Chess Graphs

Atul Suresh Kumar, Dhushyanth Mohan Kumar, Manoghn Kandiraju

18/08/2021

```
library(ggplot2)
library(dplyr)
```

```
##
## 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(gganimate)
library(gifski)
library(png)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.3     v purrr   0.3.4
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(network)
```

```
##
## 'network' 1.17.1 (2021-06-12), part of the Statnet Project
## * 'news(package="network")' for changes since last version
## * 'citation("network")' for citation information
## * 'https://statnet.org' for help, support, and other information
```

```
library(sna)
```

```
## Loading required package: statnet.common
```

```
##
## Attaching package: 'statnet.common'
```

```
## The following objects are masked from 'package:base':
##
##     attr, order
```

```
## sna: Tools for Social Network Analysis
## Version 2.6 created on 2020-10-5.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
finaldata <- read.csv("~/Desktop/FDA/Project/clean_ds_post_move_analysis.csv")
```

```
suppressWarnings(cat("Mean White ELO: ", mean(na.omit(as.numeric(as.character(finaldata$White.ELO)))),
```

```
## Mean White ELO:  1393.914
```

```
suppressWarnings(cat("Mean Black ELO: ", mean(na.omit(as.numeric(as.character(finaldata$Black.ELO)))),
```

```
## Mean Black ELO:  1411.102
```

```
suppressWarnings(atulWhite <- as.numeric(as.character(finaldata[finaldata$White == "Atul17",]$White.ELO
suppressWarnings(dosaWhite <- as.numeric(as.character(finaldata[finaldata$White == "dosa15",]$White.ELO
suppressWarnings(manoWhite <- as.numeric(as.character(finaldata[finaldata$White == "manoghn",]$White.ELO
suppressWarnings(atulWhite <- na.omit(as.numeric(atulWhite[2:length(atulWhite)])))
suppressWarnings(dosaWhite <- na.omit(as.numeric(dosaWhite[2:length(dosaWhite)])))
suppressWarnings(manoWhite <- na.omit(as.numeric(manoWhite[2:length(manoWhite)])))
suppressWarnings(dosaBlack <- as.numeric(as.character(finaldata[finaldata$Black == "dosa15",]$Black.ELO
suppressWarnings(atulBlack <- as.numeric(as.character(finaldata[finaldata$Black == "Atul17",]$Black.ELO
suppressWarnings(manoBlack <- as.numeric(as.character(finaldata[finaldata$Black == "manoghn",]$Black.ELO

suppressWarnings(cat("Average ELO for Atul: ", mean(c(mean(atulWhite), mean(atulBlack))), "\n"))
```

```
## Average ELO for Atul:  1378.657
```

```
suppressWarnings(cat("Average ELO for Dhushyanth: ", mean(c(mean(dosaWhite), mean(dosaBlack))), "\n"))
```

```
## Average ELO for Dhushyanth:  1522.583
```

```
suppressWarnings(cat("Average ELO for Manoghn: ", mean(c(mean(manoWhite), mean(manoBlack))), "\n"))
```

```
## Average ELO for Manoghn:  1203.265
```

