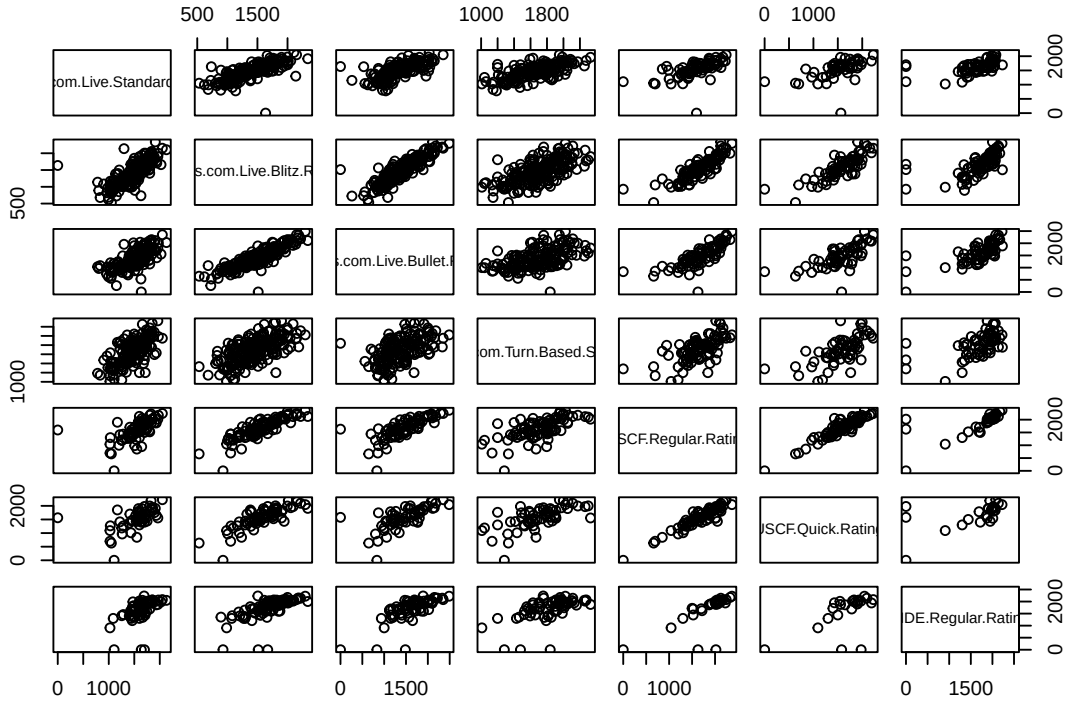
Homework 1

Alex Ojemann

2023-09-13

Problem 1



a.

b.

$\hat{y} = \beta_0 + \beta_1 * x$

```
## [1] "beta_0: 349.585 beta_1: 0.805"
```

```
## [1] "Confidence interval for beta_0: [-39.7649,738.934]"
```

```
## [1] "Confidence interval for beta_1: [0.558424,1.05195]"
```

The estimated beta_0 is not significantly different from 0, but the estimated beta_1 is significantly greater than 0.

c.

