School of Physics and Astronomy

Senior Honours Project
Computational Physics

Analyzing a ranking system

Justs Zarins
March 2013

Abstract

Simulations and data analysis were used to examine and evaluate the ranking system of British Orienteering. Generated and real data produced plausible rankings giving confidence to believe that the system is generally accurate and independent of initial setup. An issue of the rank distribution shrinking during repeated run-throughs of race data remains unresolved and impedes understanding of the system's convergence.

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 a set of entities is an important challenge. Consumers are daily tested to choose the right products, 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? These questions hardly have a conclusive answer. There are also added complications because people act differently depending on the nature of the competition. For instance, students in exams generally attempt to do their best without directly knowing how well their competition is doing, yet in the end it is their relative rather than absolute performance that matters most. In contrast, a marathon runner might go for their personal best, try to set a world record or just stay a few seconds ahead of their pursuer. These and other approaches reflect a person's underlying ability differently.
One way to approach complicated systems is to abstract away from specific details and employ statistical methods; increasing sophistication does not always produce increasingly accurate results. A ranking scheme of this type is employed by the British Orienteering Federation (BOF) to assign points to orienteering event participants. The aims of this project is to evaluate the BOF ranking scheme for accuracy and to uncover the cause of some of its quirks like the drifting of the mean of ranks over time. The problem will first be approached using computer simulated races and later the scheme will be applied to real race data.

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 include 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. When ranking, samples are being essentially from the distribution of abilities. Complications are added by the fact that this distribution changes over time and samples some times correlate if runners interact. Also, the ability is interpreted via the score assignment formula and is not drawn directly. Thus, while the CLT might apply in some states, in general it is difficult to rigorously apply it 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 has then 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 analyze the ranking scheme.

3 Method

The simulations and analysis scripts are implemented in Python 2.7. The Numpy library is used for certain numerical and statistical 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 sequence according to user-defined control variables. Details of simulation and analysis code follow.

3.1 Simulation

All simulation code follows the following pattern.

- Create a collection of runners and assign initial scores.
- (optional) Assign each runner a mean run time which represents their innate 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 according to Equation 1.

Two types of score initialization were considered for most simulations: all starting from the same score of a 1000 or assigning scores based on a a Gaussian with mean 1000 and standard deviation 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. In this project the default value $SP = 200$ was used if all runners entered with the same score.
In order for the system to have only limited information about the whole set of runners, thus closer resembling real racing, random subsets of 10 to 100 people were chosen to participate in each race. The total number of participants was chosen to be 10000. The number of races m varied between experiments and is specified together with any presented results. With these settings each person runs in $0.0055m$ events on average.
The code was progressively expanded to add more details about individual runners. Initially there was no distinction between the competitors and run times for each event were generated from a Gaussian curve with a random mean and standard deviation as one tenth of that mean. In later experiments each runner was assigned an intrinsic ability

and variability of that ability. Runners were given a mean race time drawn from a normal distribution with $\mu = 1000$ and $\sigma = 100$. Uncertainty was represented by adding a Gaussian random variable to the mean time with $\mu = 0$ and $\sigma = 10 * v$ where v is a variability measure. $v = 10$ was used for results presented in this report.
In the case of intrinsic ability, it might cause concern that the underlying ability distribution is modeled as each runner having a mean time drawn from a single distribution. In reality, courses have different lengths, hence they take a different time to complete on average. This should not be an issue, though, because run times are essentially normalized when scores are calculated because only deviations from the mean time are relevant.
In step 3 of the race generation loop, a cutoff rule was applied for some experiments. With the rule active 10% of the longest run times were excluded from calculations of MT, ST, MP and ST (from Eq 1) but the runners were still given points, in accordance with the BOF instructions[8].

3.2 Data Analysis

The main analysis 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) Mark slowest 10% to be excluded from calculations.
    2. Calculate and apply score changes according to Equation 1.

