Assignment Analysis 1

2023-02-01

Required Packages

```r
library(stringr)
library(skimr)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   1.0.1
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v forcats 0.5.2
## v readr   2.1.3
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Question 1

```r
chess = readLines("chess_classic_games.txt")    #Read raw data
#His <- read.csv("Full_Data_Classic_Q1.csv") #Used this to check my solution
                                             #with the correct one
chess_2 = str_split_fixed(chess, " ", 2)

is_metadata = str_detect(chess, "^\\[.*\\]$") # Separating meta data from moves line

pgn_meta = chess

pgn_meta[!is_metadata] = ""            #Replacing moves line with an empty line

pgn_meta2 = str_split_fixed(pgn_meta, " ", 2)
pgn_meta3= pgn_meta2[!apply(pgn_meta2 == "", 1, all),]  #Removing empty rows

vars = str_replace(pgn_meta3[,1], "\\[", "")   #Erasing [ from metadata
varnames = unique(vars)                        #Finding unique variable names
varnames = varnames[!varnames == '']           #Removing empty variable names
values = pgn_meta3[,2]

values = str_replace_all(values, "\"", "")      #Erasing [ from metadata values

values = str_replace(values, "\\]$", "")        #Erasing ] from metadata values

New_event = str_detect(pgn_meta3[,1], "Event")  #Number of line where there's a
                                                #new Event in cleaned data
New_event_index = which (New_event)
New_event_index = c(New_event_index,nrow(pgn_meta3)) #including last row index
N_games = length(New_event_index)-1    #Number of games played. -1 because of
```

```r
                                        #the previous code line

MyData <- vector( "character" , 19 )  #Initializing a vector to keep generated rows


for (i in 1:N_games){

  if (i==N_games){ #This if condition is because we have to extract last line data differently
    a=New_event_index[i]
    b=New_event_index[i+1]

  } else {

    a=New_event_index[i]
    b=New_event_index[i+1]-1

  }


  vars2 <- vars[a:b]  #Slicing variable names for the particular row [i]
  reorder_index <- match(varnames,vars2) #matching variable names[i] with unique varnames
  vars2_reordered <-vars2[reorder_index] #Ordering variable names like varnames
  values_reordered <-values[a:b][reorder_index] #ordering values to correspond to varnames


  values_reordered <- na.omit(values_reordered)  #omitting NA's so they don't mess ordered values
  ind <- varnames %in% vars2_reordered   #Used to know where each value should go
                                        #if values are not ordered as others in metadata

  New_data <- vector( "character" , 19 ) #initializing data row

  New_data[ind] <- values_reordered
  MyData <- rbind (MyData, New_data) #Adding new data below the other data

}

MyData2= MyData[!apply(MyData == "", 1, all),]   #Erasing the first row because it was blank

My_Df <- data.frame(MyData2)                   #Changing vector to data frame

colnames(My_Df)<-varnames    #Column names should be varnames(unique variable names)
rownames(My_Df)<-c()



cols.num <- c("WhiteElo","BlackElo","WhiteRatingDiff","BlackRatingDiff")
My_Df[cols.num] <- sapply(My_Df[cols.num],as.numeric)





print(My_Df[6:10,])
```

```
##                Event                         Site       Date Round        White
## 6  Rated Blitz game https://lichess.org/2TrUvPFl 2018.09.30     -  Revealchess
## 7  Rated Blitz game https://lichess.org/1PVmIgZn 2018.09.30     -    Redbull22
## 8  Rated Blitz game https://lichess.org/0zRqQX25 2018.09.30     -        zoom9
## 9  Rated Blitz game https://lichess.org/0yMXu34o 2018.09.30     -         wuju
## 10 Rated Blitz game https://lichess.org/lk3dzqXV 2018.09.30     - MeisterLuetz
##        Black Result    UTCDate  UTCTime WhiteElo BlackElo WhiteRatingDiff
## 6   oldclown    1-0 2018.09.30 22:00:08     1616     1612              10
## 7  chessfort    0-1 2018.09.30 22:00:08     2251     2266             -10
## 8     jteles    0-1 2018.09.30 22:00:08     1584     1589             -10
## 9     zifmia    0-1 2018.09.30 22:00:08     1819     1837             -10
## 10 jedissson    0-1 2018.09.30 22:00:08     2062     1958             -13
##    BlackRatingDiff ECO
## 6              -10 A43
## 7               11 B01
## 8               10 A16
## 9                9 D00
## 10              14 A01
##                                                            Opening TimeControl
## 6                                           Old Benoni Defense #2       180+0
## 7                             Scandinavian Defense: Modern Variation #2       180+0
## 8  English Opening: Anglo-Indian Defense, Queen's Knight Variation       180+0
## 9                                Queen's Pawn Game: Levitsky Attack       180+0
## 10                            Nimzo-Larsen Attack: Symmetrical Variation       180+0
##     Termination WhiteTitle BlackTitle
## 6  Time forfeit
## 7        Normal
## 8        Normal
## 9  Time forfeit
## 10 Time forfeit
```

