Using bWAR and pWAR as predictors for MLB winning percentage

Austyn Hughes

9/30/2022

Introduction:

The goal of this project is to determine if bWAR and pWAR are good predictors for team winning percentage. Batting wins above replacement (bWAR) and pitching wins above replace (pWAR) are non-standardized sabermetric baseball statistics developed to sum up "a player's total contribution to his team".[1] A player's WAR is the number of additional wins his team has achieved above the number of expected team wins of a player if the player was substituted with a replacement-level player.[2] The higher the WAR value the more successful the player is, the lower the value the less successful the player is. We will determine this, by comparing the root mean squared errors (RMSE) of various models to the pythagorean formula RMSE.

The pythagorean formula, is the standard model used by Baseball-Reference to predict winning percentages. The formula for the pythagorean winning percentage is:

$$Pythagorean\ winning\ percentage = \frac{R^{1.83}}{R^{1.83} + RA^{1.83}}$$

Where R is run scored, and RA is runs allowed.

The dataset used to build the models, the test data, was retrieve from Baseball-Reference.com using the "Team Batting" and "Team Pitching" function within Stathead Baseball (https://stathead.com/baseball/). The test data consist of both team batting, and team pitching data from 1961 through the 2020 season. The validation dataset used to test the models created, was also retrieve from Baseball-Reference.com, and consist of both team batting and team pitching data from the 2021 season. Both data sets can be found at the following link, https://github.com/AustynHughes/HarvardX-PH125.9x-Data-Science-Capstone-Project.

The first part of the project was to determine if a model could be developed using just one of the predictors, bWAR or pWAR and seeing what each of their respective RMSE's were. The second part, was to create a multilinear model using both bWAR and pWAR, determine its RMSE's and compare it to the RMSE for the pythagorean model. Overall, the model that used both bWAR and pWAR as predictors generated the smallest RMSE's, but ultimately the pythagorean formula generated the optimal RMSE.

Model	RMSE
bWAR only	0.0690378
pWAR only	0.05871541
bWAR + pWAR	0.03468432
Pythagorean	0.02986361

Analysis:

The first step in analysis for this project, was to find the data needed to build the three models in question. These models would be linear regression models, with bWAR, and pWAR each as their own predictors, with the final model combining the two variables into one model.

Using the following criteria, the test data and the validation data were downloading from Baseball-References, Stathead Baseball page. For Team Batting, sort by "Descending" and "Season". For "Seasons" select years 1961 to 2020, and from the "Statistical Filters" choose "Wins Above Replacement (WAR)". To ensure all data is capture, type "-100" next to the ">=". This make sure that teams that had negative bWAR are included. For Team Pitching, sort by "Descending" and "Season". For "Seasons" select years 1961 to 2020, and from the "Statistical Filters" choose "Wins Above Replacement (WAR)". As with team batting, to ensure that all data is capture, type "-100" next to the ">=". This will make sure teams that accumulated negative WAR from pitching are included. The above steps were followed to download the validation data. For those two pulls, the years were set to 2021.

The test data was then read into R, and the first model was created.

```
> ModelOne <- lm(WL. ~ bWAR, data = TestData)
```

