HW_Week11_108020033

Che-Wei, Chang

2023-04-27

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):

```r
# Import the data
cars <- read.table("auto-data.txt", header = FALSE, na.strings = "?")
names(cars) <- c("mpg",  "cylinders", "displacement", "horsepower", "weight",
                 "acceleration", "model_year", "origin", "car_name")

# Create data frame
cars_log <- with(cars, 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

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

```r
# Preprocess the data
cars$origin <- factor(cars$origin)
cars_log$origin <- factor(cars_log$origin)

# Create model
lm_fit <- lm(log.mpg. ~ log.cylinders. + log.displacement. + log.horsepower. +
             log.weight. + log.acceleration. + model_year + origin, data = cars_log)
summary(lm_fit)
```

```
##
## Call:
## lm(formula = log.mpg. ~ log.cylinders. + log.displacement. +
##     log.horsepower. + log.weight. + log.acceleration. + model_year +
##     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 ***
## origin2            0.050717   0.020920   2.424  0.01580 *
## origin3            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
```

```
# By the table, since the P-value of log.horsepower., log.weight., log.acceleration.,
# model_year, origin2 and origin3 are <= 10%, we can say that these factors have a
# significant effect on log.mpg. at 10% significance
```

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

```
# Create original model
lm_original <- lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration
                  + model_year + origin, data = cars)
summary(lm_original)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + model_year + 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 ***
## origin2       2.630e+00  5.664e-01   4.643 4.72e-06 ***
## origin3       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
```

```
# By the table that we do log_transform before, we can see that if we don't do log_transform,
# then cylinders, horsepower and acceleration don't have a significant effect on mpg at 10%
# significance.This is because log-transforming the variables changes their relationships with
# the dependent variable from nonlinear to linear. Therefore, the results of original and doing
# log-transforming will be different.
```

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

```
# By these two table, we can find that the factor of cylinders is still insignificant.
# It is because log(cylinders) don't have a strong linear relationship with log(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

```
# Create linear model
regr_wt <- lm(mpg ~ weight, data = 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

```
# Create linear model
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)
```

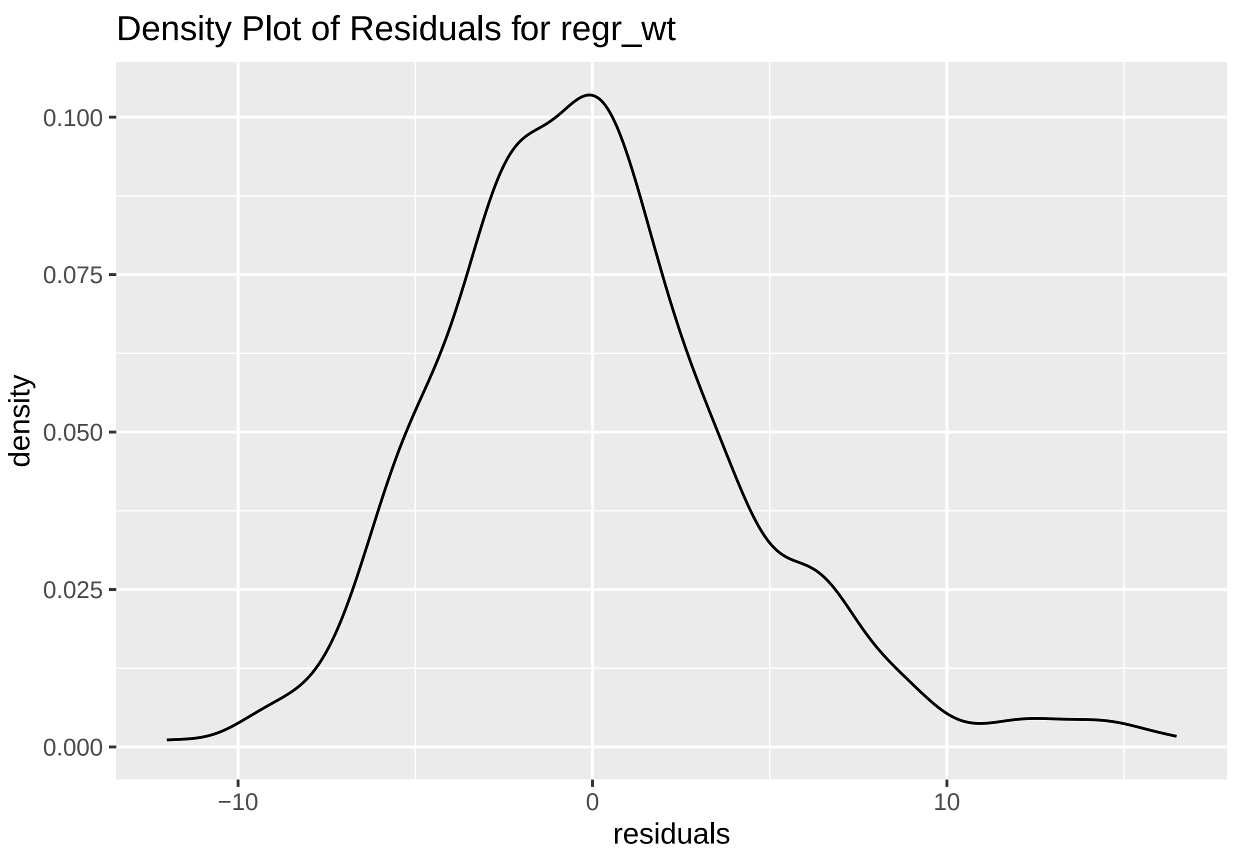
```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52408 -0.10441 -0.00805  0.10165  0.59384
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.5219     0.2349   49.06   <2e-16 ***
## log.weight.  -1.0583     0.0295  -35.87   <2e-16 ***
## ---
## 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):

