Chapter 4 Relation between Rins and Wins

Tomoka Takegaki

02/12/2021

4.1 Introduction

This chapter explores relationship between runs and wins. Understanding this relationship is a critical step towards answering questions about player's value. In fact, it is possible to estimate player's contributions in terms of runs.

Setting environment

```
# install.packages("tidyverse")
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.2

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(Lahman)
```

```
## Warning: package 'Lahman' was built under R version 4.1.2
```

Suppose that one is interested in relating the proportion of wins with with the runs scored and runs allowed for all of the seasons.

```
# Team performance since 2001
my_teams <- Teams %>%
    filter(yearID>2000) %>%
    select(teamID,yearID,lgID,G,W,L,R,RA)
my_teams %>% tail()
```

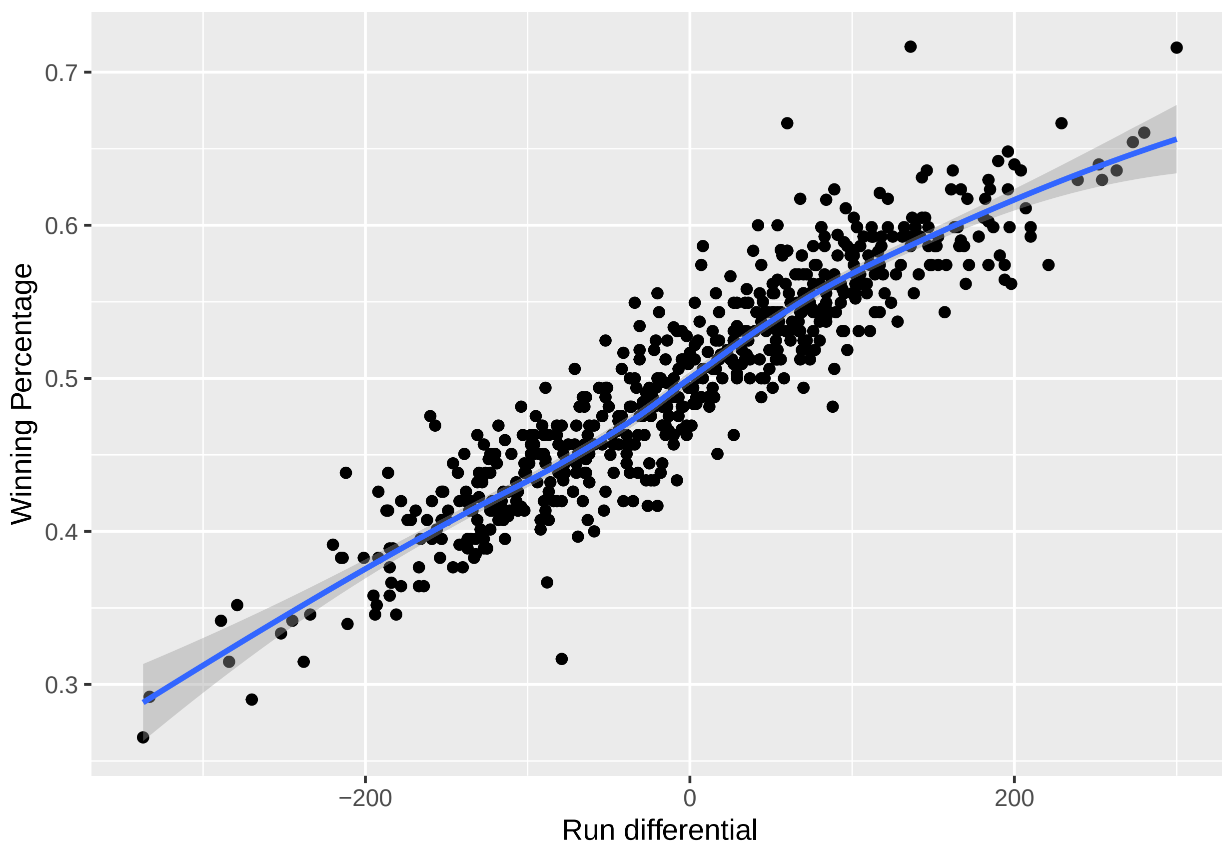
```
##     teamID yearID lgID  G  W  L   R  RA
## 595    SFN   2020   NL 60 29 31 299 297
## 596    SLN   2020   NL 58 30 28 240 229
## 597    TBA   2020   AL 60 40 20 289 229
## 598    TEX   2020   AL 60 22 38 224 312
## 599    TOR   2020   AL 60 32 28 302 312
## 600    WAS   2020   NL 60 26 34 293 301
```

```
my_teams %>%
    mutate(RD = R - RA, Wpct = W / (W+L)) ->my_teams
```

