Exploring The Pythagorean Theorem in Baseball

… with different software packages: Stata, SAS, R and Python

Developed by Bill James, the Pythagorean Theorem in baseball estimates a MLB team's winning percent as $\%wins \sim \frac{RS^2}{RS^2 + RA^2}$, where $\%wins = \frac{Wins}{Wins + Losses}$, $RS = RunsScored$ and $RA = RunsAllowed$.[1] Some are so smitten by this that they believe this metric to provide a better measure of performance than the actual winning percent.[2]

Here are a few approaches to estimating the relationship between $\%wins$ and RS and RA..[3]

A simple formula, developed by Earnshaw Cook in the mid 1960's, postulates that:[4]

Model 1: $\%wins \sim \beta \frac{RS}{RA}$, where $\beta \sim .5$.

This specification misses some of the curvature to relationship between RS/RA and %wins, and so an obvious generalization is:

Model 2: $$\%wins \sim \beta_0 + \beta_1 \frac{RS}{RA} + \beta_2 \left(\frac{RS}{RA}\right)^2$$

But suppose we are committed to James' approach. Assume that the true relationship is $\%wins \sim \frac{RS^\beta}{RS^\beta + RA^\beta}$, with β to be estimated. You could estimate β directly using non-linear least squares, but that's for another day. After a little algebra, you'll find that for the given specification, we have an equivalent representation: $\frac{Wins}{Losses} \sim \left(\frac{RS}{RA}\right)^\beta$, or equivalently:

Model 3: $$\ln\left(\frac{Wins}{Losses}\right) \sim \beta \ln\left(\frac{RS}{RA}\right).$$

And further, allowing for possibly different β's for ln*RS* and ln*RA*, we have:

[1] Note that I do not define %wins by Wins/Games… as in many early years, Wins+Losses < Games played.

[2] If you want to learn more about this, google "Bill James" and "Pythagorean Theorem" you will get about 2,960 hits. Here are a few sites worth looking at:

- Update: http://www.baseballprospectus.com/article.php?articleid=342
- Persistence: http://www.philbirnbaum.com/futureexpectations.pdf

And for the math majors only… Weibull:
http://www.math.brown.edu/~sjmiller/math/papers/PythagWonLoss_Paper.pdf

[3] Note that I do not define %wins by Wins/Games… as in many early years, Wins+Losses < Games played.

[4] Cook, E. (1966). Percentage baseball. Cambridge: M.I.T. Press.

Stata, SAS, R and Python

Model 4: $\ln\left(\frac{Wins}{Losses}\right) \sim \beta_1 \ln(RS) + \beta_2 \ln(RA)$,

And then you could test (using an F test) to see if $\beta_1 + \beta_2 = 0$

The Statistical Software packages:

1) Stata and SAS are both available on the BC apps server.
2) R: This is the first time I've worked in R, so I had to install the software.
   a) I went to https://www.r-project.org/ which led me to various mirrors: https://cran.r-project.org/mirrors.html .
   b) I chose the UCLA site, as I know they're generally terrific with stats support: http://cran.stat.ucla.edu/ .
   c) Downloading and installing R was straightforward as expected (I also used the UCLA site for support).
3) Python: I've worked a fair amount in Python over the course of the past decade or so, almost always using Python 2 (now 2.7) and always running batch files in the command window. For this analysis, I've made the switch to interactive Python using Jupyter Notebook (note the spelling of "Jupyter"). There are no doubt a gazillion ways to run regressions in Python… I'll be using *Statsmodels*. To get started you'll need to install Python, Jupyter Notebook, Pandas, Matplotlib, Numpy and Statsmodels.
   a) I went to https://www.python.org/downloads/ and downloaded and installed Python 2.7.14… and then used pip (which was included in the Python installation) to install the other packages…. go to the DOS *command* window (*terminal* with the mac) and…
      - pip install pandas
      - pip install jupyter
      - pip install matplotlib
      - pip install statsmodels

   Alternatively, and as suggested at http://jupyter.org/install , I could have just installed the Anaconda package (which includes all of the packages discussed above, and many more).
   b) To see which packages are included in Anaconda, go to https://docs.anaconda.com/anaconda/packages/pkg-docs
   c) To download and install Anaconda, go to https://docs.anaconda.com/anaconda/install/ and follow the instructions.

