q1_final_project

2025-04-18

1)

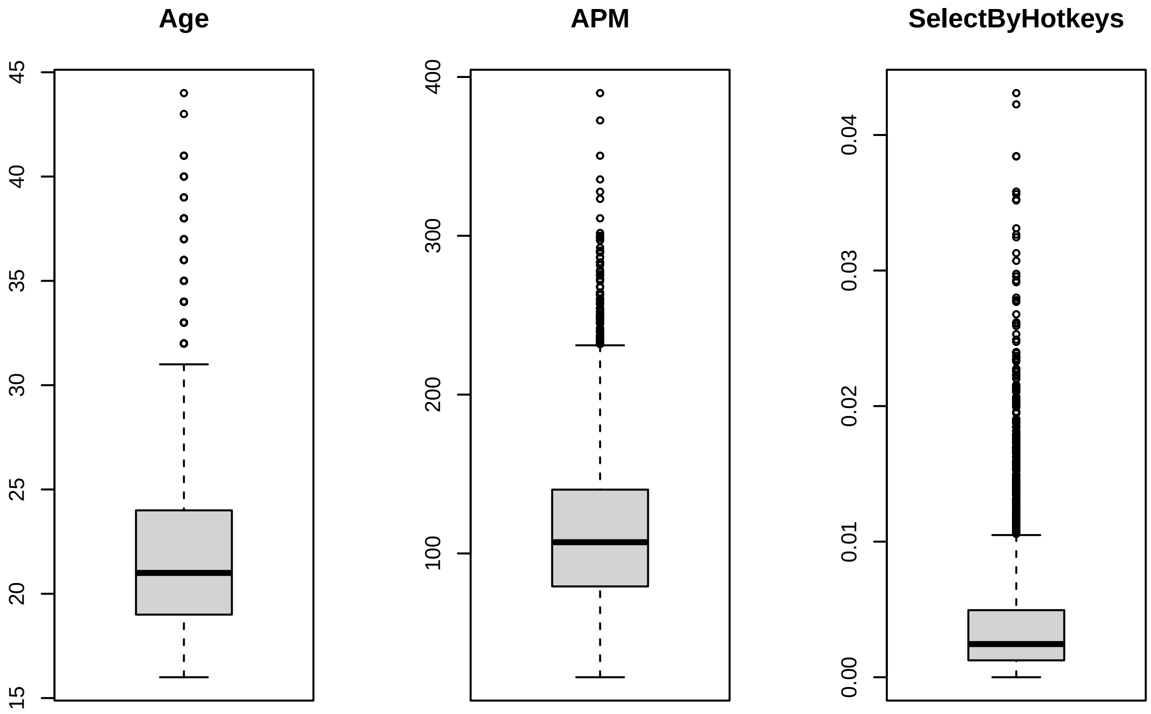
```
# read data
starcraft = read.table("starcraft.txt", header=TRUE)
starcraft$LeagueIndex = factor(starcraft$LeagueIndex)
head(starcraft)
```

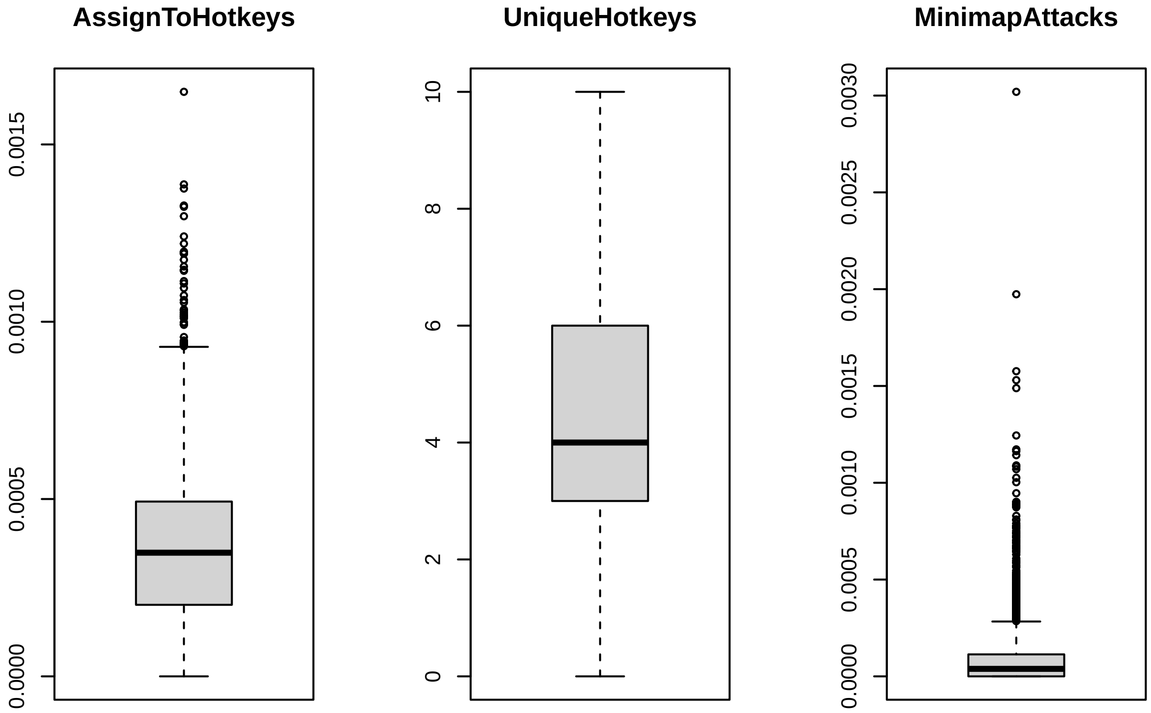
```
##   LeagueIndex Age TotalHours      APM SelectByHotkeys AssignToHotkeys
## 1           5  27       3000 143.7180     0.003515159     0.000219697
## 2           5  23       5000 129.2322     0.003303812     0.000259462
## 3           4  30        200  69.9612     0.001101091     0.000335571
## 4           3  19        400 107.6016     0.001033542     0.000213102
## 5           3  32        500 122.8908     0.001136014     0.000327326
## 6           2  27         70  44.4570     0.000978390     0.000255232
##   UniqueHotkeys MinimapAttacks MinimapRightClicks NumberOfPACs GapBetweenPACs
## 1             7    0.000109849        0.000392317  0.004849037        32.6677
## 2             4    0.000294057        0.000432436  0.004307064        32.9194
## 3             4    0.000293624        0.000461409  0.002925755        44.6475
## 4             1    0.000053300        0.000543409  0.003782551        29.2203
## 5             2    0.000000000        0.001328558  0.002368299        22.6885
## 6             2    0.000000000        0.000000000  0.002424707        76.4405
##   ActionLatency ActionsInPAC TotalMapExplored WorkersMade UniqueUnitsMade
## 1       40.8673       4.7508               28  0.00139660               6
## 2       42.3454       4.8434               22  0.00119350               5
## 3       75.3548       4.0430               22  0.00074455               6
## 4       53.7352       4.9155               19  0.00042620               7
## 5       62.0813       9.3740               15  0.00117450               4
## 6       98.7719       3.0965               16  0.00037221               6
##   ComplexUnitsMade ComplexAbilitiesUsed
## 1                0           0.00000000
## 2                0           0.00020757
## 3                0           0.00018876
## 4                0           0.00038358
## 5                0           0.00001930
## 6                0           0.00000000
```

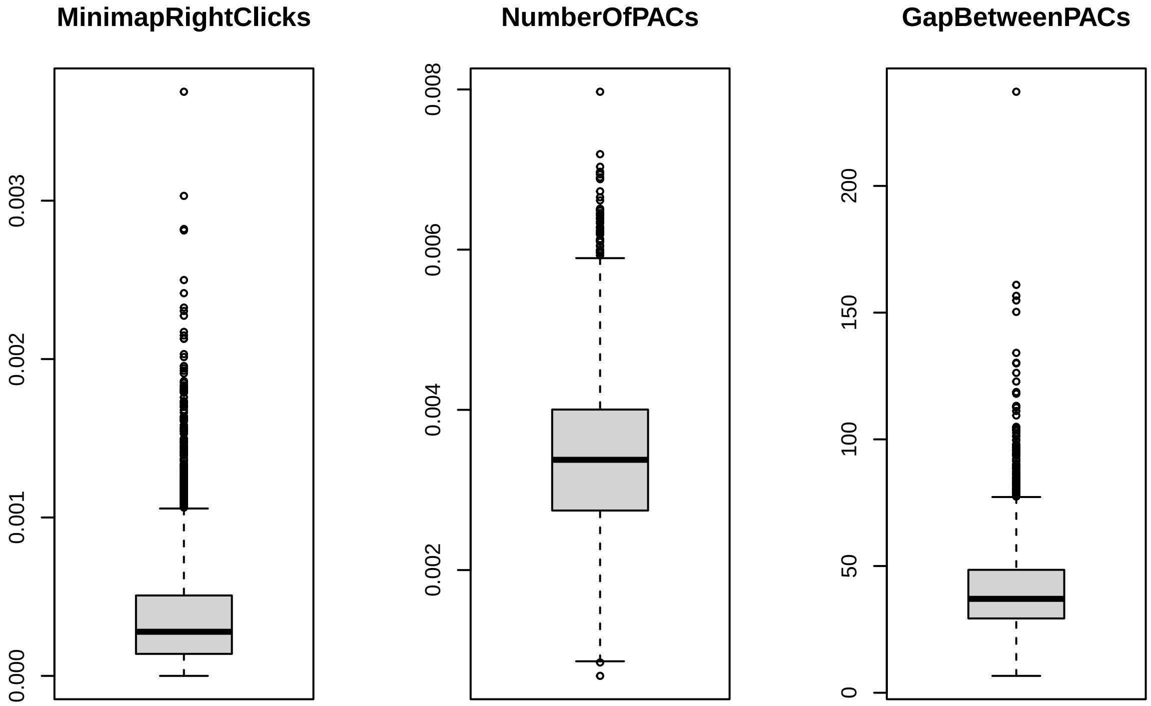
We first examine the boxplots of all features except the response variable and *LeagueIndex*, which is a qualitative variable.

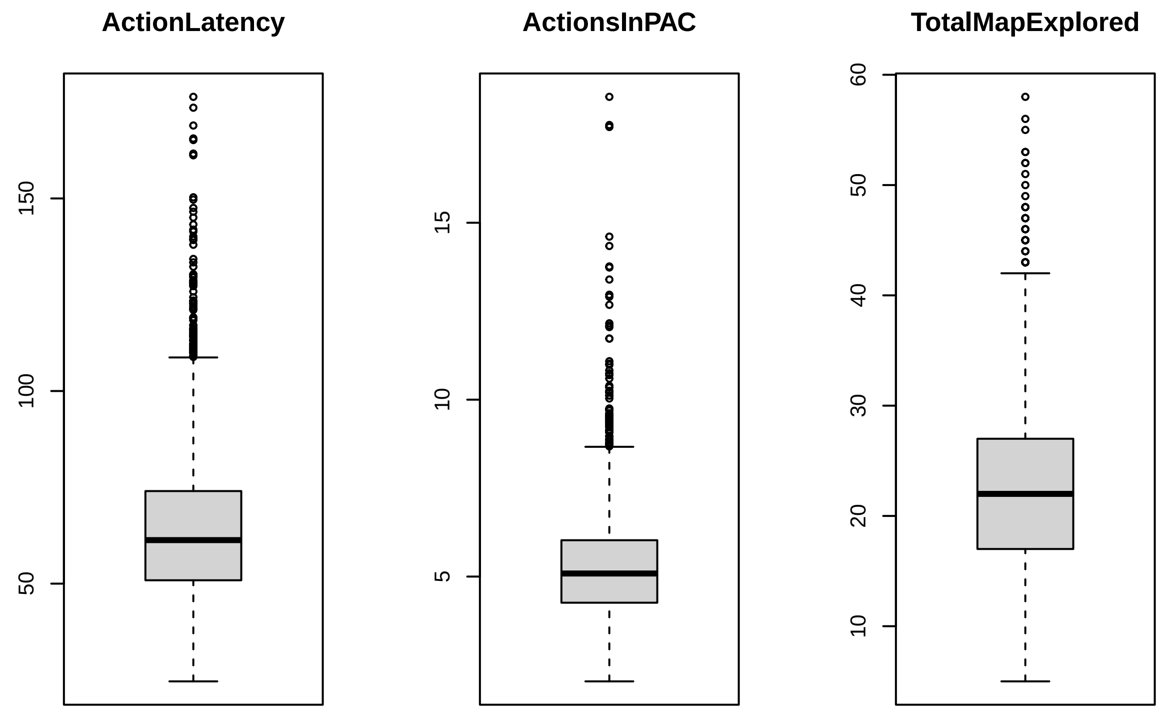
```
# boxplot of features (except response variable)
par(mfrow=c(1,3))
```

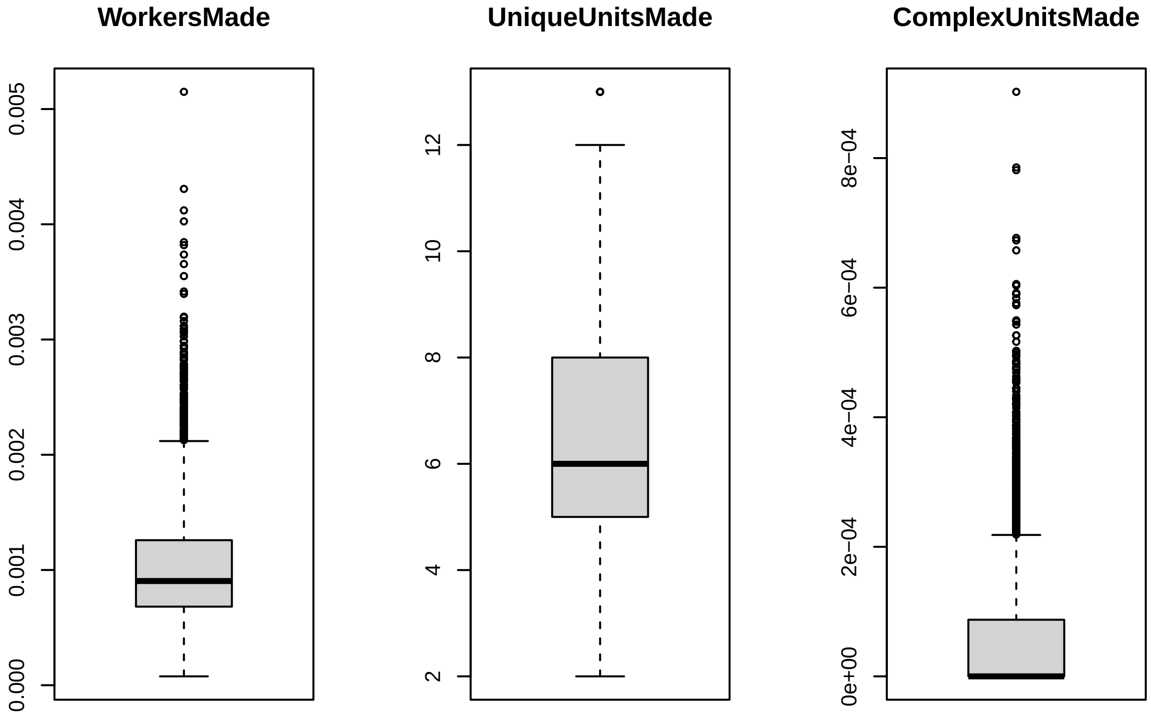
```r
for (k in colnames(starcraft)[-c(1,3)]) {
  boxplot(starcraft[k], main = k)
}
```

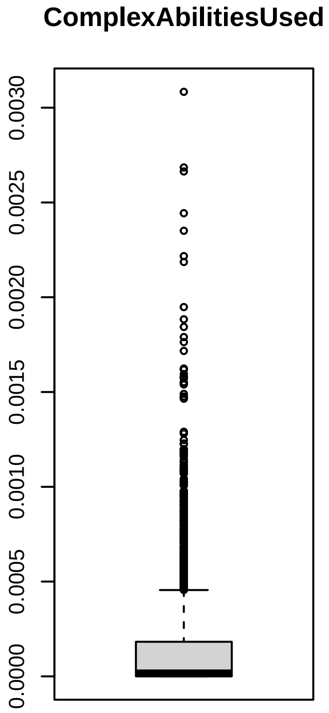



AssignToHotkeys
0.0000
0.0005
0.0010
0.0015
UniqueHotkeys
0
2
4
6
8
10
MinimapAttacks
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
0.0030


MinimapRightClicks
0.000
0.001
0.002
0.003
NumberOfPACs
0.002
0.004
0.006
0.008
GapBetweenPACs
0
50
100
150
200


