_success_3 ~ Z)

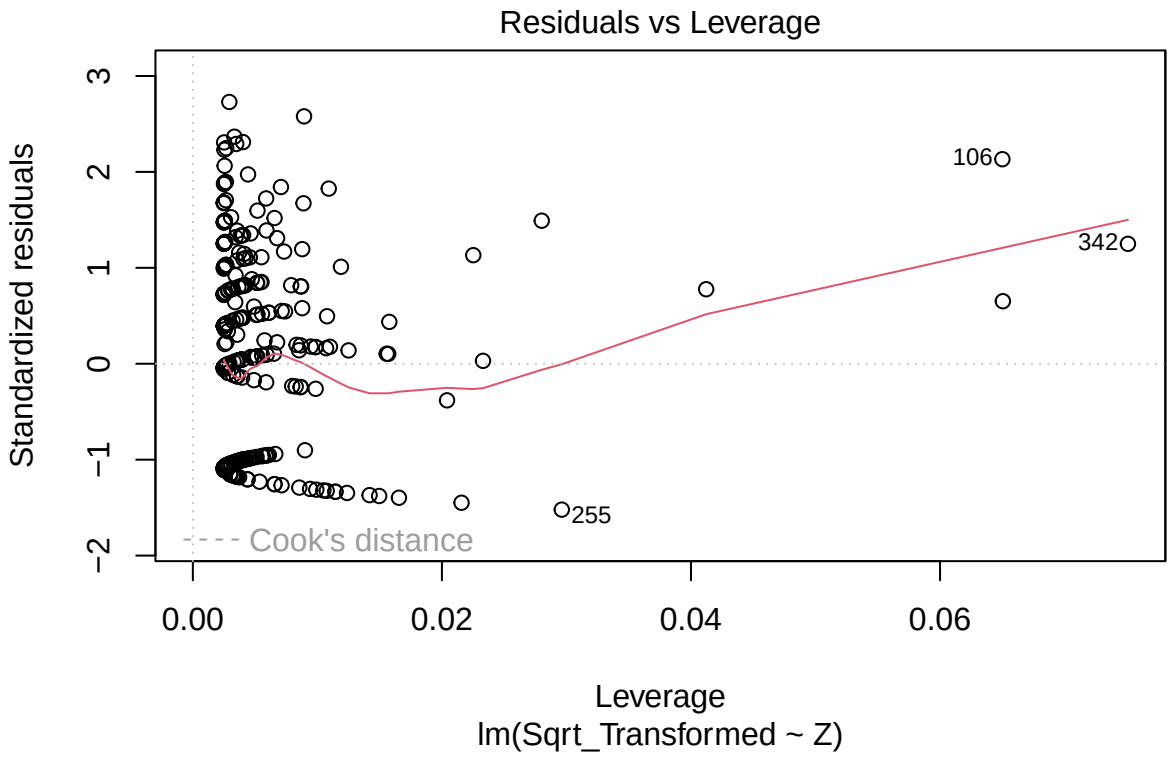



```r
df10$Sqrt_Transformed <- sqrt(df10$average_future_playoff_success_3)
fit_sqrt <- lm(Sqrt_Transformed ~ Z, data = df10)


df10$Inverse_Transformed <- 1 / (df10$average_future_playoff_success_3 + 1e-6)
fit_inverse <- lm(Inverse_Transformed ~ Z, data = df10)

df10$Exp_Transformed <- exp(df10$average_future_playoff_success_3)
fit_exp <- lm(Exp_Transformed ~ Z, data = df10)

df10$Log_Transformed <- log(df10$average_future_playoff_success_3 + 1e-6)
fit_log <- lm(Log_Transformed ~ Z, data = df10)


plot(fit_sqrt)
```

