Case Study 1

Jonathan Sneh, Ishani Tarafdar, Georges Durand, Raul Higareda

2023-03-28

```
library(ellipse)
```

```
## Warning: package 'ellipse' was built under R version 4.1.3
```

```
##
## Attaching package: 'ellipse'
```

```
## The following object is masked from 'package:graphics':
##
##     pairs
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 4.1.3
```

```
library(faraway)
```

```
## Warning: package 'faraway' was built under R version 4.1.3
```

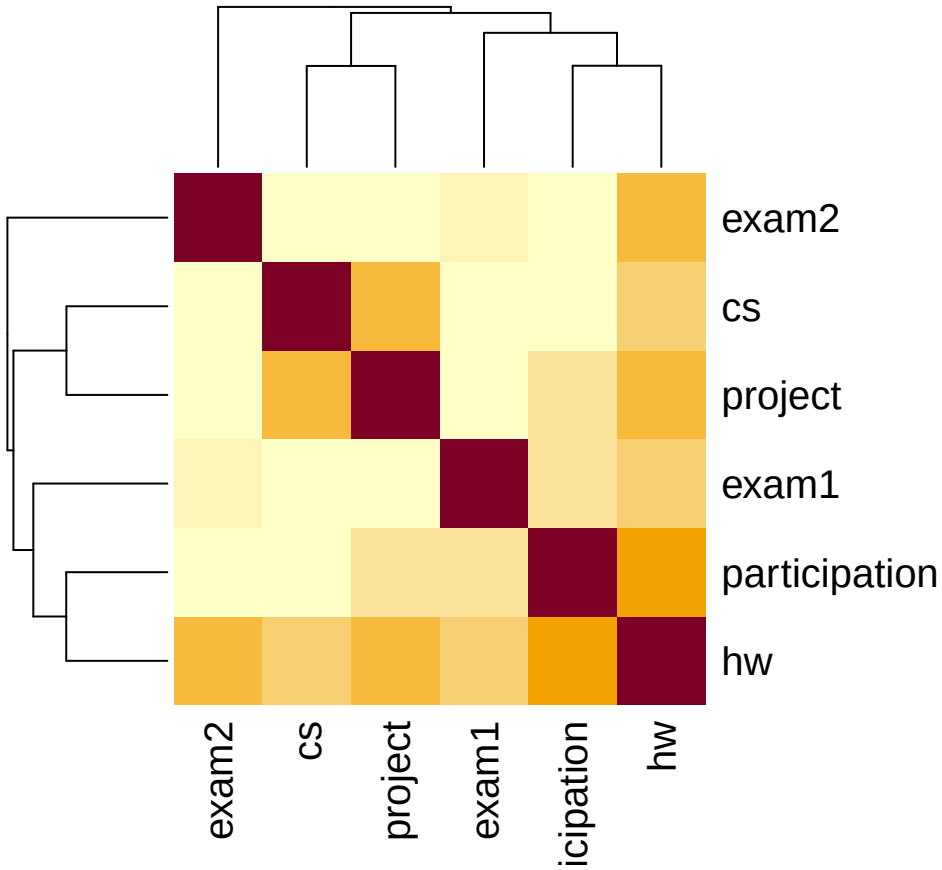
Data Explorations and Summary Statistics

```
grades <- read.csv("grades.csv", header=TRUE)
grades <- grades[1:6]
dim(grades)
```

```
## [1] 275   6
```

Model Selection

```
cor_matrix <- cor(grades)
heatmap(cor_matrix, symm = TRUE)
```



Looking at the correlation matrix, none of the variables are strongly correlated with another variable, so we do not need to drop any variables based on what we see from the correlation matrix.

We want to make a model with 95% confidence (i.e. $\alpha = 0.05$)

```
grades.mlr <- lm(exam2 ~ hw + cs + participation + exam1 + project, data=grades)
summary(grades.mlr)
```

```
##
## Call:
## lm(formula = exam2 ~ hw + cs + participation + exam1 + project,
##     data = grades)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.116  -9.270   3.431  10.486  39.096
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   25.51074    6.52460   3.910 0.000117 ***
## hw             0.50233    0.07506   6.692 1.27e-10 ***
## cs             0.02096    0.07030   0.298 0.765805
## participation -0.09622    0.06259  -1.537 0.125403
```

```
## exam1          0.22437    0.06954   3.226 0.001410 **
## project        0.00885    0.04804   0.184 0.853973
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.59 on 269 degrees of freedom
## Multiple R-squared:  0.3249, Adjusted R-squared:  0.3124
## F-statistic:  25.9 on 5 and 269 DF,  p-value: < 2.2e-16
```

Based on the full model summary above, we may want to look into dropping project, cs, and participation from the dataset since the t-values in the summary output for project, cs, and participation are all small, meaning that they're likely up to chance.

However, the individual t-tests do not tell us enough information to drop multiple predictors from our model at a time.

So, we can start by dropping an individual predictor from our model. We will check `project`.

Our null and alternative hypothesis are as follows.

$$\begin{cases} H_0, & \beta_{project} = 0 \\ H_A, & \beta_{project} \neq 0 \end{cases}$$

By conducting an individual t-test (which can be found in our summary output), we can see that the t-test statistics for project is 0.397.

Thus, we can see that the p-value for project is 0.692. $p = 0.692 > 0.05 = \alpha$. Thus, we fail to reject the null hypothesis, meaning that it is likely that $\beta_{project} = 0$. In other words, we can drop project from our model.

This leaves us with the following reduced model:

```
grades.reducedmlr1 = lm(exam2~exam1 + hw + cs + participation, data=grades)
summary(grades.reducedmlr1)
```

```
##
## Call:
## lm(formula = exam2 ~ exam1 + hw + cs + participation, data = grades)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.129  -9.371   3.396  10.476  39.564
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   25.36389    6.46413   3.924 0.000111 ***
## exam1          0.22444    0.06942   3.233 0.001377 **
## hw             0.50605    0.07216   7.013 1.88e-11 ***
## cs             0.02568    0.06535   0.393 0.694589
## participation -0.09430    0.06161  -1.531 0.127027
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.56 on 270 degrees of freedom
## Multiple R-squared:  0.3249, Adjusted R-squared:  0.3148
## F-statistic: 32.48 on 4 and 270 DF,  p-value: < 2.2e-16
```

```
anova(grades.mlr, grades.reducedmlr1)
```

