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

Clear criteria for arranging a group of entities in order of quality are not always available. In such cases it is desirable to infer the ranks from results of comparison events between the entities. The goal is to find each participant's skill level and to remove as much noise from it as possible. This will be referred to as skill based ranking.
A good starting point of skill based ranking is the Elo rating system[1]. It was developed by Arpad Elo and found its primary use in comparing chess players. The basic principle of the system is that each player's skill can be represented by a Gaussian curve. The player's skill is thought to be the mean of the curve and variation to it is allowed by the sloping sides. When two players enter a match, the likelihood if outcomes can be found by looking at the difference of both players' skill curves. The rank change of both players after the game are proportional to how unlikely that outcome was. The system has been applied to other sports as well, for example football[2].
A weakness of the Elo system is that it assigns an equal uncertainty to all players' ranks, this is addressed in the Glicko system[3]. In this system a person who plays frequently is subject to smaller rank changes than someone who plays less frequently. As a player accumulates more and consistent games, the standard deviation around their mean rank shrinks.
A further extension to multiplayer games is Microsoft's TrueSkill which was developed for the Xbox gaming system[4]. The principle is very similar to Glicko, expected outcomes leave ranks almost unchanged while upsets cause big changes. Teams of players are assigned aggregate mean skill and uncertainty, the outcome of the match usually causes different individual rank changes for each player. Where more than two teams or players are competing all against each other, the participating entities are compared pairwise. The system also includes iterative steps that run the same scores through the system multiple times and different weights for game types based on how big of a role

luck and skill plays. The exact details of the algorithm are rather involved and excellent explanations and a sample implementation can be found in other resources([5], [6]). This ranking system has an elaborate theoretical basis and has shown good predictive power[4]. An interesting expansion of TrueSkill is AllegSkill[7] where additional provision exists for team commanders' skills.
A shared feature of the discussed ranking systems is that they are geared towards evaluation one versus one competitions. While all versus all ranking applications are possible, in practice they are executed as a series of one versus one comparisons. The ranking scheme used by British Orienteering Federation, the focus of this report, is an all versus all system in essence. The central equation of interest is the score assignment formula[8]:

$$RP = MP + \frac{SP * (MT - RT)}{ST} \tag{1}$$

Where R stands for run, P for points, T for time, M for mean and S for standard deviation. The formula says that the number of points awarded to a runner in a race is the mean number of points of the runners participating in that race plus the standard deviation of the runners' points multiplied by the number of standard deviation that the runner being awarded is away from the mean time of all runners' times. The best guess of a runner's skill is the mean of their scores obtained races (published scores usually the mean of a subset of all scores).
It is tempting to include in the discussion the central limit theorem (CLT) which states that the distribution of means sampled from almost any distribution will tend to a Gaussian[9]. However, the theorem requires independent samples drawn from an identical distribution. In this case the distribution of scores changes after each recorded race, which also implies correlation between subsequent scores. Thus, while the CLT might apply in some states, in general it cannot be strictly applied to this system.
Another concept that must be mentioned is self consistency. One application of this principle is the Hartree-Fock algorithm. In the method a trial wavefunction is used to calculate the energy of a complicated quantum system. This energy is then used to adjust free variables in the wavefunction, which is then used again to gain a new estimate of the energy. The loop is continued until new approximations do not appreciably change, the system is then reached self consistency. This principle could be applied to orienteering ranking by putting race results back though the system multiple times, each time potentially improving the estimate of the players' ranks. It would be reassuring if the system eventually settled on a stable state instead of changing ranks continually.
Because systems similar to the one used by the British Orienteering Federation are not widely discussed in scientific literature, an approach based on simulations and experiments will be used to analyse the ranking scheme.

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. Most of the data analysis functions are implemented in a separate module. The main script then calls them in order according to user-defined control variables. (see SOME APPENDINX for implementation)

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 following 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. In the case of no innate ability, run times are generated from a Gaussian curve with a random mean and standard deviation as one tenth of that mean.
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...]

3.2 Analysis of Real Data

A pre-processing step was performed to speed up subsequent data processing. Outside of the main program, incomplete entries were removed it would not be possible to gain information from them later.
The main script follows the following pattern.

- Process the data file and produce a list of races and a list of all participants.
- Assign runners initial scores.
- For each race in the list of all recorded races:
    1. (optional) Identify and deal with outliers.