ActionLatency
50
100
150
ActionsInPAC
5
10
15
TotalMapExplored
10
20
30
40
50
60


WorkersMade
0.000
0.001
0.002
0.003
0.004
0.005
UniqueUnitsMade
2
4
6
8
10
12
ComplexUnitsMade
0e+00
2e−04
4e−04
6e−04
8e−04



We notice that all covariates except $UniqueHotkeys$ and $UniqueUnitsMade$ display skewness. We determined which transformation, between square root and log, suits each covariate the best and applied them.

```
#names = colnames(starcraft)[-c(1,3,7,16)]
#par(mfrow=c(1,3))

#for (k in names) {
#  boxplot((starcraft[k]), main = k)
#}

#for (k in names){
#  boxplot(sqrt(starcraft[k]))
#}

#for (k in names) {
#  boxplot(log(starcraft[k]), main = k)
#}
```

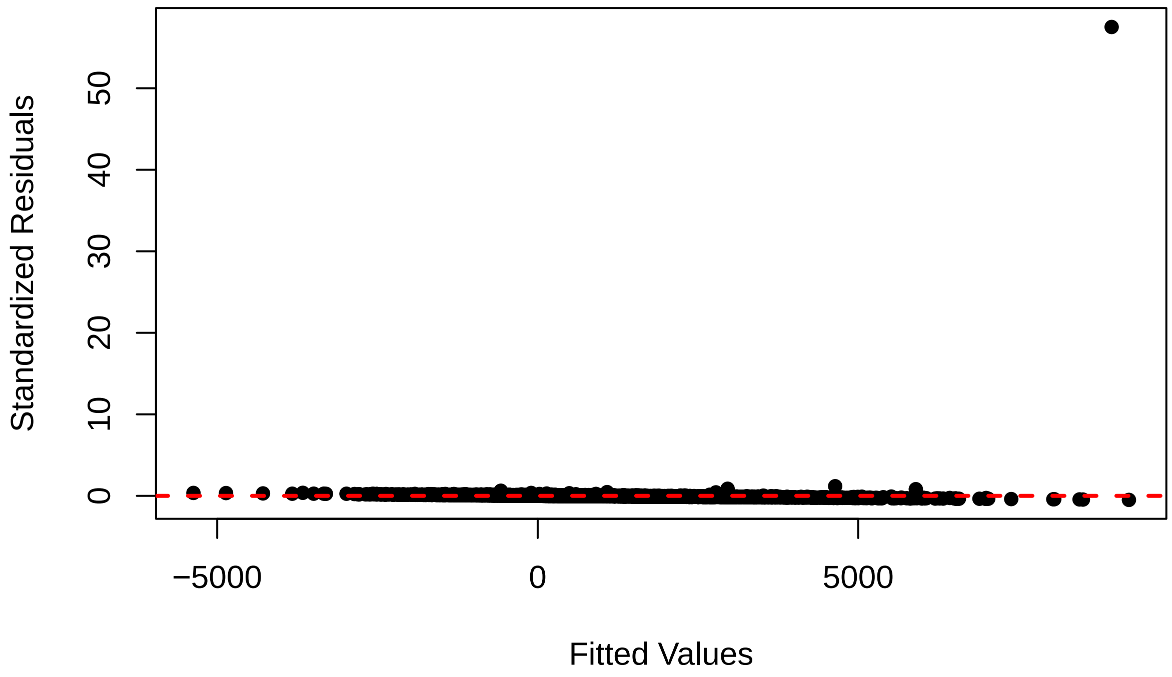
```
# apply transformations to features (except response variable and LeagueIndex)

for (name in c("NumberOfPACs", "AssignToHotkeys", "SelectByHotkeys",
               "MinimapAttacks", "TotalMapExplored", "MinimapRightClicks")){
  starcraft[name] = sqrt(starcraft[name])
}

for (name in c("APM", "GapBetweenPACs", "ActionLatency", "Age",
               "ActionsInPAC", "WorkersMade")){
  starcraft[name] = log(starcraft[name])
}
```

```
# fit a full model with transformed variables
full = lm(TotalHours ~ ., starcraft)
summary(full)
```

```
##
## Call:
## lm(formula = TotalHours ~ ., data = starcraft)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -8494  -1234   -200    694 991046
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.613e+04  2.502e+04   1.044 0.296561
## LeagueIndex2         -7.275e+01  1.661e+03  -0.044 0.965056
## LeagueIndex3         -1.776e+01  1.617e+03  -0.011 0.991239
## LeagueIndex4         -3.186e+02  1.639e+03  -0.194 0.845848
## LeagueIndex5          3.339e+02  1.750e+03   0.191 0.848658
## LeagueIndex6         -1.723e+03  1.901e+03  -0.906 0.364835
## LeagueIndex7         -2.406e+03  3.578e+03  -0.673 0.501255
## Age                  -5.728e+02  1.723e+03  -0.332 0.739535
## APM                  -1.412e+04  5.797e+03  -2.436 0.014901 *
## SelectByHotkeys       1.224e+05  3.686e+04   3.322 0.000903 ***
## AssignToHotkeys      -1.490e+04  7.269e+04  -0.205 0.837556
## UniqueHotkeys        -1.105e+02  1.487e+02  -0.743 0.457394
## MinimapAttacks       -1.575e+04  5.244e+04  -0.300 0.763955
## MinimapRightClicks    1.084e+04  4.073e+04   0.266 0.790151
## NumberOfPACs          4.205e+05  2.046e+05   2.055 0.039941 *
## GapBetweenPACs        6.307e+02  1.214e+03   0.519 0.603452
## ActionLatency        -2.736e+03  3.172e+03  -0.863 0.388408
## ActionsInPAC          1.205e+04  5.151e+03   2.339 0.019383 *
## TotalMapExplored      4.508e+02  5.348e+02   0.843 0.399337
## WorkersMade          -8.962e+01  7.641e+02  -0.117 0.906637
## UniqueUnitsMade      -1.379e+02  2.109e+02  -0.654 0.513368
## ComplexUnitsMade     -2.202e+06  3.630e+06  -0.607 0.544177
## ComplexAbilitiesUsed -1.056e+06  1.467e+06  -0.720 0.471685
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 17300 on 3315 degrees of freedom
## Multiple R-squared:  0.008627,   Adjusted R-squared:  0.002048
## F-statistic: 1.311 on 22 and 3315 DF,  p-value: 0.1504
```

We then plot the fitted values against standardized residuals.

```
# fitted values vs standard residuals plot
plot(full$fitted.values, rstandard(full), xlab = "Fitted Values",
ylab = "Standardized Residuals", pch = 16)
abline(h = 0, lty = 2, lwd = 2, col="red")
```

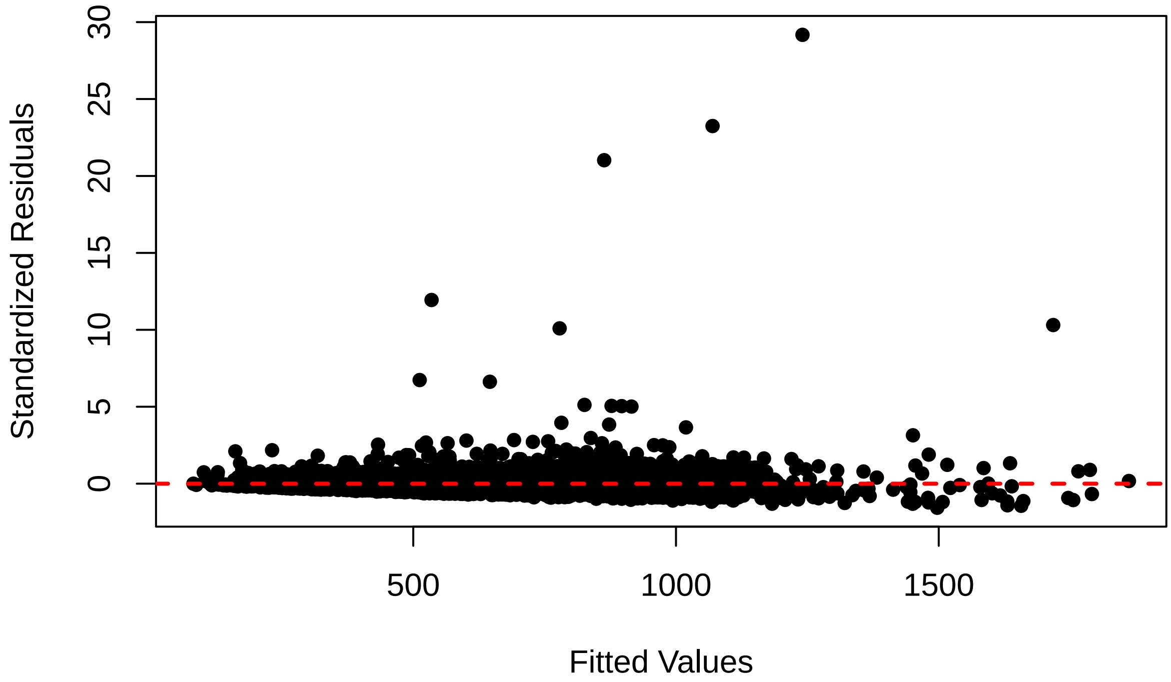


The fitted values vs. standardized residuals plot show that there is an extreme outlier. We fit another model without the outlier and examine the fitted values vs. standardized residuals plot.

```
# remove outlier and fit a different model
which.max(rstandard(full))
```

```
## 1793
## 1793
```

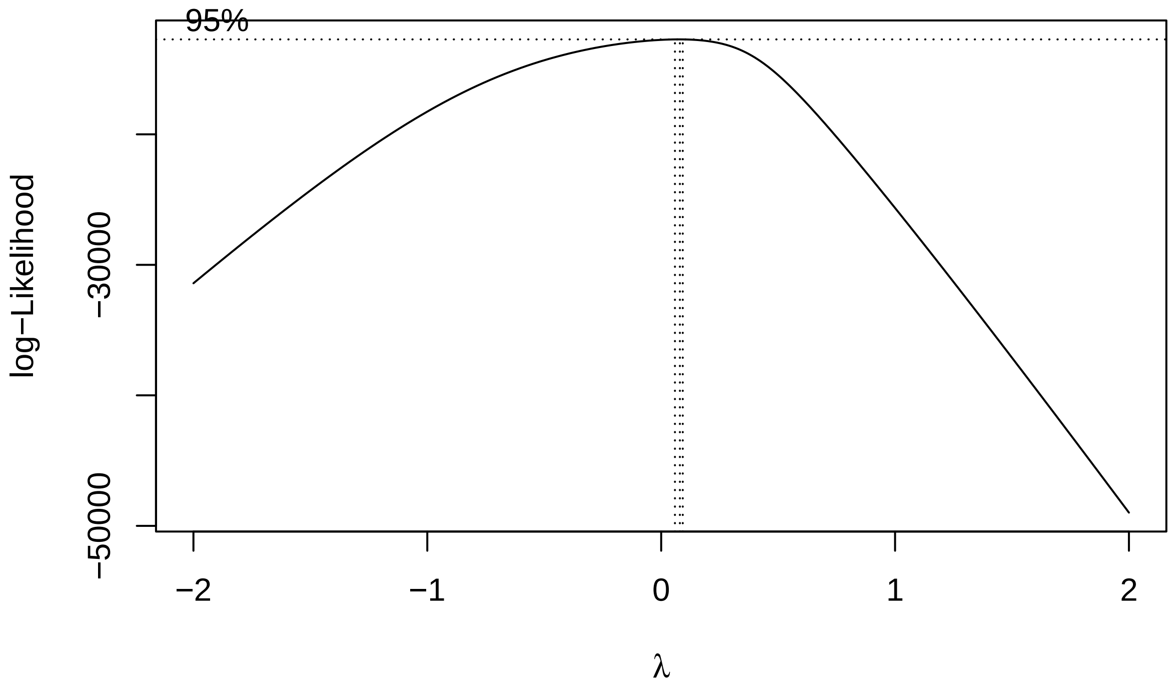
```
test = lm(TotalHours ~ ., starcraft[-1793,])
plot(test$fitted.values, rstandard(test), xlab = "Fitted Values",
ylab = "Standardized Residuals", pch = 16)
abline(h = 0, lty = 2, lwd = 2, col="red")
```



Even when the extreme outlier is removed, the fitted values vs. standard residuals plot show a slight outward funnel pattern due to other outliers.

Therefore, we decide to conduct a box-cox transformation without removing any outliers.

```
# perform box-cox analysis on full data
alpha = 0.05
lambda = seq(-2, 2, 0.01)

library(MASS)
BC = boxcox(full, lambda)
```



```r
range(BC$x[BC$y > max(BC$y) - qchisq(alpha, 1, lower.tail = FALSE) / 2])
```

```
## [1] 0.06 0.09
```

The 95% confidence interval for λ is equal to $[0.06, 0.09]$. The values in the 95% confidence interval are not intuitive. Since at $lambda = 0$, the log likelihood value is very close to the maximum. Hence, we decided to apply log transformation to the response variable.

```r
# apply log transformation to the response variable
starcraft["TotalHours"] = log(starcraft["TotalHours"])
```

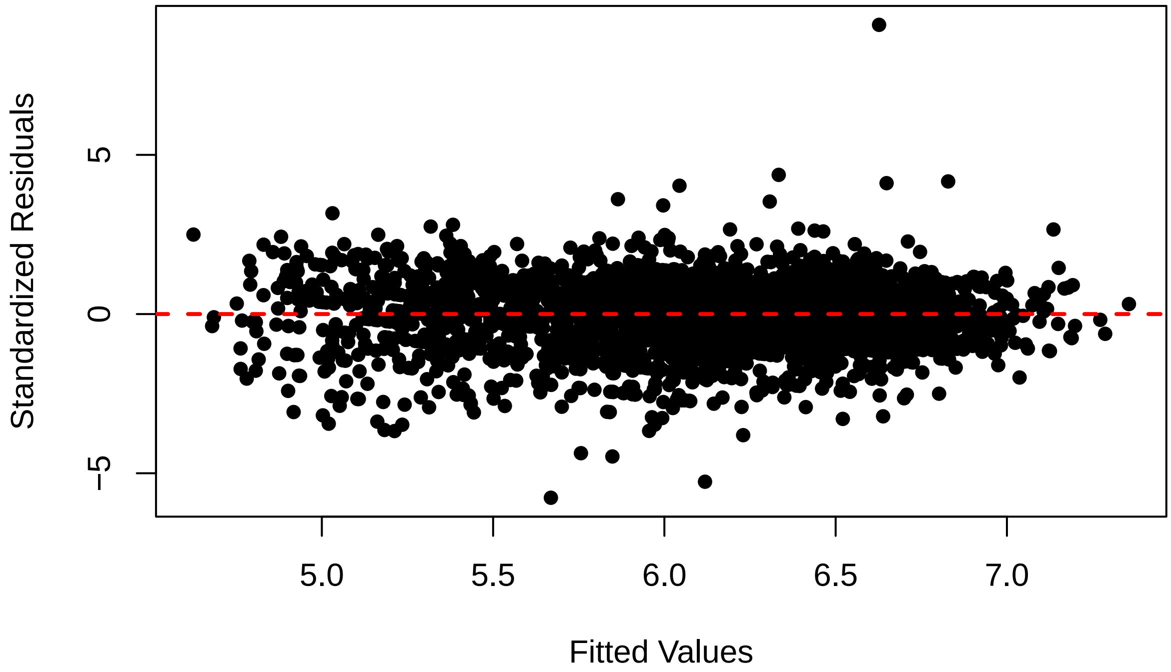
Now that the response variable is transformed, we fit a full model again.

```r
# refit the full model
full = lm(TotalHours ~ ., starcraft)
summary(full)
```

```
##
## Call:
## lm(formula = TotalHours ~ ., data = starcraft)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5700 -0.4129  0.0755  0.4874  7.1887
```

```
## 
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)    
## (Intercept)           2.265e+00  1.149e+00   1.971 0.048771 *  
## LeagueIndex2          2.668e-01  7.626e-02   3.498 0.000474 ***
## LeagueIndex3          6.775e-01  7.428e-02   9.121  < 2e-16 ***
## LeagueIndex4          8.516e-01  7.526e-02  11.315  < 2e-16 ***
## LeagueIndex5          1.144e+00  8.035e-02  14.233  < 2e-16 ***
## LeagueIndex6          1.389e+00  8.730e-02  15.913  < 2e-16 ***
## LeagueIndex7          1.747e+00  1.643e-01  10.635  < 2e-16 ***
## Age                   2.658e-01  7.912e-02   3.359 0.000790 ***
## APM                   5.515e-01  2.662e-01   2.071 0.038399 *  
## SelectByHotkeys      -4.690e-01  1.693e+00  -0.277 0.781717    
## AssignToHotkeys      -7.057e+00  3.338e+00  -2.114 0.034602 *  
## UniqueHotkeys        -1.549e-02  6.827e-03  -2.269 0.023340 *  
## MinimapAttacks        2.582e-01  2.408e+00   0.107 0.914627    
## MinimapRightClicks   -1.314e+00  1.871e+00  -0.703 0.482331    
## NumberOfPACs         -9.987e+00  9.396e+00  -1.063 0.287944    
## GapBetweenPACs        2.670e-02  5.576e-02   0.479 0.632093    
## ActionLatency        -6.819e-02  1.457e-01  -0.468 0.639726    
## ActionsInPAC         -3.410e-02  2.365e-01  -0.144 0.885396    
## TotalMapExplored      3.064e-02  2.456e-02   1.247 0.212321    
## WorkersMade          -5.217e-02  3.509e-02  -1.487 0.137225    
## UniqueUnitsMade       2.247e-02  9.685e-03   2.320 0.020399 *  
## ComplexUnitsMade     -1.665e+02  1.667e+02  -0.999 0.317896    
## ComplexAbilitiesUsed  2.240e+01  6.736e+01   0.333 0.739491    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.7945 on 3315 degrees of freedom
## Multiple R-squared:  0.2719, Adjusted R-squared:  0.267 
## F-statistic: 56.26 on 22 and 3315 DF,  p-value: < 2.2e-16
```

```
# plot fitted values vs standard residuals plot
plot(full$fitted.values, rstandard(full), xlab = "Fitted Values",
ylab = "Standardized Residuals", pch = 16)
abline(h = 0, lty = 2, lwd = 2, col="red")
```