The data was provided by BOF as a CSV (comma separated values) file that contained information about every race recorded. A pre-processing step was performed to speed up subsequent data processing. Outside of the main program, obviously erroneous entries were removed. These included entries with missing fields (recording error) or negative values. The next processing step parses the reduced file and assembles a list of races where each race is a list of participant, run time pairs. 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 the data is discarded at this stage, in order to conform to the BOF rules. Also, run times that are 0 seconds long are discarded as they are obviously wrong. Despite these steps other less-clear outliers likely remain.
The script allows to specify a minimum number of races that are needed to be taken into account when producing rank distributions and calculating various statistics about the data. Initial scores, same as before, can either be some constant value for everyone or a normally distributed random variable. The cutoff rule is the same as described in the section above.

3.3 Reruns and Rebasing

It is possible to put all run times through the score assignment system multiple times. This would ideally lead to the system reaching a steady state where any adjustments to the rank distribution are diminishingly small. Whenever reruns were applied, previously earned scores were dropped and their mean kept as the new initial score.
Scores can also be rebased to a different mean and standard distribution while still retaining the relations between all data points. The following procedure was used.

1. Calculate the current mean of scores μc.
2. Calculate the current standard deviation of scores σc.
3. Rebase scores to new mean μn and new standard deviation σn by applying to each score the transformation:

$$score \rightarrow \frac{(score - \mu c) * \sigma n}{\sigma c} + \mu n \tag{2}$$

4 Results and Discussion

4.1 Simulation results

The first simulated case is that of there being no difference between the runners (see Fig 1). 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. This was expected because 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 peak becoming sharp.
Next, the 90% cutoff rule was applied. It was observed that the rule produced a large drift (156 points) of the mean to the left, i.e. in the negative direction (see Fig 2). The drift levels off as standard deviation decreases due to increasing sampling. The underlying cause was identified to be giving points to runners that were beyond the cutoff point. Removing the longest 10% of run times for calculations of MT, ST, MP and ST 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. Applying the same cutoff rule but this time not giving points to the excluded produced the same results as in Figure 1.
Runners with differing intrinsic abilities (see section 3 for implementation details) were put through the system next. To present the comparison between the runners' innate abilities and the rank assigned to them by the system, plots of these two quantities against each other are used. For a successful ranking attempt it is expected to see a straight line with a negative slope (higher mean run time corresponds to lower skill). If the runners make no mistakes, the system was found to correctly identify runners' abilities. Both Gaussian and constant initialization produced the correct plot showing that the system

