HW11

110078509

20220430

Preface

Let's see the coefficienct of the "un-transform" data. We can tell that cylinders, horsepower , acceleration are not significant to mpg.

```
summary(lm(mpg ~ cylinders+displacement+horsepower+weight+acceleration+model_year+origin, dat
a = auto))
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + model_year + origin, data = auto)
##
## 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
## 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

```
summary(lm(log.mpg. ~ log.cylinders.+log.displacement.+ log.horsepower. +log.weight.+log.acce
leration.+model_year+origin, data = cars_log))
```

```
##
## 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
## Multiple R-squared:  0.8919, Adjusted R-squared:  0.8897
## F-statistic:   395 on 8 and 383 DF,  p-value: < 2.2e-16
```

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

- Ans:

log.horsepower., log.weight., log.acceleration., model_year, origin2 , origin3

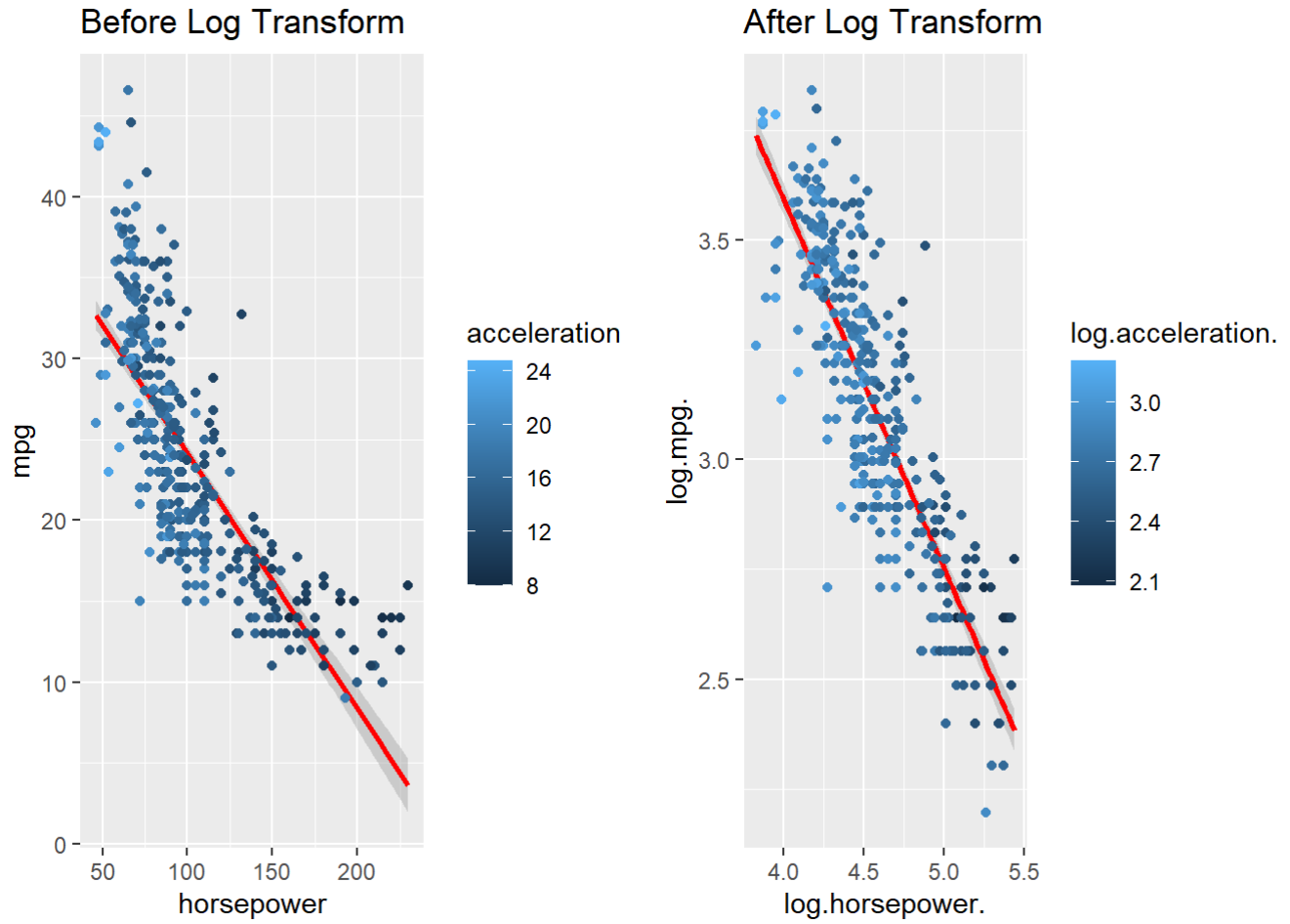
aii. Do some new factors now have effects on mpg, and why might this be?

- Ans:

Yes. Compared to the coefficient of the original dataset, “log.horsepower.”,“log.acceleration.” become significant. Because they had multicollinearity with feature-weight, however, the “weight” had higer correlation to the mpg. Therefore, the linear model gave all the credit to the weight. After transformation, the log function make the model fit the transformed dataset better. To prove it, the graph as bleow:

```
before_log <- auto %>% ggplot( aes(x = horsepower, y =mpg)) +
    geom_smooth(method = "lm", se = TRUE, colour="red", size=1)  +
    geom_point(aes(color = acceleration)) +
    ggtitle('Before Log Transform')

after_log <- cars_log %>% ggplot(aes(x = log.horsepower., y =log.mpg.)) +
    geom_smooth(method = "lm", se = TRUE, colour="red", size=1)  +
    geom_point(aes(color = log.acceleration.))+
    ggtitle('After Log Transform')

grid.arrange(before_log, after_log,ncol = 2, nrow = 1)
```

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



aiii. Which factors still have insignificant or opposite (from correlation) effects on mpg?

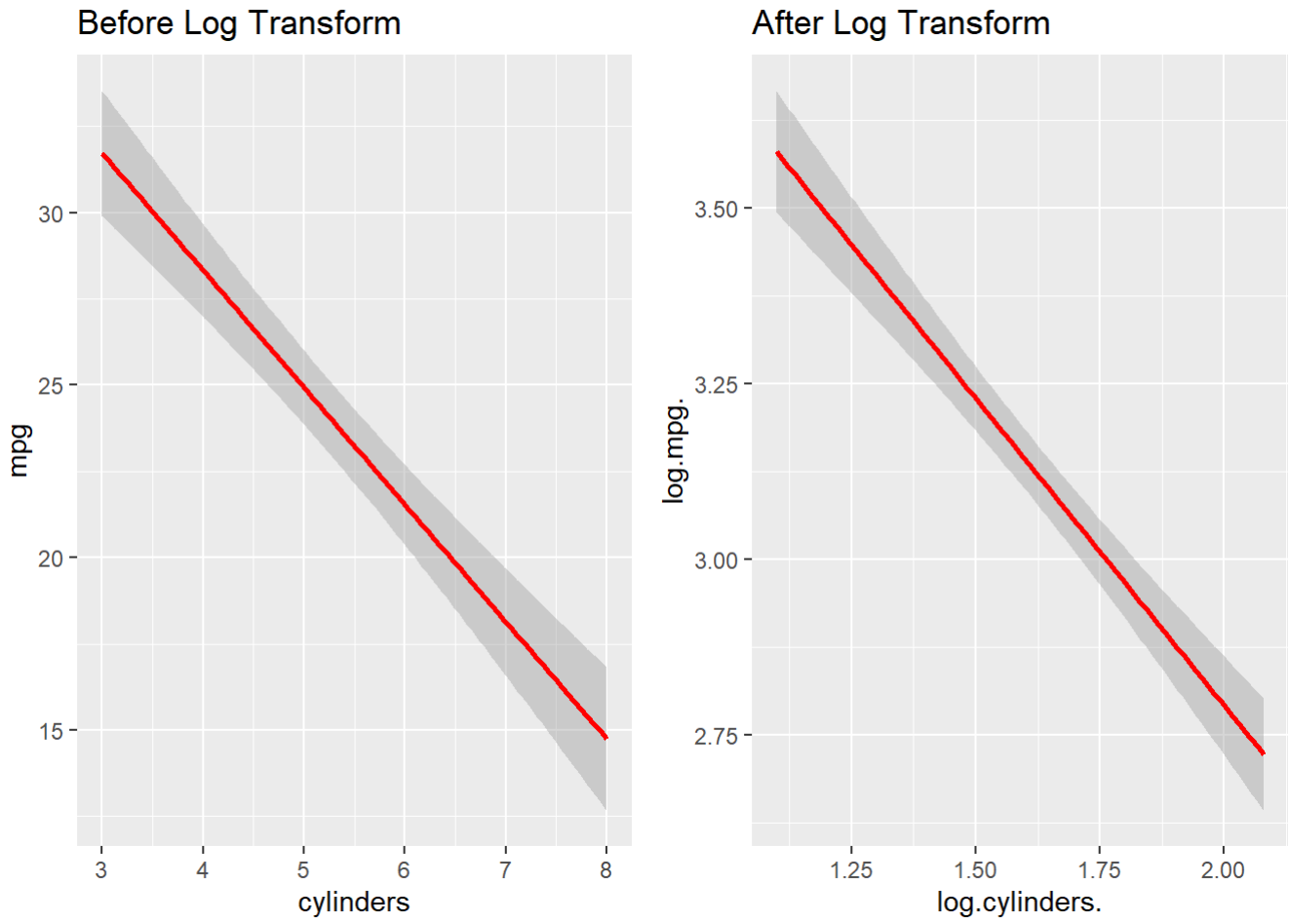
Why might this be?

log.cylinders. The purpose of the log transformation is to greatly reduce the extremely high value and slightly reduce the small value. In my opinion, the cylinder feature indicates the number of the cylinders from 1 to 8. After log transformation, these made no big difference and can't help the model improve in these case.

Here is one simulation using 100 sample form the real dataset for better clarification to the idea.

