BUSN41916 Project: A Bayesian Perspective on Plackett-Luce Ranking Models

Mingyu Liu

December 1st, 2022

1 Introduction

In this project, we investigate how the skills and the age seniorities of 24 players influence their ranking in competitions. This is done using the results for 134 games played over 21 years. Figure 1 plots the number of games each player has played and the frequency they came first and last. There is a large variation in the number of games participated by each player. Player 10 takes part in the largest number of games (= 77) while player 6 and player 14 only participate in 2 games each. Among players who participate in more than 10 games, player 19 has the highest 77.8% of coming top among the games played, indicating it is likely to be a strong player, while player 9 has the highest 87.5% of coming bottom, giving evidence that it is a weak player.

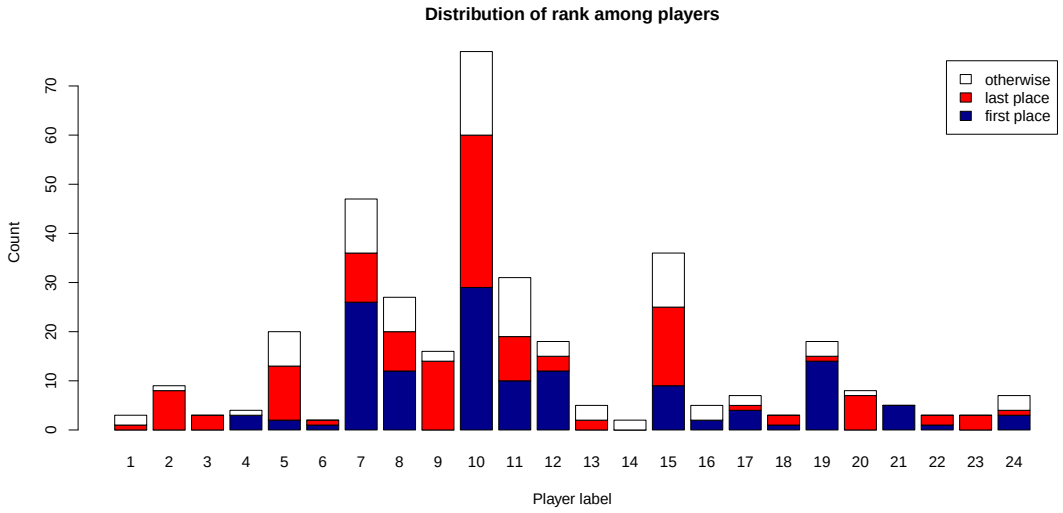


Figure 1: The distribution of ranking among the games participated by each of the 24 players.

Figure 2 plots the fractions of the three seniority levels (1-5, 6-10 and 11-16) among players who are ranked at the top and the bottom place among all 134 games. From the plot, there is no clear evidence that seniority rank influences the outcomes of the games.

In the following sections, we will consider two Plackett-Luce observation models, one taking into account the effects of seniority and one without. We will carry out model selection and further

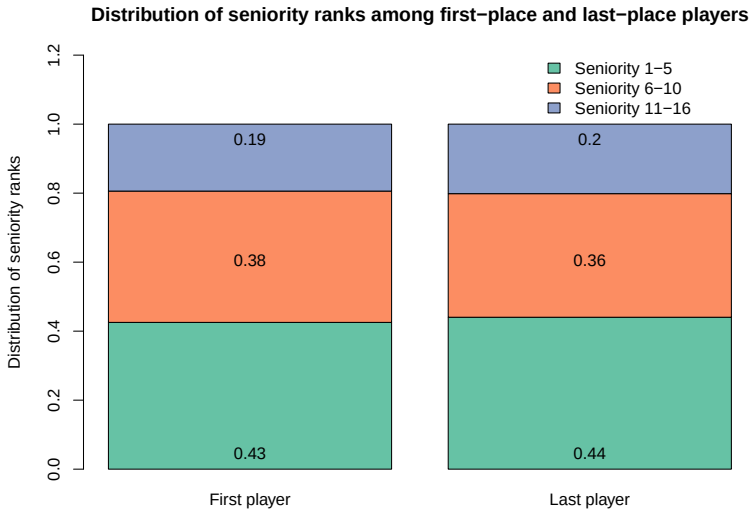


Figure 2: The distribution of seniority ranks among first-place and last-place players.

test the significance of seniority in determining the game outcomes.

2 Model Description

2.1 Plackett-Luce observation model

Model 1 (without seniority covariates): We use a Plackett-Luce observation model for these data. In a Plackett-Luce model without covariates, each player $i = 1, \ldots, n$ has a skill $\lambda_i \in R$. For $1 \leq m \leq n$ let $\{i_1, \ldots, i_m\}$ be a subset of $\{1, 2, \ldots, n\}$ and let $\mathcal{P}_{i_1,\ldots,i_m}$ denote the set of all permutations of the labels $\{i_1, \ldots, i_m\}$. In a game with m players with player labels $i_1, \ldots, i_m$, the outcome $O \in \mathcal{P}_{i_1,\ldots,i_m}$ is a random permutation $o = (o_1, \ldots, o_m), o \in \mathcal{P}_{i_1,\ldots,i_m}$ of the player labels. In the Plackett-Luce model with $\lambda = (\lambda_1, \ldots, \lambda_n)$, the probability for the random rank-outcome O of a single generic competition to be some particular ranking o is

$$\Pr\{O = o \mid \lambda\} = \prod_{i=1}^{m} \frac{\exp\left(\lambda_{o_i}\right)}{\sum_{j=i}^{m} \exp\left(\lambda_{o_j}\right)}$$