2. Calculate and apply score changes as per formula [???].

The file processing step parses the input CSV (comma separated values) file and assembles a list of races where each race is a list of a participant, run time pairs. In order to conform to the BOF rules, some data is discarded at this stage if the course is of type "yellow" or "white", if the participant is under the age of 16 or if the race has less than 10 participants. Also, run times that are 0 seconds long are discarded as they are obvious errors in recording. Still, other less-clear outliers remain.
Initial scores, same as before, can either be some constant value for everyone or a normally distributed random variable.
Possible outliers can be identified as either the slowest 10% or times that are some number of standard deviations away from the mean of all times. They can then be either excluded from calculations of MT, ST, MP and SP or removed altogether giving them no points for the race.

3.3 Reruns

It is also possible to run all scores through the score assignment system multiple times. This would ideally lead to decreasing rank adjustments each rerun as the players' true score becomes better approximated each cycle.

4 Results & Discussion

4.1 Simulation results

The first simulated case is that of there being no difference between the runners (see Fig 1). Both initial and final distributions have a mean of 1000; the initial standard deviation is 200 and the final one is 2.3. There are 10000 participants which competed in 30000 races with a maximum of 100 participants per race.
The general trend observed here was that the participants' final rank distribution is described by a sharply peaked curve with the mean matching the one of the initial rank distribution's mean. As more races are executed, the standard deviation of the ranks becomes smaller (Fig 2). This can be explained by noting that the runners have no intrinsic ability and their race times are generated at random. Over time the ranking scheme is able to deduce that all runners have the same skill level, despite being assigned different initial ranks. As the number of races increases, the system becomes more certain of this result, which is reflected by the decreasing variation in ranks.

Another observation was made that applying the 90% cutoff rule produces a drift of the mean to the left, i.e. in the negative direction. The rule is intended to eliminate outliers, however in this case there are effectively no outlying run times generated. Thus, removing the longest 10% of run times for calculations of MT, ST, MP and SP while still giving points to the excluded runners creates a bias towards the lower scores. The mean run time is artificially lowered which leads to the excluded players receiving an inappropriately lower score which in turn leads to the whole rank distribution to drift to the left over time. The decrease levels off as the standard deviation of the scores becomes

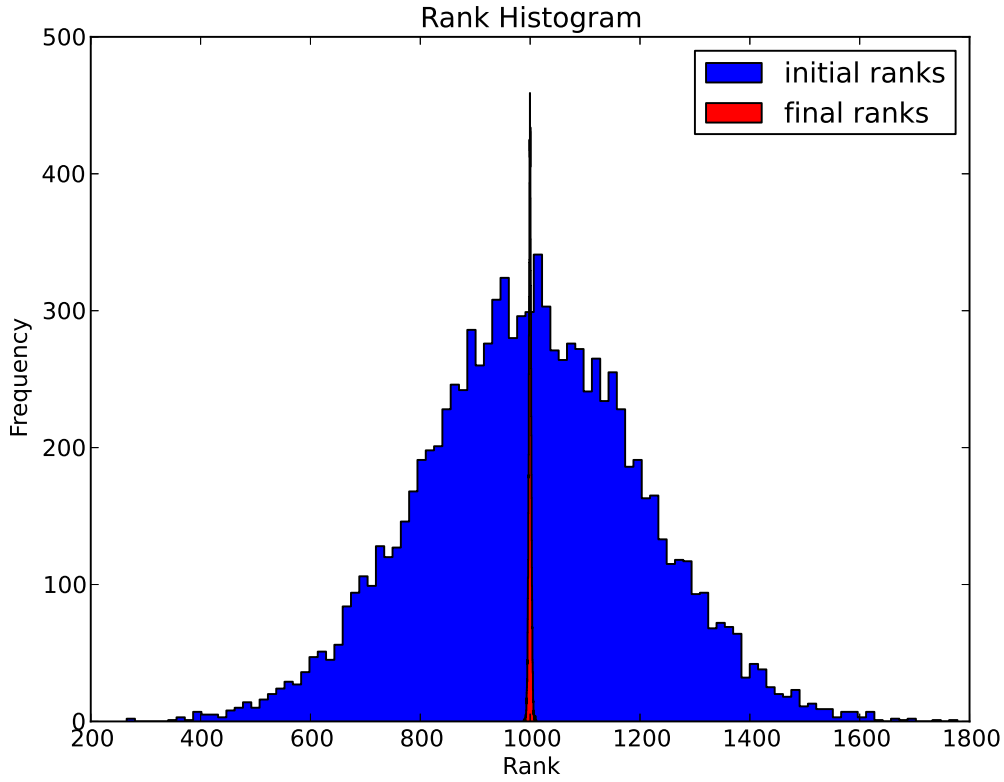


Figure 1: Rank histogram of initial ranks and final ranks for runners with no intrinsic ability. Note: bin widths are different for the two histograms.

smaller (see Fig 3).