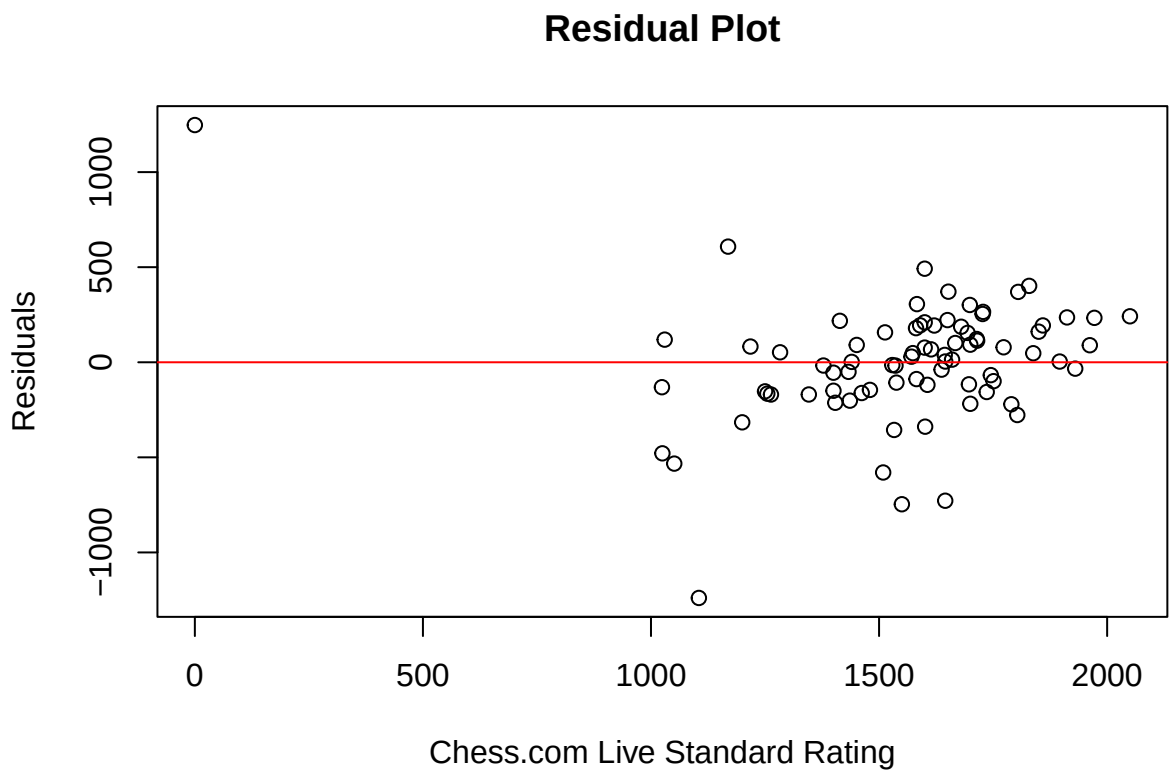
```
# Residual plot
residuals <- resid(model)
print(length(residuals))
```

```
## [1] 80
```

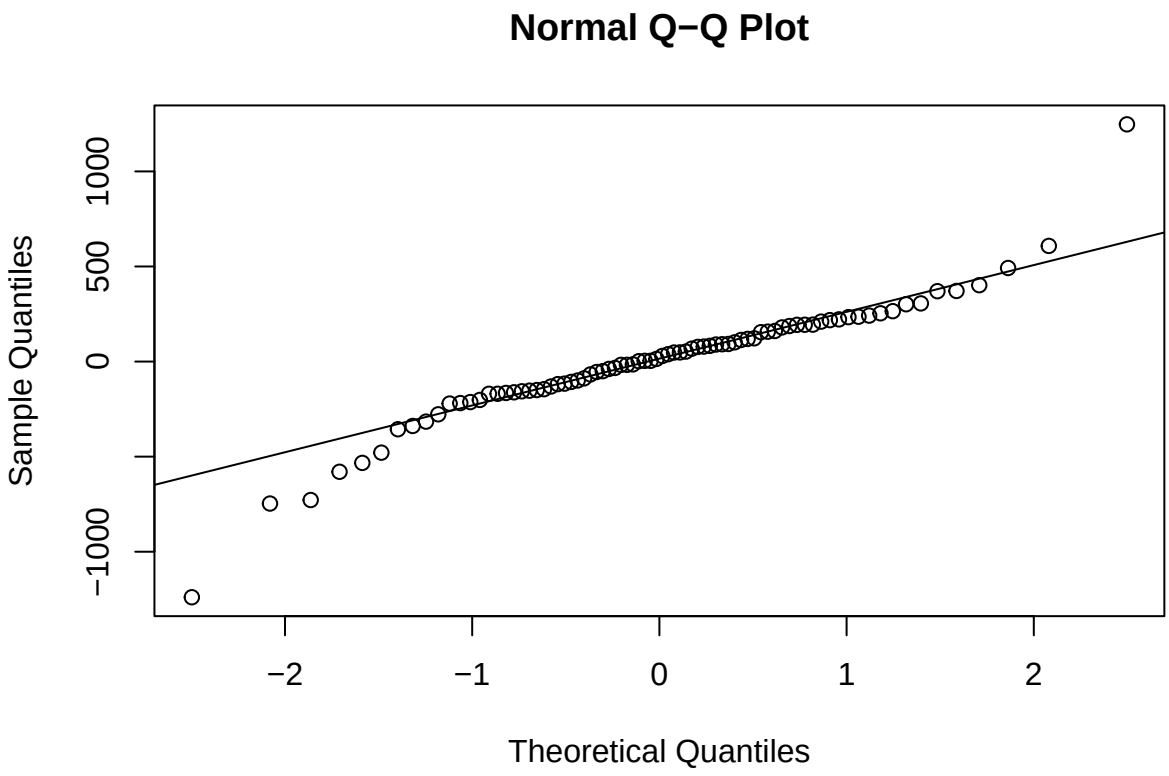
```
print(length(na.omit(filtered_data$Chess.com.Live.Standard.Rating)))
```

```
## [1] 80
```