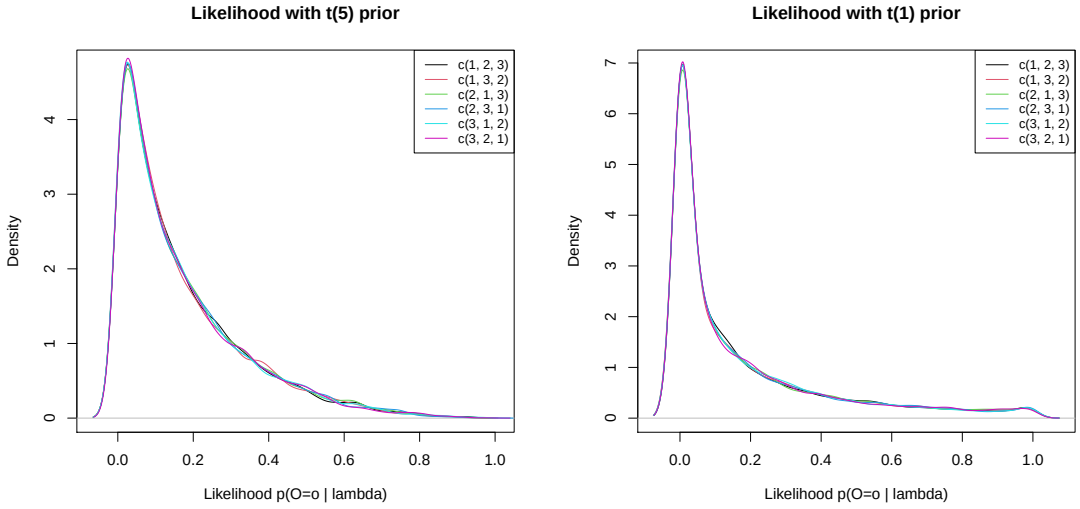
Model 2 (with seniority covariates): A vector of seniority parameters $\beta = (\beta_1, \ldots, \beta_n)$ may be introduced so that the effect of having seniority-level $k, k = 1, \ldots, n$ is $\beta_k \in R$. In this observation model

$$\Pr\{O = o \mid \lambda, \beta\} = \prod_{i=1}^{m} \frac{\exp\left(\lambda_{o_i} + \beta_{e_i}\right)}{\sum_{j=i}^{m} \exp\left(\lambda_{o_j} + \beta_{e_j}\right)}$$

where for $i \in \{i_1, \ldots, i_m\}$ the index e_i on β_{e_i} gives the seniority rank of player i at the time the game was played.

2.2 Prior elicitation and checking

In eliciting the priors for the skills λ and the effects of seniority β for each player, we should bare in mind that these priors should be non-informative with respect to the relative skills of the players and the advantage for seniority. In addition, the probability of all games outcomes corresponding to the set of all permutations of the labels $\{i_1, \ldots, i_m\}$ should be the same for any given subset $\{i_1, \ldots, i_m\}$ of $\{1, 2, \ldots, n\}$. The domains of the parameters $\lambda \in R^m$, therefore, the class of t distributions offers a good choice. We simulate $10,000$ samples from the priors $\lambda_{o_i} \overset{\text{i.i.d}}{\sim} t_5$ and plot the likelihood $\Pr\{O = o \mid \lambda\}$ for each random permutation $o \in \mathcal{P}_{1,2,3}$ of three players $\{1, 2, 3\}$ in Figure 3a. The densities of the likelihood function of the 3! outcomes match well. Marginalizing over λ, the estimate for the expectation $E_{\lambda|model}\left[\Pr\{O = o \mid \lambda\}\right]$ of each permutation is close to $1/3!$, agreeing with our prior beliefs that all permutations $o \in \mathcal{P}_{1,2,3}$ are equally probable. The distribution becomes tighter as we use a lower degree of freedom t_1 (Figure 3b). The larger degree of freedom t_5 is desired as we want to be less certain about the likelihood of the outcomes due to the randomness of the game.



(a) Model 1 likelihood with t_5 prior

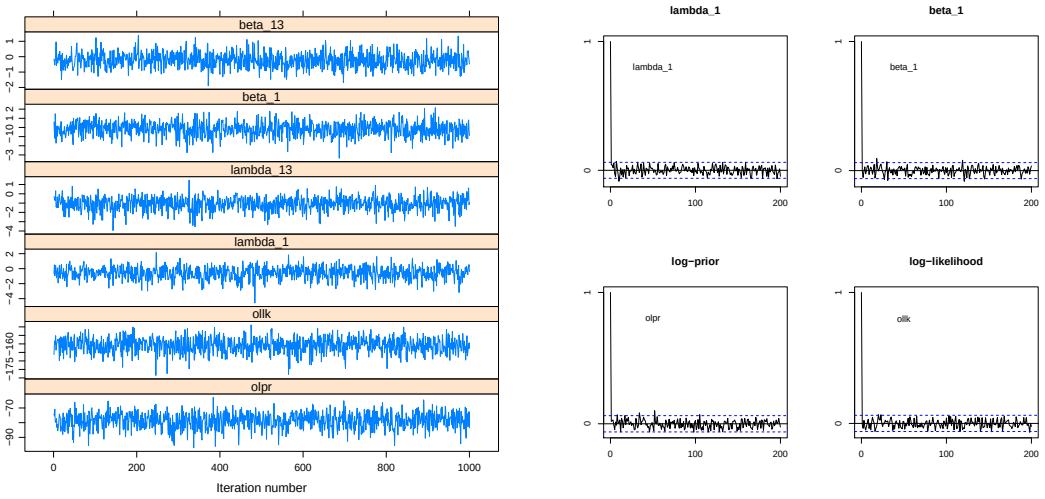
(b) Model 2 likelihood with t_1 prior

Figure 3: Likelihood densities of all 3! permutations of the labels $o \in \mathcal{P}_{1,2,3}$ in model 1 (with parameter λ). The densities of all outcomes match well, indicating the prior is non-informative with respect to the rankings of the players involved in the competition. The t_5 distribution with the larger degree of freedom gives less certainty about the outcomes and is desired.

