Exercise: a) Script below computes the empirical ratio of wins per game:

```
for i = 1:107
    Tot_games = sum([G(:,1)== i; G(:,2)== i]); Tot_wins = sum(G(:,1)== i);
    P(i) = Tot_wins/Tot_games;
end  %Then use cw2.m
```

If we understand *skill* of a player as measure of the probability of winning a game, empirical win ratio is not a good way of estimating skill, for a number of reasons:

1. It is not comparable. It ignores the important information about rivals such as number of rivals encountered and their individual strength. For example, a player that has played and lost his only game against a top player would have skill 0 and no chance of winning against any other player.
2. It does not let us infer on the outcome of a game.
3. It does not take into account other "physical" factors that can also affect the outcome of a game, such as: court-type, moment of the season (eg: improvement of players throughout the season), luck (noise), etc.

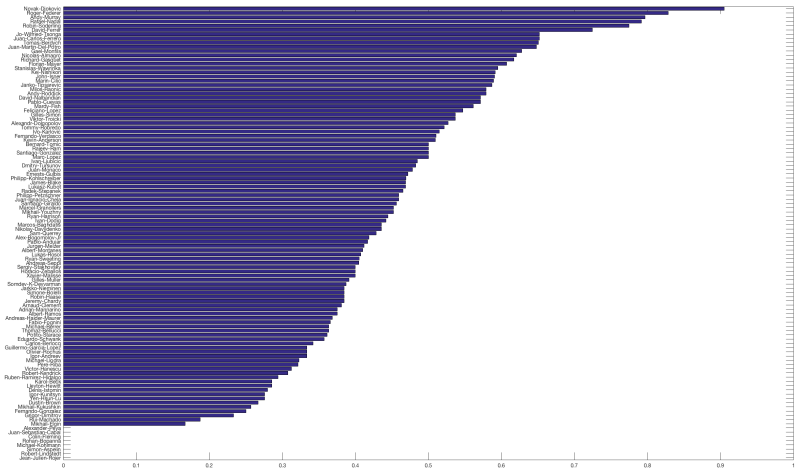


Figure 1: Ranking according to empirical win ratio.

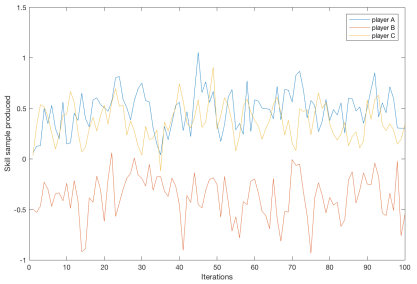
Exercise: b) From lectures we have the following updates for `m` (mean of the conditional skill distribution given the t_g samples) and `iS` (sum of precision matrices contributed) in the Gibbs sampler setting:

```
m = nan(M,1);
for p = 1:M
    m(p) = m(p) = prior_mu(p)/pv(p) + sum(((G(:,1) == p) - (G(:,2) == p)).* t);
end
iS = zeros(M,M);
for g = 1:N  %Faster than a for loop on the entries
    iS(G(g,1),G(g,1)) = iS(G(g,1),G(g,1)) + 1;       %Add one for each game  played
    iS(G(g,2),G(g,2)) = iS(G(g,2),G(g,2)) + 1;
    iS(G(g,1),G(g,2)) = iS(G(g,1),G(g,2)) - 1;       %Subtract one for each game played
    iS(G(g,2),G(g,1)) = iS(G(g,1),G(g,2));           %against a particular player
end
```

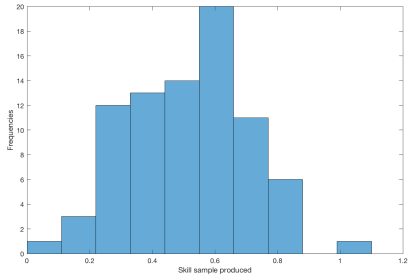
Exercise: c) The 100 skill samples generated by the Gibbs sampler are autocorrelated, which reduces the information content relative to perfectly independent samples. As shown in figure (a) and (b) below, it takes a few samples (about 10-20), but once the underlying MC forgets its initial state and leaves the burn-in phase, the Gibbs sampler seems to move around the skill posterior of the players. Thus, we will ignore these first few samples from now on (**Burn-in method**). In Murphy's book, the autocorrelation ρ_t for different lags t (time difference between samples) is given by: (with $\bar{w}$ is the sample mean of the skill of a particular player)

$$\rho_t = \frac{\frac{1}{M-t}\sum_{m=1}^{M-t}(w_m - \bar{w})(w_{m+t} - \bar{w})}{\frac{1}{M-1}\sum_{m=1}^{M}(w_m - \bar{w})^2}$$

