STA531 Final Project

Eric Su

2019-04-09

R Markdown

```
Tennis = read_csv("Data.csv")

Tennis_df = Tennis %>%
  mutate(Comment = Comment %>% str_replace("Compleed", "Completed")) %>%
  filter(Comment == "Completed", Round != "0th Round", !(`Best of` %in% c(0, 1))) %>%
  select(Location:LRank, WPts:LPts) %>%
  mutate(Series = Series %>% recode("International"        = "ATP250",
                                    "International Series" = "ATP250",
                                    "International Gold"   = "ATP500",
                                    "Masters"              = "Masters 1000")) %>%
  separate(Date, c("Day", "Month", "Year"), "/") %>%
  mutate(WPts  = as.numeric(WPts),
         LPts  = as.numeric(LPts),
         WRank = as.numeric(WRank),
         LRank = as.numeric(LRank)) %>%
  mutate(LRank_temp = LRank,
         WPts_temp = WPts,
         LRank = ifelse(Year == "2007" & Series == "Masters Cup", WPts_temp, LRank),
         WPts = ifelse(Year == "2007" & Series == "Masters Cup", LRank_temp, WPts),
         LRank = ifelse(Year == "2012" & Tournament == "Brisbane International" & Round == "The Final",
                        WPts_temp, LRank),
         WPts = ifelse(Year == "2012" & Tournament == "Brisbane International" & Round == "The Final",
                       LRank_temp, WPts)) %>%
  select(-LRank_temp, -WPts_temp)
```

```
perm_year = Tennis_df %>%
  gather(key = "WL", value = "Player", Winner, Loser) %>%
  mutate(Pts = as.numeric(ifelse(WL == "Winner", WPts, LPts)),
         logPts = log(Pts),
         Rank = as.numeric(ifelse(WL == "Winner", WRank, LRank)),
         WL = ifelse(WL == "Winner", 1, 0)) %>%
  select(Series, Pts, Rank, WL) %>%
  group_by(Series) %>%
  summarise(Pts = mean(Pts, na.rm = T),
            Rank = mean(Rank, na.rm = T))
perm_year
```

```
perm_year = Tennis_df %>%
  gather(key = "WL", value = "Player", Winner, Loser) %>%
  mutate(Pts = as.numeric(ifelse(WL == "Winner", WPts, LPts)),
         logPts = log(Pts),
         Rank = as.numeric(ifelse(WL == "Winner", WRank, LRank)),
         WL = ifelse(WL == "Winner", 1, 0)) %>%
  select(Pts, logPts, Rank, WL)
cor(perm_year, use = "complete.obs")
```

```
thresh = 100
player_names = which(c(Tennis_df$Winner, Tennis_df$Loser) %>% table() < thresh) %>% names()
Tennis_df = Tennis_df %>%
  mutate(Winner = Winner %>% sapply(function(x) if(x %in% player_names){return("Other")} else{return(x)
         Loser  = Loser  %>% sapply(function(x) if(x %in% player_names){return("Other")} else{return(x)
```

```
name = c("Federer R.", "Nadal R.", "Djokovic N.", "Ferrer D.", "Roddick A.", "Murray A.", "Berdych T.",
player_name = name[c(1:3, 6)]
perm_year = Tennis_df %>% filter(Winner %in% player_name | Loser %in% player_name) %>%
  gather(key = "WL", value = "Player", Winner, Loser) %>%
  mutate(Pts = ifelse(WL == "Winner", WPts, LPts),
         Year = as.numeric(Year)) %>%
  filter(Player %in% player_name) %>%
  group_by(Year, Player) %>%
  select(WL, Pts) %>%
  summarise(Win = length(which(WL == "Winner")) / length(WL),
            Pts = mean(as.numeric(Pts), na.rm = T))
cor(perm_year[, 3:4], use = "complete.obs")

perm_year_plot = perm_year %>% gather(key = "key", value = "value", Win, Pts) %>%
  mutate(Player = factor(Player))
ggplot(perm_year_plot, aes(x = Year, y = value, color = Player))+
  geom_line(size = 1.5)+
  facet_grid(key~., scales = "free_y")+
  theme_bw()+
  labs(title = "Average points and win percentage of the Big 4")
```

```
player_name = name[c(1:3, 6)]
perm_year = Tennis_df %>% filter(Winner %in% player_name & Loser %in% player_name) %>%
  gather(key = "WL", value = "Player", Winner, Loser) %>%
  mutate(Pts = ifelse(WL == "Winner", WPts, LPts),
         Year = as.numeric(Year)) %>%
  group_by(Year, Player) %>%
  select(WL, Pts) %>%
  summarise(Win = length(which(WL == "Winner")) / length(WL),
            Pts = mean(as.numeric(Pts), na.rm = T),
            n = n())
cor(perm_year[, 3:4], use = "complete.obs")

perm_year_plot = perm_year %>% gather(key = "key", value = "value", Win, Pts) %>%
  mutate(Player = factor(Player))
ggplot(perm_year_plot, aes(x = Year, y = value, color = Player))+
  geom_line(size = 1.5)+
  facet_grid(key~., scales = "free_y")+
  theme_bw()+
  labs(title = "Average points and win percentage(against each other) of the Big 4")
```