We perform a similar check for model 2 with seniority covariates added, we choose the additional priors $\beta_{e_i} \overset{\text{i.i.d}}{\sim} t_5$ and observe that the likelihood densities of all outcomes match. Again, a t distribution with a larger degree of freedom gives less certainty about the outcomes. The estimate for the marginal likelihood $E_{\lambda,\beta|model}\left[\Pr\{O = o \mid \lambda, \beta\}\right]$ is close to $1/m!$, matching our prior beliefs.

3 Model Fitting using MCMC

We fit each model using a sequential scan Metropolis-Hastings algorithm where we visit each component of the parameters in turn, and for each component, j propose $X_j \sim q_j(\cdot \mid X_1^{(t)}, \ldots, X_{j-1}^{(t)}, X_j^{(t-1)}, \ldots, X_d^{(t-1)})$.[1] For each run, we simulate a Markov chain of length $10,000$ and select a sub-sample for every 100 step. We summarize below the diagnostic plots of the MCMC outputs from model 2. The Markov chain demonstrates good convergence properties. From the trace plots of the parameters $\lambda^{(t)}, \beta^{(t)}$, the log-likelihood and the log-prior (Figure 4a), it is observed that the chain is stationary from the beginning of the simulation, suggesting that no burn-in period is required and we can keep the entire chain. In addition, no obvious trends in the trace plots are observed, suggesting good explorations of the parameter space of the target posterior density.



(a) Trace plots of the MCMC outputs

(b) ACFs of the MCMC outputs

Figure 4: MCMC output diagnostics: (Left) Trace plots of the variables and key functions. (Right) ACFs of the trace plots. The trace plots are stationary from the beginning, and the ACFs fall off to zero rapidly. Both plots suggest good exploration of the parameter space and convergence properties of the MCMC.

Figure 4b summarizes the auto-correlation for the MCMC output sequences. The auto-correlation falls off to zero quickly, and the effective sample size of all parameters is close to the number of sub-samples 1000 (the number of iterations divided by the step size). We perform multiple runs from different start states and check those marginal distributions agree in Figure 5, which again confirms the good convergence of the chain to the target distribution.

[1] This proposal is chosen such that the prior and the proposal cancels when evaluating the acceptance probability, and we are only left with the likelihood ratio.

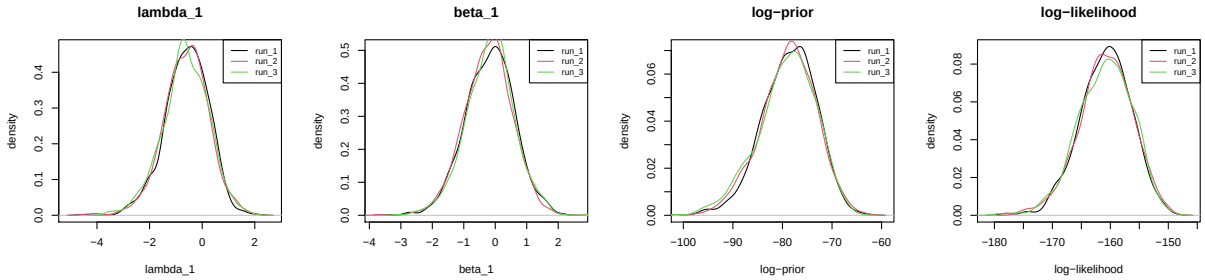


Figure 5: Density plots of the variables and key functions in three runs with different start states

4 Model Evaluation

4.1 Model selection

We use the Bayes factor $B_{1,2}$ to perform a model comparison between the model 1 (without seniority) and the model 2 (with seniority)

$$B_{1,2} = \frac{p\left(y \mid m=1\right)}{p\left(y \mid m=2\right)} = \frac{\int p\left(y \mid \theta, m=1\right) \pi_{m=1}\left(\theta\right) d\theta}{\int p\left(y \mid \theta, m=2\right) \pi_{m=2}\left(\theta\right) d\theta}$$

where we compute the marginal likelihoods $p\left(y \mid m\right)$ using the Bridge estimator

$$\hat{p} = \frac{\sum_t \pi\left(\theta^{(1,t)}\right) p\left(y \mid \theta^{(1,t)}\right) h\left(\theta^{(1,t)}\right)}{\sum_t \pi\left(\theta^{(2,t)}\right) h\left(\theta^{(2,t)}\right)}$$

where $\theta_t^{(1)} \sim \pi(\theta)$ are random samples simulated from the prior and $\theta_t^{(2)} \sim \pi(\theta \mid y)$ are simulated from the posterior. As a rule of thumb we choose $h(\theta) = \pi(\theta)^{-1} p\left(y \mid \theta\right)^{-1/2}$ so that the bridge estimator simplifies to

$$\hat{p} = \frac{\sum_t p\left(y \mid \theta^{(1,t)}\right)^{1/2}}{\sum_t p\left(y \mid \theta^{(2,t)}\right)^{-1/2}}$$

The estimated Bayes factor $\hat{B}_{1,2}$ has the order of magnitude of 10^5 and does not vary upon multiple computations, providing substantial evidence in favour of the model without seniority covariates than the model with.