```
skim(My_Df)
```

Table 1: Data summary

Name	My_Df
Number of rows	4994
Number of columns	19
Column type frequency:	
character	15
numeric	4
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Event	0	1	16	63	0	18	0
Site	0	1	28	28	0	4994	0
Date	0	1	10	10	0	1	0
Round	0	1	1	1	0	1	0
White	0	1	2	20	0	4116	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Black	0	1	3	20	0	4098	0
Result	0	1	1	7	0	4	0
UTCDate	0	1	10	10	0	1	0
UTCTime	0	1	8	8	0	510	0
ECO	0	1	1	3	0	284	0
Opening	0	1	1	87	0	947	0
TimeControl	0	1	1	8	0	100	0
Termination	0	1	6	12	0	4	0
WhiteTitle	0	1	0	2	4969	4	0
BlackTitle	0	1	0	2	4971	5	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
WhiteElo	0	1	1582.60	312.04	800	1357.25	1584	1801.00	2544	
BlackElo	0	1	1580.50	314.77	800	1357.00	1582	1797.75	2527	
WhiteRatingDiff	1	1	0.30	17.63	-306	-10.00	2	10.00	338	
BlackRatingDiff	1	1	-0.34	17.94	-221	-10.00	-2	10.00	463	

Question 2

```
NEvent = str_detect(chess_2[,1], "Event")      #Number of line where there's a new Event
tag_line = which (NEvent)                      # Finding the index of

empty_lines <- grep("^$", chess)               # Finding the index of empty lines in raw data
differences <- diff(empty_lines)
moves_line_ind <- which(differences == 2)#If distance is two, means we have a moves line between them
moves_line2 <- empty_lines[moves_line_ind]+1 # Moves line is one line after the first empty line

#Nmoves = str_detect(chess_2[,1], "1.")
moves_line = c(moves_line2,nrow(chess_2))      #Adding the last row index for last event's moves line

df2 <- data.frame(moves_line = moves_line,tag_line = tag_line) #Adding columns to the data frame
My_Df2 <- cbind(My_Df,df2)

cols.num <- c("moves_line","tag_line")
My_Df2[cols.num] <- sapply(My_Df2[cols.num],as.numeric)       #Making them numeric values

print(My_Df2[6:10,])
```

```
##               Event                           Site       Date Round        White
## 6  Rated Blitz game https://lichess.org/2TrUvPFl 2018.09.30     -  Revealchess
## 7  Rated Blitz game https://lichess.org/1PVmIgZn 2018.09.30     -    Redbull22
## 8  Rated Blitz game https://lichess.org/0zRqQX25 2018.09.30     -        zoom9
## 9  Rated Blitz game https://lichess.org/0yMXu34o 2018.09.30     -         wuju
## 10 Rated Blitz game https://lichess.org/lk3dzqXV 2018.09.30     - MeisterLuetz
##         Black Result    UTCDate  UTCTime WhiteElo BlackElo WhiteRatingDiff
## 6    oldclown    1-0 2018.09.30 22:00:08     1616     1612              10
## 7  chessfort    0-1 2018.09.30 22:00:08     2251     2266             -10
## 8      jteles    0-1 2018.09.30 22:00:08     1584     1589             -10
## 9      zifmia    0-1 2018.09.30 22:00:08     1819     1837             -10
## 10 jedissson    0-1 2018.09.30 22:00:08     2062     1958             -13
##    BlackRatingDiff ECO
## 6              -10 A43
## 7               11 B01
## 8               10 A16
## 9                9 D00
## 10              14 A01
##                                                             Opening TimeControl
## 6                                             Old Benoni Defense #2       180+0
## 7                           Scandinavian Defense: Modern Variation #2       180+0
## 8  English Opening: Anglo-Indian Defense, Queen's Knight Variation       180+0
## 9                                  Queen's Pawn Game: Levitsky Attack       180+0
## 10                             Nimzo-Larsen Attack: Symmetrical Variation       180+0
##     Termination WhiteTitle BlackTitle moves_line tag_line
## 6  Time forfeit                              119      101
## 7        Normal                              139      121
## 8        Normal                              159      141
## 9  Time forfeit                              179      161
## 10 Time forfeit                              199      181
```

```
skim(My_Df2)
```

Table 4: Data summary

Name	My_Df2
Number of rows	4994
Number of columns	21
Column type frequency:	
character	15
numeric	6
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Event	0	1	16	63	0	18	0
Site	0	1	28	28	0	4994	0
Date	0	1	10	10	0	1	0
Round	0	1	1	1	0	1	0
White	0	1	2	20	0	4116	0
Black	0	1	3	20	0	4098	0
Result	0	1	1	7	0	4	0
UTCDate	0	1	10	10	0	1	0
UTCTime	0	1	8	8	0	510	0
ECO	0	1	1	3	0	284	0
Opening	0	1	1	87	0	947	0
TimeControl	0	1	1	8	0	100	0
Termination	0	1	6	12	0	4	0
WhiteTitle	0	1	0	2	4969	4	0
BlackTitle	0	1	0	2	4971	5	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
WhiteElo	0	1	1582.60	312.04	800	1357.25	1584	1801.00	2544	
BlackElo	0	1	1580.50	314.77	800	1357.00	1582	1797.75	2527	
WhiteRatingDiff	1	1	0.30	17.63	-306	-10.00	2	10.00	338	
BlackRatingDiff	1	1	-0.34	17.94	-221	-10.00	-2	10.00	463	
moves_line	0	1	49971.11	28849.01	19	24993.00	49970	74947.00	99925	
tag_line	0	1	49953.10	28849.01	1	24975.00	49952	74929.00	99907	

Question 3

There are two approaches for this: 1. Taking the quantiles for Black and White separately: (quantiles_WhiteElo and quantiles_BlackElo) 2. Taking the quantiles for all the players, even recurring players (everybody_quantile)

