Surface

Use weighting by surface from Sipko and Knottenbelt.

```
library(TennisBT)
```

```
##
## Attaching package: 'TennisBT'
```

```
## The following object is masked from 'package:stats':
##
##     predict
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
data <- TennisTidyr(2013, 2022, 2023, 20)
train <- data$train
test <- data$test
players <- data$main_names
```

They use the correlation between surfaces a and b, according to the formula:

$$\rho_{a,b} = \frac{\sum_{i=1}^{n}(a_i - \bar{a})(b_i - \bar{b})}{(n-1)s_a s_b}$$

a_i = percentage of matches won by player i on surface a b_i = percentage of matches won by player i on surface b s_a = standard deviation of percentage of matches won on surface a s_b = standard deviation of percentage of matches won on surface b $\bar{a}$ = mean percentage of matches won on surface a $\bar{b}$ = mean percentage of matches won on surface b n = number of players

```
n <- length(players)
win_counts <- as.data.frame(matrix(0, nrow = n, ncol = 3))
colnames(win_counts) <- c("Grass", "Clay", "Hard")
rownames(win_counts) <- players

for (player in players) {
  train_player <- train[train$winner_name == player,]
  player_count <- nrow(train_player)
```

```
  win_counts[player,"Grass"] <- nrow(train_player[train_player$surface == 'Grass',])/player_count
  win_counts[player,"Clay"] <- nrow(train_player[train_player$surface == 'Clay',])/player_count
  win_counts[player,"Hard"] <- nrow(train_player[train_player$surface == 'Hard',])/player_count
}
win_counts <- na.omit(win_counts)
row_sub = apply(win_counts, 1, function(row) all(row !=0 ))
win_counts <- win_counts[row_sub,]

grass_mean <- mean(win_counts$Grass)
grass_sd <- sd(win_counts$Grass)
clay_mean <- mean(win_counts$Clay)
clay_sd <- sd(win_counts$Clay)
hard_mean <- mean(win_counts$Hard)
hard_sd <- sd(win_counts$Hard)
```

```
grass_clay <- sum((win_counts$Grass-grass_mean)*(win_counts$Clay-clay_mean))/((n-1)*grass_sd*clay_sd)
hard_clay <- sum((win_counts$Hard-hard_mean)*(win_counts$Clay-clay_mean))/((n-1)*hard_sd*clay_sd)
grass_hard <- sum((win_counts$Hard-hard_mean)*(win_counts$Grass-grass_mean))/((n-1)*hard_sd*grass_sd)
```

Now create a dataset with the weightings

```
weighting <- as.data.frame(diag(3))
rownames(weighting) <- c("Grass", "Hard", "Clay")
colnames(weighting) <- c("Grass", "Hard", "Clay")
weighting['Grass', 'Hard'] <- grass_hard
weighting['Hard', 'Grass'] <- grass_hard
weighting['Clay', 'Hard'] <- hard_clay
weighting['Hard', 'Clay'] <- hard_clay
weighting['Clay', 'Grass'] <- grass_clay
weighting['Grass', 'Clay'] <- grass_clay
```

Now create three new columns in the data, weighting according to the match that the surface was played on.

```
# Find the surface of each match in train, and add the corresponding weighting for each surface
for (i in 1:nrow(train)) {
  train[i, 'grass_weight'] <- weighting[train[i, 'surface'], 'Grass']
  train[i, 'clay_weight'] <- weighting[train[i, 'surface'], 'Clay']
  train[i, 'hard_weight'] <- weighting[train[i, 'surface'], 'Hard']
}
```

Train three different models weighted by each of the surfaces.

```
library(BradleyTerry2)
btm_grass <- BTm(outcome = 1, factor(winner_name, levels=players), factor(loser_name, levels=players),
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
btm_clay <- BTm(outcome = 1, factor(winner_name, levels=players), factor(loser_name, levels=players), w
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
btm_hard <- BTm(outcome = 1, factor(winner_name, levels=players), factor(loser_name, levels=players), w
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

Finally predict the models.

```
# Create a dataframe with the Bradley-Terry estimates for each surface
btm_grass_df <- as.data.frame(BTabilities(btm_grass))
btm_clay_df <- as.data.frame(BTabilities(btm_clay))
btm_hard_df <- as.data.frame(BTabilities(btm_hard))

coeff_frames <- list('Grass' = btm_grass_df, 'Clay' = btm_clay_df,
                     'Hard' = btm_hard_df)

draws <- test[, c("winner_name", "loser_name", "surface")]
draws$pred <- NA

# predict the test set using a different model based on the surface
for (i in 1:nrow(draws)) {
  player1 <- draws$winner_name[i]
  player2 <- draws$loser_name[i]
  draw_surface <- draws$surface[i]

  pred <- predict(player1, player2, as.data.frame(coeff_frames[draw_surface]))
  draws$pred[i] <- pred[2]
}
```

```
score_predictions(draws)
```

```
## $accuracy
## [1] 0.6258675
##
## $avg_probability
## [1] 0.6806244
##
## $avg_prob_se
## [1] 0.003567084
##
## $avg_log_probability
## [1] -0.3981347
##
## $abg_log_prob_se
## [1] 0.005184012
```

These results are not as good as the time weighting model. We conduct a grid search below to find the optimal parameters. WARNING this code will take approximately 4 hours to run (machine dependent), hence it is stored in a function so that the markdown will knit.

