BACS HW11

109090046 assisted by 109090035

2023-04-25

```
auto <- read.table("D:/下載/auto-data.txt", header=FALSE, na.strings = "?")
names(auto) <- c("mpg", "cylinders", "displacement", "horsepower", "weight", "acceleratio
n", "model_year", "origin", "car_name")
```

Question 1)

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

```
cars_log <- with(auto, data.frame(log(mpg), log(cylinders), log(displacement),
                                  log(horsepower), log(weight), log(acceleration),
                                  model_year, origin))
```

a.

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

```
regr_log <- lm(log.mpg. ~ log.cylinders. + log.displacement. + log.horsepower. + log.weig
ht. + 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
##   (因為不存在，6 個觀察量被刪除了)
## 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?

ANS: Variables with p-values less than 0.1 are significant at the 10% level. So `log.horsepower.`, `log.weight.`, `log.acceleration.` and `model_year` are significant at the 10% level.

ii.

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

ANS: Yes, `acceleration.` and `horsepower.` are new factors, and because the log transformation may have linearized the relationships between mpg and those factors. This makes it easier for the linear regression model to capture the relationships.

iii.

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

ANS: `cylinder` still has insignificant effects on mpg. Because insignificant factors might not have a strong relationship with mpg or their relationship may still be nonlinear even after the log transformation. Also, the presence of multi-collinearity between independent variables can lead to unstable estimates of the coefficients, making it difficult to interpret the results. Factors with opposite signs compared to the correlation might be due to suppression effects or interactions between variables that were not accounted for in the model.

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, data = auto)
```

