CUED - Engineering Tripos Part IIB 2021-2022 **Module Coursework**

Module	4F13	Title of report	Coursework 2: Probabilistic Ranking
Date submitted: 19/11/2021		Assessment for this module is ☒ 100% / ☐ 25% coursework of which this assignment forms 33.33 %	

UNDERGRADUATE STUDENTS ONLY		POST GRADUATE STUDENTS ONLY			
Candidate number:	5588C	Name:		College:	

Feedback to the student

☐ **See also comments in the text**

		Very good	**Good**	Needs improvmt
CONTENT	**Completeness, quantity of content:** Has the report covered all aspects of the lab? Has the analysis been carried out thoroughly?			
	Correctness, quality of content Is the data correct? Is the analysis of the data correct? Are the conclusions correct?			
	Depth of understanding, quality of discussion Does the report show a good technical understanding? Have all the relevant conclusions been drawn?			
	Comments:			
PRESENTATION	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			
	Comments:			

Indicative grades are not provided for the FINAL piece of coursework in a module

Assessment (circle one or two grades)		A*	A	B	C	D
Indicative grade guideline		>75%	65-75%	55-65%	40-55%	<40%
Penalty for lateness:		*20% of maximum achievable marks per week or part week that the work is late.*				

Marker: Date:

4F13 Probabilistic Machine Learning
Coursework 2: Probabilistic Ranking

February 8, 2022

1 Exercise (a)

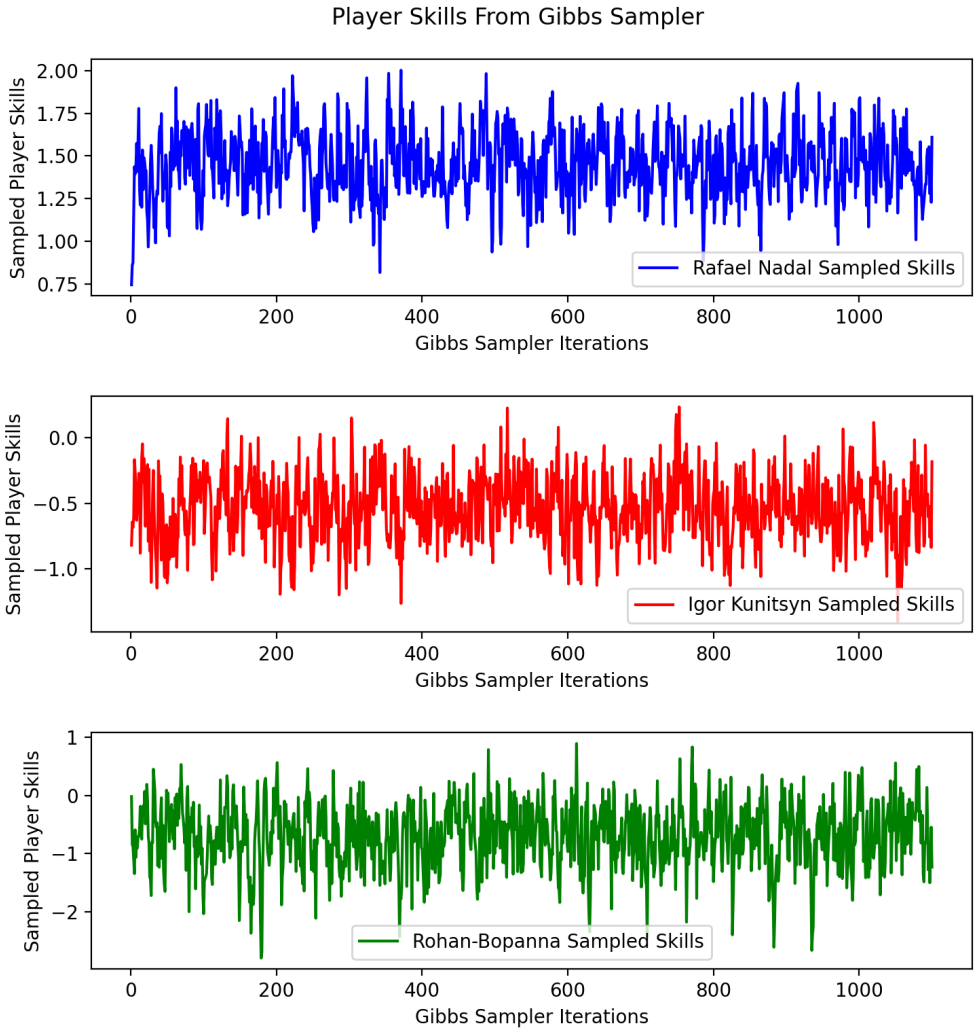

Figure 1: *Plot showing player skills from joint posterior (on the y-axis) against Gibbs sampler iterations (on the x-axis).*

```
for p in range(M):
    m[p] = t.transpose().dot((p == G[:, 0]) * 1 - (p == G[:, 1]) * 1)
iS = np.zeros((M, M))
for p1 in range(M):
    for p2 in range(M):
        if p1 == p2:
            iS[p1, p1] = sum((p1 == G[:, 0])) + sum((p1 == G[:, 1]))
        else:
            iS[p1, p2] = -1 * np.array((p1 == G[:, 0]) * 1).transpose().dot((p2 == G[:, 1]) * 1) \
            - np.array((p1 == G[:, 1]) * 1).transpose().dot((p2 == G[:, 0]) *1)
```

Source Code 1: *Code for jointly sampling skills from Gibbs sampler*

The burn-in period is the number of iterations required to generate samples drawn from the target distribution, since pseudo-random initialisation means the Markov Chain could start sampling in low density regions far away from the target invariant distribution [3]. To estimate the burn-in for these players, Table 1 and Figures 2 - 3 show that the mean stabilises after 40 iterations for the different players.

Iteration interval	$40 < t < 200$	$201 < t < 1100$	**% Change**
Rafael Nadal μ	1.5124	1.4827	-1.96%
Rafael Nadal σ	0.1906	0.1967	3.2%
Igor Kunitsyn μ	-0.5110	-0.5307	3.86%
Igor Kunitsyn σ	0.2643	0.2529	-4.31%
Rohan-Bopanna μ	-0.6630	-0.6428	-3.05%
Rohan-Bopanna σ	0.5884	0.5595	4.91%

