Making an Initial Model

First we need to load the data and the BradleyTerry2 library. Then we can construct the initial model.

```
library(BradleyTerry2)

matches <- read.csv("../data/matches.csv")

model <- BTm(cbind(wins1,
                   wins2),
             as.factor(player1), as.factor(player2),
             data = matches)
```

Then, we can open a new file for the predictions.

```
# open the 2023 file and make predictions
matches_2023 <- read.csv("../data/atp_matches_2023.csv")
pred_df <- matches_2023

# make training data
players <- unique(c(matches$player1, matches$player2))
matches_2023 <- matches_2023[matches_2023$winner_name %in% players & matches_2023$loser_name %in% playe
matches_2023 <- matches_2023[, c("winner_name", "loser_name")]
```

We now need to make all of the coefficients in the model positive. This is done by setting the 'worst' player as the reference category.

```
s <- summary(model)
names <- sort(players)

# make all coefficients positive by setting the worst player
# as the reference category
if (min(s$coefficients) < 0) {
  # redo the model with refcat set to the name of the worst player
  worst_player <- names[which.min(s$coefficients[,1])+1]
  model <- BTm(cbind(wins1, wins2),
               as.factor(player1), as.factor(player2), data = matches,
               refcat = worst_player)
}
```

If this statement does get called, we need to update the value of s.

```
s <- summary(model)
df_coeff <- as.data.frame(s$coefficients)
rownames(df_coeff) <- lapply(rownames(df_coeff),
                             function(x) substr(x, 3, nchar(x)))
```

But if not, then 'worst_player' will be NULL. Also, the coefficients of the worst player are not included in the summary, so we need to add them.

```
# if worst_player is NULL
if (is.null(worst_player)) {
  worst_player <- names[1]
}

df_coeff[worst_player,] <- rep(0, 2)
```

Now we can make the predictions.

```
predict <- function(player1, player2) {
  lambda1 <- df_coeff[player1,1]
  lambda2 <- df_coeff[player2,1]

  pred <- exp(lambda1) / (exp(lambda1) + exp(lambda2))
  return(c(pred>0.5, pred))
}

# make predictions
draws <- pred_df[,c("winner_name","loser_name")]
draws <- draws[draws$winner_name %in% players & draws$loser_name %in% players, ]

for (i in 1:nrow(draws)) {
  player1 <- draws$winner_name[i]
  player2 <- draws$loser_name[i]
  pred <- predict(player1, player2)
  draws$pred[i] <- pred[2]
}
```

Once we have made the predictions, we can find the accuracy of the model.

```
# find the accuracy
draws$correct <- draws$pred > 0.5
sum(draws$correct) / nrow(draws)
```

```
## [1] 0.6177452
```

The model achieves 0.62 accuracy.