BACS - HW11

109006241

Let's go back and take another look at our analysis of the cars dataset. Recall our variables:

1. mpg: miles-per-gallon (dependent variable)
2. cylinders: cylinders in engine
3. displacement: size of engine
4. horsepower: power of engine
5. weight: weight of car
6. acceleration: acceleration ability of car (seconds to achieve 0-60mph)
7. model_year: year model was released
8. origin: place car was designed (1: USA, 2: Europe, 3: Japan)

Did you notice the following from doing a full regression model of mpg on all independent variables?

- Only weight, year, and origin had significant effects
- Non-significant factors cylinders, displacement & horsepower were highly correlated with weight
- Displacement has the opposite effect in the regression from its visualized effect!
- Several factors, like horsepower, seem to have a nonlinear (exponential) relationship with mpg

---

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", 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, 5)
```

```
##   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
##   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
```

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

```
summary(lm(log.mpg. ~ . + factor(origin) - origin, cars_log))
```

```
##
## Call:
## lm(formula = log.mpg. ~ . + factor(origin) - origin, data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39727 -0.06880  0.00450  0.06356  0.38542
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        7.301938   0.361777  20.184  < 2e-16 ***
## log.cylinders.    -0.081915   0.061116  -1.340  0.18094
## log.displacement.  0.020387   0.058369   0.349  0.72707
## log.horsepower.   -0.284751   0.057945  -4.914 1.32e-06 ***
## log.weight.       -0.592955   0.085165  -6.962 1.46e-11 ***
## log.acceleration. -0.169673   0.059649  -2.845  0.00469 **
## model_year         0.030239   0.001771  17.078  < 2e-16 ***
## factor(origin)2    0.050717   0.020920   2.424  0.01580 *
## factor(origin)3    0.047215   0.020622   2.290  0.02259 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.113 on 383 degrees of freedom
##   (6 observations deleted due to missingness)
## Multiple R-squared:  0.8919, Adjusted R-squared:  0.8897
## F-statistic:   395 on 8 and 383 DF,  p-value: < 2.2e-16
```

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

The horsepower, weight, acceleration, model_year, origin=2, and origin=3 variables.

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

Yes, this is because we have applied log transformation into our dataset, transforming it into a more normal distribution, so that the relationship would be more linear.

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

The cylinders and displacement variables. This might be because the log transformation is not suitable for these variables / columns, so they still have an insignificant effect on mpg.

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)
```

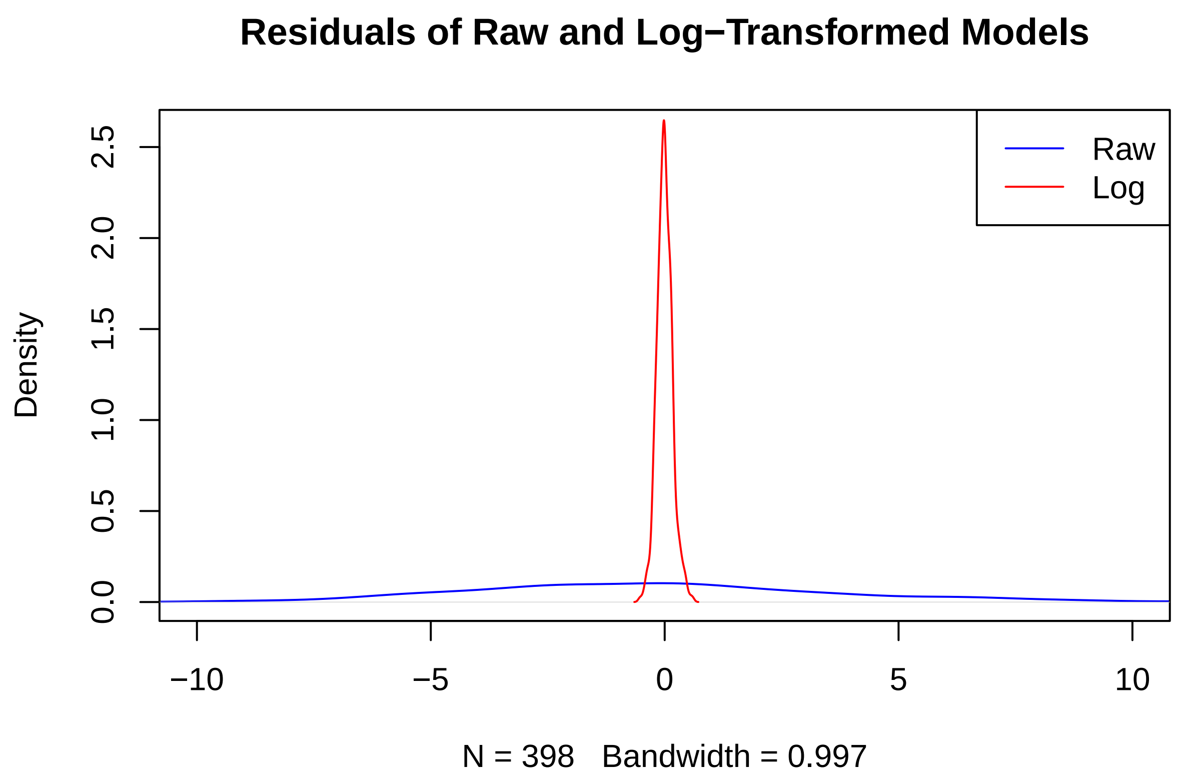
ii. Create a regression (call it regr_wt_log) of log.mpg. on log.weight. from cars_log

```
regr_wt_log = lm(log.mpg. ~ log.weight., cars_log)
```