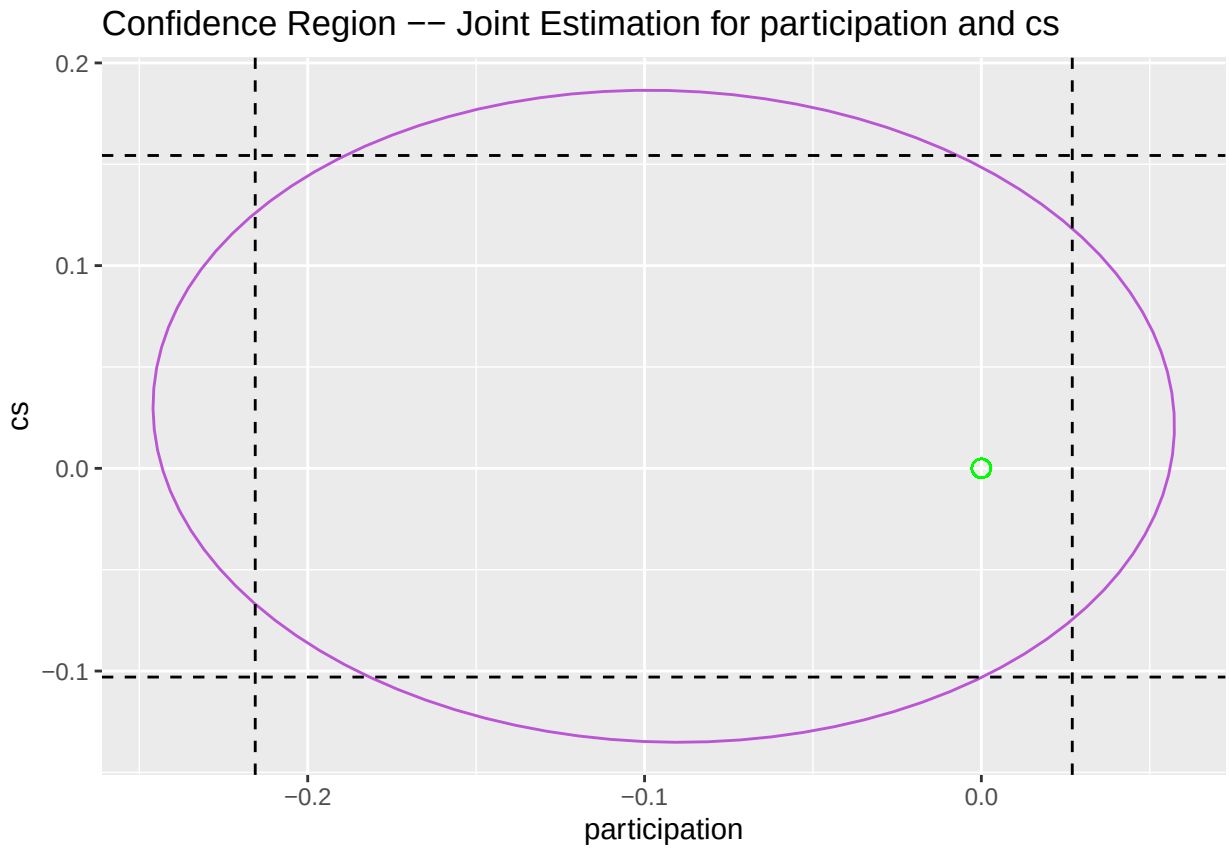
```
## Analysis of Variance Table
##
## Model 1: exam2 ~ hw + cs + participation + exam1 + project
## Model 2: exam2 ~ exam1 + hw + cs + participation
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1    269 57240
## 2    270 57247 -1   -7.2221 0.0339  0.854
```

From the summary, We can see that the p-values for cs and participation have changed. They are still high, so we can conduct a different test.

$$\begin{cases} H_0, & \beta_{participation} = \beta_{cs} = 0 \\ H_A, & \text{Either } \beta_{participation} \text{ or } \beta_{cs} \text{ is not equal to zero} \end{cases}$$

We can draw the confidence region (as an ellipse) for both participation and for cs. If the point (0,0) falls inside of our confidence region, then it is likely that both the coefficients β_{cs} and $\beta_{participation}$ are zero—as according to the null hypothesis.

```
intervals <-confint(grades.reducedmlr1)
cr_ellipse <- ellipse(grades.reducedmlr1, c(4,5), level=0.95)

par_interval <- confint(grades.reducedmlr1, level = 0.95, 'participation')
cs_interval <- confint(grades.reducedmlr1, level = 0.95, 'cs')

cr_df <- as.data.frame(cr_ellipse)
cr_plot <-
ggplot(data=cr_df, aes(x=participation, y=cs)) +
  ggtitle("Confidence Region -- Joint Estimation for participation and cs") +
  geom_path(aes(x=participation,y=cs), colour='mediumorchid') +
  geom_point(x=coef(grades.reducedmlr1)[2], y=coef(grades.reducedmlr1)[3],
             shape=3, size=3, colour='mediumorchid') +
  geom_hline(yintercept = cs_interval[1], lty=2) +
  geom_hline(yintercept = cs_interval[2], lty=2) +
  geom_vline(xintercept = par_interval[1], lty=2) +
  geom_vline(xintercept = par_interval[2], lty=2)+
  geom_point(x=0, y=0, shape=1, size=3, colour='green')

plot(cr_plot)
```



As we can see, the origin—which is the green dot—falls inside the confidence region. Thus, it is likely enough that both β_{cs} and $\beta_{participation}$ are zero. Therefore, we can drop them both from our model.

So, our final model (before diagonistcs) is the following:

```
grades.reducedmlr = lm(exam2 ~ exam1 + hw, data=grades)
summary(grades.reducedmlr)
```

```
##
## Call:
## lm(formula = exam2 ~ exam1 + hw, data = grades)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.150  -9.195   3.301  10.349  40.987
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.64208    5.61643   4.387 1.64e-05 ***
## exam1        0.20858    0.06780   3.077  0.00231 **
## hw           0.46396    0.05768   8.044 2.69e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.57 on 272 degrees of freedom
## Multiple R-squared:  0.3187, Adjusted R-squared:  0.3137
## F-statistic: 63.62 on 2 and 272 DF,  p-value: < 2.2e-16
```

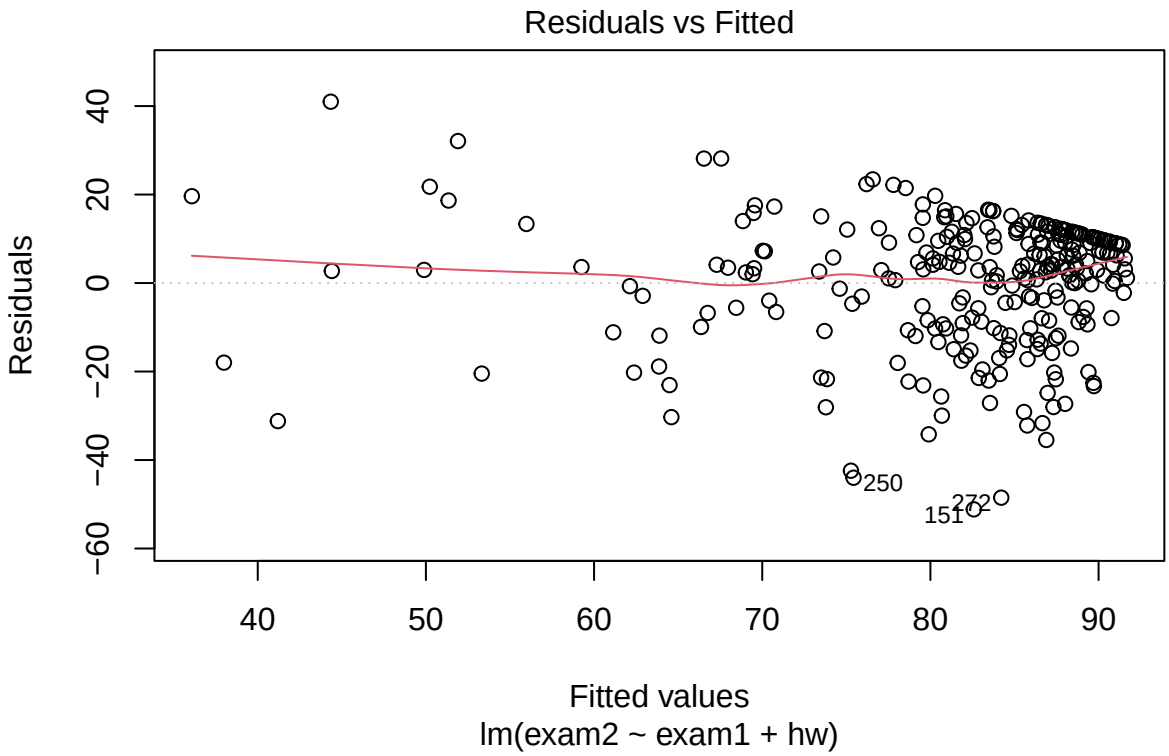
Unusual Observations and Model Assumptions

Now we can analyze the final model for unusual observations and check for deviations from the model assumptions.

