AI Final Project Data Analysis

2024-04-27

```r
# read CSVs into dataframes
mc_data <- read.csv("montecarlo.csv")
em_data <- read.csv("expectiminimax.csv")
mm_data <- read.csv("minimax.csv")

# extract columns
mc_moves <- mc_data$Number.of.Moves
mc_score <- mc_data$Final.Score
mc_wl <- (mc_data$Win.Loss == "W") * 1
mc_time <- mc_data$Time.Elapsed

em_moves <- em_data$Number.of.Moves
em_score <- em_data$Final.Score
em_wl <- em_data$Win.Loss
em_time <- em_data$Time.Elapsed

mm_moves <- mm_data$Number.of.Moves
mm_score <- mm_data$Final.Score
mm_wl <- mm_data$Win.Loss
mm_time <- mm_data$Time.Elapsed

# fit linear regression models
mc_model <- lm(mc_score ~ mc_moves)
em_model <- lm(em_score ~ em_moves)
mm_model <- lm(mm_score ~ mm_moves)

# calculate range for non-outliers
mcq1 <- quantile(mc_moves, 0.25)
mcq3 <- quantile(mc_moves, 0.75)
mciqr <- mcq3 - mcq1
mcthreshold <- 1.5 * mciqr

em_moves_q1 <- quantile(em_moves, 0.25)
em_moves_q3 <- quantile(em_moves, 0.75)
em_moves_iqr <- em_moves_q3 - em_moves_q1
em_thresholdeshold <- 1.5 * em_moves_iqr

mmq1 <- quantile(mm_score, 0.25)
mmq3 <- quantile(mm_score, 0.75)
mmiqr <- mmq3 - mmq1
mmthreshold <- 1.5 * mmiqr

# get and remove outliers
```

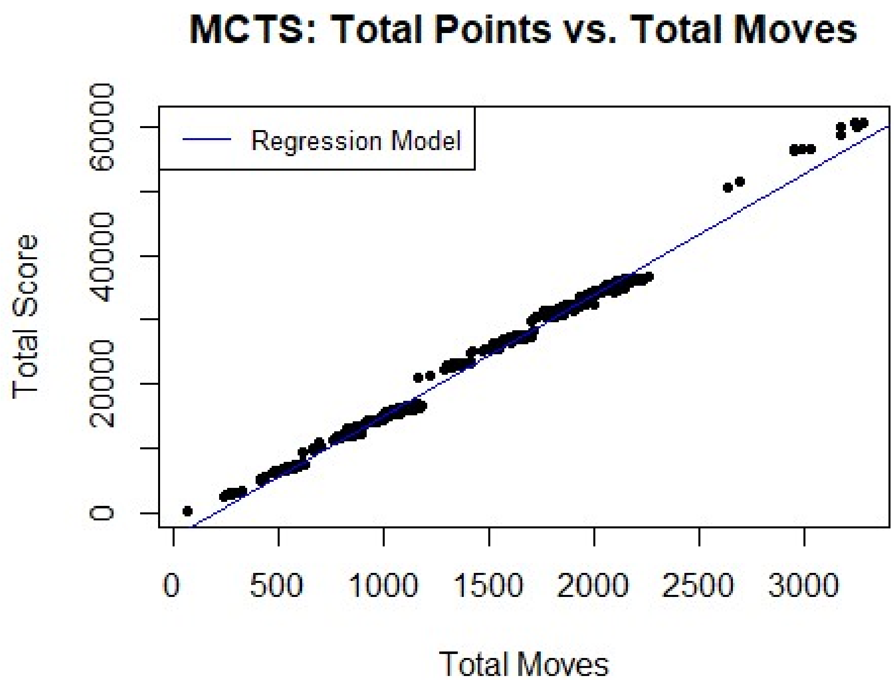
```r
mcoutliers <- mc_moves < (mcq1 - mcthreshold) | mc_moves > (mcq3 +
mcthreshold)
mc_score_no_outlier <- mc_score[!mcoutliers]
mc_moves_no_outlier <- mc_moves[!mcoutliers]

em_move_outliers <- em_moves < (em_moves_q1 - em_thresholdeshold) | em_moves
> (em_moves_q3 + em_thresholdeshold)
em_score_no_outlier <- em_score[!em_move_outliers]
em_moves_no_outlier <- em_moves[!em_move_outliers]

mmoutliers <- mm_score < (mmq1 - mmthreshold) | mm_score > (mmq3 +
mmthreshold)
mm_score_no_outlier <- mm_score[!mmoutliers]
mm_moves_no_outlier <- mm_moves[!mmoutliers]

# regression models without outliers
mc_no_outlier_model <- lm(mc_score_no_outlier ~ mc_moves_no_outlier)
em_no_outlier_model <- lm(em_score_no_outlier ~ em_moves_no_outlier)
mm_no_outlier_model <- lm(mm_score_no_outlier ~ mm_moves_no_outlier)
```