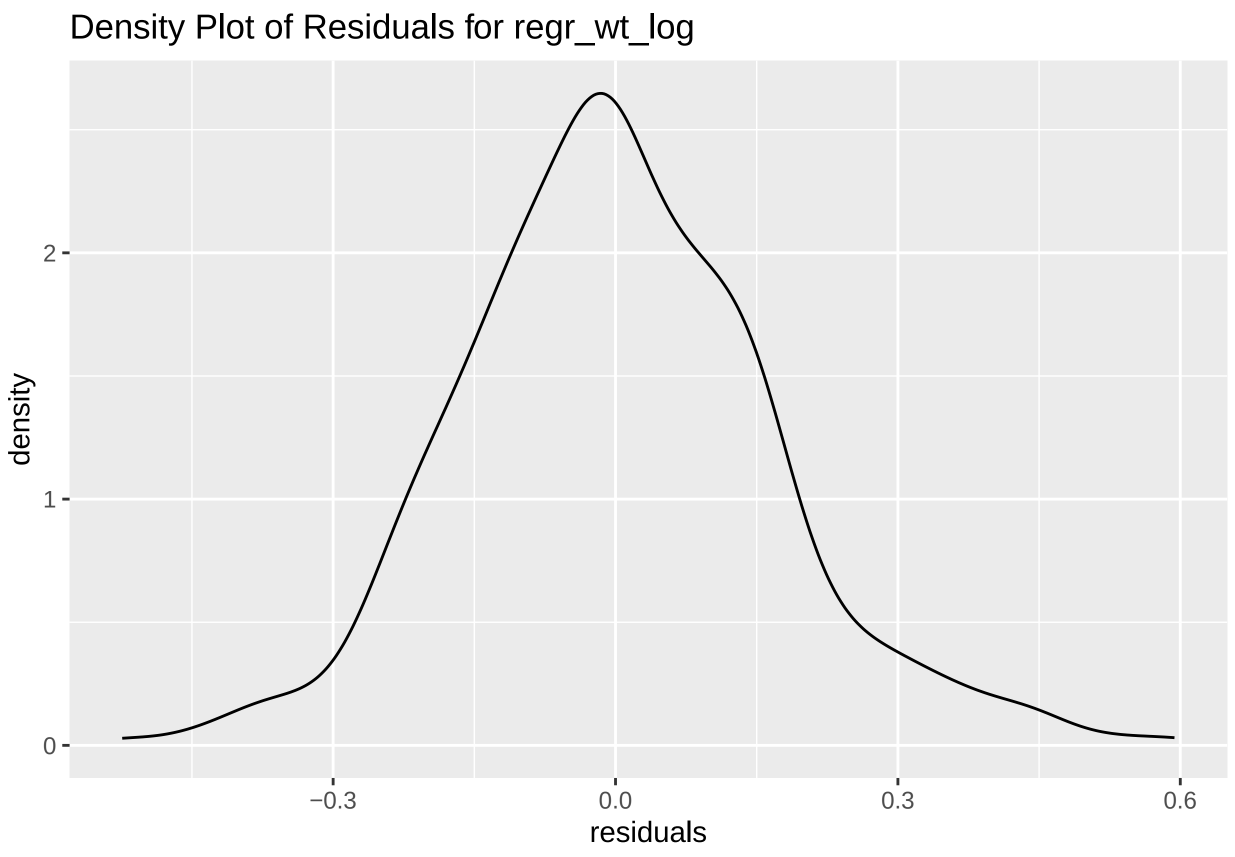
1. density plots of residuals

```r
# Import the library
library(ggplot2)

# Density plot of residuals for regr_wt
ggplot(data.frame(residuals = resid(regr_wt)), aes(x = residuals)) +
  geom_density() +
  ggtitle("Density Plot of Residuals for regr_wt")
```