Constant variances

First, we can check the model assumption for constant variances by checking the residual vs. fitted plot.

```
plot(grades.reducedmlr, which=1)
```

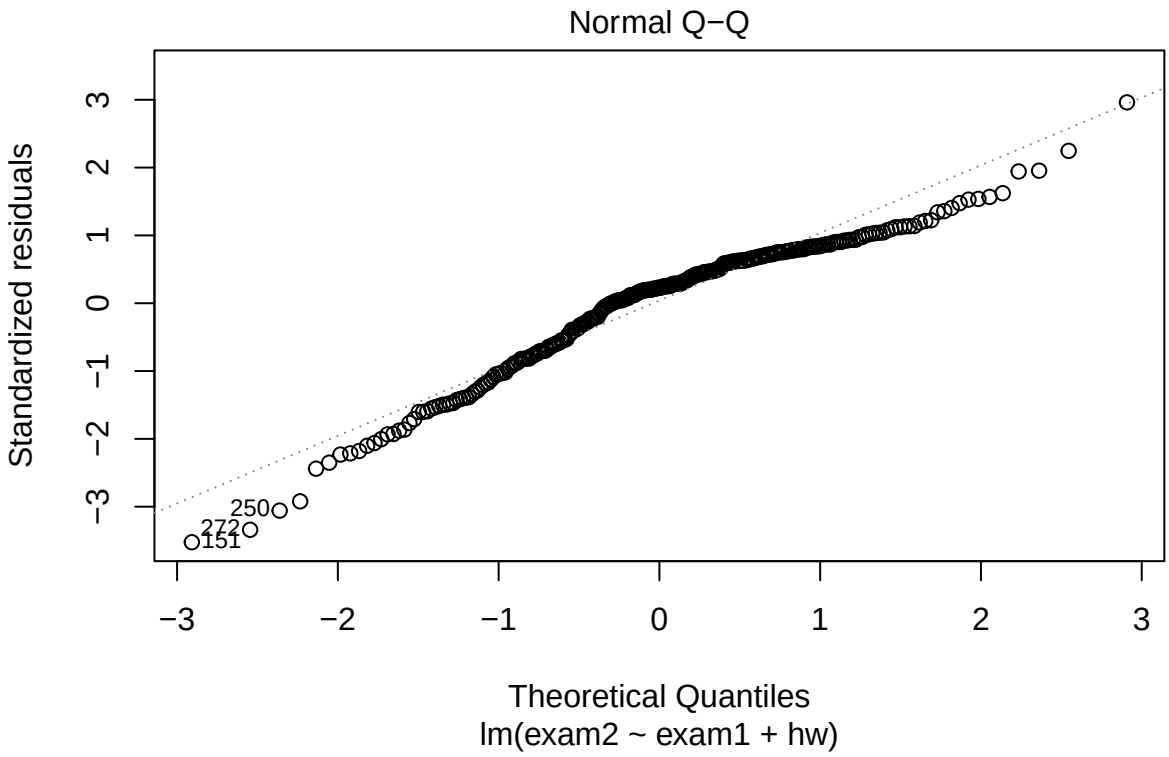


From the residuals vs. fitted plots, we can see the the assumptions for constant variance are not met because the residuals are not evenly disributed around the 0 line, and seem to decrease in magnitude as the residuals increase.

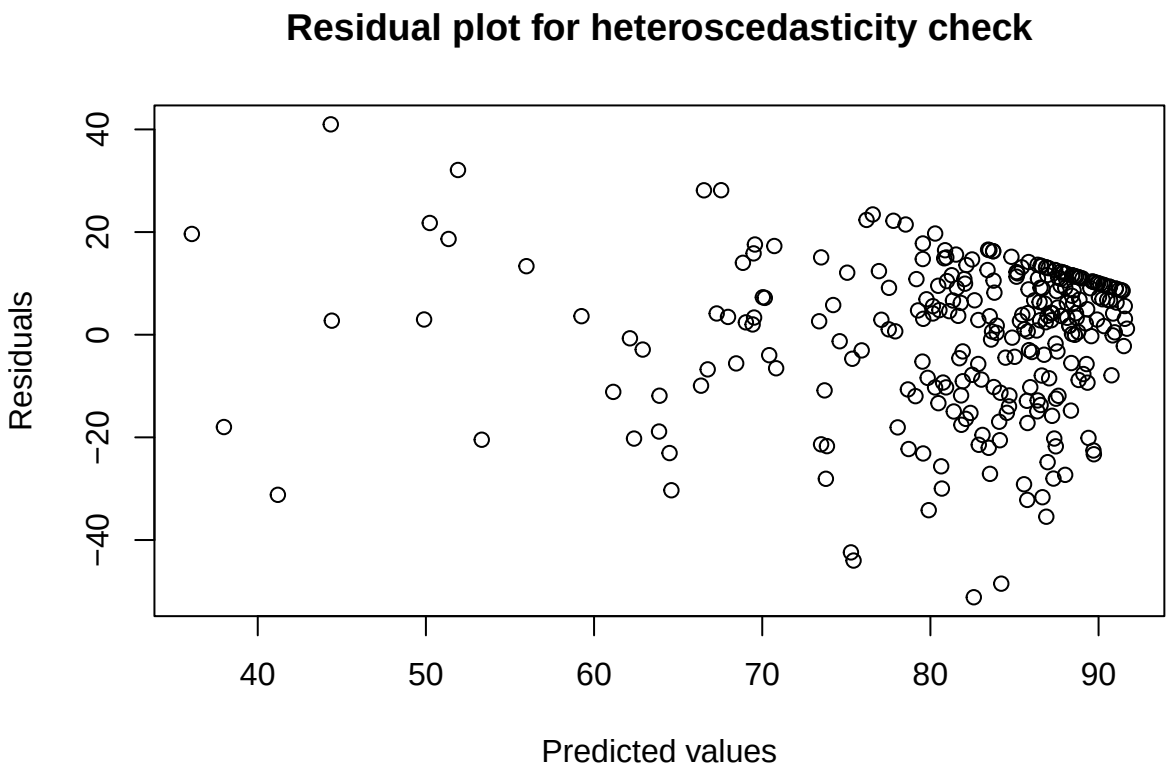
Normality

Next, we can chck for normaltiy of the residuals by creating a QQ plot.

```
plot(grades.reducedmlr, which=2)
```