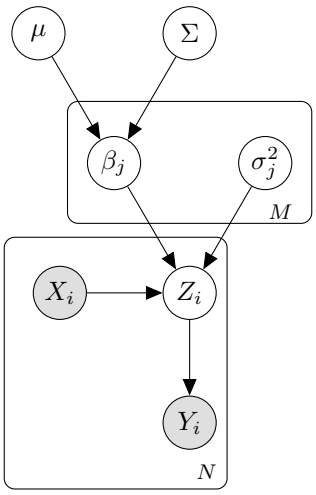
```
name = c("Federer R.", "Nadal R.", "Djokovic N.", "Ferrer D.", "Roddick A.", "Murray A.", "Berdych T.",
player_name = name[c(1:3, 6)]
perm_year = Tennis_df %>% filter(Winner %in% player_name | Loser %in% player_name) %>%
  gather(key = "WL", value = "Player", Winner, Loser) %>%
  mutate(Year = as.numeric(Year)) %>%
  filter(Player %in% player_name) %>%
```

```
  group_by(Year, Player, Series) %>%
  select(WL) %>%
  summarise(Win = length(which(WL == "Winner")) / length(WL))

ggplot(perm_year, aes(x = Year, y = Win, color = factor(Series)))+
  geom_line(size = 1.5)+
  facet_grid(Player~.)+
  theme_bw()+
  labs(title = "Win percentage of the Big 4 for different series")
```

Model

The graphical model this project uses can be expressed using the graph below.



where matches are represented using the index $i = 1, ..., N$ and players are represented using the index $j = 1, ..., M$ with variables

X_i : Vector of external conditions (Series, Court, Surface, Round, Best of 3/4, Opponent rank)
Y_i : Match outcome
$Z_{i,1:2}$: Performance level of the two players
β_j : Vector of regression coefficients
σ_j^2 : Variance parameter for performance
μ : Mean hyperparameter for β_j
Σ : Variance hyperparameter for β_j

The outcome of each match will be modelled using the distribution:

$$Y_i = \begin{cases} 1 & \text{if } Z_{i,1} \geq Z_{i,2} \\ 0 & \text{if } Z_{i,1} < Z_{i,2} \end{cases}$$

For player j, his performance level in match i will be a linear combination of X_i(external factors) as shown below.

$$Z_{i,1 \text{ or } 2}^{(j)} = X_i^T \beta_j + \epsilon_{i,j} \quad \epsilon_{i,j} \sim N(0, \sigma_j^2)$$

with

$$\beta_j \sim N(\mu, \Sigma)$$

The prior distributions for this model will be

$$\begin{aligned}\mu &\sim N(\mu_0, \Lambda_0)\\ \Sigma &\sim \text{inverse-Wishart}(\eta_0, S_0^{-1})\\ \sigma_j^2 &\sim \text{inverse-gamma}(\nu_0/2, \nu_0\sigma_0^2/2)\end{aligned}$$

The parameters of our model would be estimated using a Gibbs sampler with full conditionals as follows. We will use Z_j and X_j to indicate all (matrix) performace levels and external conditions for player j and n_j to represent the number of matches player j is involved.

$$p(Z_{i,1}^{(j)} \mid X_i, Y_i, Z_{i,2}, \beta_j, \sigma_j^2) = \begin{cases} \text{Truncated Normal}(X_i\beta_j, \sigma_j^2, Z_{i,2}, \infty) & \text{if } Y_i = 1\\ \text{Truncated Normal}(X_i\beta_j, \sigma_j^2, -\infty, Z_{i,2}) & \text{if } Y_i = 0\end{cases}$$

$$p(Z_{i,2}^{(j)} \mid X_i, Y_i, Z_{i,1}, \beta_j, \sigma_j^2) = \begin{cases} \text{Truncated Normal}(X_i\beta_j, \sigma_j^2, -\infty, Z_{i,1}) & \text{if } Y_i = 1\\ \text{Truncated Normal}(X_i\beta_j, \sigma_j^2, Z_{i,1}, \infty) & \text{if } Y_i = 0\end{cases}$$