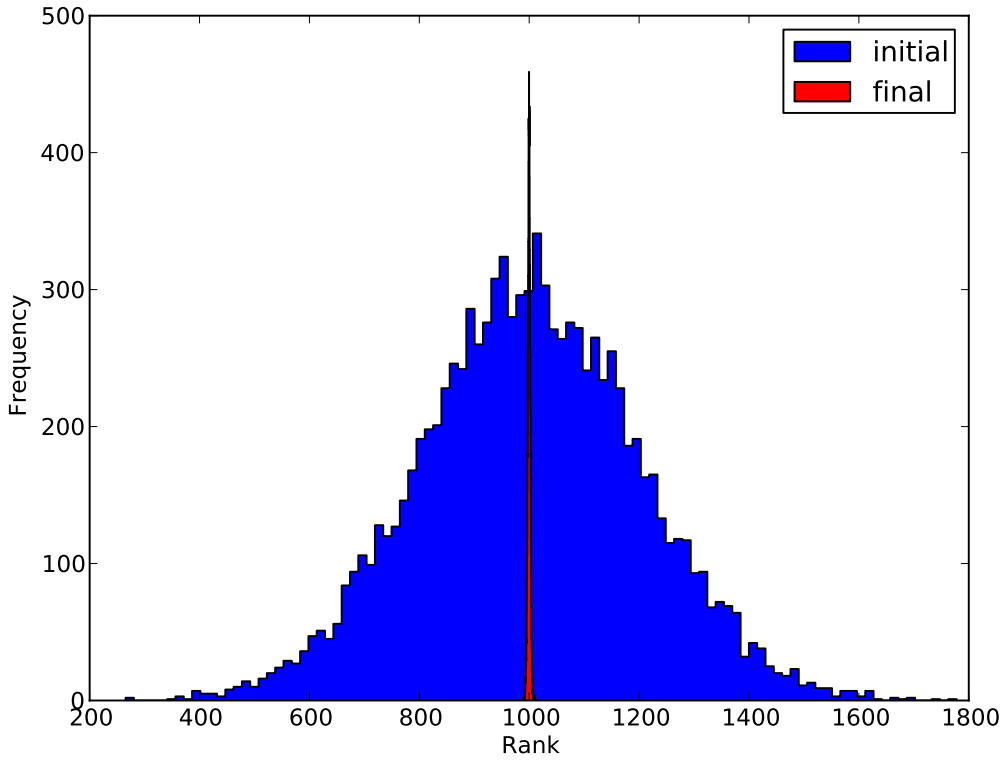


Figure 1: Rank histogram of initial ranks and final ranks for runners with no intrinsic ability. Initial $\mu = 1000$ and $\sigma = 200$, final $\mu = 1000$ and $\sigma = 2$; 30000 races in total. Note: bin widths are different for the two histograms.

identified runners' intrinsic abilities correctly. However, with constant initialization some data (about 3%) have a comparatively high standard deviation(see Fig 3). Examining these data points revealed that they usually have one race early on that produced a distant outlier, probably 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, small standard deviations. It is interesting that this effect did not arise when the system was initialized with Gaussian ranks. This was probably caused by initial "smoothing" of *SP* so that very good runs did not earn an unreasonably high amount of points. These results inspire confidence that the code is correct.
Adding in running mistakes, i.e. increasing noise, makes it more difficult to distinguish runners. The rank-ability plot bulges out and data points have high standard deviation (Fig 4a). 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, as one might expect, but the noise remains (Fig 4b). Interestingly, if the initial ranks were set to correctly reflect the runners' abilities, the rank-ability plot still bulged out as before.
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

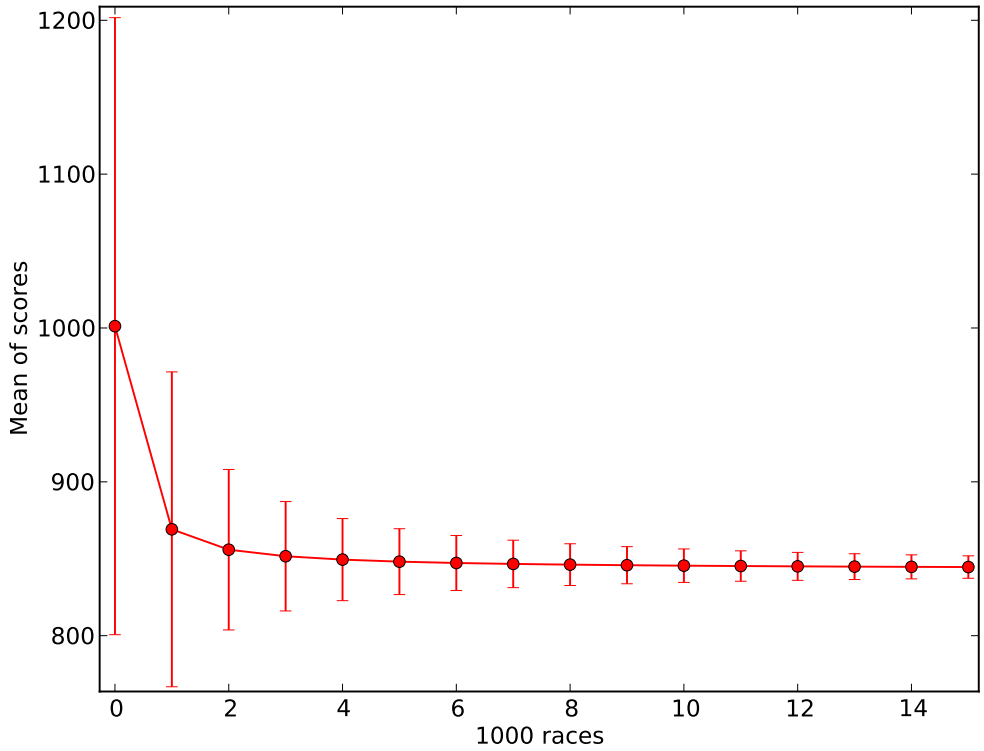


Figure 2: Mean of scores against race count for runners with no intrinsic ability. Initial $\mu = 1000$ and $\sigma = 200$. The 90% cutoff rule is applied. Error bars show standard deviation.

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 rather than 100 which was the standard deviation of the abilities but at least it was correctly identified that all runners are not the same.