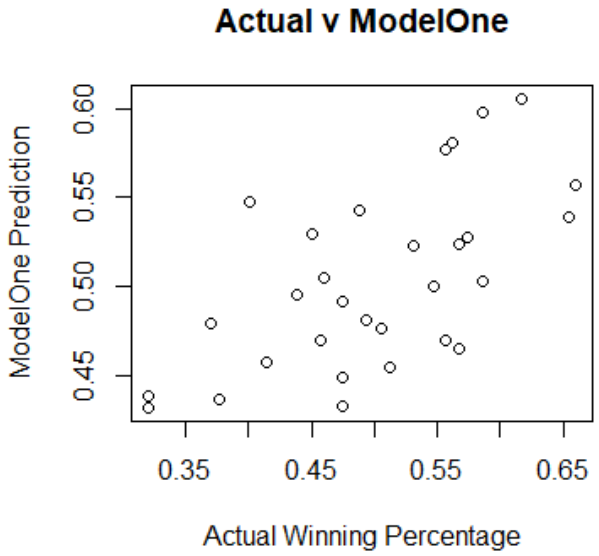
The summary of the model was then done to determine the significance of bWAR.

```
> summary(ModelOne)

Call:
lm(formula = WL. ~ bWAR, data = TestData)

Residuals:
      Min        1Q    Median        3Q       Max
-0.170630 -0.034359 -0.001572  0.034270  0.237837

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.3869299  0.0030819  125.55   <2e-16 ***
bWAR        0.0059124  0.0001469   40.26   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05072 on 1594 degrees of freedom
Multiple R-squared:  0.5042,    Adjusted R-squared:  0.5038
F-statistic:  1621 on 1 and 1594 DF,  p-value: < 2.2e-16
```

From this we see that bWAR is significant (p-value = $2x10^{16}$) and the model itself is significant also (p-value = $2.2x10^{-16}$).

These are all good results. But it is also nice to see a visual of this relationship. The following codes produces a scatterplot which shows the relationship between the actual winning percentage and the predicted winning percentage from ModelOne.

```
#Plotting actual winning percentage v ModelOne predicted winning percentage
ModelOnePlot <- plot(ValData$WL., ValData$M1pWL., xlab = "Actual Winning Percentage",
                     ylab = "ModelOne Prediction", main = "Actual v ModelOne")
```



The above process was then repeated for pWAR.

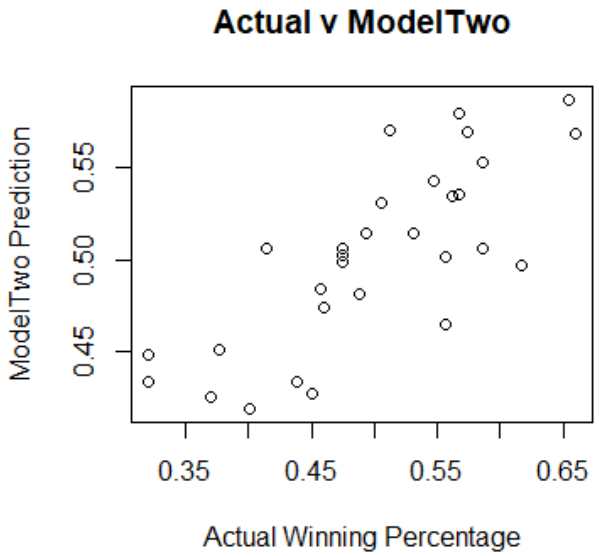
```
> ModelTwo <- lm(WL. ~ pWAR, data = TestData)
```

```
> summary(ModelTwo)

Call:
lm(formula = WL. ~ pWAR, data = TestData)

Residuals:
      Min        1Q    Median        3Q       Max
-0.213279 -0.042221  0.000654  0.043749  0.240416

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.4278563  0.0035467  120.63   <2e-16 ***
pWAR        0.0054142  0.0002388   22.67   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06263 on 1594 degrees of freedom
Multiple R-squared:  0.2438,    Adjusted R-squared:  0.2434
F-statistic:   514 on 1 and 1594 DF,  p-value: < 2.2e-16
```

From the summary, like bWAR, the p-value for pWAR is significant ($2x10^{-16}$), and the overall model is significant also (p-value = $2.2x10^{-16}$). But the major difference between these first two models is the R-squared and the adjusted R-squared.

For bWAR the R-squared was 0.5042 and adjusted R-squared was 0.5038. For pWAR the R-squared was 0.2438 and the adjusted R-squared was 0.2434. This difference would be something to investigate in a future project.

