Training Linear Regression

We train a linear regression model to predict the number of moves left until the game is over.

```
board.data <- read.csv("./c4_generated.csv")
head(board.data)
```

```
##   pos_1 pos_2 pos_3 pos_4 pos_5 pos_6 pos_7 pos_8 pos_9 pos_10 pos_11 pos_12
## 1     1     1     1     2     2     1     1     2     2      1      1      2
## 2     2     0     2     2     1     1     1     1     0      2      0      0
## 3     1     2     1     2     1     2     1     2     2      1      0      1
## 4     0     2     0     0     0     1     2     0     1      0      0      0
## 5     2     1     2     1     1     2     1     2     2      1      2      2
## 6     2     1     1     2     1     2     1     1     2      1      1      2
##   pos_13 pos_14 pos_15 pos_16 pos_17 pos_18 pos_19 pos_20 pos_21 pos_22 pos_23
## 1      2      1      2      2      2      1      1      0      1      1      2
## 2      0      1      1      0      2      0      0      0      0      2      0
## 3      1      2      1      1      0      0      2      0      2      0      0
## 4      0      0      0      0      0      0      0      0      0      0      0
## 5      1      1      2      1      1      1      2      1      2      0      2
## 6      0      2      2      0      0      2      2      0      0      1      0
##   pos_24 pos_25 pos_26 pos_27 pos_28 pos_29 pos_30 pos_31 pos_32 pos_33 pos_34
## 1      2      1      2      0      2      2      1      1      0      1      0
## 2      0      0      0      0      0      0      0      0      0      0      0
## 3      0      0      0      0      0      0      0      0      0      0      0
## 4      0      0      0      0      0      0      0      0      0      0      0
## 5      0      2      0      0      1      0      2      0      1      0      0
## 6      0      1      1      0      0      0      0      0      0      0      0
##   pos_35 pos_36 pos_37 pos_38 pos_39 pos_40 pos_41 pos_42 score_1 score_2
## 1      1      2      0      2      0      2      0      1     100      -3
## 2      0      0      0      0      0      0      0      0     -15     -15
## 3      0      0      0      0      0      0      0      0     -12     -12
## 4      0      0      0      0      0      0      0      0      -2       2
## 5      0      0      1      0      2      0      0      0       7     100
## 6      0      0      0      0      0      0      0      0       2      11
##   score_3 score_4 score_5 score_6 score_7 boardValue movesLeft
## 1     100       4     100       4     100          2         1
## 2     -15     -15     -15     -15     -15          2         1
## 3     -12     -11     -12     -12     -12          1         2
## 4       3       3       0       2       0          1        17
## 5      -7     100       7      -7      -7          1         1
## 6      11       7       3      10       2          2         2
```

```
lm.fit <- lm(movesLeft~.,data=board.data)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = movesLeft ~ ., data = board.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.656  -4.008  -0.046   3.821  16.872
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 28.141027   0.439850  63.979  < 2e-16 ***