```
quantiles_WhiteElo <- quantile(My_Df2$WhiteElo,probs=c(0.01,0.05,0.25,0.5,0.75,0.90,0.99,0.999,1))
quantiles_BlackElo <- quantile(My_Df2$BlackElo,probs=c(0.01,0.05,0.25,0.5,0.75,0.90,0.99,0.999,1))
mixed_players <- cbind(My_Df2$BlackElo,My_Df2$WhiteElo)
#The above line has every player's Elo in front of it
everybody_quantile <- quantile(mixed_players, probs=c(0.01,0.05,0.25,0.5,0.75,0.90,0.99,0.999,1))

print(quantiles_BlackElo)
```

```
##       1%       5%      25%      50%      75%      90%      99%    99.9%
##  913.930 1072.650 1357.000 1582.000 1797.750 1990.000 2315.070 2493.035
##     100%
## 2527.000
```

```
print(quantiles_WhiteElo)
```

```
##       1%       5%      25%      50%      75%      90%      99%    99.9%
##  922.000 1081.000 1357.250 1584.000 1801.000 1996.000 2307.140 2484.084
##     100%
## 2544.000
```

```
print(everybody_quantile)
```

```
##       1%       5%      25%      50%      75%      90%      99%    99.9%
##  919.000 1076.350 1357.000 1582.000 1799.000 1994.000 2313.000 2493.039
##     100%
## 2544.000
```

Question 4

```r
#Extracting white player score

Score <- sub("1/2","0.5",sub("\\-.*","",My_Df2$Result) ) #Replacing 1/2 with 0.5

White_Score <- as.numeric(Score)
```

```
## Warning: NAs introduced by coercion
```

```r
Diff_Elos <- My_Df2$WhiteElo - My_Df2$BlackElo #Difference of White and Black Elos

#linear model
White_Score_Df_Elo<-as.data.frame(cbind(White_Score,Diff_Elos))
Model_1 <- lm( White_Score~Diff_Elos , data = White_Score_Df_Elo)

# How many points worse should the first player own to win exactly half of the time?
#First we find the average white score : (1+0)/2 = 0.5 then see what Diff_Elos this correspond to.
Diff_To_win_half <- ((0.5 - Model_1$coefficients[1])/Model_1$coefficients[2])

print(Diff_To_win_half)
```

```
## (Intercept)
##   -19.14472
```

```r
summary(Model_1)
```

```
##
## Call:
## lm(formula = White_Score ~ Diff_Elos, data = White_Score_Df_Elo)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.96014 -0.50192  0.04909  0.47213  1.11500
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.171e-01  6.724e-03   76.91   <2e-16 ***
## Diff_Elos   8.950e-04  5.001e-05   17.90   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4751 on 4991 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.0603, Adjusted R-squared:  0.06011
## F-statistic: 320.3 on 1 and 4991 DF,  p-value: < 2.2e-16
```

According to the value "Diff_To_win_half", White should have about 19 less Elos to win exactly half of the time. This proves the slight advantage of playing first (playing with white pieces).

Question 5

```r
#Filtering for players with 2000 Elos abd above:
Filtered_Df2 <- filter(My_Df2,WhiteElo >= 2000 & BlackElo >=2000)

#...And doing the same as the previous question:
Score2 <- sub("1/2","0.5",sub("\\-.*","",Filtered_Df2$Result) )

White_Score2 <- as.numeric(Score2)
```

```
## Warning: NAs introduced by coercion
```

```r
Diff_Elos2 <- Filtered_Df2$WhiteElo - Filtered_Df2$BlackElo


White_Score_Df_Elo2<-as.data.frame(cbind(White_Score2,Diff_Elos2))
Model_2 <- lm( White_Score2~Diff_Elos2 , data = White_Score_Df_Elo2)

Diff_To_win_half2 <- ((0.5 - Model_2$coefficients[1])/Model_2$coefficients[2])

print(Diff_To_win_half2)
```

```
## (Intercept)
##   -18.47285
```

```r
summary(Model_2)
```

```
##
## Call:
## lm(formula = White_Score2 ~ Diff_Elos2, data = White_Score_Df_Elo2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.96089 -0.46138  0.08919  0.42220  0.92983
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.5250065  0.0255149   20.58  < 2e-16 ***
## Diff_Elos2  0.0013537  0.0002169    6.24 1.39e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4571 on 319 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.1088, Adjusted R-squared:  0.106
## F-statistic: 38.94 on 1 and 319 DF,  p-value: 1.389e-09
```

As we can see there's not a significant change in the answer. It is known that professional players playing with black pieces can reduce the advantage of white players with certain moves and strategies. However, the answer "-18.47" is not so different than -19.144.

Question 6