```
set.seed(110078509)
simulation <- auto[1:2]

random_sample <- simulation[sample(nrow(simulation), 100), ]

log.random_sample <- random_sample %>%
    mutate(log.mpg. = log(mpg)) %>%
     mutate(log.cylinders. = log(cylinders))


before_log <- random_sample %>% ggplot( aes(x = cylinders, y =mpg)) +
    geom_smooth(method = "lm", se = TRUE, colour="red", size=1) +
    ggtitle('Before Log Transform')

after_log <- log.random_sample %>% ggplot(aes(x = log.cylinders., y =log.mpg.)) +
    geom_smooth(method = "lm", se = TRUE, colour="red", size=1)  +
    ggtitle('After Log Transform')

grid.arrange(before_log, after_log,ncol = 2, nrow = 1)
```

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



You can tell that it make no difference after the log transformation. Therefore, it still have insignificant correlation on mpg.

b. Let's take a closer look at weight, because it seems to be a major explanation of mpg

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

```
##
## Call:
## lm(formula = mpg ~ weight, data = auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.9736  -2.7556  -0.3358   2.1379  16.5194
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 46.216524   0.798673   57.87   <2e-16 ***
## weight      -0.007647   0.000258  -29.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.333 on 390 degrees of freedom
## Multiple R-squared:  0.6926, Adjusted R-squared:  0.6918
## F-statistic: 878.8 on 1 and 390 DF,  p-value: < 2.2e-16
```

bi. Create a regression (call it regr_wt) 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
## -11.9736  -2.7556  -0.3358   2.1379  16.5194
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 46.216524   0.798673   57.87   <2e-16 ***
## weight      -0.007647   0.000258  -29.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.333 on 390 degrees of freedom
## Multiple R-squared:  0.6926, Adjusted R-squared:  0.6918
## F-statistic: 878.8 on 1 and 390 DF,  p-value: < 2.2e-16
```