Scatter plot for Run differential vs Winning percentage

```
ggplot(my_teams,aes(x=RD,y=Wpct))+
    geom_point() + geom_smooth()+
    scale_x_continuous("Run differential")+
    scale_y_continuous("Winning Percentage") -> run_diff
run_diff
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



4.3 Linear Regression

To predict a team's winning percentage using runs scored and runs allowed is with linear regression.

Wpct = a + b * RD + e,

lm is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance

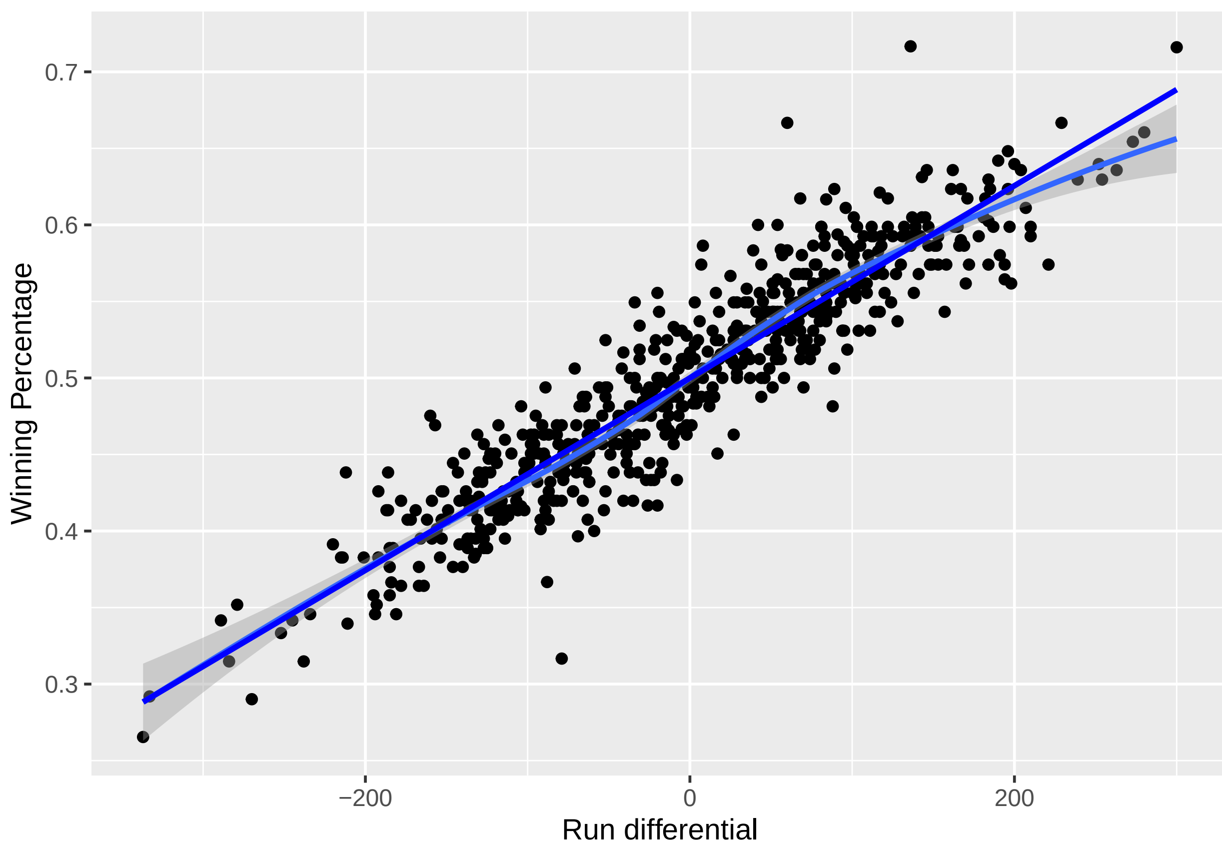
```
linfit <- lm(Wpct ~RD, data=my_teams)
linfit
```

```
##
## Call:
## lm(formula = Wpct ~ RD, data = my_teams)
##
## Coefficients:
## (Intercept)           RD
##    0.499984     0.000628
```

Once we have a fitted model, we use the function augment() from broom package to calculate the predicted values from the model. As well as the residuals, which measure the difference between the response value and the fitted value.