Total Moves vs. Total Scores

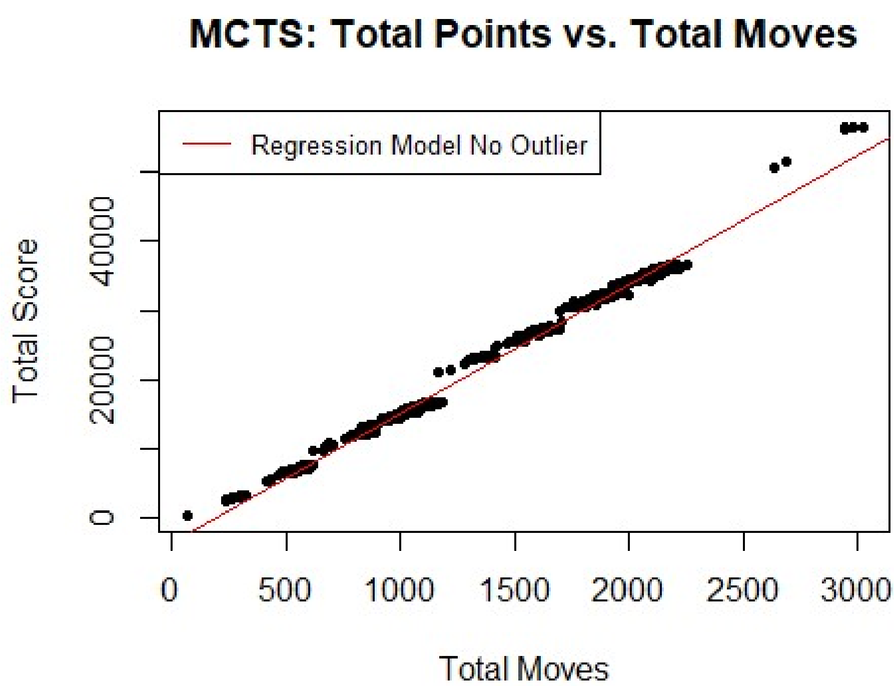
There are two plots for each algorithm: One with all the data, and another with the outliers removed. Regression models are fitted to each of the data sets and the slope from the model without outliers is used as a less biased measure of the correlation between total points and total moves.

Monte Carlo Tree Search

```r
# plot with outliers
plot(mc_moves, mc_score,
     main = "MCTS: Total Points vs. Total Moves",
     xlab = "Total Moves", ylab = "Total Score", pch = 20)
abline(mc_model, col = "blue", lwd = 1.5)
legend("topleft", legend=c("Regression Model"),
       col=c("blue"), lty=1:1, cex=0.8)
```




```r
# plot without outliers
plot(mc_moves_no_outlier, mc_score_no_outlier,
     main = "MCTS: Total Points vs. Total Moves",
     xlab = "Total Moves", ylab = "Total Score", pch = 20)
abline(mc_no_outlier_model, col = "red", lwd = 1.5)
legend("topleft", legend=c("Regression Model No Outlier"),
       col=c("red"), lty=1:1, cex=0.8)
```



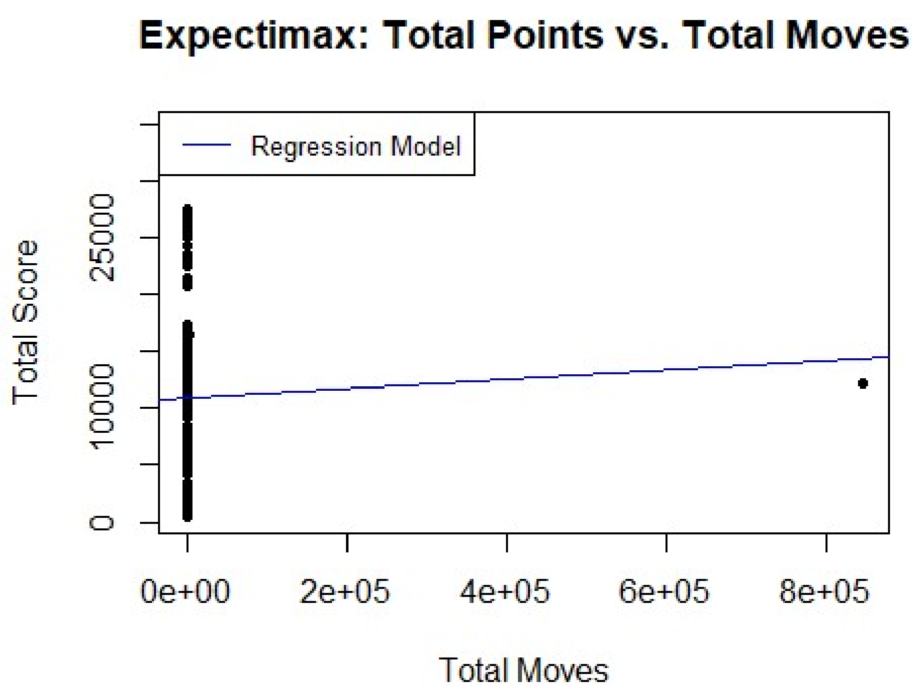

```r
# summary stats
summary(mc_no_outlier_model)
```

```
##
## Call:
## lm(formula = mc_score_no_outlier ~ mc_moves_no_outlier)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1893.6  -611.9   -36.3   509.1  5124.3
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -3.487e+03  7.552e+01  -46.18   <2e-16 ***
## mc_moves_no_outlier  1.862e+01  5.198e-02  358.25   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 871 on 958 degrees of freedom
## Multiple R-squared:  0.9926, Adjusted R-squared:  0.9926
## F-statistic: 1.283e+05 on 1 and 958 DF,  p-value: < 2.2e-16
```