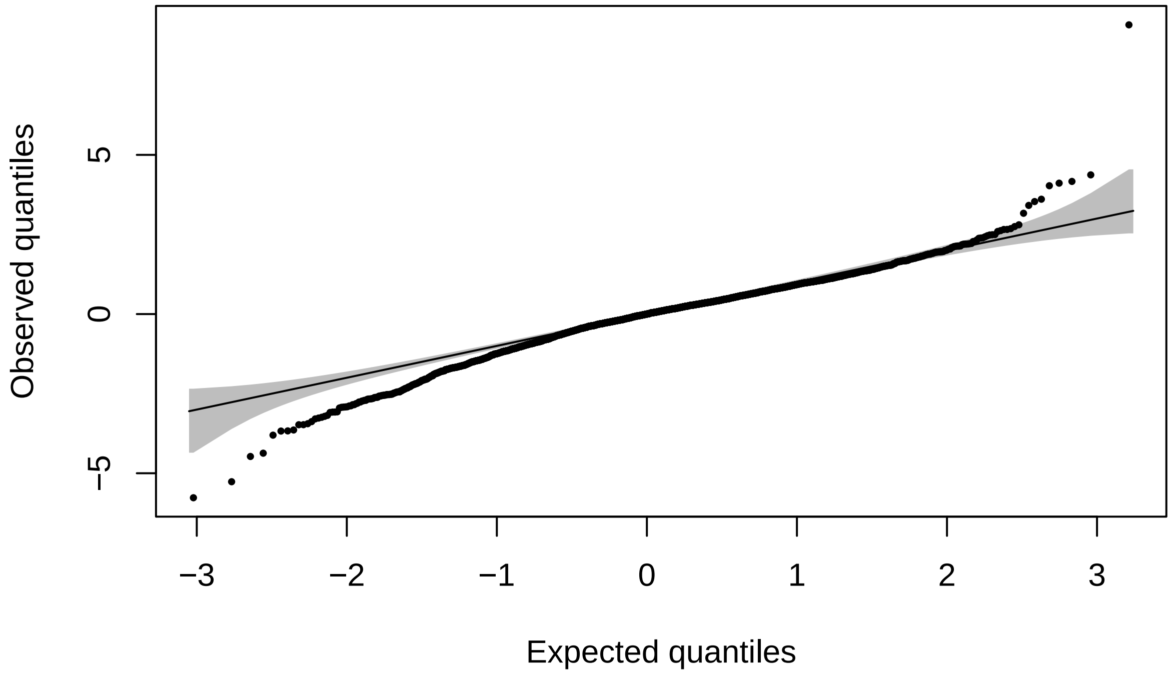
After the log transformation of the response variable, the fitted values vs. residuals plot looks devoid of heteroskedasticity and biasedness, even with all the outliers included in the data.

```
# qqplot
library(qqconf)
```

```
## Warning: package 'qqconf' was built under R version 4.4.3
```

```
qq_conf_plot(rstandard(full), points_params = list(pch = 16, cex = 0.5))
```

```
## no dparams supplied. Estimating parameters from the data...
```



The QQ-plot of standardized residuals show that the standardized residuals are not normally distributed. It suggests that the distribution of residuals is heavy-tailed, which aligns with how multiple boxplots of covariates displayed outliers.

2)

Next, we look at an interaction model. We examine interaction effects between *LeagueIndex* and *APM*.

```r
# interaction between LeagueIndex and APM
starcraft$LeagueIndex = factor(starcraft$LeagueIndex)
fit = lm(TotalHours ~ LeagueIndex * APM, starcraft)
summary(fit)
```

```
##
## Call:
## lm(formula = TotalHours ~ LeagueIndex * APM, data = starcraft)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7017 -0.4276  0.0782  0.4986  7.0712
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       4.92834    0.67231   7.330 2.87e-13 ***
## LeagueIndex2     -2.04105    0.88600  -2.304  0.02130 *
## LeagueIndex3     -0.35606    0.81053  -0.439  0.66048
## LeagueIndex4     -0.61214    0.78571  -0.779  0.43598
## LeagueIndex5     -0.45240    0.80995  -0.559  0.57650
## LeagueIndex6      0.92081    0.85627   1.075  0.28229
## LeagueIndex7     -2.03619    3.38572  -0.601  0.54761
## APM               0.03173    0.16666   0.190  0.84900
## LeagueIndex2:APM  0.55864    0.21443   2.605  0.00922 **
## LeagueIndex3:APM  0.25772    0.19518   1.320  0.18678
## LeagueIndex4:APM  0.35094    0.18845   1.862  0.06265 .
## LeagueIndex5:APM  0.37047    0.19098   1.940  0.05249 .
## LeagueIndex6:APM  0.13716    0.19720   0.696  0.48678
## LeagueIndex7:APM  0.76889    0.65639   1.171  0.24153
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.7975 on 3324 degrees of freedom
## Multiple R-squared:  0.2645, Adjusted R-squared:  0.2616
## F-statistic: 91.94 on 13 and 3324 DF,  p-value: < 2.2e-16
```

Coefficient Interpretations: (Intercept): Avg log(TotalHours) given LeagueIndex of 1 (reference level) and log(APM) of 0

(LeagueIndex2): Avg difference in log(TotalHours) given LeagueIndex of 2 (vs reference level) and log(APM) of 0

(APM): Avg difference in log(TotalHours) given LeagueIndex of 1 (reference level) and log(APM) increasing by 1

(LeagueIndex4:APM): Avg difference in effect of log(APM) on log(TotalHours) between LeagueIndex of 4 vs reference level

The interaction effect between $LeagueIndex2$ and APM is significant.

```
anova(fit)
```

```
## Analysis of Variance Table
##
## Response: TotalHours
##                   Df  Sum Sq Mean Sq  F value    Pr(>F)
## LeagueIndex        6  716.22 119.370 187.7098 < 2.2e-16 ***
## APM                1   36.92  36.921  58.0587 3.297e-14 ***
## LeagueIndex:APM    6    6.95   1.158   1.8209   0.09107 .
## Residuals       3324 2113.82   0.636
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

From the analysis of variance table, we see that overall, the interaction between $LeagueIndex$ and APM is not significant.

3)

The linear constraints are presented below.

$$H_0 : beta_0 + 3 * beta_{LeagueIndex7} = beta_0 + 2 * beta_{LeagueIndex5} + 2 * beta_{LeagueIndex6} = 10$$

We first perform the exact F test.

```
# exact F test
X = model.matrix(full)
beta = full$coefficients
n = dim(X)[1]
p = dim(X)[2]
S = sqrt(sum(full$residuals^2)/(n - p))
R = matrix(0, nrow=2, ncol = p)
R[1,1] = 1
R[1,7] = 3
R[2,1] = 1
R[2,5] = 2
R[2,6] = 2
r = c(10, 10)
k = length(r)
FT = crossprod(R %*% beta - r, solve(R %*% solve(crossprod(X),
                                         t(R)), R %*% beta - r))[,] / (k * S^2)
print(FT)
```

```
## [1] 2.666038
```

```
pvalue = pf(FT, k, n - p, lower.tail = FALSE)
print(pvalue)
```

```
## [1] 0.06967623
```

There is no strong evidence to reject the null hypothesis.

We then perform three asymptotic tests: Wald test, Score test, Likelihood-Ratio Test.

```
# wald test
WT = k * FT
print(WT)
```

```
## [1] 5.332076
```

```
pvalue = pchisq(WT, k, lower.tail = FALSE)
print(pvalue)
```

```
## [1] 0.06952716
```

```
# score test
beta0 = beta - solve(crossprod(X), crossprod(R, solve(R %*%
                                 solve(crossprod(X), t(R)), R %*% beta - r)))[,]
Y = starcraft$TotalHours
residuals0 = Y - X %*% beta0
sigma0 = sqrt(mean(residuals0^2))

SF = crossprod(X, residuals0) / sigma0^2
FI = crossprod(X) / sigma0^2
ST = crossprod(SF, solve(FI, SF))[,]
print(ST)
```

```
## [1] 5.360448
```

```r
pvalue = pchisq(ST, k, lower.tail = FALSE)
print(pvalue)
```

```
## [1] 0.06854778
```

```r
# likelihood ratio test
sigma = sqrt(mean(full$residuals^2))
LR = -n * log(sigma^2 / sigma0^2)
print(LR)
```

```
## [1] 5.364757
```

```r
pvalue = pchisq(LR, k, lower.tail = FALSE)
print(pvalue)
```

```
## [1] 0.06840026
```

The three asymptotic tests lead to the same conclusion that there is no strong evidence to reject the null hypothesis.

4)

We removed the qualitative covariate, *LeagueIndex*, and applied jackknife and nonparametric bootstrap method.

```r
# remove qualitative covariate
alpha = 0.05
full = lm(TotalHours ~ ., starcraft[,-1])
summary(full)
```

```
##
## Call:
## lm(formula = TotalHours ~ ., data = starcraft[, -1])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5540 -0.4482  0.0943  0.5306  6.8889
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              4.346950   1.197613   3.630 0.000288 ***
## Age                      0.324774   0.083156   3.906 9.59e-05 ***
## APM                      0.894231   0.271396   3.295 0.000995 ***
## SelectByHotkeys         -0.643133   1.719600  -0.374 0.708427
## AssignToHotkeys          0.833670   3.483776   0.239 0.810887
## UniqueHotkeys           -0.009900   0.007165  -1.382 0.167143
## MinimapAttacks           9.595683   2.472331   3.881 0.000106 ***
## MinimapRightClicks    -1.588506   1.971721  -0.806 0.420506
## NumberOfPACs         -12.961439   9.698111  -1.336 0.181480
## GapBetweenPACs        -0.083016   0.058465  -1.420 0.155725
## ActionLatency         -0.444652   0.151423  -2.936 0.003342 **
## ActionsInPAC          -0.278316   0.243466  -1.143 0.253062
## TotalMapExplored       0.005138   0.025848   0.199 0.842434
## WorkersMade            0.024325   0.036642   0.664 0.506821
## UniqueUnitsMade        0.020055   0.010193   1.967 0.049221 *
## ComplexUnitsMade     -77.891734 175.415847  -0.444 0.657042
## ComplexAbilitiesUsed  54.707999  70.929875   0.771 0.440586
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.8376 on 3321 degrees of freedom
## Multiple R-squared:  0.1892, Adjusted R-squared:  0.1853
## F-statistic: 48.43 on 16 and 3321 DF,  p-value: < 2.2e-16
```

```
Y = starcraft$TotalHours
X = starcraft[,-c(1,3)]
remove_qual = starcraft[,-c(1)]
n = dim(X)[1]
p = dim(X)[2]+1
beta = full$coefficients
```

```
# jackknife SE
betajack = matrix(0, n, p)
for (i in 1:n)
{
  jack = lm(TotalHours ~ ., remove_qual, setdiff(1:n, i))
  betajack[i,] = jack$coefficients
}
betabar = colMeans(betajack)
print(betabar)
```

```
##  [1]   4.346951709   0.324773593   0.894230070  -0.643129312   0.833668528
##  [6]  -0.009900359   9.595684374  -1.588506235 -12.961426151  -0.083015772
## [11]  -0.444652389  -0.278315941   0.005138444   0.024325393   0.020054514
## [16] -77.891345551  54.707830445
```

```
SE = sqrt(apply(betajack, 2, sd) * (n - 1)^2 / n)
print(SE)
```

```
##  [1]  8.5208888  2.1804983  4.1327946 10.5619155 13.9996503  0.6435464
##  [7] 11.8705691 10.3187062 23.6821709  1.8613282  2.9640136  3.8546092
## [13]  1.2095051  1.4535337  0.7516805 97.9113810 66.5111537
```

```
# nonparametric bootstrap SE
nboot = 10000
betaboot = matrix(0, nboot, p)
for (k in 1:nboot)
{
  boot = lm(TotalHours ~ ., remove_qual, sample(n, replace = TRUE))
  betaboot[k,] = boot$coefficients
}
betabar = colMeans(betaboot)
print(betabar)
```

```
##  [1]   4.354481404   0.323751946   0.892192315  -0.627451551   0.842370043
##  [6]  -0.009887573   9.583898519  -1.602229382 -12.915644900  -0.083086976
## [11]  -0.445007452  -0.276933576   0.005309233   0.024190193   0.019964192
## [16] -75.900327081  53.890527515
```

```
SE = apply(betaboot, 2, sd)
print(SE)
```

```
##  [1] 1.238015e+00 8.263924e-02 2.955573e-01 1.921302e+00 3.383995e+00
##  [6] 7.134769e-03 2.431358e+00 1.829752e+00 9.769415e+00 5.951326e-02
## [11] 1.512360e-01 2.581202e-01 2.520130e-02 3.679251e-02 9.743373e-03
## [16] 1.668204e+02 7.594660e+01
```

Although the mean of regression coefficient estimates are pretty similar, the standard errors of regression coefficients returned by the jackknife and nonparametric bootstrap methods are very different. This is most likely because the jackknife method is more sensitive to outliers, given that we noted multiple outliers in the data before.

```
# OLS SE
ols_model = lm(TotalHours ~ ., data = remove_qual)
ols_se = summary(ols_model)$coefficients[, "Std. Error"]
print(ols_se)
```

```
##          (Intercept)                  Age                  APM
##         1.197613e+00         8.315612e-02         2.713959e-01
##      SelectByHotkeys      AssignToHotkeys        UniqueHotkeys
##         1.719600e+00         3.483776e+00         7.165115e-03
##       MinimapAttacks   MinimapRightClicks         NumberOfPACs
##         2.472331e+00         1.971721e+00         9.698111e+00
##       GapBetweenPACs        ActionLatency         ActionsInPAC
##         5.846509e-02         1.514232e-01         2.434658e-01
##     TotalMapExplored          WorkersMade      UniqueUnitsMade
##         2.584779e-02         3.664196e-02         1.019346e-02
##     ComplexUnitsMade ComplexAbilitiesUsed
##         1.754158e+02         7.092988e+01
```

The standard errors returned by the nonparametric bootstrap method are more similar to the standard errors returned by the OLS model than those returned by the jackknife method.

We then take a look at the 95% confidence intervals for the regression coefficients returned by the OLS model.

```
# OLS 95% CI
alpha = 0.05
confint(full)
```

```
##                              2.5 %        97.5 %
## (Intercept)           1.998817e+00   6.695083713
## Age                   1.617312e-01   0.487816076
## APM                   3.621104e-01   1.426350644
## SelectByHotkeys      -4.014716e+00   2.728450266
## AssignToHotkeys      -5.996895e+00   7.664236032
## UniqueHotkeys        -2.394884e-02   0.004148136
## MinimapAttacks        4.748236e+00  14.443129704
## MinimapRightClicks   -5.454417e+00   2.277404859
## NumberOfPACs         -3.197632e+01   6.053439673
## GapBetweenPACs       -1.976469e-01   0.031615626
## ActionLatency        -7.415448e-01  -0.147760193
## ActionsInPAC         -7.556743e-01   0.199042063
## TotalMapExplored     -4.554075e-02   0.055817650
## WorkersMade          -4.751770e-02   0.096168506
## UniqueUnitsMade       6.842612e-05   0.040040614
## ComplexUnitsMade     -4.218258e+02 266.042357073
## ComplexAbilitiesUsed -8.436269e+01 193.778684680
```

We then use bootstrap estimates to construct different types of 95% CIs for the regression coefficients.