Table 1: *Mean and Standard Deviation for player skills.*

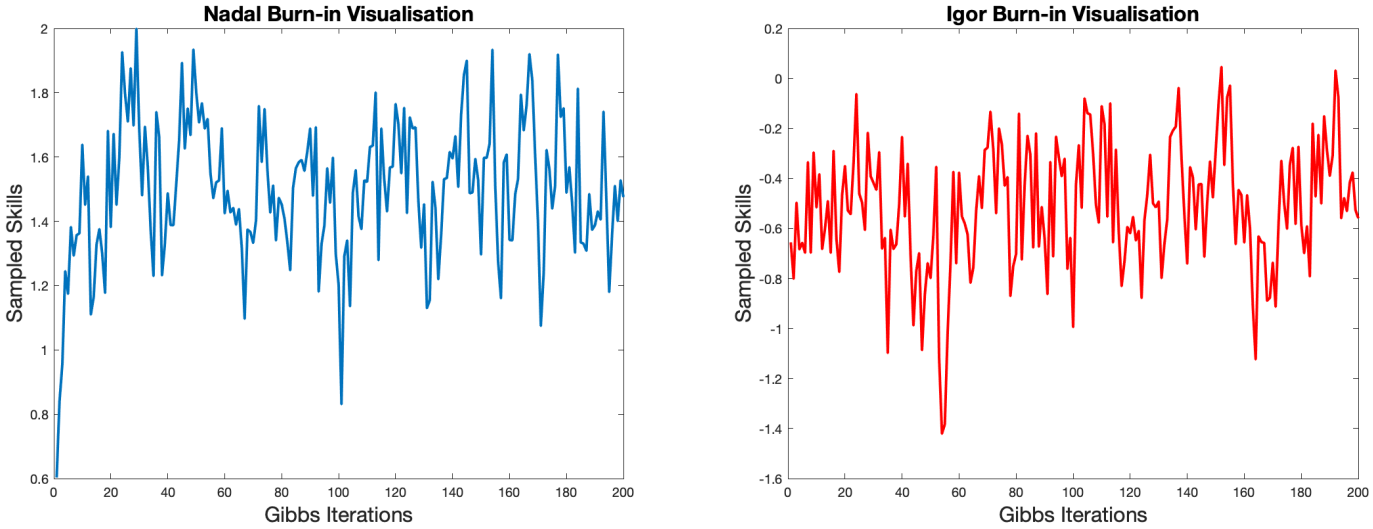

Figure 2: *Sampled Skills for Players 1 and 68 averaged over different random initialisations of the sampler, focused on earlier times to estimate burn-in.*

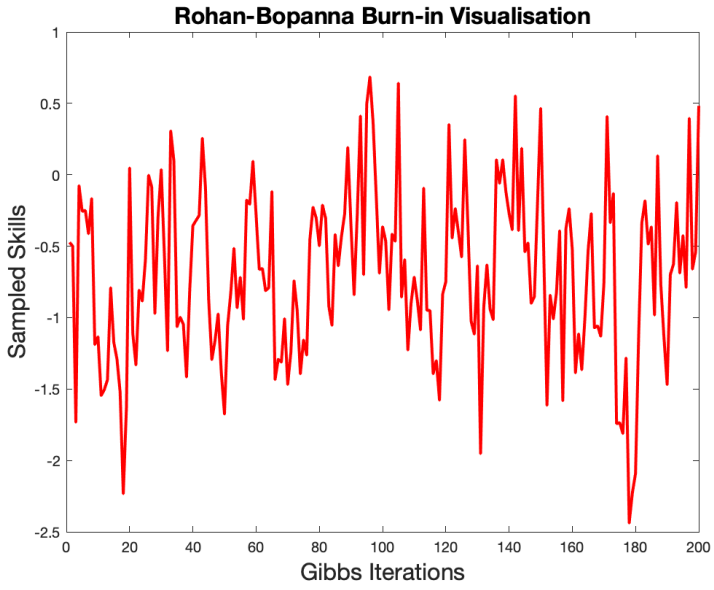

Figure 3: *Sampled Skills for Player 102 averaged over different random initialisations of the sampler, focused on earlier times to estimate burn-in.*

The auto-correlation time, τ, is the number of Markov chain transitions equivalent to a single independent draw from the invariant distribution [2], and it is estimated, from Figures 4-5, as the positive lag after which the autocovariance function hits 0 for the first time.

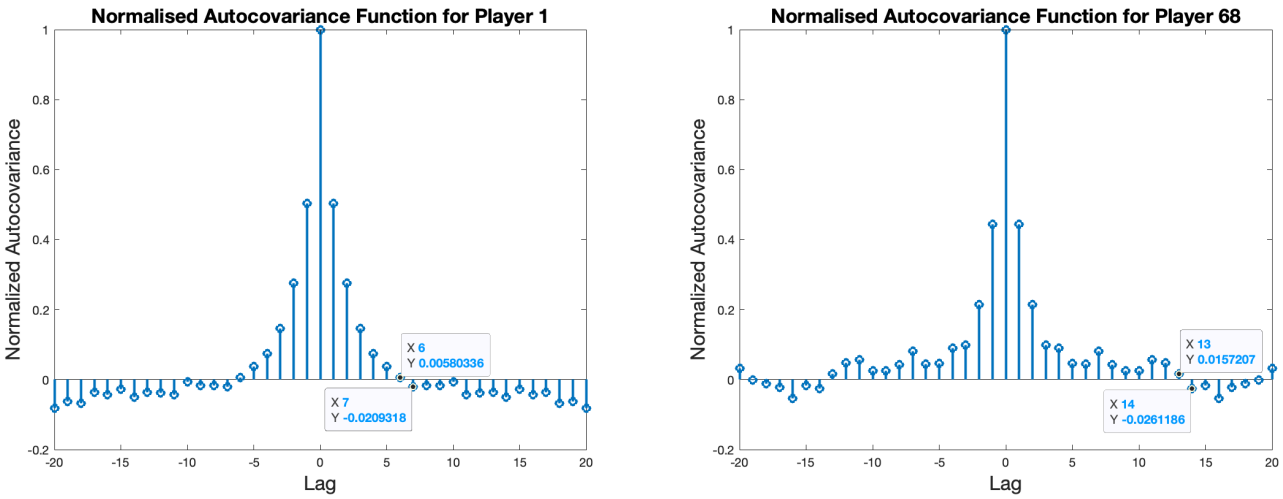

Figure 4: *Auto-covariance Function for Nadal, left, and Igor, right.*

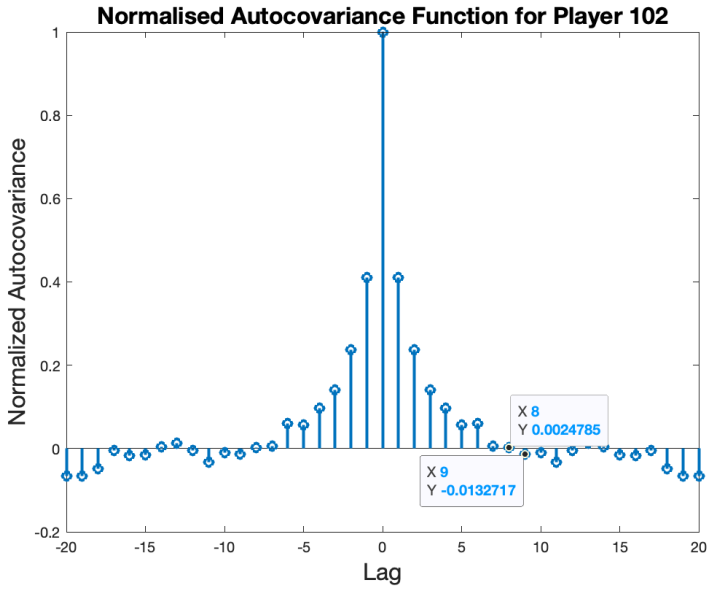

Figure 5: *Auto-covariance Function for Rohan.*

Table 2 shows that the exact autocorrelation time varies across different players. For safety, the correlation length, $L_{eff} = 20$, can be used to thin the samples by only accepting every L_{eff} samples. For safety, Figures 6-7 shows the sampled skills after thinning with length L_{eff} and discarding the samples for periods $t < t_{burn-in} \approx 40$.