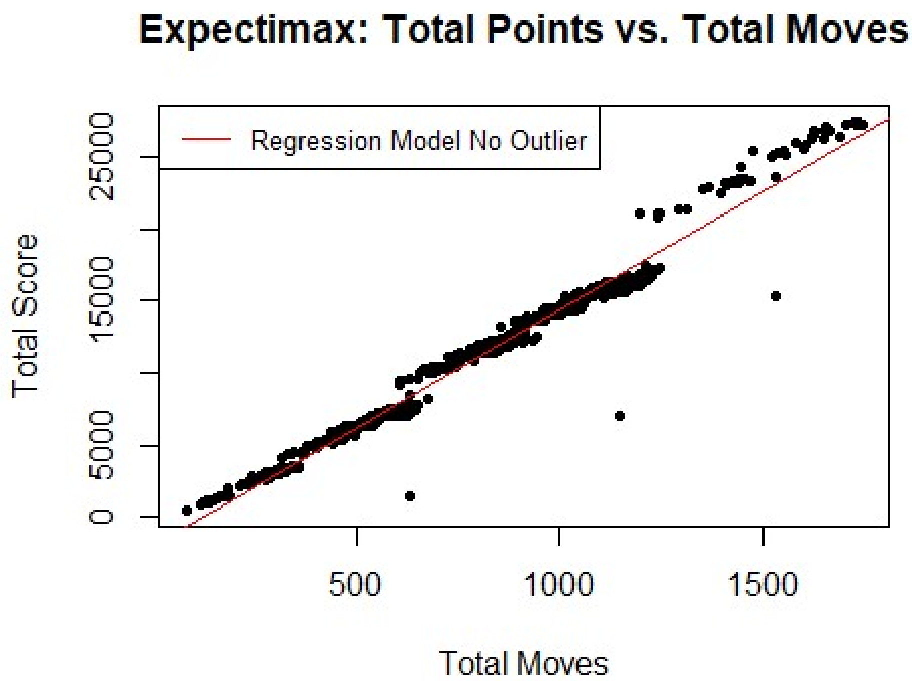
For Monte Carlo Tree Search, the slope for the change in points per move is 18.54. The R^2 value for the relationship between total points and total moves is 0.9929, which means 99.29% of the variability in the total score can be explained by its relationship with the total moves made.

Expectimax

```r
# plot with outliers
plot(em_moves, em_score,
     main = "Expectimax: Total Points vs. Total Moves",
     xlab = "Total Moves", ylab = "Total Score", pch = 20)
abline(em_model, col = "blue", lwd = 1.5)
legend("topleft", legend=c("Regression Model"),
       col=c("blue"), lty=1:1, cex=0.8)
```




```r
# plot without outliers
plot(em_moves_no_outlier, em_score_no_outlier,
     main = "Expectimax: Total Points vs. Total Moves",
     xlab = "Total Moves", ylab = "Total Score", pch = 20)
abline(em_no_outlier_model, col = "red", lwd = 1.5)
legend("topleft", legend=c("Regression Model No Outlier"),
       col=c("red"), lty=1:1, cex=0.8)
```

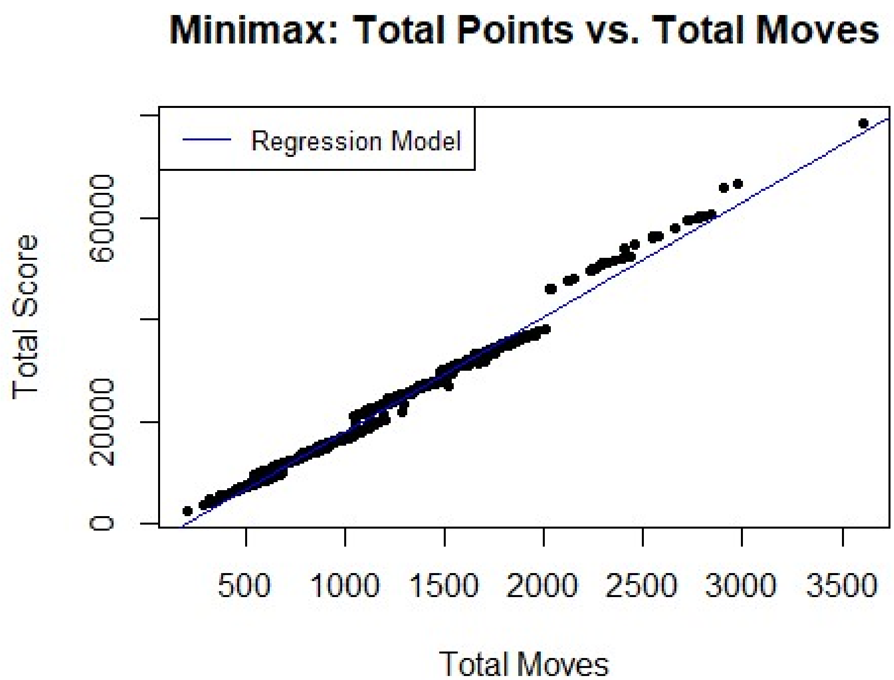



```
# summary stats
summary(em_no_outlier_model)

##
## Call:
## lm(formula = em_score_no_outlier ~ em_moves_no_outlier)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9794.8  -388.0   -57.2   341.7  3477.9
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -1.960e+03  6.602e+01  -29.69   <2e-16 ***
## em_moves_no_outlier  1.639e+01  7.765e-02  211.04   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 805.7 on 977 degrees of freedom
## Multiple R-squared:  0.9785, Adjusted R-squared:  0.9785
## F-statistic: 4.454e+04 on 1 and 977 DF,  p-value: < 2.2e-16
```