```
run_diff +
    geom_smooth(method="lm", se=FALSE, color="blue")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

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



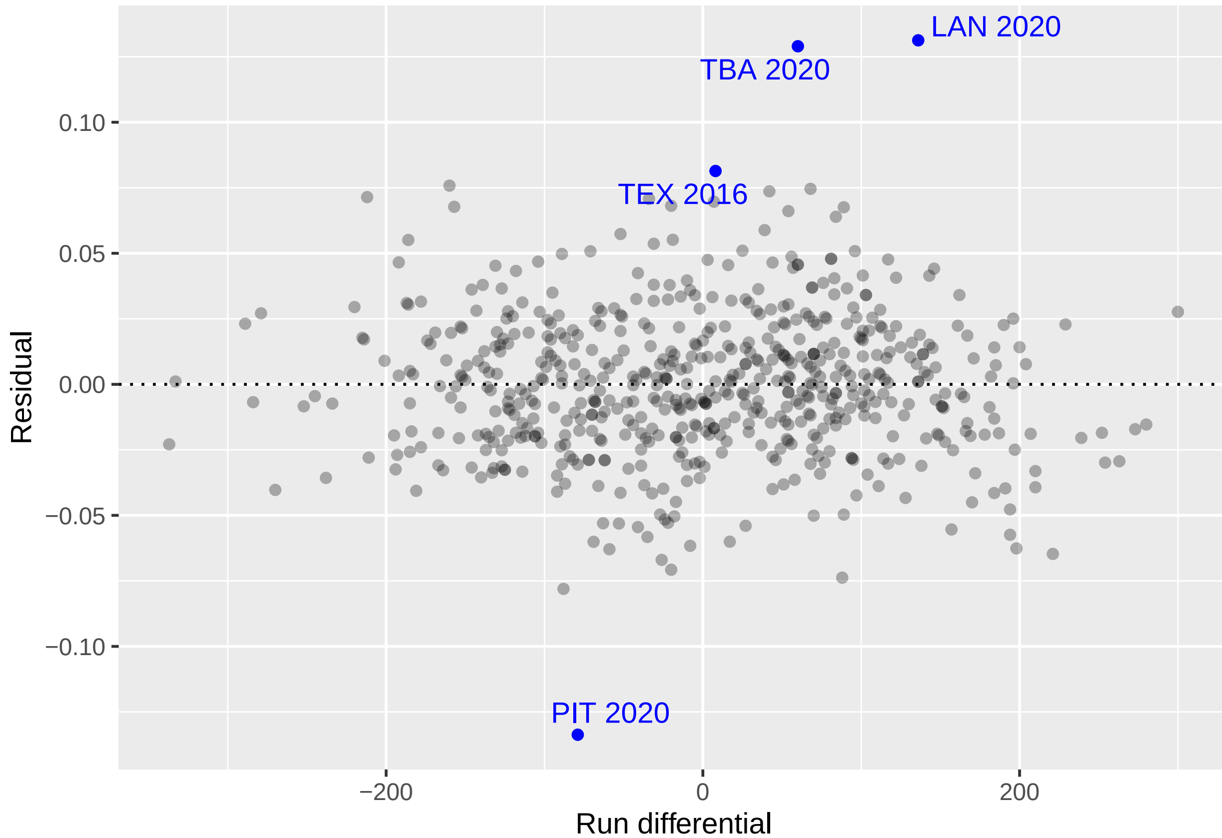
```r
library(broom)

my_teams_aug <- augment(linfit, data = my_teams)

base_plot <- ggplot(my_teams_aug, aes(x=RD, y=.resid)) +
    geom_point(alpha=0.3)+
    geom_hline(yintercept = 0, linetype=3)+
    xlab("Run differential")+ylab("Residual")

highlight_teams <- my_teams_aug %>%
                        arrange(desc(abs(.resid))) %>%
                                head(4)

highlight_teams
```

```
## # A tibble: 4 x 16
##   teamID yearID lgID      G     W     L     R    RA    RD  Wpct .fitted  .resid
##   <fct>   <int> <fct> <int> <int> <int> <int> <int> <int> <dbl>   <dbl>   <dbl>
## 1 PIT      2020 NL       60    19    41   219   298   -79 0.317   0.450 -0.134
## 2 LAN      2020 NL       60    43    17   349   213   136 0.717   0.585  0.131
## 3 TBA      2020 AL       60    40    20   289   229    60 0.667   0.538  0.129
## 4 TEX      2016 AL      162    95    67   765   757     8 0.586   0.505  0.0814
## # ... with 4 more variables: .hat <dbl>, .sigma <dbl>, .cooksd <dbl>,
## #   .std.resid <dbl>
```

```r
library(ggrepel)
```

```
## Warning: package 'ggrepel' was built under R version 4.1.2
```

```r
base_plot +
    geom_point(data=highlight_teams, color = 'blue')+
    geom_text_repel(data = highlight_teams, color = 'blue',
                    aes(label=paste(teamID, yearID)))
```