Rafael Nadal τ	**Igor Kunitsyn** τ	**Rohan-Bopanna** τ
7	14	9

Table 2: *Auto-correlation times.*

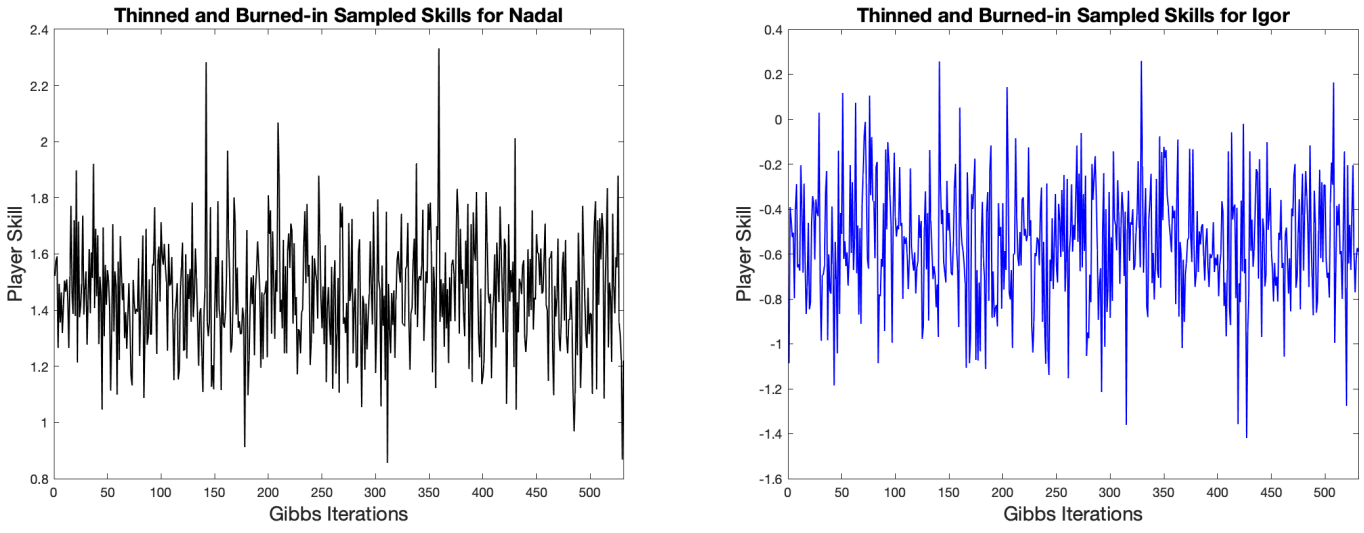

Figure 6: *Thinned and burned skill distributions for Rafael Nadal and Igor Kunitsyn.*

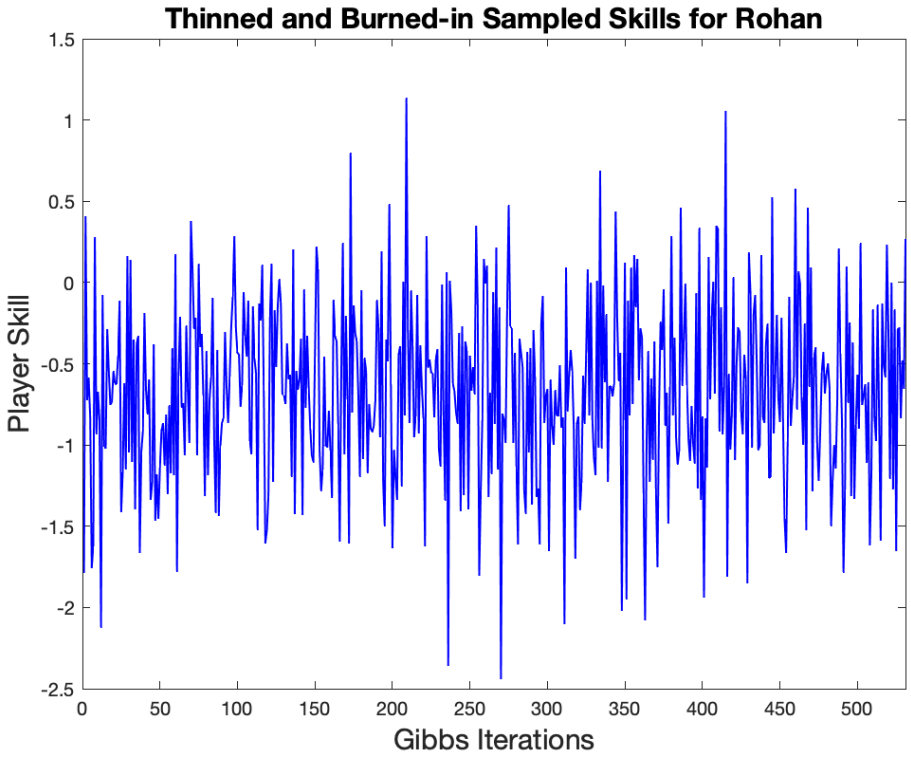

Figure 7: *Thinned and burned skill distributions for Rohan-Bopanna.*

2 Exercise (b)

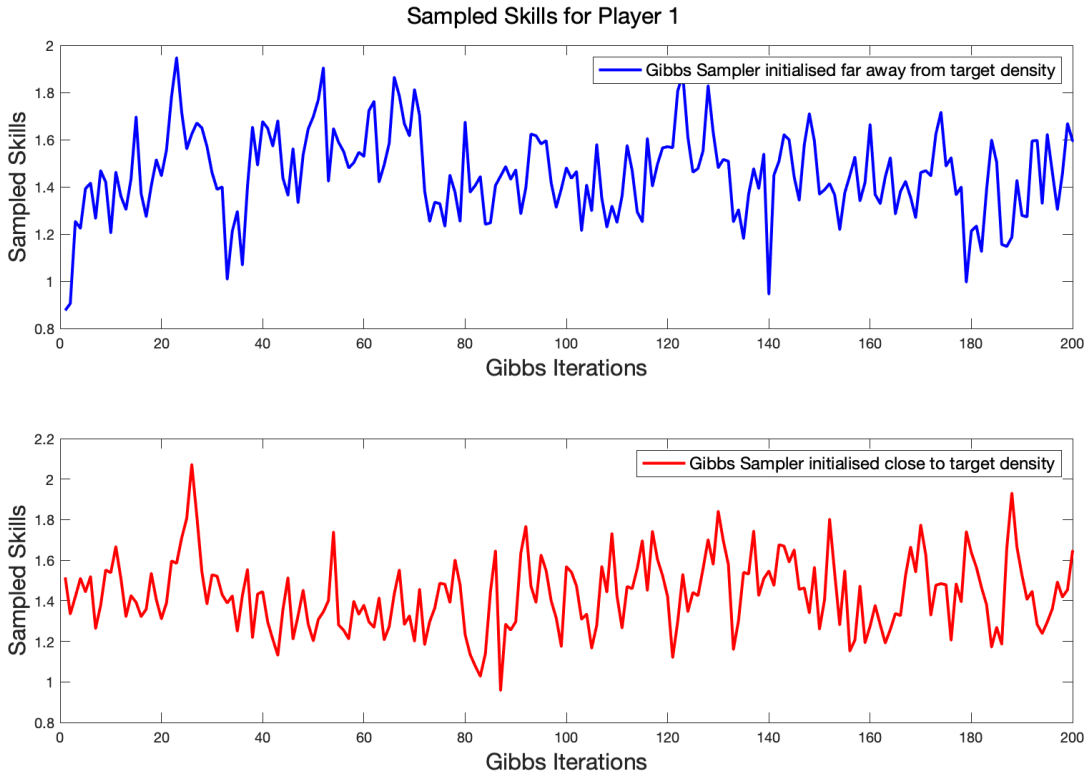

Figure 8: *Sampled skills for Nadal, initialised differently twice. Top plot shows a starting point far from Nadal's marginal mean.*

Convergence in Gibbs sampling means samples are drawn from the joint posterior distribution of player skills. Convergence time is affected by the sampler's initialisation, since far away initialisations increase the burn-in period. Figure 8 shows that initialising the sampler near the distribution shortens burn-in, as the top plot has a burn-in period of between 30 and 40 iterations, and the bottom plot's burn-in time is too small to estimate visually.