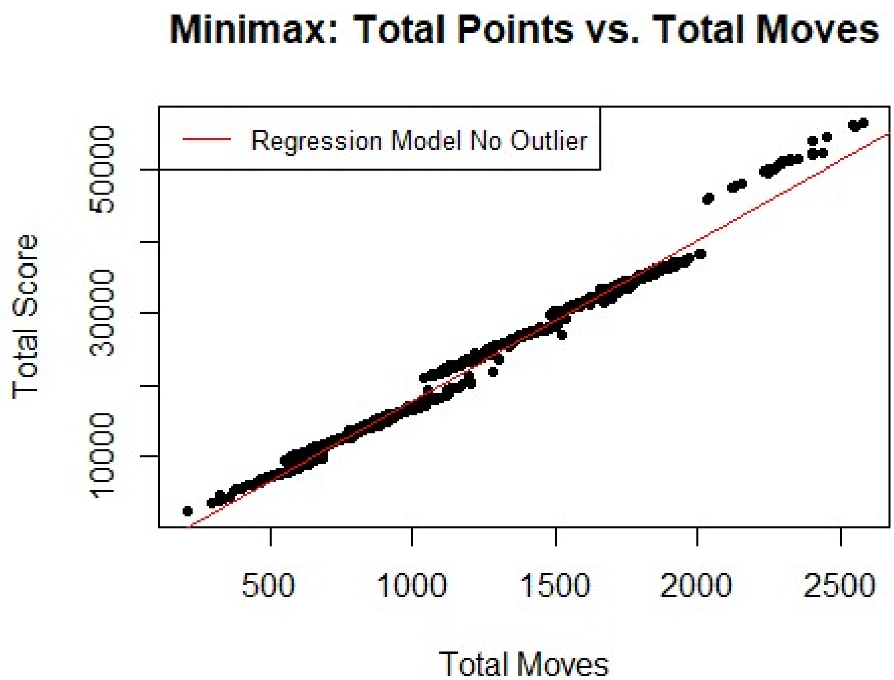
For Expectimax, the slope for the change in points per move is 16.39. The R^2 value for the relationship between total points and total moves is 0.9785, which means 97.85% of the variability in the total score can be explained by its relationship with the total moves made.

Minimax

```r
# plot with outliers
plot(mm_moves, mm_score,
     main = "Minimax: Total Points vs. Total Moves",
     xlab = "Total Moves", ylab = "Total Score", pch = 20)
abline(mm_model, col = "blue", lwd = 1.5)
legend("topleft", legend=c("Regression Model"),
       col=c("blue"), lty=1:1, cex=0.8)
```




```r
# plot without outliers
plot(mm_moves_no_outlier, mm_score_no_outlier,
     main = "Minimax: Total Points vs. Total Moves",
     xlab = "Total Moves", ylab = "Total Score", pch = 20)
abline(mm_no_outlier_model, col = "red", lwd = 1.5)
legend("topleft", legend=c("Regression Model No Outlier"),
       col=c("red"), lty=1:1, cex=0.8)
```




```r
# summary stats
summary(mm_no_outlier_model)
```

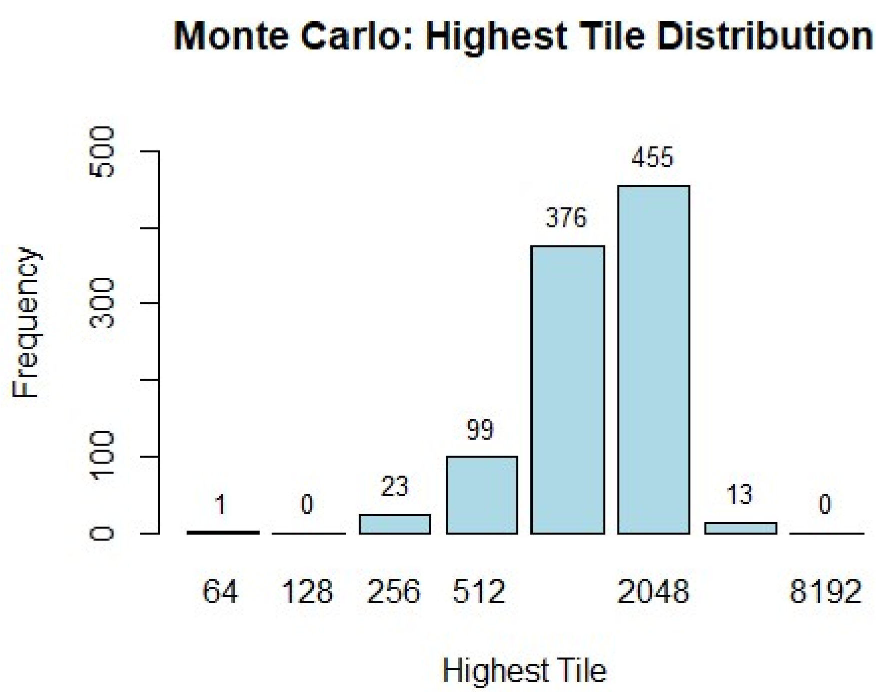
```
##
## Call:
## lm(formula = mm_score_no_outlier ~ mm_moves_no_outlier)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2475.5  -732.5  -133.5   539.0  5170.6
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -4455.3279    93.6750  -47.56   <2e-16 ***
## mm_moves_no_outlier    22.2613     0.0738  301.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1096 on 984 degrees of freedom
## Multiple R-squared:  0.9893, Adjusted R-squared:  0.9893
## F-statistic: 9.098e+04 on 1 and 984 DF,  p-value: < 2.2e-16
```

For Minimax, the slope for the change in points per move is 22.26 The R^2 value for the relationship between total points and total moves is 0.9893, which means 98.93% of the variability in the total score can be explained by its relationship with the total moves made.