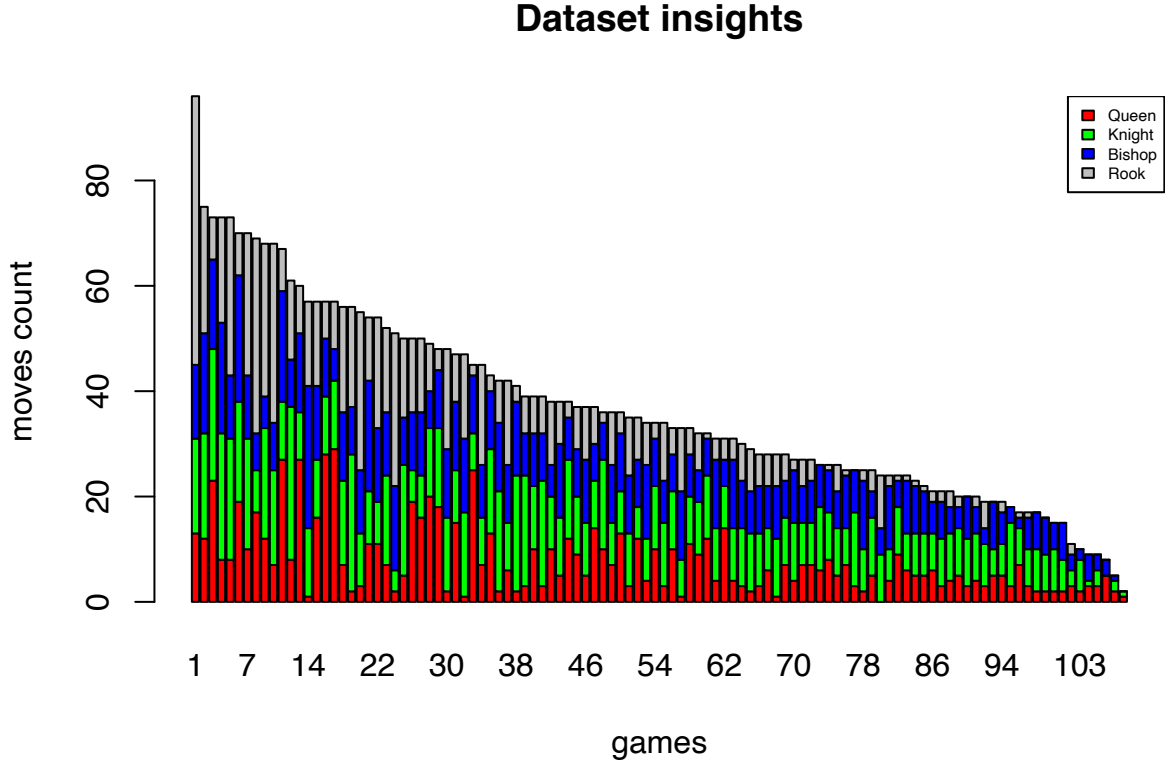
```
suppressWarnings(cat("Average moves per game: ", mean(finaldata$Num.Moves), "\n"))
```

```
## Average moves per game:  64.5463
```

The above calculations help us obtain a rough idea of what the skill level of our chess engine is going to be, as it will be trained based on data retreived from this dataset at hand.