```
# 95% Normal CI
CInormal = cbind(betabar - qnorm(1 - alpha / 2) * SE, betabar +
qnorm(1 - alpha / 2) * SE)
print(CInormal)
```

```
##                [,1]          [,2]
##  [1,]  1.928017e+00   6.780946182
##  [2,]  1.617820e-01   0.485721874
##  [3,]  3.129107e-01   1.471473940
##  [4,] -4.393135e+00   3.138231779
##  [5,] -5.790138e+00   7.474878524
##  [6,] -2.387146e-02   0.004096318
##  [7,]  4.818524e+00  14.349272576
##  [8,] -5.188478e+00   1.984018967
##  [9,] -3.206335e+01   6.232057620
## [10,] -1.997308e-01   0.033556874
## [11,] -7.414245e-01  -0.148590376
## [12,] -7.828399e-01   0.228972764
## [13,] -4.408442e-02   0.054702882
## [14,] -4.792180e-02   0.096302188
## [15,]  8.675319e-04   0.039060853
## [16,] -4.028623e+02 251.061667741
## [17,] -9.496207e+01 202.743123994
```

```
# 95% Percentile CI
CIpercentile = cbind(apply(betaboot, 2, quantile, probs = alpha
/ 2), apply(betaboot, 2, quantile, probs = 1 - alpha / 2))
print(CIpercentile)
```

```
##                [,1]          [,2]
##  [1,]  1.932585e+00   6.755357755
##  [2,]  1.592279e-01   0.485652012
##  [3,]  3.101447e-01   1.465606023
##  [4,] -4.360239e+00   3.189249436
##  [5,] -5.754550e+00   7.445651324
##  [6,] -2.393958e-02   0.003964345
##  [7,]  4.755780e+00  14.275930608
##  [8,] -5.125181e+00   2.020257885
##  [9,] -3.191800e+01   6.577252056
## [10,] -1.992418e-01   0.032669461
## [11,] -7.423812e-01  -0.153488182
## [12,] -7.780579e-01   0.235540930
## [13,] -4.387351e-02   0.055327145
## [14,] -4.721677e-02   0.096435885
## [15,]  1.069296e-03   0.039190711
## [16,] -3.956908e+02 258.187919892
## [17,] -9.666561e+01 202.390243110
```

```r
# 95% Pivotal CI
CIpivotal = cbind(2 * beta - apply(betaboot, 2, quantile, probs
= 1 - alpha / 2), 2 * beta - apply(betaboot, 2, quantile,
probs = alpha / 2))
print(CIpivotal)
```

```
##                                [,1]          [,2]
## (Intercept)           1.938543e+00   6.761315819
## Age                   1.638953e-01   0.490319433
## APM                   3.228550e-01   1.478316331
## SelectByHotkeys      -4.475515e+00   3.073973437
## AssignToHotkeys      -5.778311e+00   7.421890939
## UniqueHotkeys        -2.376505e-02   0.004138874
## MinimapAttacks        4.915435e+00  14.435585090
## MinimapRightClicks   -5.197270e+00   1.948169533
## NumberOfPACs         -3.250013e+01   5.995126780
## GapBetweenPACs       -1.987007e-01   0.033210569
## ActionLatency        -7.358168e-01  -0.146923749
## ActionsInPAC         -7.921732e-01   0.221425720
## TotalMapExplored     -4.505025e-02   0.054150403
## WorkersMade          -4.778508e-02   0.095867575
## UniqueUnitsMade       9.183295e-04   0.039039744
## ComplexUnitsMade     -4.139714e+02 239.907377182
## ComplexAbilitiesUsed -9.297425e+01 206.081609121
```

The 95% CIs constructed using the bootstrap estimates are comparable to those returned by the OLS model, with similar sizes and marginally different bounds. We noted various violations of OLS assumptions such as normality and skewness. However, the convergence of the OLS 95% CIs with those constructed using the bootstrap estimates that the OLS model is robust thanks to the large sample size.

5)

We now conduct a nonparametric test of overall statistical significance using permutation F test.

```r
# permutation F test
f = summary(full)$fstatistic[1]
print(f)
```

```
##    value
## 48.43204
```

```r
pvalue = pf(f, p, n - p - 1, lower.tail = FALSE)
print(pvalue)
```

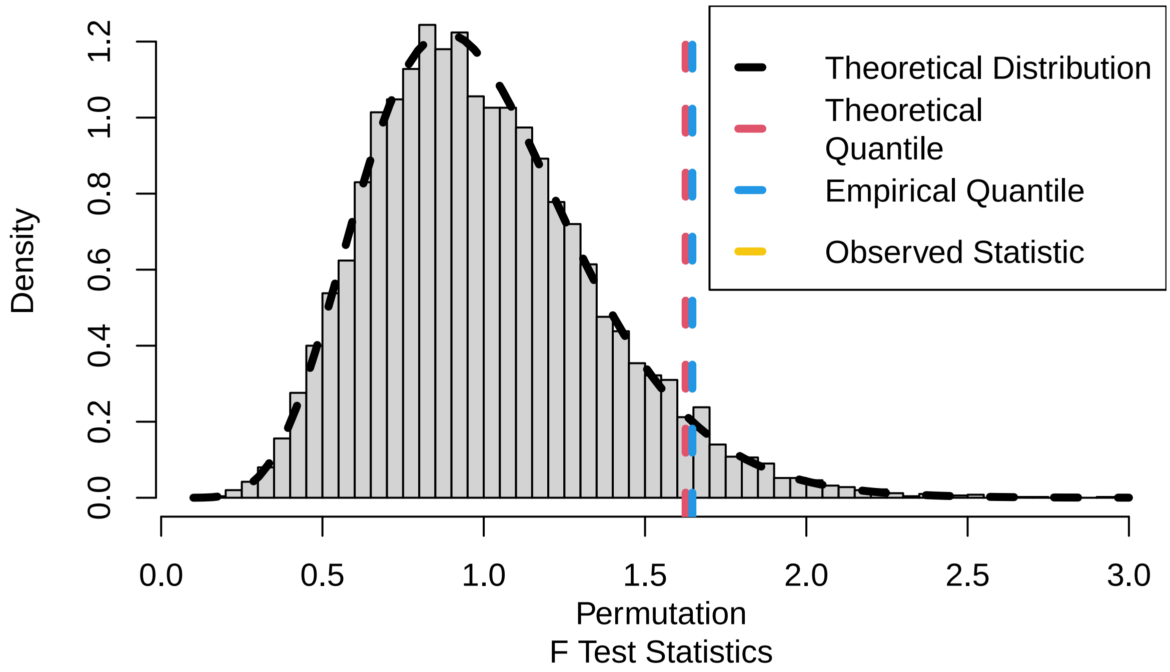
```
##         value
## 1.104321e-145
```

```r
nperm = 10000
fperm = numeric(nperm)
for (k in 1:nperm)
{
  Yperm = sample(Y)
  perm = lm(Yperm ~ ., X)
  fperm[k] = summary(perm)$fstatistic[1]
}

pvalue = (1 + sum(fperm > f)) / (1 + nperm)
print(pvalue)
```

```
## [1] 9.999e-05
```

The p-value of the permutation F-test is smaller than 0.05, but not as small as the original p-value of the F-test on the OLS model. This is likely because the OLS model has a set of assumptions such as normality, which are violated, as shown earlier. Therefore, the nonparametric test returns a larger p-value. However, even then, it suggests that there is strong evidence to reject the null hypothesis. In other words, the permutation F-test suggests that there is a statistically significant relationship between the response variable and the covariates encoded in the OLS model.

```r
# histogram of permutation F test statistics
hist(fperm, "FD", freq = FALSE, main = NA, xlab = "Permutation
F Test Statistics")

curve(df(x, p, n - p - 1), add = TRUE, lty = 2, lwd = 4)

abline(v = qf(1 - alpha, p, n - p - 1), col = 2, lty = 2, lwd =
4)

abline(v = quantile(fperm, 1 - alpha), col = 4, lty = 2, lwd =
4)

abline(v = f, col = 7, lty = 2, lwd = 4)

legend("topright", c("Theoretical Distribution", "Theoretical
Quantile", "Empirical Quantile", "Observed Statistic"), col
= c(1, 2, 4, 7), lty = rep(2, 4), lwd = rep(4, 4))
```



The observed f-test statistic is not shown on the graph above because it is very large, which is why the original p-value of the f-test was very small.

The theoretical and empirical quantile are very similar, like how the theoretical distribution and the distribution of the permutation f-test statistics are similar. The theoretical distribution is more to the left of the distribution of the permutation f-test statistics, which aligns with how the p-value of the permuation f-test was bigger than the p-value of the original f-test.

This shift in the distribution of the permutation f-test statistics demonstrate the impact of model assumptions, as mentioned above.

We then conduct a nonparametric test of statistical significance for the effect of APM on the response variable.

```
# permutation t-test
t = summary(full)$coefficients["APM", 3]
print(t)
```

```
## [1] 3.29493
```

```
pvalue = 2 * pt(abs(t), n - p - 1, lower.tail = FALSE)
print(pvalue)
```

```
## [1] 0.0009947995
```

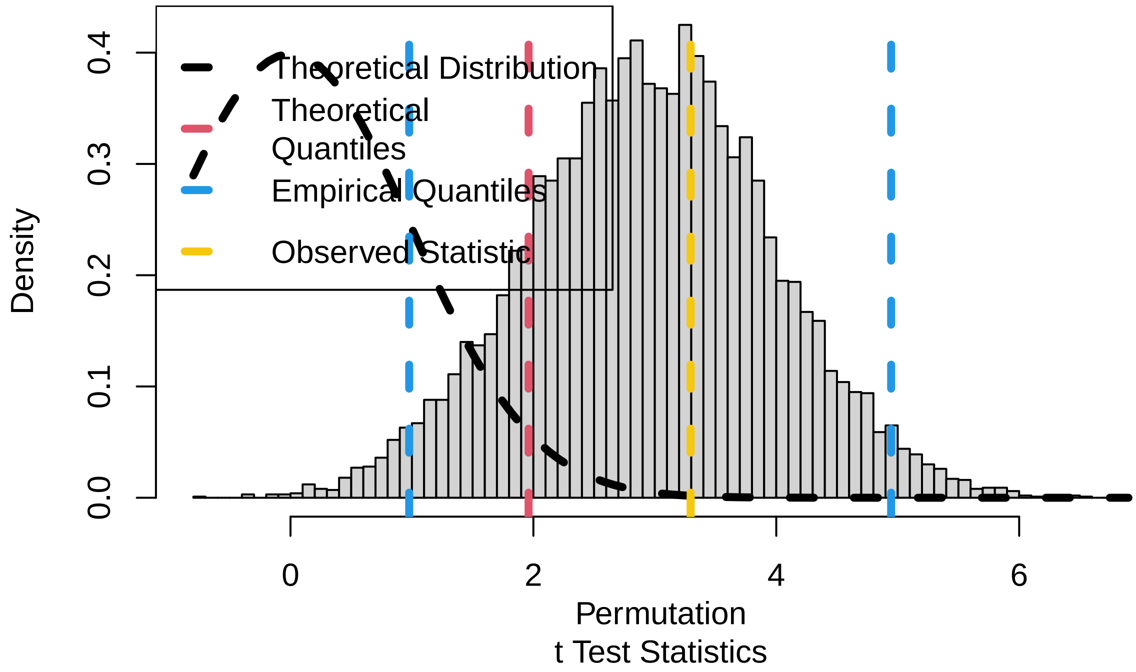
```
aux = lm(Y ~ ., X[,-3])
fitted = aux$fitted.values
residuals = aux$residuals
tperm = numeric(nperm)

for (k in 1:nperm)
{
  Yperm = fitted + sample(residuals)
  perm = lm(Yperm ~ ., X)
  tperm[k] = summary(perm)$coefficients["APM", 3]
}

pvalue = (1 + sum(abs(tperm) > abs(t))) / (1 + nperm)
print(pvalue)
```

```
## [1] 0.3737626
```

The p-value of the permutation t-test is smaller than 0.05, but larger than the p-value of the original t-test on the OLS model. The difference in p-values most likely stem from violations of OLS assumptions, as highlighted above. The difference in p-values for the t-test is not as large as the difference in p-values for the f-test, which is likely because f-test considers all covariates which increases complexity of the assumption violations.

Since the p-value of the permutation t-test is smaller than 0.05, there is strong evidence to reject the null hypothesis. In other words, the permutation t-test suggests that APM covariate has a signficant effect on the response variable.

```
# histogram of permutation t test statistics
hist(tperm, "FD", freq = FALSE, main = NA, xlab = "Permutation
t Test Statistics")

curve(dt(x, n - p - 1), add = TRUE, lty = 2, lwd = 4)

abline(v = qt(c(alpha / 2, 1 - alpha / 2), n - p - 1), col = 2,
lty = 2, lwd = 4)

abline(v = quantile(tperm, c(alpha / 2, 1 - alpha / 2)), col =
4, lty = 2, lwd = 4)

abline(v = t, col = 7, lty = 2, lwd = 4)

legend("topleft", c("Theoretical Distribution", "Theoretical
Quantiles", "Empirical Quantiles", "Observed Statistic"),
col = c(1, 2, 4, 7), lty = rep(2, 4), lwd = rep(4, 4))
```



We plot the histogram of permutation t-test statistics with the theoretical distribution. We notice that the distribution of permutation t-test statistics is more heavy-tailed. This might be related to the outliers that we observed in the data and violations of OLS assumptions.

The theoretical and empirical quantiles are very similar, which is reflected in the small difference in p-values.

6)

We then examine if the variance of the random error terms are non-constant as a function of any of the quantitative or qualitative covariates and perform a suitable heteroscedasticity test.

```
# breusch-pagan test on quantitative covariates
fit = lm(TotalHours ~ . -LeagueIndex, data = starcraft)
X = starcraft[,-c(1,3)]
p = dim(X)[2]
Yaux = residuals(fit)^2
aux = lm(Yaux ~ ., X)
BP = n * summary(aux)$r.squared
print(BP)
```

```
## [1] 64.97394
```

```
pvalue = pchisq(BP, p, lower.tail = FALSE)
print(pvalue)
```

```
## [1] 7.443153e-08
```

```
library(lmtest)
```

```
## Warning: package 'lmtest' was built under R version 4.4.3

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 4.4.3

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
bptest(fit)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  fit
## BP = 64.974, df = 16, p-value = 7.443e-08
```

We conducted Breusch-Pagan test on the quantitative covariates and concluded that there is strong evidence to reject the null hypothesis. In other words, the variance of random errors are non-constant as a function of the quantitative covariates.

```
# brown-forsythe test on qualitative covariate
fit = lm(TotalHours ~ LeagueIndex, data = starcraft)
Y = starcraft$TotalHours
Ymed = aggregate(TotalHours ~ LeagueIndex, starcraft,
                 median)[starcraft$LeagueIndex,2]
Yaux = abs(Y - Ymed)
aux = lm(Yaux ~ LeagueIndex, starcraft)
anova(aux)
```

```
## Analysis of Variance Table
##
## Response: Yaux
##                 Df Sum Sq Mean Sq F value    Pr(>F)
## LeagueIndex    6  49.87  8.3121  27.834 < 2.2e-16 ***
## Residuals   3331 994.73  0.2986
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
library(car)
```

```
## Loading required package: carData
```

```r
leveneTest(fit)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value    Pr(>F)
## group    6  27.834 < 2.2e-16 ***
##       3331
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We then conducted the Brown-Forsythe test on the qualitative covariate and concluded that there is strong evidence to reject the null hypothesis. In other words, the variance of random errors are non-constant as a function of the qualitative covariate, $LeagueIndex$.

Therefore, we can conclude that there is the variance of random errors are heteroskedastic. We now address the heteroskedasticity using various methods.

