2 Pythagorean expectation and MLB

17 July 2020

Pythagorean Expectation and the Indian Premier League

The Indian Premier League (IPL) is the biggest cricket competition in the world, which has all of the world's best players in an eight week tournament involving eight teams playing sixty games in total. Each team plays every other team, once at home and then away, and the competition finishes with the four best teams competing in semi-finals and then a final.

Cricket, like baseball, is a bat and ball game, where teams score runs and the team scoring the highest number of runs is the winner. There are, of course, many differences, but statistically speaking, we can generate the same Pythagorean statistic that we generated for baseball. Our data here is derived from the competition that took place in 2018.

The IPL is played in the T20 format, in which each team has up to 120 balls to score as many runs as they can (the game takes less than three hours to complete). One difference from baseball is that runs are much easier to score - in the IPL an average score is 170 runs - and outs (wickets) are much more costly - each team has only ten outs(called wickets) in the entire game, and if you run out of wickets before the 120 balls have been bowled (pitched) then your inning is over.

With this background, let's construct the Pythagorean Expectation for the IPL in 2018.

```r
# As with the previous notebook, we first important the packages
# we will need to process the data.
library("readxl",quietly = TRUE)
library("tidyverse",quietly = TRUE)
```

```r
# Now we import the data, which comes in the form of
# a list of games played in the 2018 season.
# We print out the list of variables names in the dataframe
IPL18  <- read_excel("IPL2018teams.xlsx")
names(IPL18)
```

```
##  [1] "scorecard_id"              "start_date"
##  [3] "phase"                     "name"
##  [5] "home_team"                 "away_team"
##  [7] "toss_winner"               "toss_decision"
##  [9] "inn1team"                  "innings1"
## [11] "wickets1"                  "overs1"
## [13] "closure1"                  "innings2"
## [15] "wickets2"                  "overs2"
## [17] "closure2"                    "adjusted_target_indicator"
## [19] "adjusted_target"             "team1_overs"
## [21] "team2_overs"                 "mom_player_id"
## [23] "mom_player"                  "scoring_status"
## [25] "result_type"                 "result_margin"
## [27] "winning_team"
```

```
# We can see what our dataframe looks like simply by typing its name:

IPL18
```

```
## # A tibble: 60 x 27
##    scorecard_id start_date          phase name  home_team away_team
##           <dbl> <dttm>              <chr> <chr> <chr>     <chr>
##  1      1056637 2018-04-07 00:00:00 <NA>  Wank~ Mumbai I~ Chennai ~
##  2      1056638 2018-04-08 00:00:00 <NA>  Punj~ Kings XI~ Delhi Da~
##  3      1056639 2018-04-08 00:00:00 <NA>  Eden~ Kolkata ~ Royal Ch~
##  4      1056640 2018-04-09 00:00:00 <NA>  Raji~ Sunrisers Rajastha~
##  5      1056641 2018-04-10 00:00:00 <NA>  MA C~ Chennai ~ Kolkata ~
##  6      1056642 2018-04-11 00:00:00 <NA>  Sawa~ Rajastha~ Delhi Da~
##  7      1056643 2018-04-12 00:00:00 <NA>  Raji~ Sunrisers Mumbai I~
##  8      1056644 2018-04-13 00:00:00 <NA>  M Ch~ Royal Ch~ Kings XI~
##  9      1056645 2018-04-14 00:00:00 <NA>  Wank~ Mumbai I~ Delhi Da~
## 10      1056646 2018-04-14 00:00:00 <NA>  Eden~ Kolkata ~ Sunrisers
## # ... with 50 more rows, and 21 more variables: toss_winner <chr>,
## #   toss_decision <chr>, inn1team <chr>, innings1 <dbl>, wickets1 <dbl>,
## #   overs1 <dbl>, closure1 <chr>, innings2 <dbl>, wickets2 <dbl>,
## #   overs2 <dbl>, closure2 <chr>, adjusted_target_indicator <chr>,
## #   adjusted_target <dbl>, team1_overs <dbl>, team2_overs <dbl>,
## #   mom_player_id <dbl>, mom_player <chr>, scoring_status <chr>,
## #   result_type <chr>, result_margin <dbl>, winning_team <chr>
```

```
# This cell compelete a number tasks. First we identify when the home team is
# the winning team, and when the visiting team
# is the winner. Next we identify the runs scored by the home team and the away team
# (note: unlike baseball, where there are nine innings for each team,
# in T20 cricket each team gets only one inning, and once the first
# completes its inning, the opposing
# team has its inning). Finally, we include a counter which we can add up
# to give total number of games for each team.

IPL18[,'hwin']= ifelse(IPL18$home_team==IPL18$winning_team,1,0)
IPL18[,'awin']= ifelse(IPL18$away_team==IPL18$winning_team,1,0)

IPL18[,'htruns']= ifelse(IPL18$home_team==IPL18$inn1team,
                         IPL18$innings1,IPL18$innings2)
```

```r
IPL18[,'atruns']= ifelse(IPL18$away_team==IPL18$inn1team,
