# A Predictive Model for ODI World Cup Matches

Krishna Kumar, Srihari Srinivasan

APR 30, 2023

## Introduction

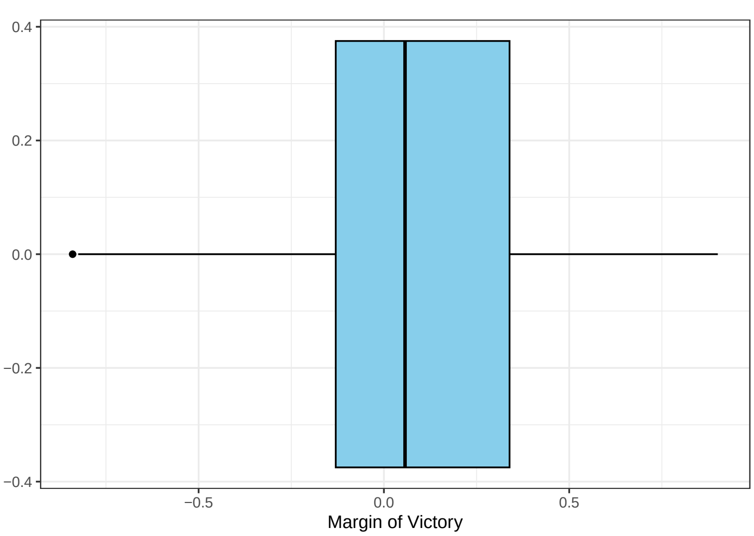
Given that this is a One Day International (ODI) World Cup year, our analysis of batting and bowling performance metrics on the margin of victory, `mov`, is focused on prior World Cup matches. With knowledge of how the ODI format has changed since the introduction of the shorter, Twenty20 (T20) format, particularly the 2007 T20 World Cup, we have limited our data to the last five ODI World Cups (2003, 2007, 2011, 2015, and 2019). After wrangling six batting metrics and four bowling metrics from 4774 individual player performances, we are looking to build a predictive model and gain insight as to which metrics, if any, have an effect on `mov`. In doing so, while this may be beyond the scope of this project, we hope to ultimately use our model to predict the outcome of the 2023 ODI World Cup.

## Exploratory Analysis

### Response Variable

`mov` is the difference between the target set by the team batting first and the total that the chasing team achieved. For example, in a 2003 match between England and Pakistan, England scored 246 runs, setting 247 as the target for Pakistan to chase. They, however, were bowled out for 134, resulting in England winning by 112 runs. Therefore, the `mov` for this match is calculated as $winner\_margin/target = 112/247 = 0.453$.

The figure below is a boxplot of `mov` values.




```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.8400 -0.1300  0.0570  0.0709  0.3390  0.9010
```

From the figure and summary, we can see that `mov` is approximately normally distributed on 7.1%. The single outlier is a 2011 match between Kenya and New Zealand in which Kenya lost by 84.0%. While large `mov` values can be attributed to a blowout, the extreme values probably occurred due to a wide skill gap between two teams. New Zealand, for example, is an established cricketing nation with a strong, experienced team compared to Kenya.

**Predictor Variables**

Batting metrics:
Given that top order batsmen (players 1, 2, and 3) generally play a different role to that of middle order batsmen (players 4, 5, 6, and 7), these metrics have been separated by batting position.
1. `to_runs_pct` - percentage of total runs scored by the top order
2. `mo_runs_pct` - percentage of total runs scored by the middle order
3. `to_mins_pct` - percentage of total time spent in crease by the top order
4. `mo_mins_pct` - percentage of total time spent in crease by the middle order
5. `to_bf_pct` - percentage of total balls faced by the top order
6. `mo_bf_pct` - percentage of total balls faced by the middle order
7. `pct_4s` - percentage of total runs that are 4s
8. `pct_6s` - percentage of total runs that are 6s
9. `to_sr` - average strike rate of the top order
10. `mo_sr` - average strike rate of the middle order

Bowling metrics:
1. `bowlers_used` - total number of bowlers used
2. `pct_mdns` - percentage of total overs that are maidens (an over in which no runs are scored)
3. `wkts` - total number of wickets taken
4. `econ` - average number of runs conceded per over bowled