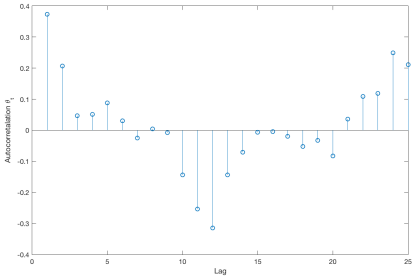
We can implement this into `MATLAB` to obtain figure (c) below that shows how autocorrelation drops significantly then samples differ in more than 5 iterations. 7-9 lags seems to produce the least autocorrelation in this case. Thus, to reduce autocorrelation we will only keep one in every 8 samples after the burn in phase, which is known as the **thinning method**.



(a) 3 sequences of Gibbs skill samples

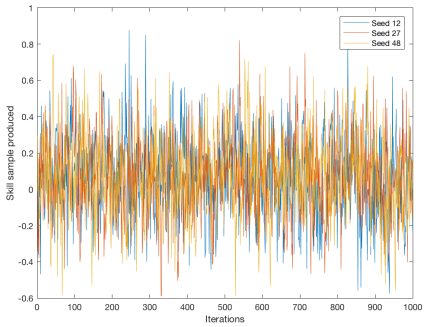


(b) Histogram of Skill A 20-100 samples

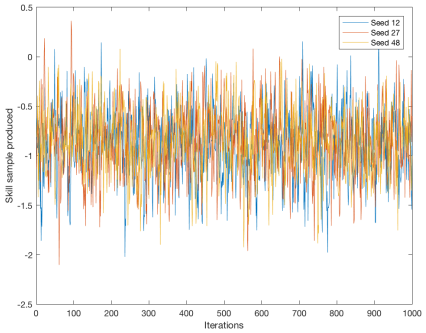


(c) Autocorrelation ρ_t in player A samples

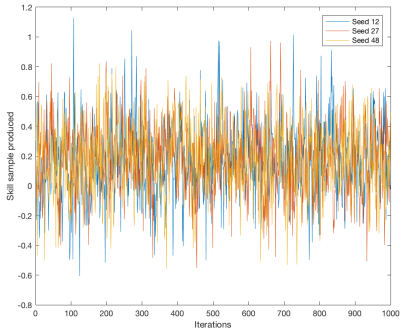
Exercise: d) We say a Markov Chain X_t with transition matrix T_{ij} and initial state π_0 converges when it reaches a stationary distribution π over the states such that $\pi = T\pi$. That is, the MC forgets the initial state π_0 after the repeated application T, reaching a limit distribution that cannot leave. For the Gibbs sampler, the alternated sampling of $\mathbf{t}' \sim p(\mathbf{t} \mid \mathbf{w})$ and $\mathbf{w}' \sim p(\mathbf{w} \mid \mathbf{t}')$ aims to converge a set of samples from the joint distribution $p(\mathbf{w}, \mathbf{t})$. Assessing convergence of the Gibbs sampler is not an easy task. We can try to do so by running multiple chains with different random seeds (hence very diffent starting points) and obtain a mixed **trace plot** for various samples from a 1D (marginal) skill distribution $p(w_m)$ as shown in the figures below.