Residuals can be interpreted as the error of the linear model in predicting the actual winning percentage.

In order to estimate the average magnitude of the errors, we first square the residuals so that each error has a positive value. Calculate the mean of the squared residuals, and take the square root of each mean value to get back to original scale.

```
resid_summary <- my_teams_aug %>%
                summarize(N=n(), avg=mean(.resid),
                          RMSE = sqrt(mean(.resid^2)))

resid_summary
```

```
## # A tibble: 1 x 3
##       N      avg   RMSE
##   <int>    <dbl>  <dbl>
## 1   600 1.35e-15 0.0282
```

```
rmse <- resid_summary %>%
    pull(RMSE)
```

"RMSE = Root Mean Square Error"

If the errors are normally distributed, approximately two thirds of the residuals fall between -RMSE and + RMSE. And 95 % of the residuals are between 2(-RMSE) and 2(RMSE).

We can confirm with the following line.

```
my_teams_aug %>%
    summarize(N=n(),
              within_one = sum(abs(.resid)< rmse),
              within_two = sum(abs(.resid)< 2 * rmse)) %>%
    mutate(within_one_pct = within_one / N,
           within_two_pct = within_two / N)
```

```
## # A tibble: 1 x 5
##       N within_one within_two within_one_pct within_two_pct
##   <int>      <int>      <int>          <dbl>          <dbl>
## 1   600        431        571          0.718          0.952
```

This is the break down of the outcome. The errors are normally distributed.

600 = Number of rows in data set 431 = Number of data between -RMSE and + RMSE 571 = Number of data between 2(-RMSE) and 2(RMSE) 0.781 = Percentage for data between -RMSE and + RMSE 0.952 = Percentage for data between 2(-RMSE) and 2(RMSE)

4.4 The Pythagorean Formula for Winning Percentage.

Bill James derived the following non-linear formula to estimate winning percentage, Pythagorean Formula.

W pct = R^2 / R^2 * RA^2

```
my_teams <- my_teams %>%
    mutate(Wpct_pyt = R^2 / (R^2 * RA^2))

my_teams <- my_teams %>%
    mutate(residuals_pyt = Wpct - Wpct_pyt)

my_teams %>%
    summarize(rmse = sqrt(mean(residuals_pyt^2)))
```

```
##        rmse
## 1 0.5054643
```

```
## 4.4.1 The Exponent in the Pythagorean model

my_teams %>%
    mutate(logWratio = log(W/L),
           logRratio = log(R/RA)) -> my_teams

pytFit <- lm(logWratio ~ 0+logRratio, data=my_teams)
pytFit
```

```
##
## Call:
## lm(formula = logWratio ~ 0 + logRratio, data = my_teams)
##
## Coefficients:
## logRratio
##      1.85
```

The R output suggests best-fit Pythagorean exponent of 1.85, which is significantly smaller than the value of 2.