                         IPL18$innings1,IPL18$innings2)
IPL18[,'count']=1
```

```r
# Now we use a .groupby command to aggregate the performance of home teams
# during the season. Compare back to the MLB notebook
# to see how similar the commands are.

IPLhome  <-  IPL18 %>% group_by(home_team)%>%
            dplyr::summarise(count = sum(count),
                             hwin= sum(hwin),
                             htruns = sum(htruns),
                             atruns = sum(atruns)
                           )%>%
                           ungroup()%>%
                           rename(team = home_team,
                                  htrunsh = htruns,
                                  atrunsh = atruns,
                                  Ph = count)
IPLhome
```

```
## # A tibble: 8 x 5
##   team                           Ph  hwin htrunsh atrunsh
##   <chr>                       <dbl> <dbl>   <dbl>   <dbl>
## 1 Chennai Super Kings             9     8    1577    1486
## 2 Delhi Daredevils                7     4    1258    1122
## 3 Kings XI Punjab                 7     4    1188    1202
## 4 Kolkata Knight Riders           9     5    1468    1417
## 5 Mumbai Indians                  7     3    1194    1171
## 6 Rajasthan Royals                7     5    1120     994
## 7 Royal Challengers Bangalore     7     4    1298    1286
## 8 Sunrisers                       7     5    1070    1050
```

```r
# Now we aggregate the performance of away teams in a different df.

IPLaway <-  IPL18 %>% group_by(away_team)%>%
            dplyr::summarise(count = sum(count),
                             awin= sum(awin),
                             htruns = sum(htruns),
                             atruns = sum(atruns)
                           )%>%
                           ungroup()%>%
                           rename(team = away_team,
                                  htrunsa = htruns,
                                  atrunsa = atruns,
                                   Pa = count)
IPLaway
```

```
## # A tibble: 8 x 5
##   team                           Pa  awin htrunsa atrunsa
##   <chr>                       <dbl> <dbl>   <dbl>   <dbl>
## 1 Chennai Super Kings             7     3    1264    1232
## 2 Delhi Daredevils                7     1    1265    1085
## 3 Kings XI Punjab                 7     2    1124    1022
## 4 Kolkata Knight Riders           7     4    1326    1291
## 5 Mumbai Indians                  7     3    1111    1186
## 6 Rajasthan Royals                8     2    1362    1237
## 7 Royal Challengers Bangalore     7     2    1097    1024
## 8 Sunrisers                      10     5    1624    1651
```

```
# Now we merge the two dfs to obtain a full record for each team across the season.

IPL18 <- merge(x=IPLhome,y=IPLaway,by=c('team'))
IPL18
```

```
##                          team Ph hwin htrunsh atrunsh Pa awin htrunsa
## 1         Chennai Super Kings  9    8    1577    1486  7    3    1264
## 2            Delhi Daredevils  7    4    1258    1122  7    1    1265
## 3             Kings XI Punjab  7    4    1188    1202  7    2    1124
## 4       Kolkata Knight Riders  9    5    1468    1417  7    4    1326
## 5              Mumbai Indians  7    3    1194    1171  7    3    1111
## 6            Rajasthan Royals  7    5    1120     994  8    2    1362
## 7 Royal Challengers Bangalore  7    4    1298    1286  7    2    1097
## 8                   Sunrisers  7    5    1070    1050 10    5    1624
##   atrunsa
## 1    1232
## 2    1085
## 3    1022
## 4    1291
## 5    1186
## 6    1237
## 7    1024
## 8    1651
```

```
# We now aggregate the home and away data for wins, games played and runs

IPL18[,'W'] = IPL18[,'hwin'] + IPL18[,'awin']
IPL18[,'G'] = IPL18[,'Ph'] + IPL18[,'Pa']
IPL18[,'R'] = IPL18[,'htrunsh'] + IPL18[,'atrunsa']
IPL18[,'RA'] = IPL18[,'atrunsh'] + IPL18[,'htrunsa']
IPL18
```

```
##                          team Ph hwin htrunsh atrunsh Pa awin htrunsa
## 1         Chennai Super Kings  9    8    1577    1486  7    3    1264
## 2            Delhi Daredevils  7    4    1258    1122  7    1    1265
## 3             Kings XI Punjab  7    4    1188    1202  7    2    1124
## 4       Kolkata Knight Riders  9    5    1468    1417  7    4    1326
## 5              Mumbai Indians  7    3    1194    1171  7    3    1111
## 6            Rajasthan Royals  7    5    1120     994  8    2    1362
## 7 Royal Challengers Bangalore  7    4    1298    1286  7    2    1097
## 8                  Sunrisers  7    5    1070    1050 10    5    1624
##   atrunsa  W  G    R   RA
## 1    1232 11 16 2809 2750
## 2    1085  5 14 2343 2387
## 3    1022  6 14 2210 2326
## 4    1291  9 16 2759 2743
## 5    1186  6 14 2380 2282
## 6    1237  7 15 2357 2356
## 7    1024  6 14 2322 2383
## 8    1651 10 17 2721 2674
```