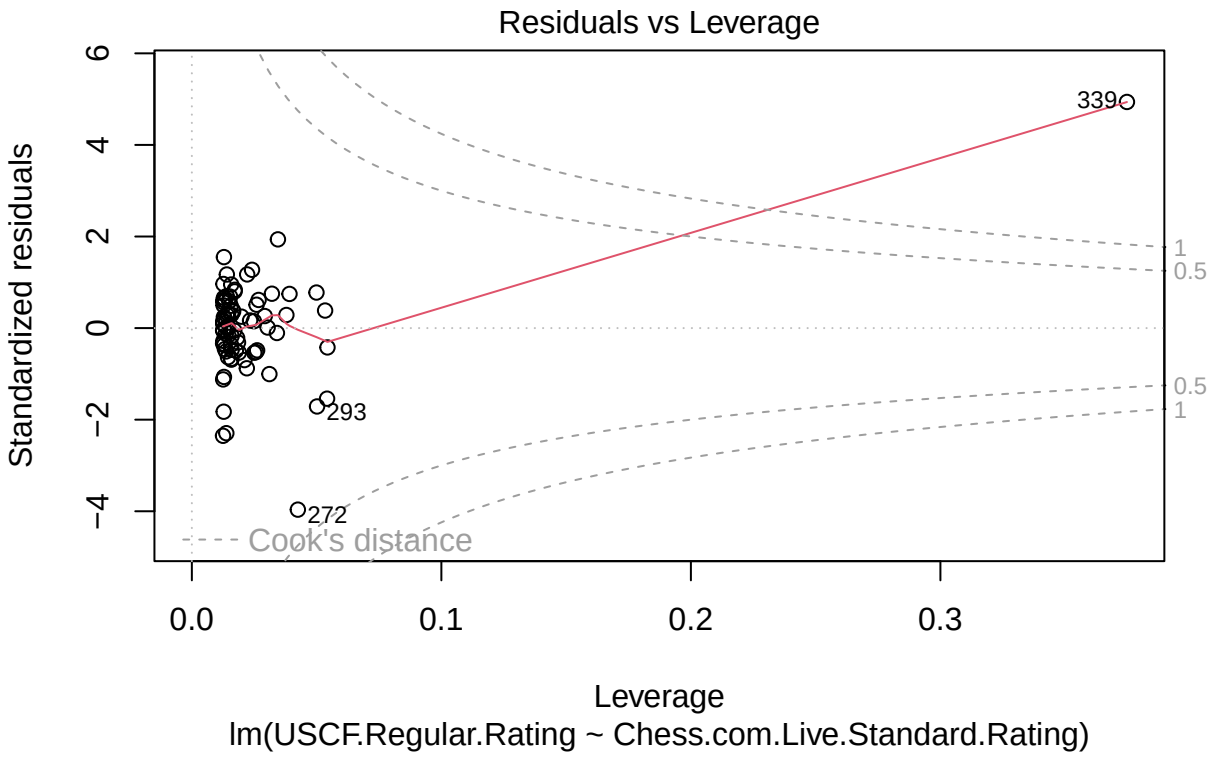
```
plot(filtered_data$Chess.com.Live.Standard.Rating, residuals, main = "Residual Plot", xlab = "Chess.com
abline(h = 0, col = "red")  # Add a horizontal line at y = 0
```




```
qqnorm(residuals)
qqline(residuals)
```




```
plot(model,which=5)
```



There is no noticeable trend or changing variance in the first residuals plot, so the linear model is adequate for the most part. There is some deviation from the expected quantiles based on a normal distribution, specifically that the tails are further from the median than expected. There is one very high leverage point that's also an outlier (339) that exceeds both the 0.5 and the 1 threshold of Cook's distance as seen in the third plot. In context this value makes no sense because the player has a chess.com score of 0 and the rating floor is said to be 100. 272 is also an outlier but does not exceed either of the Cook's distance thresholds because its leverage is much lower than 339.