2011, Boston Red Sox scored 875 runs, 737 allowed runs. ### 162 * 875^2 / 875^2 + 737^2 = 95 games win are expected

They are expected to win 95 games but they won 90 games.

Why did they win 5 games fewer than expected from run differential?

*gl2011.txt contains detailed information of eavery game played in 2011.

```
glheaders <- read_csv("../data/csv_files/game_log_header.csv")
```

```
## Rows: 0 Columns: 161

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (161): Date, DoubleHeader, DayOfWeek, VisitingTeam, VisitingTeamLeague, ...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
gl2011 <- read_csv("../data/csv_files/gl2011.txt",
                   col_names = names(glheaders),
                   na = character())
```

```
## Warning: One or more parsing issues, see `problems()` for details

## Rows: 2429 Columns: 161

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (73): DayOfWeek, VisitingTeam, VisitingTeamLeague, HomeTeam, HomeTeamLea...
## dbl (83): Date, DoubleHeader, VisitingTeamGameNumber, HomeTeamGameNumber, Vi...
## lgl  (4): ForfeitInfo, ProtestInfo, UmpireLFID, UmpireRFID

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
BOS2011 <- gl2011 %>%
    filter(HomeTeam == "BOS" | VisitingTeam == "BOS") %>%
    select(VisitingTeam, HomeTeam, VisitorRunsScored,HomeRunsScore)

head(BOS2011)
```

```
## # A tibble: 6 x 4
##   VisitingTeam HomeTeam VisitorRunsScored HomeRunsScore
##   <chr>        <chr>                <dbl>         <dbl>
## 1 BOS          TEX                      5             9
## 2 BOS          TEX                      5            12
```

```
## 3 BOS          TEX                      1              5
## 4 BOS          CLE                      1              3
## 5 BOS          CLE                      4              8
## 6 BOS          CLE                      0              1
```

We create new columns, ScoreDiff and W.

```
BOS2011 <- BOS2011 %>%
    mutate(ScoreDiff = ifelse(HomeTeam == "BOS",
                              HomeRunsScore - VisitorRunsScored,
                              VisitorRunsScored - HomeRunsScore),
           W = ScoreDiff > 0)
```

Data summary on ScoreDiff grouped by W.

```
library(skimr)

BOS2011 %>%
    group_by(W) %>%
    skim(ScoreDiff)
```

Table 1: Data summary

Name	Piped data
Number of rows	162
Number of columns	6
Column type frequency:	
numeric	1
Group variables	W

Variable type: numeric

skim_variable	W	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
ScoreDiff	FALSE	0	1	-3.46	2.56	-11	-4	-3	-1	-1	
ScoreDiff	TRUE	0	1	4.30	3.28	1	2	4	6	14	

The 2011 Red Sox had their victories decided by a larger margin than their losses (4.3 vs -3.5 runs average), leading to their under-performance of the Pythagorean prediction by 5 games.

Now, create a new data frame called "results" from gl2011.

```
results <- gl2011 %>%
    select(VisitingTeam, HomeTeam,
           VisitorRunsScored, HomeRunsScore) %>%
    mutate(winner = ifelse(HomeRunsScore > VisitorRunsScored,
                           HomeTeam, VisitingTeam),
           diff = abs(VisitorRunsScored -HomeRunsScore))

head(results)
```

```
## # A tibble: 6 x 6
##   VisitingTeam HomeTeam VisitorRunsScored HomeRunsScore winner  diff
##   <chr>        <chr>                <dbl>         <dbl> <chr>  <dbl>
## 1 MIL          CIN                      6             7 CIN        1
## 2 SFN          LAN                      1             2 LAN        1
## 3 SDN          SLN                      5             3 SDN        2
## 4 ATL          WAS                      2             0 ATL        2
## 5 ANA          KCA                      4             2 ANA        2
## 6 DET          NYA                      3             6 NYA        3
```