An estimate to determine convergence is to ensure the mean and variance of non-overlapping time intervals are similar, as done in Exercise (1). The Geweke convergence diagnostic [1] in Equation (1), however, is quicker to compute for all players. At a 95% level, if all players' z-values are within ± 1.96, the chain has converged. Therefore, Figure 9 shows the chain has converged after at most 40 samples, since $n_1 = 400 - 40$, $n_2 = 1100 - 560$.

$$\theta_G = \frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2}} \tag{1}$$

Where $\mu_1, \mu_2, \sigma_1, \sigma_2$ are the means and standard deviations of two partitions of the sampled skill sequence for each player.

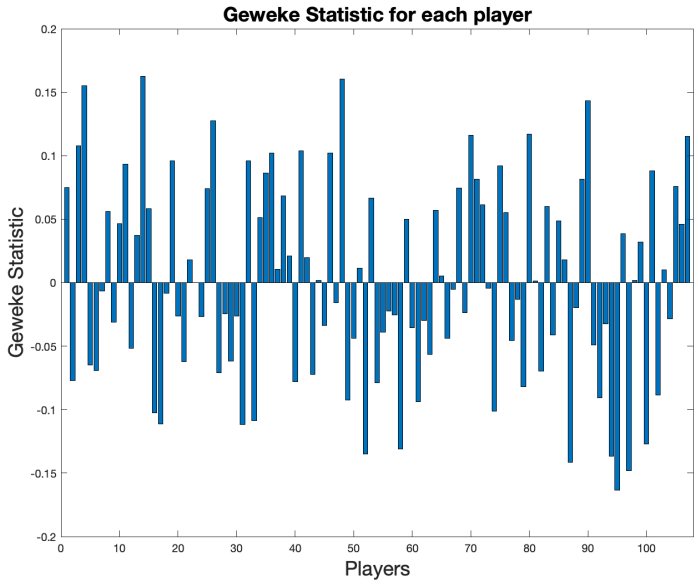

Figure 9: *Plot of Geweke diagnostic for all players.*

In messaging passing, the aim is to converge to the marginal skill distribution for each player, which is assumed to be Gaussian. Convergence is reached when the L_2 norm of the difference between consecutive marginal means and precisions reaches some upper bound, the value of which is a trade-off between accuracy and computational power. For $\epsilon = 0.001$, Figures 10 to 12 show that fewer iterations are required for convergence compared to the Gibbs sampler, and convergence estimation is more straightforward. Time to convergence is also affected by initialisation of the prior of the marginals, as shown in Figure 13.

