White Advantage in Chess and How to Counter It

Jun Won (Lakon) Park (Group Leader) *79453940, Group Leader, data collection, data analysis, report-writing*
Sarah Li *60136959, data collection, data analysis, report-writing*

Research question: Is white at an advantage in chess and if so, what are some optimal strategies for black to increase their winning probability?

Introduction

For several centuries, millions of people worldwide have been playing chess as a recreational and competitive board game at their homes, in clubs, in tournaments, and even online nowadays. In the recent decades, chess has been one of the most popular topic in machine learning and artificial intelligence. The first move advantage has been researched extensively since the end of 18th century, and many studies have been shown that white has an inherent advantage.

Although there are general set chess openings for black according to white's first move, less research has been done on the effects of those openings on the final outcome. This paper intends to confirm white's first move advantage and study the relationship between the openings and the victory status.

This paper's data collection consists basic player information and game information of over 20000 chess games obtained exclusively from Lichess, a very popular internet chess platform. The data includes game length, number of turns, winner, player elo*, all moves in Standard Chess Notation, Opening Eco*, Opening Name, and Opening Ply* (some stuff about sampling method and target population)

Elo : A numerical measurement to quantify a player's skill level
Eco : Standardised code for any given opening
Ply : Number of moves in the opening phasenewline

