HW11

111078513

Help By 108078467

Question 1) Let's deal with nonlinearity first. Create a new dataset that log-transforms several variables from our original dataset (called cars in this case):

```
cars <- read.table("auto-data.txt", header=FALSE, na.strings = "?")
names(cars) <- c("mpg", "cylinders", "displacement", "horsepower", "weight",
                 "acceleration", "model_year", "origin", "car_name")
cars_log <- with(cars, data.frame(log(mpg), log(cylinders), log(displacement),
                                  log(horsepower),log(weight),log(acceleration),
                                  model_year, origin))
head(cars_log)
```

```
##   log.mpg. log.cylinders. log.displacement. log.horsepower. log.weight.
## 1 2.890372       2.079442          5.726848        4.867534    8.161660
## 2 2.708050       2.079442          5.857933        5.105945    8.214194
## 3 2.890372       2.079442          5.762051        5.010635    8.142063
## 4 2.772589       2.079442          5.717028        5.010635    8.141190
## 5 2.833213       2.079442          5.710427        4.941642    8.145840
## 6 2.708050       2.079442          6.061457        5.288267    8.375860
##   log.acceleration. model_year origin
## 1          2.484907         70      1
## 2          2.442347         70      1
## 3          2.397895         70      1
## 4          2.484907         70      1
## 5          2.351375         70      1
## 6          2.302585         70      1
```

a. Run a new regression on the cars_log dataset, with mpg.log. dependent on all other variables

(i) Which log-transformed factors have a significant effect on log.mpg. at 10% significance?

```
cars_log_lm <- lm(log.mpg.~ log.cylinders. + log.displacement.
                  +log.horsepower. + log.weight. + log.acceleration.
                  + model_year + origin ,data=cars_log)
cars_log_lm_logcoef = summary(cars_log_lm)$coefficients[2:6, ]
rownames(cars_log_lm_logcoef[cars_log_lm_logcoef[, 4] < 0.1, ])
```

```
## [1] "log.horsepower."   "log.weight."       "log.acceleration."
```

(ii) Do some new factors now have effects on mpg, and why might this be?

Horsepower, accleration now have effect on mpg, because taking the log of variables can give them more symmetric distributions.

(iii) Which factors still have insignificant or opposite (from correlation) effects on mpg? Why might this be?

Cylinders, displacement still have insignificant effects on mpg. It might means that those variables indeed don't have the effect on mpg, otherwise, it might means that there's the problem of multicollinearity on those variables.

b. Let's take a closer look at weight, because it seems to be a major explanation of mpg

(i) Create a regression (call it regr_wt) of mpg over weight from the original cars dataset

```
regr_wt <- lm(mpg ~ weight,cars)
summary(regr_wt)
```

```
##
## Call:
## lm(formula = mpg ~ weight, data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.012  -2.801  -0.351   2.114  16.480
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 46.3173644  0.7952452   58.24   <2e-16 ***
## weight      -0.0076766  0.0002575  -29.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.345 on 396 degrees of freedom
## Multiple R-squared:  0.6918, Adjusted R-squared:  0.691
## F-statistic: 888.9 on 1 and 396 DF,  p-value: < 2.2e-16
```

(ii) Create a regression (call it regr_wt_log) of log.mpg. on log.weight. from cars_log

```
regr_wt_log <- lm(log.mpg. ~ log.weight., cars_log)
summary(regr_wt_log)
```

