R Exercise: The Pythagorean Win Percentage Exponent in Basketball

Jake Singleton

7/7/2020

Motivation: Baseball

In this quick R Exercise, I will demonstrate how to estimate the exponent k in the Pythagorean expectation formula that was originally invented by sabermetrician Bill James for baseball. The formula is the following:

$\text{Win Pct} = \frac{\text{runs scored}^k}{\text{runs scored}^k + \text{runs allowed}^k}$.

In baseball, James empirically found that $k \approx 2$. Since, further research has been done showing that the optimal k in baseball is 1.83. The goal of this walkthrough is to find the optimal k in the basketball setting–that is, in the formula $\text{Win Pct} = \frac{\text{points scored}^k}{\text{points scored}^k + \text{points allowed}^k}$.

The Data

To be able to find k, we must first obtain the appropriate data. Fortunately, we don't need anything too crazy, as the only four variables we are interested in are wins, losses, points scored, and points allowed. The following code chunk loads in the appropriate data from the last 19 NBA regular seasons (the 2000-2001 season through the 2018-2019 season).

```
# Set up necessary packages and web-scraping
library(tidyverse)
library(rvest)
library(lubridate)
```

```
# Containers
years = seq(2001, 2019, 1)  # years of interest
lst = list()  # empty list we will append to
i = 1  # index

# Loop through each year, scraping appropriate data
for (yr in years) {
  team_stats_url = paste0("https://basketball.realgm.com/nba/team-stats/", yr, "/Totals/Team_Totals/Regular_Seaso
n/gp/desc")  # URL for each team's own stats
  opp_stats_url = paste0("https://basketball.realgm.com/nba/team-stats/", yr, "/Totals/Opponent_Totals/Regular_Se
ason/gp/desc")  # URL for teams' opponents' stats (how we get their points allowed)
  team_w_l_url = paste0("https://basketball.realgm.com/nba/standings/league/", yr)

  # Scrape team's points scored
  df_team = read_html(team_stats_url) %>%
    html_table() %>%
    .[[1]] %>%
    mutate(YR = yr) %>% # Add year column for clarity
    select(Team, PTS, YR)
  #print(head(df_team))

  # Scrape each team's points allowed
  df_opp_pts = read_html(opp_stats_url) %>%
    html_table() %>%
    .[[1]] %>%
    mutate(YR = yr) %>%
    rename(PTS_ALL = PTS) %>%
    select(Team, PTS_ALL, YR)
  #print(head(df_opp_pts))

  # Scrape each team's wins and losses
  df_w_l = read_html(team_w_l_url) %>%
    html_table() %>%
    .[[1]] %>%
    mutate(YR = yr) %>%
    select(Team, W, L, YR)
  df_w_l$Team = gsub("\\s*\\w*$", "", df_w_l$Team)
  #print(head(df_w_l))

  # Perform inner join to combine data
  df_combined = df_team %>%
    inner_join(df_opp_pts, by = c("Team", "YR")) %>%
    inner_join(df_w_l, by = c("Team", "YR"))
  #print(head(df_combined))

  # Append data frame to list, appending the points allowed column
  lst[[i]] = df_combined

  # Increment index
  i = i + 1
}

# Combine data
all_data = bind_rows(lst)

head(all_data)
```

```
##          Team  PTS   YR PTS_ALL  W  L
## 1     Atlanta 7459 2001    7886 25 57
## 2  New Jersey 7552 2001    7966 26 56
## 3    New York 7275 2001    7059 48 34
## 4     Orlando 7992 2001    7911 43 39
## 5 Philadelphia 7763 2001    7412 56 26
## 6     Phoenix 7710 2001    7529 51 31
```

```
tail(all_data)
```

```
##          Team  PTS   YR PTS_ALL  W  L
## 504 Milwaukee 9686 2019    8959 60 22
## 505  Brooklyn 9204 2019    9210 42 40
## 506   Atlanta 9294 2019    9788 29 53
## 507     Miami 8668 2019    8687 39 43
## 508  New York 8575 2019    9330 17 65
## 509   Orlando 8800 2019    8742 42 40
```

Looks great! For each team and year, we have their points scored, `PTS`, points allowed, `PTS_ALL`, their wins `W`, and their losses `L`. We are ready to proceed with some math.

