ChessScriptAnalysisExample

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
library(stringr)
```

```
chess_games = read.csv('../Data/chess_games.csv')
head(chess_games)
```

```
##         id rated  created_at last_move_at turns victory_status winner
## 1 TZJHLljE FALSE 1.50421e+12  1.50421e+12    13      outoftime  white
## 2 l1NXvwaE  TRUE 1.50413e+12  1.50413e+12    16         resign  black
## 3 mIICvQHh  TRUE 1.50413e+12  1.50413e+12    61           mate  white
## 4 kWKvrqYL  TRUE 1.50411e+12  1.50411e+12    61           mate  white
## 5 9tXo1AUZ  TRUE 1.50403e+12  1.50403e+12    95           mate  white
## 6 MsoDV9wj FALSE 1.50424e+12  1.50424e+12     5           draw   draw
##   increment_code      white_id white_rating      black_id black_rating
## 1           15+2      bourgris         1500          a-00         1191
## 2           5+10          a-00         1322     skinnerua         1261
## 3           5+10        ischia         1496          a-00         1500
## 4           20+0 daniamurashov         1439  adivanov2009         1454
## 5           30+3     nik221107         1523  adivanov2009         1469
## 6           10+0     trelynn17         1250 franklin14532         1002
##
## 1
## 2
## 3
```

```
## 4
## 5 e4 e5 Nf3 d6 d4 Nc6 d5 Nb4 a3 Na6 Nc3 Be7 b4 Nf6 Bg5 O-O b5 Nc5 Bxf6 Bxf6 Bd3 Qd7 O-O Nxd3 Qxd3 c6
## 6
##   opening_eco                           opening_name opening_ply
## 1         D10       Slav Defense: Exchange Variation           5
## 2         B00 Nimzowitsch Defense: Kennedy Variation           4
## 3         C20  King's Pawn Game: Leonardis Variation           3
## 4         D02 Queen's Pawn Game: Zukertort Variation           3
## 5         C41                       Philidor Defense           5
## 6         B27   Sicilian Defense: Mongoose Variation           4
```

Get one

```
script1 = chess_games$moves[3]
```

```
script1_split = strsplit(script1,"[[:space:]]")[[1]]
script1_split
```

```
##  [1] "e4"   "e5"   "d3"   "d6"   "Be3"  "c6"   "Be2"  "b5"   "Nd2"  "a5"
## [11] "a4"   "c5"   "axb5" "Nc6"  "bxc6" "Ra6"  "Nc4"  "a4"   "c3"   "a3"
## [21] "Nxa3" "Rxa3" "Rxa3" "c4"   "dxc4" "d5"   "cxd5" "Qxd5" "exd5" "Be6"
## [31] "Ra8+" "Ke7"  "Bc5+" "Kf6"  "Bxf8" "Kg6"  "Bxg7" "Kxg7" "dxe6" "Kh6"
## [41] "exf7" "Nf6"  "Rxh8" "Nh5"  "Bxh5" "Kg5"  "Rxh7" "Kf5"  "Qf3+" "Ke6"
## [51] "Bg4+" "Kd6"  "Rh6+" "Kc5"  "Qe3+" "Kb5"  "c4+"  "Kb4"  "Qc3+" "Ka4"
## [61] "Bd1#"
```

```
script1_white_moves = script1_split[c(TRUE, FALSE)]
script1_black_moves = script1_split[c(FALSE, TRUE)]
```

Functions needed for such, test code examples

```
test_move = "Bxf8"
take = "x"

grepl(take, test_move, fixed=TRUE)
```

```
## [1] TRUE
```

```
strsplit(test_move, "[x]")[[1]][2]
```

```
## [1] "f8"
```

```
test_move2 = "Rf8"
strsplit(test_move2, "[x]")[[1]]
```

```
## [1] "Rf8"
```

```
grepl('f8', test_move, fixed = TRUE)
```

```
## [1] TRUE
```

```
strsplit(test_move2, "f8")[[1]][1]
```

```
## [1] "R"
```

```
str_detect("b", "[[:lower:]]")
```

```
## [1] TRUE
```

```
str_replace(test_move, 'x', '')
```

```
## [1] "Bf8"
```

Full moves for reference

```
script1
```

```
## [1] "e4 e5 d3 d6 Be3 c6 Be2 b5 Nd2 a5 a4 c5 axb5 Nc6 bxc6 Ra6 Nc4 a4 c3 a3 Nxa3 Rxa3 Rxa3 c4 dxc4 d5
```