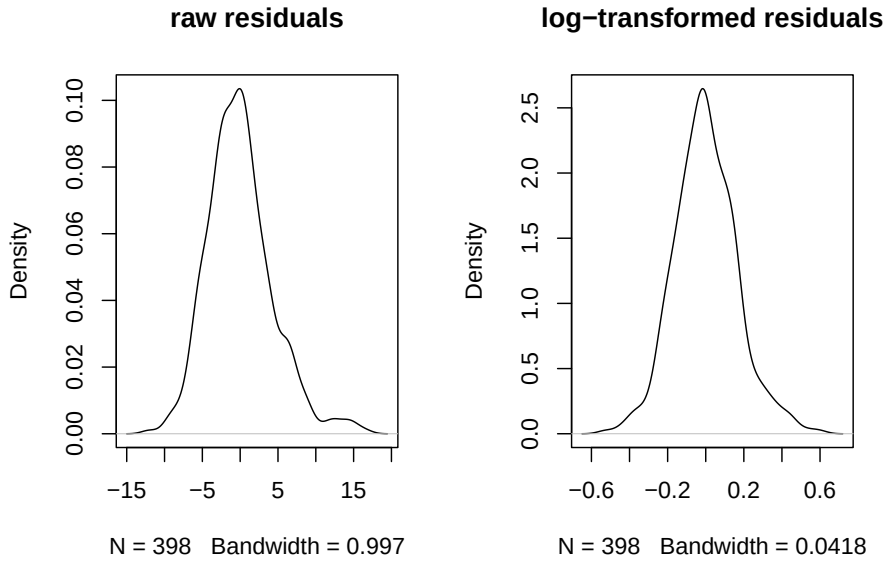
```
##
## Call:
## lm(formula = log.mpg. ~ log.weight., data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52408 -0.10441 -0.00805  0.10165  0.59384
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.5219     0.2349   49.06   <2e-16 ***
```

```
## log.weight.  -1.0583     0.0295  -35.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.165 on 396 degrees of freedom
## Multiple R-squared:  0.7647, Adjusted R-squared:  0.7641
## F-statistic:  1287 on 1 and 396 DF,  p-value: < 2.2e-16
```

(iii) Visualize the residuals of both regression models (raw and log-transformed):

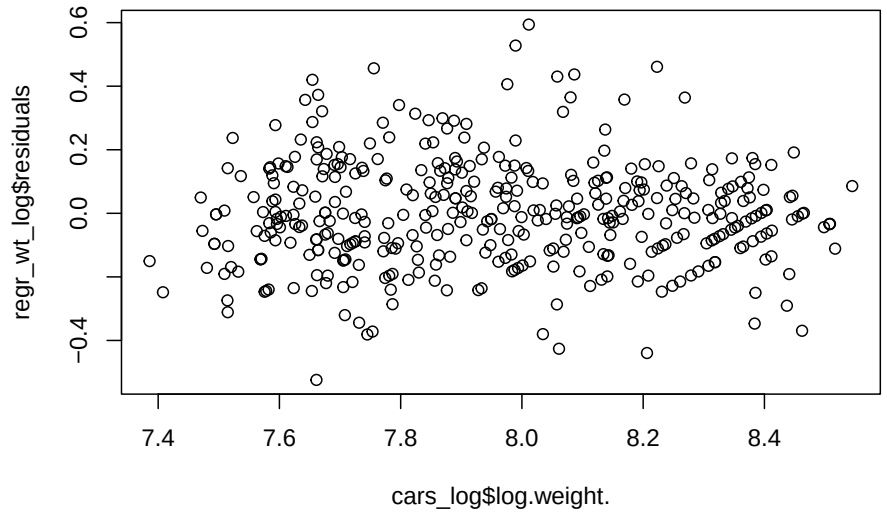
1. density plots of residuals

```
par(mfrow= c(1,2))
plot(density(regr_wt$residuals), main = "raw residuals")
plot(density(regr_wt_log$residuals), main = "log-transformed residuals")
```



2. scatterplot of log.weight. vs. residuals

```
plot(cars_log$log.weight., regr_wt_log$residuals)
```



(iv) Which regression produces better distributed residuals for the assumptions of regression?

log-transformed regression produces better distributed residuals for the assumptions of regression.

(v) How would you interpret the slope of log.weight. vs log.mpg. in simple words?

I would interpret it as the description that "1% change in weight leads to 1.06% decrease in mpg".