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

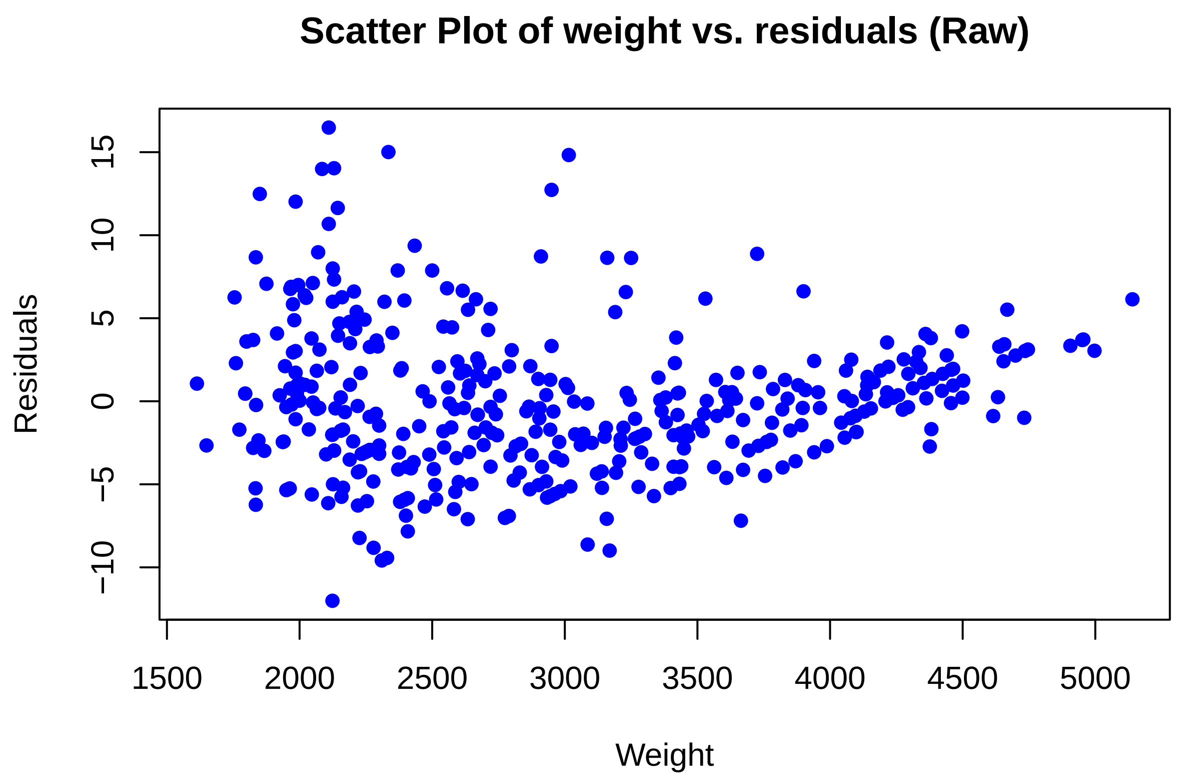
1. density plots of residuals

```
plot(density(regr_wt$residuals), main="Residuals of Raw and Log-Transformed Models", col="blue", xlim=c
lines(density(regr_wt_log$residuals), col="red")
legend("topright", legend=c("Raw", "Log"), col=c("blue", "red"), lty=c(1, 1))
```