Analysis

```r
# Load data
df <- read.csv("games.csv")

# Calculate the average elo of the game
df <- mutate(df %>% rowwise(),
       average_elo = rowMeans(cbind(black_rating,white_rating)))

# Filter games by average elo
df <- filter(df, average_elo >= 1200)

# Filter games by average elo
df <- filter(df, victory_status != "outoftime")

df <- subset(df,
             select = c(turns, white_rating, black_rating, victory_status,
                        winner, moves, opening_eco, opening_name, opening_ply, average_elo ))

# Simple Random Sampling
N <- nrow(df)
n <- 2000
set.seed(1234)
sample.index <- sample(1:N, size=n, replace = FALSE)
srs.sample <- df[sample.index,]
```

```r
# Determine minimum and maximum before stratifying
min(df$average_elo)
```

```
## [1] 1200
```

```r
max(df$average_elo)
```

```
## [1] 2475.5
```

```r
# Stratified sampling
df$elo_range <- cut(df$average_elo,
                    c(1200, 1400, 1600, 1800, 2000, 2200, 2400, 2600))
levels(df$elo_range) <- c("1200-1400", "1400-1600", "1600-1800", "1800-2000",
                          "2000-2200", "2200-2400", "2400+")
df$winner <- as.factor(df$winner)
# levels(df$winner) <- list("white"=c("white"),
```

```r
#                              "black"=c("black")
#                              "not_white"=c("black", "draw"))

# Check if standard deviations of the strata are identical
se.by.strata <- aggregate(as.numeric(df$winner), by=list(df$elo_range), FUN=sd)
se.by.strata
```

```
##     Group.1         x
## 1 1200-1400 0.9783955
## 2 1400-1600 0.9777180
## 3 1600-1800 0.9737201
## 4 1800-2000 0.9697871
## 5 2000-2200 0.9618646
## 6 2200-2400 0.9323589
## 7     2400+ 0.8232726
```

```r
# Standard deviations within strata are not identical, \
# so find optimal sample sizes
pop_sizes <- aggregate(df$winner, by=list(df$elo_range), FUN=length)
denom <- sum(pop_sizes[2] * se.by.strata[2])
sample_sizes <- (pop_sizes[2] * se.by.strata[2]) / denom


# Sample from each strata
stratified_sample <- df[FALSE,]
colnames(stratified_sample) <- names(df)
for (i in 1:length(levels(df$elo_range))) {
  strata <- which(df$elo_range == levels(df$elo_range)[i])
  sample_indices <- sample(strata,
                           size = ceiling(sample_sizes$x[i] * n),
                           replace = FALSE)
  sample <- df[sample_indices,]
  stratified_sample <- rbind(stratified_sample, sample)
}
table <- table(stratified_sample$elo_range)
table
```

```
##
## 1200-1400 1400-1600 1600-1800 1800-2000 2000-2200 2200-2400     2400+
##       398       652       489       310       126        27         1
```

```r
# Stratified sample contains 1003 samples due to rounding of the proportions,
# so we randomly remove three from random strata
strata_for_removal <- sample(1:7, 3)
for (s in strata_for_removal) {
  to_remove <- sample(which(stratified_sample$elo_range == levels(df$elo_range)[s]), 1)
  stratified_sample <- stratified_sample[-to_remove,]
```

```
}
table(stratified_sample$elo_range)
```

```
##
## 1200-1400 1400-1600 1600-1800 1800-2000 2000-2200 2200-2400     2400+
##       397       651       489       310       125        27         1
```

```
z.95 <- qnorm(0.975)
# Calculate white's win rate
win.prop <- srs.sample %>%
  count(winner) %>%
  group_by(winner) %>%
  mutate(win.prop = n / N)

white.p <- as.numeric(win.prop[3,3])
black.p <- as.numeric(win.prop[1,3])

srs.se <- sqrt((1-n/N)*(white.p * (1-white.p) + black.p * (1-black.p) - 2*white.p*black.p)/n)

(white.p-black.p) + z.95 * srs.se * c(-1, 1)
```

```
## [1] -0.01005471  0.01571161
```

```
strata <- c("1200-1400", "1400-1600", "1600-1800", "1800-2000",
                          "2000-2200", "2200-2400", "2400+")
stratified_sample <- stratified_sample %>% group_by(elo_range)

# Calculate Nh/N, the strata proportion
nh <- stratified_sample %>% count(elo_range)
strata.size.prop <- nh[2]/n

# Calculate white's win proportion by each strata
win.prop <- stratified_sample %>%
  count(winner) %>%
  group_by(elo_range) %>%
  mutate(win.prop = n / sum(n))

# The estimated aggregated win proportion for white
white.prop <- win.prop[win.prop$winner == "white", ]
white.p.str.est <- sum(white.prop$win.prop * strata.size.prop)

black.prop <- win.prop[win.prop$winner == "black", ]
black.p.str.est <- sum(black.prop$win.prop * strata.size.prop)