Highest Tile Distributions

Monte Carlo Tree Search

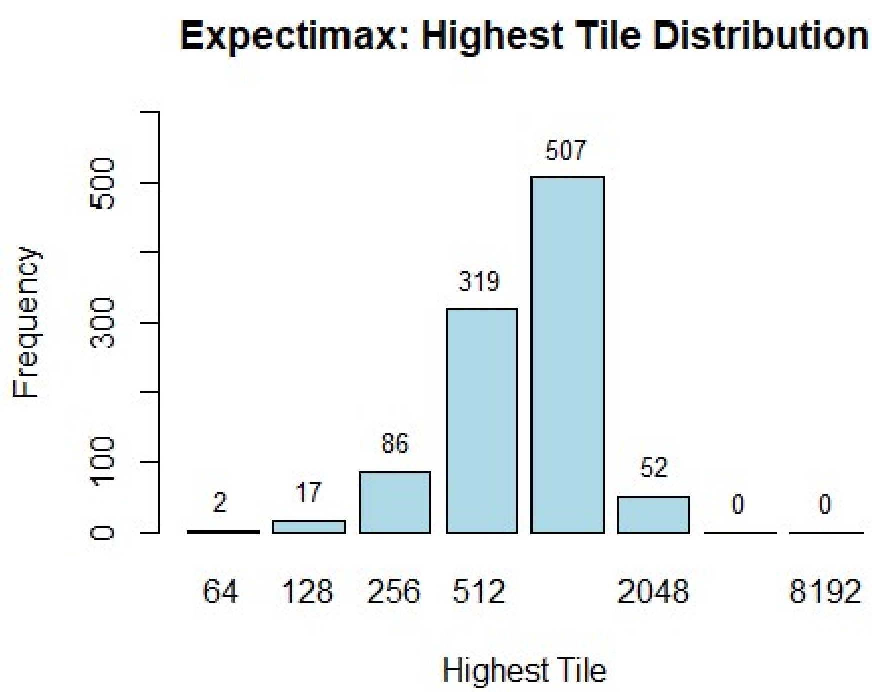
```r
mc_64 <- which(mc_data$Highest.Tile  == 64)
mc_128 <- which(mc_data$Highest.Tile == 128)
mc_256 <- which(mc_data$Highest.Tile == 256)
mc_512 <- which(mc_data$Highest.Tile == 512)
mc_1024 <- which(mc_data$Highest.Tile == 1024)
mc_2048 <- which(mc_data$Highest.Tile == 2048)
mc_4096 <- which(mc_data$Highest.Tile == 4096)
mc_8192 <- which(mc_data$Highest.Tile == 8192)
values <- c(64, 128, 256, 512, 1024, 2048, 4096, 8192)

mc_freq <- c(length(mc_64), length(mc_128), length(mc_256), length(mc_512),
length(mc_1024), length(mc_2048), length(mc_4096), length(mc_8192))

mc_tile_plot <- barplot(mc_freq, names.arg = values, col = "lightblue", main
= "Monte Carlo: Highest Tile Distribution", xlab = "Highest Tile", ylab =
"Frequency", ylim = c(0, 550))

text(x = mc_tile_plot, y = mc_freq, labels = mc_freq, cex = 0.8, pos = 3)
```



Expectimax

```r
em_64 <- which(em_data$Highest.Tile  == 64)
em_128 <- which(em_data$Highest.Tile == 128)
```

```r
em_256 <- which(em_data$Highest.Tile == 256)
em_512 <- which(em_data$Highest.Tile == 512)
em_1024 <- which(em_data$Highest.Tile == 1024)
em_2048 <- which(em_data$Highest.Tile == 2048)
em_4096 <- which(em_data$Highest.Tile == 4096)
em_8192 <- which(em_data$Highest.Tile == 8192)
values <- c(64, 128, 256, 512, 1024, 2048, 4096, 8192)

em_freq <- c(length(em_64), length(em_128), length(em_256), length(em_512),
length(em_1024), length(em_2048), length(em_4096), length(em_8192))

em_tile_plot <- barplot(em_freq, names.arg = values, col = "lightblue", main
= "Expectimax: Highest Tile Distribution", xlab = "Highest Tile", ylab =
"Frequency", ylim = c(0, 600))

text(x = em_tile_plot, y = em_freq, labels = em_freq, cex = 0.8, pos = 3)
```

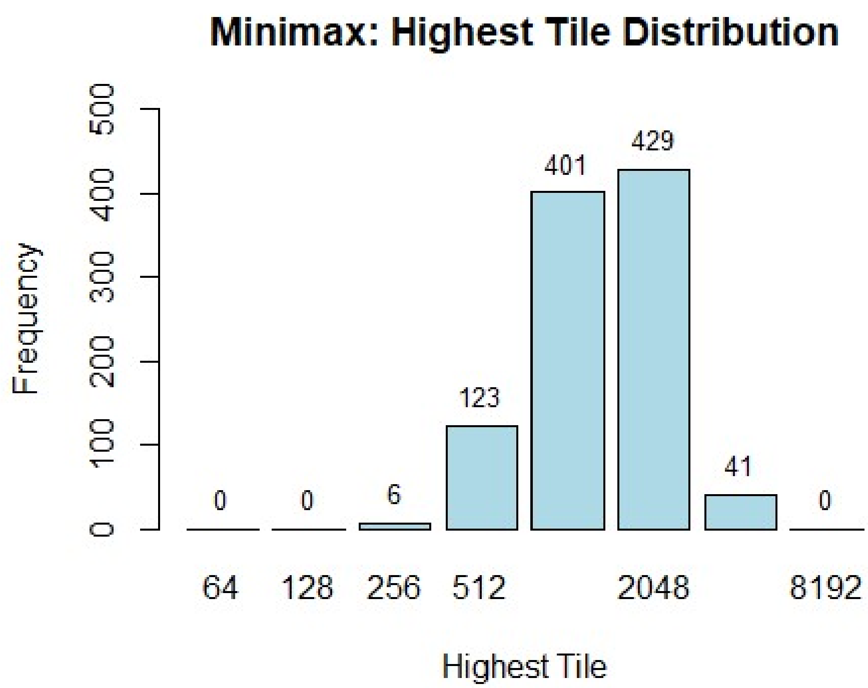



```r
unique(mm_data$Highest.Tile)
```

```
## [1] 1024 2048  512 4096  256
```

```r
mm_64 <- which(mm_data$Highest.Tile  == 64)
mm_128 <- which(mm_data$Highest.Tile == 128)
mm_256 <- which(mm_data$Highest.Tile == 256)
mm_512 <- which(mm_data$Highest.Tile == 512)
mm_1024 <- which(mm_data$Highest.Tile == 1024)
mm_2048 <- which(mm_data$Highest.Tile == 2048)
```

