Group Report: Fantasy Football Optimiser

|  |  |  |
| --- | --- | --- |
| Group Members | Student URN | Percentage of Mark |
| Christopher Hawes | 6361307 | 20 |
| Anoop Bindra Martinez | * 6364379 | 20 |
| Luke Lockley | 6355410 | 20 |
| Thomas Lungley | 6322805 | 20 |
| Joshua Wadey | 6329061 | 20 |

Motivation

The motivation for our project originated from the desire to find an interesting application of our recently acquired Matlab skills and our mutual interest in the sport of football. We aimed to create a Fantasy Football side that could compete with the most successful teams created during last season and the season thus far, using coding. We felt that creating a code that used existing data to generate an optimal team, subject to a variety of constraints, would be both a fun and challenging application. Several features such as the Graphical User Interface (GUI) were then added to further demonstrate our variety of skills whilst presenting the code in a more user-friendly format.

Methodology

After the initial idea, the first part of the project was to find statistical data that we could use. Here we came across our first decision - which Fantasy Football Provider to base the project on? We chose to follow the Fantasy Premier League as it would provide us with each individual’s data, giving us the opportunity to calculate the points ourselves, adding an extra layer of difficulty to the project. This also presented the opportunity of generating a team using existing data as well as a current team based off continuously updating data through scraping.

Our second issue was how to tackle the problem of optimising the team using the available data. We collectively decided that this process was not strictly an optimisation as we had once thought; it was a sorting problem. This was due to the difficulty of adhering to the range of constraints that we faced in the optimisation process.

The thought process we followed in our sorting problem is as follows:

1. After collecting the data we organised it so that it was ordered by points (in descending order) per position (starting from goalkeeper).
2. From this a team of the highest points total was selected, applying only the constraint of a maximum number of players in each position.
3. We would then implement the financial constraint. This involved removing the player with the lowest point-cost ratio (average points) and replacing him with the player of that position with the next highest number of points (not already included) until the constraint was satisfied.
4. The constraint of being limited to three players per club is then implemented. If this constraint is violated then the same selection method as above is used for all players from the violating club, leaving us with our optimal team.

Once we had solved our sorting problem, the next step was to decide on how to present our results. We chose to explore the option of using a GUI that would include the use of pushbuttons to display the outcome in a user-friendly fashion.

How to run the code

Please see the ‘readme’ for complete instructions on how to run the code, including details of exterior libraries

Main problems faced

Our first issue was extracting the 2016/17 data after it had been passed through the json parser. This was given in structure form, with sub structures, which meant that a knowledge of fields and how to access them was required. This was solved by understanding the format in which the data was presented and then creating a loop which extracted each structure, turned it into a row and added it to a table each row at a time. We then had to manipulate the data to make it usable for our optimisation such as removing empty cells turning the numbers into “number” form.

We also needed to create points for each individual player due to the fact that the points were not always distributed uniformly across position/minutes, for example clean sheet points aren’t awarded for midfielders and forwards. This was solved by creating a “for” loop that ran through the conditions for every individual row. Due to nature of some point scoring categories, such as the number of times each player conceded over 2 goals, we had to use overall team data to create an estimate for each individual. Approximations such as this mean that there is likely to be a degree of inaccuracy in our results in the 2015/16 case.

The toughest coding aspect of our project was addressing the club constraint issue. We needed to use an infinite loop as sometimes there were more than four players from each club (i.e. running it through once wouldn’t reduce it to the maximum three). The difficulty was how to exit this loop when it was completed. This issue was solved by creating a temporary team (by copying the initial team) and running a “for” loop that checked the club constraint condition. The appropriate changes were then implemented to satisfy the constraint. The temporary team, with the changes applied, was then compared to the original final team and if these were the same then there was no violation of the 3 players per team rule. Therefore, it exited the loop.

There was one final unforeseen problem that presented itself after the code for the optimisation had already been completed and tested. Initially we had no issues running the scrape part of the code however, after an update one following a round of fixtures, the code began to run through all options for a certain position (forwards), turning out a team that had the ***worst*** two players in terms of average points as part of the team. During this particular week the forwards had performed poorly in terms of points and, through our process, were consistently being sorted as having the lowest average points and being replaced. Eventually the code ran through all the options then failed.

Addressing this wasn’t straightforward. After attempting multiple fixes by adapting the existing code we realised a completely different approach was needed. Chris added a ‘get out clause’ into the code which would exit the loop if it ran through every option and then to solve the problem, created a new piece of code which it would run through if and only if it had exited the initial loop.

This works the same for every position but I’ll use the example of ‘forwards’ to describe how it fixes the issue. The secondary loop works by pre-setting the forwards as the best three forwards (sorting by average points) and setting these as given. There is then a loop which solves for the remaining three positions (in this case defenders, midfielders and goalkeepers), using the budget constraint as ‘100 – the sum of the given forwards’ in the same way that it optimises usually – sorting by average points and replacing the lowest scoring.

This isn’t a complete fix; it sidesteps the issue by setting the best players in the problem area, which may not be the best use of the budget. That being said, it ensures that the code doesn’t fail and still creates an excellent, albeit not perfect, team. The issue presenting so late was also a factor in this. In the remaining time it wasn’t practical to rewrite the entire optimisation method based on an issue which may not be replicated after the next fixtures, especially when we could get extremely close by pre-setting the problem area.

Results

The obvious limitation of our code, that we haven’t explicitly pointed out, is that it works retrospectively, it uses existing data to generate the optimal team. The obvious next step for this code is to expand on the scraped data, creating an algorithm that takes into account the next opposition and their statistics (propensity to concede, score etc.). This is likely to be imprecise and extremely time consuming and probably better suited to another programming language.

However, for all of the limitations, ultimately we feel we were successful in applying the skills learnt throughout this module by recreating a Fantasy Football Team. We have produced a code that met our objective of generating an optimal team subject to the constraints given by Barclays Fantasy Football.

While on the face of it we have simply ‘chosen the best team’, once you delve into the process of writing code to automate those decisions, it is and was an extremely difficult and time-consuming undertaking. Among other things, we learnt to use a GUI, learnt to understand and transform structures and sub-structures as well as using the concept of ‘scraping’, a process which will keep our code relevant and useful in the future.

We can be, and are, pleased with the outcome, especially because our code was in fact such a success that the 2015[[1]](#footnote-1) team calculated surpassed the highest points total achieved last season and the 2016 team would currently be winning the English and Worldwide leagues[[2]](#footnote-2).

1. <http://www.fantasyfootballscout.co.uk/2015/05/29/interview-simon-march-fantasy-premier-league-winner/?hc_page=1&hc_sort_by=comment_date#hc_comments> [↑](#footnote-ref-1)
2. <https://fantasy.premierleague.com/a/leagues/standings/313/classic>

   <https://fantasy.premierleague.com/a/leagues/standings/261/classic> [↑](#footnote-ref-2)