(vi) From its standard error, what is the 95% confidence interval of the slope of log.weight. vs log.mpg.?

```
confint(regr_wt_log, level = 0.95)
```

```
##                 2.5 %    97.5 %
## (Intercept) 11.060154 11.983659
## log.weight. -1.116264 -1.000272
```

Question 2) Let's tackle multicollinearity next. Consider the regression model:

```
regr_log <- lm(log.mpg. ~ log.cylinders. + log.displacement.
               + log.horsepower.+ log.weight. + log.acceleration.
               + model_year + factor(origin), data=cars_log)
```

a. Using regression and R2, compute the VIF of log.weight. using the approach shown in class

```
weight_regr <- lm(log.weight. ~ log.cylinders. + log.displacement.
                  + log.horsepower. + log.acceleration. + model_year +
                    factor(origin), data=cars_log)
r2_weight <- summary(weight_regr)$r.squared
vif_weight <- 1 / (1 - r2_weight)
sqrt(vif_weight)
```

```
## [1] 4.192269
```

b. Let's try a procedure called Stepwise VIF Selection to remove highly collinear predictors. Start by Installing the 'car' package in RStudio – it has a function called vif()

```
require(car)
```

```
## Loading required package: car
```

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

(i) Use vif(regr_log) to compute VIF of the all the independent variables

```
vif(regr_log)
```

```
##                        GVIF Df GVIF^(1/(2*Df))
## log.cylinders.    10.456738  1        3.233688
## log.displacement. 29.625732  1        5.442952
```

```
## log.horsepower.   12.132057  1        3.483110
## log.weight.       17.575117  1        4.192269
## log.acceleration.  3.570357  1        1.889539
## model_year         1.303738  1        1.141814
## factor(origin)     2.656795  2        1.276702
```

(ii) Eliminate from your model the single independent variable with the largest VIF score that is also greater than 5

```
regr_log_adj1 <- lm(log.mpg. ~ log.cylinders. + log.horsepower. +
                    log.weight.+ log.acceleration. + model_year +
                    factor(origin), data=cars_log)
vif(regr_log_adj1)
```

```
##                       GVIF Df GVIF^(1/(2*Df))
## log.cylinders.     5.433107  1        2.330903
## log.horsepower.   12.114475  1        3.480585
## log.weight.       11.239741  1        3.352572
## log.acceleration.  3.327967  1        1.824272
## model_year         1.291741  1        1.136548
## factor(origin)     1.897608  2        1.173685
```

(iii) Repeat steps (i) and (ii) until no more independent variables have VIF scores above 5

1.

```
regr_log_adj2 <- lm(log.mpg. ~ log.cylinders. +log.weight.+ log.acceleration.
                    + model_year + factor(origin), data=cars_log)
vif(regr_log_adj2)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## log.cylinders.    5.321090  1        2.306749
## log.weight.       4.788498  1        2.188264
## log.acceleration. 1.400111  1        1.183263
## model_year        1.201815  1        1.096273
## factor(origin)    1.792784  2        1.157130
```

2.

```
regr_log_adj3 <- lm(log.mpg. ~ log.weight.+ log.acceleration.
                    + model_year + factor(origin), data=cars_log)
vif(regr_log_adj3)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## log.weight.       1.926377  1        1.387940
## log.acceleration. 1.303005  1        1.141493
## model_year        1.167241  1        1.080389
## factor(origin)    1.692320  2        1.140567
```

There is no more independent variables have VIF scores above 5.

(iv) Report the final regression model and its summary statistics

```
summary(regr_log_adj3)
```

```
##
## Call:
## lm(formula = log.mpg. ~ log.weight. + log.acceleration. + model_year +
##     factor(origin), data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38275 -0.07032  0.00491  0.06470  0.39913
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        7.431155   0.312248  23.799  < 2e-16 ***
## log.weight.       -0.876608   0.028697 -30.547  < 2e-16 ***
## log.acceleration.  0.051508   0.036652   1.405  0.16072
## model_year         0.032734   0.001696  19.306  < 2e-16 ***
## factor(origin)2    0.057991   0.017885   3.242  0.00129 **
## factor(origin)3    0.032333   0.018279   1.769  0.07770 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1156 on 392 degrees of freedom
## Multiple R-squared:  0.8856, Adjusted R-squared:  0.8841
## F-statistic: 606.8 on 5 and 392 DF,  p-value: < 2.2e-16
```

c. Using stepwise VIF selection, have we lost any variables that were previously significant? If so, how much did we hurt our explanation by dropping those variables? (hint: look at model fit)

We have lost 'log.horsepower.' which was previously significant but not significant anymore after log-transformation. Furthermore, after dropping it, the R^2 decreased form 0.8912 to 0.8856. Clearly, such a numerical change has a small impact.

d. From only the formula for VIF, try deducing/deriving the following:

1. If an independent variable has no correlation with other independent variables, what would its VIF score be?

If an independent variable has no correlation with other independent variables, its VIF score would be 1, indicating no multicollinearity issue.