d.

```
## [1] "beta_0: -398.11 beta_1: 1.27"
```

```
## [1] "Confidence interval for beta_0: [-809.068,12.8567]"
```

```
## [1] "Confidence interval for beta_1: [1.01226,1.52992]"
```

Once again, beta_0 is not significantly different from 0 but beta_1 is. The outlier likely occurred due to human error as it had a chess.com score of 0 which is not possible as they have a floor of 100. The outlier had an enormous influence on the model because it was the only x value anywhere close to the intercept and it deviated substantially from what a hypothetical y value given an x value of 0 would've been predicted to be by the rest of the data.

e. With the high leverage outlier included, we estimate that a hypothetical player with a 0 score on chess.com would have a 349.585 USCF rating on average and each unit increase in chess.com score results in an increase of 0.805 in USCF rating on average. We're 95% confident that the true population

intercept value lies between -39.7649 and 738.934 and the true population slope value lies between 0.558424 and 1.05195. Without the high leverage outlier included, we estimate that a hypothetical player with a 0 score on chess.com would have a -398.11 USCF rating on average and each unit increase in chess.com score results in an increase of 1.27 in USCF rating on average. We're 95% confident that the true population intercept value lies between -809.068 and 738.934 and the true population slope value lies between 1.01226 and 1.52992.

f.

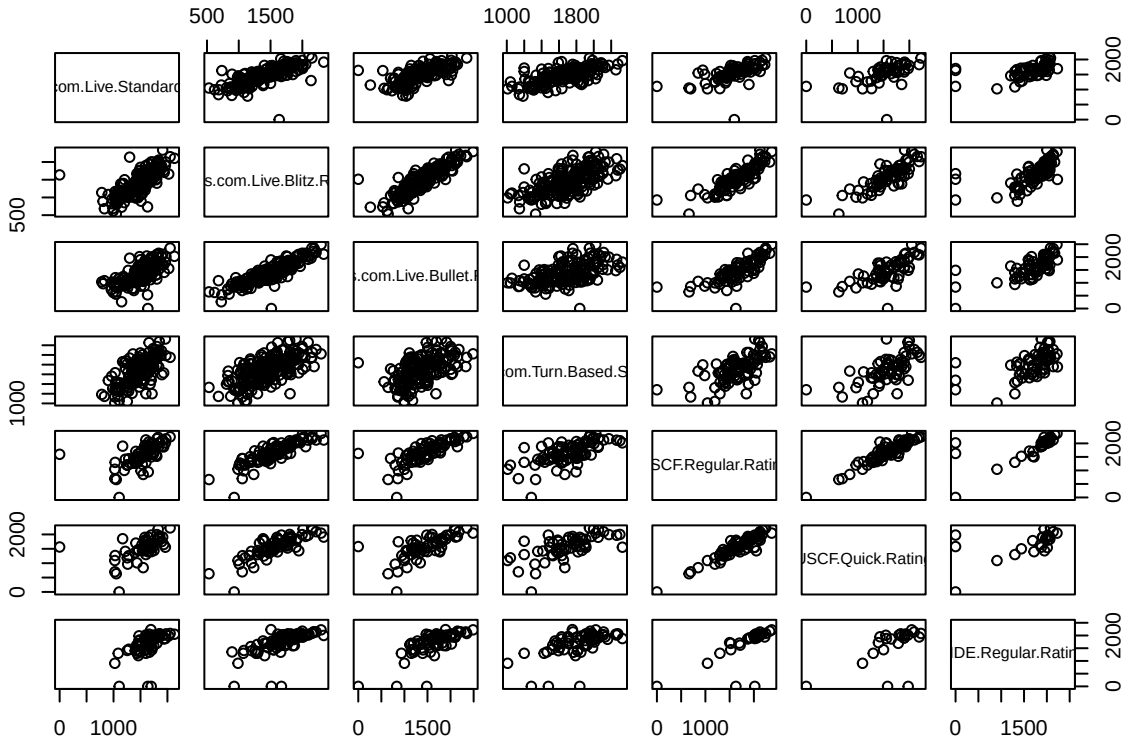
The average 1600 rated player on chess.com would have a 1637.89 USCF rating. The 95% confidence bounds for the model fit at that point are [1565.77,1710] while the 95% confidence bounds for the prediction at that point are [997.803,2277.97].

g.