The Math

To estimate k, we will do some algebraic manipulation of our Win Pct equation above to make it linear in k. The idea is that, with a linear equation in k, we can use linear regression to estimate k. Let's do the math. For convenience, let's write points scored as PTS and points allowed as PTS_ALL.

$$
\begin{gathered}
\text{Win Pct} = \frac{\text{W}}{\text{W}+\text{L}} = \frac{\text{PTS}^k}{\text{PTS}^k + \text{PTS\_ALL}^k} \implies \\
\frac{\frac{W}{L}}{\frac{W}{L}+1} = \frac{\frac{\text{PTS}^k}{\text{PTS\_ALL}^k}}{\frac{\text{PTS}^k}{\text{PTS\_ALL}^k}+1} \implies \\
\frac{\text{W}}{\text{L}}\left(\frac{\text{PTS}^k}{\text{PTS\_ALL}^k}+1\right) = \frac{\text{PTS}^k}{\text{PTS\_ALL}^k}\left(\frac{W}{L}+1\right) \implies \\
\frac{\text{W}}{\text{L}} + \frac{\text{W}\cdot\text{PTS}^k}{\text{L}\cdot\text{PTS\_ALL}^k} = \frac{\text{W}\cdot\text{PTS}^k}{\text{L}\cdot\text{PTS\_ALL}^k} + \frac{\text{PTS}^k}{\text{PTS\_ALL}^k} \implies \\
\frac{\text{W}}{\text{L}} = \frac{\text{PTS}^k}{\text{PTS\_ALL}^k} = \left(\frac{\text{PTS}}{\text{PTS\_ALL}}\right)^k. \\
\text{Now, we take the natural log of both sides:} \\
\log\left(\frac{\text{W}}{\text{L}}\right) = k\log\left(\frac{\text{PTS}}{\text{PTS\_ALL}}\right). \\
\text{There we have it--an equation linear in k!}
\end{gathered}
$$

Parameter Estimation

All that's left to do now is simple linear regression. Remember, k is the slope parameter we want to estimate, $\log\left(\frac{\text{W}}{\text{L}}\right)$ is our response variable, and $\log\left(\frac{\text{PTS}}{\text{PTS\_ALL}}\right)$ is the explanatory variable. The code is simple.

```
# Add response and explanatory variables, and do linear regression
all_data = all_data %>%
  mutate(logWratio = log(W / L),  logPTSratio = log(PTS / PTS_ALL))

pyth_fit <- lm(logWratio ~ 0 + logPTSratio, data = all_data)
summary(pyth_fit)
```

```
##
## Call:
## lm(formula = logWratio ~ 0 + logPTSratio, data = all_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.55625 -0.11032 -0.00729  0.09907  0.69779
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## logPTSratio  14.1961     0.1598   88.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1624 on 508 degrees of freedom
## Multiple R-squared:  0.9395, Adjusted R-squared:  0.9394
## F-statistic:  7894 on 1 and 508 DF,  p-value: < 2.2e-16
```

Conclusion

This tells us that $k = 14.2$! We have our estimate, yielding $\text{Win Pct} = \frac{\text{points scored}^{14.2}}{\text{points scored}^{14.2} + \text{points allowed}^{14.2}}$. Before accepting this result, though, we should check and see that it makes sense. Let's write some code that computes pythagorean wins using this formula.

```
# Function that computes number of wins by our pythagorean formula
pythag_wins = function(PTS, PTS_ALL) {
  return(82 * PTS^14.2 / (PTS^14.2 + PTS_ALL^14.2))  # Multiply by 82 since there are 82 games in a season
}

pythag_wins(8000, 8000)
```

```
## [1] 41
```

So, we see that our formula predicts exactly 41 wins (out of 82 games) for a team that totals 8,000 points and 8,000 points allowed over an entire season. This is a winning percentage of 0.500, which intuitively makes sense!