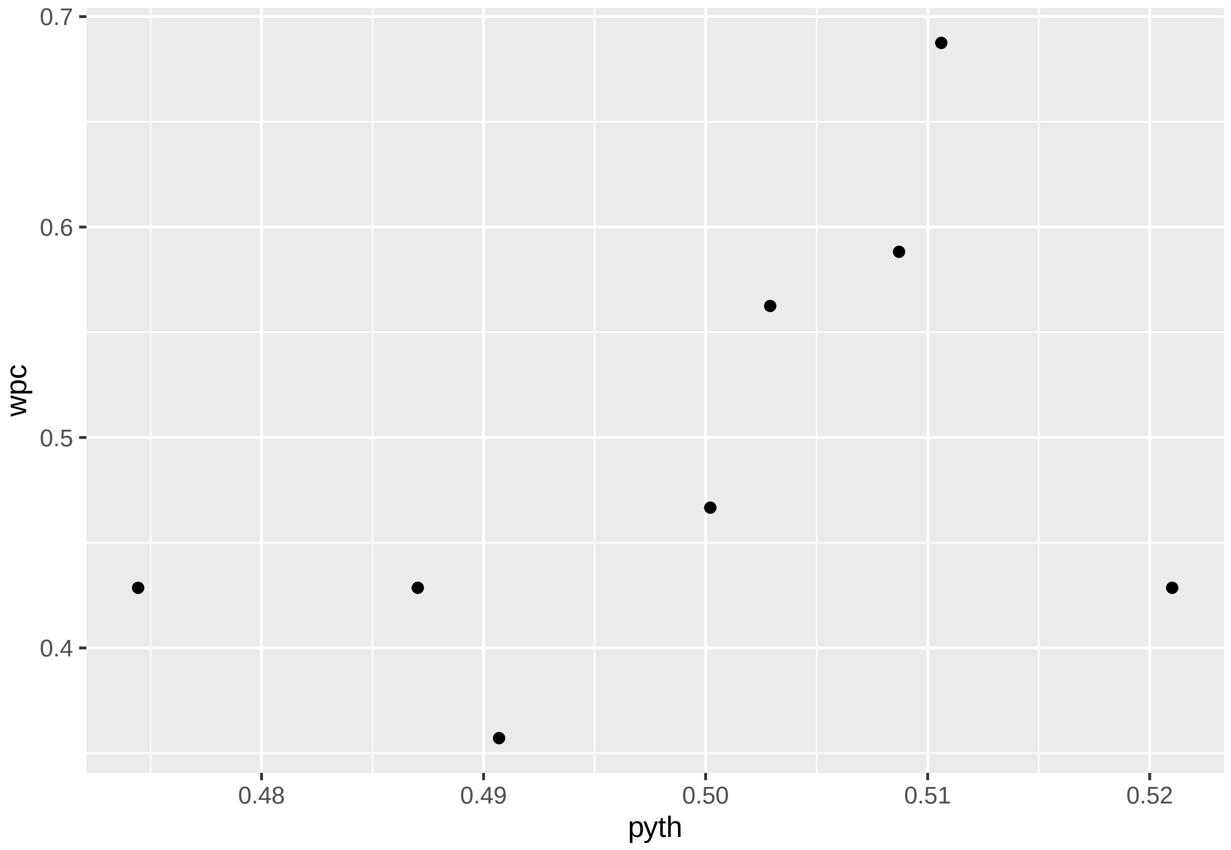
```
# The last step in organizing the data is to create variables for win percentage (wpc)

IPL18[,'wpc'] = IPL18[,'W']/IPL18[,'G']
IPL18[,'pyth'] = IPL18[,'R']**2/(IPL18[,'R']**2 + IPL18[,'RA']**2)
IPL18
```

```
##                          team Ph hwin htrunsh atrunsh Pa awin htrunsa
## 1         Chennai Super Kings  9    8    1577    1486  7    3    1264
## 2            Delhi Daredevils  7    4    1258    1122  7    1    1265
## 3             Kings XI Punjab  7    4    1188    1202  7    2    1124
## 4       Kolkata Knight Riders  9    5    1468    1417  7    4    1326
## 5              Mumbai Indians  7    3    1194    1171  7    3    1111
## 6            Rajasthan Royals  7    5    1120     994  8    2    1362
## 7 Royal Challengers Bangalore  7    4    1298    1286  7    2    1097
## 8                  Sunrisers  7    5    1070    1050 10    5    1624
##   atrunsa  W  G    R   RA       wpc      pyth
## 1    1232 11 16 2809 2750 0.6875000 0.5106122
## 2    1085  5 14 2343 2387 0.3571429 0.4906985
## 3    1022  6 14 2210 2326 0.4285714 0.4744435
## 4    1291  9 16 2759 2743 0.5625000 0.5029080
## 5    1186  6 14 2380 2282 0.4285714 0.5210117
## 6    1237  7 15 2357 2356 0.4666667 0.5002122
## 7    1024  6 14 2322 2383 0.4285714 0.4870372
## 8    1651 10 17 2721 2674 0.5882353 0.5087111
```

```
# Having prepared the data, we are now ready to examine it. First,
# we generate and xy plot use the Seaborn package.
```

```r
# This illustrates nicely the close correlation between win percentage
# and the Pythagorean Expectation.
ggplot(data = IPL18,aes(x = pyth,y = wpc )) + geom_point()
```