Next, runners with differing intrinsic abilities were put through the system. To present the comparison between the runners' innate abilities and the rank assigned to them by the system, plots these two quantities against each other are used. For a successful ranking attempt it is expected to see a straight line. It If the runners make no mistakes, the system was found to correctly identify runners' abilities. Figure 4 and Figure 5 show rank-ability plots for Gaussian and constant initialization respectively. In both cases the system identifies runners' intrinsic abilities correctly. However, some data (about 3%) have a comparatively high standard deviation. Examining these data points revealed that they usually have one race early on that produced a distant outlier, possibly by putting them in a poorly matched race. Rerunning the simulation with the previously estimated ranks as the new initial ranks removed all outliers and produced consistent standard deviations. It is interesting that this effect did not arise when the system was initialized with Gaussian ranks. This was probably caused by "smoothing" of SP so that very good runs did not earn an unreasonably high amount of points.

Increasing the propensity of runners to make mistakes, i.e. increasing noise, makes it much more difficult to distinguish runners. The rank-ability plot bulges out and data

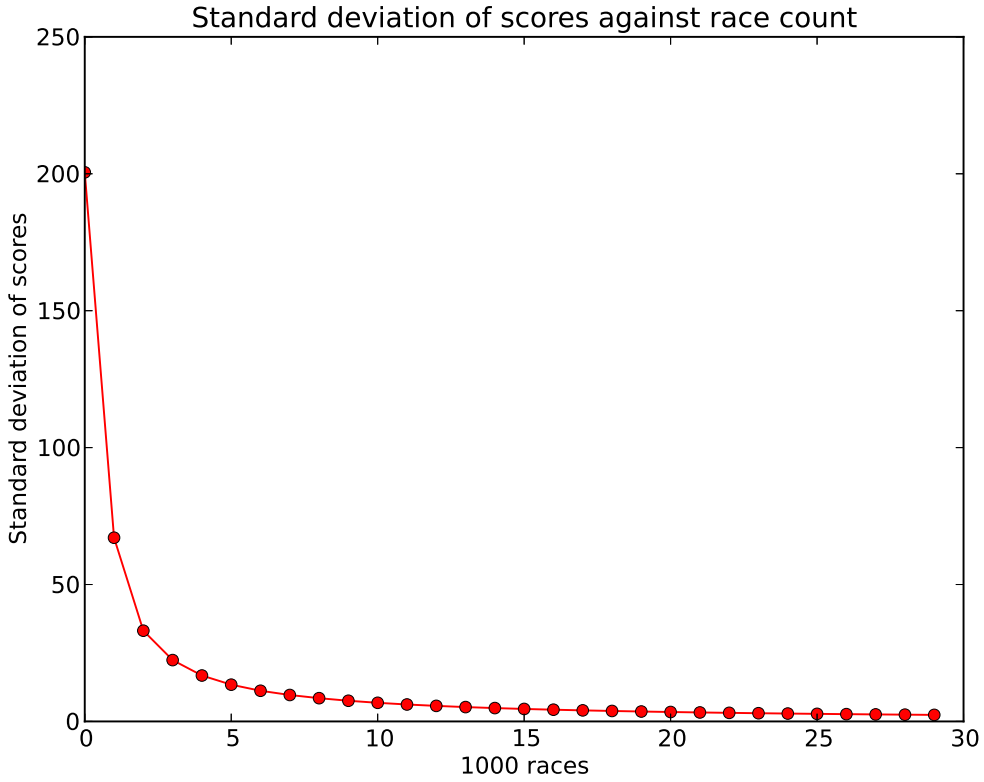


Figure 2: Standard deviation of means against race count for runners with no intrinsic ability.

points have high standard deviation (Fig 6). Still, the general linear shape is retained so runners with significantly different abilities are correctly ranked with respect to each other. Increasing the number of races improves the ordering of runners, but the noise remains.

