Data Analytics: Assignment 1 (D/L Method)

Sharath C R (21744)

May 2023

1 Data Description

Source : `https://ece.iisc.ac.in/~rajeshs/E0259/04_cricket_1999to2011.csv`

The data is present in a single csv file containing **38** different columns describing different aspects of the game of cricket. There are a total of **126,768** rows of data with each row representing an *over* of data. Total of **1,423** Day-Night and Day matches are catalogued in the dataset.

The data can be interpreted in many ways for calculating the ***Run Production Functions***. Many of the experiments and observations are documented in the *Experiments and Resutls* section.

2 Data Cleanup

Multiple ways of data cleanup has been attempted. Each consideration will results in a slightly different set of ***Run Production Funtions***.
Below are the different cleanup steps that are implemented.

- Before any further processing is done, only the 1^{st} innings data from the entire dataset is taken.
- Only the matches that are **Uninterrupted** by rain or any other reason are considered. The switch *uninterrupted_only* is set to *True* by default and can be set to *False* to include the **Interrupted** matches as well.
- Some of the data is flagged as erroneous using the column *Error.In.Data*. These rows are removed by default. They can be included by setting the argument *remove_errors* to *False*
- By default both *Day* and *Day-Night* matches are included. Using the argument *match_type*, just the *Day* matches or *Day-Night* matches can be selected.
- The extras scored in the match are not separated from the actual runs scored. This sometimes leads to the illusion that the data is incorrect. *(For some of the Matches, I manually verified the scores on `https://www.cricbuzz.com/`)*. Such instances prompted me to add this part to the cleanup process. When the argument *consider_extras* is set to *False*, all the extras scored in the matches will be ignored. By default this is set to *True* so that all the extras are used in the calculations.

3 Model

The model to be used for generating the ***Run Production function*** is,

$$Z(u, w) = Z_0(w)(1 - e^{\frac{-Lu}{Z_0(w)}}) \tag{1}$$

4 Loss function

The loss function is the standard ***Sum of squared error*** summed across overs, wickets and data points. Therefore, for K data points, assuming the slope of all the 10 functions are same at $u = 0$, we can write the following,

Let the ***Squared error*** be,

$$l(\hat{y}, y) = (\hat{y} - y)^2 \tag{2}$$

Let the data set be $\mathcal{D} = \{((u_1, w_1), y_1), ((u_2, w_2), y_2), ((u_3, w_3), y_3), ...((u_K, w_K), y_K)\}$ where y_k is the runs scored when w_k wickets are in hand with u_k overs remaining.

- *Normalized Loss*

$$\mathcal{L}(Z_0, L) = \frac{1}{K} \sum_{k=1}^{K} l(Z(u_k, w_k), y_k) \tag{3}$$

- *Un-normalized Loss*

$$\mathcal{L}(Z_0, L) = \sum_{k=1}^{K} l(Z(u_k, w_k), y_k) \tag{4}$$

Both (3) and (4) are implemented differently in python to optimize run-time. Instead of going through all the data points individually, I have grouped the data points based on *wickets* and *overs*. This is possible because python functions can take vector arguments and produce vector outputs. Code snipped can be found in the *Miscellaneous* section.

5 Non Linear Regression

Using the Loss functions defined in equations (3)&(4) we can perform Non Linear Regression to obtain the best possible function parameters as follows,

$$Z_0^*, L^* = \underset{Z_0, L}{\operatorname{argmin}} \mathcal{L}(Z_0, L) \tag{5}$$

with the following constraints.

$$0 < Z_0(w) < \infty, \forall w \in \{1, 2, 3, ...10\} \tag{6}$$

$$0 < Z_0(w+1) - Z_0(w) < \infty, \forall w \in \{1, 2, 3, ...9\} \tag{7}$$

$$0 < L < \infty \tag{8}$$

Please note that Z_0 in equations (3), (4), and (5) is a vector quantity representing

$$Z_0(1), Z_0(2), Z_0(3), ...Z_0(10)$$

6 Experiments and Results

Several experiments are done and corresponding results and observations are documented in this section. 1^{st} and 2^{nd} experiment are the most pertinant ones to look at.