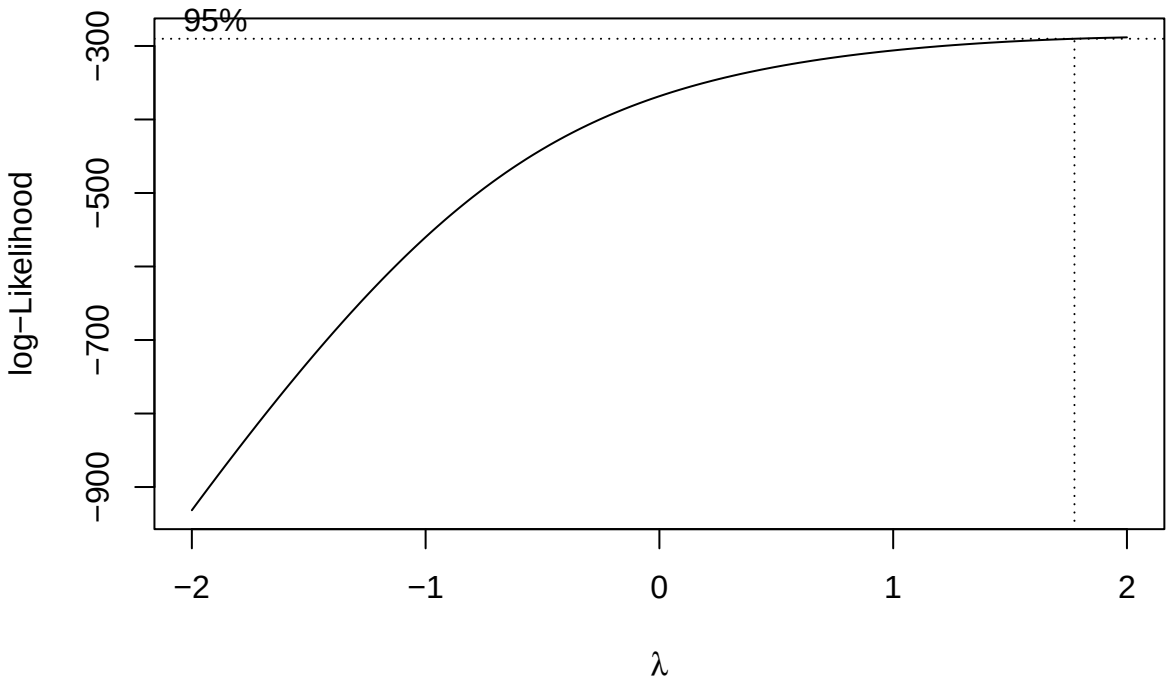



```
predicted_values <- predict(grades.reducedmlr)
plot(x = predicted_values, y = residuals(grades.reducedmlr),
     xlab = "Predicted values", ylab = "Residuals",
     main = "Residual plot for heteroscedasticity check")
```



From the QQ plot, we can see that we seem to have departures from the normality assumption as points along the edges of the plot don't follow the straight line. We can attempt to remedy this and reduce the non-normality of the errors by performing a Box-Cox transformation.

```
grades.transformation = boxcox(grades.reducedmlr, lambda=seq(-2, 2, length=400))
```

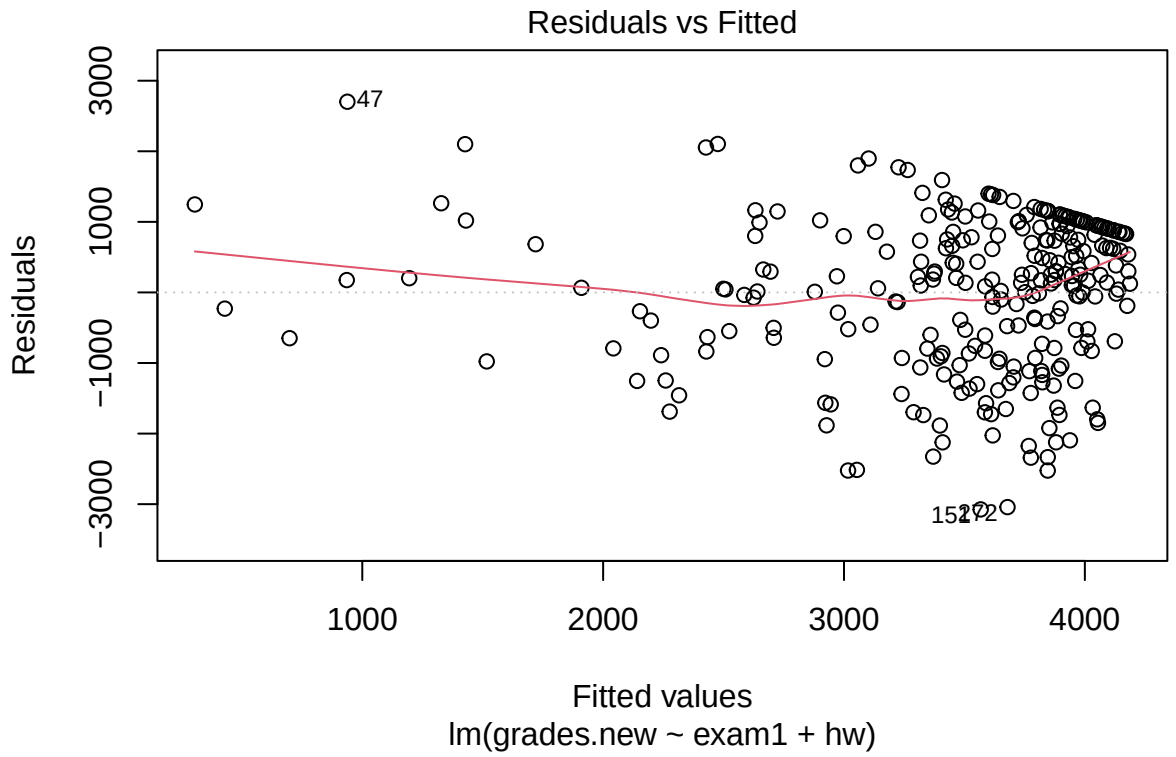


Looking at the Box-Cox plot, the optimal λ is somewhere between 1.5 and 2. We can try both to see which provides better results.