Stacked bar chart of Number of Pieces moved

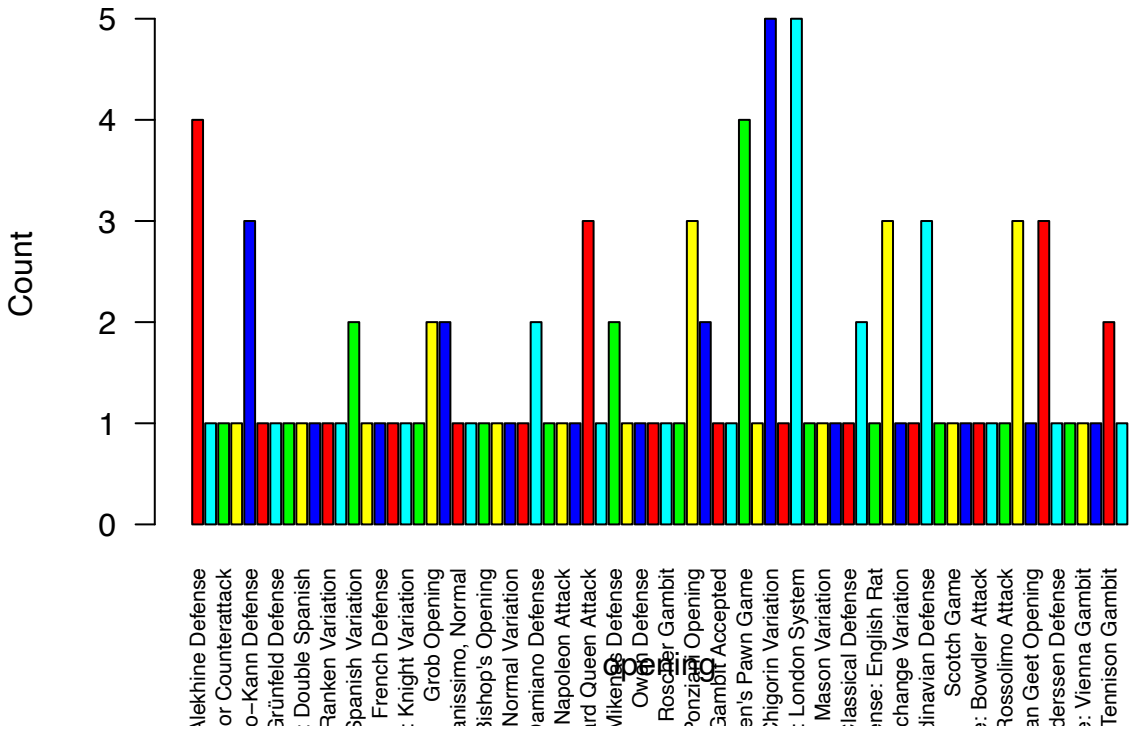
```
withoutpawns<-finaldata[c('Queen','Knight','Bishop','Rook')]
withoutpawns$totalmoves=withoutpawns$Queen+withoutpawns$Knight+withoutpawns$Bishop+withoutpawns$Rook
finaldatawithoutpawn1<-withoutpawns %>% arrange(desc(totalmoves))
pieces <- c("Queen","Knight","Bishop","Rook")
games<-c(1:108)

Values <- matrix(c(finaldatawithoutpawn1$Queen,finaldatawithoutpawn1$Knight,
                   finaldatawithoutpawn1$Bishop,finaldatawithoutpawn1$Rook),
                 nrow = 4, ncol = 108, byrow = TRUE)

bargraph<-barplot(Values, main = "Dataset insights",names.arg = games,
                  xlab = "games", ylab = "moves count", col = c("red","green","blue","grey"))
legend("topright", pieces, cex = 0.5, fill =c("red","green","blue","grey"))
```



The above generated bar graph denotes how often each different chess piece is moved in each game from our dataset.