Rerunning the same data multiple times (while keeping the total number of races the same) did not improve runner rank estimation. This can be explained by noting that generating unique races creates new information and thus improves the estimate of the mean. It was noted that the standard deviation of ranks would shrink when rerunning the same sequence of races multiple times. To explore this effect, runners with no mistakes were looped 10 times with 5000 races each loop. The mean and standard deviation of scores leveled off at 1000 and 38 respectively while the mean of standard deviations of individual racers' scores went to 0. The mean and individual deviations thus behaved correctly. It is unclear why the rank deviation went to 38 but at least it was correctly identified that all runners are not the same.

To continue, runners with randomly given error capability were run through 10 loops as well. The mean of ranks remained at 1000 but both rank standard deviation and individual standard deviations went to 0. To examine this, the scores were rebased after every except the last run to have mean 1000 and standard deviation 200. Now the standard deviation of ranks went to 90 and that of individual scores to 85, but increasing

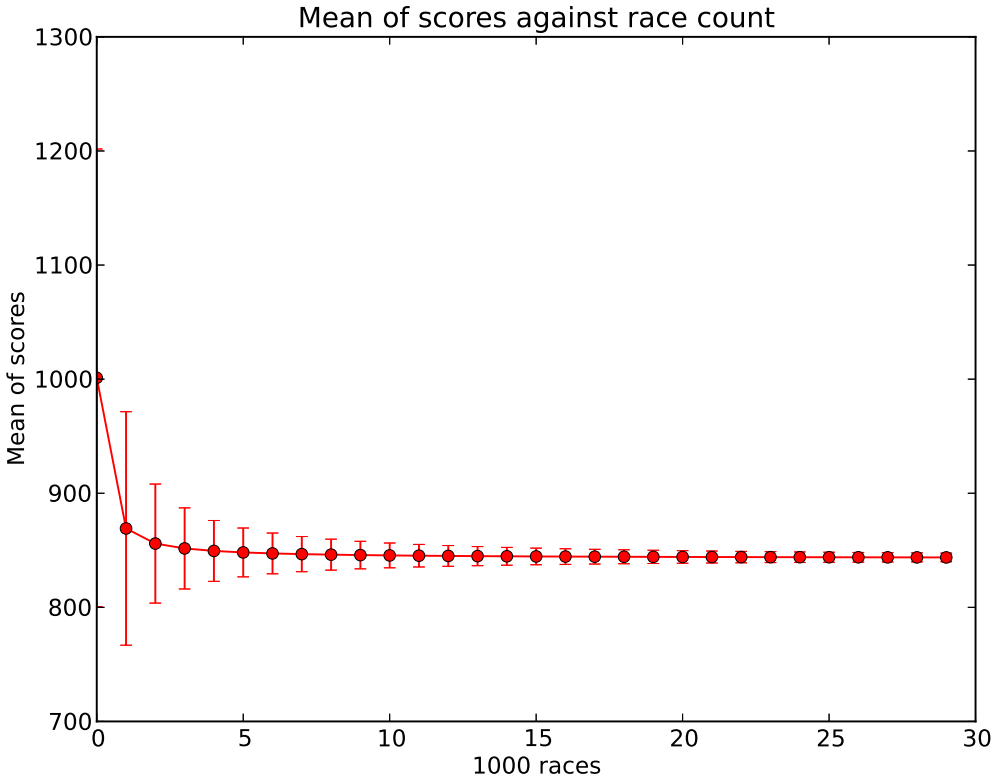


Figure 3: Mean of scores against race count for runners with no intrinsic ability. The 90% cutoff rule is applied. Error bars show standard deviation.

the number of races in each loop made these fall further. This indicates that all rebasing does is reset the initial standard deviation which then falls to where it intends to be.
It is unclear what exactly is causing the shrinking distribution. It seems to echo the Central Limit Theorem which, for a simple case like averaging coin tosses, would predict that the mean of averages would converge to a sharp peak. Still, the rank-ability plots do not change qualitatively so the ordering of players, which is the ultimate goal, is not impeded. For practical reasons, though, it is desirable to rebase scores between reruns as doing so when the standard deviation falls very low can be difficult due to division by a very small number.