To continue, runners with error making capability were run through 10 loops of 5000 races each as well. The mean of ranks remained at 1000 but both rank standard deviation and individual standard deviations approached 0. To examine this, the scores were rebased after every run except the last one 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 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. Different settings showed similar results.

It is unclear what exactly is causing the shrinking distribution. It seems to echo the

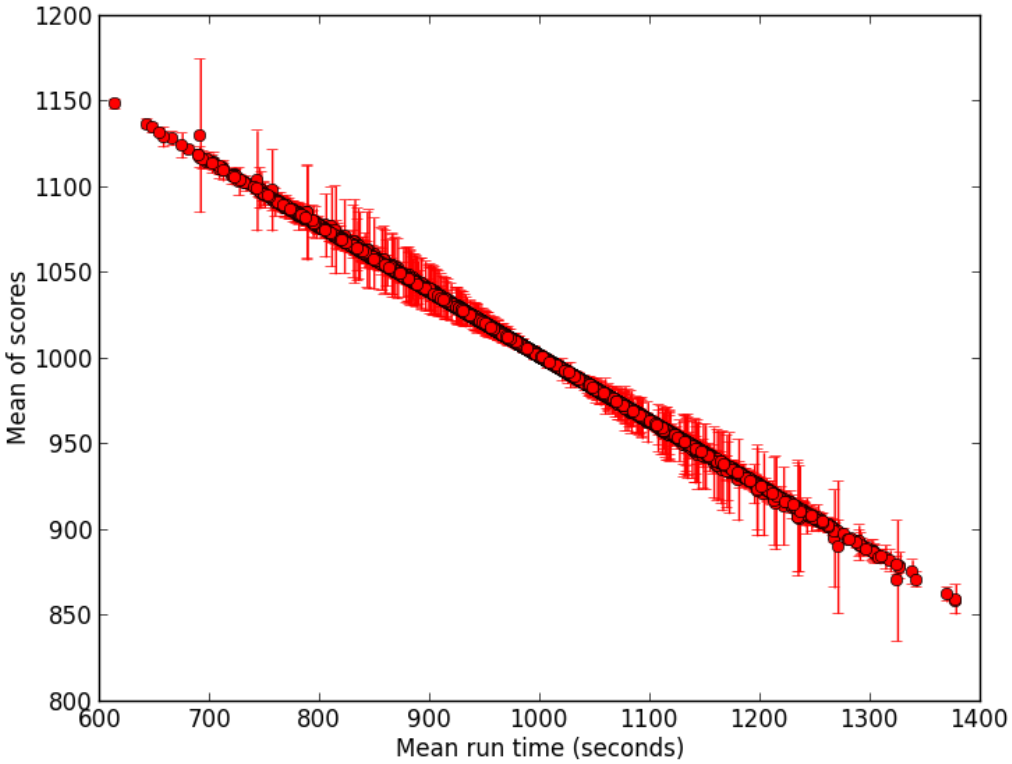
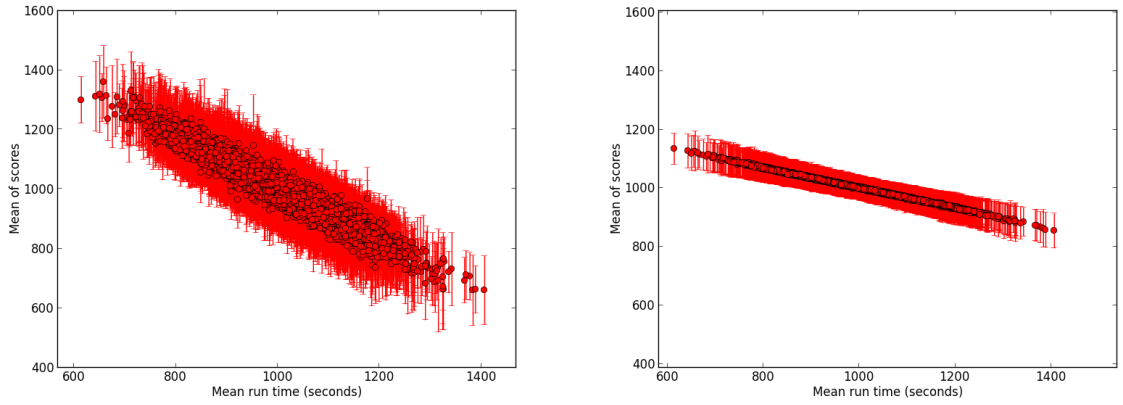


Figure 3: Mean of awarded scores plotted against mean run time. Initial ranks were all 1000. Final ranks have $\mu = 1000$ and $\sigma = 38$; 30000 races in total. Error bars show standard deviation, standard errors are too small to see. Points with high standard deviation represent about 3% of the data.