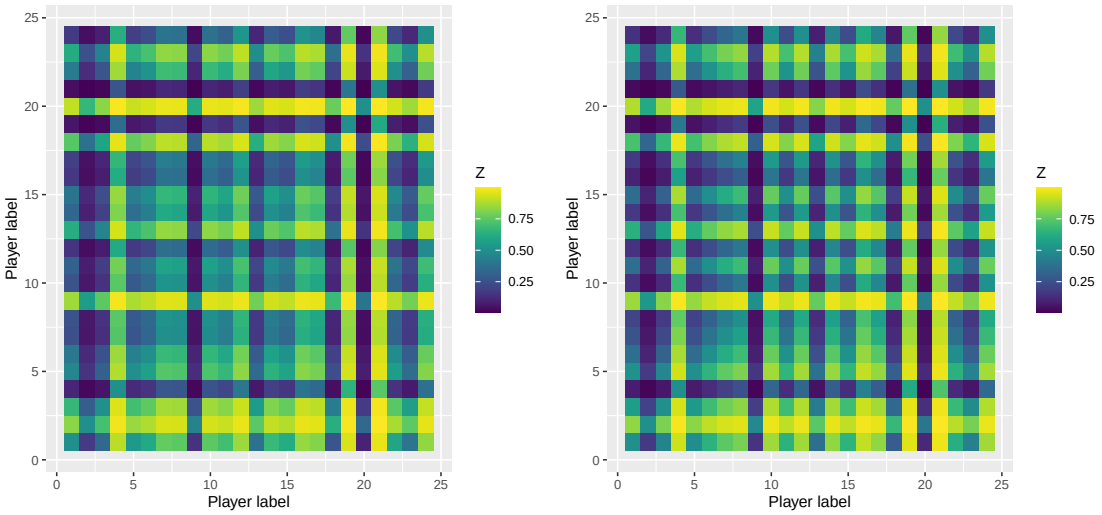
4.2 Player skills

We report the posterior mean and the HPD intervals of parameters from model one below. A similar summary is reported in Appendix B. From the posterior mean, we can conclude that players $21, 19, 4$ have the strongest skills, whereas players $20, 2, 9$ have the weakest. We perform a comparison of the relative skills of each pair $\{i, j\}$ of players in the dataset by considering the probability that i wins in the game involving only the two players (Figure 6). The sign convention adopted here is that the value in the heat map indicates the probability that the row player wins in a game against the column player.

$$\Pr\{i \text{ win}\} = \frac{\exp(\hat{\lambda}_{o_i})}{\exp(\hat{\lambda}_{o_i}) + \exp(\hat{\lambda}_{o_j})}$$

Param.	Post. Mean	HPD interval	ESS
λ_1	-0.67	(-2.18, 0.85)	1132
λ_2	-2.31	(-4.09, -0.44)	884
λ_3	-1.35	(-3.68, 0.52)	1000
λ_4	1.35	(-0.11, 2.97)	1000
λ_5	-0.45	(-1.41, 0.42)	1000
λ_6	-0.11	(-1.75, 1.50)	1000
λ_7	0.76	(0.07, 1.51)	1000
λ_8	0.30	(-0.54, 1.13)	1230
λ_9	-2.25	(-3.58, -1.01)	1000
λ_{10}	0.39	(-0.19, 1.05)	1000
λ_{11}	0.20	(-0.51, 0.94)	1000
λ_{12}	0.90	(-0.26, 1.81)	1000
λ_{13}	-0.82	(-2.08, 0.45)	960
λ_{14}	0.09	(-1.15, 1.44)	1000
λ_{15}	-0.26	(-1.07, 0.40)	1000
λ_{16}	1.17	(-0.23, 2.56)	1000
λ_{17}	0.91	(-0.28, 2.04)	1077
λ_{18}	-1.44	(-3.41, 0.05)	798
λ_{19}	2.03	(0.79, 3.36)	1000
λ_{20}	-2.36	(-4.57, -0.57)	825
λ_{21}	2.91	(0.23, 6.02)	879
λ_{22}	-0.19	(-1.84, 1.44)	1000
λ_{23}	-0.97	(-3.05, 1.26)	1000
λ_{24}	0.81	(-0.42, 2.12)	1000
$olpr$	-44.39	(-52.62, -36.49)	1000
$ollk$	-161.1	(-168.27, -153.29)	1000

Table 1: Summary of the posterior mean, HDP intervals and the ESS from model one



(a) Model 1 pairwise skill comparisons

(b) Model 2 pairwise skill comparisons

Figure 6: Pairwise comparisons of the relative skills of the players. The value in the heat map indicates the probability of the row player winning in a game against the column player.

4.3 A Simple Example

Consider a simple example where game 68 involving players $s = \{4, 7, 11, 5, 10, 14, 15, 13, 9\}$ was replayed (i.e. with the same players and same seniority values), we are interested in the probability that player 7 wins. We estimate this by summing the probabilities of all 7! permutations where player 7 comes at the top

$$\sum_{o_1=7} E_{\lambda,\beta|model} \left[\Pr\{O = o \mid \lambda, \beta\}\right] \quad \text{where} \quad O \in \mathcal{P}_s$$

The marginals are estimated using the Bridge estimator. In model one without seniority covariates, the probability is 17.4% and is stable upon repeated runs of the bridge estimation. With model two, when the seniority covariates are added, the probability that player 7 wins drops to 4.1%, indicating that seniority has non-trivial effects on the outcomes. Both estimates are stable upon multiple runs of the bridge estimator.

