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, 2600))
levels(df$elo_range) <- c("1200-1400", "1400-1600", "1600-1800", "1800-2000",
                          "2000-2200", "2200+")
df$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+ 0.9293261
```

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


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

```
##
## 1200-1400 1400-1600 1600-1800 1800-2000 2000-2200     2200+
##       398       652       489       310       126        28
```

```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(sample.str$elo_range == levels(df$elo_range)[s]), 1)
  sample.str <- sample.str[-to.remove,]
```

```r
}
table(sample.str$elo_range)
```

```
##
## 1200-1400 1400-1600 1600-1800 1800-2000 2000-2200     2200+
##       397       651       489       310       125        28
```

```r
z.95 <- qnorm(0.975)
# Returns the sample variance of a given proportion
var.est <- function(p) {
  p * (1 - p)
}

# 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)*(var.est(white.p) + var.est(black.p) + 2*white.p*black.p)/n)

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

```
## [1] -0.01082512  0.01648202
```

```r
strata <- c("1200-1400", "1400-1600", "1600-1800", "1800-2000",
                           "2000-2200", "2200+")
sample.str <- sample.str %>% group_by(elo_range)

# Calculate Nh/N, the strata proportion
Nh <- df %>% count(elo_range, .drop=FALSE)
Nh <- Nh[complete.cases(Nh),]
nh <- sample.str %>% count(elo_range, .drop=FALSE)
strata.size.prop <- Nh[2] / N

# Calculate white's win proportion by each strata
win.prop <- sample.str %>%
  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", ]
```

```r
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
# TODO: remove if unused
# white.se.by.strata <- sqrt((1-nh[2]/Nh[2]) * var.est(white.prop$win.prop)/nh[2])
# white.str.se <- sum(strata.size.prop^2 * (1 - nh[2]/Nh[2]) * white.se.by.strata^2)
#
# black.se.by.strata <- sqrt((1-nh[2]/Nh[2]) * var.est(black.prop$win.prop)/nh[2])
# black.str.se <- sum(strata.size.prop^2 * (1 - nh[2]/Nh[2]) * black.se.by.strata^2)

# Their difference,
diff.se <- sqrt(1-n/N) *sqrt(var.est(white.p.str.est) + var.est(black.p.str.est) + 2*white.p.st
(white.p.str.est - black.p.str.est) + z.95 * diff.se * c(-1, 1)
```

```
## [1] 0.0131518 0.0936011
```

```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 (split by 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 of win rate.
openings.df.s <- data.frame(table(srs.sample$opening_name))
names(openings.df.s) <- c("name", "frequency")

# Guess the most conservative variance
# Find minimum domain sample size for desired CI width
var.guess <- 0.25
ci.width <- 0.2
n0 <- z.95 ** 2 * var.guess / ci.width ** 2

# Include openings with sample size large enough for usable CI
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,]

# Include openings with sample sizes yielding the desired CI width
domain.sizes <- c()
for (name in openings.freq$name) {
```

```r
  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)) {
  # Find estimate and CI for difference in win rate for white/black
  # for one opening
  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

  diff.se <- sqrt(1-n.d/N.d) *sqrt(var.est(white.p) + var.est(black.p) + 2*white.p*black.p)/sq
  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,]
# Report openings with only positive and only negative 95 CIs
white.higher
```

```
##                  name 95.CI.lower 95.CI.upper
## 5 Philidor Defense #3   0.3902489   0.8764177
```

```r
# The intervals which only include negative values suggest that there is
# a higher black win rate. For these openings, we can reject the null hypothesis
# in favour of the alternative which states that there is a difference in win rates.
# More specifically, there is evidence that black has a higher win rate when
# these openings are used.
white.lower
```

```
##                                name 95.CI.lower 95.CI.upper
## 4                Philidor Defense #2  -0.8771838  -0.2532509
## 10 Sicilian Defense: Bowdler Attack  -0.6857342  -0.1267658
```

```r
estimates <- rep(0, nrow(openings.valid))
intervals <- matrix(0, nrow(openings.valid), 2)
for (i in 1:nrow(openings.valid)) {
  # Find estimate and CI for difference in win rate for white/black
  # for one domain
  domain.name <- openings.valid[i, 1]
  domain.s <- sample.str[sample.str$opening_name == domain.name,]
  domain.p <- df[df$opening_name == domain.name,]

  n.d <- openings.valid[i, 2]
  N.d <- nrow(domain.p)
  nh.d <- domain.s %>% count(elo_range, .drop=FALSE)
  Nh.d <- domain.p %>% count(elo_range, .drop=FALSE)
  strata.size.prop <- Nh.d[2]/N.d

  # Calculate white's win proportion by each strata within a single domain
  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)

  # TODO: remove if unused
  # white.se.by.strata <- sqrt((1-n.d/N.d) * var.est(white.p$win.prop)/nh.d[2])
  # white.str.se <- sum(strata.size.prop^2 * (1 - nh.d[2]/Nh.d[2]) * white.se.by.strata^2)
  #
  # black.se.by.strata <- sqrt((1-n.d/N.d) * var.est(black.p$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(var.est(white.p.str.est) + var.est(black.p.str.est) + 2*whi
  (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)
}

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,]
# All of the openings include 0, so there does not seem to be a difference
# in win rate between white and black for each opening. We cannot reject
# the null hypothesis that win rate is the same for a given opening.
```

```r
openings
```

```
##                                          name 95.CI.lower 95.CI.upper
## 1              French Defense: Knight Variation  -0.2614431   0.3817744
## 2                               Horwitz Defense  -0.3235549   0.4354852
## 3                                   Indian Game  -0.3372053   0.4381305
## 4                           Philidor Defense #2  -0.3106491   0.4304839
## 5                           Philidor Defense #3  -0.2575136   0.3756607
## 6             Queen's Pawn Game: Mason Attack  -0.2607863   0.3724620
## 7  Scandinavian Defense: Mieses-Kotroc Variation  -0.3249116   0.4336238
## 8                                   Scotch Game  -0.3268557   0.4531017
## 9                              Sicilian Defense  -0.2914832   0.4124210
## 10             Sicilian Defense: Bowdler Attack  -0.2540011   0.3807743
## 11                          Van't Kruijs Opening  -0.2468374   0.3514205
```

```r
# mean number of turns for white wins vs black wins?
# maybe auxiliary variable could be average_elo
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)

srs.se <- sqrt((1-n/N)*var(srs.sample$turns)/n)

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