6.1 Normalized Error with all default parameters

Parameters Used

```
uninterrupted_only = True
remove_errors = True
match_type = 'all'
consider_extras = True
init_type = 'mean'
norm_err=True
print_debug=True
```

Here only *Uninterrupted* matches are considered where all *errors* in the data are removed and *extras* are considered. The *error* is ***normalized***.

The 11 parameters are shown below along with the ***Run Production Functions*** plot and ***Resource Remaining*** plot. The final error after optimization is : ***4,982.96***

L	$Z_0(1)$	$Z_0(2)$	$Z_0(3)$	$Z_0(4)$	$Z_0(5)$	$Z_0(6)$	$Z_0(7)$	$Z_0(8)$	$Z_0(9)$	$Z_0(10)$
10.46	14.15	29.96	57.84	90.88	116.55	153.38	183.59	228.44	260.18	299.27

Table 1: Z_0 and L values with default parameters and normalized error

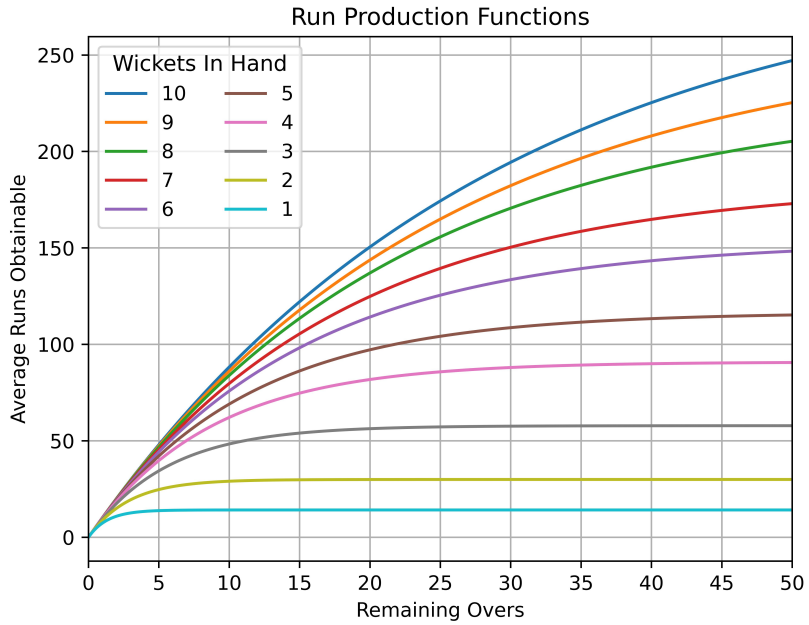