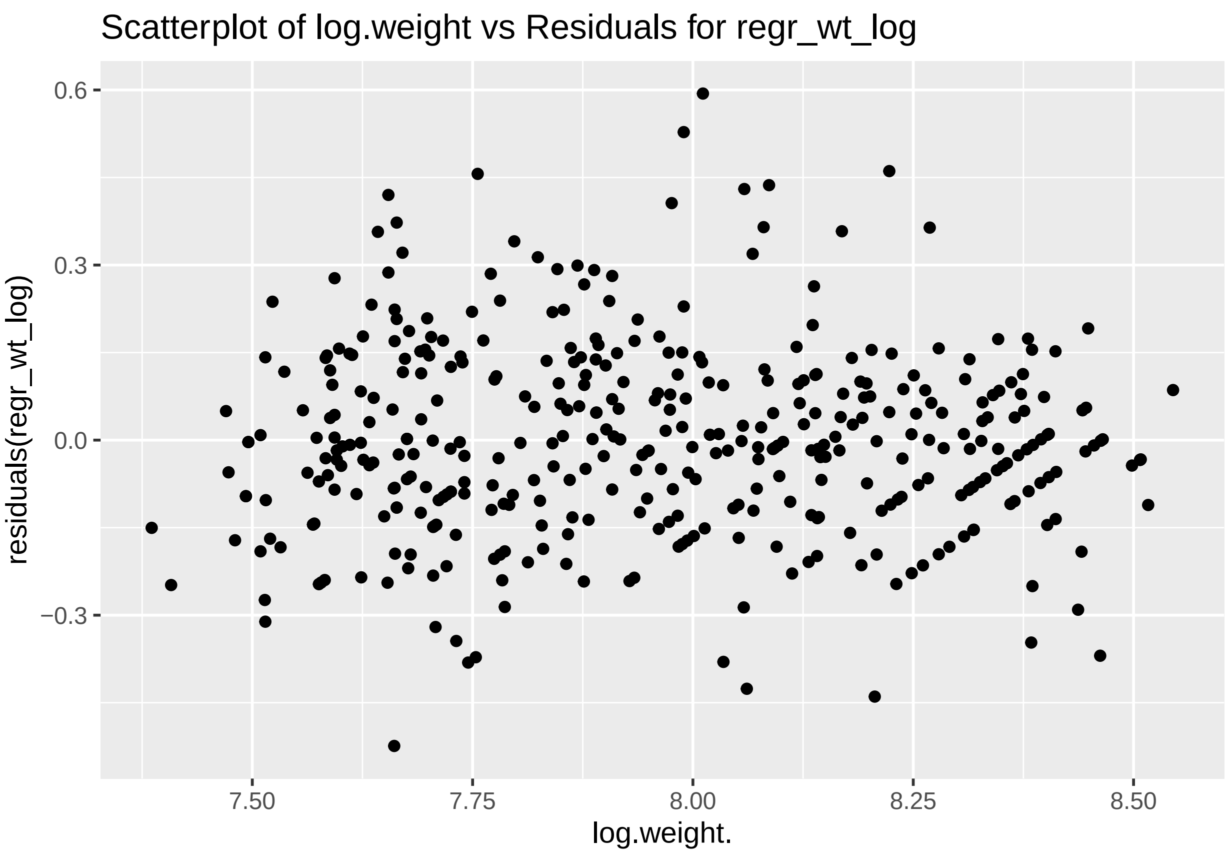


```r
# Density plot of residuals for regr_wt_log
ggplot(data.frame(residuals = resid(regr_wt_log)), aes(x = residuals)) +
  geom_density() +
  ggtitle("Density Plot of Residuals for regr_wt_log")
```



2. scatterplot of log.weight. vs. residuals

```r
# Scatterplot of log.weight vs residuals for regr_wt_log
ggplot(data = cars_log, aes(x = log.weight., y = residuals(regr_wt_log))) +
  geom_point() +
  ggtitle("Scatterplot of log.weight vs Residuals for regr_wt_log")
```