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

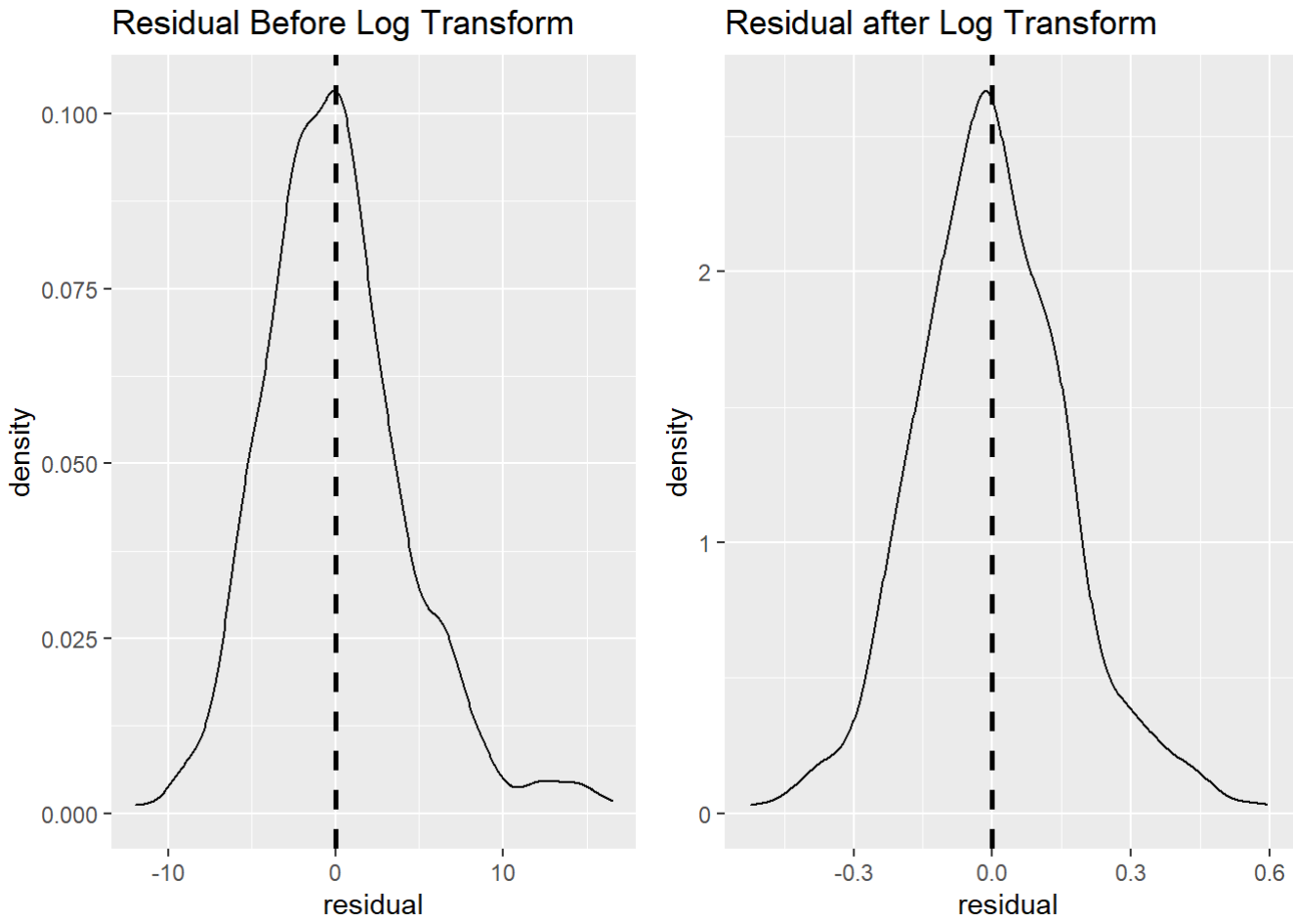
```
regr_wt_log <- lm(log.mpg. ~log.weight., data = cars_log)
summary(regr_wt_log)
```

```
##
## Call:
## lm(formula = log.mpg. ~ log.weight., data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52321 -0.10446 -0.00772  0.10124  0.59445
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.5152     0.2365   48.69   <2e-16 ***
## log.weight.  -1.0575     0.0297  -35.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1651 on 390 degrees of freedom
## Multiple R-squared:  0.7648, Adjusted R-squared:  0.7642
## F-statistic:  1268 on 1 and 390 DF,  p-value: < 2.2e-16
```

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

- (biii-1). density plots of residuals

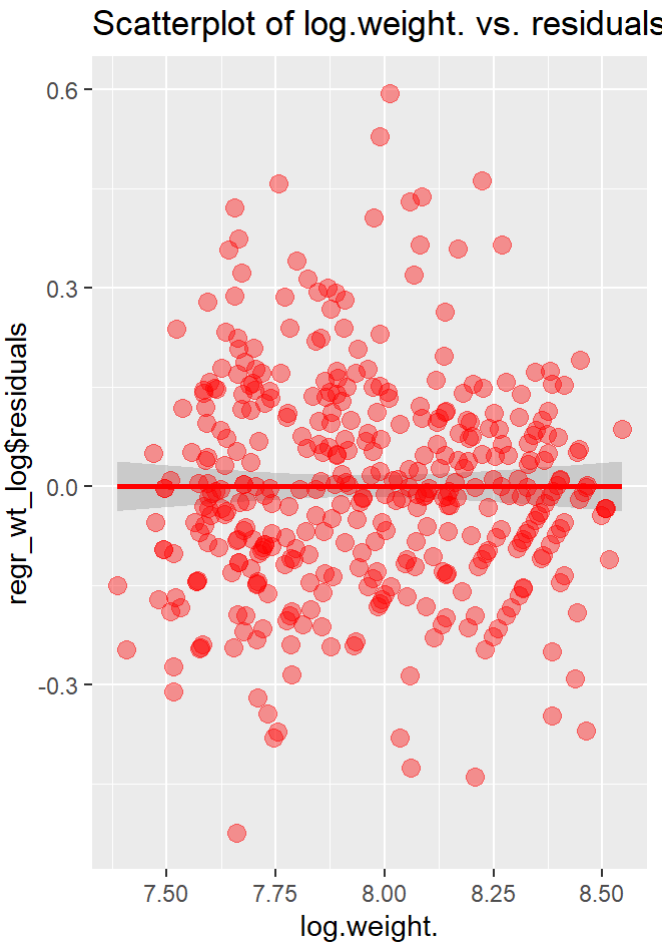
```
regr_wt.df <- data.frame(residual = regr_wt$residuals)
regr_wt_log.df <- data.frame(residual = regr_wt_log$residuals)

resplot1 <- ggplot(regr_wt.df, aes(x=residual)) +
    geom_density()+
    geom_vline(aes(xintercept=mean(residual)),color="black", linetype="dashed", size=1)+
    labs(title  = 'Residual Before Log Transform')

resplot2 <- ggplot(regr_wt_log.df, aes(x=residual)) +
  geom_density()+geom_vline(aes(xintercept=mean(residual)),
            color="black", linetype="dashed", size=1)+labs(title  = 'Residual after Log Trans
form')

# Plot them together
grid.arrange(resplot1, resplot2,ncol = 2, nrow = 1)
```