7)

First, we compute heteroskedasticity-consistent estimates of the covariance matrix and compare it to the corresponding OLS estimate.

```r
# OLS estimate of the covariance matrix
beta = full$coefficients
residuals = full$residuals
vcovOLS = summary(full)$sigma^2 * summary(full)$cov.unscaled
print(vcovOLS)
```

```
##                        (Intercept)           Age           APM SelectByHotkeys
## (Intercept)           1.4342759376 -2.373356e-02 -1.237536e-01    7.379972e-01
## Age                  -0.0237335635  6.914940e-03  9.672455e-04   -3.469113e-03
## APM                  -0.1237535900  9.672455e-04  7.365573e-02   -4.323988e-01
## SelectByHotkeys       0.7379972380 -3.469113e-03 -4.323988e-01    2.957024e+00
## AssignToHotkeys      -0.6548609293  1.586865e-03  1.946649e-01   -1.996149e+00
## UniqueHotkeys         0.0004538785 -4.800018e-05  2.114798e-05   -5.877892e-04
## MinimapAttacks        0.0544555362 -2.025324e-02  3.511345e-03   -9.747527e-02
## MinimapRightClicks    0.1033980818 -1.748887e-03 -1.376895e-02    1.381783e-01
## NumberOfPACs         -1.9319270660  3.567674e-03 -2.157232e+00    1.222717e+01
## GapBetweenPACs       -0.0102921698  3.383310e-04 -3.812847e-04   -1.875217e-03
## ActionLatency        -0.1450668591 -6.078536e-04  5.100189e-03   -2.387978e-02
## ActionsInPAC          0.0241669454 -1.415376e-04 -6.097891e-02    3.466242e-01
## TotalMapExplored      0.0023282313 -8.751421e-05  1.612131e-04   -1.151043e-04
## WorkersMade           0.0176543609 -7.750469e-05 -1.874665e-04    7.331307e-04
## UniqueUnitsMade       0.0003545885 -3.725650e-05  1.289024e-04   -3.670477e-04
## ComplexUnitsMade     -0.4284016313  6.680165e-01  1.217546e-01    8.825088e-01
## ComplexAbilitiesUsed -3.6631277799  2.296933e-02  2.471093e+00   -1.328676e+01
##                      AssignToHotkeys UniqueHotkeys MinimapAttacks
## (Intercept)            -6.548609e-01  4.538785e-04   0.0544555362
## Age                     1.586865e-03 -4.800018e-05  -0.0202532413
## APM                     1.946649e-01  2.114798e-05   0.0035113453
## SelectByHotkeys        -1.996149e+00 -5.877892e-04  -0.0974752659
## AssignToHotkeys         1.213670e+01 -5.663229e-03  -0.7449616144
## UniqueHotkeys          -5.663229e-03  5.133887e-05  -0.0006652770
## MinimapAttacks         -7.449616e-01 -6.652770e-04   6.1124222565
## MinimapRightClicks     -4.043564e-02 -1.207677e-04  -0.9562949799
## NumberOfPACs           -6.025802e+00 -3.254471e-03   0.0879461606
## GapBetweenPACs          1.970924e-02  1.390248e-05   0.0199476319
## ActionLatency           2.828896e-02 -3.657374e-05  -0.0056421296
## ActionsInPAC           -1.479372e-01 -2.104074e-05  -0.0082204814
## TotalMapExplored        2.245722e-03 -1.309388e-05  -0.0078074821
## WorkersMade            -8.322568e-03  9.164422e-06   0.0023951380
## UniqueUnitsMade         7.712906e-05 -5.730264e-06  -0.0004460299
## ComplexUnitsMade       -2.290816e+01  1.061989e-02   5.2584020443
## ComplexAbilitiesUsed   -7.853778e+00  3.427066e-03   4.1051344365
##                      MinimapRightClicks  NumberOfPACs GapBetweenPACs
## (Intercept)                0.1033980818  -1.931927066  -1.029217e-02
## Age                       -0.0017488869   0.003567674   3.383310e-04
## APM                       -0.0137689479  -2.157231796  -3.812847e-04
## SelectByHotkeys            0.1381782805  12.227168219  -1.875217e-03
## AssignToHotkeys           -0.0404356415  -6.025802205   1.970924e-02
## UniqueHotkeys             -0.0001207677  -0.003254471   1.390248e-05
## MinimapAttacks            -0.9562949799   0.087946161   1.994763e-02
## MinimapRightClicks         3.8876830409  -0.329435610   3.500344e-03
## NumberOfPACs              -0.3294356104  94.053357933   5.774728e-02
## GapBetweenPACs             0.0035003440   0.057747281   3.418166e-03
## ActionLatency             -0.0113302970   0.527804888  -2.100359e-03
## ActionsInPAC              -0.0337714542   2.150393265   2.152419e-03
## TotalMapExplored          -0.0032424020  -0.040240549  -3.357755e-04
## WorkersMade               -0.0046636020  -0.045918356  -2.005266e-04
## UniqueUnitsMade           -0.0017756735  -0.007659534   1.760187e-05
## ComplexUnitsMade          10.3462599108 -12.772090169  -5.225642e-01
## ComplexAbilitiesUsed      -3.1206874312 -71.756672888  -1.148860e-01
##                      ActionLatency  ActionsInPAC TotalMapExplored   WorkersMade
## (Intercept)          -1.450669e-01  2.416695e-02     2.328231e-03  1.765436e-02
## Age                  -6.078536e-04 -1.415376e-04    -8.751421e-05 -7.750469e-05
## APM                   5.100189e-03 -6.097891e-02     1.612131e-04 -1.874665e-04
## SelectByHotkeys      -2.387978e-02  3.466242e-01    -1.151043e-04  7.331307e-04
## AssignToHotkeys       2.828896e-02 -1.479372e-01     2.245722e-03 -8.322568e-03
## UniqueHotkeys        -3.657374e-05 -2.104074e-05    -1.309388e-05  9.164422e-06
## MinimapAttacks       -5.642130e-03 -8.220481e-03    -7.807482e-03  2.395138e-03
## MinimapRightClicks   -1.133030e-02 -3.377145e-02    -3.242402e-03 -4.663602e-03
## NumberOfPACs          5.278049e-01  2.150393e+00    -4.024055e-02 -4.591836e-02
## GapBetweenPACs       -2.100359e-03  2.152419e-03    -3.357755e-04 -2.005266e-04
## ActionLatency         2.292899e-02  2.650094e-03    -1.051892e-04 -5.343430e-04
## ActionsInPAC          2.650094e-03  5.927560e-02    -3.776749e-04 -1.005543e-03
## TotalMapExplored     -1.051892e-04 -3.776749e-04     6.681081e-04  6.069716e-05
## WorkersMade          -5.343430e-04 -1.005543e-03     6.069716e-05  1.342633e-03
## UniqueUnitsMade      -1.097770e-04 -7.649738e-05    -1.111046e-04 -7.848627e-06
## ComplexUnitsMade     -9.428370e-02 -1.034159e+00    -2.731467e-01 -7.513229e-01
## ComplexAbilitiesUsed  5.285145e-01 -2.272318e+00    -3.741600e-02  1.226354e-01
##                      UniqueUnitsMade ComplexUnitsMade ComplexAbilitiesUsed
## (Intercept)             3.545885e-04    -4.284016e-01        -3.663128e+00
## Age                    -3.725650e-05     6.680165e-01         2.296933e-02
## APM                     1.289024e-04     1.217546e-01         2.471093e+00
## SelectByHotkeys        -3.670477e-04     8.825088e-01        -1.328676e+01
## AssignToHotkeys         7.712906e-05    -2.290816e+01        -7.853778e+00
## UniqueHotkeys          -5.730264e-06     1.061989e-02         3.427066e-03
## MinimapAttacks         -4.460299e-04     5.258402e+00         4.105134e+00
## MinimapRightClicks     -1.775673e-03     1.034626e+01        -3.120687e+00
## NumberOfPACs           -7.659534e-03    -1.277209e+01        -7.175667e+01
## GapBetweenPACs          1.760187e-05    -5.225642e-01        -1.148860e-01
## ActionLatency          -1.097770e-04    -9.428370e-02         5.285145e-01
## ActionsInPAC           -7.649738e-05    -1.034159e+00        -2.272318e+00
## TotalMapExplored       -1.111046e-04    -2.731467e-01        -3.741600e-02
## WorkersMade            -7.848627e-06    -7.513229e-01         1.226354e-01
## UniqueUnitsMade         1.039066e-04    -3.410865e-01        -2.962190e-02
## ComplexUnitsMade       -3.410865e-01     3.077072e+04        -6.836233e+03
## ComplexAbilitiesUsed   -2.962190e-02    -6.836233e+03         5.031047e+03
```

```
print(diag(vcovOLS))
```

```
##          (Intercept)                  Age                  APM
##         1.434276e+00         6.914940e-03         7.365573e-02
##      SelectByHotkeys      AssignToHotkeys        UniqueHotkeys
##         2.957024e+00         1.213670e+01         5.133887e-05
##       MinimapAttacks   MinimapRightClicks         NumberOfPACs
##         6.112422e+00         3.887683e+00         9.405336e+01
##       GapBetweenPACs        ActionLatency         ActionsInPAC
##         3.418166e-03         2.292899e-02         5.927560e-02
##     TotalMapExplored          WorkersMade      UniqueUnitsMade
##         6.681081e-04         1.342633e-03         1.039066e-04
##     ComplexUnitsMade ComplexAbilitiesUsed
##         3.077072e+04         5.031047e+03
```

```
# heteroskedasticity-consistent estimate of the covariance matrix
library(sandwich)
```

```
## Warning: package ’sandwich’ was built under R version 4.4.3
```

```
vcovHC3 = vcovHC(full)
print(vcovHC3)
```

```
##                         (Intercept)           Age           APM SelectByHotkeys
## (Intercept)            1.5806788222 -2.722057e-02 -0.1721028734     1.087820325
## Age                   -0.0272205655  6.778398e-03  0.0012206019    -0.004244732
## APM                   -0.1721028734  1.220602e-03  0.0874740693    -0.541279326
## SelectByHotkeys        1.0878203255 -4.244732e-03 -0.5412793255     3.731425749
## AssignToHotkeys       -0.9386266143 -1.003399e-02  0.2614874265    -2.296594777
## UniqueHotkeys          0.0005666106 -6.970470e-05  0.0000644104    -0.000993496
## MinimapAttacks         0.0865201277 -1.245945e-02 -0.0039998317    -0.141199753
## MinimapRightClicks     0.1076043132  5.979587e-03 -0.0080077617     0.146979833
## NumberOfPACs          -0.9597974610  1.496462e-02 -2.3616297966    14.166225110
## GapBetweenPACs        -0.0068381736  3.478767e-05 -0.0008307369     0.004019835
## ActionLatency         -0.1542263911 -9.341168e-05  0.0088828588    -0.054912528
## ActionsInPAC           0.0570143395 -9.465457e-05 -0.0703614232     0.425061708
## TotalMapExplored       0.0039068299 -7.611022e-05 -0.0002890412     0.003686215
## WorkersMade            0.0198007605 -2.139642e-04 -0.0005162787     0.003359334
## UniqueUnitsMade        0.0002200429 -3.267887e-05  0.0001652051    -0.001104581
## ComplexUnitsMade     -3.8856027609  1.193083e+00  0.8206928679    -4.284621435
## ComplexAbilitiesUsed -7.0018591492 -2.063869e-01  3.0342142906   -21.392950234
##                      AssignToHotkeys UniqueHotkeys MinimapAttacks
## (Intercept)             -0.938626614  5.666106e-04   0.0865201277
## Age                     -0.010033989 -6.970470e-05  -0.0124594471
## APM                      0.261487426  6.441040e-05  -0.0039998317
## SelectByHotkeys         -2.296594777 -9.934960e-04  -0.1411997526
## AssignToHotkeys         11.517886499 -5.437685e-03  -0.8157427755
## UniqueHotkeys           -0.005437685  5.143084e-05  -0.0011329004
## MinimapAttacks          -0.815742776 -1.132900e-03   5.9537446952
## MinimapRightClicks       0.083522281 -2.861376e-04  -0.8928993828
## NumberOfPACs            -6.439810143 -5.066213e-03   0.3947276691
## GapBetweenPACs           0.009558337  8.649070e-07   0.0146089860
## ActionLatency            0.073862512 -2.420516e-05  -0.0004778306
## ActionsInPAC            -0.211331768 -6.483626e-05  -0.0029239435
## TotalMapExplored        -0.003521332 -1.096785e-05  -0.0072332214
## WorkersMade             -0.010784243  1.955860e-05   0.0095569931
## UniqueUnitsMade          0.002362865 -3.731945e-06   0.0001976019
## ComplexUnitsMade       -24.464043914 -1.869000e-02   6.1429933287
## ComplexAbilitiesUsed     0.413795296  1.014828e-03   5.4116938922
##                      MinimapRightClicks  NumberOfPACs GapBetweenPACs
## (Intercept)                0.1076043132  -0.959797461  -6.838174e-03
## Age                        0.0059795870   0.014964624   3.478767e-05
## APM                       -0.0080077617  -2.361629797  -8.307369e-04
## SelectByHotkeys            0.1469798332  14.166225110   4.019835e-03
## AssignToHotkeys            0.0835222809  -6.439810143   9.558337e-03
## UniqueHotkeys             -0.0002861376  -0.005066213   8.649070e-07
## MinimapAttacks            -0.8928993828   0.394727669   1.460899e-02
## MinimapRightClicks         3.3994216954  -0.767395980   2.134250e-03
## NumberOfPACs              -0.7673959801  94.317024978   6.071905e-02
## GapBetweenPACs             0.0021342499   0.060719054   3.599117e-03
## ActionLatency             -0.0157783397   0.423742401  -2.353411e-03
## ActionsInPAC              -0.0413354722   2.284868258   2.401094e-03
## TotalMapExplored          -0.0010228829  -0.031315863  -1.968188e-04
## WorkersMade               -0.0050840831  -0.047647721  -6.052714e-05
## UniqueUnitsMade           -0.0019513238  -0.007960471  -1.835443e-05
## ComplexUnitsMade          -1.9929582450 -25.386273403  -1.024123e+00
## ComplexAbilitiesUsed       4.5161089902 -65.151380256  -1.114742e-02
##                      ActionLatency  ActionsInPAC TotalMapExplored   WorkersMade
## (Intercept)          -1.542264e-01  5.701434e-02     3.906830e-03  1.980076e-02
## Age                  -9.341168e-05 -9.465457e-05    -7.611022e-05 -2.139642e-04
## APM                   8.882859e-03 -7.036142e-02    -2.890412e-04 -5.162787e-04
## SelectByHotkeys      -5.491253e-02  4.250617e-01     3.686215e-03  3.359334e-03
## AssignToHotkeys       7.386251e-02 -2.113318e-01    -3.521332e-03 -1.078424e-02
## UniqueHotkeys        -2.420516e-05 -6.483626e-05    -1.096785e-05  1.955860e-05
## MinimapAttacks       -4.778306e-04 -2.923943e-03    -7.233221e-03  9.556993e-03
## MinimapRightClicks   -1.577834e-02 -4.133547e-02    -1.022883e-03 -5.084083e-03
## NumberOfPACs          4.237424e-01  2.284868e+00    -3.131586e-02 -4.764772e-02
## GapBetweenPACs       -2.353411e-03  2.401094e-03    -1.968188e-04 -6.052714e-05
## ActionLatency         2.314328e-02 -1.038105e-04    -3.714359e-04 -8.417006e-04
## ActionsInPAC         -1.038105e-04  6.619504e-02    -6.167958e-05 -7.608964e-04
## TotalMapExplored     -3.714359e-04 -6.167958e-05     6.417041e-04  5.865847e-05
## WorkersMade          -8.417006e-04 -7.608964e-04     5.865847e-05  1.338457e-03
## UniqueUnitsMade      -4.936699e-05 -1.148835e-04    -1.112085e-04  2.445434e-07
## ComplexUnitsMade      1.152425e-01 -1.218819e+00    -2.265265e-01 -8.133662e-01
## ComplexAbilitiesUsed  9.355266e-01 -2.640000e+00    -9.659723e-02  1.044313e-01
##                      UniqueUnitsMade ComplexUnitsMade ComplexAbilitiesUsed
## (Intercept)             2.200429e-04       -3.8856028        -7.001859e+00
## Age                    -3.267887e-05        1.1930825        -2.063869e-01
## APM                     1.652051e-04        0.8206929         3.034214e+00
## SelectByHotkeys        -1.104581e-03       -4.2846214        -2.139295e+01
## AssignToHotkeys         2.362865e-03      -24.4640439         4.137953e-01
## UniqueHotkeys          -3.731945e-06       -0.0186900         1.014828e-03
## MinimapAttacks          1.976019e-04        6.1429933         5.411694e+00
## MinimapRightClicks     -1.951324e-03       -1.9929582         4.516109e+00
## NumberOfPACs           -7.960471e-03      -25.3862734        -6.515138e+01
## GapBetweenPACs         -1.835443e-05       -1.0241230        -1.114742e-02
## ActionLatency          -4.936699e-05        0.1152425         9.355266e-01
## ActionsInPAC           -1.148835e-04       -1.2188187        -2.640000e+00
## TotalMapExplored       -1.112085e-04       -0.2265265        -9.659723e-02
## WorkersMade             2.445434e-07       -0.8133662         1.044313e-01
## UniqueUnitsMade         9.572762e-05       -0.2302809        -4.949318e-02
## ComplexUnitsMade       -2.302809e-01    27557.3058626        -7.564517e+03
## ComplexAbilitiesUsed   -4.949318e-02    -7564.5171430         5.867890e+03
```

```
print(diag(vcovHC3))
```

```
##          (Intercept)                  Age                  APM
##         1.580679e+00         6.778398e-03         8.747407e-02
##      SelectByHotkeys      AssignToHotkeys        UniqueHotkeys
##         3.731426e+00         1.151789e+01         5.143084e-05
##       MinimapAttacks   MinimapRightClicks         NumberOfPACs
##         5.953745e+00         3.399422e+00         9.431702e+01
##       GapBetweenPACs        ActionLatency         ActionsInPAC
##         3.599117e-03         2.314328e-02         6.619504e-02
##     TotalMapExplored          WorkersMade      UniqueUnitsMade
##         6.417041e-04         1.338457e-03         9.572762e-05
##     ComplexUnitsMade ComplexAbilitiesUsed
##         2.755731e+04         5.867890e+03
```

We notice that the heteroskedastic-consistent estimate of the covariance matrix is different than but similar overall to the correponsing OLS estimate.

