Business Analytics using Statistical Modeling
Assignment 11

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
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 over 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 non-linear (exponential) relationship with mpg

Question 1

Let's deal with non-linearity first. Create a new dataset that log-transforms several variables from our original dataset:

```
cars <- read.table("../10-auto-data.txt", header = FALSE, na.strings = "?")[-9]
names(cars) <- c("mpg", "cylinders", "displacement", "horsepower", "weight",
                 "acceleration", "model_year", "origin")
cars_log <- with(cars, data.frame(log(mpg), log(cylinders), log(displacement),
                                  log(horsepower), log(weight), log(acceleration),
                                  model_year, origin))
regr_mpg <- lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration +
                 model_year + factor(origin), data = cars)
```

```r
summary(regr_mpg)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + model_year + factor(origin), data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0095 -2.0785 -0.0982  1.9856 13.3608
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.795e+01  4.677e+00  -3.839 0.000145 ***
## cylinders       -4.897e-01  3.212e-01  -1.524 0.128215
## displacement     2.398e-02  7.653e-03   3.133 0.001863 **
## horsepower      -1.818e-02  1.371e-02  -1.326 0.185488
## weight          -6.710e-03  6.551e-04 -10.243  < 2e-16 ***
## acceleration     7.910e-02  9.822e-02   0.805 0.421101
## model_year       7.770e-01  5.178e-02  15.005  < 2e-16 ***
## factor(origin)2  2.630e+00  5.664e-01   4.643 4.72e-06 ***
## factor(origin)3  2.853e+00  5.527e-01   5.162 3.93e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.307 on 383 degrees of freedom
##   (6 observations deleted due to missingness)
## Multiple R-squared:  0.8242, Adjusted R-squared:  0.8205
## F-statistic: 224.5 on 8 and 383 DF,  p-value: < 2.2e-16
```

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

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

```r
summary(regr_mpg_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
```

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

`log.horsepower.`, `log.weight.`, and `log.acceleration.` have significant effects on `log.mpg` at 10% significance.

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

`horsepower` and `acceleration` are now significant because of the log transformation.

iii. Which factors still have insignificant or opposite effect on mpg? Why might this be?

`cylinders` still has insignificant effect, meanwhile `displacement` has the opposite effect on `mpg`. This might be due to multicollinearity.

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 on weight from the original cars dataset

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

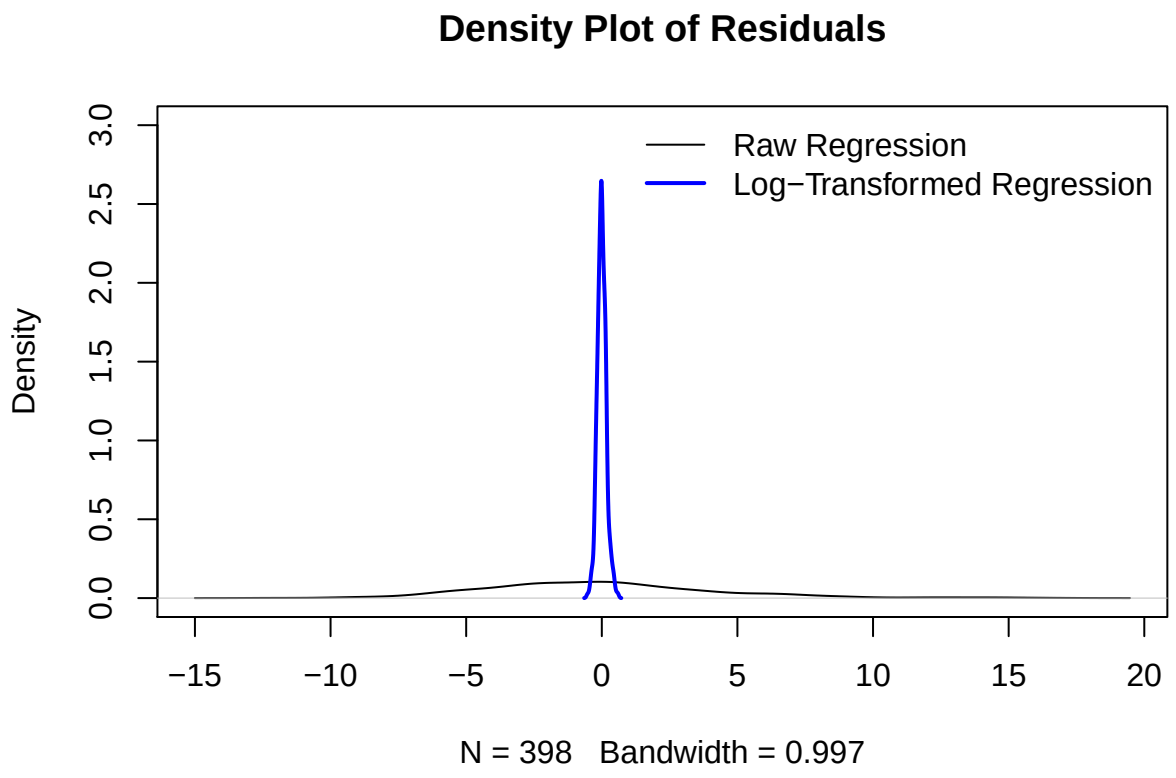
ii. Create a regression (call it regr_wt_log) of log.mpg. on log.weight. from cars_log