Then, create a new data frame containing only the games decided by one run.Count the number of wins group by winners.

```
one_run_wins <- results %>%
    filter(diff == 1) %>%
    group_by(winner) %>%
    summarize(one_run_w = n())

one_run_wins
```

```
## # A tibble: 30 x 2
##    winner one_run_w
##    <chr>      <int>
##  1 ANA           27
##  2 ARI           28
##  3 ATL           29
##  4 BAL           22
##  5 BOS           19
##  6 CHA           24
##  7 CHN           25
##  8 CIN           29
##  9 CLE           30
## 10 COL           21
## # ... with 20 more rows
```

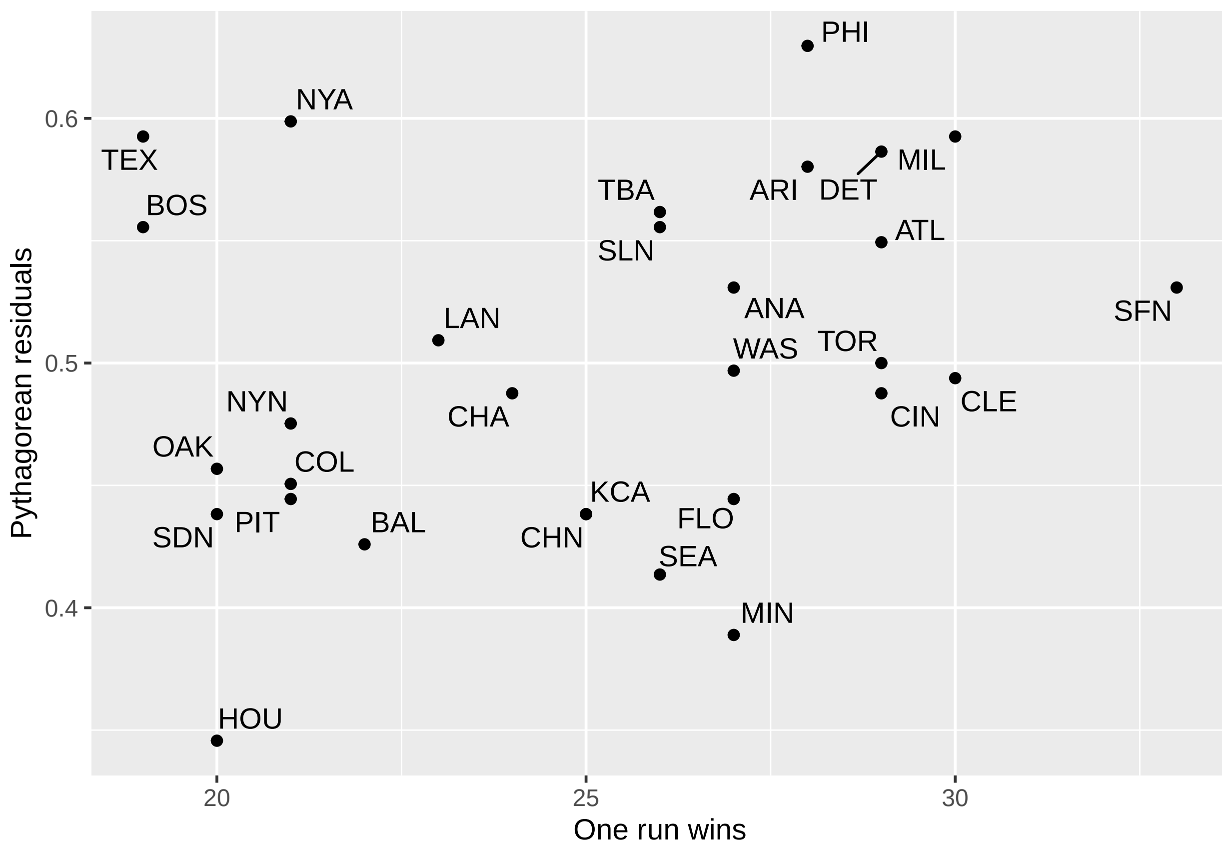
We look at the relation between the Pythagorean residuals and the number of one-run victories.

```
# Create a table and join with my_teams table
teams2011 <- my_teams %>%
    filter(yearID == 2011) %>%
    mutate(teamID = ifelse(teamID == 'LAA', 'ANA',
                           as.character(teamID))) %>%
    inner_join(one_run_wins, by = c('teamID' = 'winner'))


head(teams2011)
```

```
##   teamID yearID lgID   G  W  L   R  RA   RD      Wpct     Wpct_pyt
## 1    ARI   2011   NL 162 94 68 731 662   69 0.5802469 2.281834e-06
## 2    ATL   2011   NL 162 89 73 641 605   36 0.5493827 2.732054e-06
## 3    BAL   2011   AL 162 69 93 708 860 -152 0.4259259 1.352082e-06
## 4    BOS   2011   AL 162 90 72 875 737  138 0.5555556 1.841048e-06
## 5    CHA   2011   AL 162 79 83 654 706  -52 0.4876543 2.006276e-06
```

```
## 6    CHN   2011   NL 162 71 91 654 756 -102 0.4382716 1.749671e-06
##   residuals_pyt   logWratio   logRratio one_run_w
## 1     0.5802446  0.32378708  0.09914790        28
## 2     0.5493800  0.19817693  0.05780100        29
## 3     0.4259246 -0.29849299 -0.19448830        22
## 4     0.5555537  0.22314355  0.17163599        19
## 5     0.4876523 -0.04939276 -0.07650789        24
## 6     0.4382699 -0.24817963 -0.14493402        25
```

```
# plot
ggplot(data = teams2011,aes(x=one_run_w, y= residuals_pyt))+
    geom_point() +
    geom_text_repel(aes(label=teamID)) +
    xlab("One run wins") + ylab("Pythagorean residuals")
```