4.2 Data analysis results

Taking some of the observations from simulated data, a first look at the real British Orienteering data was performed. The 90% cutoff rule was not applied as it often removes legitimate results, causing the distribution to drift significantly over time. Another concern with real data is poor sampling of some individuals. In the generated case runners were entered into races at random so with enough total races everyone was sampled well. In reality people have different levels of motivation, some participate frequently while

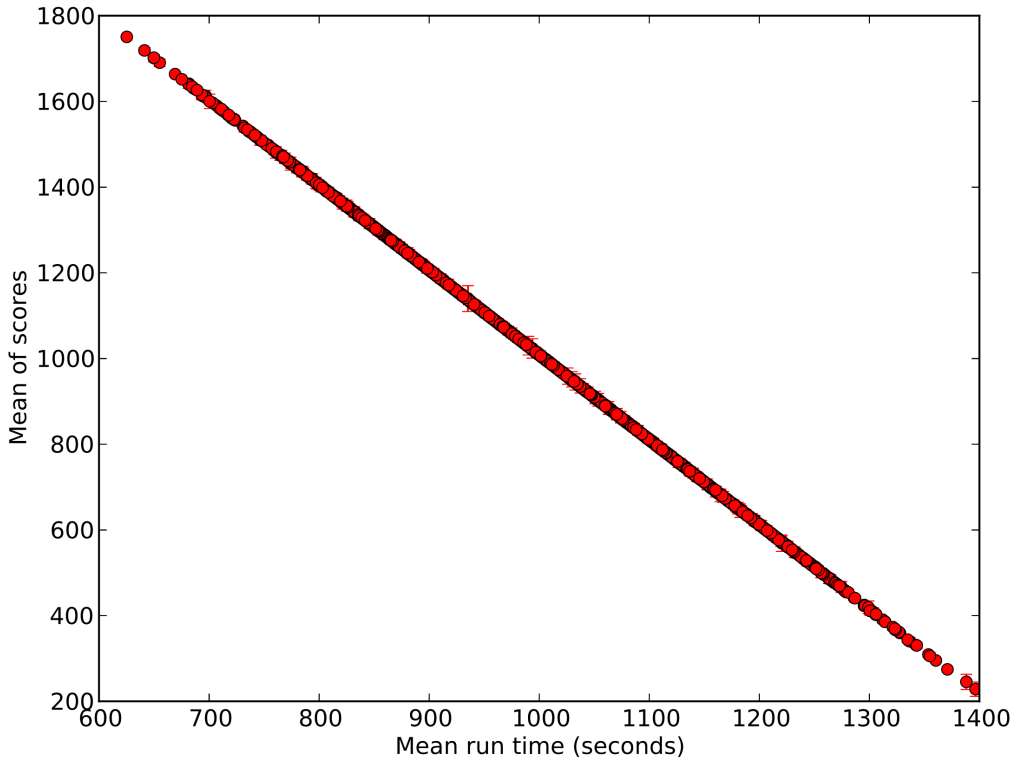


Figure 4: Mean of awarded scores plotted against mean run time(ability). Initial ranks were distributed from a Gaussian with $\mu = 1000$ and $\sigma = 200$. Final ranks have $\mu = 1006$ and $\sigma = 199$. Error bars show standard deviation, standard errors are too small to see.

others give up after one race. Both would be entered into the database. Thus in the final rank distribution only racers that have participated in 6 or more events were included. This choice was made arbitrarily to match the the fact that published scores are often the mean of a runner's best 6 scores. Figure 7 shows the resulting rank distributions for the two types of initialization. In both cases the distributions have tails on the low rank side (THIS IS EXPECTED?). Constant initialization produces a smoother distribution than Gaussian initialization, however it also produces distant outliers. Similarly as before, these runners were found to have a particularly low or high first few scores that biased their mean score. However, rerunning the the data again with the improved initial scores did not remove these outliers as was the case for simulated data. This suggests that the bad scores are based on actually outlying run times. To address this, a Gaussian cutoff rule was applied that ignores run times more than 2σ outside the mean. Applying this rule and 2 runs through the data removed the high point outlier but the low ones remained. The outliers on the low end kept changing so it appears that there truly are some poor runners that belong there. However, the outlier at the top was the same person and it was observed that none of their runs were eliminated by the cutoff, but the initial high score disappeared still. Thus it can be concluded that the high outlier was caused by someone else's anomalous run time.