- (biii-2). Scatterplot of log.weight. vs. residuals

```
before_log <- regr_wt %>% ggplot( aes(x = weight, y = regr_wt$residuals)) +
    geom_smooth(method = "lm", se = TRUE, colour="red", size=1)  +
    geom_point(colour = "blue", size = 3, alpha = 0.4) +
    ggtitle('Scatterplot of weight. vs. residuals')

after_log <- regr_wt_log %>% ggplot( aes(x = log.weight., y = regr_wt_log$residuals)) +
    geom_smooth(method = "lm", se = TRUE, colour="red", size=1)  +
    geom_point(colour = "red", size = 3, alpha = 0.4) +
    ggtitle('Scatterplot of log.weight. vs. residuals')

# Plot them together
grid.arrange(before_log, after_log,ncol = 2, nrow = 1)
```

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



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

```
# Before Log
sprintf("Before Log:")
```

```
## [1] "Before Log:"
```

```
shapiro.test(regr_wt$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  regr_wt$residuals
## W = 0.96938, p-value = 2.525e-07
```

```
# After Log
sprintf("After Log:" )
```

```
## [1] "After Log:"
```

```
shapiro.test(regr_wt_log$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  regr_wt_log$residuals
## W = 0.98983, p-value = 0.008087
```

The p-value of the regr_wt$residuals is far smaller than the one after log. Hence, after the log transform, the regression produces better distributed residuals for the assumptions of regression.

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

```
regr_wt_log$coefficients
```

```
## (Intercept) log.weight.
##   11.515197   -1.057506
```

- Ans:

1 percentage change in weight will leads to -1.057506% change in weight mpg.

