Improving Duckworth-Lewis: Statistical Methods for Revising Score Targets in Limited-Overs Cricket

Matthew Knowles

Department of Mathematics,
University of York,
United Kingdom

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Mathematics

Abstract

Write this last

Acknowledgements

The author would like to thank (INSERT SUPERVISOR HERE) for their invaluable supervision and advice throughout this project. Secondly, my eternal thanks and admiration for my friends in York, who have made my time here so wonderful from the very begining, and for all the support you have given me every step of the way. Finally, to the UYMCC, who have enriched my university experience beyond belief.

Contents

Chapter 1

Background

1.1 Limited-Overs Cricket and the need for Duckworth-Lewis

The purest form of cricket, so-called “first class” (FC) cricket, gave way to limited overs cricket in early 1950 in India. As attendance at *County Championship* games in England began to drop, a limited overs tournamentm the “Gillette Cup”, was introduced in 1963. The idea that a game only lasting over the course of one day instead of four or five would boost crowd numbers. Originally, limited-overs and one day cricket were synonymous, but now limited-overs cricket comes in two forms: One Day International (50 Overs) and Twenty20 (20 Overs). Limited-overs cricket was designed to be played in knockout competition, which means a definitive result is attained on the day. This is in stark contrast to FC cricket, in which a draw is a highly likely result.

The first version of the Duckworth-Lewis-Stern method[1] (DLS) was published by Duckworth [1] in 1998, to solve a very specific problem. In limited overs cricket, when rain or bad-light interrupt play, and the full number of overs cannot be completed, DLS is used to revise the number of runs needed by the batting team in the second innings, to a total that would be considered an equivalent target. Duckworth states in the original paper that this method is designed in such a way that no team benefits or suffers from the shortening of the game. This is in contrast to unlimited-overs cricket, where a shortening of the game could actually massively favour the losing side into getting a draw. Due to the tournament-based nature of limited-overs cricket, a result is necessary, and so DLS is designed to reduce the likelehood of the two teams drawing.

1.2 A Look at the Original Duckworth-Lewis Method

This method was designed to fit 5 requirements. We summarise these requirements from the original paper in the following list.

1. Method must be equally fair to both teams.
2. Results must be sensible in all possible scenarios.
3. Result should not depend on the scoring pattern of the first team.
4. Ease of application: requiring a table of numbers and a calculator.
5. Ease of understanding by all involved in cricket.

[1]Commonly still just reffered to as Duckworth-Lewis

Points (1-3) and (5) are perfectly understandable, however point (4) may be a little dated. With quick computational methods and access to computers, this requirement could be relaxed a little bit.

1.3 Stern's Extension

Chapter 2

The Data and Associated Methods

2.1 Data Wrangling Methodology

The data that forms the backbone of this project is courtesy of [2], which provides ball-by-ball match data for over 2000 One-Day International matches and over 1500 Twenty20 matches. The ball-by-ball aspect of this data isn't immediately very useful, but the extra match information will be used extensively throught this paper. Each match comes in the form of a .JSON file, so must be decoded first in order to make use of. All the code that will be discussed throught this section can be found in appendix one.

In order to get the data into a usable form, we wrote several Python programs to comb through and select datapoints that were of interest. Firstly, the data was split into games that were decided on DLS, and those that played out a full innings. This resulted in a CSV file containing information on 190 ODIs that were decided on DLS. The datapoints collected were the home and away team, the ground, the winner of the game, how much they won by, and what their target was. The rationale behind adding the ground is that the ground a game is played at has a considerable influence on how high a score will go. For example, consider the two grounds Edgbaston and Trent Bridge, in Birmingham and Nottingham respectively. The follwing table shows the averages of 6 international teams at the two English grounds in their entire ODI playing history.

Team	Edgbaston Average	Trent Bridge Average
Australia	196.3	272.9
Bangladesh	211.5	268.7
England	227.8	246.1
India	224.7	226.3
New Zealand	202.2	241.9
Pakistan	189.6	250.4
Overall	208.7	251.1

So one can see that on average, a team playing at Trent Bridge is going to have a high scoring game than a team playing at Edgebaston. But this is something that DLS does not take into account when revising the score. It is for this reason the ground has been included in the data. We also included the runs required to win, and how many overs are available. Finally, we have how many runs or wickets the winning team won by.

2.2 Analysis of DLS Match Data

We first look at the relationship betweem how many overs were available and how many runs required in those overs. The distubution of this can be seen in Figure 2.1.