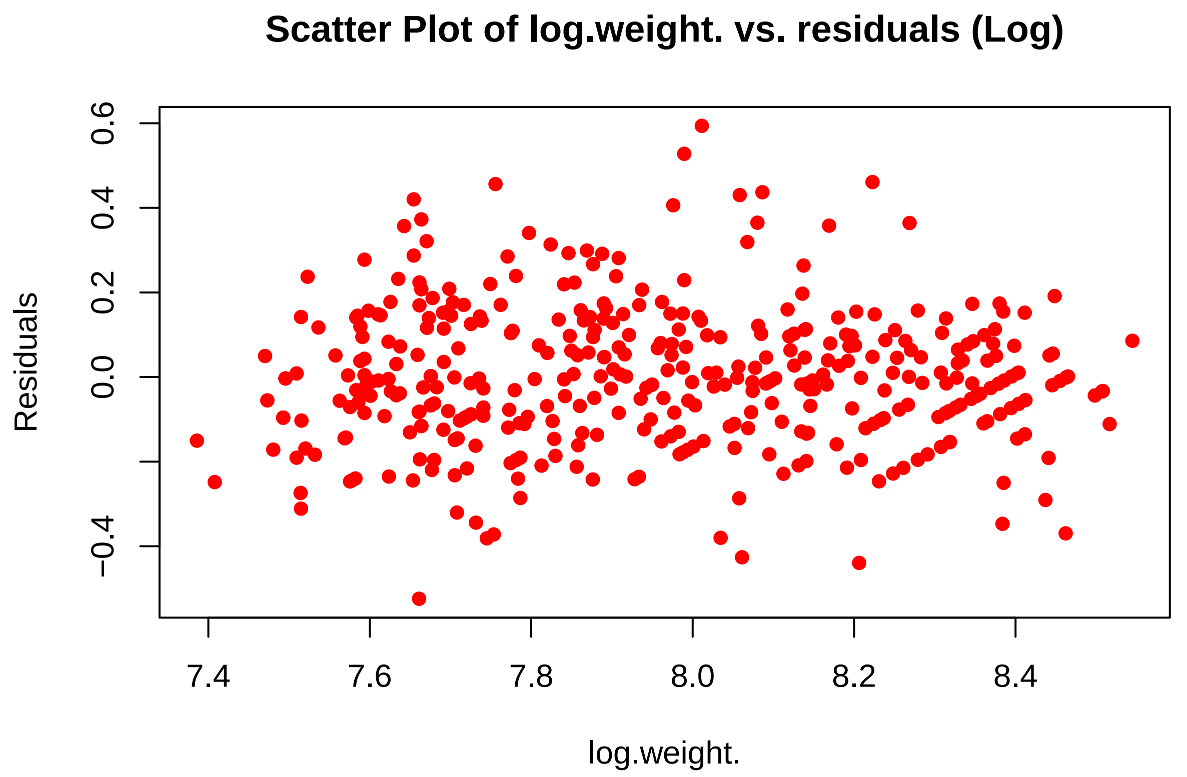


2. scatterplot of log.weight. vs. residuals

```r
plot(cars$weight, regr_wt$residuals, pch=16, col="blue",
     main="Scatter Plot of weight vs. residuals (Raw)", xlab="Weight", ylab="Residuals")
```



```r
plot(cars_log$log.weight., regr_wt_log$residuals, pch=16, col="red",
     main="Scatter Plot of log.weight. vs. residuals (Log)", xlab="log.weight.", ylab="Residuals")
```