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

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

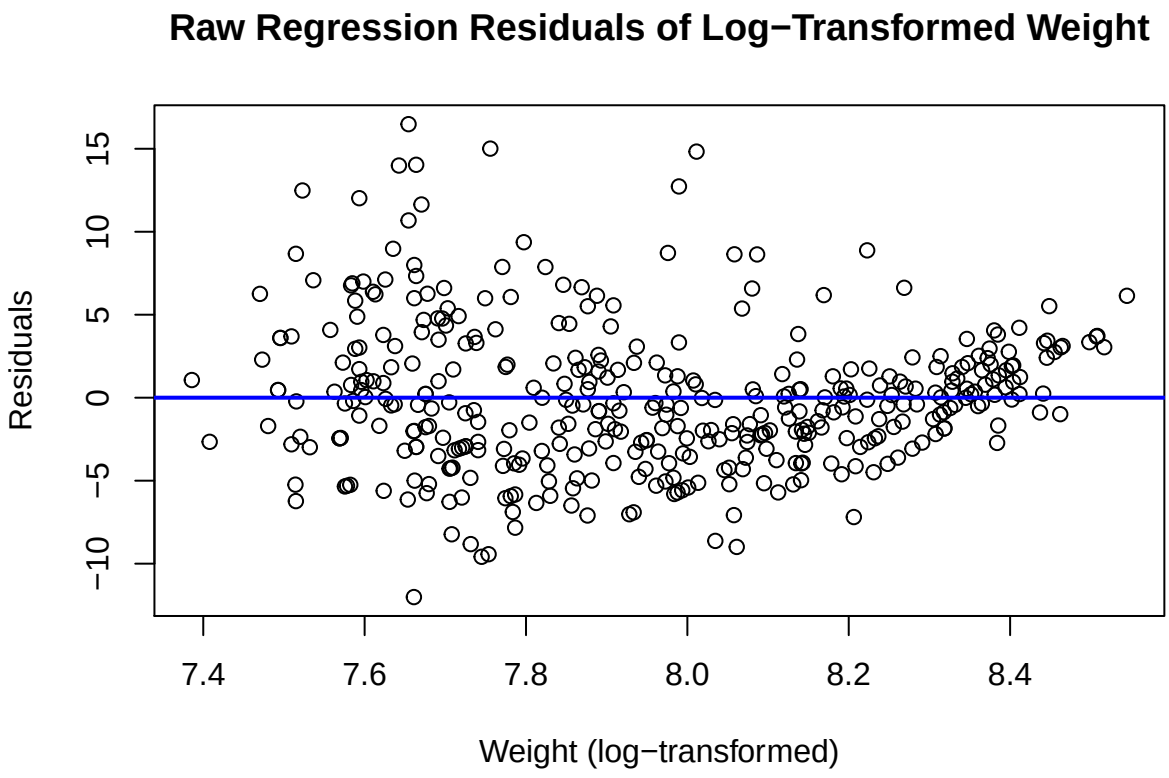
1. density plots of residuals

```
plot(density(regr_wt$residuals), ylim = c(0, 3), main = 'Density Plot of Residuals')
lines(density(regr_wt_log$residuals), col = 'blue', lwd = 2)
legend('topright',legend = c('Raw Regression', 'Log-Transformed Regression'),
       col = c('black', 'blue'), lwd = c(1, 2), bty = 'n')
```