## Model Development

For our model development, we decided to use a backward step-wise selection approach. Because we use a multiple regression model to predict `mov`, we decided to compare *RMSE*(Root Mean Squared Errror) for determining the "best" collection of predictor variables to include in the model. *RMSE* is the average difference between values predicted by a model and the actual values. This will tell us generally how far apart our predicted values are from the actual values. We also want to look at the *Adjusted R-Squared* value as it tells us the quality of the fitting using our model by explaining how much of the response variable's variation we are able to account for using our predictor variables. Our objective is to minimize *RMSE* and maximize *Adjusetd R-Squared*. First, we split our `odi` dataset into a *training set* and *test set*. The training set includes data from 2003 to 2015 ODI World Cups and the test set includes data from the 2019 ODI World Cup.

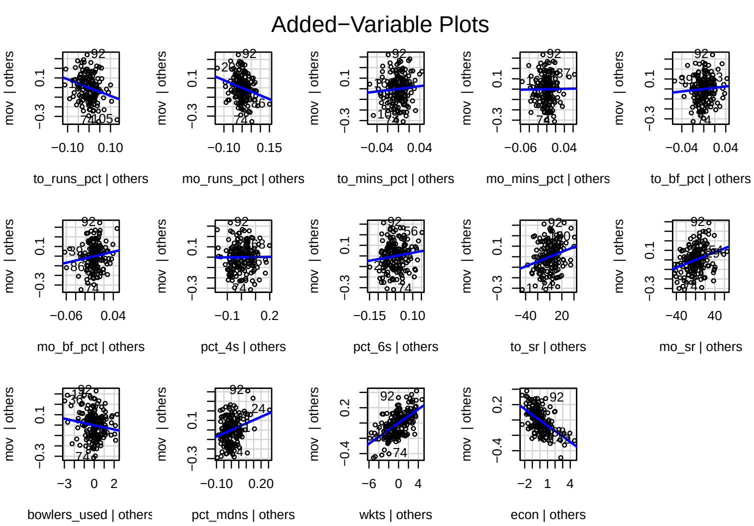
```
##
## Call:
## lm(formula = mov ~ ., data = train_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.31506 -0.08535  0.00520  0.08190  0.33314
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.4882452  0.1587330  -3.076 0.002467 **
## to_runs_pct  -0.8628535  0.2637747  -3.271 0.001310 **
## mo_runs_pct  -0.8042278  0.2725020  -2.951 0.003638 **
```

```
## to_mins_pct   0.6292071  0.6287787   1.001 0.318482
## mo_mins_pct   0.0862708  0.6099045   0.141 0.887692
## to_bf_pct     0.6309025  0.6772988   0.931 0.352991
## mo_bf_pct     1.1391437  0.6397287   1.781 0.076853 .
## pct_4s        0.0158615  0.1478799   0.107 0.914717
## pct_6s        0.2999686  0.1898668   1.580 0.116095
## to_sr         0.0023055  0.0006801   3.390 0.000880 ***
## mo_sr         0.0020120  0.0005138   3.916 0.000133 ***
## bowlers_used -0.0208609  0.0112526  -1.854 0.065587 .
## pct_mdns      0.6888521  0.1977250   3.484 0.000637 ***
## wkts          0.0451133  0.0046679   9.665  < 2e-16 ***
## econ         -0.0713556  0.0086423  -8.257 5.15e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.132 on 161 degrees of freedom
## Multiple R-squared:  0.8906, Adjusted R-squared:  0.881
## F-statistic: 93.57 on 14 and 161 DF,  p-value: < 2.2e-16
```

```
## [1] 0.1531916
```

```
## [1] 0.1327484
```