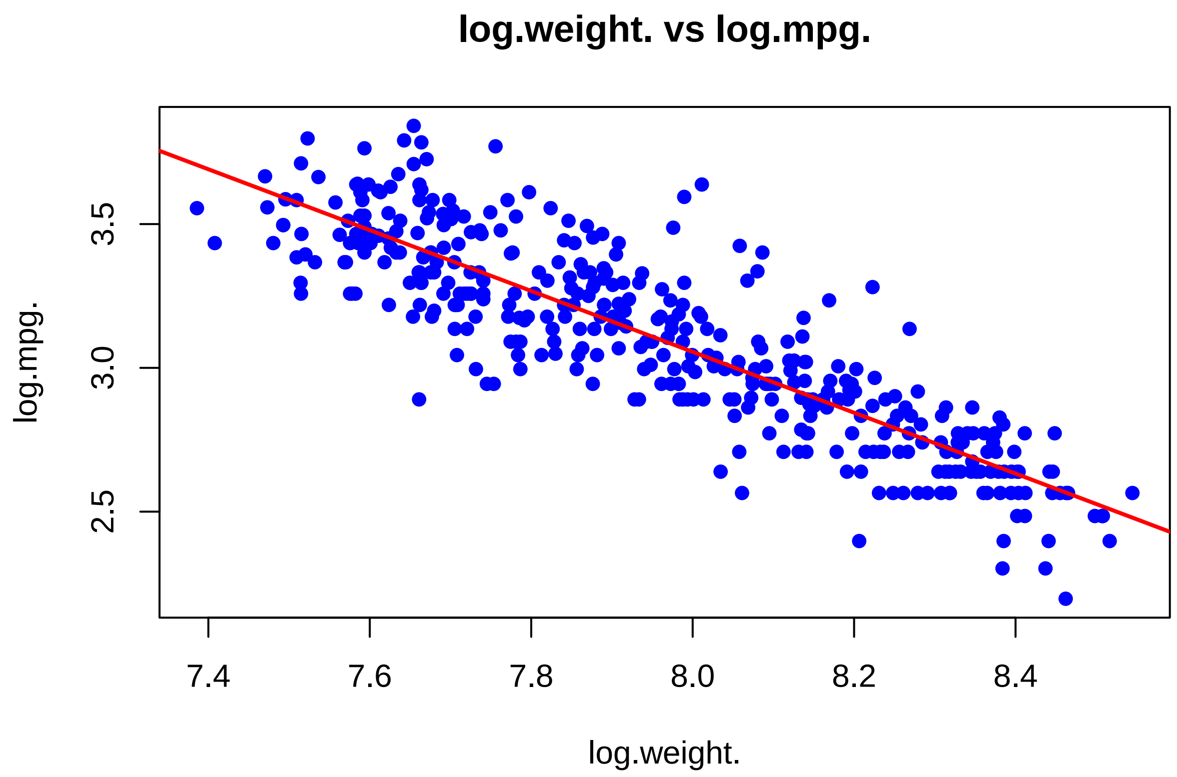


iv. Which regression produces better distributed residuals for the assumptions of regression?

The log-transformed model produces a better distributed residual, as we can see from the plots above, where the residuals of the log-transformed model is more centered towards the middle.

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

```
plot(cars_log$log.weight., cars_log$log.mpg.,  pch=16, col="blue",
     main="log.weight. vs log.mpg.", xlab="log.weight.", ylab="log.mpg.")
abline(lm(log.mpg. ~ log.weight., cars_log), col="red", lwd=2)
```



```
slope = regr_wt_log$coefficients[2]
cat("Slope of log.weight. vs log.mpg. =", slope)
```

```
## Slope of log.weight. vs log.mpg. = -1.058268
```

A 1% increase in log.weight. will result in a -1.058268% increase in log.mpg.

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

```
slope_se = summary(regr_wt_log)$coefficients[2, "Std. Error"]
df = nrow(cars_log) - 1
alpha = 0.05
t_val = qt(1 - alpha/2, df)

ci95_lower = slope - t_val * slope_se
ci95_upper = slope + t_val * slope_se

cat("95% CI of the slope of log.weight. vs log.mpg. =", ci95_lower, "to", ci95_upper)
```

```
## 95% CI of the slope of log.weight. vs log.mpg. = -1.116263 to -1.000273
```

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)
summary(regr_log)
```

```
##
## Call:
## lm(formula = log.mpg. ~ log.cylinders. + log.displacement. +
##     log.horsepower. + log.weight. + log.acceleration. + model_year +
##     factor(origin), data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39727 -0.06880  0.00450  0.06356  0.38542
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        7.301938   0.361777  20.184  < 2e-16 ***
## log.cylinders.    -0.081915   0.061116  -1.340  0.18094
## log.displacement.  0.020387   0.058369   0.349  0.72707
## log.horsepower.   -0.284751   0.057945  -4.914 1.32e-06 ***
## log.weight.       -0.592955   0.085165  -6.962 1.46e-11 ***
## log.acceleration. -0.169673   0.059649  -2.845  0.00469 **
## model_year         0.030239   0.001771  17.078  < 2e-16 ***
## factor(origin)2    0.050717   0.020920   2.424  0.01580 *
## factor(origin)3    0.047215   0.020622   2.290  0.02259 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.113 on 383 degrees of freedom
##   (6 observations deleted due to missingness)
## Multiple R-squared:  0.8919, Adjusted R-squared:  0.8897
## F-statistic:   395 on 8 and 383 DF,  p-value: < 2.2e-16
```

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)
cat("VIF of log.weight. =", vif_weight)
```

```
## VIF of log.weight. = 17.57512
```