(a) Trace plot of Player A



(b) Trace plot of Player B

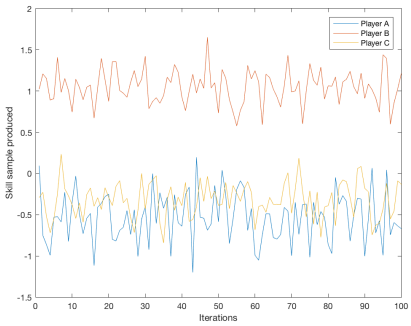


(c) Trace plot of Player C

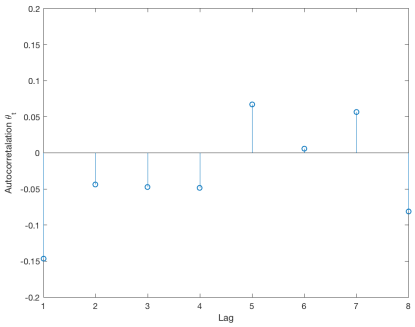
Figure 3: Trace plots of players A, B and C with random seeds 12, 27 and 48 after 1000 iterations

For each player, the different random seeds seem to produce a very mixed skill sample sequences after leaving the burn-in phase. This is a good indicator that the Gibbs sampler has reached convergence.

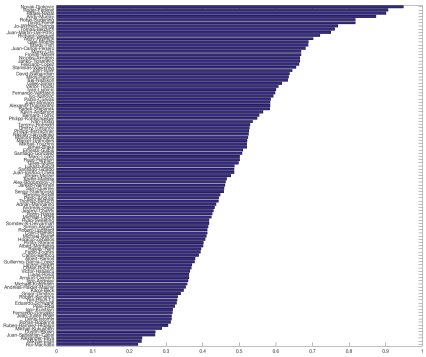
Exercise: e) Figure (a) below shows 3 skill sample sequences of length 100 produced by the Gibbs sampler after burn-in (ignore first 200 iterations) and thinning (keeping 1 in every 8). Figure (b) shows the small autocorrelation observed in one of these skill sample sequences independently of the lag.



(a) 'Independent' skill sample sequences



(b) Autocorrelation for skill samples



(c) Ranking based on average winning prob. from Gibbs sampler

The script below computes the average winning probability of each `player` by making it play against the rest of `player2` in each of the 100 `samples` and then generates a ranking based on that:

```
P = zeros(M, 1); skill_selection = skills(:,200:8:1000);
for player = 1:M
    win_prob = [];
        for sample = 1:101
            ws = skill_selection(:,sample);
            for player2 = 1:M
                if player2 ~= player
                    win_prob = [win_prob; normcdf(ws(player) - ws(player2))];
                end
            end
        end
    P(player) = mean(win_prob);
end % Then use cw2.m
```

Here, the probability of `player` winning against `player2` in each of these games is given by the likelihood:

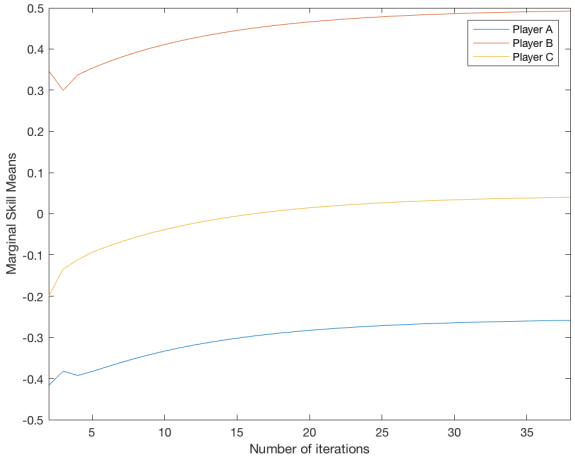
$$p(y = 1 \mid w_{player}, w_{player2}) = \Phi((w_{player} - w_{player2}))$$

Figure (c) above shows a more fair ranking system where despite maintaining the correlation between rank and true skill (in the case of well known top players like Nadal, Federer, etc.), does not have either an absolute loser (players with skill 0) nor absolute winner. Compared to the empirical win rate ranking, this ranking is based on a **comparable skill** that takes into account the skill of rivals each player has compete against, as well as (hypothetical) larger game database (107x106x101) than `G`.