Then, we compute the OLS and heteroskedasticity-consistent t-test statistics.

```
# OLS t-test statistics
library(lmtest)
coeftest(full)
```

```
##
## t test of coefficients:
##
##                          Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)             4.3469504   1.1976126  3.6297 0.0002881 ***
## Age                     0.3247737   0.0831561  3.9056 9.587e-05 ***
## APM                     0.8942305   0.2713959  3.2949 0.0009948 ***
## SelectByHotkeys        -0.6431327   1.7196001 -0.3740 0.7084272
## AssignToHotkeys        0.8336703   3.4837764  0.2393 0.8108872
## UniqueHotkeys         -0.0099004   0.0071651 -1.3817 0.1671434
## MinimapAttacks         9.5956826   2.4723313  3.8812 0.0001059 ***
## MinimapRightClicks    -1.5885059   1.9717208 -0.8056 0.4205056
## NumberOfPACs         -12.9614388   9.6981110 -1.3365 0.1814804
## GapBetweenPACs        -0.0830156   0.0584651 -1.4199 0.1557255
## ActionLatency         -0.4446525   0.1514232 -2.9365 0.0033423 **
## ActionsInPAC          -0.2783161   0.2434658 -1.1431 0.2530618
## TotalMapExplored       0.0051384   0.0258478  0.1988 0.8424341
## WorkersMade            0.0243254   0.0366420  0.6639 0.5068212
## UniqueUnitsMade        0.0200545   0.0101935  1.9674 0.0492212 *
## ComplexUnitsMade     -77.8917344 175.4158472 -0.4440 0.6570423
## ComplexAbilitiesUsed  54.7079986  70.9298753  0.7713 0.4405858
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# heteroskedasticity-consistent t-test statistics
coeftest(full, vcov. = vcovHC3)
```

```
##
## t test of coefficients:
##
##                         Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)            4.3469504   1.2572505  3.4575 0.0005521 ***
## Age                    0.3247737   0.0823310  3.9447 8.154e-05 ***
## APM                    0.8942305   0.2957602  3.0235 0.0025178 **
## SelectByHotkeys       -0.6431327   1.9316899 -0.3329 0.7392022
## AssignToHotkeys        0.8336703   3.3938012  0.2456 0.8059722
## UniqueHotkeys         -0.0099004   0.0071715 -1.3805 0.1675233
## MinimapAttacks         9.5956826   2.4400297  3.9326 8.575e-05 ***
## MinimapRightClicks    -1.5885059   1.8437521 -0.8616 0.3889910
## NumberOfPACs         -12.9614388   9.7116953 -1.3346 0.1820917
## GapBetweenPACs        -0.0830156   0.0599926 -1.3838 0.1665239
## ActionLatency         -0.4446525   0.1521291 -2.9229 0.0034917 **
## ActionsInPAC          -0.2783161   0.2572840 -1.0817 0.2794436
## TotalMapExplored       0.0051384   0.0253319  0.2028 0.8392685
## WorkersMade            0.0243254   0.0365849  0.6649 0.5061592
## UniqueUnitsMade        0.0200545   0.0097840  2.0497 0.0404706 *
## ComplexUnitsMade     -77.8917344 166.0039333 -0.4692 0.6389459
## ComplexAbilitiesUsed  54.7079986  76.6021551  0.7142 0.4751639
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimates of the regression coefficients are the same, because they are unbiased estimates computed by minimizing the residual sum of squares.

Using the heteroskedasticity consistent estimate of the covariance matrix does slightly change the t-test statistics and the p-values. Still, the list of covariates that have significant effect on the response variable do not change: *Intercept*, *Age*, *APM*, *MinimapAttacks*, *ActionLatency*, *UniqueUnitsMade*. This aligns with how the heteroskedasticity-consistent estimate of the covariance matrix is similar to the corresponding OLS estimate.

Lastly, we compute the OLS and heteroskedasticity-consistent 95% confidence intervals.

```r
# OLS 95% CI
confint(coeftest(full))
```

```
##                             2.5 %       97.5 %
## (Intercept)          1.998817e+00   6.695083713
## Age                  1.617312e-01   0.487816076
## APM                  3.621104e-01   1.426350644
## SelectByHotkeys     -4.014716e+00   2.728450266
## AssignToHotkeys     -5.996895e+00   7.664236032
## UniqueHotkeys       -2.394884e-02   0.004148136
## MinimapAttacks       4.748236e+00  14.443129704
## MinimapRightClicks  -5.454417e+00   2.277404859
## NumberOfPACs        -3.197632e+01   6.053439673
## GapBetweenPACs      -1.976469e-01   0.031615626
## ActionLatency       -7.415448e-01  -0.147760193
## ActionsInPAC        -7.556743e-01   0.199042063
## TotalMapExplored    -4.554075e-02   0.055817650
## WorkersMade         -4.751770e-02   0.096168506
## UniqueUnitsMade      6.842612e-05   0.040040614
## ComplexUnitsMade    -4.218258e+02 266.042357073
## ComplexAbilitiesUsed -8.436269e+01 193.778684680
```

```r
# heteroskedasticity-consistent 95% CI
confint(coeftest(full, vcov. = vcovHC3))
```

```
##                             2.5 %       97.5 %
## (Intercept)          1.881886e+00   6.812014469
## Age                  1.633490e-01   0.486198331
## APM                  3.143399e-01   1.474121117
## SelectByHotkeys     -4.430556e+00   3.144290209
## AssignToHotkeys     -5.820483e+00   7.487823543
## UniqueHotkeys       -2.396142e-02   0.004160714
## MinimapAttacks       4.811569e+00  14.379796473
## MinimapRightClicks  -5.203511e+00   2.026499250
## NumberOfPACs        -3.200295e+01   6.080073959
## GapBetweenPACs      -2.006419e-01   0.034610668
## ActionLatency       -7.429288e-01  -0.146376129
## ActionsInPAC        -7.827673e-01   0.226135055
## TotalMapExplored    -4.452922e-02   0.054806121
## WorkersMade         -4.740588e-02   0.096056690
## UniqueUnitsMade      8.711441e-04   0.039237896
## ComplexUnitsMade    -4.033721e+02 247.588619259
## ComplexAbilitiesUsed -9.548421e+01 204.900202157
```

In line with the observations made above, the OLS and heteroskedasticity-consistent 95% confidence intervals are different but similar. The size and range of the 95% confidence intervals significantly overlap and does not change whether a confidence interval contains zero or not.

In part 6, we concluded that heteroskedasticity exists. However, the observations made in part 7 on the heteroskedasticity consistent estimates suggest that the degree of heteroskedasticity is not severe.

8)

Next, we apply the weighted least squares method. We first apply the iteratively reweighted least squares method to specify suitable weights for the variances of the random error terms as a function of *Age* covariate.

```r
# apply the iteratively reweighted least squares method and apply
# the weighted least squares method
w = rep(1, n)
err = Inf
residuals = residuals(full)
while (err > 1e-5)
{
  aux = lm(log(residuals^2) ~ log(Age), remove_qual)
  err = sum(abs(w - exp(-aux$fitted.values))) / sum(w)
  w = exp(-aux$fitted.values)
  WLS = lm(TotalHours ~ Age, remove_qual, weights = w)
  residuals = WLS$residuals
}
summary(aux)
```

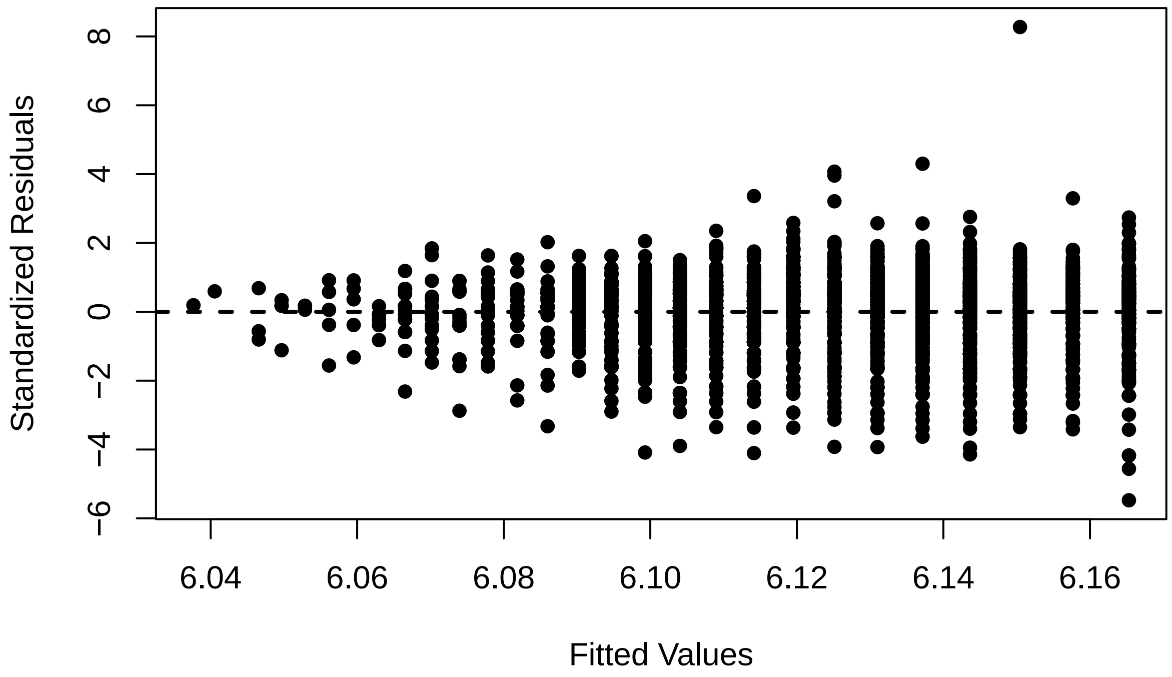
```
##
## Call:
## lm(formula = log(residuals^2) ~ log(Age), data = remove_qual)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.5441  -1.0275   0.3434   1.4056   5.6972
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.6713     0.7078  -2.361   0.0183 *
## log(Age)      0.0447     0.6334   0.071   0.9437
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 2.163 on 3336 degrees of freedom
## Multiple R-squared:  1.493e-06,  Adjusted R-squared:  -0.0002983
## F-statistic: 0.004981 on 1 and 3336 DF,  p-value: 0.9437
```

```r
summary(WLS)
```

```
##
## Call:
## lm(formula = TotalHours ~ Age, data = remove_qual, weights = w)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -11.4221  -0.9753   0.2373   1.2454  17.2637
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.51522    0.26902  24.218   <2e-16 ***
## Age          -0.12620    0.08784  -1.437    0.151
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.087 on 3336 degrees of freedom
## Multiple R-squared:  0.0006184,  Adjusted R-squared:  0.0003188
## F-statistic: 2.064 on 1 and 3336 DF,  p-value: 0.1509
```

Now we plot the fitted values against the standard residuals of the weighted least squares model.

```
# apply the weighted least squares method
plot(WLS$fitted.values, rstandard(WLS), xlab = "Fitted Values",
ylab = "Standardized Residuals", pch = 16)
abline(h = 0, lty = 2, lwd = 2)
```



We notice that the variance of the random errors is heteroskedastic even after the weighted least squares method is applied. This might be because not all covariates are accounted for in the weighted least squares model.

```
WLS = lm(TotalHours ~ Age, remove_qual, weights = w)
cov_wls <- vcov(WLS)
print(cov_wls)
```

```
##               (Intercept)          Age
## (Intercept)  0.07237287 -0.023588409
## Age         -0.02358841  0.007715647
```

```r
coeftest(WLS)
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.515220   0.269022 24.2182   <2e-16 ***
## Age         -0.126201   0.087839 -1.4367   0.1509
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

```r
confint(coeftest(WLS))
```

```
##                  2.5 %     97.5 %
## (Intercept)  5.9877547 7.04268459
## Age         -0.2984238 0.04602278
```

```r
OLS <- lm(TotalHours ~ Age, data = remove_qual)
vcov(OLS)
```

```
##             (Intercept)          Age
## (Intercept)  0.07228303 -0.023555323
## Age         -0.02355532  0.007703608
```

```r
coeftest(OLS)
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.51773    0.26886 24.2426   <2e-16 ***
## Age         -0.12702    0.08777 -1.4472   0.1479
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

```r
confint(OLS)
```

```
##                  2.5 %     97.5 %
## (Intercept)  5.9905970 7.04487189
## Age         -0.2991104 0.04506733
```

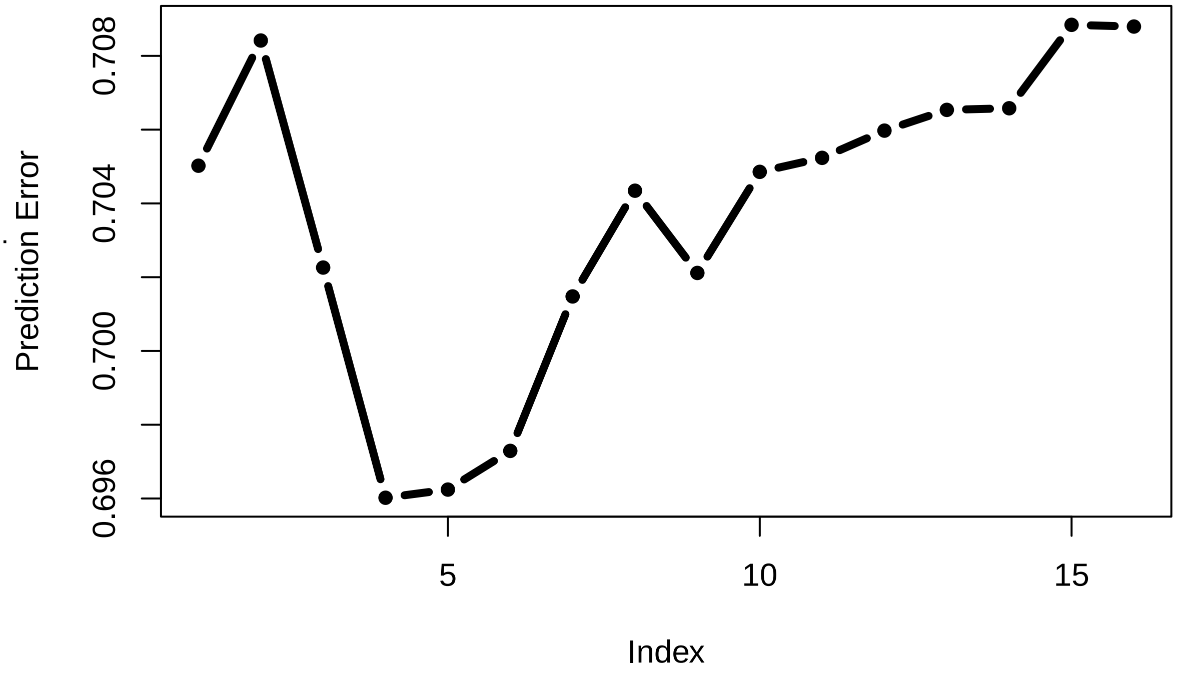
The regression coefficient estimates, t-test statistics, and 95% CIs are similar for both OLS and weighted least squares, although the CIs for the weighted least squares are slightly shifted.

This suggests that although heteroskedasticity exists and is not fully corrected after applying the weighted least squares method, it is not severe enough to have a significant effect on the regression coefficients, t-test statistics, and 95% CIs.

9)

Ignoring heteroskedasticity issues from now on, we perform appropriate model selection and draw inferences on the regression coefficients of the final model.