# The estimated se
white.se.by.strata <- sqrt((1-nh[2]/N) * white.prop$win.prop * (1-white.prop$win.prop)/nh[2])
```

```r
white.str.se <- sum(strata.size.prop^2 * (1 - nh[2]/pop_sizes[2]) * white.se.by.strata^2)

black.se.by.strata <- sqrt((1-nh[2]/N) * black.prop$win.prop * (1-black.prop$win.prop)/nh[2])
black.str.se <- sum(strata.size.prop^2 * (1 - nh[2]/pop_sizes[2]) * black.se.by.strata^2)

# Their difference,
diff.se <- sqrt(1-n/N) *sqrt(white.p.str.est*(1-white.p.str.est) + black.p.str.est*(1-black.p.s
(white.p.str.est - black.p.str.est) + z.95 * diff.se * c(-1, 1)
```

```
## [1] 0.04515198 0.06237950
```

```r
# The confidence interval does not contain 0 so we can reject the null hypothesis in favour of
```

```r
# Due to the many possible openings a game can start with,
# the sample size in each possible domain (opening_name)
# may be very small. In order to ensure that the confidence
# interval is of reasonable width, we will only estimate if
# sample size in the domain yields a confidence interval including
# +-0.2 of our estimate.
openings.df.s <- data.frame(table(srs.sample$opening_name))
names(openings.df.s) <- c("name", "frequency")

var.guess <- 0.25
ci.width <- 0.2
n0 <- z.95 ** 2 * var.guess / ci.width ** 2
openings.freq <- openings.df.s[openings.df.s$frequency > 15,]

openings.df.p <- data.frame(table(df$opening_name))
names(openings.df.p) <- c("name", "frequency")
openings.size.p <- openings.df.p[openings.df.p$name %in% openings.freq$name,]

domain.sizes <- c()
for (name in openings.freq$name) {
  domain.sizes <- append(domain.sizes, n0 / (1 + n0 / openings.df.p[openings.df.p$name == name
}
openings.valid <- openings.freq[openings.freq$frequency > domain.sizes,]

estimates <- rep(0, nrow(openings.valid))
intervals <- matrix(0, nrow(openings.valid), 2)
for (i in 1:nrow(openings.valid)) {
  domain.name <- openings.valid[i, 1]
  domain.s <- srs.sample[srs.sample$opening_name == domain.name,]
  n.d <- openings.valid[i, 2]
  domain.p <- df[df$opening_name == domain.name,]
  N.d <- nrow(domain.p)

  white.win.count <- nrow(domain.s[domain.s$winner == "white",])
```

```
  black.win.count <- nrow(domain.s[domain.s$winner == "black",])
  white.p <- white.win.count / n.d
  black.p <- black.win.count / n.d
  estimates[i] <- white.p - black.p

  white.se <- sqrt((1 - n.d / N.d) * white.p * (1 - white.p) / n.d)
  black.se <- sqrt((1 - n.d / N.d) * black.p * (1 - black.p) / n.d)
  # TODO: make functions
  diff.se <- sqrt(1-n.d/N.d) *sqrt(white.p*(1-white.p) + black.p*(1-black.p) - 2*white.p*black

  intervals[i,] <- (white.p - black.p) + z.95 * diff.se * c(-1, 1)
}

openings <- data.frame(openings.valid$name, intervals)
colnames(openings) <- c("name", "95.CI.lower", "95.CI.upper")
white.higher <- openings[openings$`95.CI.lower` > 0,]
white.lower <- openings[openings$`95.CI.upper` < 0,]
white.higher
```

```
##                                name 95.CI.lower 95.CI.upper
## 1 French Defense: Knight Variation  0.29032258  0.29032258
## 2                   Horwitz Defense  0.09090909  0.09090909
## 5               Philidor Defense #3  0.57531073  0.69135594
## 8                       Scotch Game  0.05340792  0.21931936
```

```
white.lower
```

```
##                                              name 95.CI.lower 95.CI.upper
## 3                                     Indian Game -0.37056883 -0.20085974
## 4                             Philidor Defense #2 -0.56521739 -0.56521739
## 7  Scandinavian Defense: Mieses-Kotroc Variation -0.04347826 -0.04347826
## 9                                Sicilian Defense -0.25925926 -0.25925926
## 10               Sicilian Defense: Bowdler Attack -0.50087285 -0.31162715
## 11                            Van't Kruijs Opening -0.30009904 -0.19990096
```

```
estimates <- rep(0, nrow(openings.valid))
intervals <- matrix(0, nrow(openings.valid), 2)
for (i in 1:nrow(openings.valid)) {
  domain.name <- openings.valid[i, 1]
  domain.s <- stratified_sample[stratified_sample$opening_name == domain.name,]
  n.d <- openings.valid[i, 2]
  domain.p <- df[df$opening_name == domain.name,]
  N.d <- nrow(domain.p)

  nh.d <- domain.s %>% count(elo_range, .drop=FALSE)
  Nh.d <- domain.p %>% count(elo_range, .drop=FALSE)
  print(nh.d)
```

```
  strata.size.prop <- Nh.d[2]/N.d

  # Calculate white's win proportion by each strata
  win.prop <- domain.s %>%
    count(winner) %>%