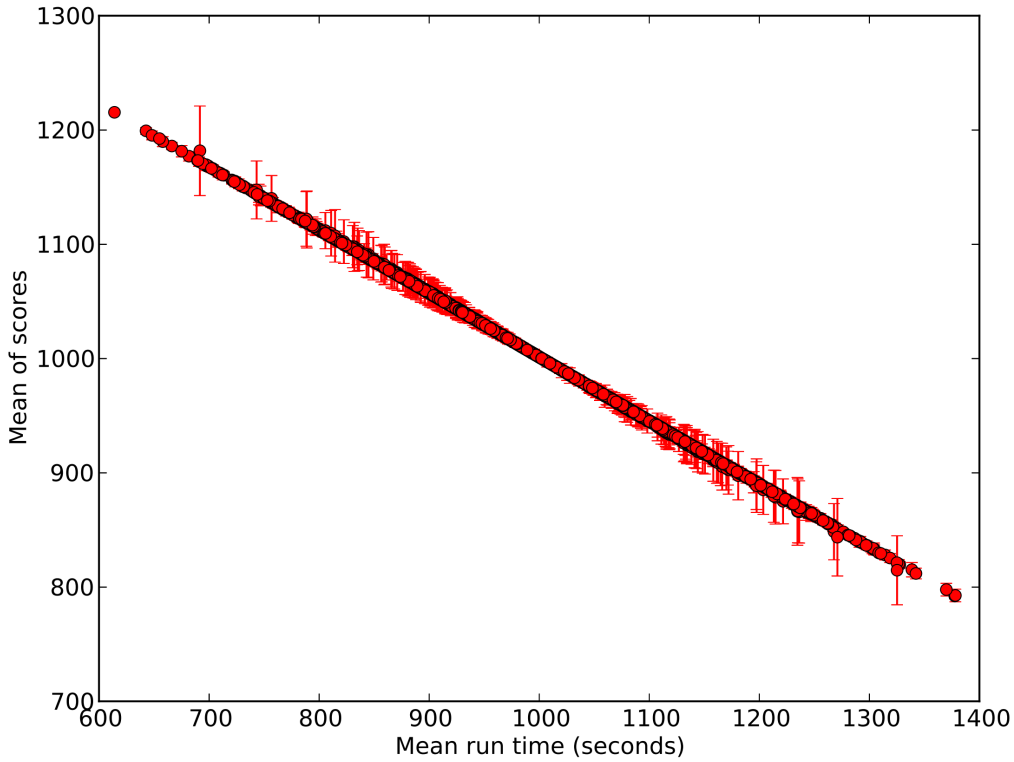
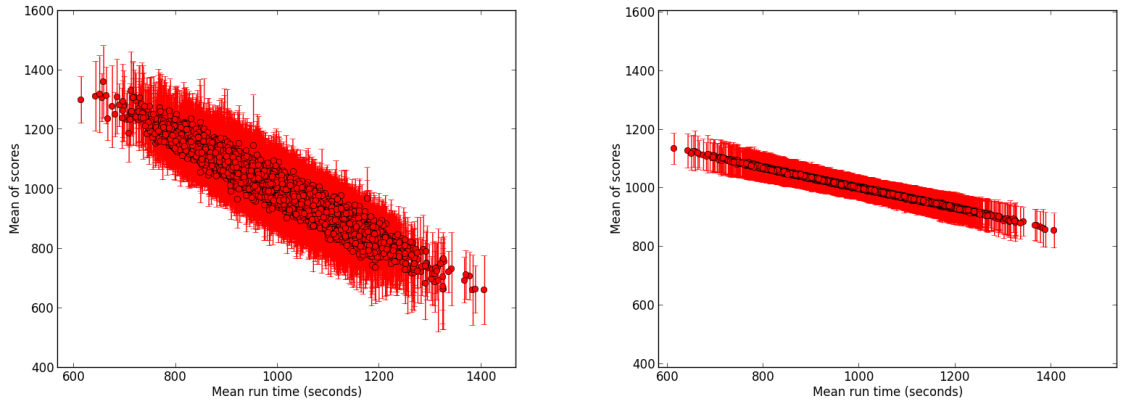