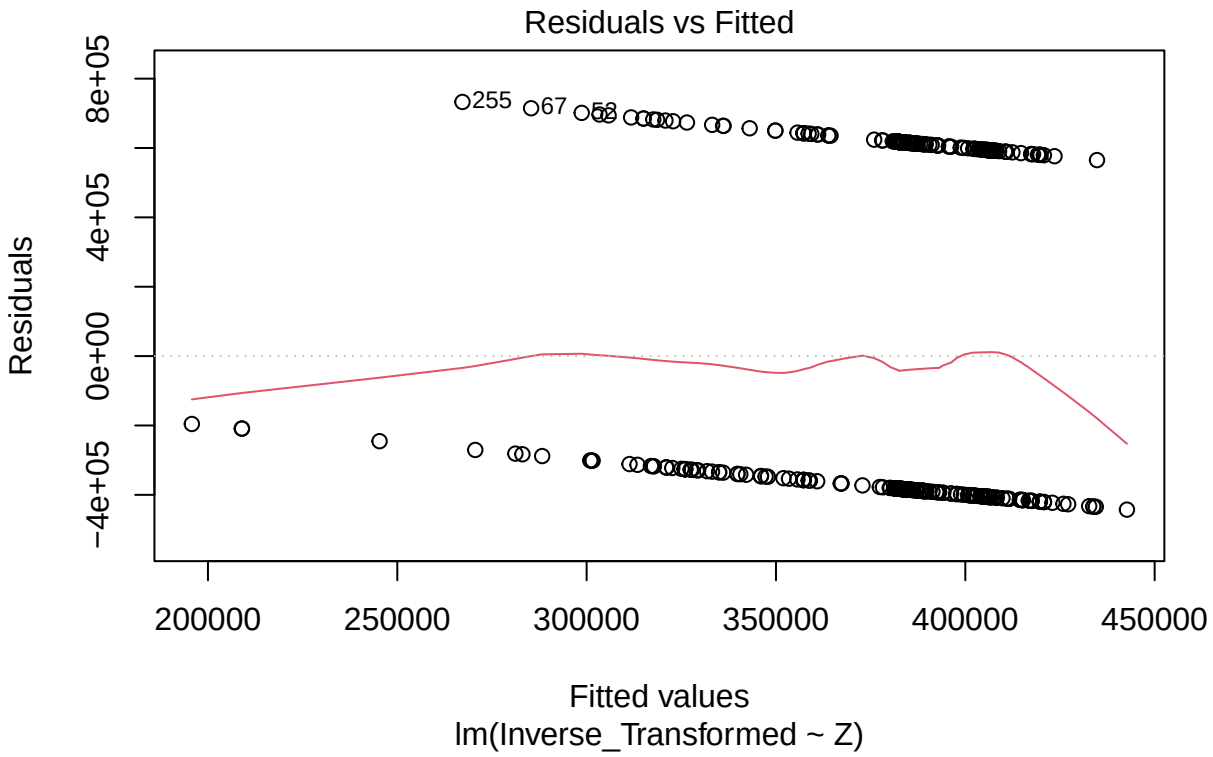

Residuals vs Fitted
Residuals
1.5
1.0
0.5
0.0
−1.0
182
373
17
0.5
0.6
0.7
0.8
0.9
Fitted values
lm(Sqrt_Transformed ~ Z)

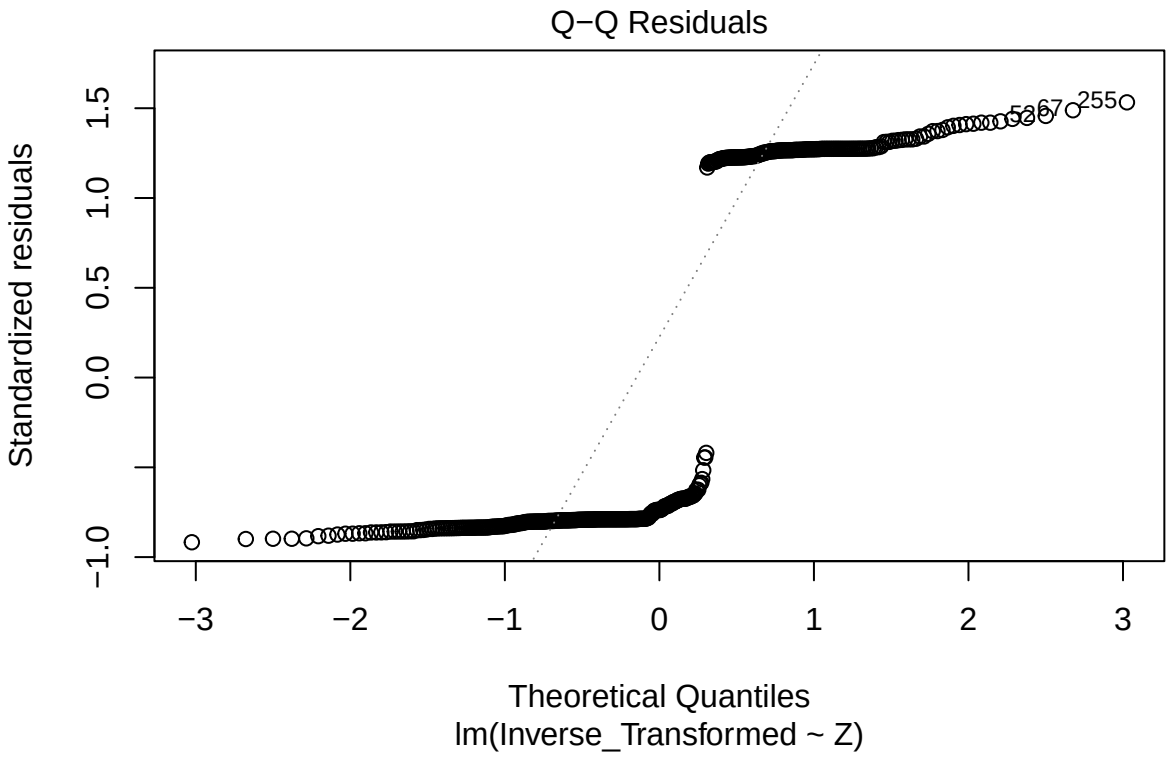

Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(Sqrt_Transformed ~ Z)
373
182
17
−3
−2
−1
0
1
2
3