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. This, however, does not fully explain (i) why the case with no mistakes levels off a $\sigma = 38$ instead of the standard distribution of the abilities(100) and (ii) why introducing even a small mistake propensity makes the distribution of scores shrink. 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.



(a) 3k races

(b) 100k races

Figure 4: Mean of awarded scores plotted against mean run time 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.

4.2 Data analysis results

To get an idea of what the rank distribution should look like, raw race time data was analyzed. For every race time its distance in standard deviations from the mean of that particular race was calculated. The resulting distribution can be seen in Figure 5. It has a tail on the negative side which corresponds to race times that are larger than the mean. This is intuitive as achieving a very fast run is difficult while there is essentially no limit to how slow a race can be done. It is reasonable to expect that the rank distribution would have a similar shape.

Taking some of the observations from simulated data, a first look at the real Orienteering data was performed. The 90% cutoff rule was not applied as it causes the distribution to drift significantly over time. Another concern with real data is poor sampling of some individuals. In the simulations 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 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 means of a runners' best 6 scores. Figure 6 shows the resulting rank distributions. The data was initialized from a Gaussian an was run through twice to reduce the effect of initialization. The ranks display the same tail as the time deviation data as hoped. The fall on the right side is not as fast as that of the the time data. This could be due to only taking better sampled people into account who could be expected to perform better overall than the average due to practice.

Unfortunately, the shrinking of the distribution problem persisted here as with the simulated data. Repeated runs through the data caused the standard deviation of scores to fall each rerun without an obvious non-zero lower limit. Rebasing the scores between runs did not appear to improve the situation. This remains an unresolved problem and

properties of convergence are unclear. Nevertheless, similarity between the time deviation and rank distributions and the earlier results from simulated data provide evidence that the people are still ranked as correctly as possible given limited data.