```
  grid_search_params <- function() {
  best_ijk <- c(0,0,0)
  best_accuracy <- 0
```

```
for (i_ in seq(0,1,0.1)) {
  for (j_ in seq(0,1,0.1)) {
    for (k_ in seq(0,1,0.1)) {
      grass_clay <- i_
      hard_clay <- j_
      grass_hard <- k_

      weighting <- as.data.frame(diag(3))
      rownames(weighting) <- c("Grass", "Hard", "Clay")
      colnames(weighting) <- c("Grass", "Hard", "Clay")
      weighting['Grass', 'Hard'] <- grass_hard
      weighting['Hard', 'Grass'] <- grass_hard
      weighting['Clay', 'Hard'] <- hard_clay
      weighting['Hard', 'Clay'] <- hard_clay
      weighting['Clay', 'Grass'] <- grass_clay
      weighting['Grass', 'Clay'] <- grass_clay

      for (i in 1:nrow(train)) {
        train[i, 'grass_weight'] <- weighting[train[i, 'surface'], 'Grass']
        train[i, 'clay_weight'] <- weighting[train[i, 'surface'], 'Clay']
        train[i, 'hard_weight'] <- weighting[train[i, 'surface'], 'Hard']
      }

      btm_grass <- BTm(outcome = 1, factor(winner_name, levels=players), factor(loser_name, levels=pla
      btm_clay <- BTm(outcome = 1, factor(winner_name, levels=players), factor(loser_name, levels=play
      btm_hard <- BTm(outcome = 1, factor(winner_name, levels=players), factor(loser_name, levels=play

      # Create a dataframe with the Bradley-Terry estimates for each surface
      btm_grass_df <- as.data.frame(BTabilities(btm_grass))
      btm_clay_df <- as.data.frame(BTabilities(btm_clay))
      btm_hard_df <- as.data.frame(BTabilities(btm_hard))

      coeff_frames <- list('Grass' = btm_grass_df, 'Clay' = btm_clay_df,
                           'Hard' = btm_hard_df)

      draws <- test[, c("winner_name", "loser_name", "surface")]
      draws$pred <- NA

      # predict the test set using a different model based on the surface
      for (i in 1:nrow(draws)) {
        player1 <- draws$winner_name[i]
        player2 <- draws$loser_name[i]
        draw_surface <- draws$surface[i]

        pred <- predict(player1, player2, as.data.frame(coeff_frames[draw_surface]))
        draws$pred[i] <- pred[2]
      }

      acc <- score_predictions(draws)$accuracy
      if (acc > best_accuracy) {
        best_accuracy <- acc
        best_ijk <- c(i_,j_,k_)
        print(best_accuracy)
```

```
          print(best_ijk)
        }
      }
    }
  }
}
```

```
final_grass_clay <- 0.4
final_hard_clay <- 0.8
final_grass_hard <- 0.5
```

```
weighting <- as.data.frame(diag(3))
rownames(weighting) <- c("Grass", "Hard", "Clay")
colnames(weighting) <- c("Grass", "Hard", "Clay")
weighting['Grass', 'Hard'] <- final_grass_hard
weighting['Hard', 'Grass'] <- final_grass_hard
weighting['Clay', 'Hard'] <- final_hard_clay
weighting['Hard', 'Clay'] <- final_hard_clay
weighting['Clay', 'Grass'] <- final_grass_clay
weighting['Grass', 'Clay'] <- final_grass_clay

for (i in 1:nrow(train)) {
  train[i, 'grass_weight'] <- weighting[train[i, 'surface'], 'Grass']
  train[i, 'clay_weight'] <- weighting[train[i, 'surface'], 'Clay']
  train[i, 'hard_weight'] <- weighting[train[i, 'surface'], 'Hard']
}

btm_grass <- BTm(outcome = 1, factor(winner_name, levels=players), factor(loser_name, levels=players), w
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
btm_clay <- BTm(outcome = 1, factor(winner_name, levels=players), factor(loser_name, levels=players), we
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
btm_hard <- BTm(outcome = 1, factor(winner_name, levels=players), factor(loser_name, levels=players), we
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
# Create a dataframe with the Bradley-Terry estimates for each surface
btm_grass_df <- as.data.frame(BTabilities(btm_grass))
btm_clay_df <- as.data.frame(BTabilities(btm_clay))
btm_hard_df <- as.data.frame(BTabilities(btm_hard))

coeff_frames <- list('Grass' = btm_grass_df, 'Clay' = btm_clay_df,
                     'Hard' = btm_hard_df)

draws <- test[, c("winner_name", "loser_name", "surface")]
draws$pred <- NA

# predict the test set using a different model based on the surface
```

```r
for (i in 1:nrow(draws)) {
  player1 <- draws$winner_name[i]
  player2 <- draws$loser_name[i]
  draw_surface <- draws$surface[i]

  pred <- predict(player1, player2, as.data.frame(coeff_frames[draw_surface]))
  draws$pred[i] <- pred[2]
}

score_predictions(draws)
```

```
## $accuracy
## [1] 0.6403785
##
## $avg_probability
## [1] 0.6742218
##
## $avg_prob_se
## [1] 0.00352877
##
## $avg_log_probability
## [1] -0.4078452
##
## $abg_log_prob_se
## [1] 0.005172765
```

This model performs better.