I would use the R squared value to decide which other rating system best predicts USCF regular ratings because R squared shows which predictor eliminates the most variation in the response variable. The highest R squared value for a simple linear regression model to predict USCF regular ratings using one of the other ratings that aren't blitz or bullet is the USCF quick ratings with an R squared of 0.871. It's not clear to me whether you wanted me to include this, as quick is not regular but you say to only exclude blitz and bullet. If you only include other regular ratings, the best predictor is the FIDE regular rating with an R squared of 0.472.

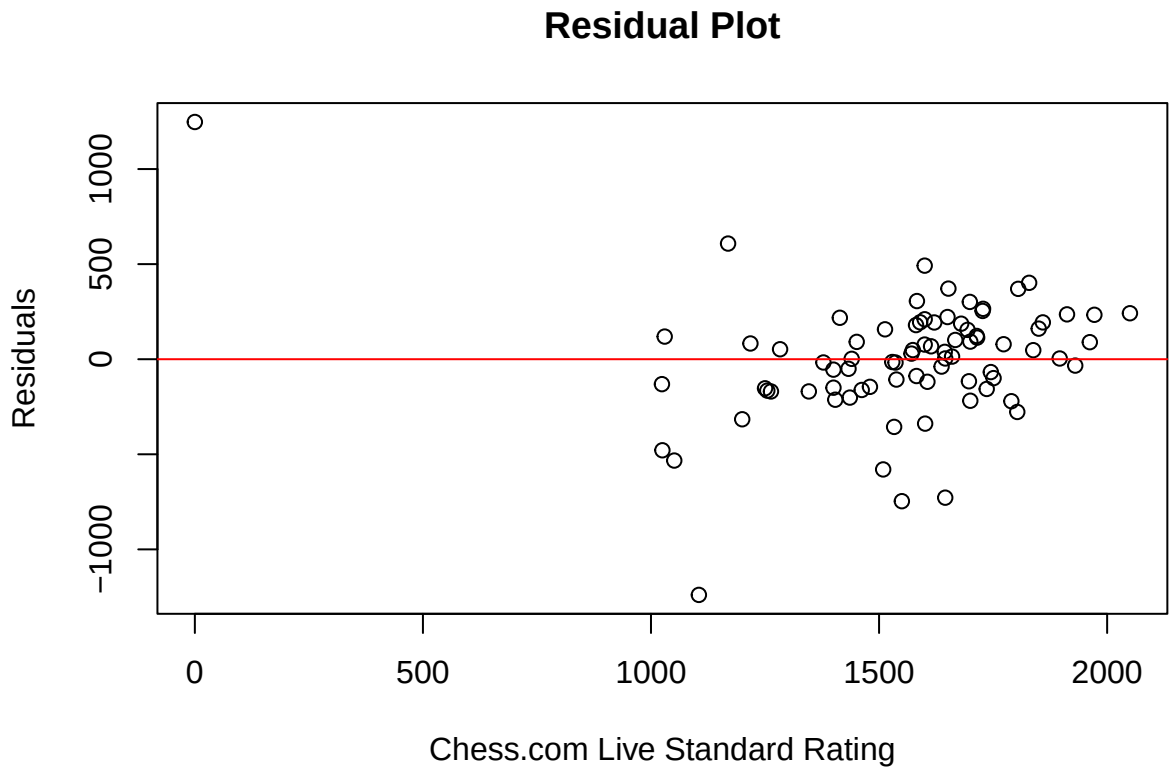
##Appendix (R code)

```
#Part a

df = read.csv('ChessRatingComparison.csv')
numeric_data <- df[, sapply(df, is.numeric)]
pairs(numeric_data)
```