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

```
# The density plot of residuals from the regression of log.mpg on log.weight appears to be
# more symmetrically distributed around 0, indicating that the assumptions of linear
# regression are better met when using the log transformation of the variables.
```

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

```
# If x shows log.weight. and y shows log.mpg., then the slope of of log.weight vs log.mpg can
# be interpreted as how much log.mpg.(y-axis) changes for each units change in the
# log.weight.(x-axis).
```

(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 R-square, compute the VIF of log.weight. using the approach shown in class

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

R_square_weight <- summary(regr_log_weight)$r.squared
VIF_log_weight <- 1 / (1 - R_square_weight)
cat("VIF: ", VIF_log_weight, "\n")
```

```
## VIF:  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!)

```
library(car)
```

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

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

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

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

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

```
# i. use vif(regr_log) to compute VIF
vif_scores <- vif(regr_log)
vif_scores <- as.data.frame(vif_scores)
vif_scores
```

```
##                        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
```

```r
# ii and iii
# select the factors that VIF scores are greater than 5
factor_name <- c()
for (row_name in rownames(vif_scores)) {
  if(vif_scores[row_name, "GVIF"] <= 5) {
    factor_name <- cbind(factor_name, row_name)
  }
}

# Show the result whose VIF scores are smaller or equal to 5
factor_name
```

```
##      row_name            row_name     row_name
## [1,] "log.acceleration." "model_year" "factor(origin)"
```

```r
# Show the vif table whose vif scores are smaller or equal to 5
new_vif_scores <- data.frame()
for (row_name in rownames(vif_scores)) {
  for (i in 1:length(factor_name)) {
    if (factor_name[i] == row_name) {
      new_vif_scores <- rbind(new_vif_scores, vif_scores[row_name,])
    }
  }
}
new_vif_scores
```

```
##                         GVIF Df GVIF^(1/(2*Df))
## log.acceleration. 3.570357  1        1.889539
## model_year        1.303738  1        1.141814
## factor(origin)    2.656795  2        1.276702
```

```r
# Transform the name from factor(origin) to origin.
factor_name[3] <- "origin"
factor_name
```

```
##      row_name            row_name     row_name
## [1,] "log.acceleration." "model_year" "origin"
```

```r
# Leave the data that its factor of vif scores are smaller or equal to 5
# and use these data to create new linear model
sel_cars_log <- subset(cars_log, select = factor_name)
sel_cars_log <- cbind(sel_cars_log, cars_log$log.weight.)
colnames(sel_cars_log)[4] <- "log.weight."

# Create the model
new_regr_model <- lm(log.weight. ~ log.acceleration. + model_year + factor(origin), data = sel_cars_log

# Report final regression model
summary(new_regr_model)
```

```
##
```

```
## Call:
## lm(formula = log.weight. ~ log.acceleration. + model_year + factor(origin),
##     data = sel_cars_log)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5841 -0.1309  0.0041  0.1561  0.5006
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        9.834445   0.234854  41.875  < 2e-16 ***
## log.acceleration. -0.393291   0.061297  -6.416 4.02e-10 ***
## model_year        -0.009049   0.002945  -3.072  0.00227 **
## factor(origin)2   -0.270976   0.028311  -9.571  < 2e-16 ***
## factor(origin)3   -0.346154   0.026972 -12.834  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2032 on 393 degrees of freedom
## Multiple R-squared:  0.4809, Adjusted R-squared:  0.4756
## F-statistic: 91.02 on 4 and 393 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_weight)
```

```
##
## Call:
## lm(formula = log.weight. ~ +log.cylinders. + log.displacement. +
##     log.horsepower. + log.acceleration. + model_year + factor(origin),
##     data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42851 -0.04105 -0.00126  0.03826  0.21177
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        2.501569   0.175203  14.278  < 2e-16 ***
## log.cylinders.    -0.077327   0.036408  -2.124   0.0343 *
## log.displacement.  0.411486   0.027969  14.712  < 2e-16 ***
## log.horsepower.    0.424806   0.027122  15.663  < 2e-16 ***
## log.acceleration.  0.424184   0.028441  14.914  < 2e-16 ***
## model_year         0.004663   0.001034   4.510 8.61e-06 ***
## factor(origin)2    0.067427   0.012054   5.594 4.22e-08 ***
## factor(origin)3    0.016966   0.012326   1.376   0.1695
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06768 on 384 degrees of freedom
##   (6 observations deleted due to missingness)
## Multiple R-squared:  0.9431, Adjusted R-squared:  0.9421
## F-statistic: 909.3 on 7 and 384 DF,  p-value: < 2.2e-16
```

```
# If we use stepwise VIF selection, then we will leave acceleration, model_year, origin.
# (i.e. we remove cylinders displacement horsepower that are significant from original table)
# From the original table and the table we eliminate, we can find that it changes a little
# only the factor(origin)3 changes from insignificant to significant.
# Hence, I think that I hurt a little about our explanation by dropping those variables.
```