$$p(\beta_j \mid X_j, Z_j, \mu, \Sigma) = N((\Sigma^{-1} + X_j^T X_j/\sigma_j^2)^{-1}(\Sigma^{-1}\mu + X_j^T Z_j/\sigma_j^2), (\Sigma^{-1} + X_j^T X_j/\sigma_j^2)^{-1})$$

$$p(\mu \mid \beta_{1:M}, \Sigma) = N((\Lambda_0^{-1} + M\Sigma^{-1})^{-1}(\Lambda_0^{-1}\mu_0 + M\Sigma^{-1}\bar{\beta}), (\Lambda_0^{-1} + M\Sigma^{-1})^{-1})$$

$$p(\Sigma \mid \beta_{1:M}, \mu) = \text{inverse-Wishart}(\eta_0 + M, (S_0 + \sum_{j=1}^{M}(\beta_j - \mu)(\beta_j - \mu)^T)^{-1})$$

$$p(\sigma_j^2 \mid \beta_j, X_j) = \text{inverse-gamma}((\nu_0 + n_j)/2, [\nu_0\sigma_0^2 + \sum_{i=1}^{n_j}(Z_{i,j} - \beta_j^T X_{i,j})^2]/2)$$

```
unique_player = c(Tennis_df$Winner, Tennis_df$Loser) %>% unique()
X = model.matrix(~ Series + Court + Surface + Round + `Best of` + WRank + LRank, data = Tennis_df)
X_win = X[, -ncol(X)]
X_lose = X[, -(ncol(X) - 1)]

n_player = length(unique_player)
n_match = nrow(Tennis_df)
n_beta = ncol(X) - 1
```

```
player_index = data.frame(player = unique_player, index = 1:n_player)
win_dex = plyr::mapvalues(Tennis_df$Winner, player_index$player, player_index$index) %>% as.numeric()
lose_dex = plyr::mapvalues(Tennis_df$Loser, player_index$player, player_index$index) %>% as.numeric()

player_match_index = vector("list", n_player)
for (i in 1:n_player) {
  wins = which(Tennis_df$Winner == player_index$player[i])
  loses = which(Tennis_df$Loser == player_index$player[i])
  player_match_index[[i]] = data.frame(index = c(wins, loses),
                                       WL = c(rep(1, length(wins)), rep(0, length(loses))))
}
```

```
mu_0 = rep(0, ncol(X))
L_0 = diag(1, ncol(X))
iL_0 = solve(L_0)
eta_0 = 1
S_0 = t(L_0 - colMeans(L_0)) %*% (L_0 - colMeans(L_0))
nu_0 = 2
sigma_0 = 1
```

```r
n_Gibbs = 1000 + 10000
n_player = length(unique_player)
n_match = nrow(Tennis_df)
n_beta = ncol(X) - 1

Z = matrix(NA, ncol = 2, nrow = n_match)

BETA = vector("list", n_player)
for (i in 1:n_player) {
  BETA[[i]] = matrix(NA, ncol = n_beta, nrow = n_Gibbs)
}

MU = matrix(NA, ncol = n_beta, nrow = n_Gibbs)

SIGMA = vector("list", n_Gibbs)
for (i in 1:n_Gibbs) {
  SIGMA[[i]] = matrix(NA, ncol = n_beta, nrow = n_beta)
}

sig = matrix(NA, ncol = n_player, nrow = n_Gibbs)
```

```r
library(truncnorm)
library(mvtnorm)
beta = matrix(NA, ncol = n_beta, nrow = n_player)
iSigma = solve(diag(n_beta))

for (i in 1:n_Gibbs) {
  #sample Z
  beta_win = beta[win_dex, ]
  beta_lose = beta[lose_dex, ]
  Z[, 1] = rtruncnorm(n_match, a = Z[, 2], b = Inf, mean = X_win %*% t(beta_win), sd = sqrt(sig))
  Z[, 2] = rtruncnorm(n_match, a = -Inf, b = Z[, 1], mean = X_lose %*% t(beta_lose), sd = sqrt(sig))

  #sample BETA
  for (j in 1:n_player) {
    beta_var = solve(solve(SIGMA) + )
    beta_mu =
    beta_samp = rmvnorm(1, )
    beta[j, ] = beta_samp
    BETA[[j]][i, ] = beta_samp
  }

  #sample MU
  mu_var = solve(iL_0 + n_player * iSigma)
  mu_mean = mu_var %*% (iL_0 %*% mu_0 + iSigma %*% colSums(beta))
  mu_samp = rmvnorm(1, mu_mean, mu_var)
  mu = mu_samp
  MU[i, ] = mu_samp

  #sample SIGMA
  M_mu = matrix(mu, n_player, n_beta, byrow = TRUE)
  iSigma = rWishart(1, eta_0 + n_player, solve(S_0 + t(BETA - M_mu) %*% (BETA - M_mu)))

  #sample sig
```

```
}
```