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(). (note: CAR package stands for Companion to Applied Regression – it isn't about cars!)

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

```r
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

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

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

```r
vif(regr_log)
```

```
##                         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
```

```r
# Eliminate log.horsepower.
regr_log = lm(log.mpg. ~ log.cylinders. + log.weight. +
                         log.acceleration. + model_year +
                         factor(origin), data=cars_log)
```

```r
vif(regr_log)
```

```
##                        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
```

```r
# Eliminate log.cylinders.
regr_log = lm(log.mpg. ~ log.weight. + log.acceleration. +
                         model_year + factor(origin),
                         data=cars_log)
```

```
vif(regr_log)
```

```
##                       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
```

iv. Report the final regression model and its summary statistics

```
summary(regr_log)
```

```
##
## 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)

   Yes. The R-squared value only decreases a little bit, so by dropping those variables, it doesn't hurt our explanation that much.

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

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

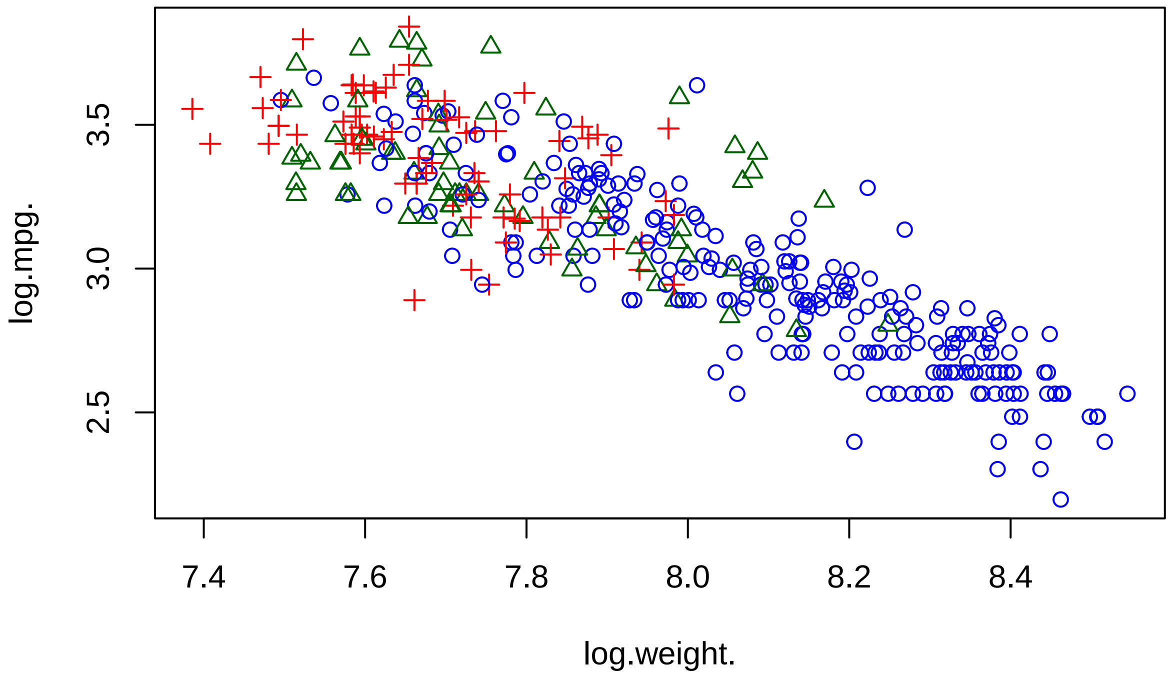
   If an independent variable has no correlation with other independent variables, then the R-squared, will be zero. According to the VIF formula, which is 1 / (1 - R-squared), the VIF score would be 1.

ii. 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?