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., data = cars_log)
```

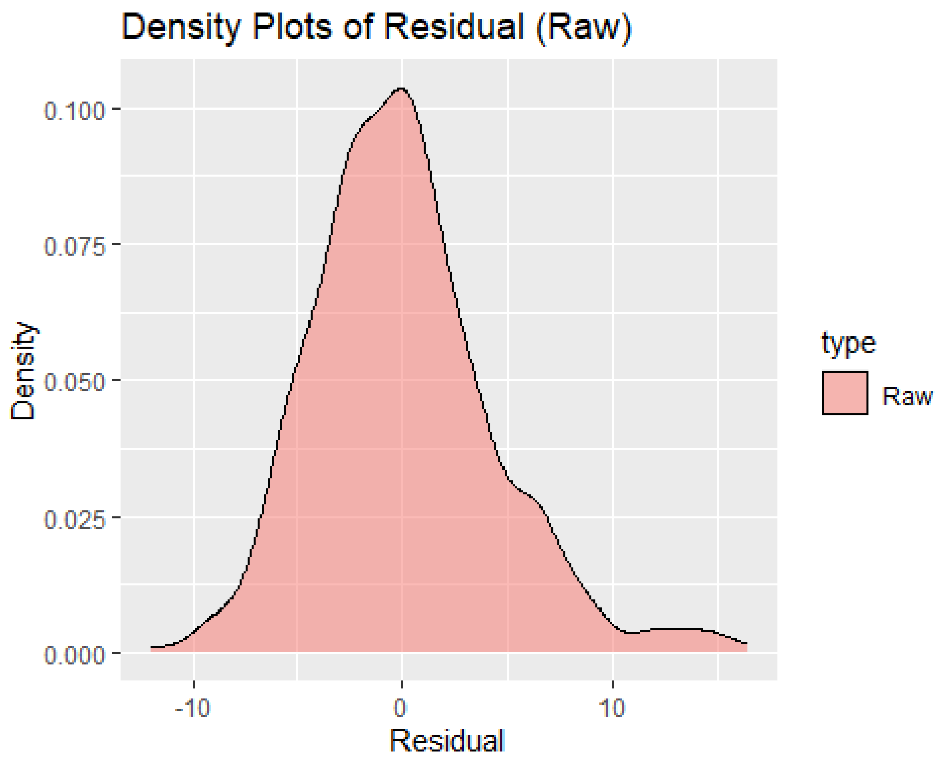
iii.

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

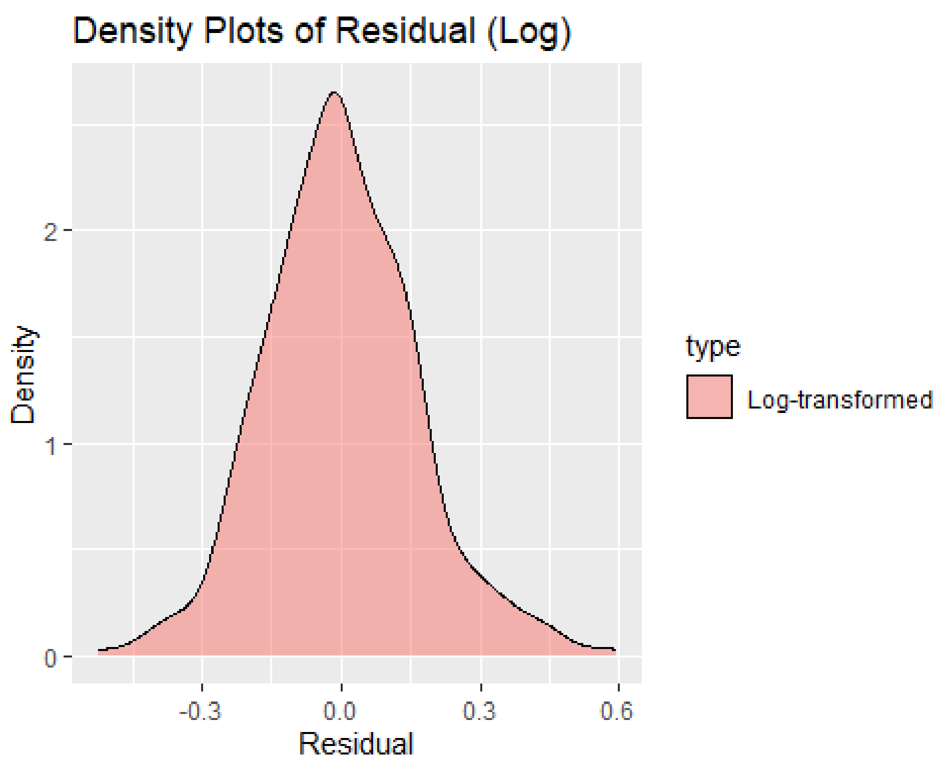
1.

density plots of residuals

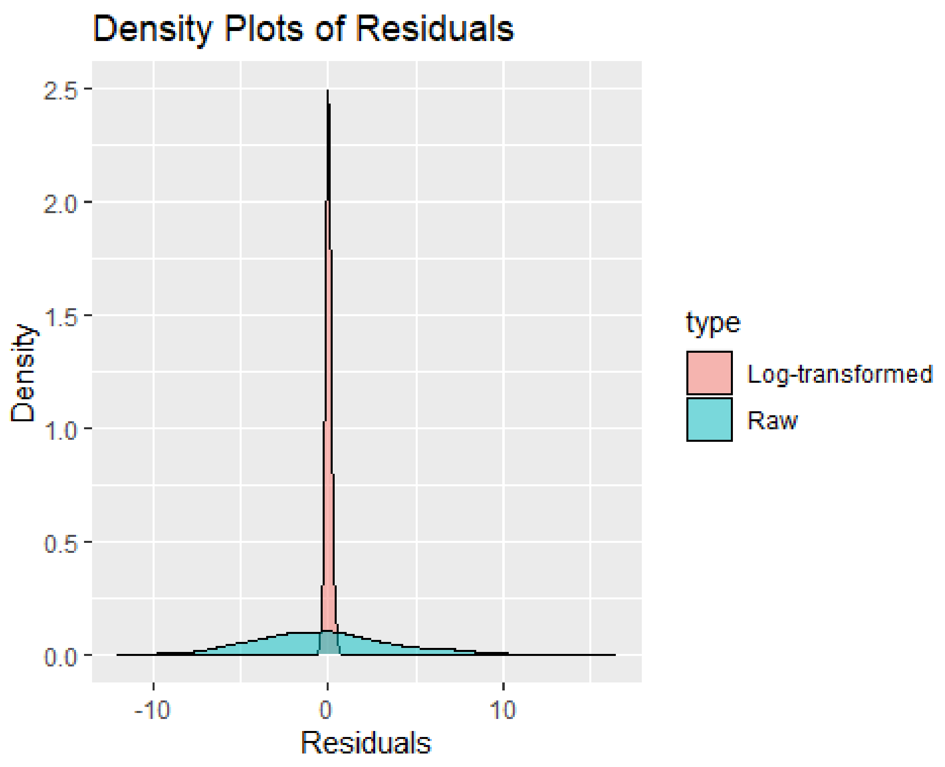
```
residuals_raw <- data.frame(residuals = regr_wt$residuals, type = "Raw")
residuals_log <- data.frame(residuals = regr_wt_log$residuals, type = "Log-transformed")
residuals_combined <- rbind(residuals_raw, residuals_log)