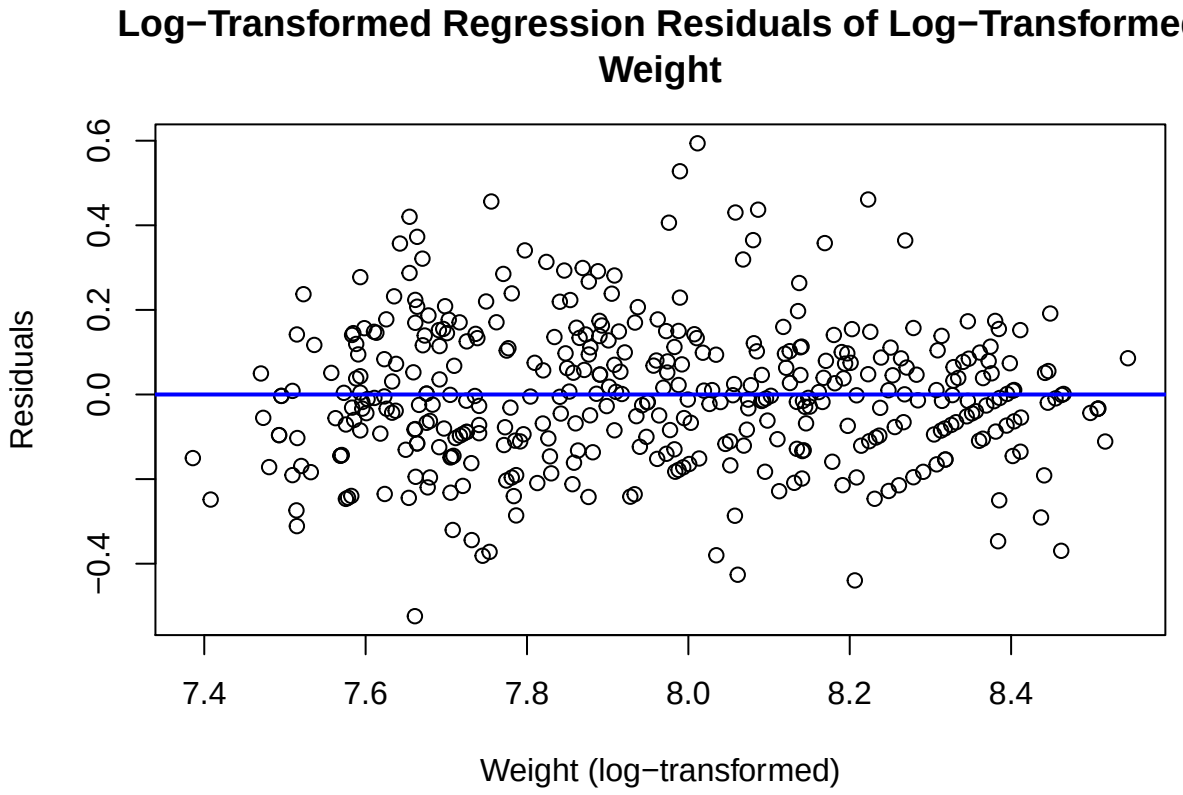


2. scatterplot of log.weight. vs. residuals

```
plot(cars_log$log.weight., regr_wt$residuals, xlab = 'Weight (log-transformed)',
     ylab = 'Residuals',
     main = 'Raw Regression Residuals of Log-Transformed Weight')
abline(h = 0, col = 'blue', lwd = 2)
```



```r
plot(cars_log$log.weight., regr_wt_log$residuals, xlab = 'Weight (log-transformed)',
     ylab = 'Residuals',
     main = 'Log-Transformed Regression Residuals of Log-Transformed\nWeight')
abline(h = 0, col = 'blue', lwd = 2)
```



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

The log-transformed regression produces better residuals as it more follows the assumptions of regression that residuals are random, normally disributed, have a mean of zero, and have a variance that is the same for all values.