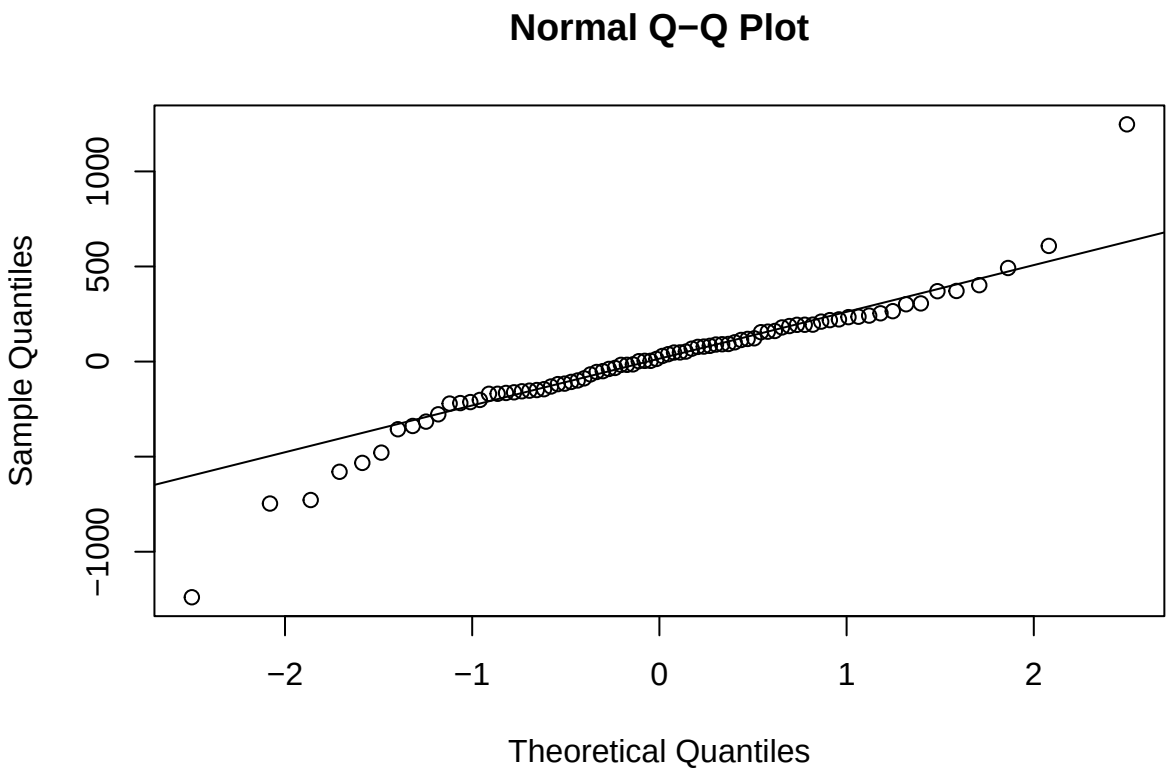



```
View(df)

#Part b

#NAs must be omitted in order to be able to plot residuals
#If not the number of residuals and number of x values don't #match
filtered_data = numeric_data[c("USCF.Regular.Rating","Chess.com.Live.Standard.Rating")]
filtered_data = na.omit(filtered_data)
model = lm(USCF.Regular.Rating ~ Chess.com.Live.Standard.Rating, filtered_data)
#summary(model) (Used to find coefficients)
print("beta_0: 349.585 beta_1: 0.805")
intervals = confint(model)
#Find interval bounds (next 4 lines)(wouldn't let me print them together)
#print(intervals[1,1])
#print(intervals[1,2])
#print(intervals[2,1])
#print(intervals[2,2])
print("Confidence interval for beta_0: [-39.7649,738.934]")
print("Confidence interval for beta_1: [0.558424,1.05195]")

#Part c

# Residual plot
residuals <- resid(model)
print(length(residuals))
print(length(na.omit(filtered_data$Chess.com.Live.Standard.Rating)))
```

