HW11

110077443

4/29/2022

Please note that all code in this document is presented in a grey box and the output reflected below each box

- The below code allows lengthy lines of comments to display neatly within the grey box (wrapping it)

```
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 60), tidy = TRUE)
```

1) Dataset that log-transforms several variables from "cars" dataset

```
# Importing data
auto <- read.table("auto-data.txt", header = FALSE, na.strings = "?")

# Renaming variables
names(auto) <- c("mpg", "cylinders", "displacement", "horsepower",
    "weight", "acceleration", "model_year", "origin", "car_name")

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

a) Running regression on cars_log dataset, with mpg.log. dependent on all other variables

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

```
##
## 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
```

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

ANSWER ##

- **"log.horsepower.", "log.weight.", "log.acceleration."**

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

ANSWER ##

- **"horsepower" and "acceleration"** have effects on mpg now
- Using the logarithm of these variables instead of the un-logged form as previously done, makes the effective relationship non-linear, while still preserving the linear model.
- It is also now more normally distributed after log-transforming

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

ANSWER ##

- **"cylinders" and "displacement"** are still insignificant
- This may be due to the correlation simply not being strong enough or multi-collinearity issues exist, undermining the significance of these variables.

b) Taking a closer look at "weight", because it seems to be a major explanation of mpg

i) Create a regression of mpg over weight from the original cars dataset

```
regr_wt <- lm(mpg ~ weight, data = auto)
summary(regr_wt)
```

```
##
## Call:
## lm(formula = mpg ~ weight, data = auto)
##
## 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 of log.mpg. on log.weight. from cars_log