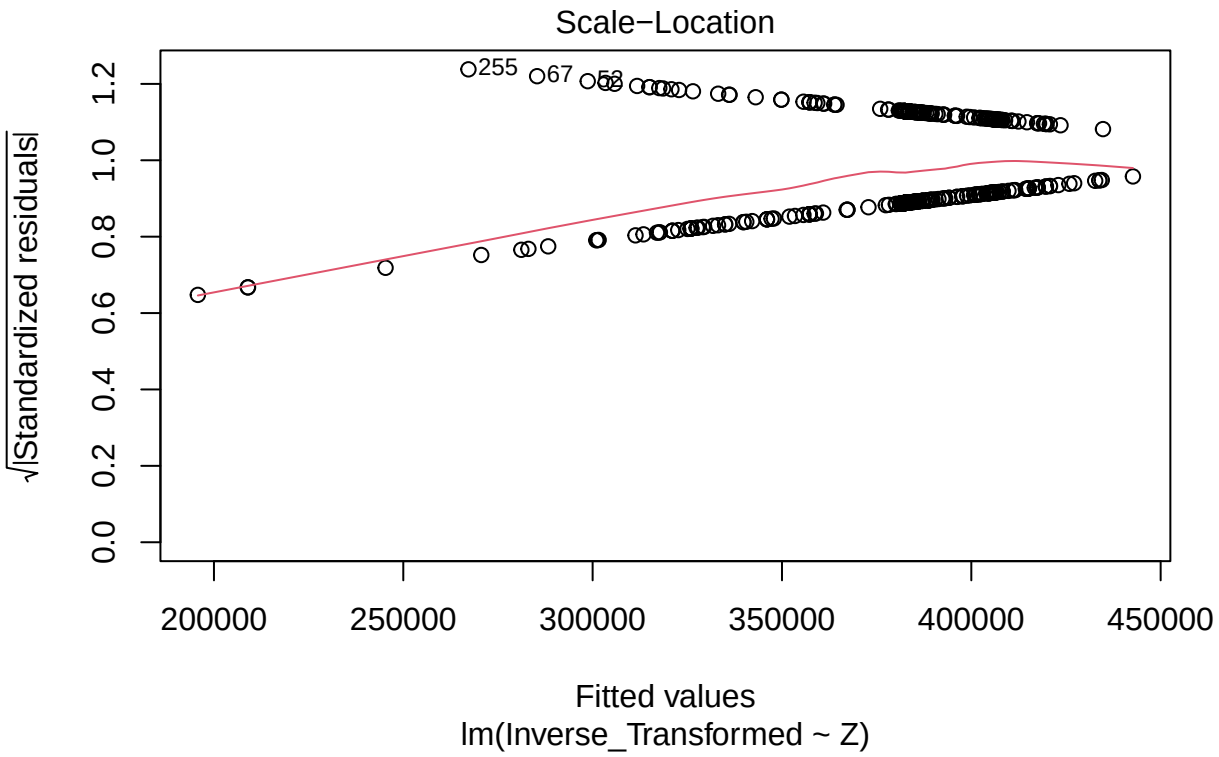

Scale−Location
182
373
17
√|Standardized residuals|
0.0
0.5
1.0
1.5
0.5
0.6
0.7
0.8
0.9
Fitted values
lm(Sqrt_Transformed ~ Z)