Teams with top quality closers will tend to preserve small leads, and will be able to over-perform their Pythagorean expected winning percentage. We will check this conjecture.

```
# From Ptching data frame, select pichers who has more than 50GF and less than 2.5 ERA.
top_closers <- Pitching %>%
    filter(GF>50 & ERA < 2.5) %>%
    select(playerID,yearID,teamID)

head(top_closers)
```

```
##    playerID yearID teamID
```

```
## 1 kindeel01   1953    BOS
## 2 arroylu01   1961    NYA
## 3  facero01   1962    PIT
## 4 radatdi01   1962    BOS
## 5 millest01   1963    BAL
## 6 radatdi01   1963    BOS
```

```
my_teams %>%
    inner_join(top_closers) %>%
    pull(residuals_pyt) %>%
    summary()
```

```
## Joining, by = c("teamID", "yearID")
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3827  0.5000  0.5556  0.5404  0.5882  0.6420
```

```
# 4.5 How many Runs for a Win?
D(expression(G * R^2 / (R^2 + RA^2)),"R")
```

```
## G * (2 * R)/(R^2 + RA^2) - G * R^2 * (2 * R)/(R^2 + RA^2)^2
```

```
IR <- function(RS=5,RA=5){
    (RS^2+RA^2)^2/(2*RS*RA^2)
}

ir_table <- expand.grid(RS=seq(3,6,.5),
                        RA=seq(3,6,.5))
head(ir_table)
```

```
##    RS RA
## 1 3.0  3
## 2 3.5  3
## 3 4.0  3
## 4 4.5  3
## 5 5.0  3
## 6 5.5  3
```

```
tail(ir_table)
```

```
##     RS RA
## 44 3.5  6
## 45 4.0  6
## 46 4.5  6
## 47 5.0  6
## 48 5.5  6
## 49 6.0  6
```

```
ir_table %>%
    mutate(IRW=IR(RS,RA)) %>%
    spread(key = RA, value = IRW, sep = '=') %>%
    round(1)
```

```
##     RS RA=3 RA=3.5 RA=4 RA=4.5 RA=5 RA=5.5 RA=6
## 1 3.0  6.0    6.1  6.5    7.0  7.7    8.5  9.4
## 2 3.5  7.2    7.0  7.1    7.5  7.9    8.5  9.2
## 3 4.0  8.7    8.1  8.0    8.1  8.4    8.8  9.4
## 4 4.5 10.6    9.6  9.1    9.0  9.1    9.4  9.8
## 5 5.0 12.8   11.3 10.5   10.1 10.0   10.1 10.3
## 6 5.5 15.6   13.4 12.2   11.4 11.1   11.0 11.1
## 7 6.0 18.8   15.8 14.1   13.0 12.4   12.1 12.0
```