The Exercise:

You'll be working with four csv files taken from 2001-2016 MLB team data downloaded from Sean Lahman's website.[5] The datasets:

- winslosses.csv (includes wins (W) and losses (L)),
- runsscored.csv (includes runs scored (R)),
- runsallowed1.csv (includes runs allowed (RA) for half of the teams), and
- runsallowed2.csv (includes runs allowed (RA) for the other half).

To make life more interesting there are (deliberately) missing values in the csv files. All of the packages handle missing values easily (Excel does not). Here's what you'll be doing:

1. Read the four .csv files into four datasets… append the two RA datasets, and merge the resulting dataset together with the other two datasets (merge by yearID and teamID).
2. Sort the data by yearID and teamID and print out the first and last two observations in the dataset.
3. Extract seven variables: yearID (year), lgID (league), teamID (team), W (wins), L (losses), R (runs scored) and RA (runs allowed)
4. … and limit the data further: yearID = 2016
5. Generate summary statistics for the variables in the dataset
6. Generate boxplots for R and RA, in one diagram
7. Generate a histogram for R
8. Generate a scatterplot of R (y-axis) v. RA (x-axis)
9. Generate two new variables: rsra = R/RA and wprcnt = W/(W+L)
10. Generate a scatterplot of wprcnt (y-axis) v. rsra (x-axis)
11. Model 1: Regress wprcnt on rsra (include intercept in the model)… and produce the usual set of regression results and statistics, including 95% confidence intervals for the estimated parameters.
12. Generate four more variables: lnrs=ln(R), lnra=ln(RA), lnrsra=ln(RS/RA), and lnwl=ln(W/L).
13. Model 3: Regress lnwl on lnrsra (no intercept), and produce the usual set of regression results and statistics, including 95% confidence intervals for the estimated parameters.
14. Model 4: Regress lnwl on lnrs and lnra (no intercept), and produce the usual set of regression results and statistics, including 95% confidence intervals for the estimated parameters.
15. Focusing on Model 4, conduct an F test on the null hypothesis that the two parameter values (for lnrs and lnra) are equal in magnitude and opposite in sign.

[5] http://www.seanlahman.com/baseball-archive/statistics/

Stata (is case sensitive)

```
import delimited "C:\winslosses.csv"
save "C:\winslosses.dta"

import delimited "C:\runsscored.csv", clear
save "C:\runsscored.dta"

import delimited "C:\runsallowed1.csv", clear
save "C:\runsallowed1.dta"

import delimited "C:\runsallowed2.csv", clear

append using "C:\runsallowed1.dta"

merge 1:1 yearid teamid using "C:\runsscored.dta"
drop _merge
merge 1:1 yearid teamid using "C:\winslosses.dta"
drop _merge

sort yearid teamid

list in 1/2
list in -2/-1

keep yearid lgid teamid w l r ra
drop if yearid != 2016

summ
graph box r ra
histogram r
scatter r ra

gen rsra=r/ra
gen wprcnt = w/(w+l)
scatter wprcnt rsra
reg wprcnt rsra

gen lnwl=ln(w/l)
gen lnrs=ln(r)
gen lnra=ln(ra)
gen lnrsra=ln(r/ra)

reg lnwl lnrsra, noconstant
reg lnwl lnrs lnra, noconstant
test lnrs+lnra=0
```

Stata generates one file showing commands (and error messages) and results.

SAS (is not case sensitive)

```
libname use '.';
options ls=80 ps=55 nocenter;

proc import datafile="C:\winslosses.csv" out=wl; run;
proc import datafile="C:\runsscored.csv" out=rs; run;
proc import datafile="C:\runsallowed1.csv" out=ra1; run;
proc import datafile="C:\runsallowed2.csv" out=ra2; run;

data temp; set ra1 ra2; run;
proc sort data = temp; by yearID teamID; run;
proc sort data = wl; by yearID teamID; run;
proc sort data = rs; by yearID teamID; run;
date temp; merge temp wl rs; by yearID teamID; run;

proc print data=temp(obs=2); run;
data last2obs; set temp nobs=nobs; if _n_ > nobs-2; run;
proc print data=last2obs; run;

data temp2; set temp(keep = yearID lgID teamID divID W L R RA); run;
data temp2; set temp(where=(yearID = 2016)); run;
data temp2; set temp2; id=_n_; run;

proc sgplot data=temp2; scatter x = ra y = r; run;

proc sort data=temp2; by id; run;
proc transpose data=temp2(keep = id r ra) out=temp2_t; by id; run;
data temp2_t; set temp2_t; label _name_ = "Variable"; label col1 = "Value";
run;
proc sgplot data=temp2_t; vbox col1 / group=_name_ ; run;

*SAS is NOT case sensitive...  here's an example;

data temp2; set temp2; rsra=r/ra; wprcnt=w/(w+L); run;

proc sgplot; scatter x = rsra y = wprcnt; run;

proc reg; model wprcnt = rsra; run;

data temp2; set temp2; lnrs=log(r); lnra=log(ra);
lnrsra=log(r/ra); lnwl=log(w/l); run;

proc reg; model lnwl = lnrsra / noint clb; run;
proc reg; model lnwl = lnrs lnra / noint clb; test lnra+lnrs=0; run;
```