density_residual_raw <- ggplot(residuals_raw, aes(x = residuals, fill = type)) +
  geom_density(alpha = 0.5) +
  theme_grey() +
  labs(title = "Density Plots of Residual (Raw)", x = "Residual", y = "Density")

density_residual_raw
```



```
density_residual_log <- ggplot(residuals_log, aes(x = residuals, fill = type)) +
  geom_density(alpha = 0.5) +
  theme_grey() +
  labs(title = "Density Plots of Residual (Log)", x = "Residual", y = "Density")

density_residual_log
```



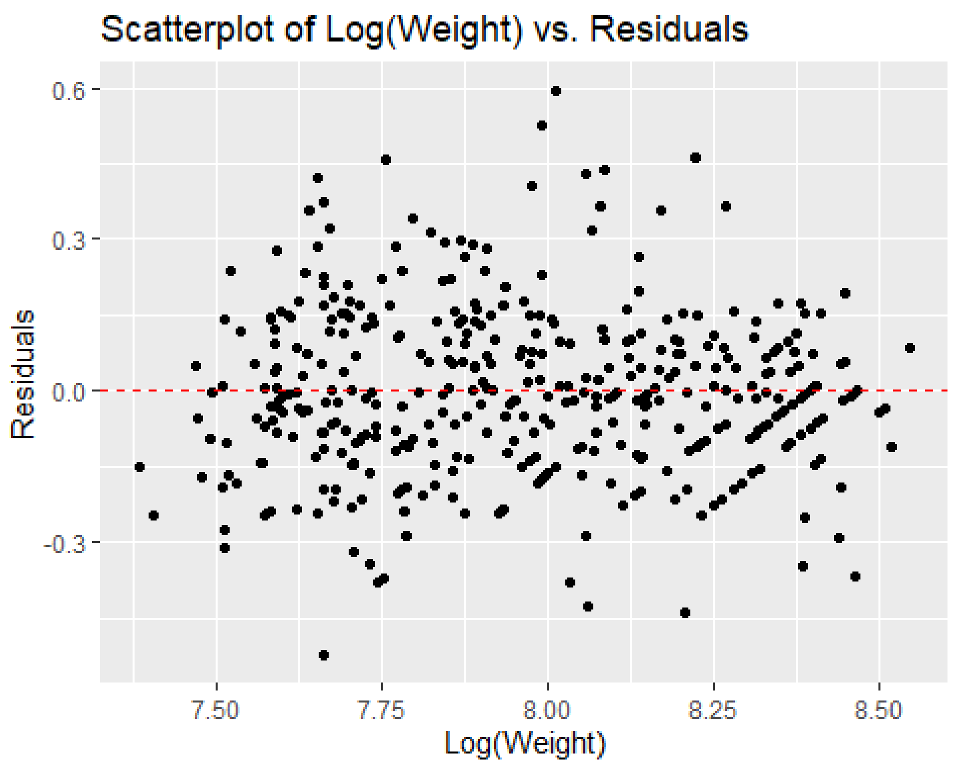
```r
density_residuals <- ggplot(residuals_combined, aes(x = residuals, fill = type)) +
  geom_density(alpha = 0.5) +
  theme_grey() +
  labs(title = "Density Plots of Residuals", x = "Residuals", y = "Density")

density_residuals
```



2.

scatterplot of log.weight. vs. residuals

```
scatter_residuals <- ggplot(cars_log, aes(x = log.weight., y = regr_wt_log$residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  theme_grey() +
  labs(title = "Scatterplot of Log(Weight) vs. Residuals", x = "Log(Weight)", y = "Residu
als")