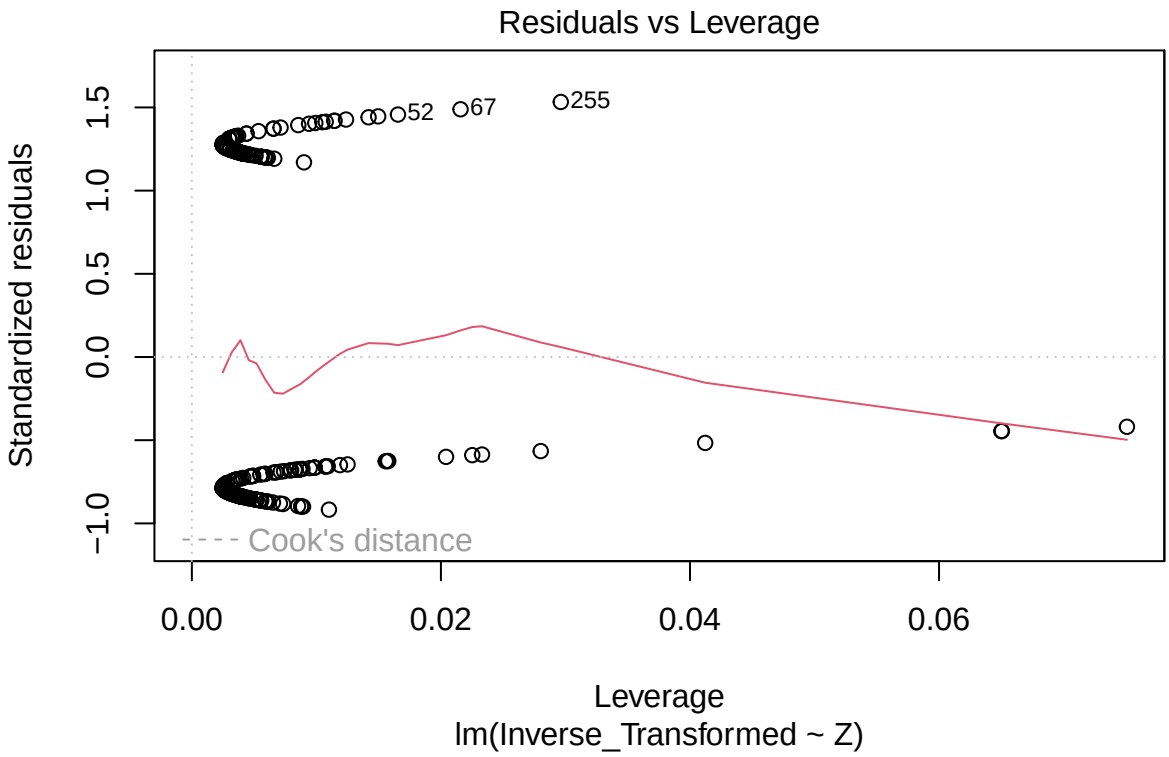



```
plot(fit_inverse)
```

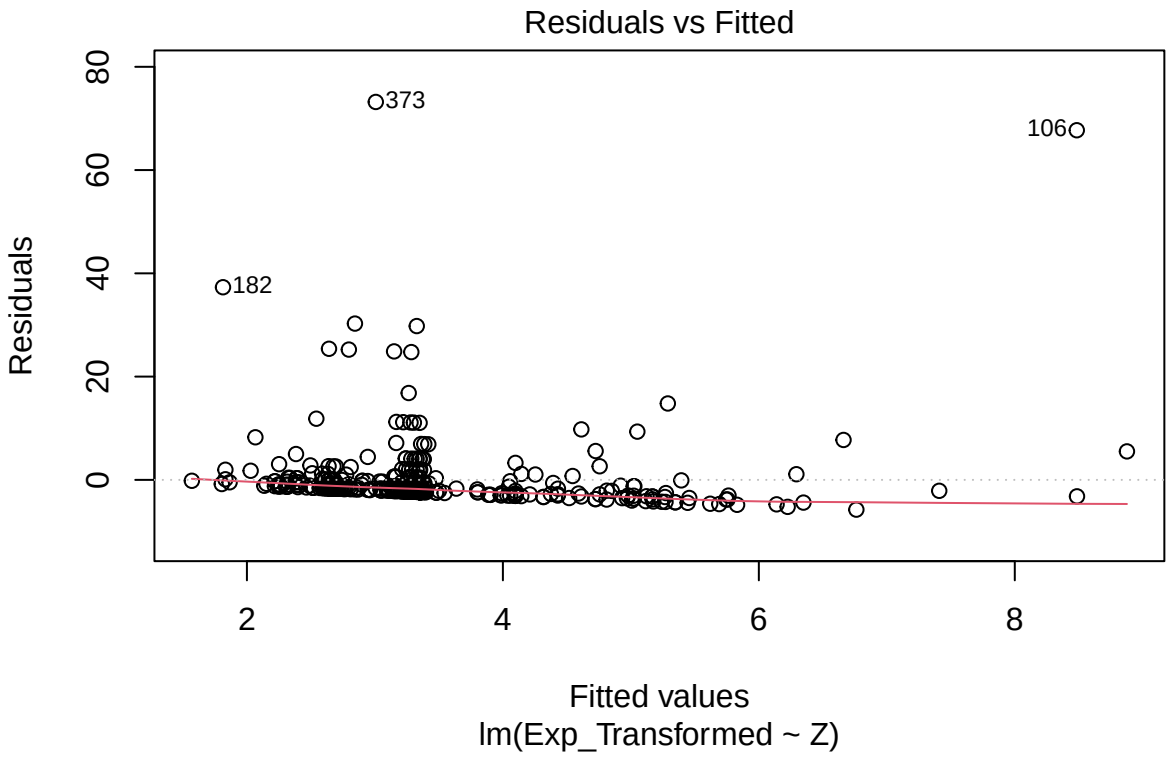

Residuals vs Fitted
Residuals
8e+05
4e+05
0e+00
−4e+05
255
67
200000
250000
300000
350000
400000
450000
Fitted values
lm(Inverse_Transformed ~ Z)