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

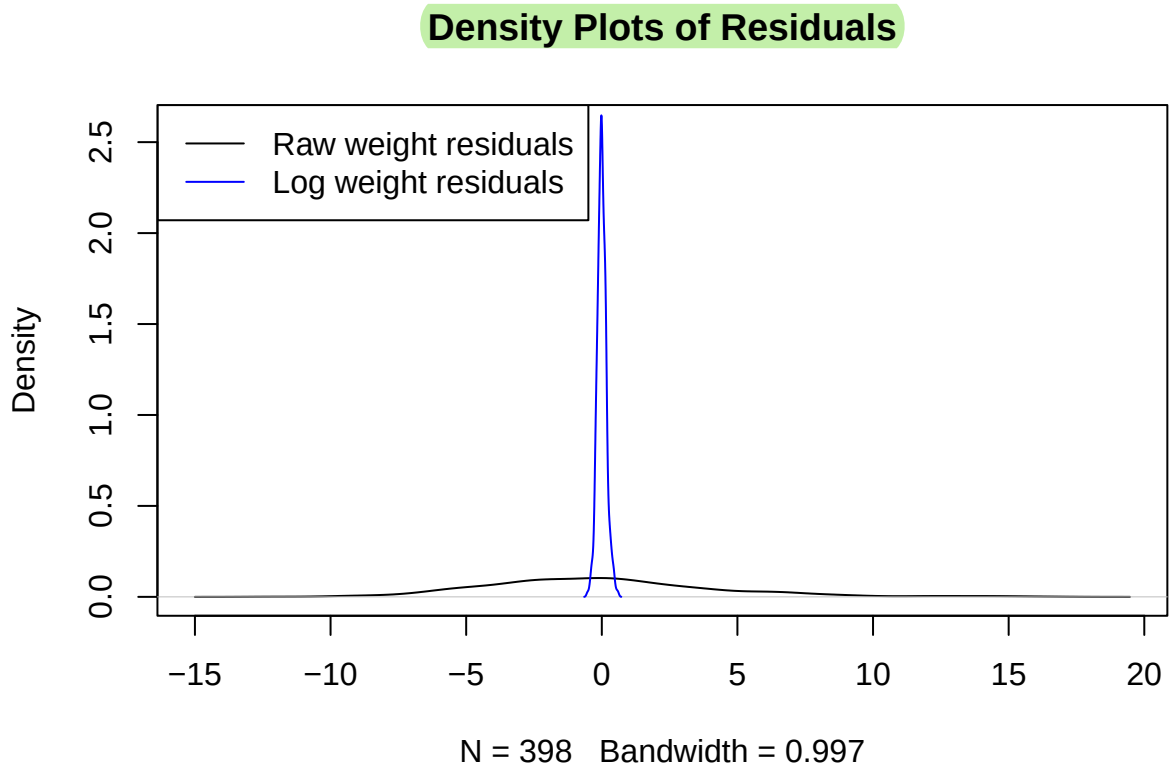
```
##
## Call:
## lm(formula = log.mpg. ~ log.weight., data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52408 -0.10441 -0.00805  0.10165  0.59384
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.5219     0.2349   49.06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

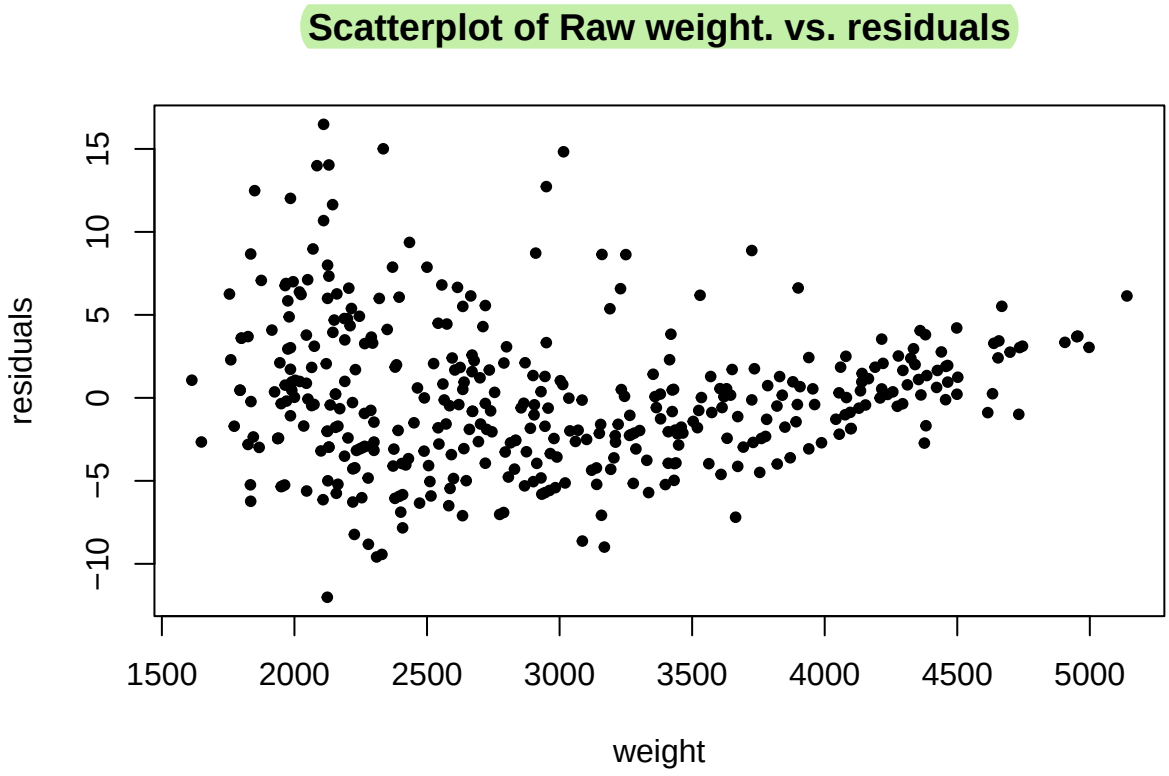
```
# Raw
plot(density(regr_wt$residuals), col = "black", ylim = c(0, 2.6),
    main = "Density Plots of Residuals")