5 Conclusion

In sum, we conclude that the data provide more evidence in favour of the simpler observation model without seniority rank covariate. From the model, we identify the strongest and the weakest players that match our observations from the exploratory data analysis. It is also found that if game 68 was replayed, model one predicts that player 7 has 17.4% change of coming as the top player. The rejected model with seniority covariates has non-trivial effects on this prediction.

There are two noticeable limitations of our analysis: First, it is common that some of the players have not played directly with each other, and their relative skills are inferred indirectly from comparisons with the players they have commonly played with. This is likely to weaken our pairwise comparison. Second, we have a relatively low number of observations for several players

whose skill level is more likely to be affected by the choice of prior. Better analysis can be performed if more data are available.

There are two potential extensions of our analysis: First of all, we could consider a wider class of prior models, for example, t-distributions with different degrees of freedom that represents different beliefs about the certainty of the probability of each outcome before seeing the data. A larger degree of freedom represents a higher degree of certainty about the probability of each random permutation $O = (O_1, \ldots, O_m)$. We need to check that results are insensitive to the choice of priors. Second of all, we could develop more efficient computation methods. Each MH simulation for the model with β covariates took 30 minutes to complete. Better computational methods (e.g. alternative proposal distributions such as random walk) can facilitate convergence, enable us to reduce subsample size and make this process more efficient.