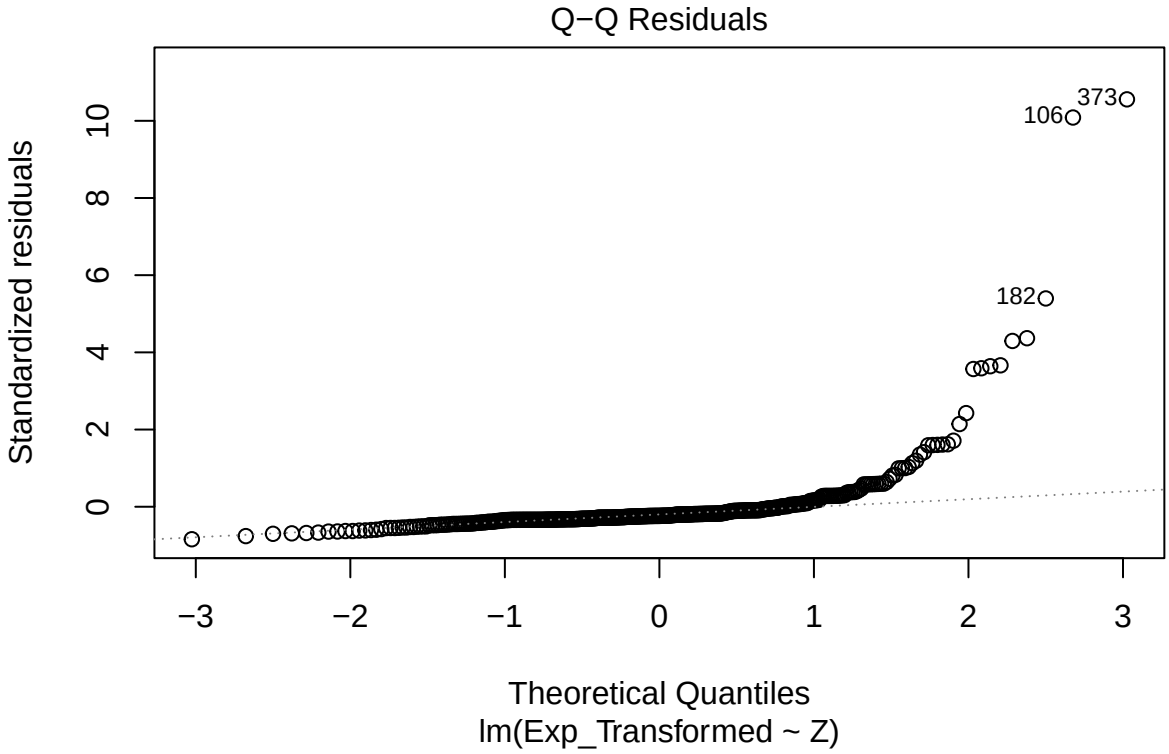

Q−Q Residuals
Standardized residuals
−1.0
0.0
0.5
1.0
1.5
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(Inverse_Transformed ~ Z)
255

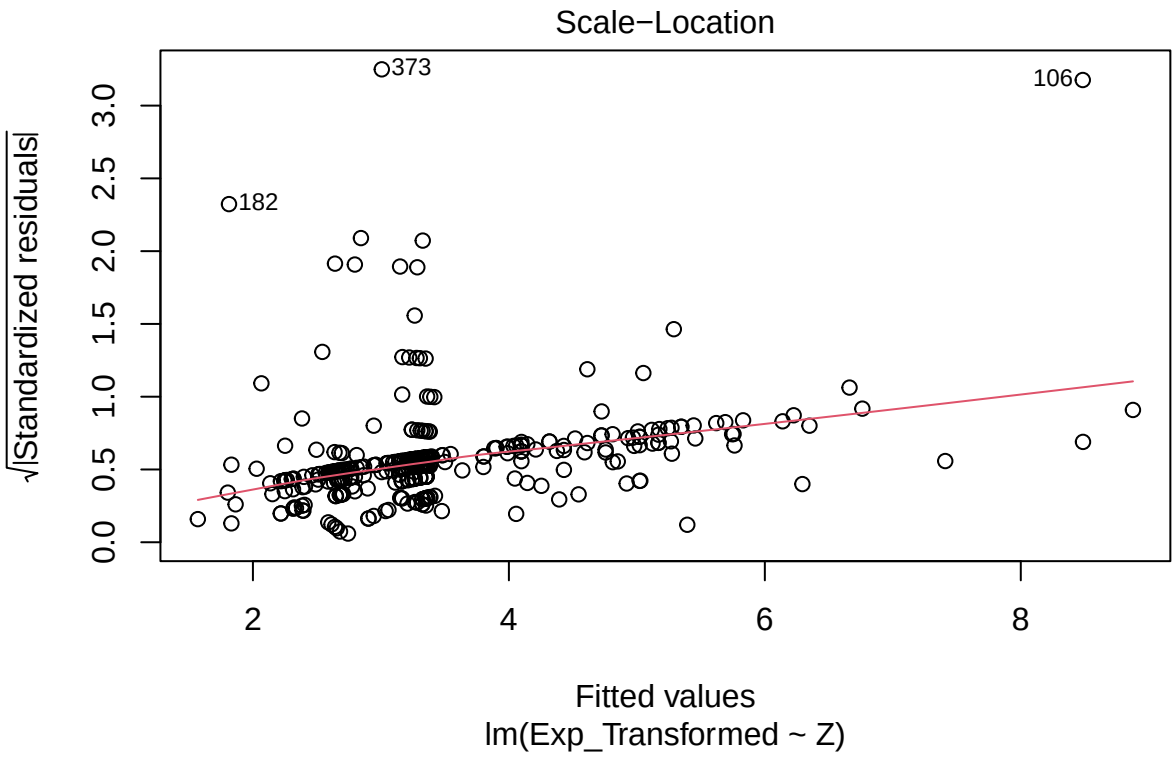

Scale−Location
255
67
52
√|Standardized residuals|
0.0
0.2
0.4
0.6
0.8
1.0
1.2
200000
250000
300000
350000
400000
450000
Fitted values
lm(Inverse_Transformed ~ Z)