# log-transformed
lines(density(regr_wt_log$residuals), col = "blue")
legend("topleft", lty = 1, c("Raw weight residuals", "Log weight residuals"),
    col = c("black", "blue"))
```

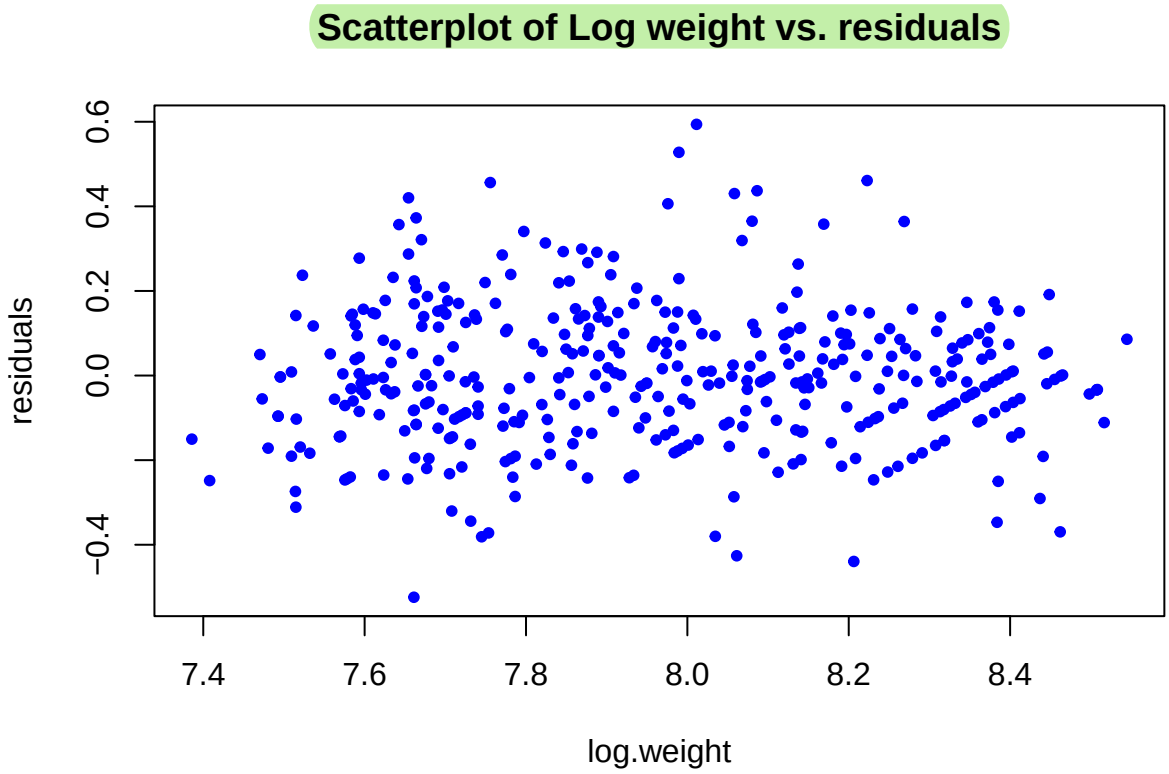


- 2. Scatterplot of log.weight. vs. residuals

```
# Raw
plot(auto$weight, regr_wt$residual, pch = 20, xlab = "weight",
    ylab = "residuals", main = "Scatterplot of Raw weight. vs. residuals")
```