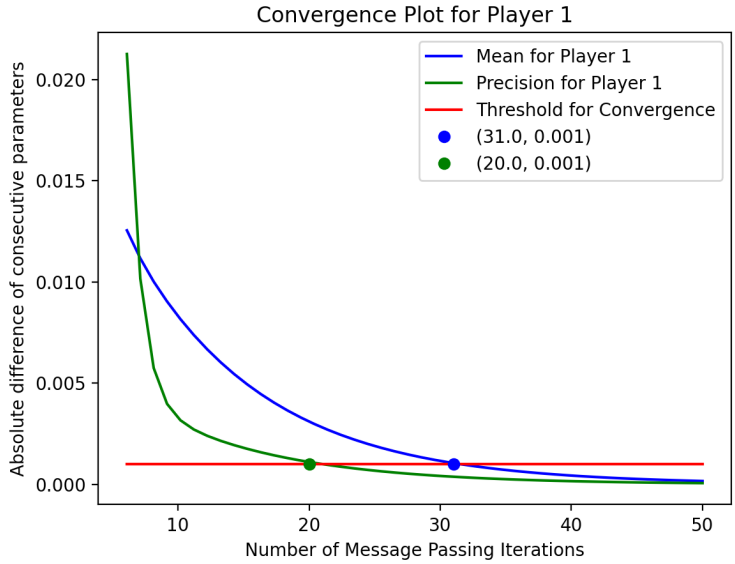

Figure 10: *Convergence occurs after 31 iterations.*

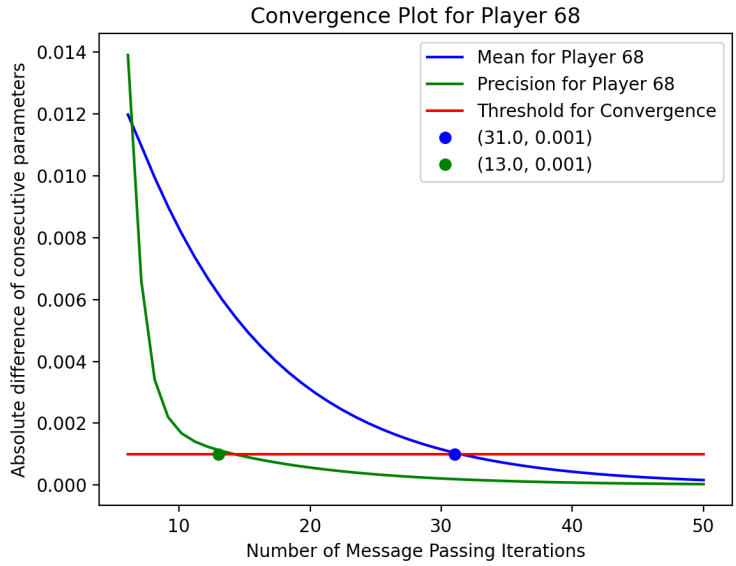

Figure 11: *Convergence occurs after 31 iterations.*

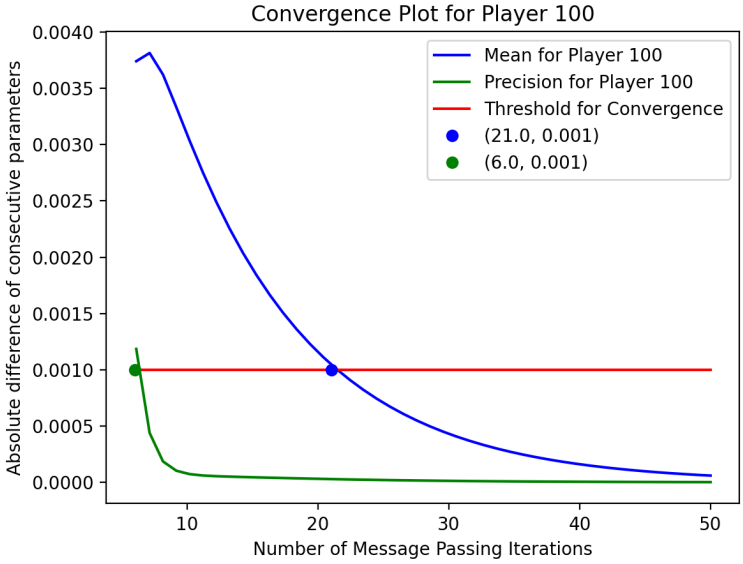

Figure 12: *Convergence occurs 21 iterations.*

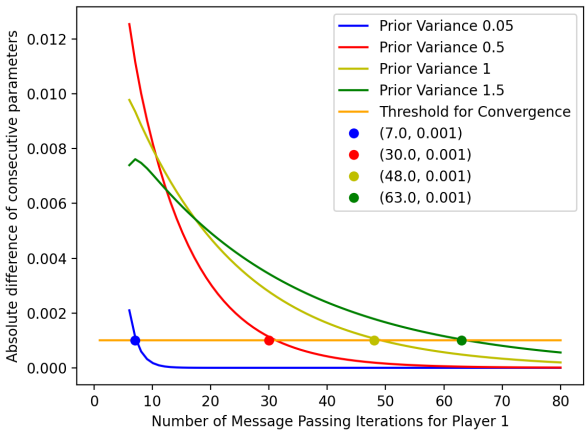

Figure 13: *Convergence plots for players 1, initialised with different prior variances.*

3 Exercise (c)

```
arg = (mean_player_skills[players[i]] - mean_player_skills[players[j]]) / (np.sqrt(\
    (1 /precision_player_skills[players[i]] + 1 / precision_player_skills[players[j]])))
prob = scipy.stats.norm.cdf(arg, 0, 1)
probs[j, i] = prob
```

Source Code 2: *Code for calculating probability player i has higher skill than player j*

If $W_i \sim p(w_i) = \mathrm{N}(\mu_i, \sigma_i^2)$, and $W_j \sim p(w_j) = \mathrm{N}(\mu_j, \sigma_j^2)$, the probability player i has a greater skill than player j is:

$$\mathrm{P}(W_i > W_j) = \mathrm{P}(W_i - W_j > 0) = \mathrm{P}(X > 0) = \Phi\left(\frac{\mu_i - \mu_j}{\sqrt{\sigma_i^2 + \sigma_j^2}}\right) \tag{2}$$

Where $\Phi(\mu_i - \mu_j)$ is the standard normal cumulative distribution.

	Djokovic	**Nadal**	**Federer**	**Murray**
Djokovic	-	0.0602	0.0911	0.0147
Nadal	0.9398	-	0.5729	0.2335
Federer	0.9089	0.4271	-	0.1892
Murray	0.9853	0.7665	0.8108	-

Table 3: *Probability column player has greater skills than row player.*

```
arg = (mean_player_skills[players[i]] - mean_player_skills[players[j]]) / np.sqrt((
                1 + 1 / precision_player_skills[players[i]] + \
                1 / precision_player_skills[players[j]]))
prob = scipy.stats.norm.cdf(arg, 0, 1)
probs[j, i] = prob
```

Source Code 3: *Code for calculating probability of player i winning a game against player j*

The probability player i wins a game against player j is:

$$\begin{aligned}
\mathrm{P}(W_i \text{ wins } W_j) &= \int\int \mathrm{P}(y=1|w_i - w_j)p(w_i)p(w_j)dw_i dw_j \\
&= \int\int P(t>0|w_i - w_j)p(w_i)p(w_j)dw_i dw_j \\
&= \int\int \mathrm{P}(w_i - w_j + z > 0)p(w_i)p(w_j)dw_i dw_j,\ z \sim \mathbb{N}(0,1) \\
&= \int\int \Phi(w_i - w_j)\mathrm{N}(w_i;\mu_i,\sigma_i^2)\mathrm{N}(w_j;\mu_j,\sigma_j^2)dw_i dw_j \\
&= \Phi\left(\frac{\mu_i - \mu_j}{\sqrt{1+\sigma_i^2+\sigma_j^2}}\right)
\end{aligned}$$

The difference between Tables 3 and 4 highlights how higher relative skills do not equate to wins, as a player can lose against a less skilled player due to player inconsistencies, modelled by additive normal random noise in the model, $t = w_i - w_j + z$. If the noise was 0, both tables would have the same entries, but its presence smoothens the likelihood of winning. Nonetheless, it is clear that there is a correlation between higher skills and likelihood of winning a game.

	Djokovic	**Nadal**	**Federer**	**Murray**
Djokovic	-	0.3446	0.3620	0.2802
Nadal	0.6554	-	0.5184	0.4269
Federer	0.6380	0.4816	-	0.4091
Murray	0.7198	0.5731	0.5909	-

Table 4: *Probability column player beats row player in a game, for message passing algorithm.*

4 Exercise (d)

	Djokovic	**Nadal**	**Federer**	**Murray**
Djokovic	-	0.0772	0.1066	0.0219
Nadal	0.9228	-	0.5568	0.2528
Federer	0.8934	0.4448	-	0.2167
Murray	0.9781	0.7472	0.7833	-

Table 5: *Probability column player has greater skill than row player by approximating the marginal distributions as Gaussians.*

	Djokovic	**Nadal**	**Federer**	**Murray**
Djokovic	-	0.0515	0.0791	0.0136
Nadal	0.9485	-	0.5625	0.2259
Federer	0.9209	0.4359	-	0.2033
Murray	0.9864	0.7741	0.7967	-

Table 6: *Probability column player has greater skill than row player by jointly approximating distribution pairs of players as Gaussians.*

	Djokovic	**Nadal**	**Federer**	**Murray**
Djokovic	-	0.0491	0.0787	0.0075
Nadal	0.9509	-	0.5415	0.2075
Federer	0.9213	0.4585	-	0.1981
Murray	0.9925	0.7925	0.8019	-

Table 7: *Probability column player has greater skill than row player using direct samples from Gibbs Sampler.*

Marginal Approx	**Joint Approx**	**Direct Gibbs**
0.9228	0.9485	0.9509

Table 8: *Probability Djokovic has higher skill than Nadal.*

Comparing Table 5 to 6, players who are higher-ranked in the lecture notes have a lower probability of being more skilled than lower-ranked players, and the opposite is true when calculating the probability of lower-ranked players against higher-skilled ones. The same holds between the other pairs of Tables. This is because the first method does not consider the covariance between player skills, whereas the second method considers only the covariance the covariance between pairs of players, as shown in the Equation below.

$$P(W_i > W_j) = \Phi\left(\frac{\mu_i - \mu_j}{\sqrt{\sigma_i^2 + \sigma_j^2 - 2\sigma_{ij}}}\right) \tag{3}$$

All covariances between all 107 players are positive or nearly 0, which, from the equation above, increases the likelihood player i is more skilled than player j. This is why the joint and Gibbs sampling methods yield higher probabilities than the marginal distributions. The covariances between players are positive because each game introduces negative correlation between two players, but then all other players who lost to the winning player will have their skills adjusted upwards, since they were beat by a more skilled player. This reflect the fact that losing to a more skilled player has less of an effect on a player's skill than losing against a weaker player.