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

```r
summary(regr_wt_log)$coefficients
```

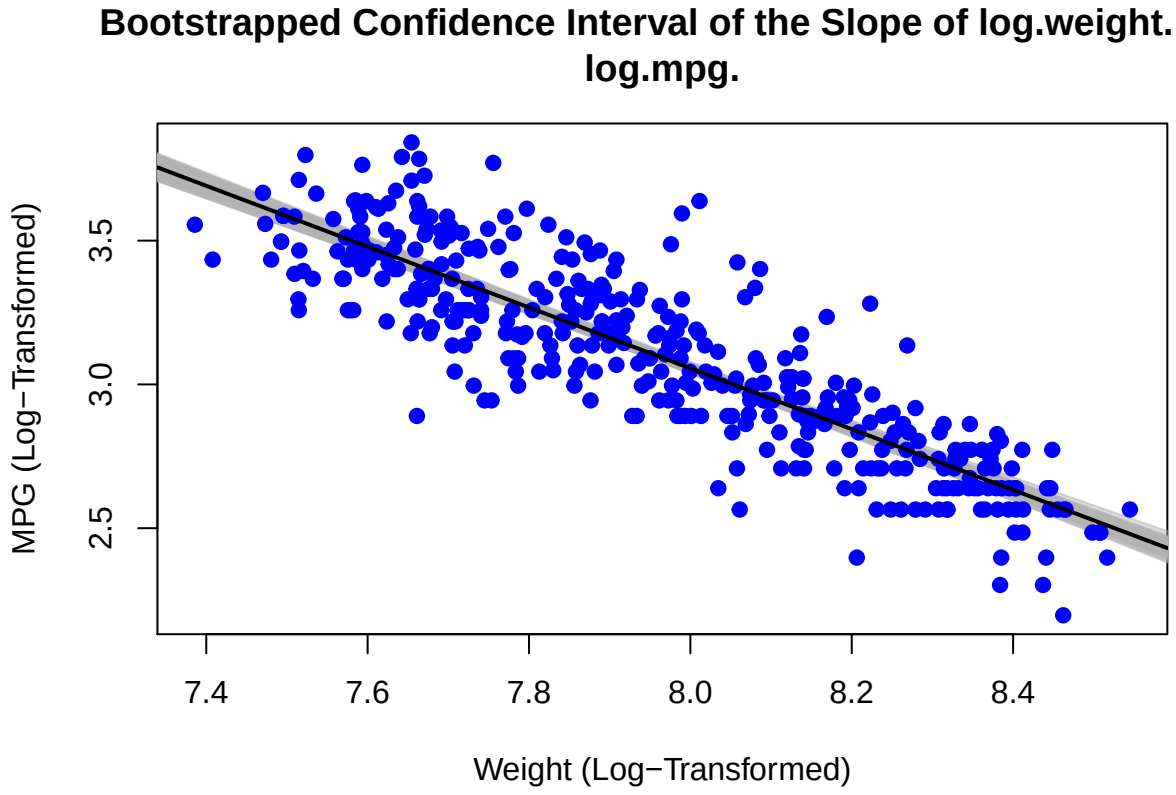
```
##              Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) 11.521907 0.23487232  49.05604 2.319931e-170
## log.weight. -1.058268 0.02949981 -35.87373 1.752225e-126
```

1% increase in weight leads to 1.06% decrease in mpg.