```
first_move_col <- data.frame(c("")) #Intializing column for first move
first_move_ind <- data.frame(c("")) #Intializing column for first move indicator
N <- nrow(My_Df2) #This is number of games

for (j in 1:N){
  clean <- sub(" \\{.*","",chess[My_Df2$moves_line][j]) #using raw data and
  clean <- sub("(.*? .*?) .*", "\\1",clean)
             # moves line indicator to clean first move from the moves line

  cleaned = str_split_fixed(clean, " ", 2) #isolating moves line from move number
                                           #(e.g.: 1. from d4)
  moves_df <- data.frame(cleaned)
  if (moves_df$X2 == "e4"){

    first_move_ind[j,] = 1  #If first move is e4, indicator shows 1 else, 0.
  } else {
    first_move_ind[j,] = 0
  }

  first_move_col[j,] <- moves_df$X2
}

#Adding new column to the data set
df3 <- data.frame(first_move = first_move_col,first_move_ind = first_move_ind)
My_Df3_Q6 <- cbind(My_Df2,df3)

colnames(My_Df3_Q6)[22] <- "first move"
colnames(My_Df3_Q6)[23] <- "first move indicator"

table(My_Df3_Q6$`first move`)
```

```
##
##   0-1    a3    a4    b3    b4    c3    c4    d3    d4    e3    e4    f3    f4
##    10     7     4    68    17     6   170    46  1334    93  2868     6    47
##    g3    g4    h3    h4 Na3?!   Nc3   Nf3   Nh3
##    86    13     3    10     1    28   176     1
```

```
table(My_Df3_Q6$`first move indicator`)
```

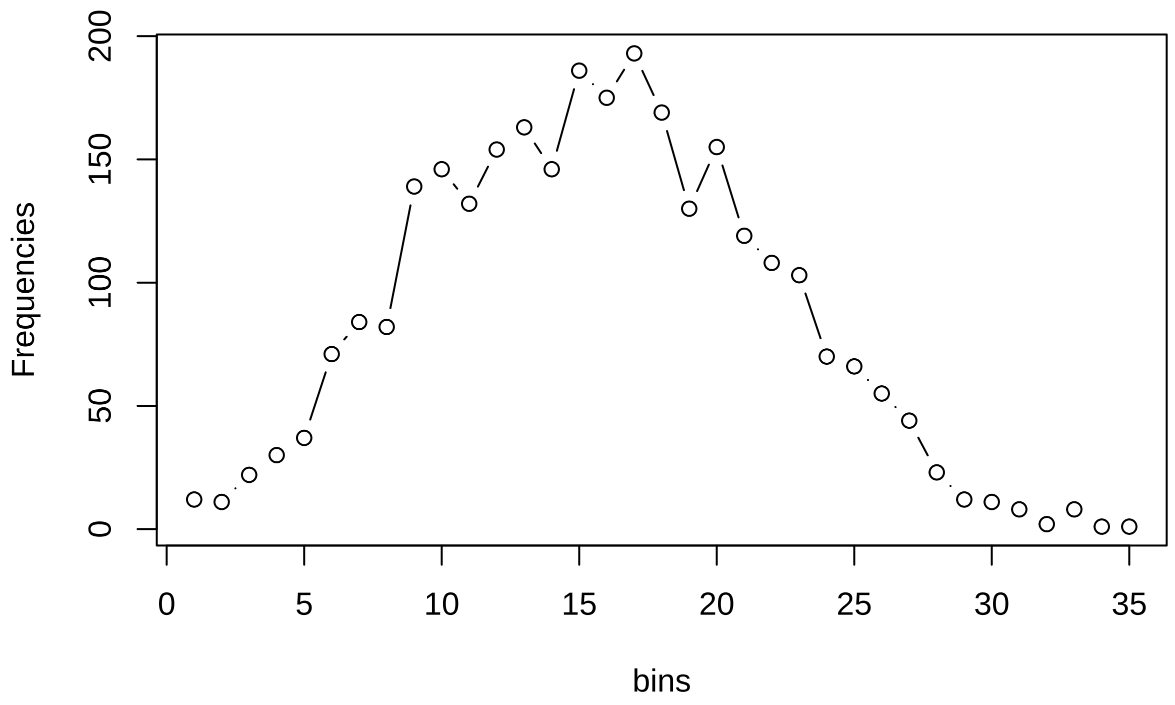
```
##
##    0    1
## 2126 2868
```

Question 7

```
My_Df3_Q7 <- filter(My_Df3_Q6, first_move_ind==1)
freq <- table (findInterval(My_Df3_Q7$WhiteElo,c(-1000000,seq(850,2550,50))))

freq_df <- as.data.frame(freq)
freq_df[,2]
```

```
##  [1]  12  11  22  30  37  71  84  82 139 146 132 154 163 146 186 175 193 169 130
## [20] 155 119 108 103  70  66  55  44  23  12  11   8   2   8   1   1
```

```
freq_df[,1] <-as.numeric(freq_df[,1])

bin = c("<850","850-900","900-950","950-1000","1000-1050","1050-1100"
        ,"1100-1150","1150-1200", "1200-1250",
        "1250-1300","1300-1350","1350-1400","1400-1450",
        "1450-1500","1500-1550","1550-1600","1600-1650","1650-1700",
        "1700-1750", "1750-1800","1800-1850","1850-1900","1900-1950",
        "1950-2000","2000-2050","2050-2100","2100-2150","2150-2200", "2200-2250",
        "2250-2300","2300-2350","2350-2400","2400-2450","2450-2500","2500-2550")
#freq_df[,1]= bin

plot(freq_df[,1],freq_df[,2],type='b',xlab = "bins",ylab = "Frequencies")
```



Question 8