A R Codes

```
## Setup
library(MCMCpack)
library(lattice)
obs.dat = dget("CompRank21.txt")

num.play = 24
concat_o = c()
concat_e = c()
concat_rank = c()
first = c()
final = c()
for (i in 1:length(obs.dat)) {
  num.part = length(obs.dat[[i]]$o)
  concat_o = c(concat_o, obs.dat[[i]]$o)
  concat_e = c(concat_e, obs.dat[[i]]$e)
  concat_rank = c(concat_rank, c(1, rep(3, num.part - 2), 2))
  first = c(first, obs.dat[[i]]$e[1])
  final = c(final, obs.dat[[i]]$e[num.part])
}


## 1. EDA
first_level = c()
final_level = c()
for (i in 1:length(first)) {
  if (first[i] >= 1 & first[i] <= 5) {
    first_level = c(first_level, 1)
  }
  if (first[i] >= 6 &
      first[i] <= 10) {
    first_level = c(first_level, 2)
  }
  if (first[i] >= 11) {
    first_level = c(first_level, 3)
  }
  if (final[i] >= 1 & final[i] <= 5) {
    final_level = c(final_level, 1)
  }
  if (final[i] >= 6 &
      final[i] <= 10) {
    final_level = c(final_level, 2)
  }
  if (final[i] >= 11) {
    final_level = c(final_level, 3)
  }
```

```
}

library(ggplot2)
library(RColorBrewer)
t1 = prop.table(table(c(first_level, final_level), c(rep(0, 134), rep(1, 134))), 2)
pdf("seniority.pdf", width = 8, height = 6)
bp1 = barplot(
  t1,
  names = c("First player", "Last player"),
  main = "Distribution of seniority ranks among first-place and last-place players",
  ylab = "Distribution of seniority ranks",
  ylim = c(0, 1.2),
  col = brewer.pal(5, "Set2")
)
legend(
  x = 1.7,
  y = 1.22,
  legend = c("Seniority 1-5", "Seniority 6-10", "Seniority 11-16"),
  bty = "n",
  fill = brewer.pal(5, "Set2")
)
text(bp1, 0, round(t1[1, ], 2), cex = 1, pos = 3)
text(bp1, 0.56, round(t1[2, ], 2), cex = 1, pos = 3)
text(bp1, 0.91, round(t1[3, ], 2), cex = 1, pos = 3)
dev.off()

pdf("player.pdf", width = 12, height = 6)
counts = table(concat_rank, concat_o)
barplot(
  counts,
  main = "Distribution of rank among players",
  xlab = "Player label",
  ylab = "Count",
  col = c("darkblue", "red", "white"),
  legend = c("first place", "last place", "otherwise")
)
dev.off()


## 2. Observation Model
### Prior Function
lpr_l <- function(lambda_dim, beta) {
  log.prior <- sum(dt(lambda, df = 5, log = TRUE))
  return(log.prior)
}

lpr <- function(lambda, beta) {
```

```
  log.prior <-
    sum(dt(lambda, df = 5, log = TRUE)) + sum(dt(beta, df = 5, log = TRUE))
  return(log.prior)
}
### First Model
model_one <- function(lambda, oli) {
  # oli: list of player labels
  prob <- 1
  for (i in 1:length(oli)) {
    density <- 0
    for (j in i:length(oli)) {
      density <- density + exp(lambda[oli[j]])
    }
    prob <- prob * (exp(lambda[oli[i]]) / density)
  }
  return(prob)
}

llk_one <- function(lambda) {
  log.likelihood <- 0
  for (k in 1:length(obs.dat)) {
    oli <- obs.dat[[k]]$o
    log.likelihood <- log.likelihood + log(model_one(lambda, oli))
  }
  return(log.likelihood)
}
# llk_one(rt(num.play, 5))
K <- 100000
out <- matrix(NA, K, 1)
for (k in 1:K) {
  lambda <- rt(num.play, df = 5)
  out[k, ] <- model_one(lambda, c(1:24))
}
plot(as.mcmc(out))

### Second Model
model_two <- function(lambda, beta, oli, eli) {
  # oli: list of player labels
  # eli: list of seniority ranks
  prob <- 1
  for (i in 1:length(oli)) {
    density <- 0
    for (j in i:length(oli)) {
      density <- density + exp(lambda[oli[j]] + beta[eli[j]])
    }
    prob <- prob * exp(lambda[oli[i]] + beta[eli[i]]) / density
  }
```

```
  return(prob)
}

llk_two <- function(lambda, beta) {
  log.likelihood <- 0
  for (k in 1:length(obs.dat)) {
    oli <- obs.dat[[k]]$o
    eli <- obs.dat[[k]]$e
    log.likelihood <-
      log.likelihood + log(model_two(lambda, beta, oli, eli))
  }
  return(log.likelihood)
}
# llk_one(rt(num.play, 5), rt(num.play, 5))

### Validation -- Model 1
require(gtools)
K <- 10000
out_one <- matrix(NA, K, 6)
for (k in 1:K) {
  lambda = rt(24, df = 1)
  perm_o = permutations(3, 3, v = c(1, 2, 3))
  for (i in c(1:6)) {
    out_one[k, i] = model_one(lambda, perm_o[i,])
  }
}
pdf("one_t1.pdf", width = 6, height = 6)
plot(
  density(na.omit(out_one)[, 1]),
  xlab = expression("Likelihood p(O=o | lambda)"),
  ylab = "Density",
  main = c('Likelihood with t(1) prior')
)
for (i in c(2:6)) {
  lines(density(na.omit(out_one)[, i]), col = i)
}
legend(
  "topright",
  legend = c(
    "c(1, 2, 3)",
    "c(1, 3, 2)",
    "c(2, 1, 3)",
    "c(2, 3, 1)",
    "c(3, 1, 2)",
    "c(3, 2, 1)"
  ),
  col = c(1, 2, 3, 4, 5, 6),
```

```
  lty = 1,
  cex = 0.8
)
dev.off()

### Validation -- Model 2
K <- 10000
out_two <- matrix(NA, K, 36)
for (k in 1:K) {
  lambda = rt(24, df = 5)
  beta = rt(24, df = 5)
  perm_o = permutations(3, 3, v = c(1, 2, 3))
  perm_b = permutations(3, 3, v = c(1, 2, 3))
  for (i in c(1:6)) {
    for (j in c(1:6)) {
      out_two[k, 6 * (i - 1) + j] = model_two(lambda, beta, perm_o[i,], perm_b[j,])
    }
  }
}
pdf("two_t5.pdf", width = 8, height = 7)
plot(
  density(na.omit(out_two)[, 1]),
  xlab = expression("Likelihood p(O=o | lambda, beta)"),
  ylab = "Density",
  main = c('Likelihood with t(5) prior')
)
for (i in c(2:36)) {
  lines(density(na.omit(out_two)[, i]), col = i)
}
legend(
  "topright",
  legend = c("perm(o1, o2, o3), perm(e1, e2, e3)"),
  col = c(1),
  lty = 1,
  cex = 0.8
)
# legend(0.8, 4.7, lty=1, cex=0.8)
dev.off()


## 3. MCMC
### First MCMC
### Sampling from the first model ###
#initialise state for MCMC
K = 100000
SS = 100
```

```
# mcmc_1st: lambda=rep(0, num.play)
# mcmc_2nd: lambda=rnorm(num.play)
# mcmc_3rd: lambda=rt(num.play, 5)
lambda = rep(0, num.play)

olpr_l = lpr_l(lambda)
ollk = llk_one(lambda)

out.mcmc = matrix(NA, K / SS, num.play + 2)
colnames(out.mcmc) = c(c(1:24), 'olpr', 'ollk')
for (k in 1:K) {
  #proposal distribution is the prior so the MHR is L(theta';y)/L(theta;y)
  for (i in 1:num.play) {
    lambdap = lambda
    lambdap[i] = rt(1, df = 5)
    nllk = llk_one(lambdap)
    logMHR = nllk - ollk
    if (log(runif(1)) < logMHR) {
      lambda = lambdap
      olpr_l = lpr_l(lambda)
      ollk = nllk
    }
  }
  if (k %% SS == 0) {
    out.mcmc[k / SS, ] = c(lambda, olpr_l, ollk)
  }
}
### Second MCMC
### Sampling from the second model ###
K = 100000
SS = 100