2. Given a regression with only two independent variables (X1 and X2), how correlated would X1 and X2 have to be, to get VIF scores of 5 or higher? To get VIF scores of 10 or higher?

If we want VIF(X1) and VIF(X2) to be at least 5, we can set these equations equal to 5 and solve for r: $5 = 1/(1 - r^2)$

```
R_squ_5 <- 1-(1/5)
sqrt(R_squ_5)
```

```
## [1] 0.8944272
```

Therefore, X1 and X2 would need to be highly correlated, with a correlation coefficient of at least 0.894, to get VIF scores of 5 or higher.

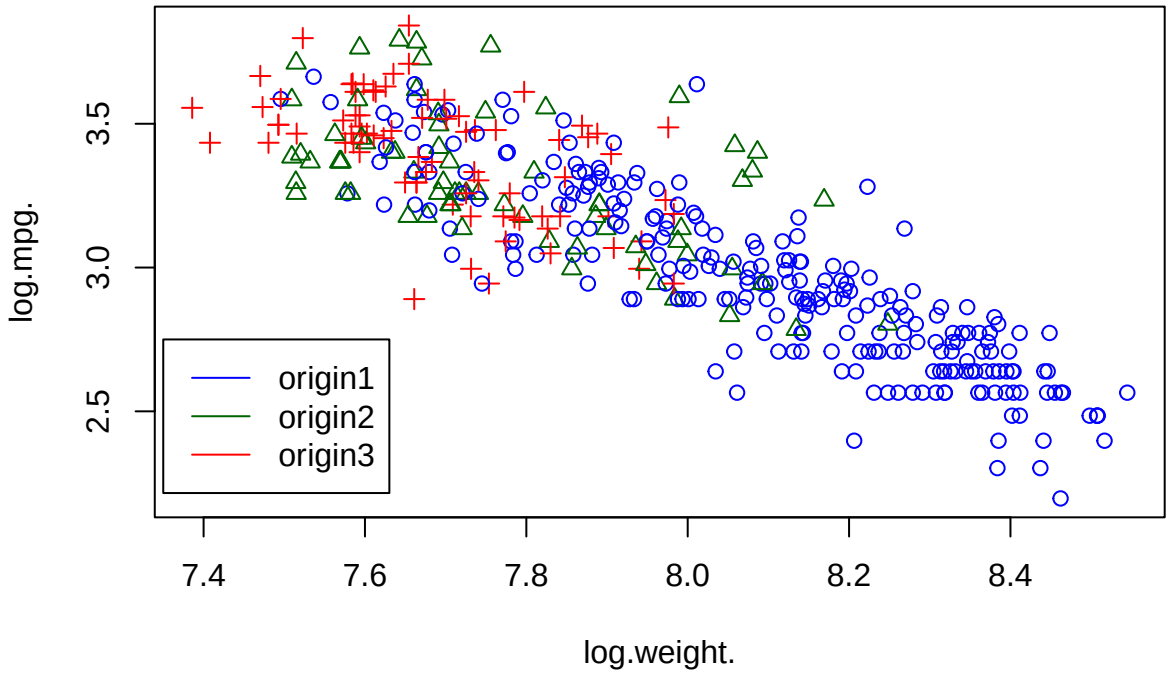
```
R_squ_10 <- 1-(1/10)
sqrt(R_squ_10)
```

```
## [1] 0.9486833
```

According the computation above, X1 and X2 would need to be very highly correlated, with a correlation coefficient of at least 0.953, to get VIF scores of 10 or higher.