```
script1_white_moves
```

```
##  [1] "e4"   "d3"   "Be3"  "Be2"  "Nd2"  "a4"   "axb5" "bxc6" "Nc4"  "c3"
## [11] "Nxa3" "Rxa3" "dxc4" "cxd5" "exd5" "Ra8+" "Bc5+" "Bxf8" "Bxg7" "dxe6"
## [21] "exf7" "Rxh8" "Bxh5" "Rxh7" "Qf3+" "Bg4+" "Rh6+" "Qe3+" "c4+"  "Qc3+"
## [31] "Bd1#"
```

```
script1_black_moves
```

```
##  [1] "e5"   "d6"   "c6"   "b5"   "a5"   "c5"   "Nc6"  "Ra6"  "a4"   "a3"
## [11] "Rxa3" "c4"   "d5"   "Qxd5" "Be6"  "Ke7"  "Kf6"  "Kg6"  "Kxg7" "Kh6"
## [21] "Nf6"  "Nh5"  "Kg5"  "Kf5"  "Ke6"  "Kd6"  "Kc5"  "Kb5"  "Kb4"  "Ka4"
```

helper function to translate string piece name to current value

```
name2value_func = function(piece_scriptname) {
  piece_value = 0

  if(piece_scriptname == "" || str_detect(piece_scriptname, "[[:lower:]]")) {
    piece_value = 1
  } else if (piece_scriptname == "N" || piece_scriptname == "B") {
    piece_value = 3
  } else if (piece_scriptname == "R") {
    piece_value = 5
  } else if (piece_scriptname == "Q") {
    piece_value = 9
  }

  return(piece_value)
}

name2value_func("a")
```

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

```
name2value_func("B")
```

```
## [1] 3
```

```
name2value_func("Q")
```

```
## [1] 9
```

```
name2value_func("")
```

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

helper function to translate string piece name to full name (to add to nicer dataframe)

```
scriptname2fullname_func = function(piece_scriptname) {
  if(piece_scriptname == "" || str_detect(piece_scriptname, "[[:lower:]]")) {
    fullname = "Pawn"
  } else if (piece_scriptname == "N") {
    fullname = "Knight"
  } else if (piece_scriptname == "B") {
    fullname = "Bishop"
  } else if (piece_scriptname == "R") {
    fullname = "Rook"
  } else if (piece_scriptname == "Q") {
    fullname = "Queen"
  } else if (piece_scriptname == "K") {
    fullname = "King"
  }

  return(fullname)
}

scriptname2fullname_func("a")
```

```
## [1] "Pawn"
```

```
scriptname2fullname_func("B")
```

```
## [1] "Bishop"
```

```
scriptname2fullname_func("Q")
```

```
## [1] "Queen"
```

```
scriptname2fullname_func("")
```

```
## [1] "Pawn"
```

Helper function to tell us where piece start (use for script analysis)

```
original_piece_locations = function(piece_color, piece_loc) {
  piece = "None"
  if(piece_color == 'white') {
    if (piece_loc %in% c('a2', 'b2', 'c2', 'd2', 'e2', 'f2', 'g2', 'h2')) {
      piece = ""
    } else if (piece_loc %in% c('a1', 'h1')) {
      piece = "R"
    } else if (piece_loc %in% c('b1', 'g1')) {
      piece = "N"
    } else if (piece_loc %in% c('c1', 'f1')) {
      piece = "B"
    } else if (piece_loc == 'd1') {
      piece = "Q"
    } else if (piece_loc == 'e1') {
      piece = "K"
    }}
  else {
    if (piece_loc %in% c('a7', 'b7', 'c7', 'd7', 'e7', 'f7', 'g7', 'h7')) {
      piece = ""
    } else if (piece_loc %in% c('a8', 'h8')) {
      piece = "R"
    } else if (piece_loc %in% c('b8', 'g8')) {
      piece = "N"
    } else if (piece_loc %in% c('c8', 'f8')) {
      piece = "B"
    } else if (piece_loc == 'd8') {
      piece = "Q"
    } else if (piece_loc == 'e8') {
      piece = "K"
    }}