```
# log-transformed
plot(cars_log$log.weight., regr_wt_log$residuals, col = "blue",
    pch = 20, xlab = "log.weight", ylab = "residuals", main = "Scatterplot of Log weight vs. residuals")
```



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

ANSWER ##

- From the density plot, we can see that the distribution using **"log.weight"** from the log-transformed regression model produces better residuals for the assumptions of regression.
- We can see that these residuals are more normally distributed, centralized with a mean around zero.

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

ANSWER ##

- A **1% increase** in **"weight"** will change **"mpg"** by **-1.0583%**.

c) Let's examine the 95% confidence interval of the slope of log.weight. vs. log.mpg

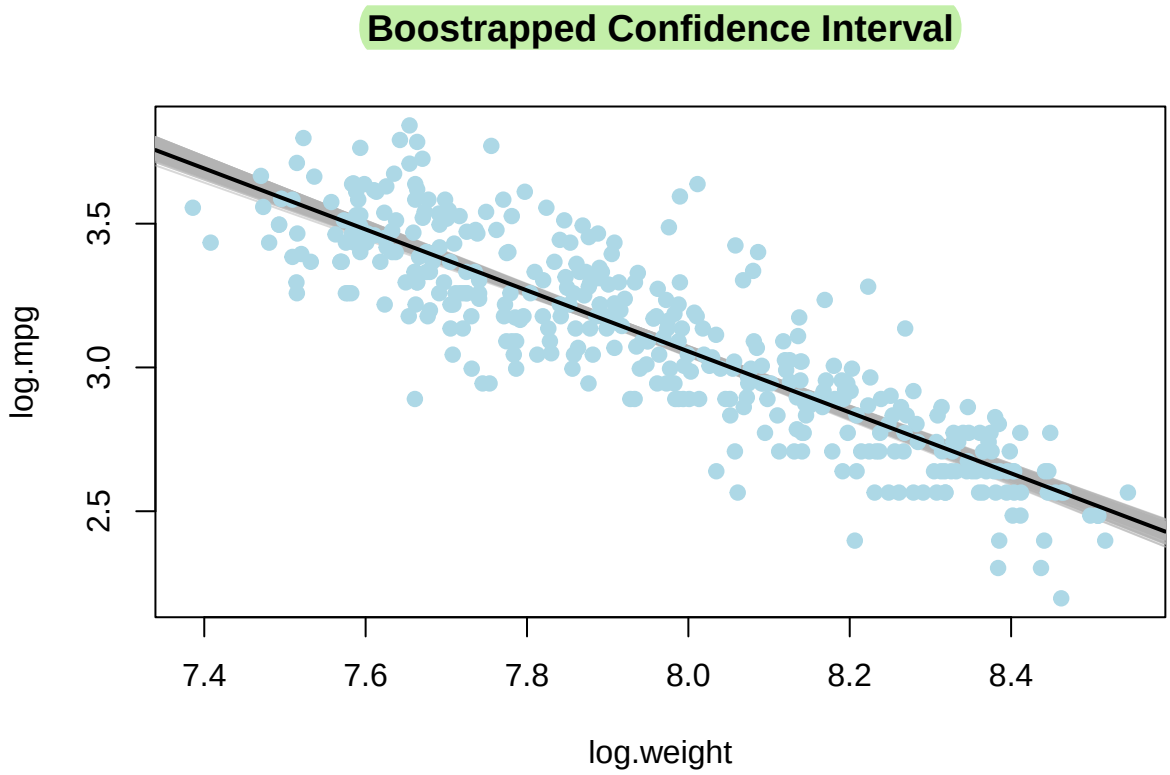
i) Create a bootstrapped confidence interval

```
# Plot empty canvas
plot(cars_log$log.weight., cars_log$log.mpg., col = NA, pch = 19,
    xlab = "log.weight", ylab = "log.mpg", main = "Boostrapped Confidence Interval")
```

```r
# Function for single re-sampled regression line
boot_regr <- function(model, dataset) {
    boot_index <- sample(1:nrow(dataset), replace = TRUE)
    data_boot <- dataset[boot_index, ]
    regr_boot <- lm(model, data = data_boot)
    abline(regr_boot, lwd = 1, col = rgb(0.7, 0.7, 0.7, 0.5))
    regr_boot$coefficients
}