Figure 1: Run Production Functions

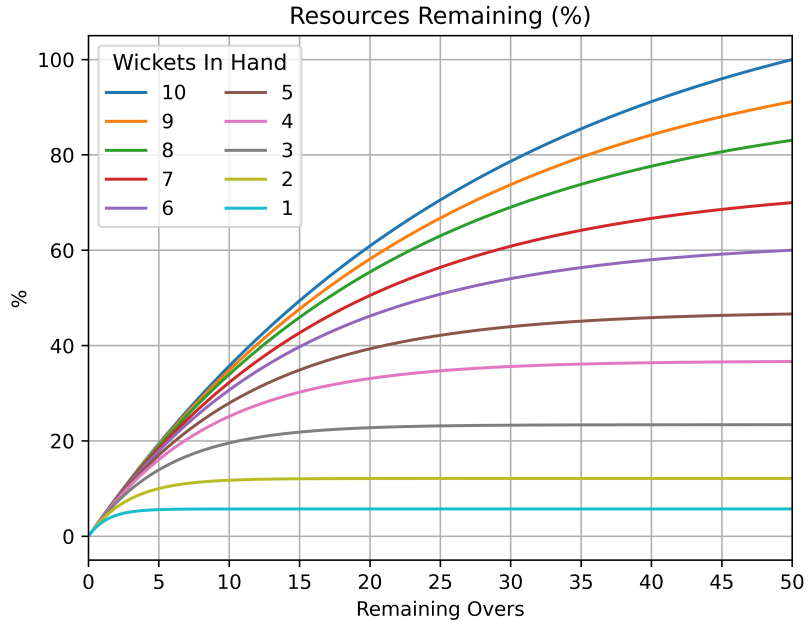

Figure 2: Resources Remaining

6.2 Un-Normalized Error with all default parameters

Parameters Used

```
uninterrupted_only = True
remove_errors = True
match_type = 'all'
consider_extras = True
init_type = 'mean'
norm_err=False
print_debug=True
```

Here only *Uninterrupted* matches are considered where all *errors* in the data are removed and *extras* are considered. The *error* is ***not-normalized***.

The 11 parameters are shown below along with the ***Run Production Functions*** plot and ***Resource Remaining*** plot. The final error after optimization is : ***84,680,891.70***

L	$Z_0(1)$	$Z_0(2)$	$Z_0(3)$	$Z_0(4)$	$Z_0(5)$	$Z_0(6)$	$Z_0(7)$	$Z_0(8)$	$Z_0(9)$	$Z_0(10)$
10.39	14.34	30.05	57.89	91.06	117.05	154.01	184.48	229.6	261.47	304.68