$$p(W_i > W_j) = \frac{1}{N} \sum_{k=1}^{N} \mathbb{1} \left(w_i^{(k)} > w_j^{(k)} \right) \tag{4}$$

Only by using the direct sample estimator in (4) can the model take into account covariances between all players, as the estimator doesn't make assumptions about the joint or marginal distributions, other than that the additive noise, and the prior distribution of skills, are normally distributed. It is therefore the preferred method, and compared to Table 3, the probabilities are also higher, since message passing only approximates the truncated Gaussians with the first and second moments.

```
I = 11000
players = [djokovic_index, nadal_index, federer_index, murray_index];
for i = 1:4
    mean_player_skills(i)= mean(Ws(400:end,players(i)));
    variance_player_skills(i) = std(Ws(400:end,players(i)))^2;
end

probs = zeros(4, 4);
for i = 1:4
    for j = 1:4
        probs(j, i) = normcdf(mean_player_skills(i) - mean_player_skills(j), 0, \
        sqrt(variance_player_skills(i) + variance_player_skills(j)));
    end
end
```

Source Code 4: *Code for marginal approximation*

```
covariance_player_skills = zeros(4,4);
for i = 1:4
    for j = 1:4
        if j ~= i
            covs =  cov(Ws(400:end,players(i)), Ws(400:end,players(j)));
            covariance_player_skills(i,j) =covs(1,2);
        end
    end
end
for i = 1:4
    for j = 1:4
        probs(i, j) = 1 - normcdf(mean_player_skills(i) - mean_player_skills(j), 0, \
        sqrt(variance_player_skills(i) + variance_player_skills(j) \
        -2*covariance_player_skills(i,j)));
     end
end
```

Source Code 5: *Code for joint approximation*

```
# Gibbs Samples need to be independent for accurate estimation
I = 11000
for i in range(4):
    for j in range(4):
        probs[j, i] = np.sum(skill_samples[players[i], 400::20] > \
        skill_samples[players[j], 400::20]) /
        \ skill_samples[players[i],400::20].shape[0]
```

Source Code 6: *Code for direct samples*

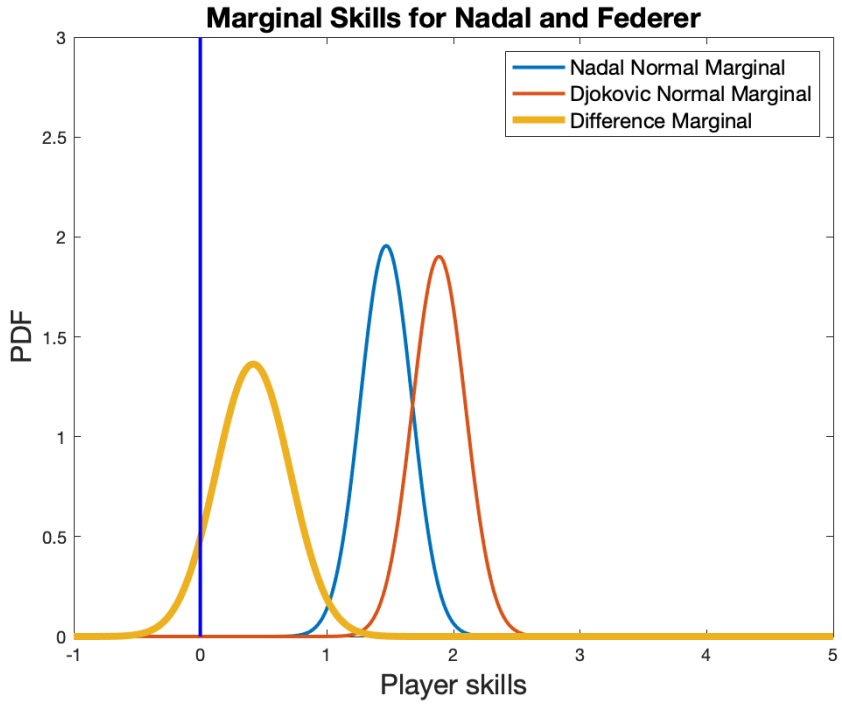

Figure 14: *Marginal skill plots for Nadal and Djokovic. The area under the orange curve and to the right of the vertical line represents the probability Djokovic has higher skill; to the left, the opposite.*

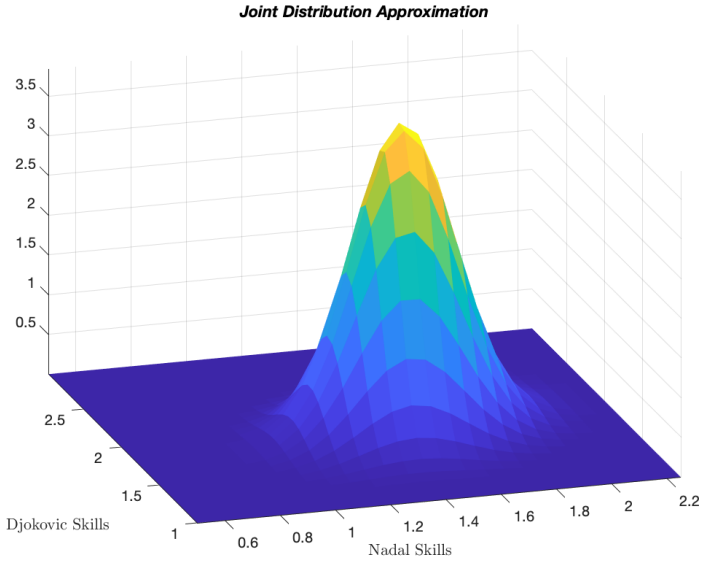

Figure 15: *Jointly Gaussian Distribution of Nadal and Djokovic skills.*

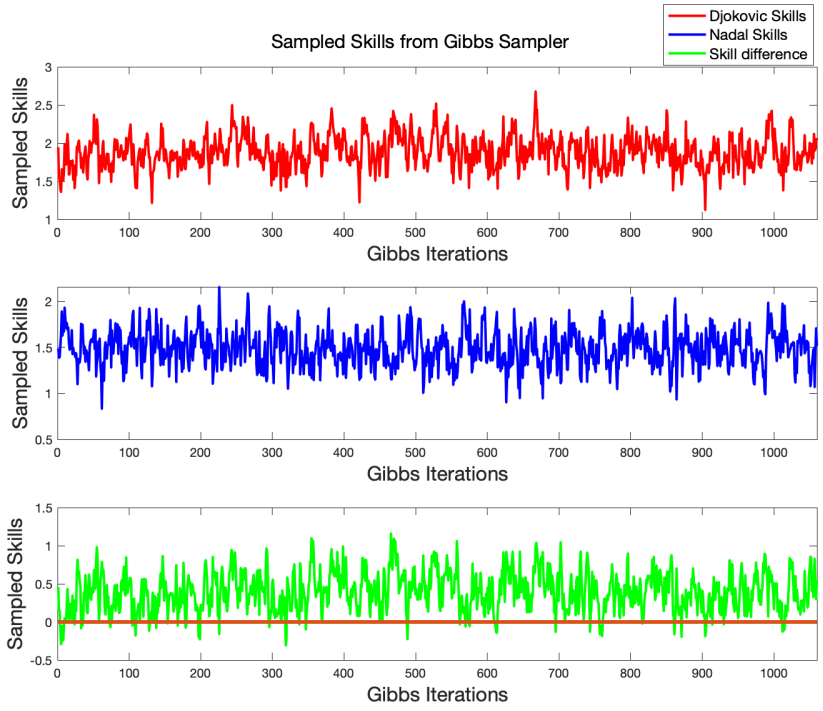

Figure 16: *Sampled skills taken directly from a Gibbs Sampler. The empirical average of the section above the orange horizontal line yields the probability Djokovic has higher skill than Nadal.*