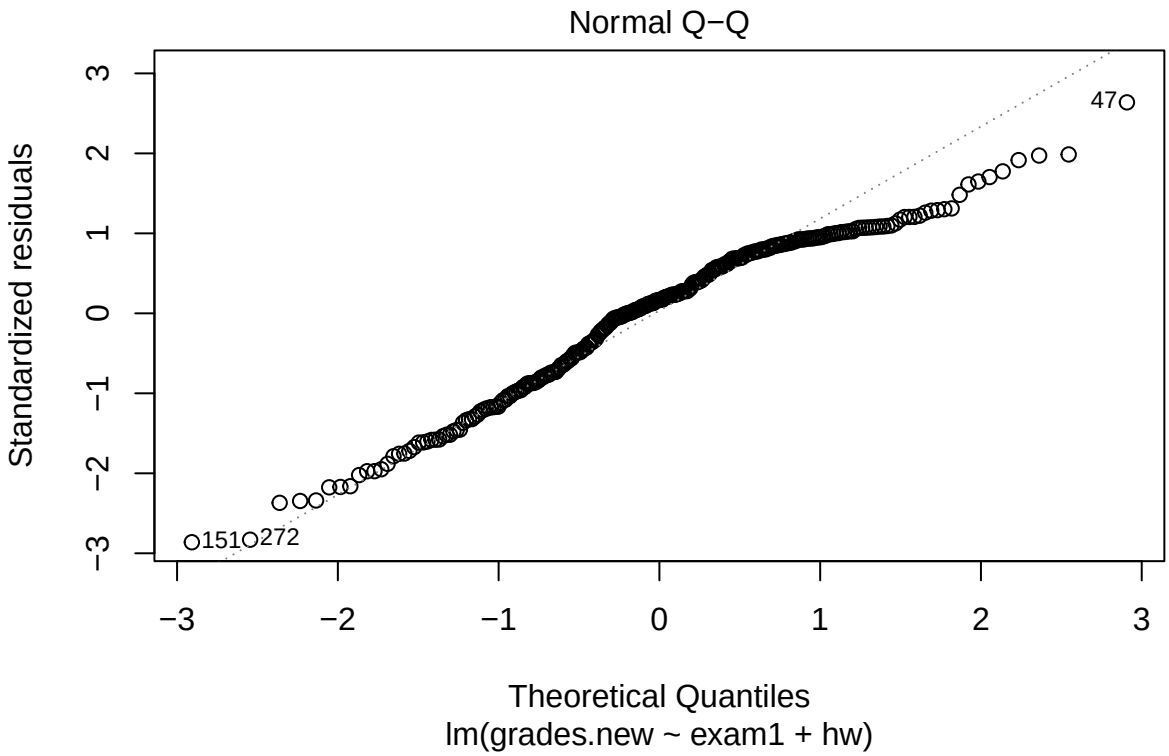
```
grades2 = grades
grades2$grades.new = ('^'(grades2$exam2,2)-1) / 2
grades.mlr.tr = lm(grades.new ~ exam1 + hw, data=grades2[])
summary(grades.mlr.tr)
```

```
##
## Call:
## lm(formula = grades.new ~ exam1 + hw, data = grades2[])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3074.2  -794.3   175.5   869.6  2702.0
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -526.872    415.703  -1.267    0.206
## exam1         15.659      5.018   3.120    0.002 **
## hw            31.640      4.269   7.412 1.58e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1079 on 272 degrees of freedom
## Multiple R-squared:  0.2932, Adjusted R-squared:  0.2881
## F-statistic: 56.43 on 2 and 272 DF,  p-value: < 2.2e-16
```

```
plot(grades.mlr.tr, which=1)
```




```
plot(grades.mlr.tr, which=2)
```



From the fitted vs. residual plot and QQ plot, the box-cox tranformation did not really fix the assumptions for constant variance and normality.

Serial Dependence

It is not possible to check serial dependence for this model because there is no order or time value associated with the data points.

Unusual Observations

High Leverage points Here we calculate which samples are our high leverage points

```
grades.leverages = lm.influence(grades.reducedmlr)$hat
head(grades.leverages)
```

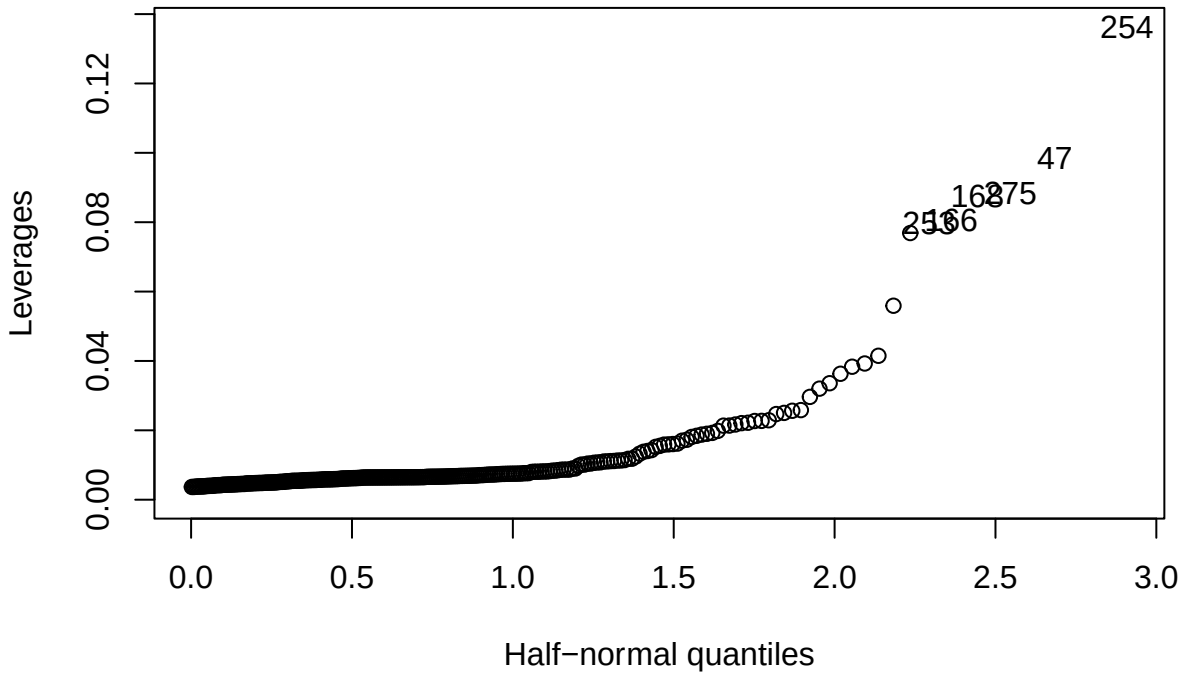
```
##           1           2           3           4           5           6
## 0.007424924 0.007274227 0.006206353 0.006622878 0.005729212 0.005620944
```

```
n = dim(grades)[1]
p = length(variable.names(grades.reducedmlr))

grades.leverages.high = grades.leverages[grades.leverages>2*p/n]
grades.leverages.high
```

```
##          33          34          39          45          47          50          51
## 0.02272018 0.02206615 0.02583001 0.03926475 0.09845203 0.03631088 0.05592355
##         138         153         159         160         162         164         165
## 0.04149202 0.02290199 0.03202090 0.03833443 0.02218738 0.02563467 0.02268266
##         166         167         168         169         250         253         254
## 0.08057361 0.03358242 0.08762130 0.07687729 0.02502428 0.07983386 0.13633458
##         273         274         275
## 0.02469497 0.02964430 0.08837007
```

Here we plot the leverage points.

```
halfnorm(grades.leverages, 6, labs=as.character(1:length(grades.leverages)), ylab="Leverages")
```



Here we calculate the percent of observations the high leverage points make up.

```
length(grades.leverages.high)/n
```

```
## [1] 0.08727273
```

We observe that we have 23 high leverage points, representing about 9% of the observations. These observations are far from the rest and are flagged in the halfnorm plot as well.