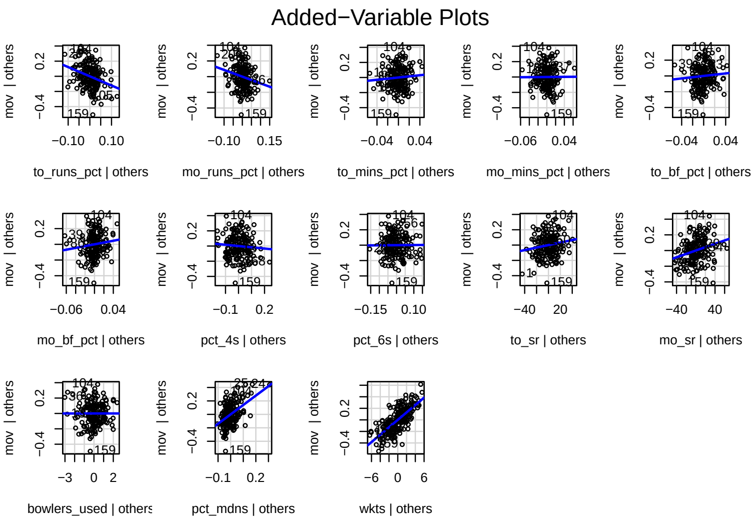
After training our multiple regression model using all 14 predictor variables, we can see the *Adjusted R-Squared* value is 0.881 meaning we are able to explain 88.1% of the variation in `mov` using our 14 predictor variables. Using this model, we predicted `mov` based on the test set and calculated the *RMSE*, getting a value of around 0.153. If we normalize the *RMSE* by dividing the RMSE by the range of the test set's `mov`, we get a value of around 0.133.




For our `odi` dataset, we have 14 potential predictor variables. Backward step-wise selection removes each of these 14 variables from the model, one at a time, and checks for improvement in the performance metric, *RMSE*. Using a loop, we remove one variable at a time from the training set. We remove the *i*-th column using `train_set[,-i]`. This allows us to iterate through columns 1 to 14.

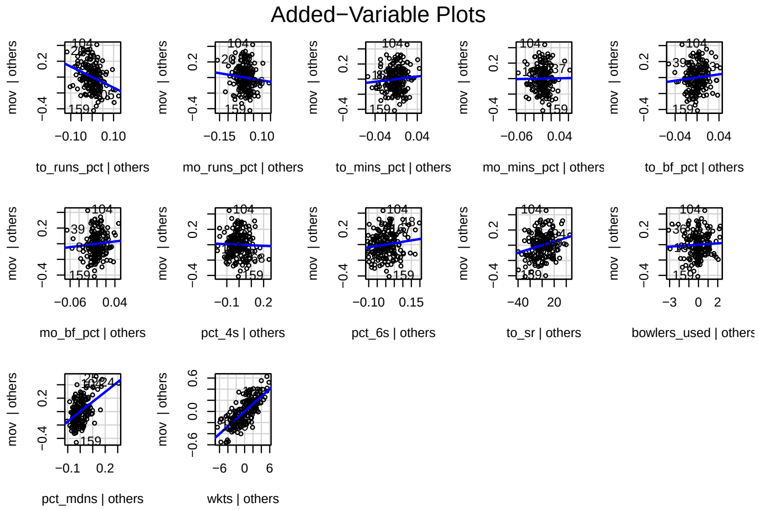
```
##         vals
## 1  0.1541412
## 2  0.1452954
## 3  0.1559078
## 4  0.1532753
## 5  0.1513644
```

```
## 6  0.1504844
## 7  0.1534606
## 8  0.1549225
## 9  0.1627135
## 10 0.1423925
## 11 0.1457553
## 12 0.1601905
## 13 0.1797444
## 14 0.1889362
```




This table shows all of the different *RMSE* values after removing one predictor variable from the training set. We want to find the lowest *RMSE* value and remove the predictor variable associated with it because the variable does not contribute to reducing the *RMSE* for our model. Based on this table, the lowest *RMSE* is around 0.142 which is the 10th row and corresponds to the `mo_sr` predictor variable. Now, we will permanently remove this variable from the training set and repeat this process again to keep minimizing the *RMSE*.

```
##         vals
## 1  0.1401743
## 2  0.1407669
## 3  0.1453592
## 4  0.1425708
## 5  0.1396741
## 6  0.1411606
## 7  0.1442560
## 8  0.1410817
## 9  0.1454950
## 10 0.1382726
## 11 0.1493844
## 12 0.1793459
## 13 0.1749365
```




This table shows all of the different *RMSE* values after removing one predictor variable from the training set without the `mo_sr` predictor variable. Based on this table, the lowest *RMSE* is around 0.138 which is the 10th row and corresponds to the `bowlers_used` predictor variable. Now, we will permanently remove this variable from the training set and repeat this process again to keep minimizing the *RMSE*.

## Model Analysis

## Conclusion