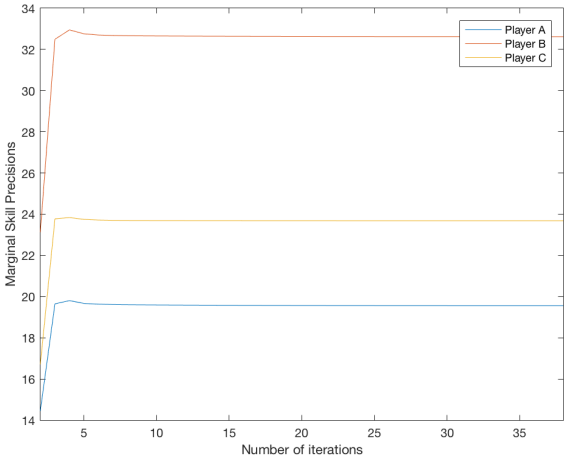
Exercise: f) We can use a similar loop to compute the average winning probability in each of these matches between the ATP Top 4 players over all the Gibbs skill samples. Here again we are using the likelihood $p(y = 1 \mid w_{player}, w_{player2}) = \Phi((w_{player} - w_{player2}))$ as our model for the winning probabilities. Table below shows the resulting probabilities which seem to agree more or less with the rankings.

⇓ **Winner–Loser** ⇒	Novak Djokovic	Rafael Nadal	Roger Federer	Andy Murray
Novak Djokovic	–	0.6518	0.6375	0.7098
Rafael Nadal	0.3482	–	0.4845	0.5650
Roger Federer	0.3625	0.5155	–	0.5799
Andy Murray	0.2902	0.4350	0.4201	–

Exercise: g) If instead we do inference in the model by using *message passing* and *expectation propagation* (EP) we see that the number of iterations until convergence to (some gaussian approximation of) the marginal posterior skill distribution is reduced drastically compared to the Gibbs sampler. The way convergence in distribution is typically assessed is by considering the convergence of the output sequence of means and precisions. For the figures (a) and (b) below, a strong **threshold** of 0.001 is set to the L_2-norm of the change in the marginal skill mean `Ms` and precision `Ps` vectors between every two iterations. It only takes 38 iterations to reach it.



(a) Convergence of marginal skill means

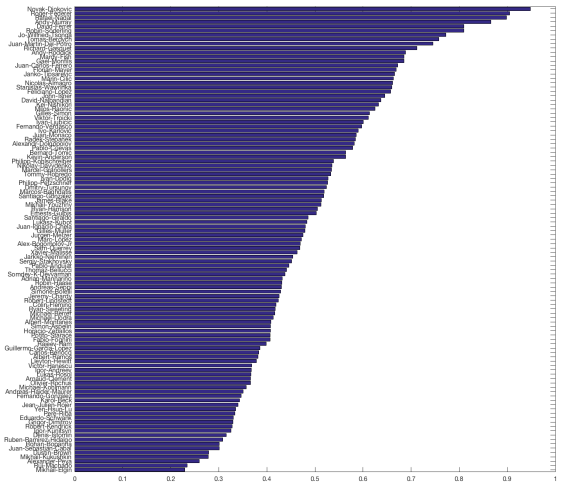


(b) Convergence of marginal skill precisions

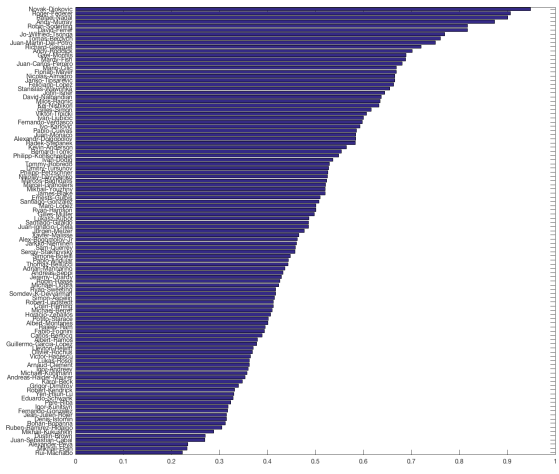
Exercise: h) After running sufficient ($\sim$ 40) iterations using message passing and EP, `eprank.m` returns the marginal skill mean `Ms` (with entries μ_i) and precision `Ps` (with entries σ_i^{-2}) vectors, so that the (approximate) marginal posterior skill distributions $p(w_i) = \mathcal{N}(w_i \mid \mu_i, \sigma_i^2)$ for each player $i = 1, ..., 107$. From here, in lectures we showed that the marginal distribution of player i winning player j (represented by $y = 1$) comes from the cdf of a Gaussian:

$$p(y = 1 \mid \mu_i, \sigma_i^2, \mu_j, \sigma_j^2) = \Phi\left(\frac{(\mu_i - \mu_j)}{\sqrt{1 + \sigma_i^2 + \sigma_j^2}}\right)$$

We can use a similar loop to the one in exercise (e) to obtain the ranking in figure (a) below.