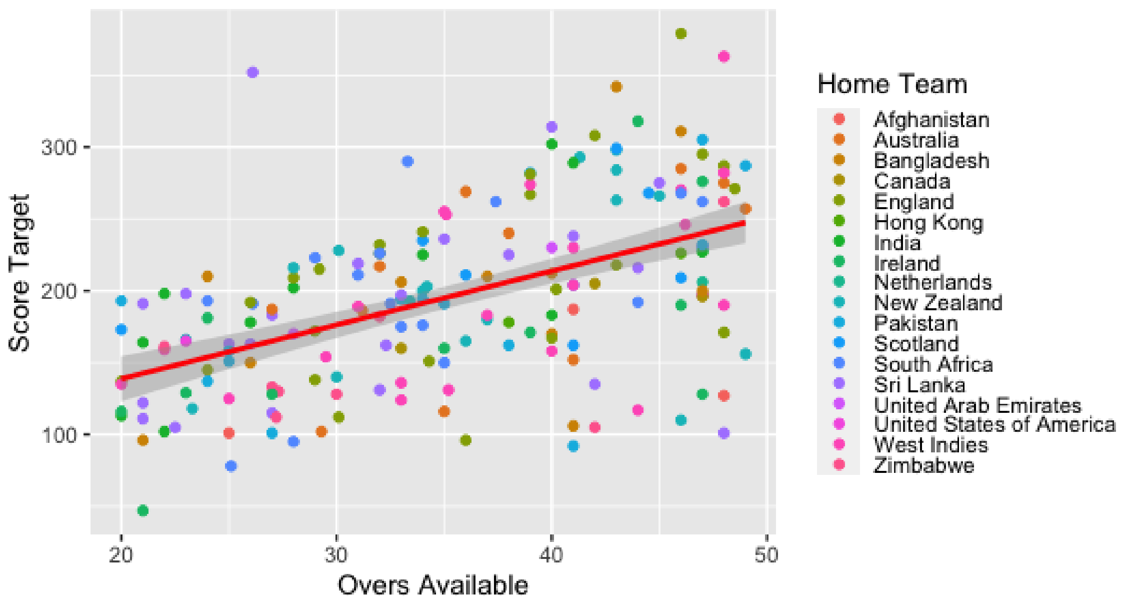

Figure 2.1: Distribution of Overs to Target, coloured by host nation

One thing that immediately sticks out is that there is a slight linear upward trend in the target score, but there is a considerable amount of variation in each over range. We have included a linear regression model, calculated in R using the *lm()* function. The model given by this calculation is given as

$$R = 63.8 + 3.7O \pm 0.45$$

Where R is the number of runs required, and O is the overs available. So does that mean instead of going through the DLS process we could put overs into this model and read off the target score? Well, no, because there is too much variation in target scores for this linear model to be reliable. To see this, we have a boxplot of the target runs produced by DLS. In the plot, each red cross represents a match.[1]

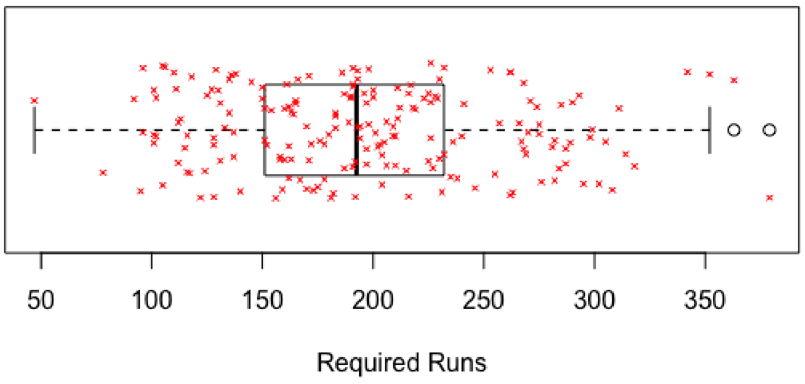

Figure 2.2: Distribution of target run scores calculated by DLS

[1]Note that the vertical spread of points in this plot is purely a styalistic one

If the two maximums were closer together, then this would arguably be a functional alternative to DLS, but the spread of scores is too large for the method to produce reliable results, and would often result in a lot of targets that didn't fit with point (2) of the original 5 criteria.

Bibliography

[1] Frank C Duckworth and Anthony J Lewis. A fair method for resetting the target in interrupted one-day cricket matches. *Journal of the Operational Research Society*, 49(3):220–227, 1998.

[2] Stephen Rushe. Cricsheet. https://cricsheet.org/. Accessed: 27/07/2021.

Appendix A

Python Code for Chapter 2

Listing A.1: Data Wrangling Code

```
import json
import os
import csv

#Moves all the games from a set of files that were decided by DLS
def get_dls_results(inputPath, outputPath):
    os.chdir(inputPath)
    for f in os.listdir():
        with open(inputPath+f,"r") as file:
            data = json.load(file)
        if("method" in data["info"]["outcome"]):
            if(data["info"]["outcome"]["method"] == "D/L"):
                os.system("mv "+f+" "+outputPath)


def get_match_data(matchPath, csvPath):
    os.chdir(matchPath)
    for f in os.listdir():
        with open(matchPath+f, "r") as file:
            data = json.load(file)

        homeTeam = data["info"]["teams"][0]
        awayTeam = data["info"]["teams"][1]
        ground = data["info"]["venue"]

        if "winner" in data["info"]["outcome"]:
            winner = data["info"]["outcome"]["winner"]
            if "runs" in data["info"]["outcome"]["by"]:
                runsWonBy = data["info"]["outcome"]["by"]["runs"]
                wicksWonBy = 0
            else:
                wicksWonBy = data["info"]["outcome"]["by"]["wickets"
                    ]
                runsWonBy = 0
        else:
            winner = "tie"
        targetOvers = data["innings"][1]["target"]["overs"]
        targetRuns = data["innings"][1]["target"]["runs"]

        row = homeTeam,awayTeam,ground,winner,str(runsWonBy),str(
            wicksWonBy),str(targetRuns),str(targetOvers)
        csvF = open(csvPath, "a")
        writer = csv.writer(csvF)
        writer.writerow(row)
        csvF.close()


get_match_data("/Users/matthew/Documents/Uni/Dissertation/Data/dls/"
    , "/Users/matthew/Documents/Uni/Dissertation/Data/dlsMaster.csv")
```