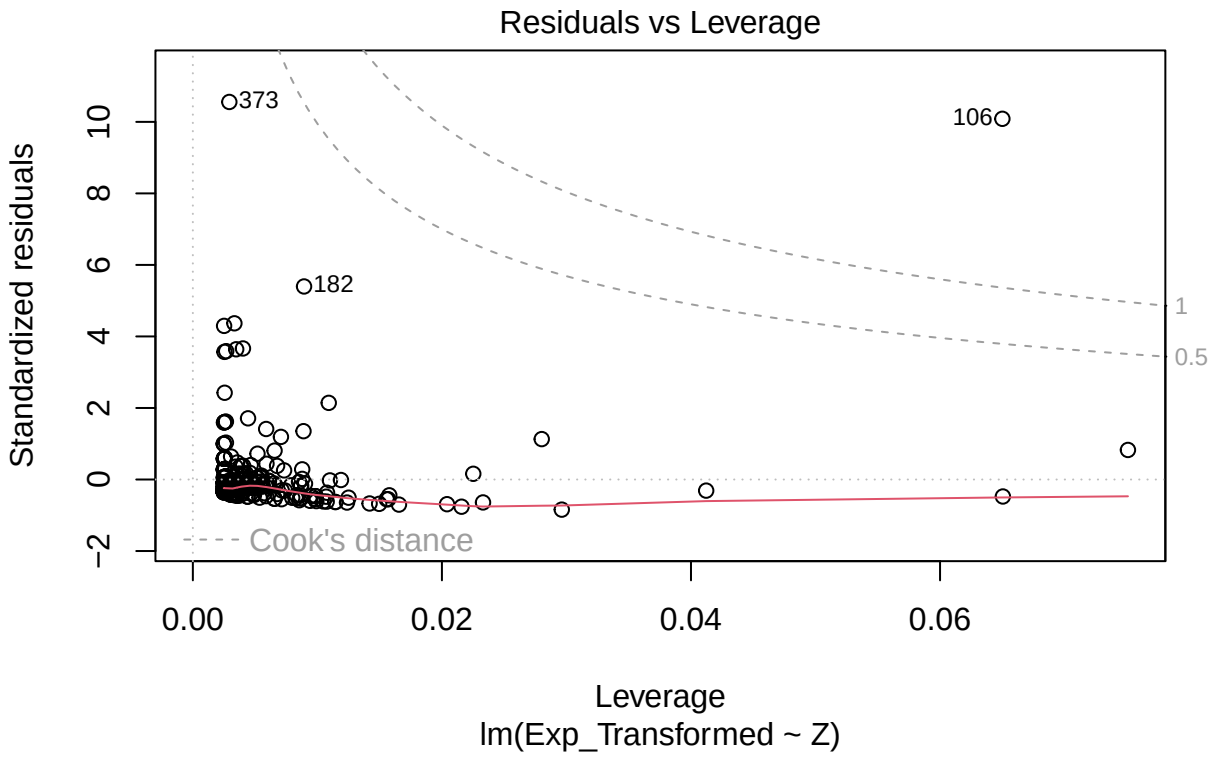

Residuals vs Leverage
Standardized residuals
1.5
1.0
0.5
0.0
−1.0
52
67
255
Cook's distance
0.00
0.02
0.04
0.06
Leverage
lm(Inverse_Transformed ~ Z)

```
plot(fit_exp)
```

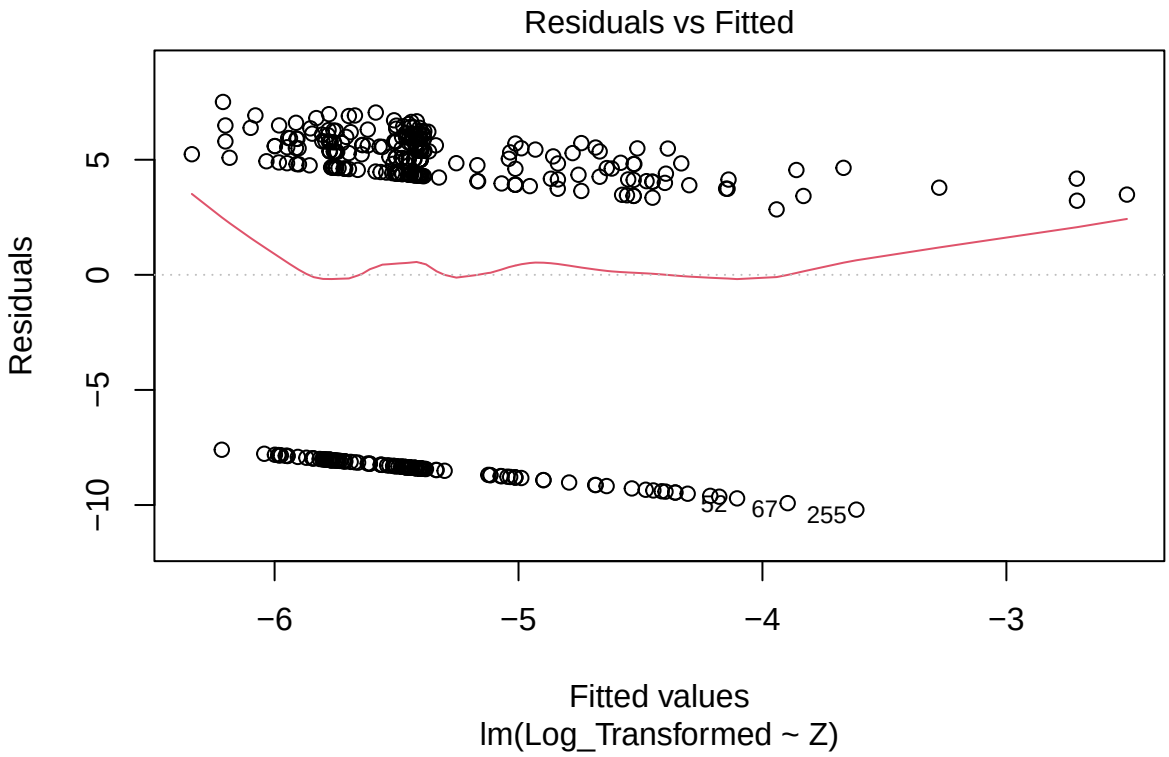

Residuals vs Fitted
Residuals
Fitted values
lm(Exp_Transformed ~ Z)
373
106
182
0
20
40
60
80
2
4
6
8