Table 2: Z_0 and L values with default parameters and un-normalized error

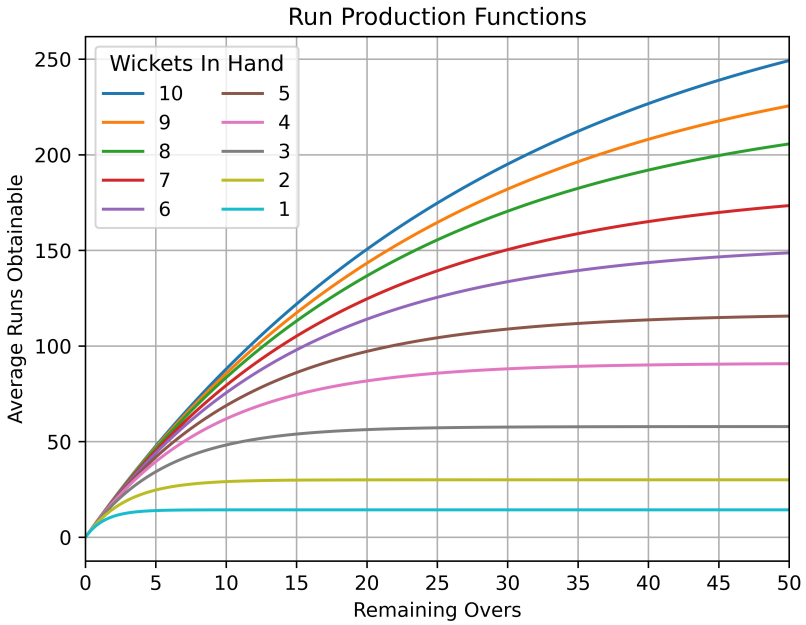

Figure 3: Run Production Functions

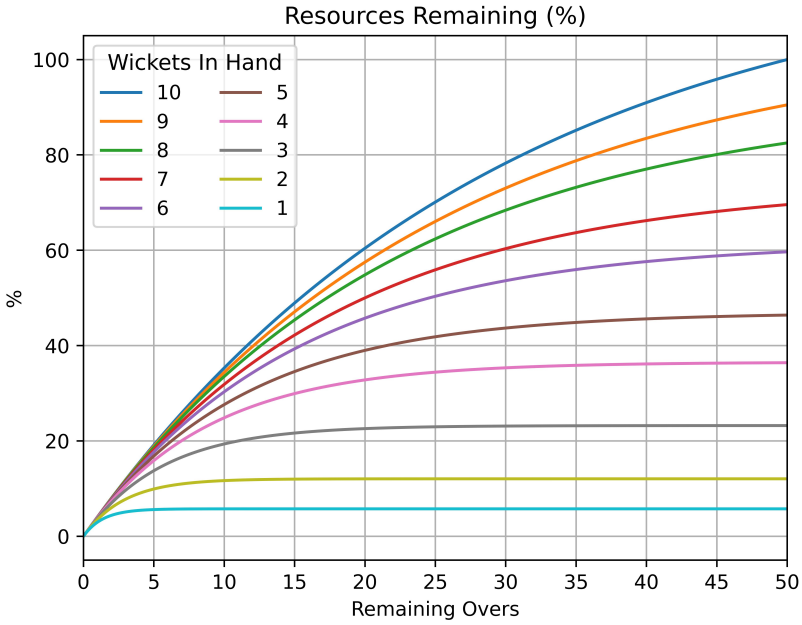

Figure 4: Resources Remaining

6.3 Normalized Error with none of the errors removed

Parameters Used

```
uninterrupted_only = True
remove_errors = False
match_type = 'all'
consider_extras = True
init_type = 'mean'
norm_err=True
print_debug=True
```

Here only *Uninterrupted* matches are considered, none of the *errors* in the data are removed and *extras* are considered. The *error* is ***normalized***.

The 11 parameters are shown below along with the ***Run Production Functions*** plot and ***Resource Remaining*** plot. The final error after optimization is : ***4,967.93***

L	$Z_0(1)$	$Z_0(2)$	$Z_0(3)$	$Z_0(4)$	$Z_0(5)$	$Z_0(6)$	$Z_0(7)$	$Z_0(8)$	$Z_0(9)$	$Z_0(10)$
10.48	14.14	29.93	57.79	91.01	116.61	153.27	183.4	228.46	259.9	299.21

Table 3: Z_0 and L values with default parameters, normalized error and errors in data not removed