```
avgElo <- as.numeric((My_Df2$WhiteElo + My_Df2$BlackElo)/2) #Taking the average
dex = My_Df3_Q6$moves_line  #finding indexes for moves line
count_blunder <- data.frame(c("")) #Initializing column for blunder moves count

for (m in 1:N){
  count_blunder[m,] <- str_count(chess[dex[m]],"\\?") #looking for "?/??/?!/.."
}

Elo_Blunder_count<-as.data.frame(cbind(count_blunder,avgElo))
colnames(Elo_Blunder_count)[1] <- "count_blunder"
Model_3 <- lm(Elo_Blunder_count$count_blunder~avgElo , data = Elo_Blunder_count)

summary(Model_3)
```

```
##
## Call:
## lm(formula = Elo_Blunder_count$count_blunder ~ avgElo, data = Elo_Blunder_count)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.269 -1.696 -1.465 -1.227 73.445
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.0700979  0.4915042   6.246 4.55e-10 ***
## avgElo      -0.0009938  0.0003051  -3.257  0.00113 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.6 on 4992 degrees of freedom
## Multiple R-squared:  0.002121,   Adjusted R-squared:  0.001921
## F-statistic: 10.61 on 1 and 4992 DF,  p-value: 0.001133
```

Question 9

```
#Extracting time control without the + sign, an adding 1 to avoid ln(0)
T_ctrl <- as.numeric(sub("\\+.*","",My_Df2$TimeControl))+1.
```

```
## Warning: NAs introduced by coercion
```

```
Model_4 <- lm(Elo_Blunder_count$count_blunder~avgElo+ I(log(T_ctrl)) , data = Elo_Blunder_count)
summary(Model_4)
```

```
##
## Call:
## lm(formula = Elo_Blunder_count$count_blunder ~ avgElo + I(log(T_ctrl)),
##     data = Elo_Blunder_count)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.721 -1.937 -1.476 -0.752 73.028
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -1.6370211  0.8065702  -2.030   0.0424 *
## avgElo         -0.0005068  0.0003110  -1.630   0.1032
## I(log(T_ctrl))  0.7678287  0.1045880   7.341 2.46e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.567 on 4975 degrees of freedom
##   (16 observations deleted due to missingness)
## Multiple R-squared:  0.01282,    Adjusted R-squared:  0.01242
## F-statistic:  32.3 on 2 and 4975 DF,  p-value: 1.161e-14
```

Question 10

```r
#Step 1
chess = readLines("chess_960_games.txt")    #Read raw data

chess_2 = str_split_fixed(chess, " ", 2)

is_metadata = str_detect(chess, "^\\[.*\\]$") # Separating meta data from moves line

pgn_meta = chess

pgn_meta[!is_metadata] = ""           #Replacing moves line with an empty line

pgn_meta2 = str_split_fixed(pgn_meta, " ", 2)
pgn_meta3= pgn_meta2[!apply(pgn_meta2 == "", 1, all),]  #Removing empty rows

vars = str_replace(pgn_meta3[,1], "\\[", "")   #Erasing [ from metadata
varnames = unique(vars)                        #Finding unique variable names
varnames = varnames[!varnames == '']           #Removing empty variable names
values = pgn_meta3[,2]

values = str_replace_all(values, "\"", "")      #Erasing [ from metadata values

values = str_replace(values, "\\]$", "")        #Erasing ] from metadata values

New_event = str_detect(pgn_meta3[,1], "Event")  #Number of line where there's a
                                                #new Event in cleaned data
New_event_index = which (New_event)
New_event_index = c(New_event_index,nrow(pgn_meta3)) #including last row index
N_games = length(New_event_index)-1     #Number of games played. -1 because of
                                        #the previous code line
#Initializing a vector to keep generated rows
MyData <- vector( "character" , length(varnames) )


for (i in 1:N_games){

  if (i==N_games){ #This if condition is because we have to extract last line data differently
    a=New_event_index[i]
    b=New_event_index[i+1]

  } else {

    a=New_event_index[i]
    b=New_event_index[i+1]-1

  }


  vars2 <- vars[a:b]  #Slicing variable names for the particular row [i]
  reorder_index <- match(varnames,vars2) #matching variable names[i] with unique varnames
  vars2_reordered <-vars2[reorder_index] #Ordering variable names like varnames
  values_reordered <-values[a:b][reorder_index] #ordering values to correspond to varnames
  values_reordered <- na.omit(values_reordered)  #omitting NA's so they don't mess ordered values
  ind <- varnames %in% vars2_reordered   #Used to know where each value should go
                                       #if values are not ordered as others in metadata

  New_data <- vector( "character" , length(varnames) ) #initializing data row

  New_data[ind] <- values_reordered
  MyData <- rbind (MyData, New_data) #Adding new data below the other data

}

MyData2= MyData[!apply(MyData == "", 1, all),]   #Erasing the first row because it was blank