5 Exercise (e)

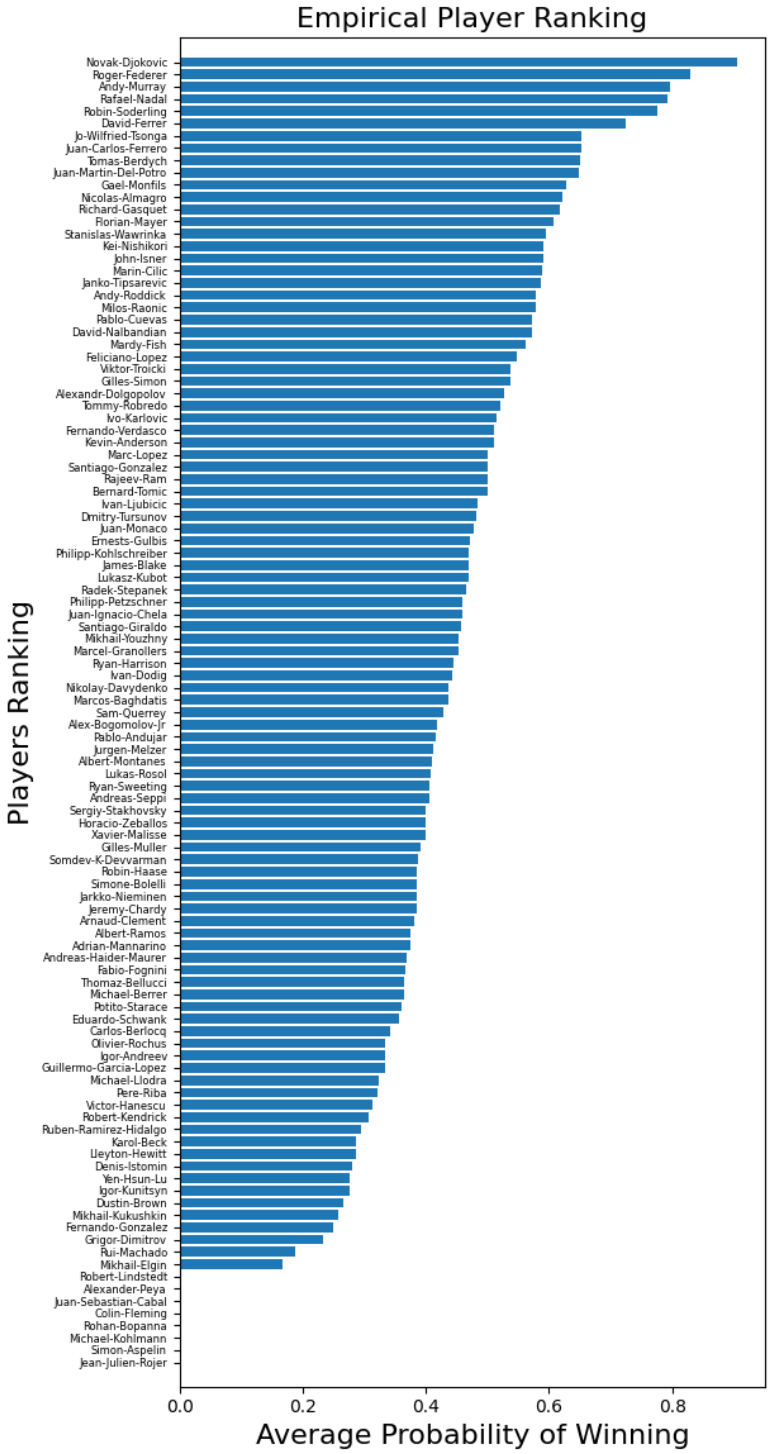

Figure 17: *Player rankings based on the proportion of games they won.*

Figure 17 shows that the empirical average of the number of wins is unsuitable, since players with the same number of wins will have exactly the same ranking, and no ranking if they haven't won any games. It is also biased in favour of incumbents with a single played game, as new players will have the highest ranking if they've won their game, and doesn't consider how relative skills influence the outcomes.

```
probs = np.zeros((M, M))
for i in range(len(skill_samples)):
    for j in range(len(skill_samples)):
        probs[j, i] = np.mean(scipy.stats.norm.cdf(\
        skill_samples[i, 400::20] - skill_samples[j, 400::20]))

av_winning_probs = np.zeros(M)
for p in range(M):
    av_winning_probs[p] = np.sum(probs[:, p]) / (M - 1); % (M -1) opponents
```

Source Code 7: *Code for direct Gibbs sampling player predictions*

```
probs = np.zeros(shape=(M,M))
for p1 in range(M):
    for p2 in range(M):
        mean_diff = mean_skills_marginal[p1] - mean_skills_marginal[p2]
        variance =1 / precision_skills_marginal[p1] + 1 / precision_skills_marginal[p2] + 1
        p_win_0 = scipy.stats.norm.cdf(mean_diff / np.sqrt(variance), 0, 1)
        probs[p1,p2] = p_win_0

av_winning_probs = np.zeros(M)
for p in range(M):
    av_winning_probs[p] = np.mean(probs[p,:])
```

Source Code 8: *Code for message passing player predictions*

Figures 18 and 19 consider the relative skill of the winning player, and hence they solve the problem of equal ranking by considering the quality of the opponent in each game as well as the number of matches played. They have similar rankings, which differ only locally based on how they approximate the average winning probabilities. Though message passing is computationally cheaper, the Gibbs sampler is the preferred method as it does not approximate the joint posterior over skills, and so its probabilities are the most exact.