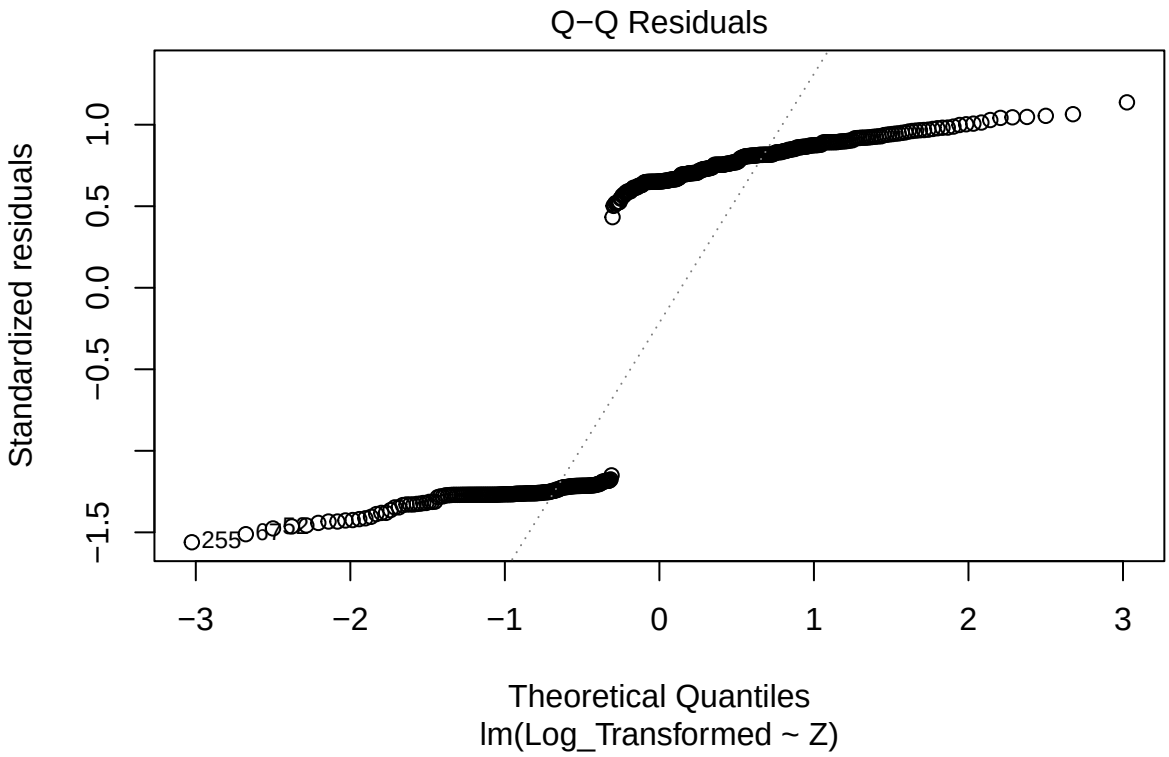

Q−Q Residuals
Standardized residuals
0
2
4
6
8
10
373
106
182
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(Exp_Transformed ~ Z)