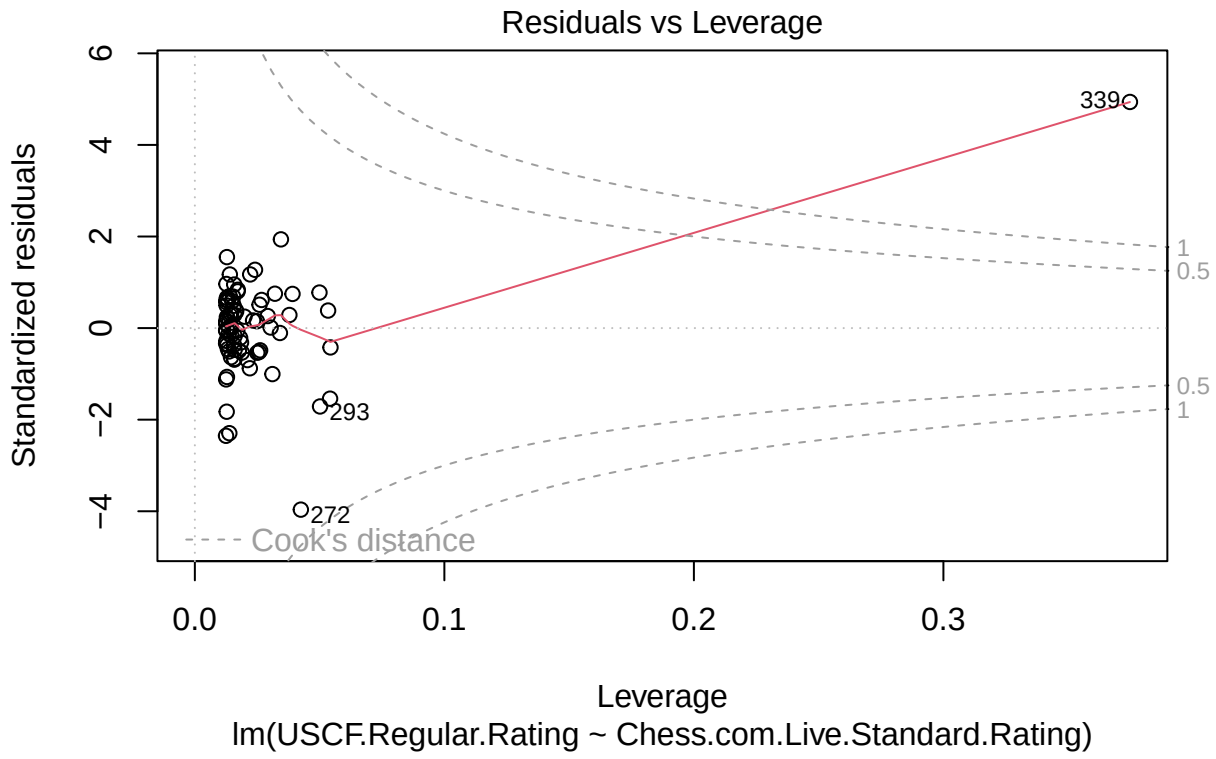
```r
plot(filtered_data$Chess.com.Live.Standard.Rating, residuals, main = "Residual Plot", xlab = "Chess.com
abline(h = 0, col = "red")  # Add a horizontal line at y = 0
```




```r
#Normal qq plot
qqnorm(residuals)
qqline(residuals)
```




```
#Residuals vs leverage with Cook's distance
plot(model,which=5)
```




```r
#Part d

#Removing the high leverage outlier
filtered_data2 = filtered_data[!(row.names(filtered_data) %in% c("339")),]

model2 = lm(USCF.Regular.Rating ~ Chess.com.Live.Standard.Rating, filtered_data2)
#summary(model2) (Used to find coefficients)
print("beta_0: -398.11 beta_1: 1.27")
intervals = confint(model2)
#Find interval bounds (next 4 lines)(wouldn't let me print them together)
#print(intervals[1,1])
#print(intervals[1,2])
#print(intervals[2,1])
#print(intervals[2,2])
print("Confidence interval for beta_0: [-809.068,12.8567]")
print("Confidence interval for beta_1: [1.01226,1.52992]")

#Part f

# 95% interval for the mean
predict(model,newdata=data.frame(Chess.com.Live.Standard.Rating=1600),interval="confidence")
# 95% prediction interval
predict(model,newdata=data.frame(Chess.com.Live.Standard.Rating=1600),interval="prediction")

#Part g
```

```r
model_clive = lm(USCF.Regular.Rating ~ Chess.com.Live.Standard.Rating, numeric_data)
summary(model_clive)

model_cturn = lm(USCF.Regular.Rating ~ Chess.com.Turn.Based.Standard, numeric_data)
summary(model_cturn)

model_uquick = lm(USCF.Regular.Rating ~ USCF.Quick.Rating, numeric_data)
summary(model_uquick)

model_fide = lm(USCF.Regular.Rating ~ FIDE.Regular.Rating, numeric_data)
summary(model_fide)
```