My_Df <- data.frame(MyData2)                    #Changing vector to data frame

colnames(My_Df)<-varnames    #Column names should be varnames(unique variable names)
rownames(My_Df)<-c()



cols.num <- c("WhiteElo","BlackElo","WhiteRatingDiff","BlackRatingDiff","SetUp")
My_Df[cols.num] <- sapply(My_Df[cols.num],as.numeric)

print(My_Df[6:10,])
```

```
##                  Event                         Site       Date Round
## 6  Rated Chess960 game https://lichess.org/ltDB8sjI 2018.07.31     -
## 7  Rated Chess960 game https://lichess.org/dCbKJLiC 2018.07.31     -
## 8  Rated Chess960 game https://lichess.org/P81nlISz 2018.07.31     -
## 9  Rated Chess960 game https://lichess.org/MSneHIsc 2018.07.31     -
## 10 Rated Chess960 game https://lichess.org/rc3xc9mI 2018.07.31     -
##             White      Black Result    UTCDate  UTCTime WhiteElo BlackElo
## 6   playchess2016    liquero    1-0 2018.07.31 22:18:49     1495     1357
## 7         goooood madmusikus    1-0 2018.07.31 22:19:56     1609     1674
## 8  philodendron68    Proxhie    1-0 2018.07.31 22:22:55     1906     1645
## 9   playchess2016     aqwjui    1-0 2018.07.31 22:24:50     1503     1386
## 10      yhuliomtz   pooridea    0-1 2018.07.31 22:27:48     1426     1485
##    WhiteRatingDiff BlackRatingDiff TimeControl Termination
## 6                8             -10      1200+0      Normal
## 7               65             -13       420+0      Normal
## 8                4             -10       300+4      Normal
## 9                8              -9       900+0      Normal
## 10             -66               9      300+10      Normal
##                                                         FEN SetUp  Variant
## 6  qrbbnkrn/pppppppp/8/8/8/8/PPPPPPPP/QRBBNKRN w KQkq - 0 1     1 Chess960
## 7  rnnbbqkr/pppppppp/8/8/8/8/PPPPPPPP/RNNBBQKR w KQkq - 0 1     1 Chess960
## 8  qbbnrnkr/pppppppp/8/8/8/8/PPPPPPPP/QBBNRNKR w KQkq - 0 1     1 Chess960
## 9  qrbbnkrn/pppppppp/8/8/8/8/PPPPPPPP/QRBBNKRN w KQkq - 0 1     1 Chess960
## 10 bqnrkbnr/pppppppp/8/8/8/8/PPPPPPPP/BQNRKBNR w KQkq - 0 1     1 Chess960
##    WhiteTitle BlackTitle
## 6
## 7
## 8
## 9
## 10
```

```
skim(My_Df)
```

Table 7: Data summary

Name	My_Df
Number of rows	4420
Number of columns	20
Column type frequency:	
character	15
numeric	5
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Event	0	1	19	65	0	17	0
Site	0	1	28	28	0	4420	0
Date	0	1	10	10	0	4	0
Round	0	1	1	1	0	1	0
White	0	1	3	20	0	1004	0
Black	0	1	3	20	0	1014	0
Result	0	1	3	7	0	3	0
UTCDate	0	1	10	10	0	4	0
UTCTime	0	1	8	8	0	4131	0
TimeControl	0	1	3	9	0	98	0
Termination	0	1	6	12	0	3	0
FEN	0	1	1	56	0	928	0
Variant	0	1	8	8	0	1	0
WhiteTitle	0	1	0	3	4315	7	0
BlackTitle	0	1	0	3	4309	8	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
WhiteElo	0	1	1754.45	235.57	966	1579	1747	1902	2721	
BlackElo	0	1	1753.95	235.81	800	1582	1748	1897	2631	
WhiteRatingDiff	0	1	0.67	41.97	-380	-10	1	10	471	
BlackRatingDiff	0	1	-0.50	43.78	-375	-10	-1	9	504	
SetUp	0	1	1.00	0.00	1	1	1	1	1	

```r
#Step 2
NEvent = str_detect(chess_2[,1], "Event")      #Number of line where there's a new Event
tag_line = which (NEvent)                      # Finding the index of

empty_lines <- grep("^$", chess)               # Finding the index of empty lines in raw data
differences <- diff(empty_lines)
moves_line_ind <- which(differences == 2)#If distance is two, means we have a moves line between them
moves_line2 <- empty_lines[moves_line_ind]+1 # Moves line is one line after the first empty line

#Nmoves = str_detect(chess_2[,1], "1.")
moves_line = c(moves_line2,nrow(chess_2))      #Adding the last row index for last event's moves line

df2 <- data.frame(moves_line = moves_line,tag_line = tag_line) #Adding columns to the data frame
My_Df2 <- cbind(My_Df,df2)

cols.num <- c("moves_line","tag_line")
My_Df2[cols.num] <- sapply(My_Df2[cols.num],as.numeric)       #Making them numeric values