# mcmc_1st
# lambda=rep(0, num.play)
# beta=rep(0, num.play)

# mcmc_2nd
lambda = rt(num.play, 5)
beta = rt(num.play, 5)

# mcmc_3rd
# lambda=rnorm(num.play)
# beta=rnorm(num.play)

olpr = lpr(lambda, beta)
ollk = llk_two(lambda, beta)
```

```
out.mcmc = matrix(NA, K / SS, num.play + num.play + 2)
colnames(out.mcmc) = c(sprintf("lambda_%d", 1:24),
                       sprintf("beta_%d", 1:24),
                       'olpr',
                       'ollk')

for (k in 1:K) {
  #proposal distribution is the prior so the MHR is L(theta';y)/L(theta;y)
  for (i in 1:num.play) {
    lambdap = lambda
    lambdap[i] = rt(1, df = 5)
    nllk = llk_two(lambdap, beta)
    logMHR = nllk - ollk
    if (log(runif(1)) < logMHR) {
      lambda = lambdap
      olpr = lpr(lambda, beta)
      ollk = nllk
    }
  }

  for (i in 1:num.play) {
    betap = beta
    betap[i] = rt(1, df = 5)
    nllk = llk_two(lambda, betap)
    logMHR = nllk - ollk
    if (log(runif(1)) < logMHR) {
      beta = betap
      olpr = lpr(lambda, beta)
      ollk = nllk
    }
  }
  if (k %% SS == 0) {
    out.mcmc[k / SS, ] = c(lambda, beta, olpr, ollk)
  }
}

### Output Analysis
model_two_mcmc_1st = readRDS("model_two_mcmc_1st.Rda")
model_two_mcmc_2nd = readRDS("model_two_mcmc_2nd.Rda")
model_two_mcmc_3rd = readRDS("model_two_mcmc_3rd.Rda")
pdf("xyplot.pdf", width = 7, height = 7)
xyplot(out.mcmc <-
         as.mcmc(model_two_mcmc_1st[, c(49, 50, 1, 13, 25, 37)]))
dev.off()
pdf("overlay.pdf", width = 12, height = 3)
par(mfrow = c(1, 4))
comps = c(1, 25, 49, 50)
```

```
mains = c("lambda_1", "beta_1", "log-prior", "log-likelihood")
for (i in 1:4) {
  plot(
    density(model_two_mcmc_1st[, comps[i]]),
    main = mains[i],
    xlab = mains[i],
    ylab = "density"
  )
  lines(density(model_two_mcmc_2nd[, comps[i]]), col = 2)
  lines(density(model_two_mcmc_3rd[, comps[i]]), col = 3)
  legend(
    "topright",
    legend = c("run_1", "run_2", "run_3"),
    col = c(1, 2, 3),
    lty = 1,
    cex = 0.8
  )
}
dev.off()
model_one_mcmc_1st = readRDS("model_one_mcmc_1st.Rda")
effectiveSize(as.mcmc(model_one_mcmc_1st))
pdf("acf.pdf", width = 8, height = 8)
par(mfrow = c(2, 2))
for (i in 1:4) {
  plot(
    acf(model_two_mcmc_1st[, comps[i]], lag.max = 200, plot = F),
    main = mains[i],
    xlab = "LAG",
    ylab = "ACF",
    type = 'l',
    ann = F,
    xaxp = c(0, 200, 2),
    yaxp = c(0, 1, 1)
  )
  text(50, 0.8, colnames(model_two_mcmc_1st)[comps[i]])
}
dev.off()


## 4. Model Validation
### Bayes Factor
OP.one = readRDS("model_one_mcmc_1st.Rda")
OP.two = readRDS("model_two_mcmc_1st.Rda")
# Bridge estimate marginal lkd of model one
n.samp = dim(OP.one)[1]
n.samp.p = 100 * n.samp
lambda.pri = matrix(NA, n.samp.p, num.play)
```

```
for (k in 1:n.samp.p) {
  lambda.pri[k, ] = rt(num.play, 5)
}
lambda.pos = OP.one[, 1:num.play]
L.pri = rep(NA, n.samp.p)
L.pos = rep(NA, n.samp)
for (k in 1:n.samp) {
  L.pos[k] = llk_one(lambda.pos[k, ])
}
for (k in 1:n.samp.p) {
  L.pri[k] = llk_one(lambda.pri[k, ])
}
ml.one = mean(exp(L.pri / 2)) / mean(exp(-L.pos / 2))

# Bridge estimate marginal lkd of model two
n.samp = dim(OP.two)[1]
n.samp.p = 100 * n.samp
lambda.pri = matrix(NA, n.samp.p, num.play)
beta.pri = matrix(NA, n.samp.p, num.play)
for (k in 1:n.samp.p) {
  lambda.pri[k, ] = rt(num.play, 5)
}
for (k in 1:n.samp.p) {
  beta.pri[k, ] = rt(num.play, 5)
}
lambda.pos = OP.two[, 1:num.play]
beta.pos = OP.two[, (num.play + 1):(2 * num.play)]
L.pri = rep(NA, n.samp.p)
L.pos = rep(NA, n.samp)
for (k in 1:n.samp) {
  L.pos[k] = llk_two(lambda.pos[k, ], beta.pos[k, ])
}
for (k in 1:n.samp.p) {
  L.pri[k] = llk_two(lambda.pri[k, ], beta.pri[k, ])
}
ml.two = mean(exp(L.pri / 2)) / mean(exp(-L.pos / 2))

Bayes.Factor = ml.two / ml.one
Bayes.Factor


### Posterior Mean
summary(OP.one)
round(HPDinterval(as.mcmc(OP.one)), 2)


## 5. Relative Skills
```

```
### Model 1
odds_matrix_one = matrix(NA, num.play, num.play)
for (i in 1:num.play) {
  for (j in 1:num.play) {
    skill_i = exp(mean(OP.two[, i]))
    skill_j = exp(mean(OP.two[, j]))
    odds_matrix_one[i, j] = skill_i / (skill_i + skill_j)
  }
}

### Model 2
odds_matrix_two = matrix(NA, num.play, num.play)
for (i in 1:num.play) {
  for (j in 1:num.play) {
    skill_i = exp(mean(OP.two[, i]) + mean(OP.two[, num.play + i]))
    skill_j = exp(mean(OP.two[, j]) + mean(OP.two[, num.play + j]))
    odds_matrix_two[i, j] = skill_i / (skill_i + skill_j)
  }
}

library(ggplot2)
library(viridis)
x = c(1:24)
y = c(1:24)
data = expand.grid(X = x, Y = y)
# data$Z=as.vector(odds_matrix_one)
data$Z = as.vector(odds_matrix_one)

pdf("heat_one.pdf", width = 5.3, height = 5)
ggplot(data, aes(X, Y, fill = Z)) +
  labs(x = "Player label", y = "Player label") +
  geom_tile() +
  scale_fill_viridis(discrete = FALSE)
dev.off()


## 6. Counterexample
require(gtools)
perm = permutations(8, 8, v = obs.dat[[68]]$o[-2])

### Model 1
model_one_mcmc = readRDS("model_one_mcmc_1st.Rda")
K = 1000
perm = permutations(8, 8, v = obs.dat[[68]]$o[-2])
result_one = matrix(NA, K, dim(perm)[1])
for (k in 1:K) {
  lambda = rep(NA, num.play)
```

```
  for (i in c(1:num.play)) {
    dens = density(model_one_mcmc[, i])
    lambda[i] = sample(dens$x, 1, replace = TRUE, prob = dens$y)
  }
  for (j in c(1:dim(perm)[1])) {
    po = c(7, perm[j,])
    result_one[k, j] = model_one(lambda, po)
  }
}
saveRDS(result_one, "result_one.Rda")
sum(colMeans(result_one))

### Model 2
model_two_mcmc = readRDS("model_two_mcmc_1st.Rda")
K = 1000
perm = permutations(8, 8, v = obs.dat[[68]]$o[-2])
result_two = matrix(NA, K, dim(perm)[1])
for (k in 1:K) {
  lambda = rep(NA, num.play)
  beta = rep(NA, num.play)
  for (i in c(1:num.play)) {
    dens = density(model_two_mcmc[, i])
    lambda[i] = sample(dens$x, 1, replace = TRUE, prob = dens$y)
  }
  for (i in c(1:num.play)) {
    dens = density(model_two_mcmc[, (i + 1):(2 * i)])
    beta[i] = sample(dens$x, 1, replace = TRUE, prob = dens$y)
  }
  for (j in c(1:dim(perm)[1])) {
    po = c(7, perm[j,])
    pe = obs.dat[[68]]$e[order(po)]
    result_two[k, j] = model_two(lambda, beta, po, pe)
  }
}
saveRDS(result_two, "result_two.Rda")

sum(colMeans(result_two))
```