Now we need to determine whether the points are good or bad high leverage points. We do this by calculating the IQR for our dependent variable Exam 2 in our original (full) data frame and use this metric to identify the high-leverage observations that don't "follow the pattern of the data".

```r
# Calculate the IQR for the dependent variable
IQR_y = IQR(grades$exam2)

# Define a range with its lower limit being (Q1 - IQR) and upper limit being (Q3 + IQR)
QT1_y = quantile(grades$exam2,0.25)
QT3_y = quantile(grades$exam2,0.75)

lower_lim_y = QT1_y - IQR_y
upper_lim_y = QT3_y + IQR_y

vector_lim_y = c(lower_lim_y,upper_lim_y)

# Range for y variable
vector_lim_y
```

```
##    25%    75%
##  46.86 120.57
```

```r
# Extract observations with high leverage points from the original data frame
grades.highlev = grades[grades.leverages > 2*p/n,]

# Select only the observations with leverage points outside the range
grades.highlev_lower = grades.highlev[grades.highlev$Y < vector_lim_y[1], ]
grades.highlev_upper = grades.highlev[grades.highlev$Y > vector_lim_y[2], ]
grades.highlev2 = rbind(grades.highlev_lower, grades.highlev_upper)
grades.highlev2
```

```
## [1] exam2         exam1         project       cs            hw
## [6] participation
## <0 rows> (or 0-length row.names)
```

From the above calculations, there are 0 bad high leverage points, so all the high leverage points are 'good' high leverage points. The good high leverage points are points where the value of y follows the pattern of the rest of the data but the x is far away from the sample mean.`project` None of our high-leverage points are 'bad' high leverage points, where the y (exam2) doesn't follow the pattern of the rest of the data.

Outliers First, we compute the outliers of the full model

```r
grades.leverages_full = lm.influence(grades.mlr)$hat
head(grades.leverages_full)
```

```
##           1           2           3           4           5           6
## 0.008115635 0.007343383 0.006359665 0.007247987 0.007358686 0.006143024
```

```r
n = dim(grades)[1]
p = length(variable.names(grades.mlr))

# Computing Studentized Residuals
grades.resid_full = rstudent(grades.mlr);
```

```
# Critical value WITH Bonferroni correction
bonferroni_cv = qt(.05/(2*n), n-p-1)
grades.resid_full.sorted = sort(abs(grades.resid_full), decreasing=TRUE)[1:10]
print(grades.resid_full.sorted)
```

```
##      151      272      250       47      247      227      241      226
## 3.674375 3.431503 3.008115 2.971698 2.937787 2.463875 2.388945 2.222698
##       45      217
## 2.221427 2.182470
```

```
grades.outliers_full = grades.resid_full.sorted[abs(grades.resid_full.sorted) > abs(bonferroni_cv)]
print(grades.outliers_full)
```

```
## named numeric(0)
```

```
length(grades.outliers_full)
```

```
## [1] 0
```

We see there are no outliers in the full model.

Then, we compute the outliers of our final reduced model.

```
grades.leverages = lm.influence(grades.reducedmlr)$hat
head(grades.leverages)
```

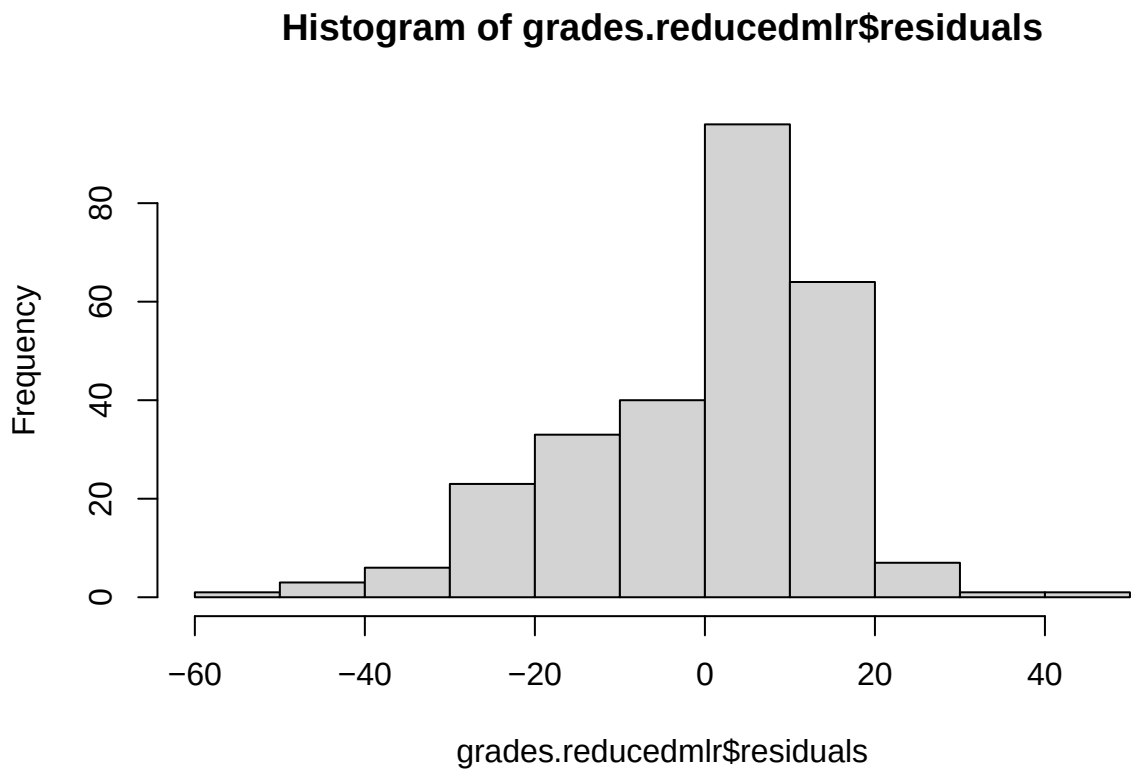
```
##           1           2           3           4           5           6
## 0.007424924 0.007274227 0.006206353 0.006622878 0.005729212 0.005620944
```

```
n = dim(grades)[1]
p = length(variable.names(grades.reducedmlr))

# Computing Studentized Residuals
grades.resid = rstudent(grades.reducedmlr);

# Critical value WITH Bonferroni correction
bonferroni_cv = qt(.05/(2*n), n-p-1)
grades.resid.sorted = sort(abs(grades.resid), decreasing=TRUE)[1:10]
grades.outliers = grades.resid.sorted[abs(grades.resid.sorted) > abs(bonferroni_cv)]
print(grades.outliers)
```