Bar chart of Openings

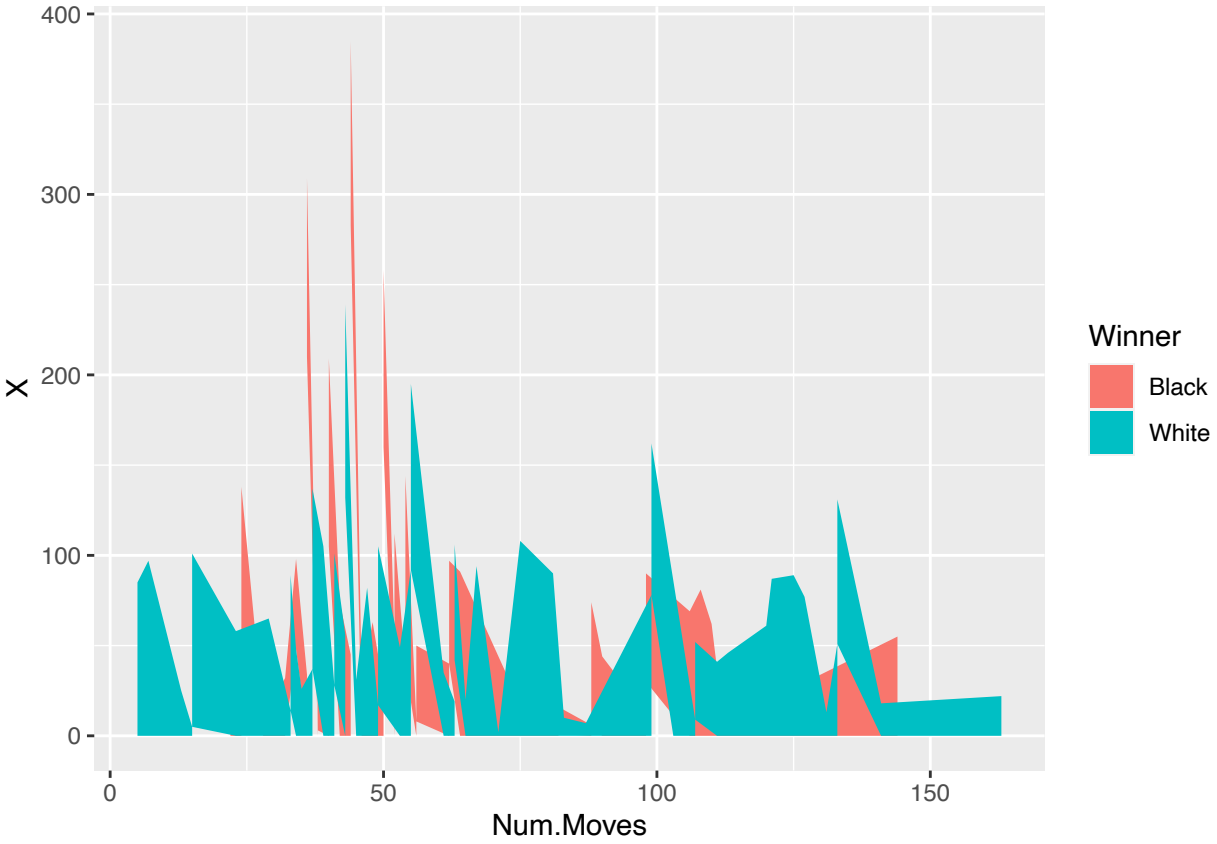
```
df1<-table(finaldata$Opening)

barplot(df1,las =2, cex.names = 0.7, xlab = "opening",ylab = "Count",
        col = rep(c("red", "cyan", "green", "yellow", "blue"), 22))
```



This bar chart determines the frequency of various openings encountered in our games, which further helps determine what chess strategy our engine is based on, and the kind of openings it is likely to play.

Area chart of Game Winner

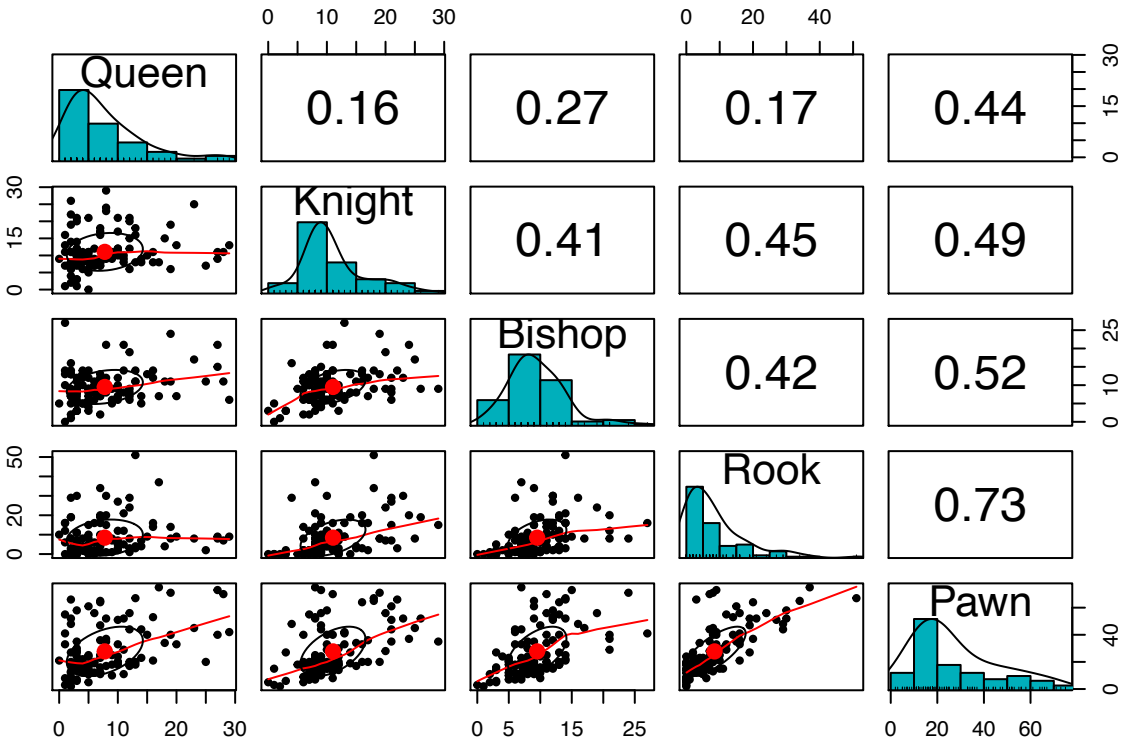
```
plot=ggplot(finaldata,aes(x=Num.Moves,y=X,fill=Winner))
plot+geom_area()
```