B Posterior Estimates from Model Two

Param.	Post. Mean	HPD interval	ESS	Param.	Post. Mean	HPD interval	ESS
λ_1	-0.59	(-2.20, 0.97)	1082	β_1	-0.18	(-1.75, 1.28)	896
λ_2	-2.14	(-4.21, -0.27)	876	β_2	-0.22	(-1.21, 0.91)	1000
λ_3	-1.28	(-3.63, 0.75)	1000	β_3	0.31	(-0.62, 1.26)	1000
λ_4	1.55	(-0.25, 3.37)	1000	β_4	0.17	(-0.81, 1.07)	1000
λ_5	-0.44	(-1.55, 0.66)	1000	β_5	-0.36	(-1.10, 0.47)	913
λ_6	-0.17	(-1.80, 1.44)	1000	β_6	-0.02	(-0.88, 0.76)	1000
λ_7	0.56	(-0.52, 1.75)	988	β_7	-0.25	(-1.25, 0.57)	1000
λ_8	0.50	(-0.37, 1.49)	1000	β_8	0.09	(-1.00, 1.05)	1000
λ_9	-2.36	(-3.96, -0.84)	1000	β_9	-0.12	(-1.10, 0.85)	1188
λ_{10}	0.49	(-0.33, 1.25)	1000	β_{10}	0.51	(-0.59, 1.58)	1139
λ_{11}	0.24	(-0.71, 1.14)	1000	β_{11}	-0.42	(-1.92, 1.31)	1000
λ_{12}	1.13	(-0.06, 2.24)	1150	β_{12}	-0.12	(-1.33, 0.96)	1000
λ_{13}	-1.08	(-2.51, 0.34)	886	β_{13}	-0.23	(-1.24, 0.68)	1000
λ_{14}	0.14	(-1.24, 1.72)	1000	β_{14}	0.72	(-0.85, 2.10)	1000
λ_{15}	-0.12	(-0.89, 0.82)	1000	β_{15}	-0.04	(-1.55, 1.65)	1000
λ_{16}	1.08	(-0.38, 2.41)	843	β_{16}	0.38	(-1.83, 2.29)	1000
λ_{17}	0.88	(-0.45, 2.15)	1000	β_{17}	-0.04	(-2.46, 2.45)	1000
λ_{18}	-1.61	(-3.53, -0.01)	1000	β_{18}	-0.04	(-2.83, 2.31)	1000
λ_{19}	2.15	(0.70, 3.48)	1000	β_{19}	-0.01	(-2.37, 2.58)	1000
λ_{20}	-2.82	(-5.14, -0.56)	536	β_{20}	0.02	(-2.46, 2.29)	1486
λ_{21}	2.60	(-0.29, 5.58)	789	β_{21}	0.08	(-2.23, 2.93)	805
λ_{22}	-0.24	(-2.03, 1.49)	1000	β_{22}	-0.01	(-2.67, 2.49)	1000
λ_{23}	-1.04	(-3.58, 1.04)	847	β_{23}	0.01	(-2.50, 2.97)	1209
λ_{24}	1.02	(-0.62, 2.60)	1000	β_{24}	0.01	(-2.36, 2.55)	1159
olpr	-78.64	(-88.97, -68.44)	770	*ollk*	-160.8	(-170.08, -153.11)	1000

Table 2: Summary of the posterior mean, HDP intervals and the ESS from model two