# Bootstrapping for confidence interval
coeffs <- replicate(300, boot_regr(log.mpg. ~ log.weight., cars_log))

# Plot points and regression line
points(cars_log$log.weight., cars_log$log.mpg., col = "lightblue",
    pch = 19)
abline(a = mean(coeffs["(Intercept)", ]), b = mean(coeffs["log.weight.",
    ]), lwd = 2)
```

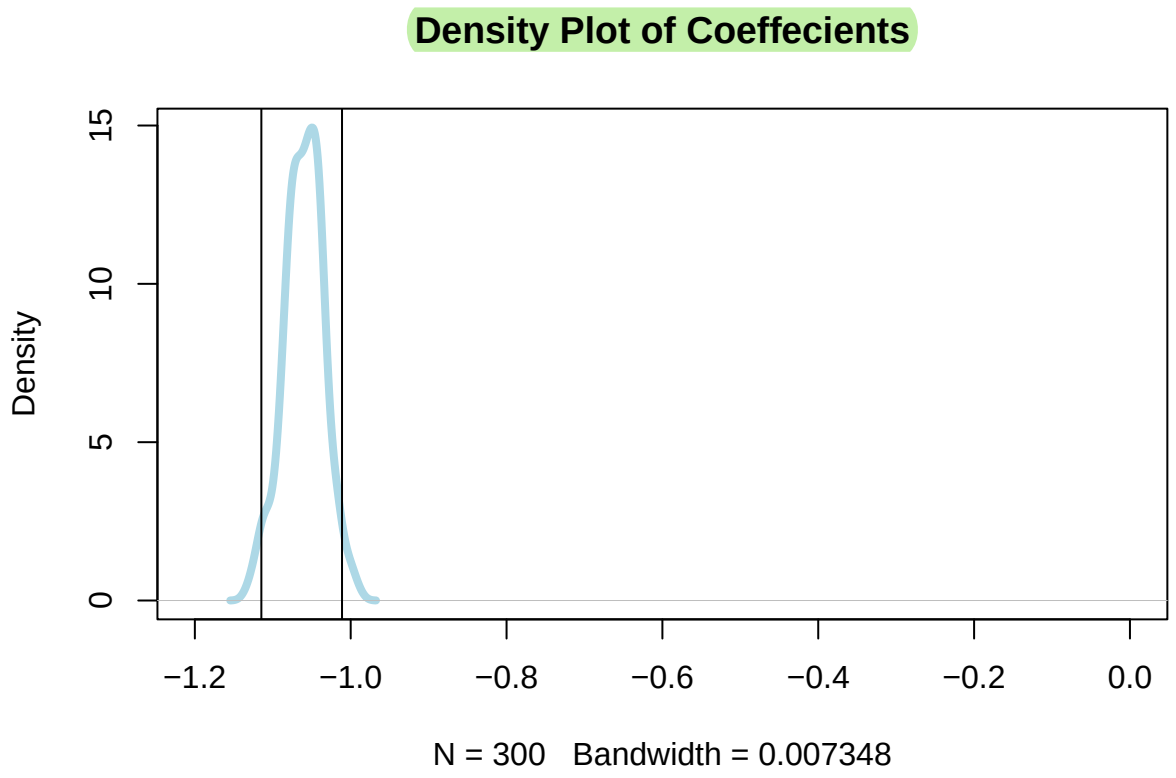


ii) Verify your results with a confidence interval using traditional statistics

```r
# Confidence interval values at 95% CI
quantile(coeffs["log.weight.", ], c(0.025, 0.975))
```

```
ANSWER ## ->      2.5%     97.5%
ANSWER ## -> -1.114626 -1.011119
```

```
# Plot confidence interval of coefficient
plot(density(coeffs["log.weight.", ]), xlim = c(-1.2, 0), col = "lightblue",
    lwd = 4, main = "Density Plot of Coeffecients")
abline(v = quantile(coeffs["log.weight.", ], c(0.025, 0.975)))
```




```
# Run regression
lm(log.mpg. ~ log.weight., cars_log)
```

```
ANSWER ## ->
ANSWER ## -> Call:
ANSWER ## -> lm(formula = log.mpg. ~ log.weight., data = cars_log)
ANSWER ## ->
ANSWER ## -> Coefficients:
ANSWER ## -> (Intercept)  log.weight.
ANSWER ## ->      11.522       -1.058
```

```
verify <- confint(regr_wt_log)  # Using 'confint' function to further verify
require(knitr)  # For creating tables with kable function
kable(verify, caption = "Verified Coefficients", align = "c")
```

Table 1: Verified Coefficients

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

2) Let's tackle multicollinearity. 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