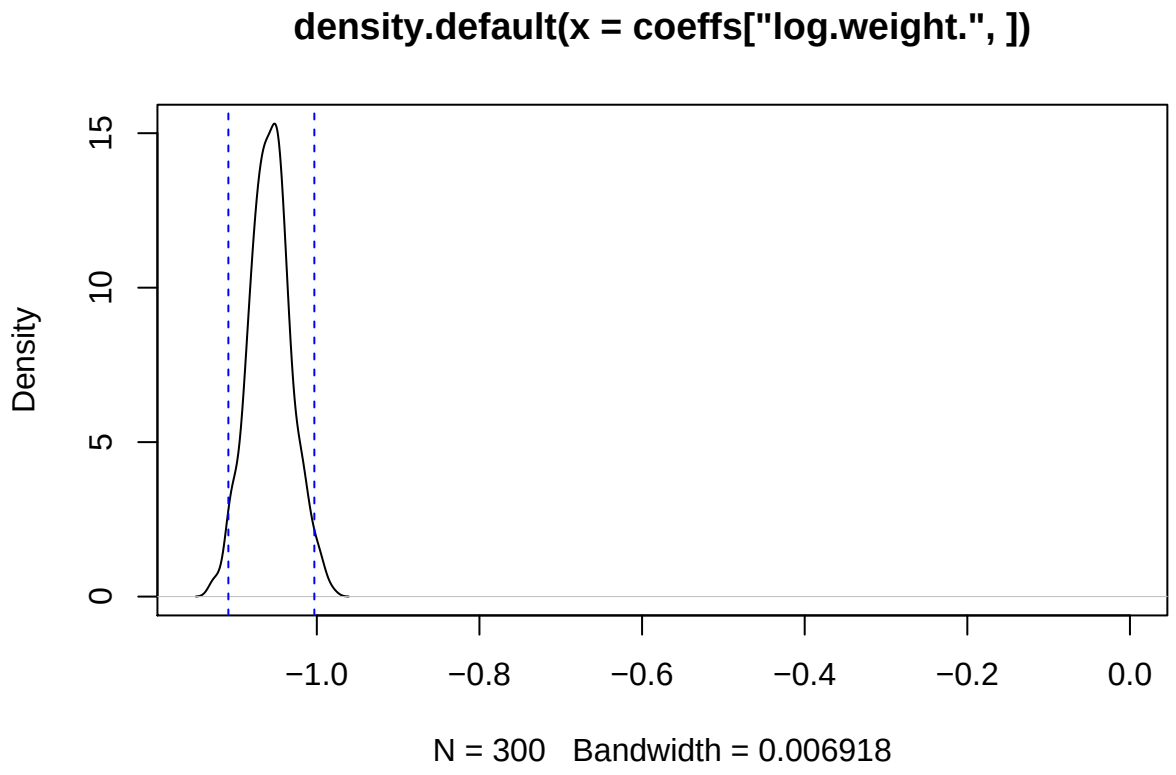
c. What is the 95% confidence interval of the slope of log.weight. vs. log.mpg.?

i. Create a bootstrapped confidence interval

```
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, col = rgb(0.7, 0.7, 0.7, 0.5))
  return(regr_boot$coefficients)
}
plot(cars_log$log.weight., cars_log$log.mpg., col = NA, ylab = 'MPG (Log-Transformed)',
     xlab = 'Weight (Log-Transformed)',
     main = 'Bootstrapped Confidence Interval of the Slope of log.weight. vs\nlog.mpg.')
coeffs <- replicate(300, boot_regr(log.mpg. ~ log.weight., cars_log))
points(cars_log$log.weight., cars_log$log.mpg., col = 'blue', pch = 19)
abline(a = mean(coeffs['(Intercept)', ]), b = mean(coeffs['log.weight.', ]), lwd = 2)
```



```
plot(density(coeffs['log.weight.', ]), xlim = c(-1.15, 0))
ci_95 <- quantile(coeffs['log.weight.', ], c(0.025, 0.975))
abline(v = ci_95, col = 'blue', lty = 2)
```




```
ci_95
```

```
##      2.5%     97.5%
## -1.108775 -1.003050
```

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

```
confint(regr_wt_log)['log.weight.', ]
```

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

Question 2

Let's tackle multicollinearity next.

a. Using regression and R2, calculate the VIF of log.weight. as demonstrated in class

```
regr_wt_log2 <- lm(log.weight. ~ log.cylinders. + log.displacement. + log.horsepower. +
                     log.acceleration. + model_year + factor(origin), data = cars_log)
vif_wt_log <- 1 / (1 - summary(regr_wt_log2)$r.squared)
vif_wt_log
```

```
## [1] 17.57512
```

b. Let's try a procedure called Stepwise VIF Selection to remove highly collinear variables.