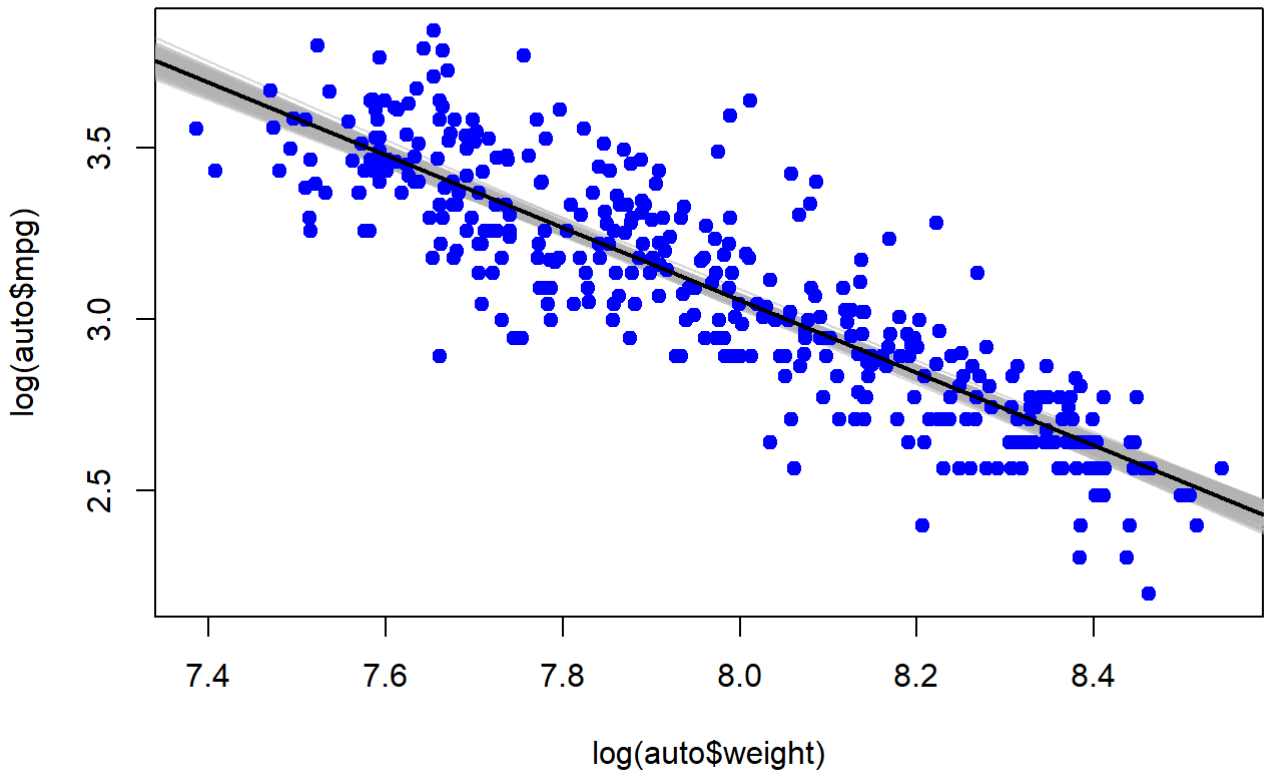
```
regr_wt_log$coefficients
```

```
## (Intercept) log.weight.
##   11.515197   -1.057506
```

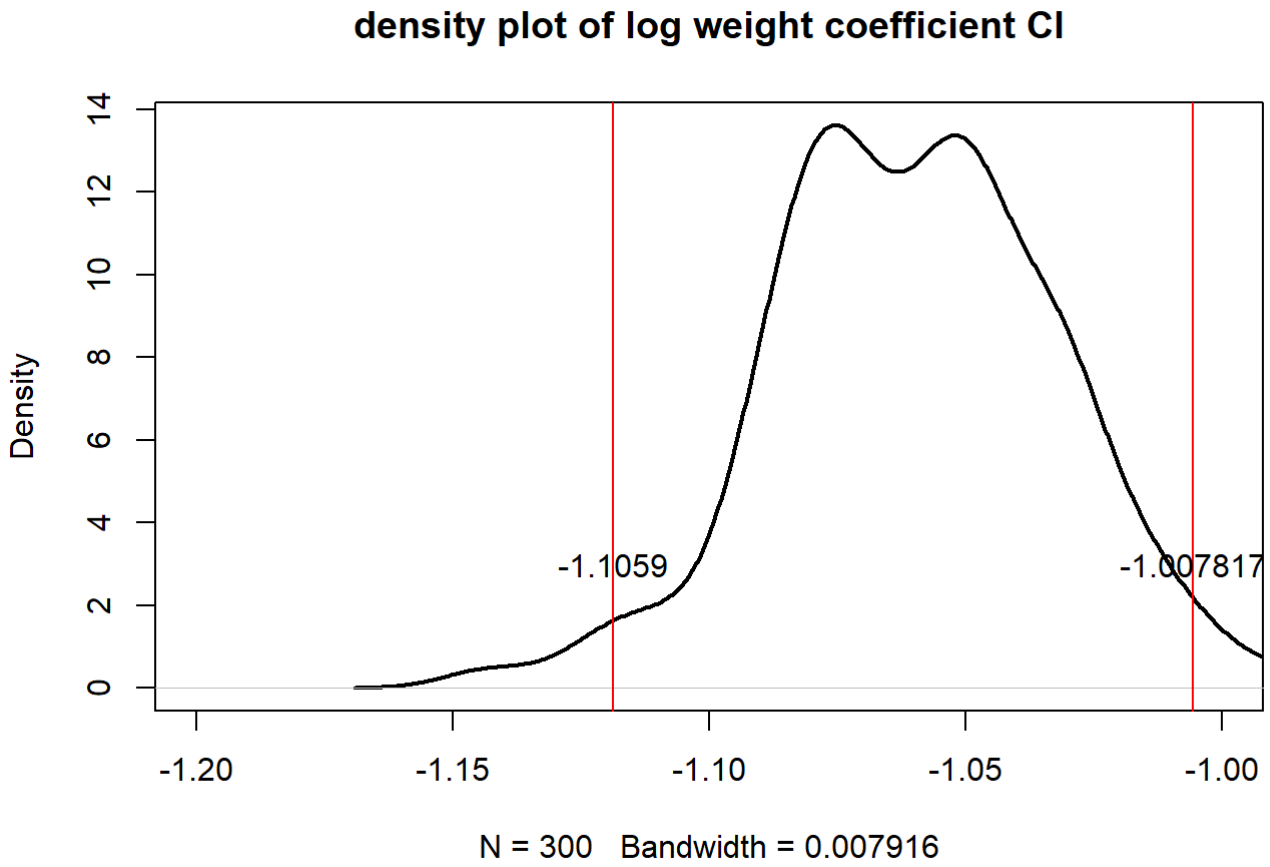
c. Let's examine the 95% confidence interval of the slope of log.weight. vs. log.mpg.