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
# R-square will be zero. Besides, VIF = 1 / (1 - R-square). Hence, VIF score will 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?

```
# Since VIF = 1 / (1 - R-square), R-square = cor(X1, X2) ^ 2.
# Hence, VIF = 1 / (1 - cor(X1, X2) ^ 2)
# To get VIF scores of 5 or higher, we set equation: 5 <= 1 / (1 - cor(X1, X2) ^ 2)
# 0.894428 <= cor(X1, X2) < 1 or -0.894428 >= cor(X1, X2) > -1
# To get VIF scores of 10 or higher, we set equation: 10 <= 1 / (1 - cor(X1, X2) ^ 2)
# 0.948683 <= cor(X1, X2) < 1 or -0.948683 >= cor(X1, X2) > -1
```

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:

```
# 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)
```

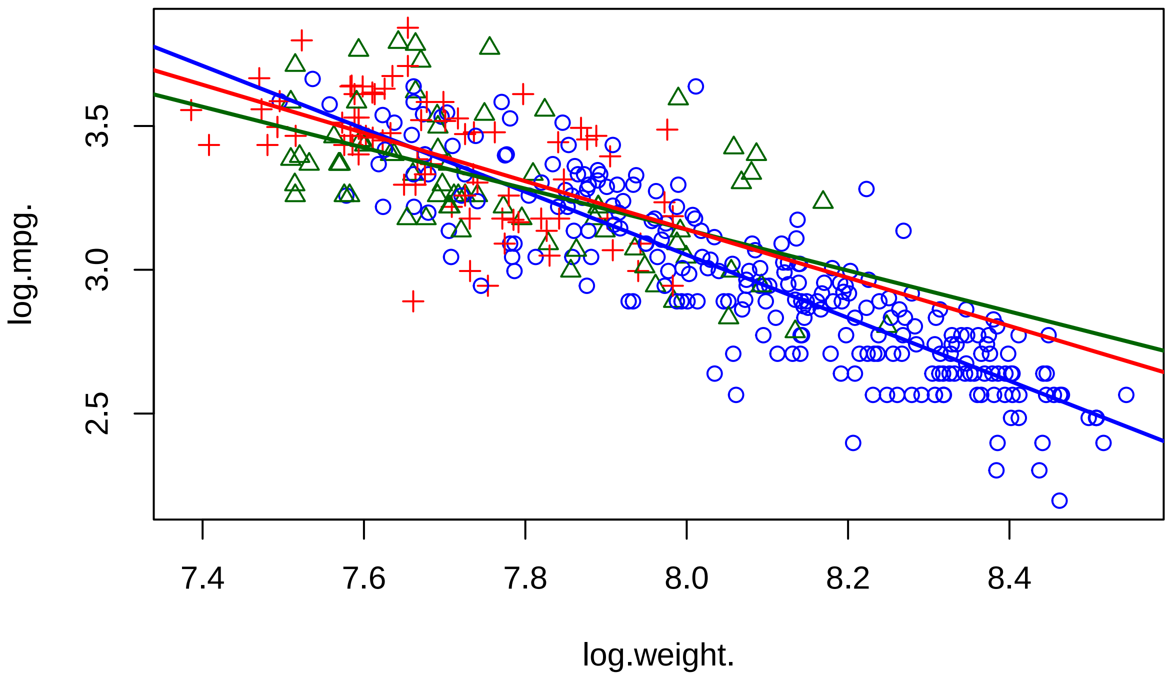
```
# Create scatter plot
origin_colors = c("blue", "darkgreen", "red")
with(cars_log, plot(log.weight., log.mpg., pch=as.numeric(origin), col=origin_colors[origin]))

# Separate data of us and add regression line
cars_us <- subset(cars_log, origin == 1)
wt_regr_us <- lm(log.mpg. ~ log.weight., data = cars_us)
```

```r
abline(wt_regr_us, col = origin_colors[1], lwd = 2)

# Separate data of Euerope and add regression line
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)

# Separate data of Japan and add regression line
cars_jp <- subset(cars_log, origin == 3)
wt_regr_jp <- lm(log.mpg. ~ log.weight., data = cars_jp)
abline(wt_regr_jp, col = origin_colors[3], lwd = 2)
```


log.mpg.
log.weight.