print(My_Df2[6:10,])
```

```
##                   Event                         Site       Date Round
## 6  Rated Chess960 game https://lichess.org/ltDB8sjI 2018.07.31     -
## 7  Rated Chess960 game https://lichess.org/dCbKJLiC 2018.07.31     -
## 8  Rated Chess960 game https://lichess.org/P81nlISz 2018.07.31     -
## 9  Rated Chess960 game https://lichess.org/MSneHIsc 2018.07.31     -
## 10 Rated Chess960 game https://lichess.org/rc3xc9mI 2018.07.31     -
##              White      Black Result    UTCDate  UTCTime WhiteElo BlackElo
## 6   playchess2016    liquero    1-0 2018.07.31 22:18:49     1495     1357
## 7         goooood madmusikus    1-0 2018.07.31 22:19:56     1609     1674
## 8  philodendron68    Proxhie    1-0 2018.07.31 22:22:55     1906     1645
## 9   playchess2016     aqwjui    1-0 2018.07.31 22:24:50     1503     1386
## 10      yhuliomtz   pooridea    0-1 2018.07.31 22:27:48     1426     1485
##    WhiteRatingDiff BlackRatingDiff TimeControl Termination
## 6                8             -10      1200+0      Normal
## 7               65             -13       420+0      Normal
## 8                4             -10       300+4      Normal
## 9                8              -9       900+0      Normal
## 10             -66               9      300+10      Normal
##                                                     FEN SetUp  Variant
## 6  qrbbnkrn/pppppppp/8/8/8/8/PPPPPPPP/QRBBNKRN w KQkq - 0 1     1 Chess960
## 7  rnnbbqkr/pppppppp/8/8/8/8/PPPPPPPP/RNNBBQKR w KQkq - 0 1     1 Chess960
## 8  qbbnrnkr/pppppppp/8/8/8/8/PPPPPPPP/QBBNRNKR w KQkq - 0 1     1 Chess960
## 9  qrbbnkrn/pppppppp/8/8/8/8/PPPPPPPP/QRBBNKRN w KQkq - 0 1     1 Chess960
## 10 bqnrkbnr/pppppppp/8/8/8/8/PPPPPPPP/BQNRKBNR w KQkq - 0 1     1 Chess960
##    WhiteTitle BlackTitle moves_line tag_line
## 6                               125      106
## 7                               146      127
## 8                               167      148
## 9                               188      169
## 10                              209      190
```

```r
skim(My_Df2)
```

Table 10: Data summary

Name	My_Df2
Number of rows	4420
Number of columns	22
Column type frequency:	
character	15
numeric	7
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Event	0	1	19	65	0	17	0
Site	0	1	28	28	0	4420	0
Date	0	1	10	10	0	4	0
Round	0	1	1	1	0	1	0
White	0	1	3	20	0	1004	0
Black	0	1	3	20	0	1014	0
Result	0	1	3	7	0	3	0
UTCDate	0	1	10	10	0	4	0
UTCTime	0	1	8	8	0	4131	0
TimeControl	0	1	3	9	0	98	0
Termination	0	1	6	12	0	3	0
FEN	0	1	1	56	0	928	0
Variant	0	1	8	8	0	1	0
WhiteTitle	0	1	0	3	4315	7	0
BlackTitle	0	1	0	3	4309	8	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
WhiteElo	0	1	1754.45	235.57	966	1579.00	1747.0	1902.00	2721	
BlackElo	0	1	1753.95	235.81	800	1582.00	1748.0	1897.00	2631	
WhiteRatingDiff	0	1	0.67	41.97	-380	-10.00	1.0	10.00	471	
BlackRatingDiff	0	1	-0.50	43.78	-375	-10.00	-1.0	9.00	504	
SetUp	0	1	1.00	0.00	1	1.00	1.0	1.00	1	
moves_line	0	1	46538.14	26865.46	20	23267.75	46547.5	69807.50	93033	
tag_line	0	1	46519.09	26865.47	1	23248.75	46528.5	69788.25	93016	

```r
#write.csv(My_Df2, "~/Desktop/UW 2/Data Analysis - Stat 874/A2/Q10.csv", row.names=FALSE)
```

Question 11

Question 6 redone for 960 games:

```
first_move_col <- data.frame(c("")) #Intializing column for first move
first_move_ind <- data.frame(c("")) #Intializing column for first move indicator
N <- nrow(My_Df2) #This is number of games

for (j in 1:N){
  clean <- sub(" \\{.*","",chess[My_Df2$moves_line][j]) #using raw data and
  clean <- sub("(.*? .*?) .*", "\\1",clean)
             # moves line indicator to clean first move from the moves line

  cleaned = str_split_fixed(clean, " ", 2) #isolating moves line from move number
                                           #(e.g.: 1. from d4)
  moves_df <- data.frame(cleaned)
  if (moves_df$X2 == "e4"){

    first_move_ind[j,] = 1  #If first move is e4, indicator shows 1 else, 0.
  } else {
    first_move_ind[j,] = 0
  }

  first_move_col[j,] <- moves_df$X2
}

#Adding new column to the data set
df3 <- data.frame(first_move = first_move_col,first_move_ind = first_move_ind)
My_Df3_Q6 <- cbind(My_Df2,df3) #adding two new column

colnames(My_Df3_Q6)[23] <- "first move"
colnames(My_Df3_Q6)[24] <- "first move indicator" #Renaming columns