## pos_1       -0.955696   0.122288  -7.815 6.45e-15 ***
## pos_2       -0.995060   0.120541  -8.255  < 2e-16 ***
## pos_3       -1.009654   0.120009  -8.413  < 2e-16 ***
## pos_4       -0.600941   0.122039  -4.924 8.70e-07 ***
## pos_5       -0.628998   0.121697  -5.169 2.44e-07 ***
## pos_6       -0.821801   0.122326  -6.718 2.01e-11 ***
## pos_7       -1.080030   0.122914  -8.787  < 2e-16 ***
## pos_8       -0.434036   0.114724  -3.783 0.000156 ***
## pos_9       -0.695378   0.115439  -6.024 1.81e-09 ***
## pos_10      -0.724866   0.115842  -6.257 4.19e-10 ***
## pos_11      -1.097423   0.115020  -9.541  < 2e-16 ***
## pos_12      -0.838945   0.116182  -7.221 5.81e-13 ***
## pos_13      -0.812157   0.116154  -6.992 3.01e-12 ***
## pos_14      -0.492199   0.115401  -4.265 2.03e-05 ***
## pos_15      -0.865315   0.134852  -6.417 1.50e-10 ***
## pos_16      -0.783236   0.136146  -5.753 9.21e-09 ***
## pos_17      -0.732066   0.133943  -5.466 4.80e-08 ***
## pos_18      -0.695552   0.135181  -5.145 2.76e-07 ***
## pos_19      -0.547723   0.136350  -4.017 5.97e-05 ***
## pos_20      -0.622684   0.133698  -4.657 3.27e-06 ***
## pos_21      -0.661433   0.132439  -4.994 6.08e-07 ***
## pos_22       0.093242   0.172259   0.541 0.588327
## pos_23      -0.219851   0.176627  -1.245 0.213285
## pos_24      -0.145539   0.174966  -0.832 0.405548
## pos_25      -0.233880   0.178315  -1.312 0.189702
## pos_26      -0.112289   0.176919  -0.635 0.525654
## pos_27      -0.013076   0.170881  -0.077 0.939005
## pos_28      -0.088576   0.171840  -0.515 0.606254
## pos_29       0.118728   0.244484   0.486 0.627251
## pos_30       0.017092   0.244392   0.070 0.944245
## pos_31       0.186851   0.243730   0.767 0.443332
## pos_32       0.219238   0.249568   0.878 0.379725
## pos_33      -0.027119   0.234447  -0.116 0.907917
## pos_34       0.191233   0.231219   0.827 0.408233
## pos_35       0.234658   0.243541   0.964 0.335323
## pos_36       0.710612   0.592348   1.200 0.230322
## pos_37       1.265312   0.569633   2.221 0.026370 *
## pos_38       1.562912   0.575112   2.718 0.006595 **
## pos_39       3.212977   0.562619   5.711 1.18e-08 ***
## pos_40       2.197948   0.559743   3.927 8.71e-05 ***
## pos_41       2.456662   0.568679   4.320 1.59e-05 ***
## pos_42       1.236192   0.551733   2.241 0.025091 *
## score_1     -0.005430   0.008926  -0.608 0.542937
## score_2     -0.015675   0.008771  -1.787 0.073985 .
## score_3     -0.025078   0.008532  -2.939 0.003304 **
## score_4     -0.053663   0.008389  -6.397 1.70e-10 ***
## score_5     -0.024896   0.008314  -2.994 0.002762 **
## score_6     -0.039555   0.008563  -4.619 3.93e-06 ***
## score_7     -0.019389   0.008556  -2.266 0.023483 *
## boardValue  -4.190657   0.135199 -30.996  < 2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.188 on 5943 degrees of freedom
## Multiple R-squared:  0.4703, Adjusted R-squared:  0.4658
## F-statistic: 105.5 on 50 and 5943 DF,  p-value: < 2.2e-16
```

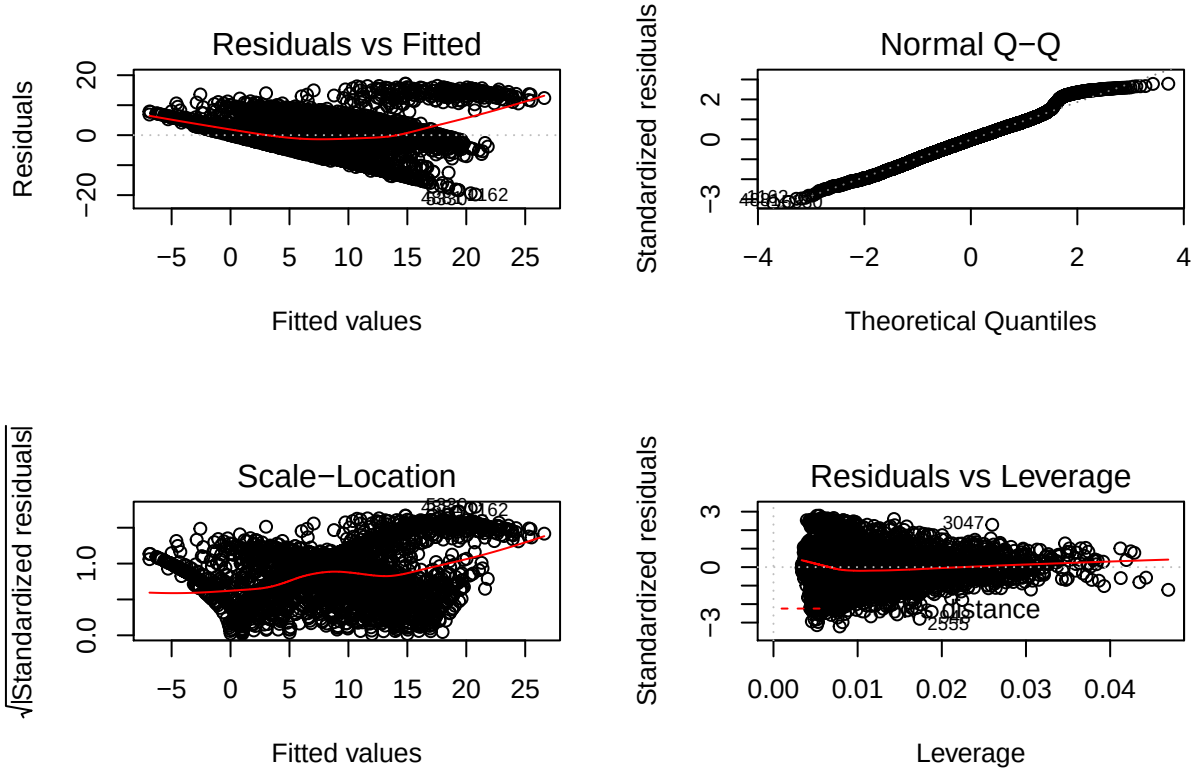
Tried training the linear regression model on the board position and the board value to predict the number of moves left for the game to end assuming perfect play. Let us now do the train/test split cross validation to get an estimate of the train and the test MSEs.

