School of Physics and Astronomy

Senior Honours Project
Computational Physics

Analyzing a ranking system

Justs Zarins

March 2013

Abstract

Declaration

I declare that this project and report is my own work.

Signature: Date:

Supervisor: Prof. Graeme Ackland 10 Weeks

Contents

1 Introduction

Objectively comparing and ranking some set of entities is an important challenge. Consumers are daily tested to choose the right product, companies look for the most fruitful strategies to pursue, online search engines order query results by relevance and competitors seek to find who is the best. If there are clear criteria for comparison, ranking is straightforward. For example, marathons are always the same distance and with similar surface properties and topology. Some additional details may need to be taken into account like different age groups of the runners, but in general it is sufficient to compare the competitors' run rimes (historic or recent) in order to establish their ability. However, not all ranking problems present clear criteria. In team sports and multi-player video games the landscape is more complicated. How do individual player contributions accumulate to reach the eventual victory or defeat? Is a team that barely wins or one that wins by a large margin better in the long run? [.....] One way to approach such systems is to abstract away from specific details and employ statistical methods. This report will examine the ranking scheme used in British orienteering competitions through computer simulation of ideal data. The scheme will then be applied to real race data and improvements will be sought.

2 Background or Theory

3 Method or Strategy

All simulations and analysis scripts are implemented in Python 2.7. The Numpy library is used for certain numerical tasks and Pyplot is used for plots and histograms.

3.1 Simulation

All simulation code follows the following pattern.

- Create a collection of runners and assign initial scores.
- (optional) Assign each runner ability.
- Execute the update loop m times.
    1. Select a group of runners from the whole list.
    2. Generate run times.
    3. (optional) Mark slowest 10% to be excluded from calculations.
    4. Calculate and apply score changes as per formula [???].

Two types of score initialization were considered: all starting from the same score of a 1000 or assign scores based on a a Gaussian peaking at 1000 and with standard deviation of 200. In the case of uniform score assignment care needs to be taken in the code to handle the case when SP is zero (all runners have the same score) leading to division by zero. The code was progressively expanded to add more details about the

runners. Initially there was no distinction between the competitors, later each runner was assigned an intrinsic ability and variability of that ability. The underlying ability distribution is modeled as each runner having a mean time. In reality, courses have different lengths, hence they take a different time to complete. The model used here corresponds to different subsets of all competitors running the same track in their own mean time. This should not be an issue because run times are essentially normalized when scores are calculated because only deviations from the mean time are relevant. The variability of a runner's performance is represented by a mean number of mistakes they make in a race. Every time they compete, a Poisson random variable with a mean of that runner's specific number of mistakes is drawn. This is multiplied by a mistake weight factor and added to their run time. [+- mistakes as well...]

4 Results & Discussion

This section should detail the obtained results in a clear, easy-to-follow manner. Remember long tables of numbers are just as boring to read as they are to type-in. Use graphs to present your results where -ever practicable. When quoting results or measurements **DO NOT FORGET ABOUT ERRORS**. Remember there are two basic types of errors, these being random and systematic, which you must consider. Remember also the difference between an error and a mistake, computer program bugs are mistakes.

Again be selective in what you include. Half a dozen tables that contain totally wrong data you collected while you forgot to switch on the power supply are **not relevant** and will frequently mask the correct results.

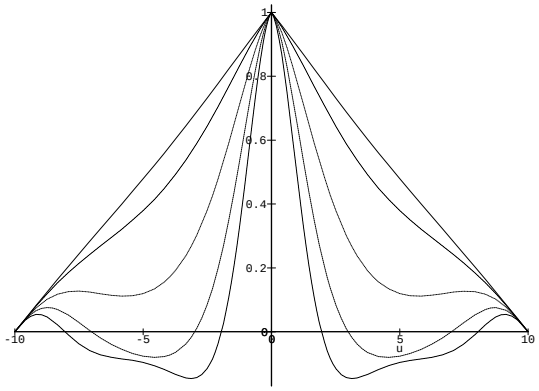

Figure 1: This is an inserted Postscript file

This section must contain a discussion of the results. This should include a discussion of the experimental and/or numerical errors, and a comparison with the predictions of the background and theory underlying the techniques used. This section should highlight particular strengths and/or weaknesses of the methods used.

5 Conclusion

This section should summarise the results obtained, detail conclusions reached, suggest future work, and changes that you would make if you repeated the experiment. This section should in general be short, 100 to 150 words being typical for most projects.
If you have opted to have multiple **Theory, Method, Results** sections, draw all the results together in a **single** conclusion.

6 References

Don't forget this section. Detail the relevant references which should be cited at the correct place in the text of the report. There are no fixed rules as to how many references are *needed*. Generally the longer the project, and the more background reading you had to do, the more references will be required.

When you cite a reference you must give sufficient information. For example, for a journal article give, *Author*, *Title of article*, *Journal Name*, *Volumn*, *Page*, and *Year*, while for a book give, *Author*, *Title*, *(Editor if there is one)*, *Publisher*, and *Year*.

A Appendices

Material that is useful background to the report, but is not essential, or whose inclusion within the report would detract from its structure and readablity, should be included in appendices. Typical material could be diagrams of electronic circuits built, specialist data tables used to analyse results, details of computer programs written for analysis and display of results, photographic plates, and, for computational projects, a copy of all written code.

Again be selective. The appendix is **not** an excuse for you to add every last detail and piece of data, but should be used to assist the reader of the report by supplying additional material. Not all reports require appendices and if the report is complete without this additional material leave it out.