ci. Create a bootstrapped confidence interval

```
plot(log(auto$weight), log(auto$mpg), col=NA, pch=19)

# Function for single resampled 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
}

coeffs<-replicate(300,boot_regr(log(mpg) ~ log(weight), auto))

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




```
#Confidence interval values
conf_int <- quantile(coeffs["log(weight)",], c(0.025, 0.975))

#Plot confidence interval of coefficient
plot(density(coeffs["log(weight)",]),xlim=c(-1.2, -1),
     main='density plot of log weight coefficient CI', lwd=2)
abline(v=quantile(coeffs["log(weight)",], c(0.025, 0.975)), col='red')
text(conf_int[1],3, "-1.1059")
text(conf_int[2],3, "-1.007817")
```



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

(i.e., estimate of coefficient and its standard error from lm() results)

```
hp_regr_log<-lm(log(mpg) ~ log(weight), auto)
confint(hp_regr_log, ,level=.95)
```

```
##                 2.5 %     97.5 %
## (Intercept) 11.050180 11.9802136
## log(weight) -1.115895 -0.9991175
```

```
print('---------------------')
```

```
## [1] "---------------------"
```

```
conf_int
```

```
##      2.5%     97.5%
## -1.118685 -1.005700
```

They are pretty close.

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 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.acce
leration. +model_year+factor(origin),data=cars_log,na.action=na.exclude)

r2_weight <-summary(weight_regr)$r.squared
vif_weight<-1 / (1-r2_weight)
sqrt(vif_weight)
```

```
## [1] 4.192269
```

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!)

bi. 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
```

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

Eliminate log.displacement.

```
bii = subset(cars_log, select = -c(log.displacement.) )
regr_log_bii <- lm(log.mpg. ~ log.cylinders.+ log.horsepower. +
                              log.weight. + log.acceleration. + model_year +
                              factor(origin), data=bii)

vif(regr_log_bii)
```

```
##                       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
```

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

```
# remove Horse Power
bii = subset(cars_log, select = -c(log.horsepower.) )
regr_log_bii2 <- lm(log.mpg. ~ log.cylinders.+ log.weight. + log.acceleration. + model_year +
factor(origin), data=bii,na.action=na.omit)

# remove log.cylinders
bii = subset(cars_log, select = -c(log.cylinders.) )
regr_log_bii3 <- lm(log.mpg. ~log.weight. + log.acceleration. + model_year + factor(origin),
 data=bii,na.action=na.omit)


vif(regr_log_bii3)
```

```
##                       GVIF Df GVIF^(1/(2*Df))
## log.weight.       1.933208  1        1.390398
## log.acceleration. 1.304761  1        1.142261
## model_year        1.175545  1        1.084225
## factor(origin)    1.710178  2        1.143564
```

biv. Report the final regression model and its summary statistics

```
regr_log_bii3
```

```
##
## Call:
## lm(formula = log.mpg. ~ log.weight. + log.acceleration. + model_year +
##     factor(origin), data = bii, na.action = na.omit)
##
## Coefficients:
##       (Intercept)        log.weight.  log.acceleration.         model_year
##           7.41097           -0.87550            0.05438            0.03279
##   factor(origin)2    factor(origin)3
##           0.05611            0.03194
```

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. Ex. We lost horsepower.