> According to the VIF formula, X1 and X2 would need to be highly correlated with an R-squared of 0.8 or higher, to get an VIF score of 5 or higher.
>
> According to the VIF formula, X1 and X2 would need to be highly correlated with an R-squared of 0.9 or higher, to get an VIF score of 10 or higher.

Question 3) Might the relationship of weight on mpg be different for cars from different origins? Let's try visualizing this. First, plot all the weights, using different colors and symbols for the three origins: (you may choose any three colors you wish or plot this using ggplot etc. – the code below is for reference)

```
origin_colors = c("blue", "darkgreen", "red")
with(cars_log, plot(log.weight., log.mpg., pch=origin, col=origin_colors[origin]))
```

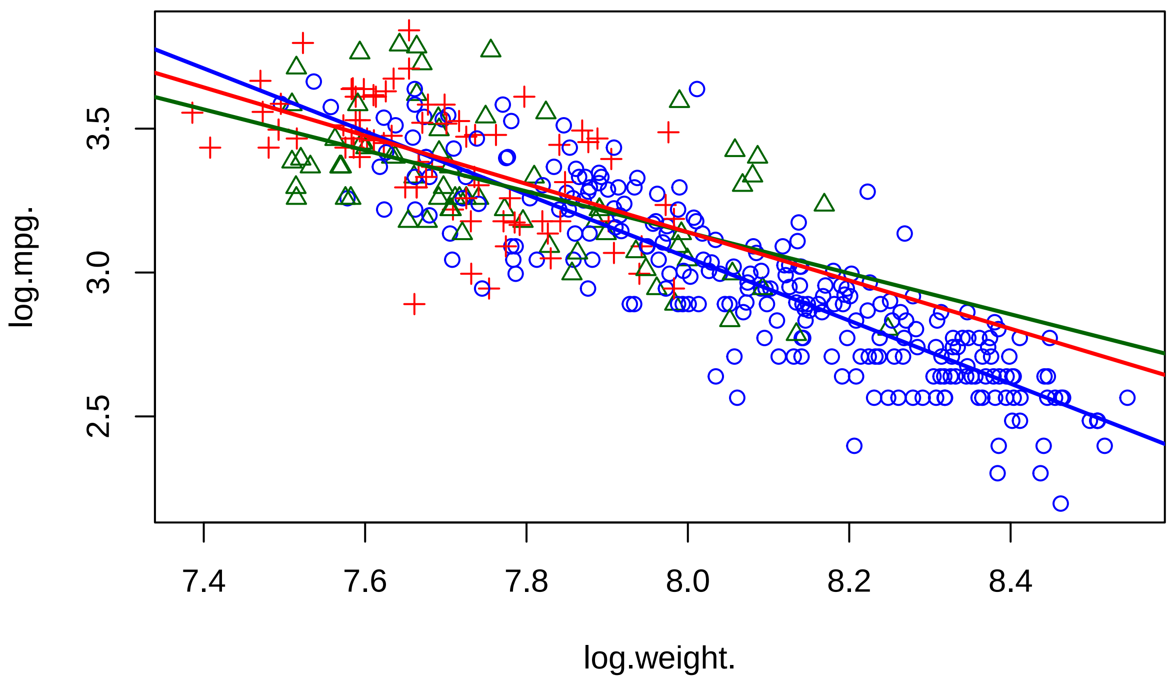


a. Let's add three separate regression lines on the scatterplot, one for each of the origins. Here's one for the US to get you started:

```r
cars_us <- subset(cars_log, origin==1)
wt_regr_us <- lm(log.mpg. ~ log.weight., data=cars_us)
cars_eu <- subset(cars_log, origin==2)
wt_regr_eu <- lm(log.mpg. ~ log.weight., data=cars_eu)
cars_jp <- subset(cars_log, origin==3)
wt_regr_jp <- lm(log.mpg. ~ log.weight., data=cars_jp)

with(cars_log, plot(log.weight., log.mpg., pch=origin, col=origin_colors[origin]))

abline(wt_regr_us, col=origin_colors[1], lwd=2)
abline(wt_regr_eu, col=origin_colors[2], lwd=2)
abline(wt_regr_jp, col=origin_colors[3], lwd=2)
```



b. [not graded] Do cars from different origins appear to have different weight vs. mpg relationships?

From the plot above, yes they seem to have different weight vs mpg relationships. But to be more sure, we would need to perform further tests to check this statement.