Let's do another check. Specifically, we will compare the root mean square error (RMSE) of the pythagorean residuals to the RMSE of the residuals if we predict win percentage with point differential (PTS - PTS_ALL), which is a natural choice.

```
# RMSE for pythagorean win % residuals
all_data = all_data %>%
  mutate(Win_pct = W / (W + L), Pyth_Win_pct = PTS^14.2 / (PTS^14.2 + PTS_ALL^14.2), pyth_residuals = Win_pct - P
yth_Win_pct)
paste("The RMSE of the Pythagorean residuals is: ", sqrt(mean(all_data$pyth_residuals^2)))
```

```
## [1] "The RMSE of the Pythagorean residuals is:  0.0358380548113744"
```

```
# RMSE for residuals if we predict win % by point differential
all_data = all_data %>%
  mutate(PT_DIFF = PTS - PTS_ALL)
point_diff_fit = lm(Win_pct ~ PT_DIFF, data = all_data)

summary(point_diff_fit)
```

```
##
## Call:
## lm(formula = Win_pct ~ PT_DIFF, data = all_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.106012 -0.023911 -0.000751  0.026274  0.088004
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.991e-01  1.611e-03  309.75   <2e-16 ***
## PT_DIFF     4.031e-04  4.434e-06   90.89   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03635 on 507 degrees of freedom
## Multiple R-squared:  0.9422, Adjusted R-squared:  0.9421
## F-statistic:  8261 on 1 and 507 DF,  p-value: < 2.2e-16
```

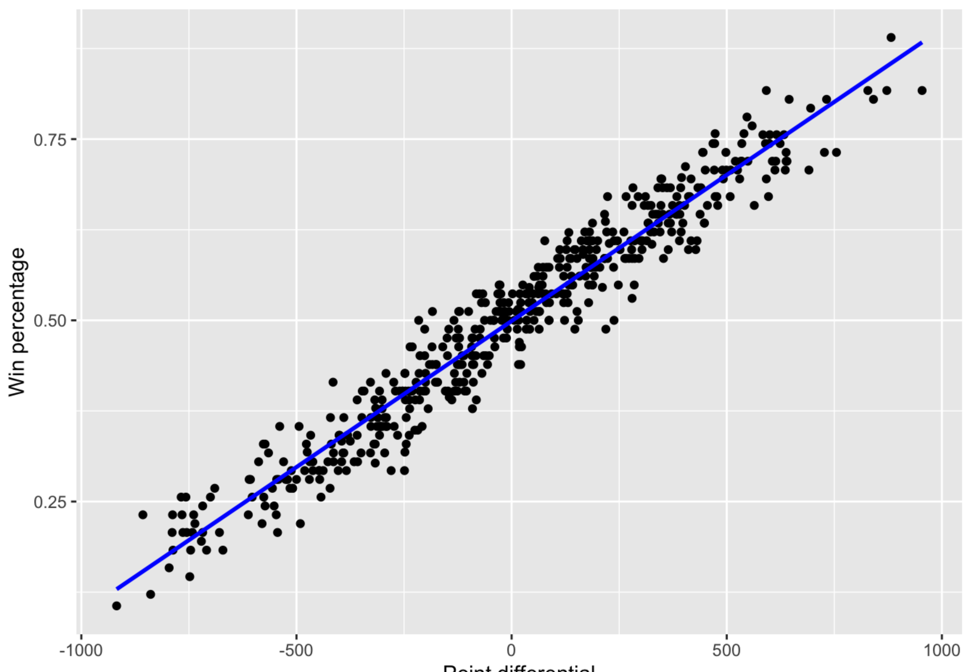
```
paste("The RMSE of the point differential fit is: ", sqrt(mean(point_diff_fit$residuals^2)))
```

```
## [1] "The RMSE of the point differential fit is:  0.0362773455161389"
```

So, the RMSE of the pythagorean fit is slightly better than that of the point differential fit. And, if you're curious, note that the point differential fit is a good model, as we can see in the following plot:

```
# Plot
ggplot(all_data, aes(PT_DIFF, Win_pct)) +
  geom_point() +
  geom_smooth(method = "lm", se = F, color = "blue") +
  scale_x_continuous("Point differential") +
  scale_y_continuous("Win percentage")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



That's it! In this R Exercise, we've walked through how one can estimate the parameter k in the formula for Pythagorean win expectation. In addition, we've seen that the Pythagorean model makes sense and is in fact better than the point differential model. Finally, if you'd like to try to do this yourself, note that it can be done for NFL and NHL data in addition to MLB and NBA. Thank you for reading my first R Exercise!

References

1. The Data: https://basketball.realgm.com/
2. Intro to Pythagorean Expectation: https://en.wikipedia.org/wiki/Pythagorean_expectation
3. Pythagorean Expectation for Baseball: 4.4 The Pythagorean Formula for Winning Percentage; Marchi, Max. Analyzing Baseball Data with R, Second Edition (Chapman & Hall/CRC The R Series) (p. 99). CRC Press. Kindle Edition.