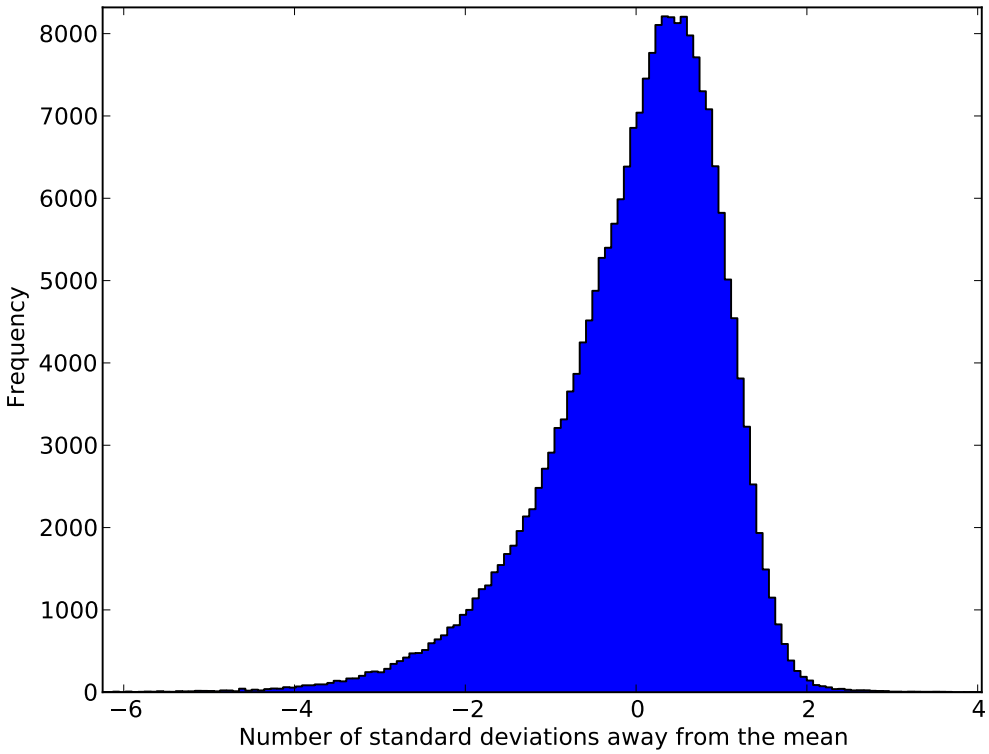


Figure 5: Distribution of all run time deviations obtained from real data. $\mu = 0$ and $\sigma = 1$. Negative deviation corresponds to race times that are larger than the mean.

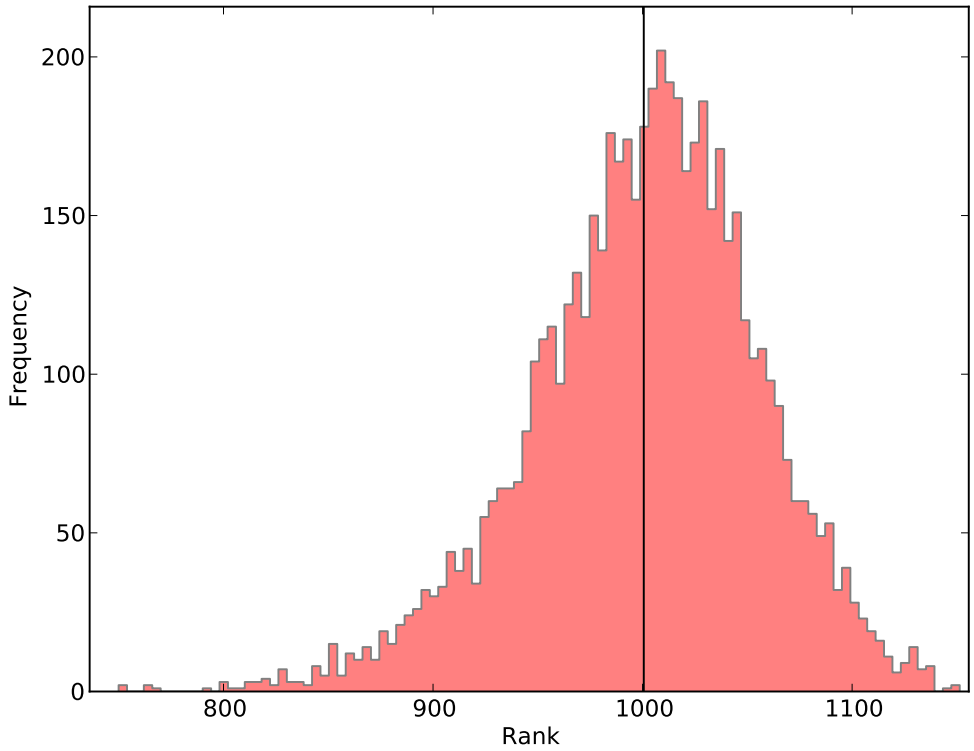


Figure 6: Rank histogram for real data with 2 different. Initial ranks were Gaussian with $\mu = 1000$ and $\sigma = 200$. The data was run through twice. Only people who participated in 6 or more races are shown. Final ranks have (a) $\mu = 1000$ and $\sigma = 55$. The black vertical line marks the mean.

5 Conclusion

Analysis of the BOF ranking system has revealed important features. It recovers the underlying ability distribution correctly for simulated data and produces a plausible rank distribution for real data. Removing outliers while still granting them points for the race causes the mean of ranks to drift significantly. Different types of rank initialization have some initial effect, but rerunning race data can reduce this. However, reruns also appear to cause the whole distribution to become more narrow, this remains unresolved and impedes understanding the convergence of the system. Future work could include a deeper look into the shrinking problem and how exactly the Central Limit Theorem can be applied to this complicated system. It would also be interesting to compare how different ranking schemes like TrueSkill order the runners.

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

An interesting aside. While plotting the rank and score standard deviation of all individuals mostly produces a big cluster of points in one area, there is still an overall trend. Higher ranked people tend to have more consistent runs.