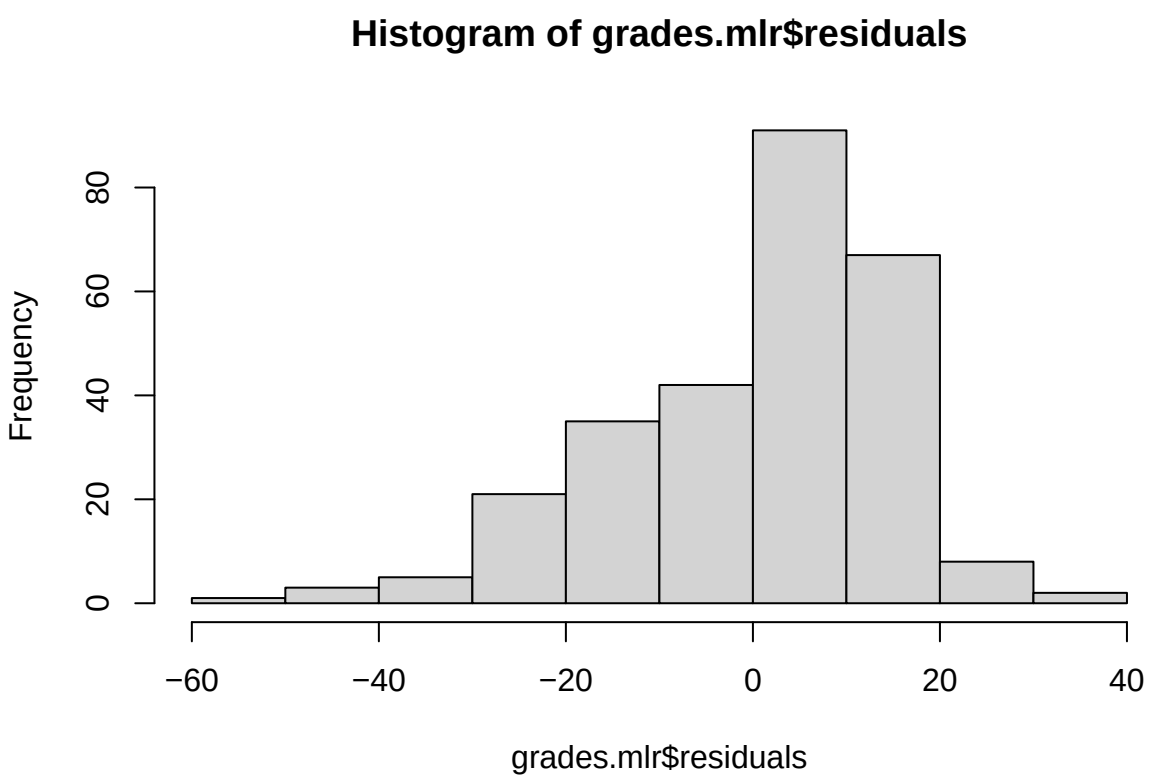
```
## named numeric(0)
```

As we can see, there are no outliers in either the reduced or the full model.

```
hist(grades.reducedmlr$residuals)
```




```
hist(grades.mlr$residuals)
```



Looking at the figures above, we used a histogram to plot the values of residuals in our full model and our reduced model. Both of them do not have any obvious outliers, which agrees with our above analysis of the outliers. Both the histograms look quite similar. The tails seem slightly more symmetric in the reduced model, but not enough to claim that the reduced model fits the normal distribution better.

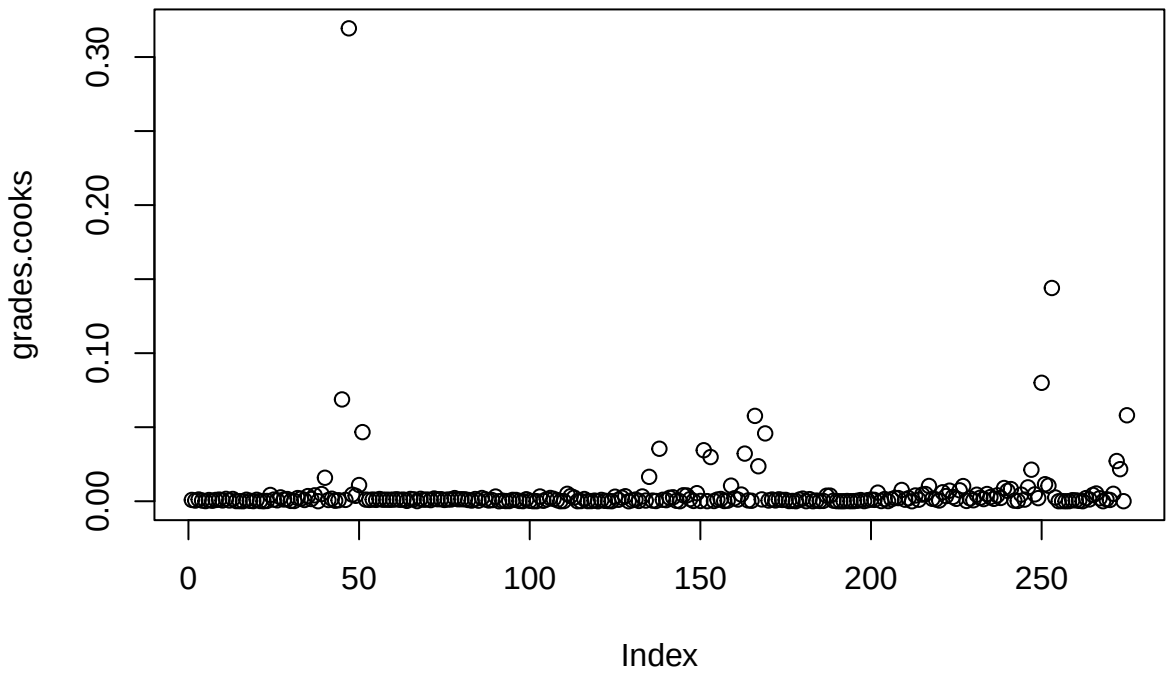
Influential Observations Here calcuate cook's distance for our reduced model

```
grades.cooks = cooks.distance(grades.reducedmlr)
sort(grades.cooks, decreasing = TRUE)[1:10]
```

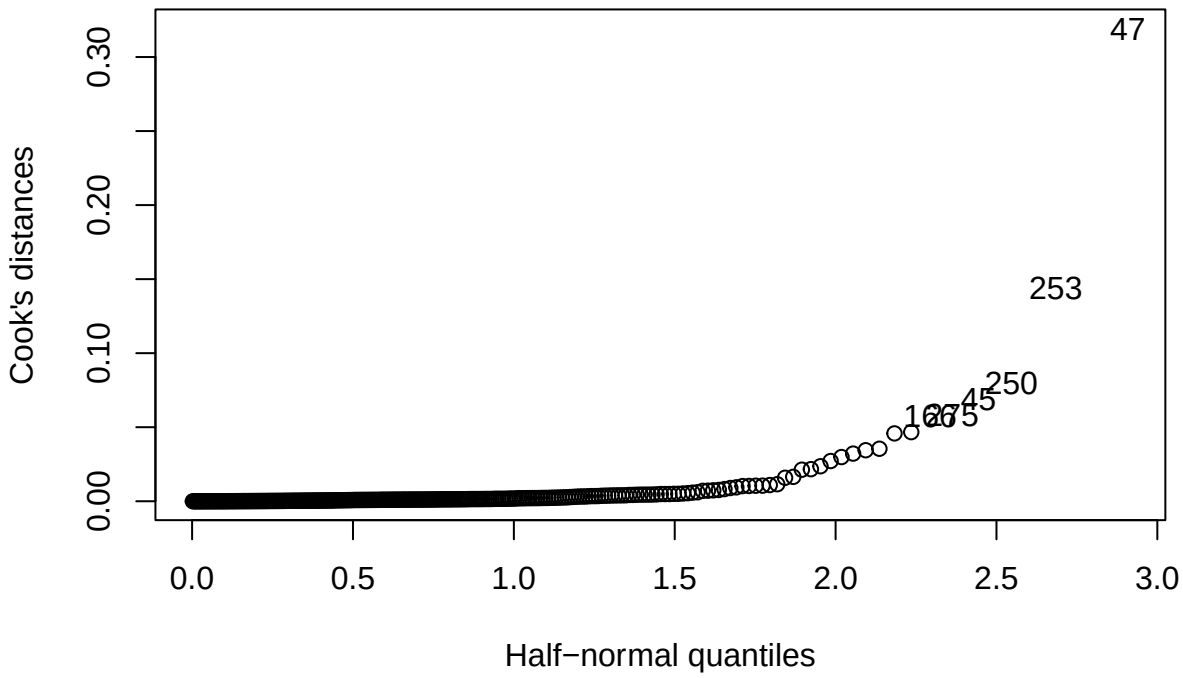
```
##         47        253        250         45        275        166         51
## 0.31936425 0.14401823 0.07998918 0.06876062 0.05806449 0.05763882 0.04667620
##        169        138        151
## 0.04581583 0.03546309 0.03443911
```