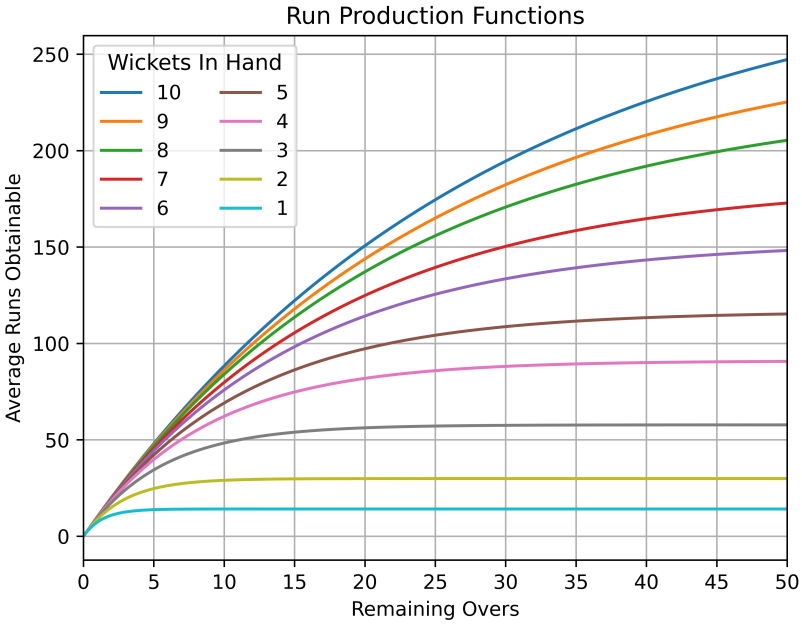

Figure 5: Run Production Functions

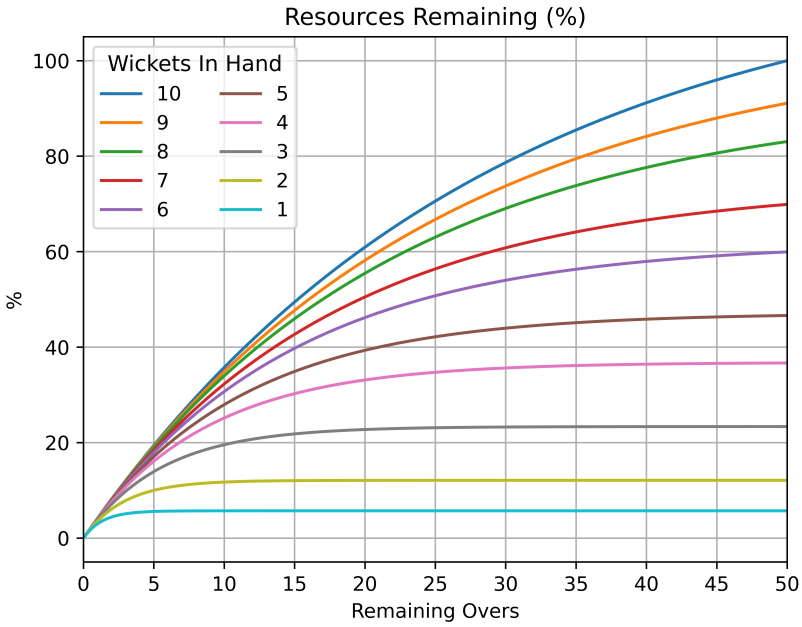

Figure 6: Resources Remaining

6.4 Normalized Error with all of the errors removed, extras are ignored

Parameters Used

```
uninterrupted_only = True
remove_errors = True
match_type = 'all'
consider_extras = False
init_type = 'mean'
norm_err=True
print_debug=True
```

Here only *Uninterrupted* matches are considered, all of the *errors* in the data are removed and *extras* are *not* considered. The *error* is ***normalized***.

The 11 parameters are shown below along with the ***Run Production Functions*** plot and ***Resource Remaining*** plot. The final error after optimization is : ***5,001.22***

L	$Z_0(1)$	$Z_0(2)$	$Z_0(3)$	$Z_0(4)$	$Z_0(5)$	$Z_0(6)$	$Z_0(7)$	$Z_0(8)$	$Z_0(9)$	$Z_0(10)$
10.47	14.27	29.9	57.79	90.87	116.44	153.22	183.43	228.11	259.65	298.451