```r
# valid post selection inference
set.seed(1)
train = sample(n, 0.6 * n)
valid = sample(setdiff(1:n, train), 0.2 * n)
test = setdiff(1:n, c(train, valid))
P = dim(X)[2]
MSPE = numeric(P)

for (p in P:1)
{
  fit = lm(Y ~ ., X[, 1:p, drop = FALSE], train)
  predictions = predict(fit, X[valid, 1:p, drop = FALSE])
  MSPE[p] = mean((Y[valid] - predictions)^2)
  ind = which.max(summary(fit)$coefficients[-1, 4])
  if (ind != p)
  {
    X[,c(ind, p)] = X[,c(p, ind)]
    names(X)[c(ind, p)] = names(X)[c(p, ind)]
  }
}

plot(MSPE, type = "b", pch = 16, lwd = 4, ylab = "Mean Squared
Prediction Error")
```



```
m = which.min(MSPE)
fit = lm(Y ~ ., X[,1:m, drop = FALSE], test)
summary(fit)
```

```
##
## Call:
## lm(formula = Y ~ ., data = X[, 1:m, drop = FALSE], subset = test)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6696 -0.4623  0.0773  0.5227  6.7733
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.10340    1.53664   3.321 0.000946 ***
## APM               0.64029    0.14029   4.564 5.98e-06 ***
## MinimapAttacks   -0.05684    5.27407  -0.011 0.991404
## Age               0.11643    0.19002   0.613 0.540265
## ActionLatency    -0.55995    0.20374  -2.748 0.006153 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8475 on 664 degrees of freedom
## Multiple R-squared:  0.1921, Adjusted R-squared:  0.1872
## F-statistic: 39.47 on 4 and 664 DF,  p-value: < 2.2e-16
```

```r
summary(full)
```

```
##
## Call:
## lm(formula = TotalHours ~ ., data = starcraft[, -1])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5540 -0.4482  0.0943  0.5306  6.8889
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.346950   1.197613   3.630 0.000288 ***
## Age                   0.324774   0.083156   3.906 9.59e-05 ***
## APM                   0.894231   0.271396   3.295 0.000995 ***
## SelectByHotkeys      -0.643133   1.719600  -0.374 0.708427
## AssignToHotkeys       0.833670   3.483776   0.239 0.810887
## UniqueHotkeys        -0.009900   0.007165  -1.382 0.167143
## MinimapAttacks        9.595683   2.472331   3.881 0.000106 ***
## MinimapRightClicks   -1.588506   1.971721  -0.806 0.420506
## NumberOfPACs        -12.961439   9.698111  -1.336 0.181480
## GapBetweenPACs       -0.083016   0.058465  -1.420 0.155725
## ActionLatency        -0.444652   0.151423  -2.936 0.003342 **
## ActionsInPAC         -0.278316   0.243466  -1.143 0.253062
## TotalMapExplored      0.005138   0.025848   0.199 0.842434
## WorkersMade           0.024325   0.036642   0.664 0.506821
## UniqueUnitsMade       0.020055   0.010193   1.967 0.049221 *
## ComplexUnitsMade    -77.891734 175.415847  -0.444 0.657042
## ComplexAbilitiesUsed  54.707999  70.929875   0.771 0.440586
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8376 on 3321 degrees of freedom
## Multiple R-squared:  0.1892, Adjusted R-squared:  0.1853
## F-statistic: 48.43 on 16 and 3321 DF,  p-value: < 2.2e-16
```

We selected four covariates: APM, $MinimapAttacks$, Age, and $ActionLatency$. The covariates that are selected may change when the qualitative covariate is also considered. Among the four covariates, the regression coefficients for $Intercept$, APM, and $ActionLatency$ are significant.

Compared to the full model without the qualitative covariate, all significant covariates are selected except $UniqueUnitsMade$ by the valid post-selection inference. The p-value of the $UniqueUnitsMade$ covariate is close to 0.05 and since we used a validation set, the significance of $UniqueUnitsMade$ could have fluctuated.

10)

We fit a ridge regression model with all available covariates except the qualitative covariate.

```r
# ridge regression model
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.4.3
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```
X = as.matrix(X)
cv.ridge = cv.glmnet(X[train, ], Y[train], alpha = 0)
ridge = glmnet(X[test, ], Y[test], alpha = 0, lambda = cv.ridge$lambda.min)
print("Ridge")
```

```
## [1] "Ridge"
```

```
print(ridge$beta[,1])
```

```
##                  APM       MinimapAttacks                  Age
##         2.809481e-01        -1.735801e-01         5.800342e-02
##        ActionLatency         NumberOfPACs         ActionsInPAC
##        -3.690668e-01         1.039966e+01         1.377971e-02
##      SelectByHotkeys   MinimapRightClicks      UniqueUnitsMade
##         4.002687e+00         2.460486e+00         1.290089e-02
##        UniqueHotkeys       GapBetweenPACs          WorkersMade
##        -1.424884e-02        -1.127675e-01         2.561252e-02
##      AssignToHotkeys     ComplexUnitsMade ComplexAbilitiesUsed
##         8.332120e-01         1.273851e+01        -1.147818e+02
##     TotalMapExplored
##         6.949505e-03
```

```
print("OLS")
```

```
## [1] "OLS"
```

```
# corresponding OLS estimates of regression coefficients
print(summary(full)$coefficients[-1, "Estimate"])
```

```
##                  Age                  APM      SelectByHotkeys
##          0.324773654          0.894230520         -0.643132711
##      AssignToHotkeys        UniqueHotkeys       MinimapAttacks
##          0.833670302         -0.009900351          9.595682640
##   MinimapRightClicks         NumberOfPACs       GapBetweenPACs
##         -1.588505913        -12.961438782         -0.083015613
##        ActionLatency         ActionsInPAC     TotalMapExplored
##         -0.444652476         -0.278316113          0.005138449
##          WorkersMade      UniqueUnitsMade     ComplexUnitsMade
##          0.024325403          0.020054520        -77.891734400
## ComplexAbilitiesUsed
##         54.707998567
```

The ridge regression estimates of coefficients are overall smaller in magnitude than the corresponding OLS estimates, which is because of the penalty imposed on the sum of squared coefficient values.

```r
# boostrap to estimate standard errors
nboot = 10000
beta = matrix(0, nboot, P)
colnames(beta) = colnames(X)

for (i in 1:nboot)
{
  ind = sample(test, replace = TRUE)
  beta[i,] = glmnet(X[ind,], Y[ind], alpha = 0,
                    lambda = cv.ridge$lambda.min)$beta[,]
}

SE = apply(beta, 2, sd)
print("Ridge")
```

```
## [1] "Ridge"
```

```r
print(SE)
```

```
##                  APM       MinimapAttacks                  Age
##           0.07006980           4.46101375           0.17538511
##        ActionLatency         NumberOfPACs         ActionsInPAC
##           0.13933195           3.90582551           0.11562973
##      SelectByHotkeys   MinimapRightClicks      UniqueUnitsMade
##           1.50023949           3.66206111           0.01615999
##        UniqueHotkeys       GapBetweenPACs          WorkersMade
##           0.01383072           0.10136442           0.06275734
##      AssignToHotkeys     ComplexUnitsMade ComplexAbilitiesUsed
##           6.27920606         269.47336916         123.34467782
##     TotalMapExplored
##           0.04339283
```

```r
print("OLS")
```

```
## [1] "OLS"
```

```r
# corresponding OLS estimates of standard errors
print(summary(full)$coefficients[-1, "Std. Error"])
```

```
##                  Age                  APM      SelectByHotkeys
##         8.315612e-02         2.713959e-01         1.719600e+00
##      AssignToHotkeys        UniqueHotkeys       MinimapAttacks
##         3.483776e+00         7.165115e-03         2.472331e+00
##   MinimapRightClicks         NumberOfPACs       GapBetweenPACs
##         1.971721e+00         9.698111e+00         5.846509e-02
##        ActionLatency         ActionsInPAC     TotalMapExplored
##         1.514232e-01         2.434658e-01         2.584779e-02
##          WorkersMade      UniqueUnitsMade     ComplexUnitsMade
##         3.664196e-02         1.019346e-02         1.754158e+02
## ComplexAbilitiesUsed
##         7.092988e+01
```

Although the ridge regression estimates of regression coefficients are generally smaller in magnitude, the bootstrapped ridge regression estimates of the standard errors are larger in general than the corresponding OLS estimates.

11)

We then fit a lasso regression model with all available covariates except the qualitative covariate.

```
# lasso regression model
library(glmnet)
X = as.matrix(X)
cv.lasso = cv.glmnet(X[train, ], Y[train], alpha = 1)
lasso = glmnet(X[test, ], Y[test], alpha = 1, lambda = cv.lasso$lambda.min)
print("Lasso")
```

```
## [1] "Lasso"
```

```
print(lasso$beta[,1])
```

```
##                 APM       MinimapAttacks                  Age
##        3.524140e-01         0.000000e+00         2.325112e-02
##       ActionLatency         NumberOfPACs         ActionsInPAC
##       -3.862309e-01         9.576663e+00         0.000000e+00
##      SelectByHotkeys    MinimapRightClicks      UniqueUnitsMade
##        3.691743e+00         1.673877e+00         9.879131e-03
##        UniqueHotkeys       GapBetweenPACs          WorkersMade
##       -9.721845e-03        -7.470130e-02         4.419519e-04
##      AssignToHotkeys     ComplexUnitsMade ComplexAbilitiesUsed
##        0.000000e+00         0.000000e+00        -8.878428e+01
##     TotalMapExplored
##        0.000000e+00
```

```
print("OLS")
```

```
## [1] "OLS"
```

```
# corresponding OLS estimates of regression coefficients
print(summary(full)$coefficients[-1, "Estimate"])
```

```
##                  Age                  APM      SelectByHotkeys
##          0.324773654          0.894230520         -0.643132711
##      AssignToHotkeys        UniqueHotkeys       MinimapAttacks
##          0.833670302         -0.009900351          9.595682640
##   MinimapRightClicks         NumberOfPACs       GapBetweenPACs
##         -1.588505913        -12.961438782         -0.083015613
##        ActionLatency         ActionsInPAC     TotalMapExplored
##         -0.444652476         -0.278316113          0.005138449
##          WorkersMade      UniqueUnitsMade     ComplexUnitsMade
##          0.024325403          0.020054520        -77.891734400
## ComplexAbilitiesUsed
##         54.707998567
```

The lasso regression estimates of coefficients are overall smaller in magnitude than the corresponding OLS estimates. Unlike ridge regression, however, lasso regression drove coefficient values to 0 for $MinimapAttacks$, $ActionsInPAC$, $AssignToHotkeys$, $ComplexUnitsMade$, and $TotalMapExplored$.

The lasso regression model could have identified them to have the least significant effect on the response variable or those covariates could be colinear with other covariates.

```
# bootstrap to estimate standard errors
nboot = 10000
beta = matrix(0, nboot, P)
colnames(beta) = colnames(X)

for (i in 1:nboot)
{
  ind = sample(test, replace = TRUE)
  beta[i,] = glmnet(X[ind,], Y[ind], alpha = 1,
                    lambda = cv.lasso$lambda.min)$beta[,]
}
print("Lasso")
```

```
## [1] "Lasso"
```

```
SE = apply(beta, 2, sd)
print(SE)
```

```
##                  APM       MinimapAttacks                  Age
##           0.23438918           3.78744104           0.16106132
##        ActionLatency         NumberOfPACs         ActionsInPAC
##           0.28418359           7.48433373           0.13119757
##      SelectByHotkeys   MinimapRightClicks      UniqueUnitsMade
##           2.49295922           3.30808092           0.01503914
##        UniqueHotkeys       GapBetweenPACs          WorkersMade
##           0.01360094           0.10606512           0.05578254
##      AssignToHotkeys     ComplexUnitsMade ComplexAbilitiesUsed
##           5.41474845         219.74878726         117.43289917
##     TotalMapExplored
##           0.03788234
```

```
print("OLS")
```

```
## [1] "OLS"
```

```
# corresponding OLS estimates of standard errors
print(summary(full)$coefficients[-1, "Std. Error"])
```

```
##                  Age                  APM      SelectByHotkeys
##         8.315612e-02         2.713959e-01         1.719600e+00
##      AssignToHotkeys        UniqueHotkeys       MinimapAttacks
##         3.483776e+00         7.165115e-03         2.472331e+00
##   MinimapRightClicks         NumberOfPACs       GapBetweenPACs
##         1.971721e+00         9.698111e+00         5.846509e-02
##        ActionLatency         ActionsInPAC     TotalMapExplored
##          1.514232e-01         2.434658e-01         2.584779e-02
##           WorkersMade      UniqueUnitsMade     ComplexUnitsMade
##          3.664196e-02         1.019346e-02         1.754158e+02
## ComplexAbilitiesUsed
##          7.092988e+01
```

Although the lasso regression estimates of regression coefficients are generally smaller in magnitude, the bootstrapped lasso regression estimates of the standard errors are larger in general than the corresponding OLS estimates.

12)

We now fit a mixed-effects model with *LeagueIndex* as a random effect. We bring back the raw dataset and use it for fitting the mixed-effects model.

```
# store raw dataset under a different name
library(lme4)
```

```
## Warning: package 'lme4' was built under R version 4.4.3
```

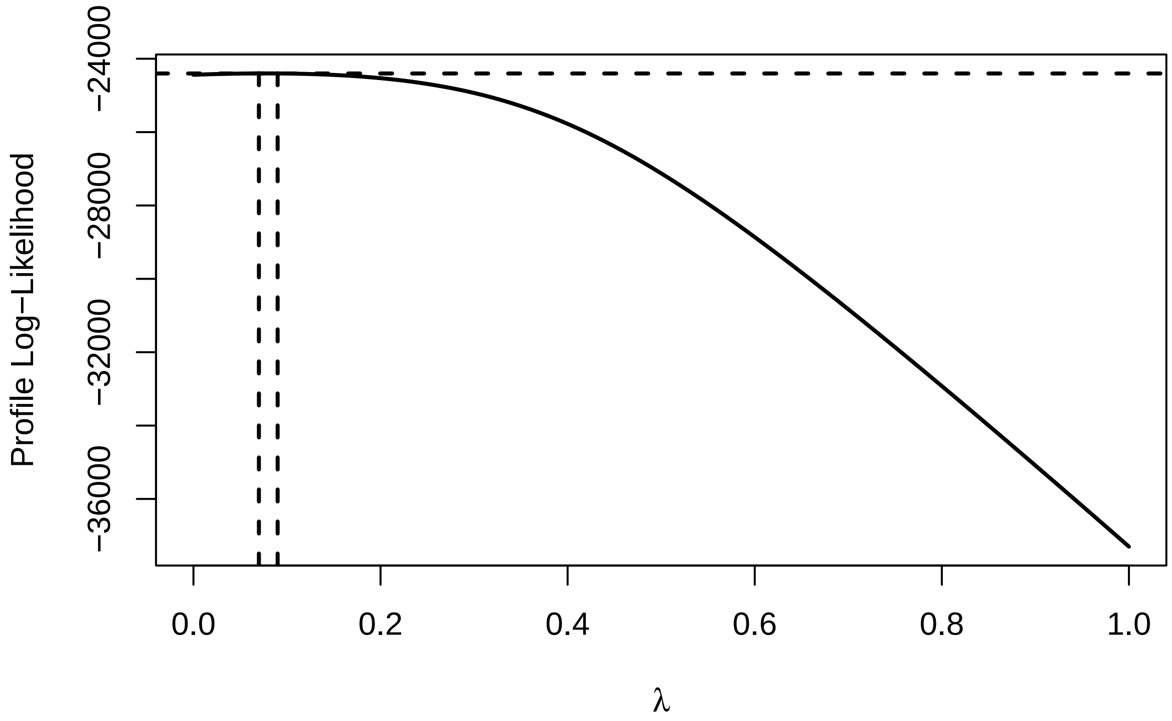
```
starcraft2 = read.table("starcraft.txt", header=TRUE)
Y = starcraft2$TotalHours
X = starcraft2[,-3]
```

We first identify a suitable transformation for the response variable.

```
# identify transformation for response variable
lambda = seq(0, 1, 0.01)
loglik = numeric(101)
starcraft2$LeagueIndex = factor(starcraft2$LeagueIndex)
for (i in 1:101) {
  if (lambda[i] == 0) {
  Ypower = log(Y)
  }
  else {
  Ypower = (Y^lambda[i] - 1)/lambda[i]
  }
  power = lmer(Ypower ~ . - LeagueIndex + (1 | LeagueIndex), X, REML = FALSE)
  loglik[i] = logLik(power)[1] + (lambda[i] - 1) * sum(log(Y))
}
CI = range(lambda[loglik > max(loglik) - qchisq(0.95, 1)/2])
print(CI)
```

```
## [1] 0.07 0.09
```

```
# draw the profile likelihood curve
plot(lambda, loglik, "l", xlab = expression(lambda),
     ylab = "Profile Log-Likelihood", lwd = 2)
abline(h = max(loglik) - qchisq(0.95, 1)/2, lty = 2, lwd = 2)
abline(v = CI, lty = 2, lwd = 2)
```



The 95% confidence interval for lambda is [0.07, 0.09]. Since the values within the 95% confidence interval are not interpretable, we decide to choose 0 for λ since the log likelihood value at 0 appears to be very close to the maximum log likelihood value in the profile-likelihood curve.

We then fit a mixed-effects model with log transformed response variable and $LeagueIndex$ as the random effect and compute the interclass correlation coefficient.