SAS generates two text files: a log file (.log) showing commands, executions and errors, and an output file (.lst) showing results. It also generates separate .png graphics files.

R (is case sensitive)

```
wl <- read.csv("C:/winslosses.csv")
rs <- read.csv("C:/runsscored.csv")
ra1 <- read.csv("C:/runsallowed1.csv")
ra2 <- read.csv("C:/runsallowed2.csv")

ra <- rbind(ra1, ra2)
rawdata <- merge(wl, rs, by=c("yearID","teamID"), all=TRUE)
rawdata <- merge(rawdata, ra, by=c("yearID","teamID"), all=TRUE)

names(rawdata)
head(rawdata, n=2)
tail(rawdata,n=2)
data <- subset(rawdata, select=c("yearID", "lgID","teamID", "divID",
"W","L","R","RA"))
data <- subset(data, yearID = 2016)
names(data)
data
summary(data)

boxplot(data$R, data$RA, col="lightgray")
hist(data$R)
plot(data$RA, data$R, ylab="RunsScores",xlab="RunsAllowed")

data$rsra <- data$R/data$RA
data$wprcnt <- data$W/(data$W+data$L)
plot(data$rsra, data$wprcnt)

ols <- lm(wprcnt ~ rsra, data = data)
coefficients(ols)
ols
summary(ols)
confint(ols, level=0.95)
data$lnrs <- log(data$R)
data$lnra <- log(data$RA)
data$lnrsra <- log(data$R/data$RA)
data$lnwl <- log(data$W/data$L)

ols2 <- lm(lnwl ~ lnrsra -1, data = data)
summary(ols2)
confint(ols2, level=0.95)

ols3 <- lm(lnwl ~ lnrs + lnra -1, data = data)
summary(ols3)
confint(ol3, level=0.95)
anova(ols2, ols3)
```

R generates one file showing commands (and error messages) and results.

Python

```
import pandas as pd
import numpy as np

import statsmodels.formula.api as smf
import matplotlib.pyplot as plt

wl = pd.read_csv('c:/winslosses.csv')
rs = pd.read_csv('c:/runsscored.csv')
ra1 = pd.read_csv('c:/runsallowed1.csv')
ra2 = pd.read_csv('c:/runsallowed2.csv')

ra= ra1.append(ra2)
ra.columns
data1=pd.merge(ra,rs,on=(u'teamID',u'yearID'), how="outer")
data2=pd.merge(wl,data1,on=(u'teamID',u'yearID'), how="outer")

data2.sort_values(by=[u'yearID', u'teamID'])

data2.head(2)
data2.tail(2)

data3=data2[[u'yearID', u'teamID', u'RA',u'R',u'W',u'L']]
dat=data3[data2.yearID==2016]

dat

#wprcnt = dat.W/(dat.W+dat.L)
#rsra = dat.R/dat.RA
#wl = dat.W/dat.L

dat['wprcnt'] = dat.W/(dat.W+dat.L)
dat['rsra'] = dat.R/dat.RA
dat['wl'] = dat.W/dat.L

dat.columns
dat.describe()

dat.boxplot(['R', 'RA'])
dat.hist(['R'])
dat.plot.scatter(x='RA', y='R')

dat.plot.scatter(x='rsra', y='wprcnt')
plt.scatter(dat.rsra, dat.wprcnt)

est = smf.ols(formula='wprcnt ~ rsra', data=dat).fit()
print(est.summary())

est = smf.ols(formula='np.log(wl) ~ np.log(rsra) -1', data=dat).fit()
print(est.summary())

est = smf.ols(formula='np.log(wl) ~ np.log(R)+np.log(RA) -1', data=dat).fit()
print(est.summary())
f_test = est.f_test('np.log(R)+np.log(RA)=0')
print(f_test)
```