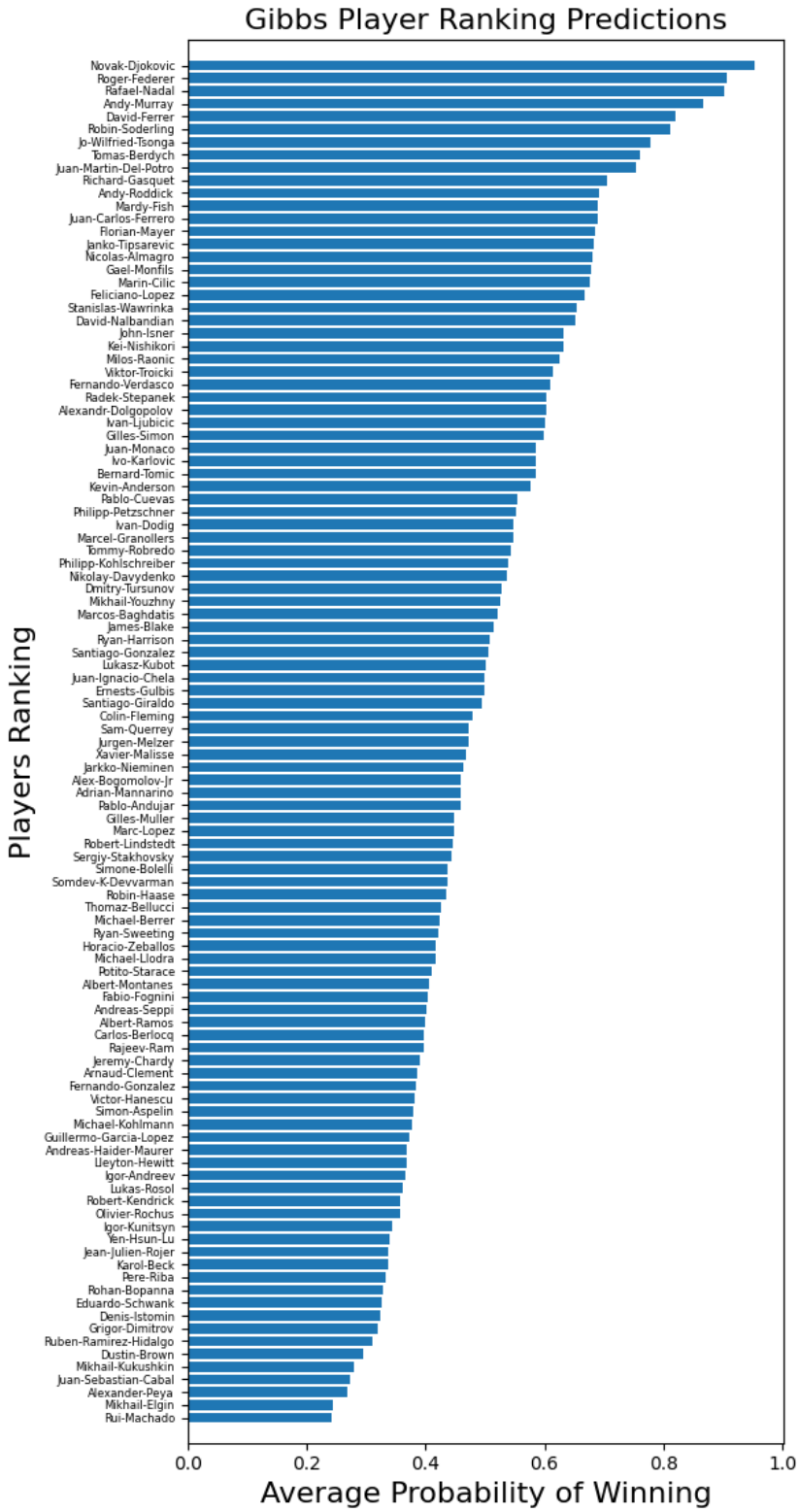

Figure 18: *Player rankings based on their average probability of winning based on the Gibbs sampler.*

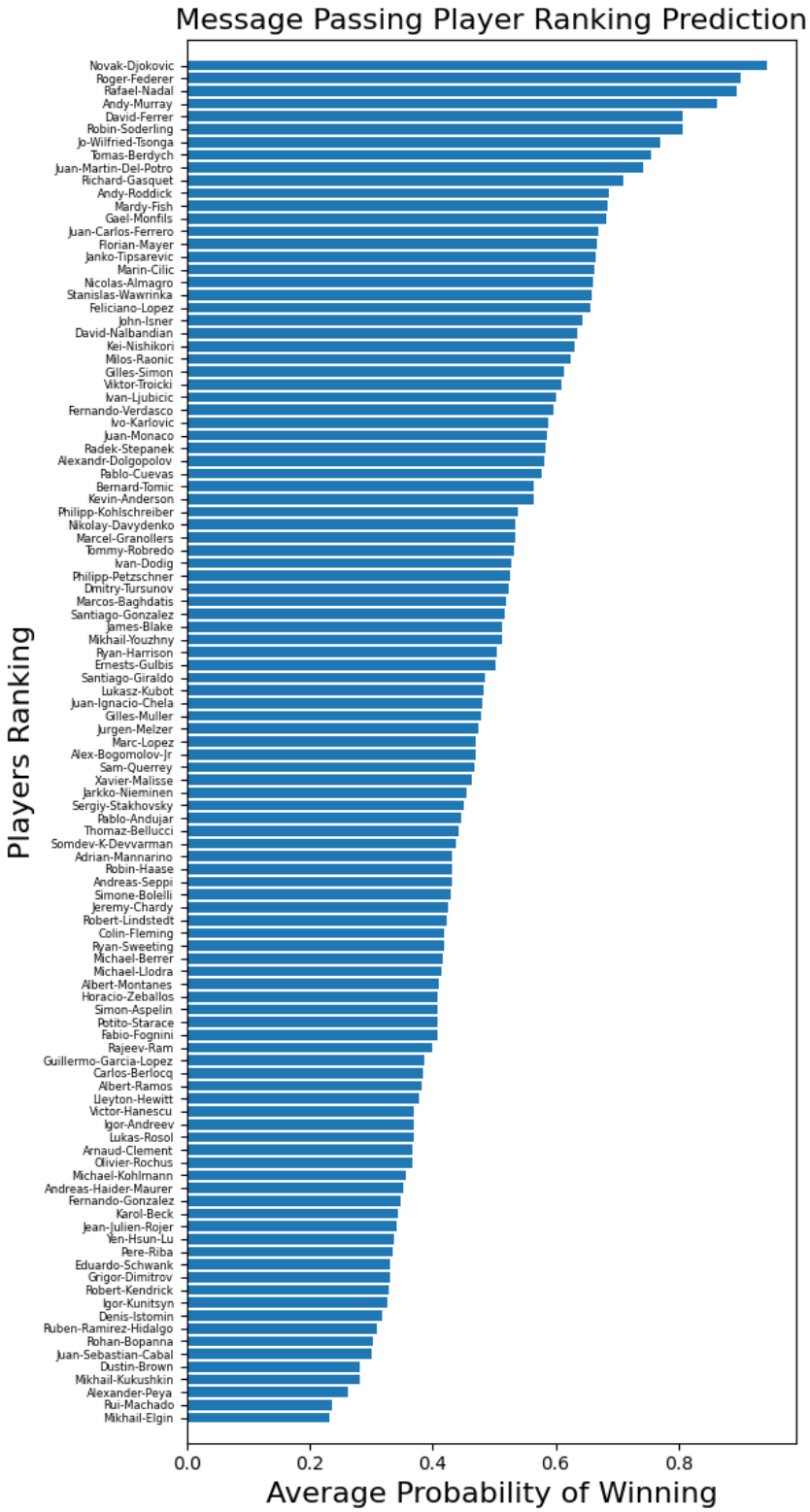

Figure 19: *Player rankings based on their average probability of winning based on the Message Passing algorithm.*

References

[1] *Appendix C: Checking Convergence Using CODNBOA*. URL: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470434567.app3.

[2] Madeleine B. Thompson. *A Comparison of Methods for Computing Autocorrelation Time*. URL: https://arxiv.org/pdf/1011.0175.pdf.

[3] Wikipedia. *Gibbs Sampling: Implementation*. URL: https://en.wikipedia.org/wiki/Gibbs_sampling#Implementation.

Word Count: 916