  return(piece)
}


original_piece_locations("white", "e2")
```

```
## [1] ""
```

```
original_piece_locations("white", "f1")
```

```
## [1] "B"
```

```
original_piece_locations("black", "g7")
```

```
## [1] ""
```

```
original_piece_locations("black", "a4")
```

```
## [1] "None"
```

```
#initialize dataframe of takes
game_takes = as.data.frame(matrix(nrow = 6, ncol = 4))
game_takes[is.na(game_takes)] = 0
rownames(game_takes) = c('Pawn', 'Knight', 'Bishop', 'Rook', 'Queen', 'King')
colnames(game_takes) = c('num_moves', 'num_checks', 'num_pieces_taken', 'value_pieces_taken')
game_takes$list_pieces_taken = ""


#Fill in column for number of moves

for (i in c(1:length(script1_white_moves))) {
  move = script1_white_moves[i]
  first_char = substr(move, 1, 1)
  piece = scriptname2fullname_func(first_char)


  game_takes[piece, 'num_moves'] =
    game_takes[piece, 'num_moves'] + 1
  }

#Fill in column for number of checks

for (i in c(1:length(script1_white_moves))) {
  move = script1_white_moves[i]

  if((grepl("+", move, fixed = TRUE)) | (grepl("#", move, fixed = TRUE))) {
    first_char = substr(move, 1, 1)
    piece = scriptname2fullname_func(first_char)


    game_takes[piece, 'num_checks'] =
      game_takes[piece, 'num_checks'] + 1
  }
}

#Fill in columns num_pieces_taken, value_pieces_taken, and list_pieces_taken

for (i in c(1:length(script1_white_moves))) {
  white_move = script1_white_moves[i]
  #Step 1: Find the moves where a piece was taken (all the moves that have an x in the term)
  if(grepl("x", white_move, fixed=TRUE)) {


    #Step 2: Extract the move number of the take move and the piece that took \
    ###Example: axb5 leads to the piece_capturer = a & board_loc = b5
    move_num <<- i
    piece_capturer <<- strsplit(white_move, "[x]")[[1]][1]
    board_loc <<- strsplit(white_move, "[x]")[[1]][2]


    #Step 3: Identify which piece it took by iterating backwards through black pieces,
    ##if piece_taken is "", its a pawn
```

```
    #Intialize, used for later in case taken piece hasn't moved
    piece_taken <<- "RESET"

    for (j in c((move_num-1):1)) {
      black_move <- script1_black_moves[j]
      piece_select = strsplit(black_move, board_loc)[[1]][1]


      if (grepl(board_loc, black_move, fixed = TRUE)) {
        #take out excess x (which indicated a previous take)
        piece_taken <<- str_replace(piece_select, 'x', '')
        #if successful piece found, break loop
        break
      }
    }


    ###Edge case: piece that was taken has not moved yet
    if (piece_taken == "RESET") {
      piece_taken <<- original_piece_locations("black", board_loc)
    }

  #step 4: Add values to dataframe
  if(i == move_num) {
    piece_capturer_name = scriptname2fullname_func(piece_capturer)
    piece_taken_name = scriptname2fullname_func(piece_taken)
    piece_taken_value = name2value_func(piece_taken)

    game_takes[piece_capturer_name, 'num_pieces_taken'] =
      game_takes[piece_capturer_name, 'num_pieces_taken'] + 1

    game_takes[piece_capturer_name, 'value_pieces_taken'] =
      game_takes[piece_capturer_name, 'value_pieces_taken'] + piece_taken_value

    string = game_takes[piece_capturer_name, 'list_pieces_taken']
    new_string = paste(string, piece_taken_name)
    game_takes[piece_capturer_name, 'list_pieces_taken'] = new_string

    }
  }
}

game_takes
```

```
##        num_moves num_checks num_pieces_taken value_pieces_taken
## Pawn          12          1                7                19
## Knight         3          0                1                 1
## Bishop         8          3                3                 7
## Rook           5          2                3                11
## Queen          3          3                0                 0
## King           0          0                0                 0
##                              list_pieces_taken
## Pawn    Pawn Knight Pawn Pawn Queen Bishop Pawn
```

```
## Knight                                  Pawn
## Bishop                        Bishop Pawn Knight
## Rook                              Rook Rook Pawn
## Queen
## King
```