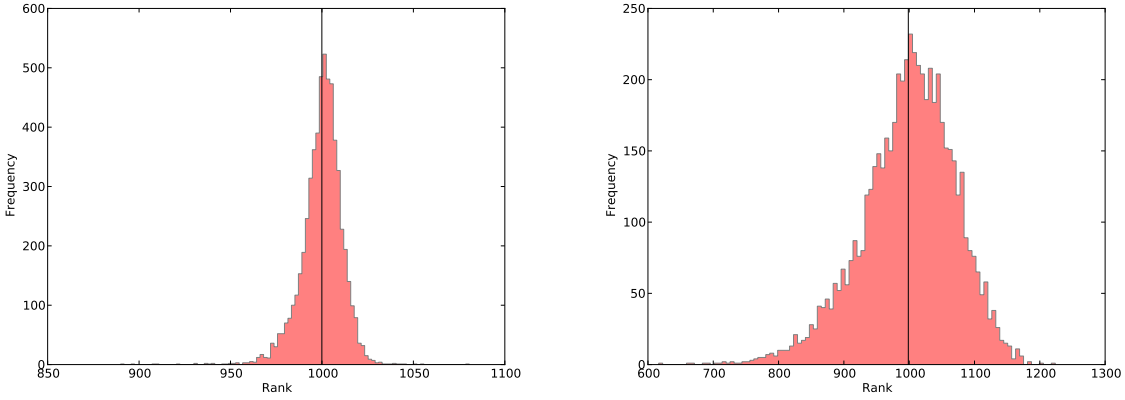
Figure 5: Mean of awarded scores plotted against mean run time(ability). Initial ranks were all 1000. Final ranks have $\mu = 1000$ and $\sigma = 56$ Error bars show standard deviation, standard errors are too small to see. Points with high standard deviation represent about 3% of the data.

Next, the number of reruns were increased to 10. It was observed that the mean would drift linearly to the left. The Gaussian cutoff rule should be more careful than the 90% rule in cutting out legitimate data. It then appears that the deeper cause of the drift is cutting out data from calculations of MT, ST, MP and SP while still assigning points to all racers. Repeating the run but this time not giving points to the outliers removed the drift previously observed.
Another observed issue was the shrinking of the standard deviation of all ranks. This was accompanied by a small and decreasing drift of the mean to the right (previously it was to the left). If the system is trying to reach a steady state, the decrease of standard deviation means that any changes to rank also become smaller. Thus a lower limit on SP was imposed. [forgot to talk about max individual stds! those level off. individual rank changes also seem to level off. [without lower SP limit everything still falls to the floor]] [need to talk about the convergence, ie max/avg rank change]



(a) 3k races (b) 100k races

Figure 6: Mean of awarded scores plotted against mean run time(ability) for 2 different numbers of total races. Initial ranks were distributed from a Gaussian with $\mu = 1000$ and $\sigma = 200$. Final ranks have (a) $\mu = 1000$ and $\sigma = 92$ and (b) $\mu = 1000$ and $\sigma = 34$. Error bars show standard deviation, standard errors are too small to see.



(a) (b)

Figure 7: Rank histogram for real data with 2 different initialization types: (a) all 1000, (b) Gaussian with $\mu = 1000$ and $\sigma = 200$. Final ranks have (a) $\mu = 1000$ and $\sigma = 11$ and (b) $\mu = 999$ and $\sigma = 72$. The black vertical line marks the mean.

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.

References

[1] A. E. Elo. *The rating of chessplayers, past and present.* Arco Publishing, New York, 1978.

[2] Lars Magnus Hvattum and Halvard Arntzen. Using elo ratings for match result prediction in association football. *International Journal of Forecasting*, 26(3):460 – 470, 2010. Sports Forecasting.

[3] Mark E Glickman. The glicko system. *Boston University*, 1998.

[4] Tom Minka Ralf Herbrich and Thore Graepel. Trueskill: A bayesian skill rating system. *Advances in Neural Information Processing Systems*, 19:569, 2007.

[5] Trueskill ranking system: Details. http://research.microsoft.com/en-us/projects/trueskill/details.aspx. Accessed: 23/03/2013.

[6] Computing your skill. http://www.moserware.com/2010/03/computing-your-skill.html. Accessed: 23/03/2013.

[7] Allegskill. http://www.freeallegiance.org/FAW/index.php/AllegSkill. Accessed: 23/03/2013.

[8] The ranking system. http://www.britishorienteering.org.uk/images/uploaded/ downloads/events_appendix_k.pdf. Accessed: 23/03/2013.

[9] Meyer Dwass. *Probability: Theory and Applications.* W. A. Benjamin, Inc., New York, 1970.

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.