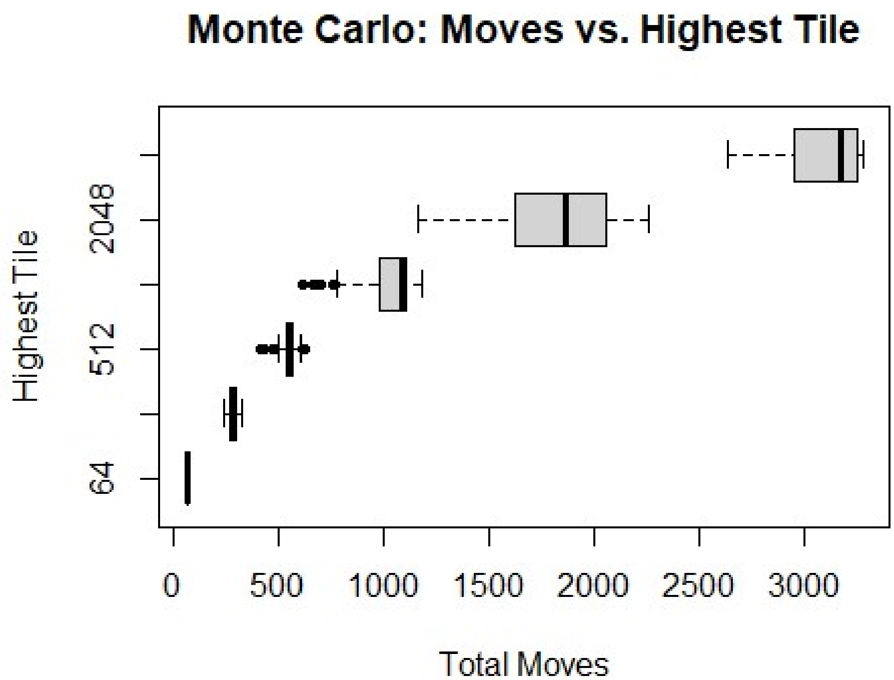
```
mm_4096 <- which(mm_data$Highest.Tile == 4096)
mm_8192 <- which(mm_data$Highest.Tile == 8192)
values <- c(64, 128, 256, 512, 1024, 2048, 4096, 8192)

mm_freq <- c(length(mm_64), length(mm_128), length(mm_256), length(mm_512),
length(mm_1024), length(mm_2048), length(mm_4096), length(mm_8192))

mm_tile_plot <- barplot(mm_freq, names.arg = values, col = "lightblue", main
= "Minimax: Highest Tile Distribution", xlab = "Highest Tile", ylab =
"Frequency", ylim = c(0, 500))

text(x = mm_tile_plot, y = mm_freq, labels = mm_freq, cex = 0.8, pos = 3)
```



Total Moves vs. Highest Tile

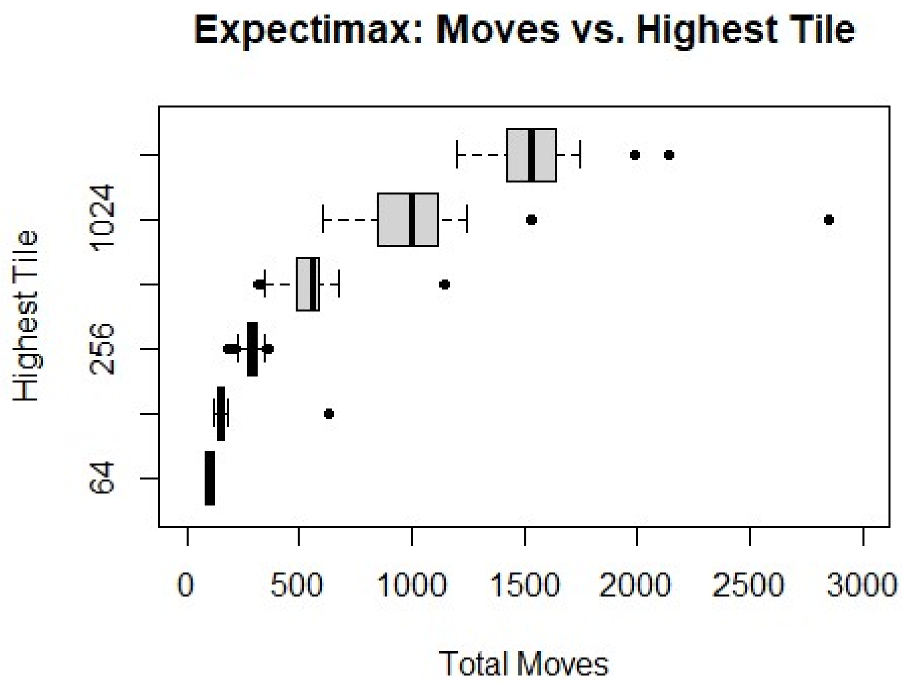
Monte Carlo Tree Search

```
boxplot(mc_data$Number.of.Moves ~ mc_data$Highest.Tile, horizontal = TRUE,
pch = 20, main = "Monte Carlo: Moves vs. Highest Tile", xlab = "Total Moves",
ylab = "Highest Tile")
```