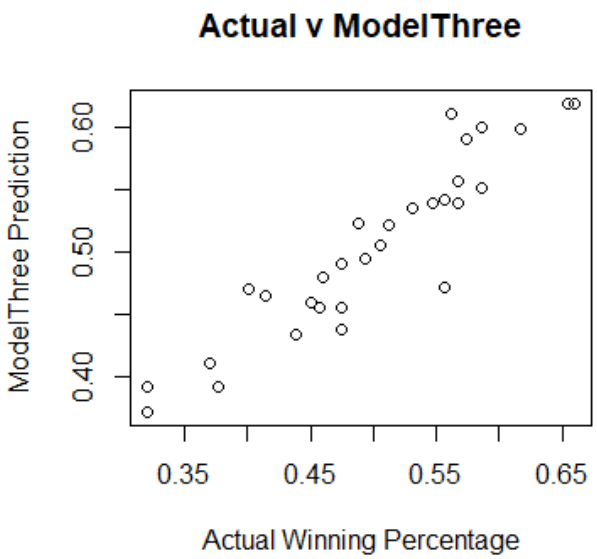
And as before, the following is a plot of the actual winning percentage and the predicted ModelTwo winning percentage.



The final model that was tested combined both bWAR and pWAR.

```
> ModelThree <- lm(WL. ~ bWAR + pWAR, data = TestData)

> summary(ModelThree)

Call:
lm(formula = WL. ~ bWAR + pWAR, data = TestData)

Residuals:
      Min        1Q    Median        3Q       Max
-0.119891 -0.024852 -0.002303  0.021084  0.259407

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.3217205  0.0029600  108.69   <2e-16 ***
bWAR        0.0057590  0.0001102   52.26   <2e-16 ***
pWAR        0.0051148  0.0001451   35.25   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03802 on 1593 degrees of freedom
Multiple R-squared:  0.7214,    Adjusted R-squared:  0.7211
F-statistic:  2063 on 2 and 1593 DF,  p-value: < 2.2e-16
```

Combining the two variables into a single model, we see the best R-squared (0.7214) and adjusted R-squared (0.7211). We can also note that each variable is significant (each with p-values = $2x10^{-16}$) and that the entire model is significant (p-value = $2.2x10^{-16}$).



Since this model has multiple variables, I also calculated the VIF (Variance inflation factor). VIF provides a measure of multicollinearity among the independent variables in a multiple regression model.[3]

```
> vif(ModelThree)
    bWAR     pWAR
1.001562 1.001562
```

A VIF = 1 means that the variables are not correlated and therefore multicollinearity does not exist.

Therefore, all three models created were significant, we can move on to testing them using our validation data set and the results of the project.

Results:

The first step is creating new variables, which are pipped into the ValData dataset. The intercepts and coefficients for each variable is derived from the summary of each model.

```
#Creating a new variable, M1pWL., and pipping it into ValData dataframe. Based on linear model
#created above (ModelOne).
ValData <- ValData %>% mutate(M1pWL. = 0.38692995 + 0.0059124*bWAR)

#Creating a new variable, M2pWL., and pipping it into ValData dataframe. Based on linear model
#created above (ModelTwo).
ValData <- ValData %>% mutate(M2pWL. = 0.4278563 + 0.0054142*pWAR)

#Creating a new variable, M3pWL., and pipping it into ValData dataframe. Based on linear model
#created above (ModelThree).
ValData <- ValData %>% mutate(M3pWL. = 0.3217205 + 0.0057590*bWAR + 0.0051148*pWAR)
```

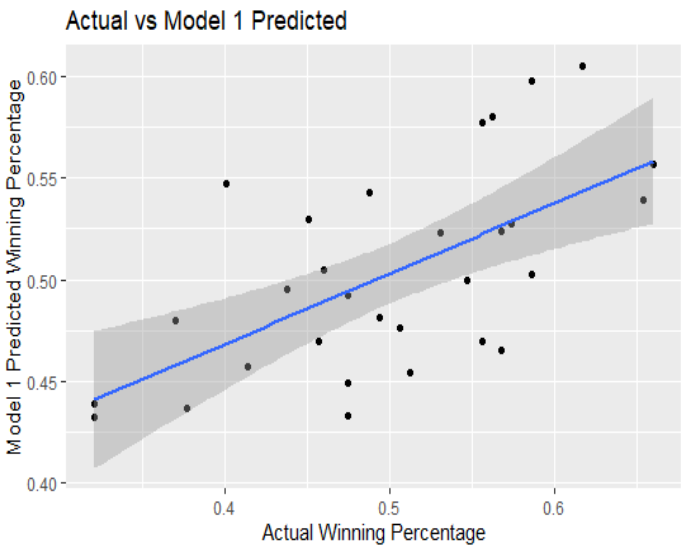
As another form of visualization, I also created variables to calculate the difference between the actual winning percentages and the winning percentages created by each of the models above.