```
# Run regression
log_weight <- lm(log.weight. ~ log.cylinders. + log.displacement. +
    log.horsepower. + log.acceleration. + model_year + factor(origin),
    data = cars_log)
# R^2
sum_log_weight <- summary(log_weight)  # Need summary to read regression model output
r_squared_weight <- sum_log_weight$r.squared  # 0.9431014
r_squared_weight  # Print
```

```
ANSWER ## -> [1] 0.9431014
```

```
# VIF
vif_weight <- 1/(1 - r_squared_weight)  # 17.57512
vif_weight  # Print
```

```
ANSWER ## -> [1] 17.57512
```

- Since VIF of "log.weight." is greater than 5, we have high multi-collinearity.
- Thus, "log.weight." shares more than half its variance with other independent variables.

b) Using Stepwise VIF Selection to remove highly collinear predictors

```
require(car)  # For VIF function
```

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

```
vif_log <- vif(regr_log)
kable(vif_log, caption = "vif of regr_log", align = "c")  # Print table with VIF scores
```

Table 2: vif of 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) Eliminating the single independent variable with largest VIF score greater than 5

```
# Running regression without displacement (highest vif
# score of 29.6)
regr_log2 <- lm(log.mpg. ~ log.cylinders. + log.horsepower. +
    log.weight. + log.acceleration. + model_year + factor(origin),
    data = cars_log)
vif_log2 <- vif(regr_log2)
kable(vif_log2, caption = "vif of regr_log2", align = "c")  # Print table with vif scores
```

Table 3: vif of regr_log2

	GVIF	Df	GVIF^(1/(2*Df))
log.cylinders.	5.433108	1	2.330903
log.horsepower.	12.114475	1	3.480585
log.weight.	11.239740	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

```
# Running regression without horsepower (new highest VIF
# score of 12.1)
regr_log3 <- lm(log.mpg. ~ log.cylinders. + log.weight. + log.acceleration. +
    model_year + factor(origin), data = cars_log)
vif_log3 <- vif(regr_log3)
kable(vif_log3, caption = "vif of regr_log3", align = "c")  # Print table with VIF scores
```

Table 4: vif of regr_log3

	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

	GVIF	Df	GVIF^(1/(2*Df))
model_year	1.201815	1	1.096273
factor(origin)	1.792784	2	1.157130

```
# Running regression without 'cylinders' (new highest vif
# score of 5.3)
regr_log4 <- lm(log.mpg. ~ log.weight. + log.acceleration. +
    model_year + factor(origin), data = cars_log)
vif_log4 <- vif(regr_log4)
kable(vif_log4, caption = "vif of regr_log4", align = "c")  # Print table with VIF scores
```

Table 5: vif of regr_log4

	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 smmary statistics

```
regr_log4 <- lm(log.mpg. ~ log.weight. + log.acceleration. +
    model_year + factor(origin), data = cars_log)
summary(regr_log4)  # Print
```