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

```
library(car)
```

```
## Warning: package 'car' was built under R version 3.2.5
```

```
vif(regr_mpg_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. Remove the independent variable with the largest VIF score greater than 5 from the model

```
which.max(vif(regr_mpg_log)[, 'GVIF'])
```

```
## log.displacement.
##                 2
```

```
regr_mpg_log2 <- lm(log.mpg. ~ log.cylinders. + log.horsepower. + log.weight. +
                     log.acceleration. + model_year + factor(origin), data = cars_log)
vif(regr_mpg_log2)
```

```
##                       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 large VIF scores

```
which.max(vif(regr_mpg_log2)[, 'GVIF'])
```

```
## log.horsepower.
##               2
```

```
regr_mpg_log3 <- lm(log.mpg. ~ log.cylinders. + log.weight. + log.acceleration. +
                      model_year + factor(origin), data = cars_log)
vif(regr_mpg_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
## model_year        1.201815  1        1.096273
## factor(origin)    1.792784  2        1.157130
```

```
which.max(vif(regr_mpg_log3)[, 'GVIF'])
```

```
## log.cylinders.
##              1
```

```
regr_mpg_log4 <- lm(log.mpg. ~ log.weight. + log.acceleration. + model_year +
                      factor(origin), data = cars_log)
vif(regr_mpg_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 summary statistics

```
summary(regr_mpg_log4)
```

```
##
## 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, was it reasonable to drop those variables? (hint: look at model fit)

```
summary(regr_mpg_log)$adj.r.squared
```

```
## [1] 0.8896522
```

```
summary(regr_mpg_log4)$adj.r.squared
```

```
## [1] 0.8841169
```

`log.horsepower.` was previously significant, but we have lost it in the new model.

The adjusted R squared did not change too much after removing 3 variables. Therefore, it is reasonable to remove them, in order to get a simpler model with a more-less similar model fit.

d. General questions on VIF:

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

Its 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?

```
vif = 5
cor_x1_x2 <- sqrt(1 - (1 / vif))
cor_x1_x2
```

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

To get VIF scores of 5 or higher, X1 and X2 have to have a correlation of at least 0.8944.

```
vif = 10
cor_x1_x2 <- sqrt(1 - (1 / vif))
cor_x1_x2
```

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

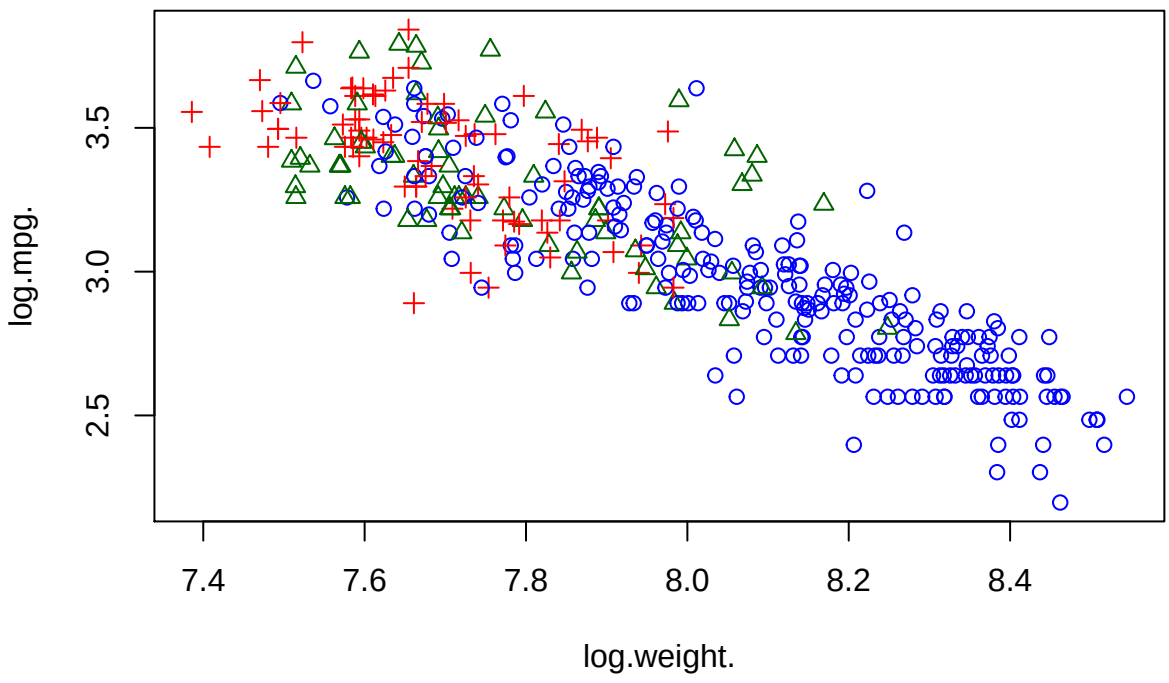
To get VIF scores of 10 or higher, X1 and X2 have to have a correlation of at least 0.9487.