This area chart helps us analyze which side of the board (Black or White) was the more likely side to win a match, given the number of moves played in that same match. Since our engine is set to only play the black pieces, this gives us a rough estimate of how well the engine, as Black, would fare in any game against a user.

Scatter plot of Pieces

```
pairs.panels(finaldata[15:19],
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
             )
```



This scatter plot gives us the correlation between any combination of two pieces on the chess board. It helps us conclude the chess patterns that the engine is expected to follow, as most tactical plays in this game would require some form of coordiation between 2 or more pieces from White or Black. From our analysis, we can see how the Rook and Pawn combination have the highest correlation of 0.73, which means the engine is likely to advance on the user with a rook-and-pawn offensive gameplay. However, considering the fact that half of the pieces on each side comprises of a pawn, the last column is likely to have a certain bias in its correlation coefficient. Hence a more accurate conclusion would be to consider the next highest coefficient outside of that column, which is 0.45 with the Rook and Knight. Therefore, aside from pawn pushes in the game, the engine is likely to use its Knights and Rooks to its fullest extent in order to best the user.

The above plotted line graph gives us an extra insight into the teammates of the project itself. Since the engine is designed to play like us, it is worthy to note our skill progression over all the games we played, constituting this dataset. The green line, which represents Atul, is shown to have a rather steady score over all the games as his playstyle appears to have been the most consistent with an approximate of a 50-50 win/loss rate. The red line which represents Dhushyanth has quite a rocky progression as his gameplay is comparatively more inconsistent. Nevertheless his overall average ELO appears to be higher than the other two team members. Last but not least is the blue line which belongs to Manoghn. He shows a clear improvement over the span of his games, as his ELO went up from approximately 950 to 1250, with the small exception of a downfall as we approach the last few games.

Network map of Openings

```
chessDataset <- read.csv("../ChessDataset.csv")
# View(chessDataset)

movelists <- list()
for(movelist in chessDataset$Game.Movelist)
  movelists <- append(movelists, strsplit(movelist, split = ","))
col_len <- max(lengths(movelists[1:length(movelists)]))

df <- as.data.frame(cbind(movelists))[[1]]
# View(df)

firstMove <- c()
for(i in df) {
  firstMove <- c(firstMove, i[[1]])
}
firstMove <- unique(firstMove)
firstMove
```

```
##  [1] "e4"   "d4"   "Nc3"  "c4"   "g3"   "e3"   "d3"   "Nf3"  "Nd4"  "g4"
## [11] "f3"   "b4"   "f4"   "c3"   "Rb1"  "Rc7"  "Rc2+" "Be3"  "Bf8"
```

```
secondMove <- c()
for(i in df) {
  secondMove <- c(secondMove, i[[2]])
}
secondMove <- unique(secondMove)
secondMove
```

```
##  [1] "Nf6"  "d5"   "e5"   "Nc6"  "c6"   "e6"   "c5"   "d6"   "f6"   "b6"
## [11] "Rb4"  "g5"   "b5"   "f5"   "Kxb1" "b2"   "bxc2"
```

```
thirdMove <- c()
for(i in df) {
  if(length(i) >= 3)
  thirdMove <- c(thirdMove, i[[3]])
```

```
}
thirdMove <- unique(thirdMove)
thirdMove
```