```
##
## 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
```

ANSWER ##

- We fixed the multi-collinearity issue by removing the predictors "displacement", "horsepower", and then "cylinders".

c) Using stepwise VIF selection, have we lost any variables that were previously significant?

ANSWER ##

- We lost "horsepower", which was previously significant after log-transforming it.
- A good model fit means we adjusted the parameters in the model to improve accuracy.
- Since we have done that by simplifying the regression model, I don't believe we hurt our explanation, instead, I believe simplicity is preferred.

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?

ANSWER ##

- A **VIF of 1** means that there is no correlation among the kth predictor and the remaining predictor variables
- Source: https://cran.r-project.org/web/packages/olsrr/vignettes/regression_diagnostics.html

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?

ANSWER ##

- If the VIF is greater than or equal to 5, R^2 should be greater than 0.8
- If VIF is greater than or equal to 10, R^2 should be greater than 0.9

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

a) Adding three separate regression lines on the scatterplot, one for each of the origins

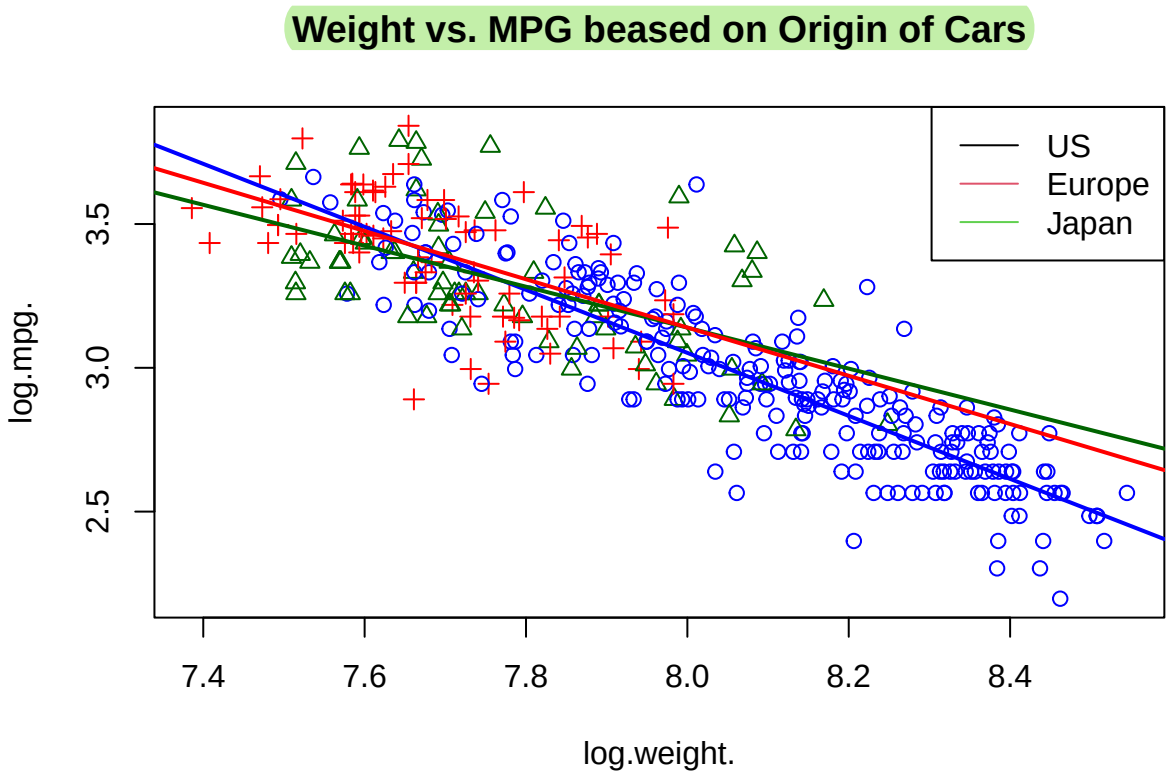
```
# plotting all the weights, using different colors and
# symbols for the three origins
origin_colors = c("blue", "darkgreen", "red")
with(cars_log, plot(log.weight., log.mpg., pch = origin, col = origin_colors[origin],
    main = "Weight vs. MPG beased on Origin of Cars"))

# Plot US regression line
```

```
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)

# Plot europe regression line
cars_us <- subset(cars_log, origin == 2)
wt_regr_eu <- lm(log.mpg. ~ log.weight., data = cars_us)
abline(wt_regr_eu, col = origin_colors[2], lwd = 2)

# Plot Japan regression line
cars_us <- subset(cars_log, origin == 3)
wt_regr_jap <- lm(log.mpg. ~ log.weight., data = cars_us)
abline(wt_regr_jap, col = origin_colors[3], lwd = 2)
legend("topright", lty = 1, c("US", "Europe", "Japan"), col = c(1,
    2, 3))
```



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

ANSWER ##

- Cars from USA tend to be a heavier weight and reach the lowest values of mpg, but it's difficult to make any conclusions since there is less data on cars from both Europe and Japan tend to be heavier and lower feul efficiency