Self test

run ggplot again, but this time write y= W instead of y= wpc. What do you find? Does it make a difference?

Running a regression

We now run the same regression as we did for the MLB data:

wpc = Intercept + coef x pyth

This time, while coefficient on pyth is positive - implying that a higher Pythagorean Expectation leads to a large win percentage, the standard error is also very large, and the t statistic of 1.353 implies a p-value of 0.225- well above the usual threshold of 0.050, which means that the coefficient estimate is in fact insignificantly different from zero.

```r
# Finally we generate a regression.
```

```r
pyth_lm = lm(formula = 'wpc ~ pyth', data = IPL18)
pyth_lm %>% summary()
```

```
##
## Call:
## lm(formula = "wpc ~ pyth", data = IPL18)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.141475 -0.048428  0.001577  0.058042  0.154395
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.281      1.312  -0.976    0.367
## pyth           3.552      2.626   1.353    0.225
##
## Residual standard error: 0.1032 on 6 degrees of freedom
## Multiple R-squared:  0.2338, Adjusted R-squared:  0.1061
## F-statistic:  1.83 on 1 and 6 DF,  p-value: 0.2248
```

Self test

Run the regression above but instead write 'wpc ~ W' instead of 'wpc ~ pyth' in the line starting pyth_lm. What difference does this make?

Conclusion

Why did the Pythagorean model produce a good fit for the baseball data but not for the cricket data? An obvious explanation is that there is some difference between the two sports which makes the model appropriate for one but not the other. For example, in cricket, the team batting second need only score one more run than the opponent to win, and so the inning ends if it reaches this milestone. If the team batting second is the winning team, then the gap in the scores will be small. However, if the team batting first can get all ten wickets cheaply, then the gap in scores could be very large. In our data the average runs difference when the team batting second won was 2, and when the team batting first won was 30. This might explain why the Pythagorean Expectation is not a good guide to winning in the IPL.