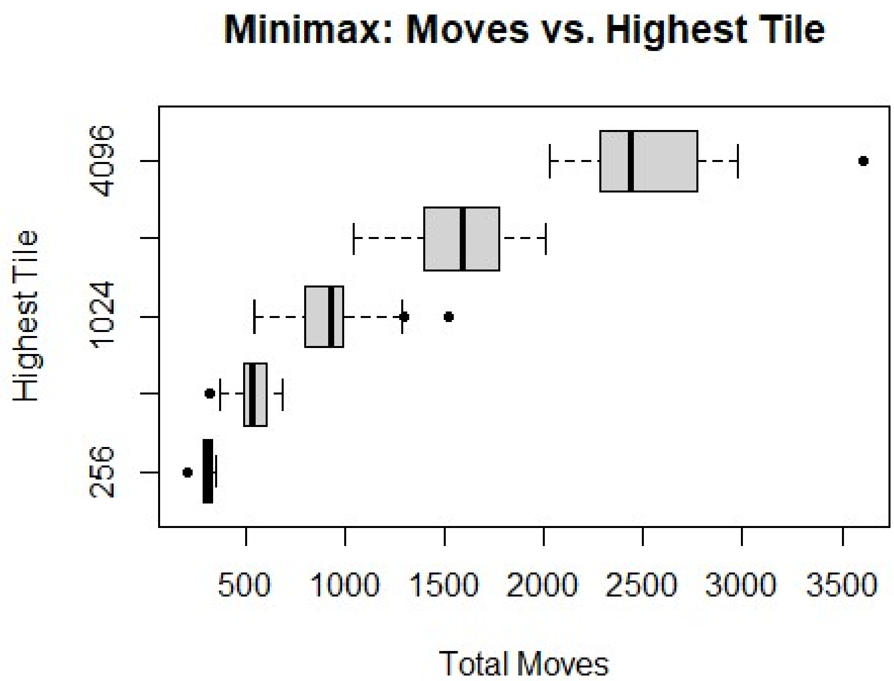
Expectimax

```
boxplot(em_data$Number.of.Moves ~ em_data$Highest.Tile, horizontal = TRUE,
pch = 20, main = "Expectimax: Moves vs. Highest Tile", xlab = "Total Moves",
ylab = "Highest Tile", ylim = c(0, 3000))
```



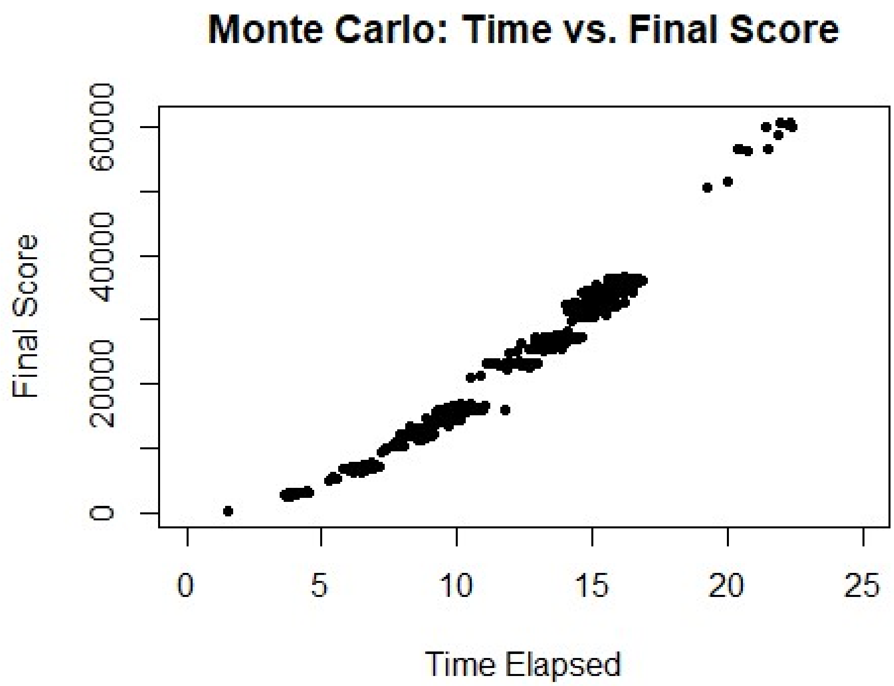
Minimax

```
boxplot(mm_data$Number.of.Moves ~ mm_data$Highest.Tile, horizontal = TRUE,
pch = 20, main = "Minimax: Moves vs. Highest Tile", xlab = "Total Moves",
ylab = "Highest Tile")
```



Time vs. Final Score

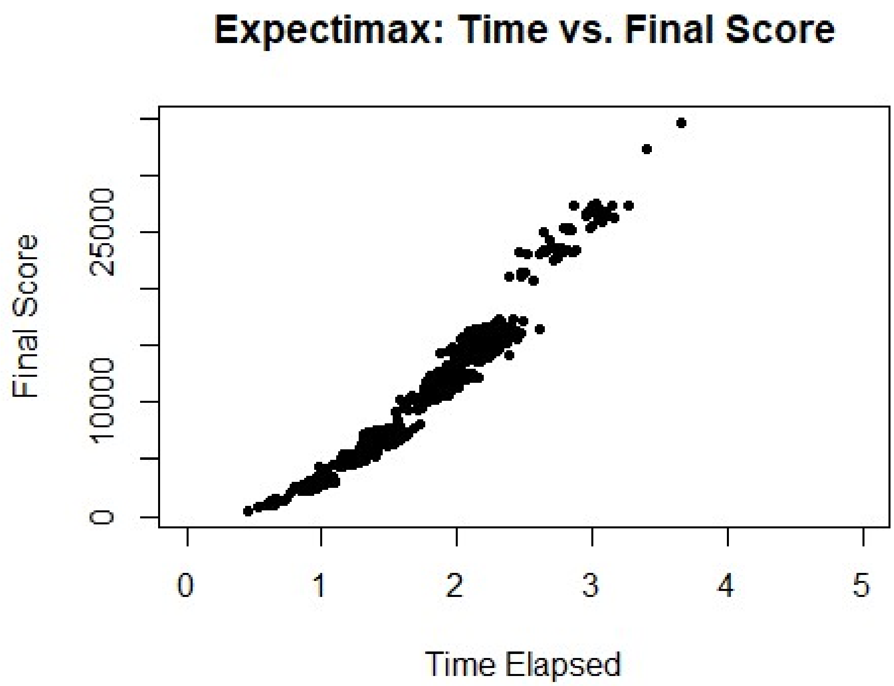
```
plot(mc_data$Time.Elapsed, mc_data$Final.Score, xlim = c(0, 25), pch = 20,
main = "Monte Carlo: Time vs. Final Score", xlab = "Time Elapsed", ylab =
"Final Score")
```