table(My_Df3_Q6$`first move`)
```

```
##
## "Chess960"]          0-1           a3           a4         a4?!           b3
##           1           11            6           16            1          544
##        b3?!           b4          b4?         b4?!           c3         c3?!
##           2           92            1            7          195            3
##          c4         c4?!           d3           d4         d4?!           e3
##         250            2           85          469            1          131
##        e3?!           e4          e4?         e4?!           f3         f3?!
##           1          878            1            5          147            4
##          f4         f4?!           g3          g3?         g3?!           g4
##         214            2          636            1           11           76
##        g4?!           h3           h4          Na3         Nab3          Nb3
##           5           10           23            3           15          137
##       Nb3?!         Nbc3          Nc3         Ncb3         Ncd3          Nd3
##           1           11           86            3            3           29
##        Ndc3          Ne3        Ne3?!         Nef3          Nf3         Nfe3
##           2           21            1            1          102            3
##         Ng3        Ng3?!         Ngf3         Nhg3          O-O
##         122            3           15           21           10
```

```
table(My_Df3_Q6$`first move indicator`)
```

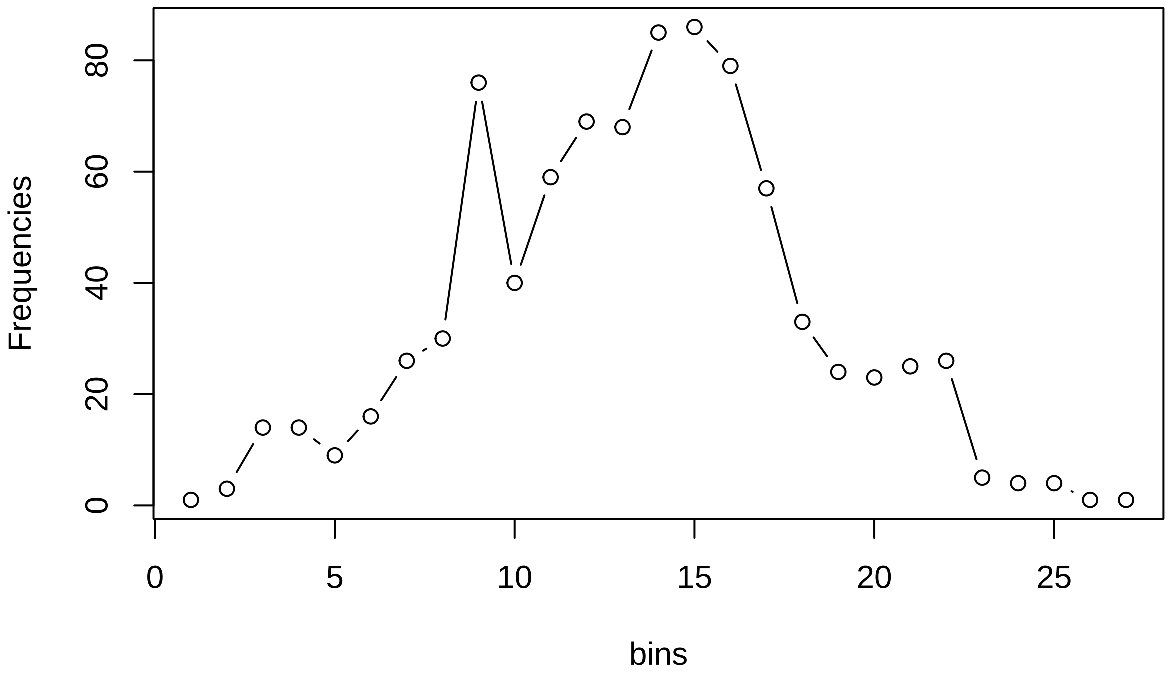
```
##
##    0    1
## 3542  878
```

Question 7 redone for 960 games:

```r
My_Df3_Q7 <- filter(My_Df3_Q6, first_move_ind==1)
freq <- table (findInterval(My_Df3_Q7$WhiteElo,c(-1000000,seq(850,2550,50))))

freq_df <- as.data.frame(freq)
freq_df[,2]
```

```
##  [1]  1  3 14 14  9 16 26 30 76 40 59 69 68 85 86 79 57 33 24 23 25 26  5  4  4
## [26]  1  1
```

```r
freq_df[,1] <-as.numeric(freq_df[,1])

bin = c("<850","850-900","900-950","950-1000","1000-1050","1050-1100"
        ,"1100-1150","1150-1200", "1200-1250",
        "1250-1300","1300-1350","1350-1400","1400-1450",
        "1450-1500","1500-1550","1550-1600","1600-1650","1650-1700",
        "1700-1750", "1750-1800","1800-1850","1850-1900","1900-1950",
        "1950-2000","2000-2050","2050-2100","2100-2150","2150-2200", "2200-2250",
        "2250-2300","2300-2350","2350-2400","2400-2450","2450-2500","2500-2550")
bin <- factor(bin)
#freq_df[,1]= bin
#par(las = 2)
plot(freq_df[,1],freq_df[,2],type='b',xlab = "bins",ylab = "Frequencies")
```



Looks like people with an average degree of skill use e4 as their first move more frequently. More professional players might use different strategies and hence might not use e4 as their first move. On the other hand less skilled players use a variety of moves other than e4 as their first move. The distribution of first moves looks like Guassian distribution, with the peak slightly to the right. We see a data point that does not follow the normal distribution. It could be an outlier specific to these data set.