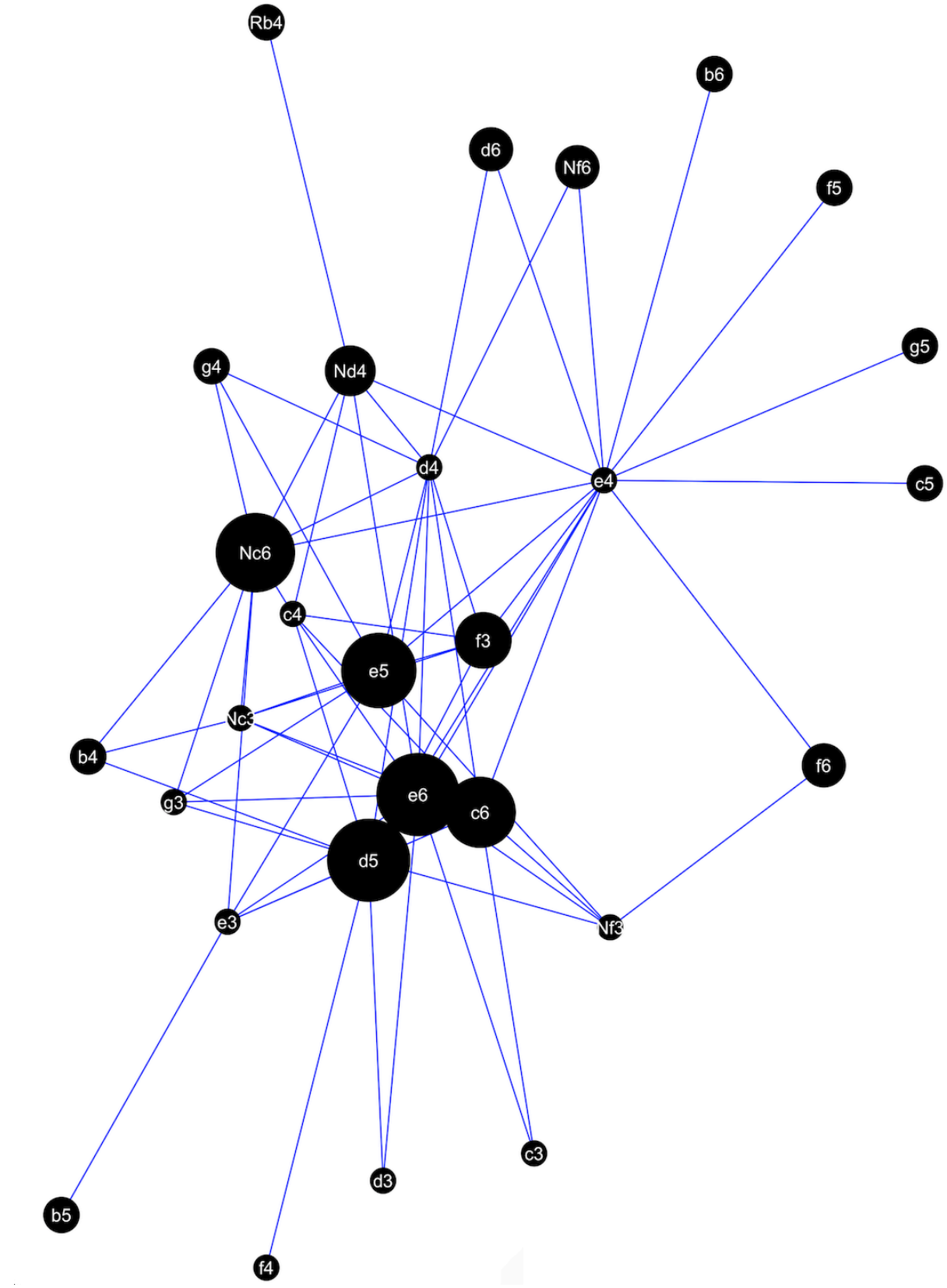
```
##  [1] "f3"   "Nc3"  "Qh5"  "Nf3"  "e3"   "Bf4"  "d4"   "exd5" "Bc4"  "d3"
## [11] "c4"   "Bg2"  "Qf3"  "Bg5"  "e4"   "Be3"  "g3"   "h3"   "f4"   "Nh3"
## [21] "Bxb5" "Bd3"  "Ne2"  "Nxe5" "Bd2"  "Ra7+" "Kd5"  "Rc2"
```

```
fourthMove <- c()
for(i in df) {
  if(length(i) >= 4)
  fourthMove <- c(fourthMove, i[[4]])
}
fourthMove <- unique(fourthMove)
fourthMove
```

```
##  [1] "d5"     "c6"     "Nc6"    "g6"     "e5"     "Nf6"    "b6"     "Qxd5"
##  [9] "f6"     "Bd6"    "d6"     "Na6"    "dxe4"   "e6"     "Bb4"    "Bc5"
## [17] "h6"     "Bb7"    "cxd4"   "Ka3"    "Bf5"    "dxc4"   "Bg4"    "Nxe4"
## [25] "b5"     "c5"     "Qd7"    "Ka2"    "Kb3"    "Kb1"    "bxc1=Q"
```

```
N <- length(c(firstMove, secondMove, thirdMove, fourthMove))
adj_matrix <- matrix(rep(0, N*N), N)

for(movelist in df) {
  for(i1 in 1:length(firstMove)) {
    for(i2 in 1:length(secondMove)) {
      for(i3 in 1:length(thirdMove)) {
        for(i4 in 1:length(fourthMove)) {
          if(length(movelist) >= 4 && movelist[[1]] == firstMove[i1] && movelist[[2]] == secondMove[i2]
            adj_matrix[i1,length(firstMove)+i2] <- 1
            # adj_matrix[length(firstMove)+i2,i1] <- 1
            if(movelist[[3]] == thirdMove[i3]) {
              adj_matrix[i2,length(c(secondMove, thirdMove))+i3] <- 1
              # adj_matrix[length(c(secondMove, thirdMove))+i3,i2] <- 1
              if(movelist[[4]] == fourthMove[i4]) {
                adj_matrix[i3,length(c(thirdMove, fourthMove))+i4] <- 1
                # adj_matrix[length(c(thirdMove, fourthMove))+i4,i3] <- 1
              }