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:

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



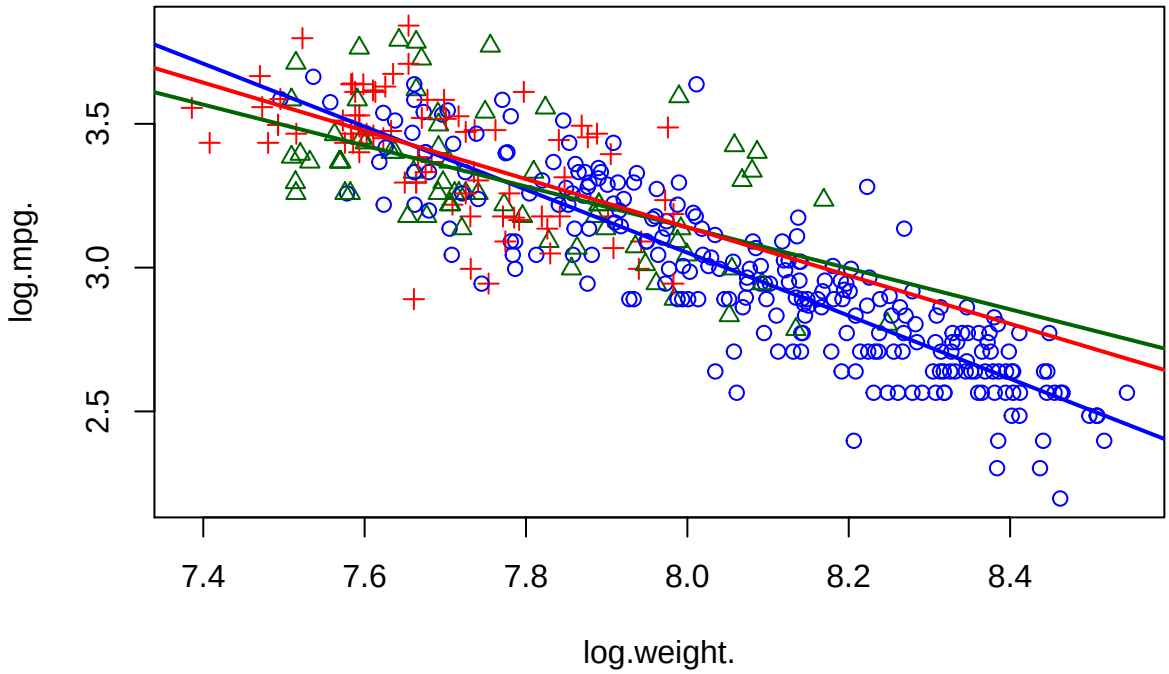
a. Let's add three separate regression lines on the scatterplot, one for each of the origins.

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

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)

cars_eu <- subset(cars_log, origin == 2)
wt_regr_eu <- lm(log.mpg. ~ log.weight., data = cars_eu)
abline(wt_regr_eu, col = origin_colors[2], lwd = 2)

cars_jpn <- subset(cars_log, origin == 3)
wt_regr_jpn <- lm(log.mpg. ~ log.weight., data = cars_jpn)
abline(wt_regr_jpn, col = origin_colors[3], lwd = 2)
```



b. [not graded]