Here we plot the cook's distances

```
plot(grades.cooks)
```




```
halfnorm(grades.cooks, 6, labs=as.character(1:length(grades.cooks)), ylab="Cook's distances")
```

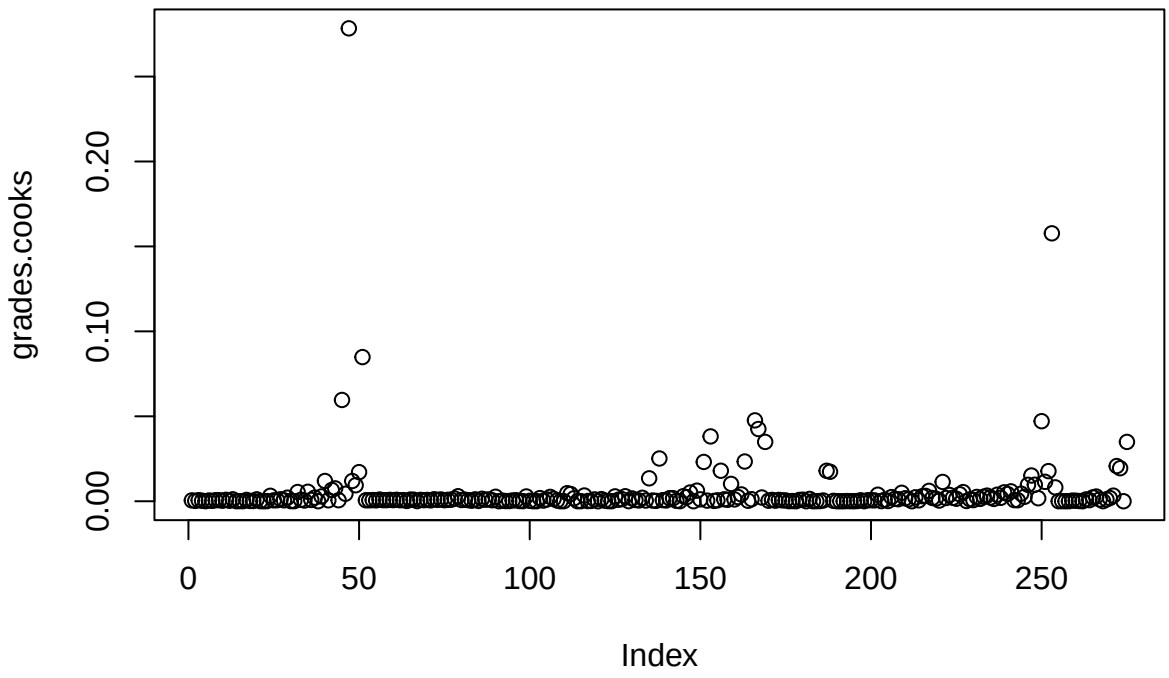


Here we calculate and plot cook's distances for the full model.

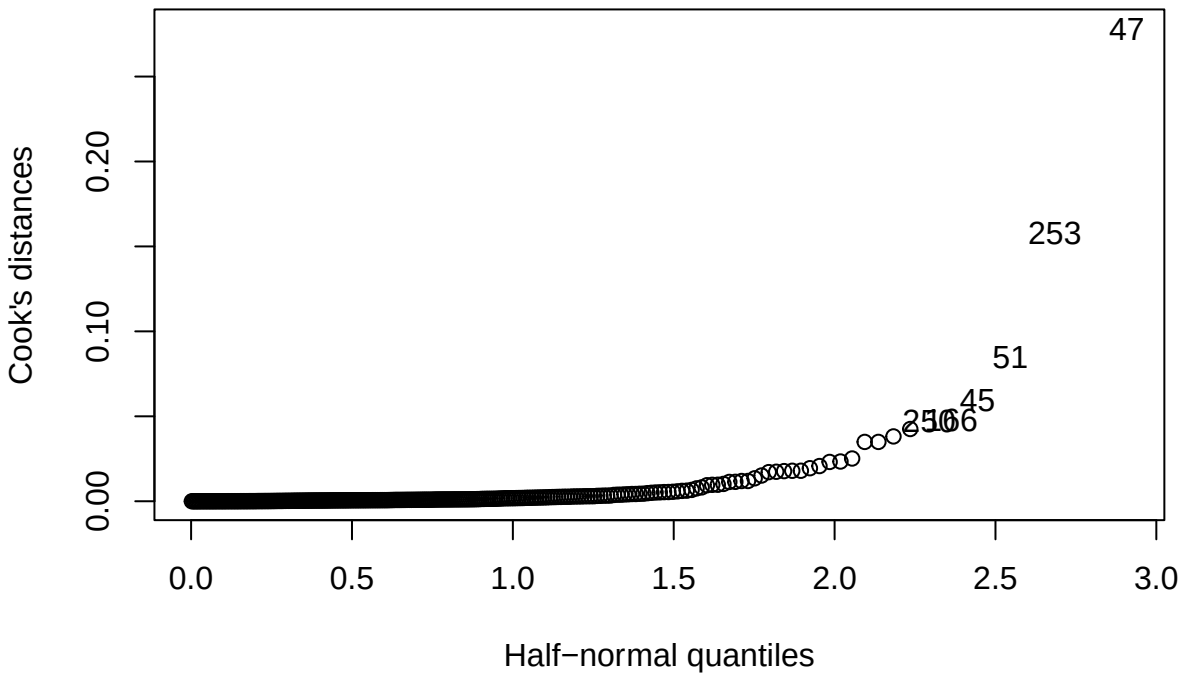
```
grades.cooks = cooks.distance(grades.mlr)
sort(grades.cooks, decreasing = TRUE)[1:10]
```

```
##         47        253         51         45        166        250        167
## 0.27837732 0.15771394 0.08485803 0.05961187 0.04761373 0.04711742 0.04252647
##        153        275        169
## 0.03816061 0.03492719 0.03492200
```

```
plot(grades.cooks)
```




```
halfnorm(grades.cooks, 6, labs=as.character(1:length(grades.cooks)), ylab="Cook's distances")
```



The number of influential points between the reduced and full models remained the same. Neither the full model nor the reduced model had influential points. To note, we can see that none of the cook's distances for the influential points are greater than 1, so we have no 'highly influential points'.