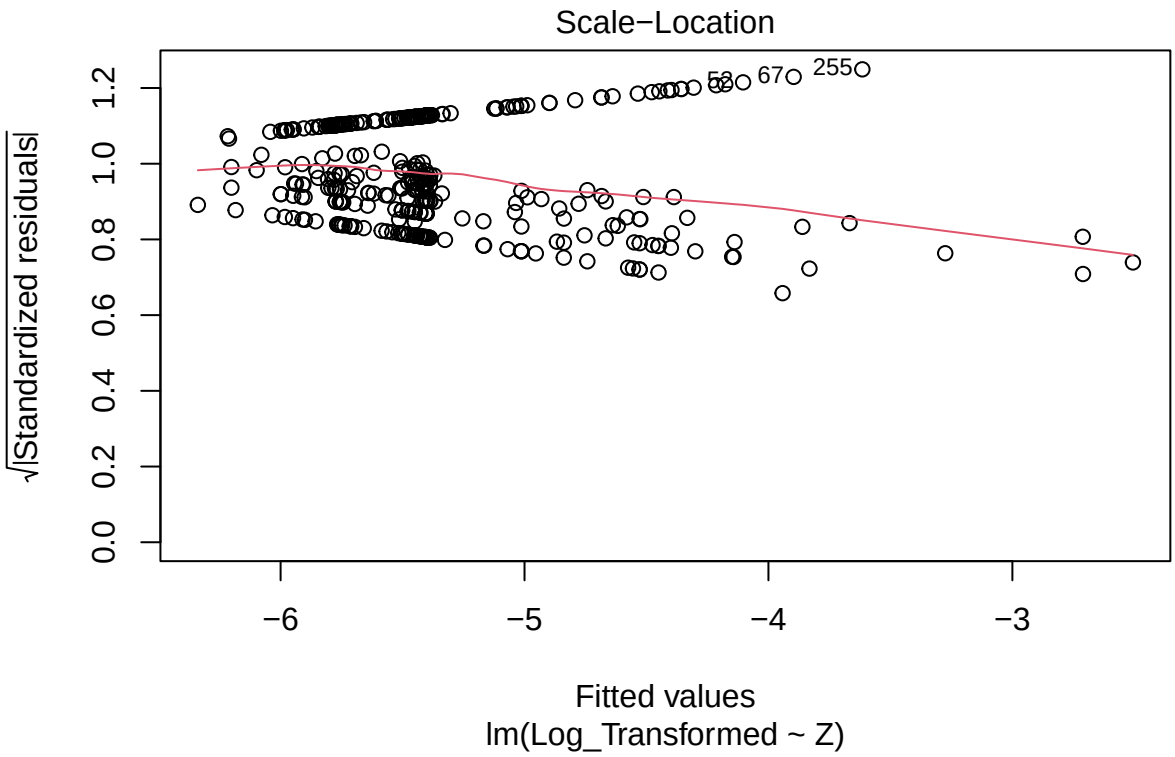

Scale−Location
373
106
182
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
2
4
6
8
Fitted values
lm(Exp_Transformed ~ Z)

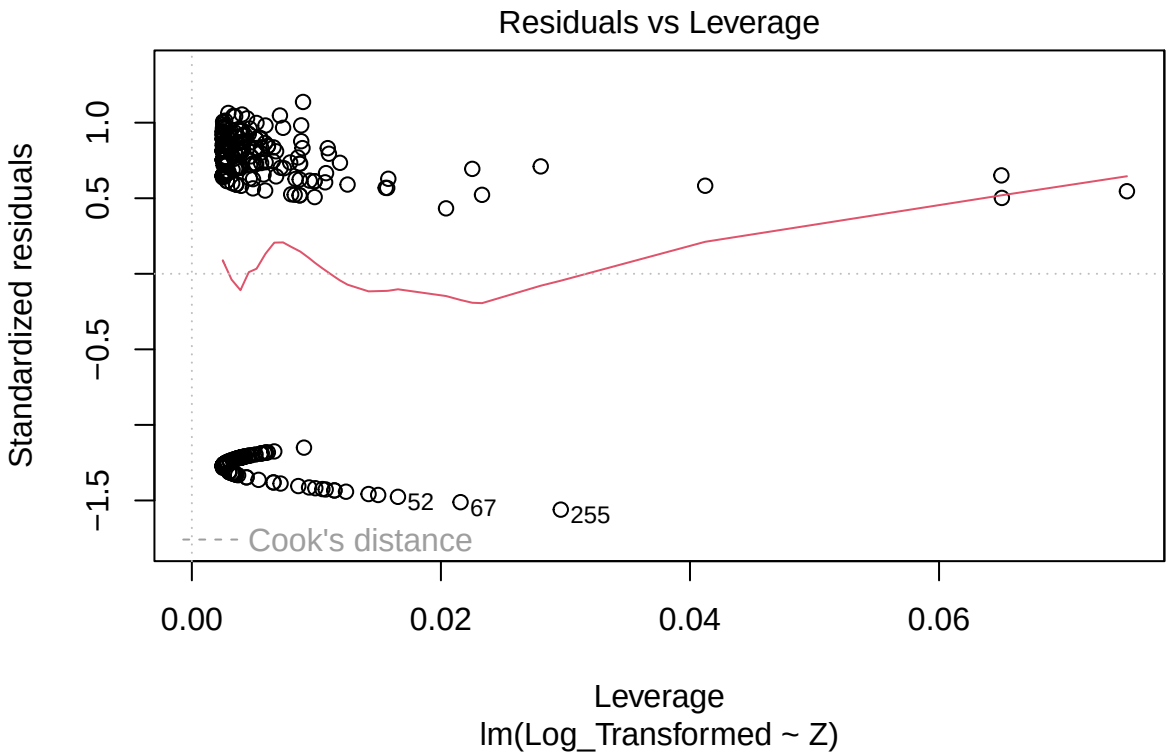

Residuals vs Leverage
Standardized residuals
10
8
6
4
2
0
−2
373
106
182
1
0.5
Cook's distance
0.00
0.02
0.04
0.06
Leverage
lm(Exp_Transformed ~ Z)

```
plot(fit_log)
```


Residuals vs Fitted
Residuals
5
0
−5
−10
−6
−5
−4
−3
52
67
255
Fitted values
lm(Log_Transformed ~ Z)


Q−Q Residuals
Standardized residuals
−1.5
−0.5
0.0
0.5
1.0
255
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(Log_Transformed ~ Z)


Scale−Location
√|Standardized residuals|
0.0
0.2
0.4
0.6
0.8
1.0
1.2
67
255
−6
−5
−4
−3
Fitted values
lm(Log_Transformed ~ Z)



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.