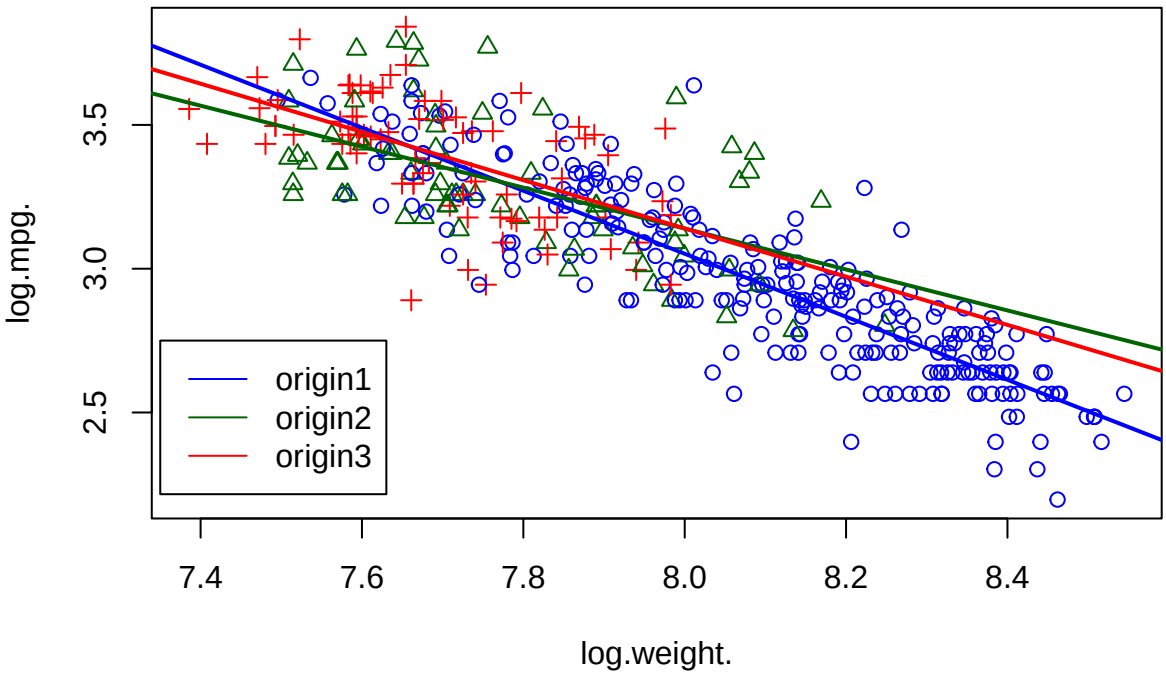
Question 3) Might the relationship of weight on mpg be different for cars from different origins?

```
origin_colors = c("blue", "darkgreen", "red")
with(cars_log, plot(log.weight., log.mpg., pch=origin, col=origin_colors[origin]))
legend(7.35,2.75, lty=1, c("origin1", "origin2", "origin3"), col=origin_colors)
```



Let's add three separate regression lines on the scatterplot, one for each of the origins.

```
with(cars_log, plot(log.weight., log.mpg., pch=origin, col=origin_colors[origin]))
legend(7.35,2.75, lty=1, c("origin1", "origin2", "origin3"), col=origin_colors)
#origin1
cars_us <- subset(cars_log, origin==1)
wt_regr_us <- lm(log.mpg. ~ log.weight., data=cars_us)
abline(wt_regr_us, col=origin_colors[1], lwd=2)
#origin2
cars_us <- subset(cars_log, origin==2)
wt_regr_us <- lm(log.mpg. ~ log.weight., data=cars_us)
abline(wt_regr_us, col=origin_colors[2], lwd=2)
#origin3
cars_us <- subset(cars_log, origin==3)
wt_regr_us <- lm(log.mpg. ~ log.weight., data=cars_us)
abline(wt_regr_us, col=origin_colors[3], lwd=2)
```


log.mpg.
2.5
3.0
3.5
7.4
7.6
7.8
8.0
8.2
8.4
log.weight.
origin1
origin2
origin3