```
# fit a mixed-effects model
fit = lmer(log(TotalHours) ~ . - LeagueIndex + (1 | LeagueIndex),
           data = starcraft2)

# compute the interclass correlation coefficient
sigma = data.frame(summary(fit)$varcor)$sdcor
names(sigma) = data.frame(summary(fit)$varcor)$grp
ICC = sigma^2/sum(sigma^2)
print(ICC)
```

```
##  LeagueIndex     Residual
## 4.667872e-10 1.000000e+00
```

The interclass correlation coefficient is close to 0 for $LeagueIndex$, indicating that almost all variation in the response variable can be explained by the residuals. Therefore, modeling $LeagueIndex$ as a random effect may not be necessary.

13)

Ignoring all random effects from now on, we apply multiple comparisons methods to compare the average response value across different levels defined by $LeagueIndex$ at 95% global confidence level.

```r
# mean comparison
alpha = 0.05
Y = starcraft$TotalHours
Q = factor(starcraft$LeagueIndex)
n = length(Q)
K = length(levels(Q))

mu = aggregate(TotalHours ~ LeagueIndex, starcraft, mean)[,2]
sigma = sqrt(sum((Y - mu[Q])^2) / (n - K))
SE = sigma * sqrt(2 * K / n)

diff = outer(mu, mu, "-")
diff = diff[lower.tri(diff)]
lab = outer(levels(Q), levels(Q), function(x, y) {paste0(x, "-", y)})
names(diff) = lab[lower.tri(lab)]

CI = cbind(diff - qt(1 - alpha/2, n - K) * SE,
           diff + qt(1 - alpha/2, n - K) * SE)
colnames(CI) = c("Lower Bound", "Upper Bound")
print(CI)
```

```
##     Lower Bound Upper Bound
## 2-1   0.2471588   0.4515509
## 3-1   0.7006399   0.9050320
## 4-1   0.9230754   1.1274675
## 5-1   1.2619876   1.4663797
## 6-1   1.5392132   1.7436053
## 7-1   1.9152471   2.1196392
## 3-2   0.3512851   0.5556772
## 4-2   0.5737206   0.7781127
## 5-2   0.9126327   1.1170248
## 6-2   1.1898583   1.3942504
## 7-2   1.5658922   1.7702843
## 4-3   0.1202394   0.3246315
## 5-3   0.4591516   0.6635437
## 6-3   0.7363772   0.9407693
## 7-3   1.1124111   1.3168032
## 5-4   0.2367161   0.4411082
## 6-4   0.5139417   0.7183338
## 7-4   0.8899756   1.0943677
## 6-5   0.1750296   0.3794217
## 7-5   0.5510634   0.7554555
## 7-6   0.2738378   0.4782299
```

The above method computes individual 95% confidence intervals for pairwise comparison of average response values across different levels of $LeagueIndex$.

The following two methods compute simultaneous 95% confidence intervals such that family-wise error rate is less than or equal to 0.05.

```
# Scheffe's method
Scheffe = cbind(diff - sqrt((K - 1) * qf(1 - alpha, K - 1, n - K)) * SE,
                diff + sqrt((K - 1) * qf(1 - alpha, K - 1, n - K)) * SE)
colnames(Scheffe) = c("Lower Bound", "Upper Bound")
print(Scheffe)
```

```
##     Lower Bound Upper Bound
## 2-1  0.16427955   0.5344301
## 3-1  0.61776069   0.9879113
## 4-1  0.84019618   1.2103468
## 5-1  1.17910833   1.5492589
## 6-1  1.45633394   1.8264845
## 7-1  1.83236781   2.2025184
## 3-2  0.26840585   0.6385564
## 4-2  0.49084134   0.8609919
## 5-2  0.82975349   1.1999041
## 6-2  1.10697909   1.4771297
## 7-2  1.48301296   1.8531636
## 4-3  0.03736020   0.4075108
## 5-3  0.37627234   0.7464229
## 6-3  0.65349795   1.0236485
## 7-3  1.02953182   1.3996824
## 5-4  0.15383685   0.5239874
## 6-4  0.43106246   0.8012131
## 7-4  0.80709633   1.1772469
## 6-5  0.09215031   0.4623009
## 7-5  0.46818418   0.8383348
## 7-6  0.19095857   0.5611092
```

Scheffe’s method is, for the most part, more conservative. The 95% confidence intervals returned by the Scheffe’s method are smaller in size compared to the individual 95% confidence intervals.

```
# Tukey's Honestly Significant Difference method
TukeyHSD(aov(TotalHours ~ LeagueIndex, starcraft))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = TotalHours ~ LeagueIndex, data = starcraft)
##
## $LeagueIndex
##          diff         lwr       upr     p adj
## 2-1 0.3493548  0.12573679 0.5729729 0.0000857
## 3-1 0.8028360  0.59318654 1.0124854 0.0000000
## 4-1 1.0252715  0.82350499 1.2270380 0.0000000
## 5-1 1.3641836  1.16226721 1.5661000 0.0000000
## 6-1 1.6414092  1.43443926 1.8483792 0.0000000
## 7-1 2.0174431  1.57604375 2.4588425 0.0000000
## 3-2 0.4534811  0.29087302 0.6160893 0.0000000
## 4-2 0.6759166  0.52360700 0.8282263 0.0000000
## 5-2 1.0148288  0.86232059 1.1673370 0.0000000
## 6-2 1.2920544  1.13291584 1.4511929 0.0000000
## 7-2 1.6680883  1.24699218 2.0891843 0.0000000
## 4-3 0.2224355  0.09149244 0.3533785 0.0000118
## 5-3 0.5613476  0.43017369 0.6925216 0.0000000
## 6-3 0.8385732  0.69974624 0.9774003 0.0000000
## 7-3 1.2146071  0.80075961 1.6284546 0.0000000
## 5-4 0.3389121  0.22074518 0.4570791 0.0000000
## 6-4 0.6161378  0.48952905 0.7427465 0.0000000
## 7-4 0.9921716  0.58226116 1.4020821 0.0000000
## 6-5 0.2772256  0.15037812 0.4040731 0.0000000
## 7-5 0.6532595  0.24327520 1.0632438 0.0000553
## 7-6 0.3760339 -0.03646273 0.7885305 0.1011921
```

The 95% confidence intervals returned by Tukey's Honestly Significant Difference Method are not as conservative as Scheffe's method, but for the most part, are smaller in size than the individual 95% confidence intervals. However, the 95% confidence interval for the average difference between level 7 and 6 defined by $LeagueIndex$ contains 0 while previously it was not the case.

14)

We now apply global testing methods to test the global null hypothesis that all individual t-tests for each covariate are significant. In other words, we test whether at least one covariate has a statistically significant effect on the response variable. To conduct this global test, we first gather the p-values of individual t-tests for each covariate.

```r
# define p-values of individual null hypotheses
alpha = 0.05
fit = lm(TotalHours ~ ., starcraft)
summary(fit)
```

```
##
## Call:
## lm(formula = TotalHours ~ ., data = starcraft)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5700 -0.4129  0.0755  0.4874  7.1887
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.265e+00  1.149e+00   1.971 0.048771 *
## LeagueIndex2          2.668e-01  7.626e-02   3.498 0.000474 ***
## LeagueIndex3          6.775e-01  7.428e-02   9.121  < 2e-16 ***
## LeagueIndex4          8.516e-01  7.526e-02  11.315  < 2e-16 ***
## LeagueIndex5          1.144e+00  8.035e-02  14.233  < 2e-16 ***
## LeagueIndex6          1.389e+00  8.730e-02  15.913  < 2e-16 ***
## LeagueIndex7          1.747e+00  1.643e-01  10.635  < 2e-16 ***
## Age                   2.658e-01  7.912e-02   3.359 0.000790 ***
## APM                   5.515e-01  2.662e-01   2.071 0.038399 *
## SelectByHotkeys      -4.690e-01  1.693e+00  -0.277 0.781717
## AssignToHotkeys      -7.057e+00  3.338e+00  -2.114 0.034602 *
## UniqueHotkeys        -1.549e-02  6.827e-03  -2.269 0.023340 *
## MinimapAttacks        2.582e-01  2.408e+00   0.107 0.914627
## MinimapRightClicks   -1.314e+00  1.871e+00  -0.703 0.482331
## NumberOfPACs         -9.987e+00  9.396e+00  -1.063 0.287944
## GapBetweenPACs        2.670e-02  5.576e-02   0.479 0.632093
## ActionLatency        -6.819e-02  1.457e-01  -0.468 0.639726
## ActionsInPAC         -3.410e-02  2.365e-01  -0.144 0.885396
## TotalMapExplored      3.064e-02  2.456e-02   1.247 0.212321
## WorkersMade          -5.217e-02  3.509e-02  -1.487 0.137225
## UniqueUnitsMade       2.247e-02  9.685e-03   2.320 0.020399 *
## ComplexUnitsMade     -1.665e+02  1.667e+02  -0.999 0.317896
## ComplexAbilitiesUsed  2.240e+01  6.736e+01   0.333 0.739491
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.7945 on 3315 degrees of freedom
## Multiple R-squared:  0.2719, Adjusted R-squared:  0.267
## F-statistic: 56.26 on 22 and 3315 DF,  p-value: < 2.2e-16
```

```r
m = dim(model.matrix(fit))[2]
pvalues = summary(fit)$coefficients[,4]
```

```r
# Bonferroni test
Bonf = min(pvalues)
print(Bonf < alpha / m)
```

```
## [1] TRUE
```

```r
# Fisher's Combination test
Fisher = -2 * sum(log(pvalues))
print(Fisher > qchisq(1 - alpha, 2 * m))
```

```
## [1] TRUE
```

```r
# Simes Test
Simes = sum(pvalues[order(pvalues)] <= (1:m) * alpha / m)
print(Simes > 0)
```

```
## [1] TRUE
```

The three global tests provide strong evidence to reject the global null hypothesis, indicating that at least one covariate has a statistically significant effect on the response variable. The convergence of results across these tests further strengthens the validity of this conclusion.

Examining the p-values for all covariates reveals multiple extremely small values, suggesting that the Bonferroni correction may be overly conservative. However, the fact that Bonferroni still rejects the null hypothesis implies that the effects of the covariates are strong enough to remain significant despite the correction.

The OLS estimate and heteroskedasticity-consistent estimate of the covariance matrix show that all covariates have non-zero covariance with each other, suggesting that the individual t-tests are likely not independent. Nevertheless, since all three tests reached the same conclusion, any possible dependence does not appear to have significantly distorted the significance of the covariates.

Given the presence of multiple extremely small and moderately small p-values, Simes' test may be the most appropriate choice. However, the covariance matrix of the covariates contained several negative values, which suggests that the p-values could exhibit non-negative dependence. To confirm the validity of applying Simes' test, further analysis is needed to determine whether the p-values satisfy the non-negative dependence assumption.

15)

Lastly, we apply multiple testing methods to determine which covariates have a statistically significant effect on your response variable by controlling the FWER or FDR at level 5%.

```r
# Holm-Bonferroni procedure
HB = numeric(m)
for (j in 1:m)
{
  HB[j] = min(max((m - (1:j) + 1) * pvalues[order(pvalues)][1:j]), 1)
}
HB = HB[order(order(pvalues))]
HB = p.adjust(pvalues, "holm")
sum(HB < alpha)
```

```
## [1] 7
```

```r
# Benjamini-Hochberg procedure
BH = numeric(m)
for (j in 1:m)
{
  BH[j] = min(min(m * pvalues[order(pvalues)][j:m] / (j:m)), 1)
}
BH = BH[order(order(pvalues))]
BH = p.adjust(pvalues, "BH")
sum(BH < alpha)
```

```
## [1] 7
```

```r
# Benjamini-Yekutieli procedure
BY = numeric(m)
for (j in 1:m)
{
  BY[j] = min(min(m * sum(1 / (1:m)) * pvalues[order(pvalues)][j:m] / (j:m)), 1)
}
BY = BY[order(order(pvalues))]
BY = p.adjust(pvalues, "BY")
sum(BY < alpha)
```

```
## [1] 7
```

Holm-Bonferroni procedure, Benjamini-Hochberg procedure, and Benjamini-Yekutieli procedure provide a more detailed conclusion compared to part 14. While part 14 only allowed us to determine whether to reject or accept the global null hypothesis that at least one covariate has a statistically significant effect on the response variable—these procedures now enable us to identify that seven specific covariates have statistically significant effects. Additionally, they ensure that the family-wise error rate (FWER) or false discovery rate (FDR) remains controlled at less than or equal to 0.05.

Holm-Bonferroni procedure is always applicable and ensures that the family-wise error rate is less than or equal to 0.05. However, it can be very conservative.

Benjamini-Hochberg procedure is only applicable for non-negative dependence. As mentioned above, we are not certain if the p-values have non-negative dependence and given the negative values in the covariance matrix, it is possible that they have negative dependence.

Benjamini-Yekutieli procedure is always applicable, but can be more conservative than Holm-Bonferroni if there is no negative dependence.

Although we do not know if the p-values have negative or non-negative dependence or any dependence at all, since the conclusions returned by the three tests converged, we can reasonably conclude that the detected significance of covariates on the response variable is valid.

16)

We now take all our previous analyses into account and make suggestions for the final model.

We applied square root and log transformations to various covariates, including log transformation to the response variable, $TotalHours$. Although they handled skewness and heteroskedasticity issues to some extent, the distribution of residuals were still not normally distributed. We still recommend to apply the same set of transformations to the covariates and the response variable.

We tested the interaction between $LeagueIndex$ and APM. Although the interaction between the second level of $LeagueIndex$ and APM was significant, the overall interaction between $LeagueIndex$ and APM was not statistically meaningful. Therefore, we recommend to not include the interaction between $LeagueIndex$ and APM.

We then tested Jackknife method against nonparametric bootstrap method. The estimated standard errors of regression coefficients returned by the Jackknife and nonparametric bootstrap methods were very different. This is most likely because the Jackknife method is more sensitive to outliers. Despite the previously noted normality violation and skewness of some covariates, the large sample size of the data ensured robust inference using nonparametric boostrap method. Therefore, we recommend using nonparametric bootstrap method over Jackknife method.

To validate the model, we performed a nonparametric permutation F-test. It confirmed a statistically significant relationship between the response variable and covariates, although its p-value was larger than the OLS F-test p-value most likely due to violations of OLS assumptions. A permutation t-test showed APM remains significant, although with a slightly higher p-value than the OLS t-test, likely due to the same assumption issues.

We expanded on this by utilizing multiple hypothesis testing approaches: Bonferroni, Fisher's Combination, and Simes' tests. We tested if at least one covariate has a statistically significant effect on the response variable. Multiple covariates showed extremely small p-values, which suggested that Bonferroni test might be too conservative. However, all three tests provided strong evidence to reject the global null hypothesis. We also executed Holm-Bonferroni, Benjamini-Hochberg, and Benjamini-Yekutieli procedures, which showed that there are seven specific covariates with statistically significant effects.

Therefore, we conclude that the OLS model is statistically significant and that multiple covariates in the full model have significant effects

We then examined heteroskedasticity. We conducted the Breusch-Pagan test for quantitative covariates and the Brown-Forsythe test for the qualitative covariate, $LeagueIndex$, both confirming non-constant variance in residuals. While weighted least squares (WLS) was tested, it did not significantly improve the model, likely due to insufficient explanatory covariates beyond Age. Given that heteroskedasticity-consistent standard errors closely align with OLS estimates and confidence intervals remain similar, the degree of heteroskedasticity does not appear severe enough to distort inference. We would recommend to not be too concerned about heteroskedasticity in the final model.

For variable selection, valid post-selection inference selected APM, $MinimapAttacks$, Age, and $ActionLatency$ among all covariates except $LeagueIndex$. Therefore, we would recommend including the four covariates in the final model.

Ridge and lasso regression demonstrated smaller coefficient magnitudes than OLS, though bootstrapped standard errors were larger, possibly due to outliers and heteroskedasticity of residuals. We recommend ridge regression if it is ideal to retain all covariates, but lasso regression if some covariates encode repetitive information and hence not all covariates are needed.

We also fit a mixed-effects model with *LeagueIndex* as a random effect showed that almost all variation in the response variable can be attributed to residuals. This suggests that modeling *LeagueIndex* as a random effect may not be necessary.

Lastly, we applied multiple comparisons methods to compare the average response value across different levels defined by *LeagueIndex* at 95% global confidence level. The 95% confidence intervals returned by Tukey's Honestly Significant Difference Method were not as conservative as Scheffe's method, but still were smaller in size than the individual 95% confidence intervals for the most part. For pairwise means comparisons, we would recommend Tukey's Honestly Signficant Difference Method.

We have recommended which transformation methods, interaction effects, covariates, nonparametric bootstrap methods, and multiple comparisons methods to incorporate in the final method. We also remarked on when violations of OLS assumptions such as normality and heteroskedasticity affected test results. We noted that in some test results, the large sample size contributed to their convergence to the corresponding OLS estimates. These provide insights into future stability and validity of inferences on the starcraft data.