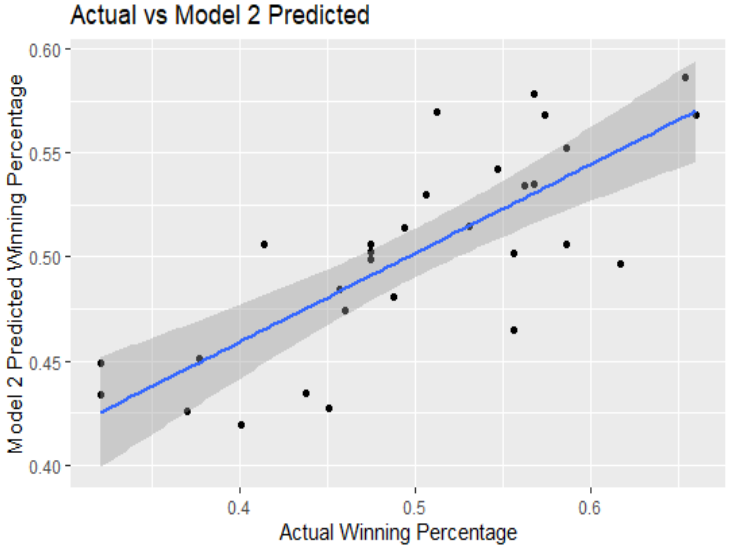
```
#Creating a new variable, M1diffWL., and pipping it into ValData dataframe. This calculates
#the difference between the M1 predicted and acutal winning percentage.
ValData <- ValData %>% mutate(M1diffWL. = M1pWL. - WL.)

#Creating a new variable, M2diffWL., and pipping it into ValData dataframe. This calculates
#the difference between the M1 predicted and acutal winning percentage.
ValData <- ValData %>% mutate(M2diffWL. = M2pWL. - WL.)

#Creating a new variable, M3diffWL., and pipping it into ValData dataframe. This calculates
#the difference between the M1 predicted and acutal winning percentage.
ValData <- ValData %>% mutate(M3diffWL. = M3pWL. - WL.)
```

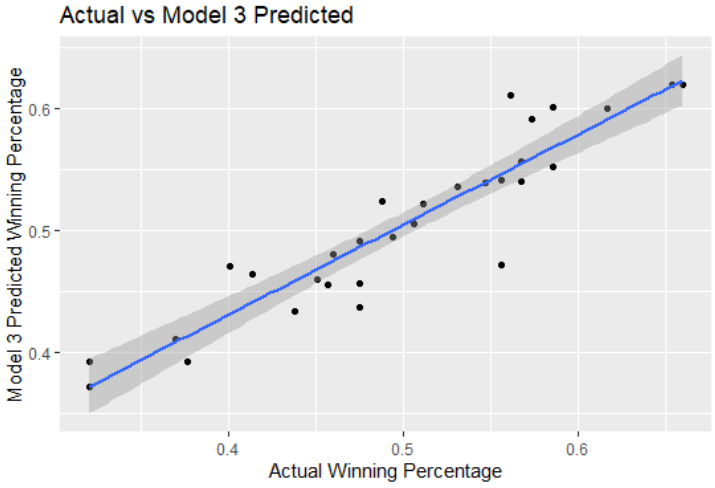
Then a variable was created to calculate the pythagorean winning percentage and it was also pipped into the ValData dataset.