```
origi_r <- summary(regr_log )$r.squared
After_dropping_r <- summary(regr_log_bii3)$r.squared
ac <- abs(After_dropping_r- origi_r)*100
sprintf("It decrease our explanation approximately %.3f percentage",ac)
```

```
## [1] "It decrease our explanation approximately 0.742 percentage"
```

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

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

- Ans:

VIF = 1. Because vif = 1/ (1-r.square). If independent variable has no correlation, r square equal to zero. Then vif =1.

dii. 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:

As r square >= 0.8, the VIF >=5

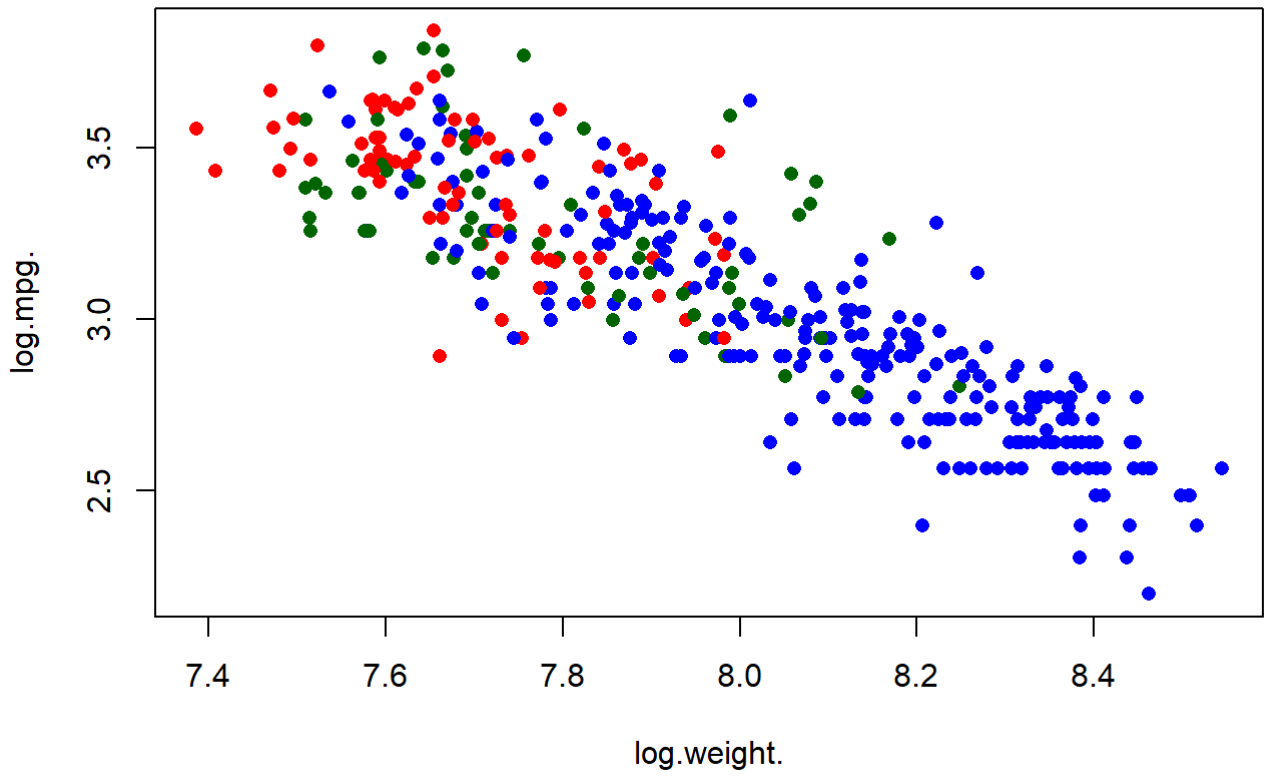
As r square >= 0.9, the VIF >=10

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: Here's one for the US to get you started:

```
origin_colors = c("blue", "darkgreen", "red")
# plot(cars_log$log.weight., cars_log$log.mpg., pch= 16, col=origin_colors[origin])
with(cars_log, plot(log.weight., log.mpg., pch=16, col=origin_colors[origin]))
```



b. [not graded] Do cars from different origins appear to have different weight vs. mpg relationships? We will investigate these relationships more in class!

Ans: Note that Origin of car (1. American, 2. European, 3. Japanese).

```
ggplot(auto, aes(x = weight, y = mpg)) + geom_point(aes(color = origin))+
    geom_smooth(method="lm", mapping=aes(x=weight,y= mpg ,color = origin), se=FALSE)
```

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

- *Ans:*

Yes.

The slope of American car (No.1) and European's car are pretty close. However, in the scatter above, we can tell that the slope of Japanese car is steeper than the former. It indicates that each unit of weight put on Japanese's car make the increasing of fuel consumption lesser than the American's car and EU's car.