(a) Ranking using message passing and EP



(b) Ranking using Gibbs sampler

This ranking is very similar in shape and order to the one obtained in exercise (e). For example the ATP top 4 players still in the top 4 and in the same order as for the Gibbs sampler. These rankings completely differ with the empirical winning ratio ranking as winning ratios are not comparable measures of skill and hence ignore key information about the rivals each player faced.

Both the Gibbs sampler and the message passing and EP make good approximations to the true skill posterior joint density (EP faster than Gibbs). However, it should be noted that as the number of iterations increases to infinity the accuracy of Gibbs sampler improves, whereas the EP/ message passing has already converge to some gaussian like approximation.

Exercise: i) The table below shows the winning probability in each of games between the ATP top 4 players using the Gaussian cdf in exercise (h) based on the marginal skill means and precisions from `eprank.m`.

⇓ **Winner–Loser** ⇒	Novak Djokovic	Rafael Nadal	Roger Federer	Andy Murray
Novak Djokovic	–	0.6554	0.6380	0.7198
Rafael Nadal	0.3446	–	0.4816	0.5731
Roger Federer	0.3620	0.5179	–	0.5909
Andy Murray	0.2802	0.4269	0.4091	–

Both tables (this one and the one in exercise (f)) have very similar probabilities and seem to agree with the rankings in the bottom of the previous page from the Gibbs sampler anf the EP/message passing. This goes to show that both methods, after the necessary refinements, produce a good approximation to the skill posterior.

Exercise: j) Let $s_{ij} = w_i - w_j$ be the different in skill. For the Gibbs sampler, $p(s_{ij} > 0)$ can be easily implemented in `MATLAB` since: $p(s_{ij} > 0) = \mathbb{E}(\mathbb{I}(w_i > w_j))$ where $\mathbb{I}$ is an indicator function. We can use a for loop similar to the one in exercise (f) to count and average (take expectation) over the all different samples of w_k-s by rewriting:

```
win_prob = [win_prob; ws(player) > ws(player2)]
```

⇓ **Winner–Loser** ⇒	Novak Djokovic	Rafael Nadal	Roger Federer	Andy Murray
Novak Djokovic	–	0.9604	0.9208	0.9901
Rafael Nadal	0.0396	–	0.4059	0.7426
Roger Federer	0.0792	0.5941	–	0.8020
Andy Murray	0.0099	0.2574	0.1980	–

The table above shows the probabilities of a player having higher skill than another rival. However, it can lead to absolute winning probabilities if used to model of game outcomes, as it is quite frequentist and ignores noise.

`eprank.m` returns the marginal skill mean `Ms` (with entries μ_i) and precision `Ps` (with entries σ_i^{-2}) vectors, so that the marginal skill distribution $p(w_i) = \mathcal{N}(w_i \mid \mu_i, \sigma_i^2)$ for each player $i = 1, ..., 107$. From linearity of means and variance, it follows that $s_{ij} \sim \mathcal{N}(w_i - w_j \mid \mu_i - \mu_i, \sigma_i^2 + \sigma_j^2)$ and hence $p(s_{ij} > 0) = p(w_i > w_j) = \Phi(\frac{(\mu_i - \mu_j)}{\sqrt{\sigma_i^2 + \sigma_j^2}})$ We can implement this by replacing:

```
win_prob = [win_prob; normcdf((Ms(player) - Ms(player2))/(sqrt(Vars(player) + Vars(player2))))]
```

The table belows shows the probabilities of a player having higher skill than another rival:

⇓ **Winner–Loser** ⇒	Novak Djokovic	Rafael Nadal	Roger Federer	Andy Murray
Novak Djokovic	–	0.9399	0.9089	0.9853
Rafael Nadal	0.0601	–	0.4271	0.7664
Roger Federer	0.0911	0.5729	–	0.8101
Andy Murray	0.0147	0.2336	0.1892	–

The main difference with the tables in (f) and (i) is that, here we ignore the possible noise in the performance. Remember $t = s + n$. This causes that the players with more skill (eg: Djokovic) have almost absolute probabilities.