            }
          }
        }
      }
    }
  }
}

nw <- network(adj_matrix, directed = TRUE)
nodeNames <- c(firstMove, secondMove, thirdMove, fourthMove)
network.vertex.names(nw) <- nodeNames
p1 <- ggnet2(nw, size = "indegree", max_size = 30,
             label=c(firstMove, secondMove, thirdMove, fourthMove),
             node.color="black",
             label.color="white", label.size = 5,
             edge.color = "blue", edge.size = 0.5,
```

```
                arrow.size = 4, arrow.gap = 0.0005)  #, "#2db339", "#1dd12e", "#0ced21"
show(p1)
```



```
ggsave("networkgraph.png", plot = p1, width=20, height=25.7,  limitsize = FALSE)
```

This highly dense network graph is quite informative from a Chess perspective. It details every single move that is possible for the engine to play as its first 4 moves. The first 4 mores are quite important in determining the exact type of openings chosen by a chess player, and helps determine whether the chosen approach is that of offense or defense. The network is ordered such that the size of the node determines the likelihood of the encompassed move(in Standard Algebraic Notation) being the first move played by Black(the engine), and incidentally the second move of the game. For example, e5 and d5 are two of the most common pawns used in the begining of a chess game (responses to the King's Pawn Opening and Queen's Pawn Opening respectively), and as we can see they are two of the largest nodes in the network. Branching off to another move such as Nf3 (from e5) denotes the move the engine expects the user to play, following with it could play something like f6.