```
## [1] -3.985820 -1.318596
```

```r
white.win <- sample.str[sample.str$winner == "white",]
black.win <- sample.str[sample.str$winner == "black",]

# Calculate average number of turns for white win by each strata
avg.turns.w <- white.win %>%
  group_by(elo_range) %>%
  summarise(mean_turns = mean(turns))

# and for black win
avg.turns.b <- black.win %>%
  group_by(elo_range) %>%
  summarise(mean_turns = mean(turns))

# The estimated average number of turns
white.avg <- sum(Nh[2]/N * avg.turns.w$mean_turns)
black.avg <- sum(Nh[2]/N * avg.turns.b$mean_turns)

# The estimated se
```

```r
white.se.by.strata <- sqrt((1-nh[2]/Nh[2]) * var(white.win$turns)/nh[2])
white.str.var <- sum(strata.size.prop^2 * (1 - nh[2]/Nh[2]) * white.se.by.strata^2)

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

# TODO: Not sure how to find the covariance term
# cov.bw <- cov(white.win$turns, black.win$turns)
#
# # Their difference,
# diff.se <- sqrt(1-n/N) * sqrt(white.str.var + black.str.var + 2*white.p.str.est*black.p.str.
# (white.p.str.est - black.p.str.est) + z.95 * diff.se * c(-1, 1)
```

Conclusion

References