scatter_residuals
```



iv.

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

ANS: Compare the density plots of residuals for both regression models. Both raw and log-transformed one are normally distributed and centered to zero, so I think it's hard to tell. But by the comparing graphic, we can see that log-transformed is more centralized in the center.

v.

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

ANS: The slope of `log.weight.` vs `log.mpg.` represents the percentage change in mpg for a 1% increase in weight. In other words, if the weight of a car increases by 1%, the miles per gallon will change by the percentage indicated by the slope.

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)["log.weight.",]
```

```
##      2.5 %    97.5 %
## -1.116264 -1.000272
```

Question 2)

Let's tackle multi-collinearity 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

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

r2_log_wei <- summary(log_wei_regr)$r.squared
vif_log_wei <- 1 / (1 - r2_log_wei)

vif_log_wei

## [1] 19.79957
```

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!)

```
library(car)

## Warning: 套件 'car' 是用 R 版本 4.2.3 來建造的

## 載入需要的套件：carData

## Warning: 套件 'carData' 是用 R 版本 4.2.3 來建造的

##
## 載入套件：'car'

## 下列物件被遮斷自 'package:dplyr':
##
##     recode

## 下列物件被遮斷自 'package:purrr':
##
##     some
```

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_vif <- regr_log
vif_scores <- vif(regr_vif)
worst_var <- row.names(vif_scores)[which.max(vif_scores)]

if (max(vif_scores) > 5) {
  regr_log_elim <- update(regr_log, as.formula(paste("~ . -", worst_var)))
}

summary(regr_log_elim)

## 
## Call:
## lm(formula = log.mpg. ~ log.cylinders. + log.horsepower. + log.weight. + 
##     log.acceleration. + model_year + factor(origin), data = cars_log)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.40059 -0.06820  0.00484  0.06208  0.39096 
## 
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)    
## (Intercept)        7.26400    0.34469  21.074  < 2e-16 ***
## log.cylinders.    -0.06712    0.04400  -1.525   0.1280    
## log.horsepower.   -0.28552    0.05784  -4.937 1.19e-06 ***
## log.weight.       -0.57510    0.06803  -8.454 5.94e-16 ***
## log.acceleration. -0.17510    0.05752  -3.044   0.0025 ** 
## model_year         0.03018    0.00176  17.143  < 2e-16 ***
## factor(origin)2    0.04717    0.01826   2.582   0.0102 *  
## factor(origin)3    0.04394    0.01834   2.396   0.0171 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.1128 on 384 degrees of freedom
##   (因為不存在，6 個觀察量被刪除了)
## Multiple R-squared:  0.8919, Adjusted R-squared:  0.8899 
## F-statistic: 452.5 on 7 and 384 DF,  p-value: < 2.2e-16
```

iii.

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

```
vif_scores <- vif(regr_log_elim)

while (max(vif_scores) > 5) {
  worst_var <- row.names(vif_scores)[which.max(vif_scores)]
  regr_log_elim <- update(regr_log_elim, as.formula(paste("~ . -", worst_var)))
  vif_scores <- vif(regr_log_elim)
}
```

iv.

Report the final regression model and its summary statistics

```
summary(regr_log_elim)

## 
## 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)

```
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
##   (因為不存在，6 個觀察量被刪除了)
## Multiple R-squared:  0.8919, Adjusted R-squared:  0.8897
## F-statistic:   395 on 8 and 383 DF,  p-value: < 2.2e-16

summary(regr_log_elim)