```
idx <- sample(nrow(board.data),0.8*nrow(board.data),replace = FALSE)
board.data.train <- board.data[idx,]
board.data.test <- board.data[-idx,]
lm.fit <- lm(movesLeft~.,data=board.data.train)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = movesLeft ~ ., data = board.data.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.7726  -4.0050  -0.0388   3.8137  17.1747
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 28.545557   0.491209  58.113  < 2e-16 ***
## pos_1       -0.966364   0.136375  -7.086 1.58e-12 ***
## pos_2       -1.021688   0.134448  -7.599 3.57e-14 ***
## pos_3       -1.067305   0.133540  -7.992 1.65e-15 ***
## pos_4       -0.711700   0.136091  -5.230 1.77e-07 ***
## pos_5       -0.684987   0.135578  -5.052 4.53e-07 ***
## pos_6       -0.879229   0.137164  -6.410 1.60e-10 ***
## pos_7       -1.195762   0.136938  -8.732  < 2e-16 ***
## pos_8       -0.394720   0.129015  -3.059 0.002230 **
## pos_9       -0.711194   0.129620  -5.487 4.31e-08 ***
## pos_10      -0.777911   0.129036  -6.029 1.78e-09 ***
## pos_11      -1.072319   0.129015  -8.312  < 2e-16 ***
## pos_12      -0.919403   0.130343  -7.054 1.99e-12 ***
## pos_13      -0.770385   0.130101  -5.921 3.42e-09 ***
## pos_14      -0.507632   0.128446  -3.952 7.86e-05 ***
## pos_15      -0.910612   0.152271  -5.980 2.39e-09 ***
## pos_16      -0.724114   0.152850  -4.737 2.23e-06 ***
## pos_17      -0.701822   0.152221  -4.611 4.12e-06 ***
## pos_18      -0.742357   0.149839  -4.954 7.51e-07 ***
## pos_19      -0.511664   0.152947  -3.345 0.000828 ***
## pos_20      -0.648560   0.149475  -4.339 1.46e-05 ***
## pos_21      -0.623292   0.147453  -4.227 2.41e-05 ***
## pos_22       0.150089   0.193757   0.775 0.438602
## pos_23      -0.199942   0.195196  -1.024 0.305738
## pos_24      -0.089844   0.200083  -0.449 0.653427
## pos_25      -0.130148   0.201052  -0.647 0.517448
## pos_26      -0.059761   0.199229  -0.300 0.764219
## pos_27       0.058391   0.194815   0.300 0.764399
## pos_28      -0.034668   0.195648  -0.177 0.859362
```

```
## pos_29       0.118682   0.273354   0.434 0.664184
## pos_30      -0.095166   0.276913  -0.344 0.731111
## pos_31       0.224369   0.275986   0.813 0.416274
## pos_32       0.027305   0.274909   0.099 0.920885
## pos_33       0.100580   0.259895   0.387 0.698774
## pos_34       0.268376   0.261855   1.025 0.305461
## pos_35       0.271191   0.275093   0.986 0.324273
## pos_36       0.789506   0.660752   1.195 0.232202
## pos_37       1.027025   0.644138   1.594 0.110909
## pos_38       1.334874   0.631713   2.113 0.034644 *
## pos_39       3.652122   0.633413   5.766 8.64e-09 ***
## pos_40       2.168683   0.624298   3.474 0.000518 ***
## pos_41       2.400245   0.638190   3.761 0.000171 ***
## pos_42       1.271156   0.618247   2.056 0.039831 *
## score_1     -0.005992   0.010016  -0.598 0.549693
## score_2     -0.010150   0.009927  -1.022 0.306613
## score_3     -0.021285   0.009369  -2.272 0.023143 *
## score_4     -0.060959   0.009547  -6.385 1.88e-10 ***
## score_5     -0.024875   0.009279  -2.681 0.007371 **
## score_6     -0.041140   0.009545  -4.310 1.67e-05 ***
## score_7     -0.023215   0.009558  -2.429 0.015190 *
## boardValue  -4.173131   0.151457 -27.553  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.187 on 4744 degrees of freedom
## Multiple R-squared:  0.4721, Adjusted R-squared:  0.4665
## F-statistic: 84.84 on 50 and 4744 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(lm.fit)
```




```
predictions <- predict(lm.fit,board.data.train)
cat("Train MSE = ",mean((predictions-board.data.train$movesLeft)^2),"\n")
```

```
## Train MSE =  37.86608
```

```
predictions <- predict(lm.fit,board.data.test)
cat("Test MSE = ",mean((predictions-board.data.test$movesLeft)^2))
```

```
## Test MSE =  38.66618
```

Looking at the Residuals vs Fitted, it seems like our linear regression model does not abide the constant variance assumption. It is not a very accurate model for prediction in this case.