Table 4: Z_0 and L values with default parameters and normalized error

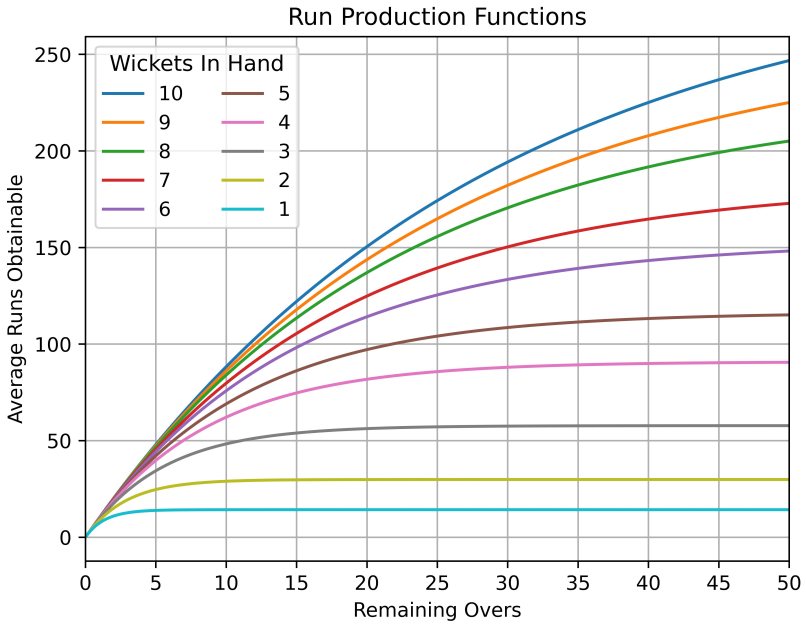

Figure 7: Run Production Functions

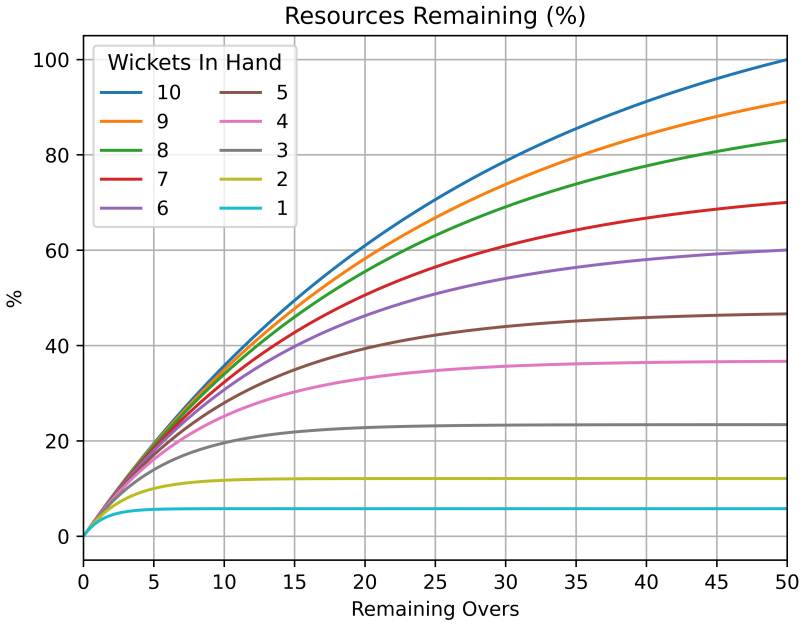

Figure 8: Resources Remaining

7 Miscellaneous

7.1 Implementation of Error function in Python

```
def error(params, data, norm_err=True, print_debug=False):
    # Get all the 11 parameters
    Z0 = params[:10]
    L = params[10]
    err1 = err2 = 0
    # Iterate through all the wickets, essentially going through the complete dataset.
    for i in range(1,11):
        # This is to capture the case where there are 50 overs to go as this will be
        # missed when we do (50 - u) to calculate the remaining overs.
        # i == 10 is needed as this can only happen at the start of a match.
        if i == 10:
            y50 = objective_fun(Z0[9], L, 50)
            # 'Over' == 1 is needed because we are looking for the beginning of the
            # match and 'Innings.Total.Runs' is taken because the the final score with
            # 50 overs and 10 wickets in hand.
            err2 += np.sum(
                (y50 - np.array(data[(data['Over'] == 1)]['Innings.Total.Runs']))**2)
        # else: ( This will be always true )
        y = objective_fun(Z0[i-1],
                          L,
                          50-df_reduced[df_reduced['Wickets.in.Hand']==i]['Over'].values)
        err1 += np.sum(
        (y -
        df_reduced[df_reduced['Wickets.in.Hand']==i]['Runs.Remaining'].values)**2)
    # Now the normalized error is minimized if norm_err = True.
    if norm_err:
        err2 /= len(np.array(data[(data['Over'] == 1)]['Innings.Total.Runs']))
        err1 /= len(data)
    err = err1 + err2
    if print_debug:
        print("Error this iter: {:5f}".format(err))
    return err
```