We cannot fit a linear regression model to these two variables as it appears to be non-linear (can maybe use log transformation) and heteroskedastic, meaning the variance in final score is not uniform as the time elapsed changes.

Expectimax

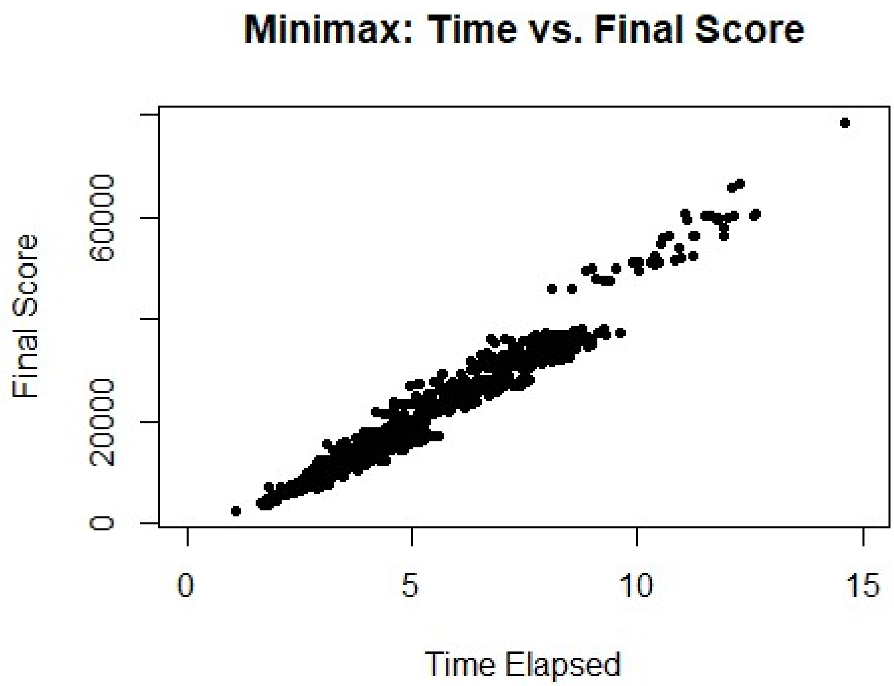
```
plot(em_data$Time.Elapsed, em_data$Final.Score, xlim = c(0, 5), pch = 20,
main = "Expectimax: Time vs. Final Score", xlab = "Time Elapsed", ylab =
"Final Score")
```



We cannot fit a linear regression model to these two variables as it appears to be non-linear (can maybe use log transformation) and heteroskedastic, meaning the variance in final score is not uniform as the time elapsed changes.

Minimax

```
plot(mm_data$Time.Elapsed, mm_data$Final.Score, xlim = c(0, 15), pch = 20,
main = "Minimax: Time vs. Final Score", xlab = "Time Elapsed", ylab = "Final
Score")
```



We cannot fit a linear regression model to these two variables. While it appears to be linear (can maybe use log transformation) and heteroskedastic, meaning the variance in final score is not uniform as the time elapsed changes.

```
# extract win percentages
mc_pctg <- length(which(mc_data$Win.Loss == "W")) / length(mc_data$Win.Loss)
em_pctg <- length(which(em_data$Win.Loss == "W")) / length(em_data$Win.Loss)
mm_pctg <- length(which(mm_data$Win.Loss == "W")) / length(mm_data$Win.Loss)
wins <- c(mc_pctg, em_pctg, mm_pctg)

# extract max scores
mc_max <- max(mc_score)
em_max <- max(em_score)
mm_max <- max(mm_score)
max <- c(mc_max, em_max, mm_max)

# extract median scores
mc_med <- median(mc_score)
em_med <- median(em_score)
mm_med <- median(mm_score)
medians <- c(mc_med, em_med, mm_med)

# extract mean scores
mc_avg <- mean(mc_score)
em_avg <- mean(em_score)
mm_avg <- mean(mm_score)
means <- c(mc_avg, em_avg, mm_avg)
```

```r
# slopes
mc_roc <- 18.54
em_roc <- 16.39
mm_roc <- 22.26
slope <- c(mc_roc, em_roc, mm_roc)

# standard deviations
mc_sd <- sd(mc_score)
em_sd <- sd(em_score)
mm_sd <- sd(mm_score)
sd <- c(mc_sd, em_sd, mm_sd)

algos <- c("Monte Carlo", "Expectimax", "Minimax")
data <- cbind(wins, max, medians, means, slope, sd)
colnames(data) <- c("Win %", "Highest Score", "Median Score", "Mean Score",
"Slope", "Standard Deviation")
rownames(data) <- algos
print(data)
```

```
##                   Win % Highest Score Median Score Mean Score Slope
## Monte Carlo 0.48397104         60628        16492   21900.44 18.54
## Expectimax  0.05289929         34620        11444   10921.70 16.39
## Minimax     0.47000000         78536        18048   22332.38 22.26
##             Standard Deviation
## Monte Carlo          10591.940
## Expectimax            5582.233
## Minimax              11548.080
```