    group_by(elo_range, .drop=FALSE) %>%
    mutate(win.prop = n / sum(n))
  white.p <- win.prop[win.prop$winner == "white", ]
  white.p.str.est <- sum(white.prop$win.prop * strata.size.prop)

  black.p <- win.prop[win.prop$winner == "black", ]
  black.p.str.est <- sum(black.prop$win.prop * strata.size.prop)

  white.se.by.strata <- sqrt((1-n.d/N.d) * white.p$win.prop * (1-white.p$win.prop)/nh.d[2])
  white.str.se <- sum(strata.size.prop^2 * (1 - nh.d[2]/Nh.d[2]) * white.se.by.strata^2)

  black.se.by.strata <- sqrt((1-n.d/N.d) * black.p$win.prop * (1-black.p$win.prop)/nh.d[2])
  black.str.se <- sum(strata.size.prop^2 * (1 - nh.d[2]/Nh.d[2]) * black.se.by.strata^2)

  diff.se <- sqrt(1-n.d/N.d) *sqrt(white.p.str.est*(1-white.p.str.est) + black.p.str.est*(1-bla
  (white.p.str.est - black.p.str.est) + z.95 * diff.se * c(-1, 1)

  estimates[i] <- white.p.str.est - black.p.str.est
  intervals[i,] <- (white.p.str.est - black.p.str.est) + z.95 * diff.se * c(-1, 1)
}
```

```
## # A tibble: 7 x 2
## # Groups:   elo_range [7]
##   elo_range     n
##   <fct>     <int>
## 1 1200-1400     7
## 2 1400-1600     9
## 3 1600-1800    12
## 4 1800-2000     4
## 5 2000-2200     1
## 6 2200-2400     0
## 7 2400+         0
## # A tibble: 7 x 2
## # Groups:   elo_range [7]
##   elo_range     n
##   <fct>     <int>
## 1 1200-1400     4
## 2 1400-1600    11
## 3 1600-1800     4
## 4 1800-2000     4
## 5 2000-2200     0
## 6 2200-2400     1
```

```
## 7 2400+          0
## # A tibble: 7 x 2
## # Groups:   elo_range [7]
##   elo_range     n
##   <fct>     <int>
## 1 1200-1400     0
## 2 1400-1600     7
## 3 1600-1800     6
## 4 1800-2000     7
## 5 2000-2200     4
## 6 2200-2400     1
## 7 2400+         0
## # A tibble: 7 x 2
## # Groups:   elo_range [7]
##   elo_range     n
##   <fct>     <int>
## 1 1200-1400     4
## 2 1400-1600     6
## 3 1600-1800     6
## 4 1800-2000     0
## 5 2000-2200     0
## 6 2200-2400     0
## 7 2400+         0
## # A tibble: 7 x 2
## # Groups:   elo_range [7]
##   elo_range     n
##   <fct>     <int>
## 1 1200-1400     6
## 2 1400-1600     4
## 3 1600-1800     3
## 4 1800-2000     1
## 5 2000-2200     1
## 6 2200-2400     0
## 7 2400+         0
## # A tibble: 7 x 2
## # Groups:   elo_range [7]
##   elo_range     n
##   <fct>     <int>
## 1 1200-1400     2
## 2 1400-1600    14
## 3 1600-1800     5
## 4 1800-2000     8
## 5 2000-2200     1
## 6 2200-2400     2
## 7 2400+         0
## # A tibble: 7 x 2
## # Groups:   elo_range [7]
##   elo_range     n
```

```
##   <fct>     <int>
## 1 1200-1400     9
## 2 1400-1600     8
## 3 1600-1800     6
## 4 1800-2000     4
## 5 2000-2200     1
## 6 2200-2400     0
## 7 2400+         0
## # A tibble: 7 x 2
## # Groups:   elo_range [7]
##   elo_range     n
##   <fct>     <int>
## 1 1200-1400     5
## 2 1400-1600    10
## 3 1600-1800    11
## 4 1800-2000     2
## 5 2000-2200     0
## 6 2200-2400     0
## 7 2400+         0
## # A tibble: 7 x 2
## # Groups:   elo_range [7]
##   elo_range     n
##   <fct>     <int>
## 1 1200-1400     8
## 2 1400-1600    19
## 3 1600-1800    13
## 4 1800-2000     6
## 5 2000-2200     3
## 6 2200-2400     0
## 7 2400+         0
## # A tibble: 7 x 2
## # Groups:   elo_range [7]
##   elo_range     n
##   <fct>     <int>
## 1 1200-1400     4
## 2 1400-1600    15
## 3 1600-1800     6
## 4 1800-2000     1
## 5 2000-2200     1
## 6 2200-2400     0
## 7 2400+         0
## # A tibble: 7 x 2
## # Groups:   elo_range [7]
##   elo_range     n
##   <fct>     <int>
## 1 1200-1400    14
## 2 1400-1600    12
## 3 1600-1800     2
```

```
## 4 1800-2000     1
## 5 2000-2200     0
## 6 2200-2400     0
## 7 2400+         0
```

```
openings <- data.frame(openings.valid$name, intervals)
colnames(openings) <- c("name", "95.CI.lower", "95.CI.upper")
white.higher <- openings[openings$`95.CI.lower` > 0,]
white.lower <- openings[openings$`95.CI.upper` < 0,]
white.higher
```

```
## [1] name        95.CI.lower 95.CI.upper
## <0 rows> (or 0-length row.names)
```

```
white.lower
```

```
## [1] name        95.CI.lower 95.CI.upper
## <0 rows> (or 0-length row.names)
```

```
# mean number of turns for white wins vs black wins?
white.win <- srs.sample[srs.sample$winner == "white",]
black.win <- srs.sample[srs.sample$winner == "black",]

white.win.turns.avg <- mean(white.win$turns)
black.win.turns.avg <- mean(black.win$turns)
```

Conclusion

References