##
## 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
```

ANS: Comparing the result of two summary. Yes, we have lost `log.horsepower.` that was previously significant. To assess the impact of dropping those variables on the overall model fit, compare the R-squared or adjusted R-squared values. The Adjusted R-squared value drops from 0.8897 to 0.8841, which indicate that dropping these variables did not hurt the model's explanatory power 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?

ANS:

If an independent variable has no correlation with other independent variables, it means that it cannot be explained by the other variables, which results in an R^2 value of 0. Using the VIF formula, we get: VIF = 1 / (1 - 0) = 1

So, 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?

ANS:

Given a regression with only two independent variables (X1 and X2), we can use the VIF formula to find the correlation required to get VIF scores of 5 or higher and 10 or higher.

For VIF = 5:

$5 = 1 / (1 - R^2)$

Solve for R^2:

$R^2 = 1 - (1 / 5) = 0.8$

For VIF = 10:

$10 = 1 / (1 - R^2)$

Solve for R^2:

$R^2 = 1 - (1 / 10) = 0.9$

Now, since there are only two independent variables (X1 and X2), their R^2 is equivalent to the square of their correlation coefficient (r^2). So, we need to find r for both cases:

For $R^2 = 0.8$:

$r = \text{sqrt}(0.8) \approx 0.894$

For $R^2 = 0.9$:

$r = \text{sqrt}(0.9) \approx 0.949$

So, X1 and X2 need to be correlated with approximately 0.894 or higher to get VIF scores of 5 or higher, and approximately 0.949 or higher to get VIF scores of 10 or higher.

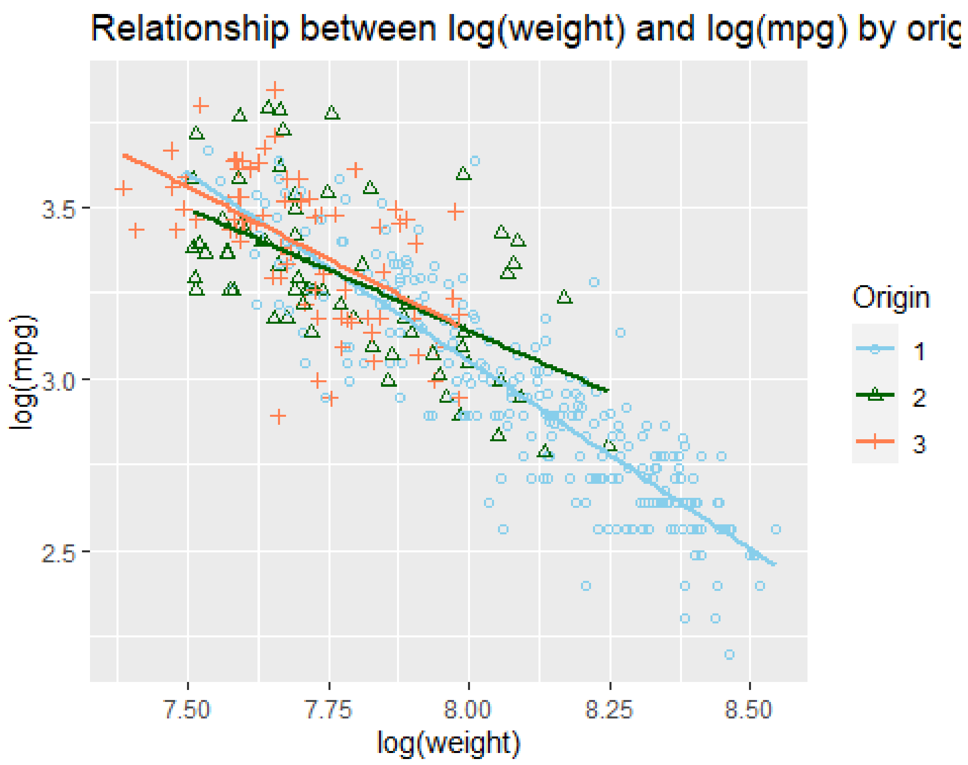
Question 3)

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:

a.

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

```
origin_colors = c("skyblue", "darkgreen", "coral")
ggplot(cars_log, aes(x=log.weight., y=log.mpg., color=factor(origin), shape=factor(origi
n))) +
  geom_point() +
  theme_grey() +
  geom_smooth(method="lm", se=FALSE, linetype="solid") +
  scale_color_manual(values=origin_colors) +
  scale_shape_manual(values=c(1, 2, 3)) +
  labs(title="Relationship between log(weight) and log(mpg) by origin",
       x="log(weight)",
       y="log(mpg)",
       color="Origin",
       shape="Origin")

## `geom_smooth()` using formula = 'y ~ x'
```



b.

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

ANS: The slope of the regression line are different, but they are close. So, we can conclude that they have same relationship that as mpg increase, weight decrease, vic versa.