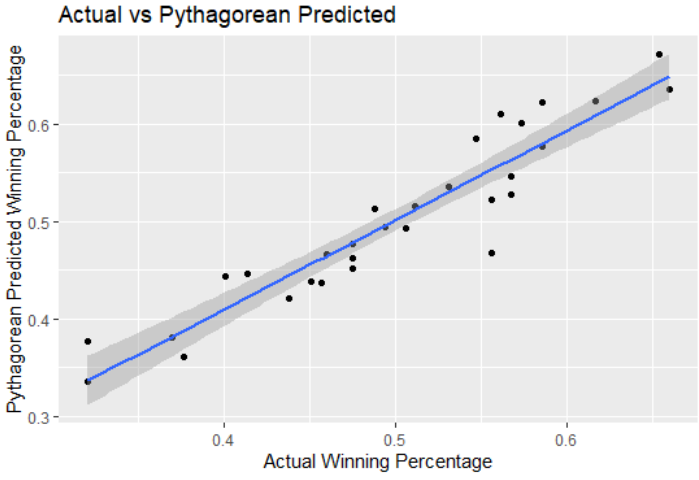
```
#Creating a new variable, pytWL., the standard prediction formula, and pipping it into the ValData dataframe.
ValData <- ValData %>% mutate(pytWL. = (R^1.83)/((R^1.83)+(RA^1.83)))
```

From here, using ggplot, we can see the plots of each of the prediction models versus the actual winning percentage. An advantage of using ggplots, is it also provides a 95% confidence interval for the models.









With each model, we see that the confidence internal is getting tighter and tighter. This demonstrates that the original idea, that bWAR and pWAR could be used as predictors of winning percentage is correct. Unfortunately, based on the RMSE for each of the models, we see that the models generated in this project are still not quite as good as the pythagorean formula.

```
> #RMSE for the standard model pytWL and each model created for pWL.
> rmse_pytWL. <- rmse(ValData$WL.,ValData$pytWL.)
> rmse_ModelOne <- rmse(ValData$WL., ValData$M1pWL.)
> rmse_ModelTwo <- rmse(ValData$WL., ValData$M2pWL.)
> rmse_ModelThree <- rmse(ValData$WL., ValData$M3pWL.)
>
> #Printing RMSE for pytWL. and the three models.
> rmse_pytWL.
[1] 0.02986361
> rmse_ModelOne
[1] 0.0690378
> rmse_ModelTwo
[1] 0.05871541
> rmse_ModelThree
[1] 0.03468342
```

Conclusion:

In conclusion, the results of this project show that the three models considered (bWAR only, pWAR only, and bWAR+pWAR) all show significance when attempting to predict team winning percentage in Major League Baseball. Based on R-squared values, bWAR+pWAR was the best model, followed by bWAR and pWAR.

Model	R-squared
bWAR + pWAR	0.7214
bWAR only	0.5042
pWAR only	0.2438

Strictly comparing RMSE for each of the three models, we again see that bWAR+pWAR is the best. It is then followed by pWAR and bWAR.

Model	RMSE
bWAR + pWAR	0.03468342
bWAR only	0.05871541
pWAR only	0.0690378

Ultimately though, the aim of this project was to determine if using linear regression, bWAR and pWAR could produce a model that could rival the pythagorean formula. It could not. Based on the RMSE of each of the models, including pythagorean, we are still approximately 16% off.

Model	RMSE
Pythagorean	0.02986361
bWAR + pWAR	0.03468342
bWAR only	0.05871541
pWAR only	0.0690378

But not to be discouraged, this does show that there could potentially be alternative methods to predicting winning percentages based on other sabermetric statistics. The only limitations would be having enough time to investigate all the options. In the future, I would like to expand the number of variables, and see if a linear model could be created to compete with the pythagorean formula.

References:

[1] What is WAR? | Sabermetrics Library (fangraphs.com)

[2]What we talk about when we talk about WAR - SweetSpot- ESPN

[3] Variance Inflation Factor (VIF) (investopedia.com)