BACS HW11 - 109006234

Credit: 109006278

April 30th 2023

Supporting Functions

```r
# Function for Bootstrapping
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
}
```

Problem 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
cars <- read.table("G:/My Drive/111_2_BACS/HW11/auto-data.txt",
    header=FALSE, na.strings = "?")
names(cars) <- c("mpg", "cylinders", "displacement",
    "horsepower", "weight", "acceleration",
    "model_year", "origin", "car_name")
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

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

```
##
## Call:
## lm(formula = log.mpg. ~ log.cylinders. + log.displacement. +
##     log.horsepower. + log.weight. + log.acceleration. + model_year +
##     factor(origin), data = cars_log, na.action = na.exclude)
##
## 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?
`horsepower`, `weight`, `acceleration`, `model_year`, amd `origin` has less than 10 percent significance.

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

```
org_regr <- lm(mpg ~ cylinders + displacement +
            horsepower + weight + acceleration +
            model_year + factor(origin), data = cars)
summary(org_regr)
```

```
##
## 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
```

From the observation above, we can see that `horsepower` and `acceleration` become significant to `mpg` after the log-transform regression. On the other hand, `displacement` becomes not significant affecting.

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

```r
cars <- cars[-9]
cor(cars$mpg, cars)
```

```
##      mpg  cylinders displacement horsepower     weight acceleration model_year
## [1,]   1 -0.7753963   -0.8042028         NA -0.8317409    0.4202889  0.5792671
##         origin
## [1,] 0.5634504
```

```r
cor(cars_log$log.mpg., cars_log)
```

```
##      log.mpg. log.cylinders. log.displacement. log.horsepower. log.weight.
## [1,]        1     -0.8204483         -0.860579              NA  -0.8744686
##      log.acceleration. model_year    origin
## [1,]         0.4640533  0.5763423 0.5583293
```

`displacement` will still be insignificant and because `cylinders` has high correlation with others (ex. `displacement`, `horsepower`, `weight`), there is no need to use variable cylinders in our regression function.

(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

```r
regr_wt <- lm(mpg ~ weight, data=cars, na.action=na.exclude)
regr_wt
```

```
##
## Call:
## lm(formula = mpg ~ weight, data = cars, na.action = na.exclude)
##
## Coefficients:
## (Intercept)       weight
##   46.317364    -0.007677
```

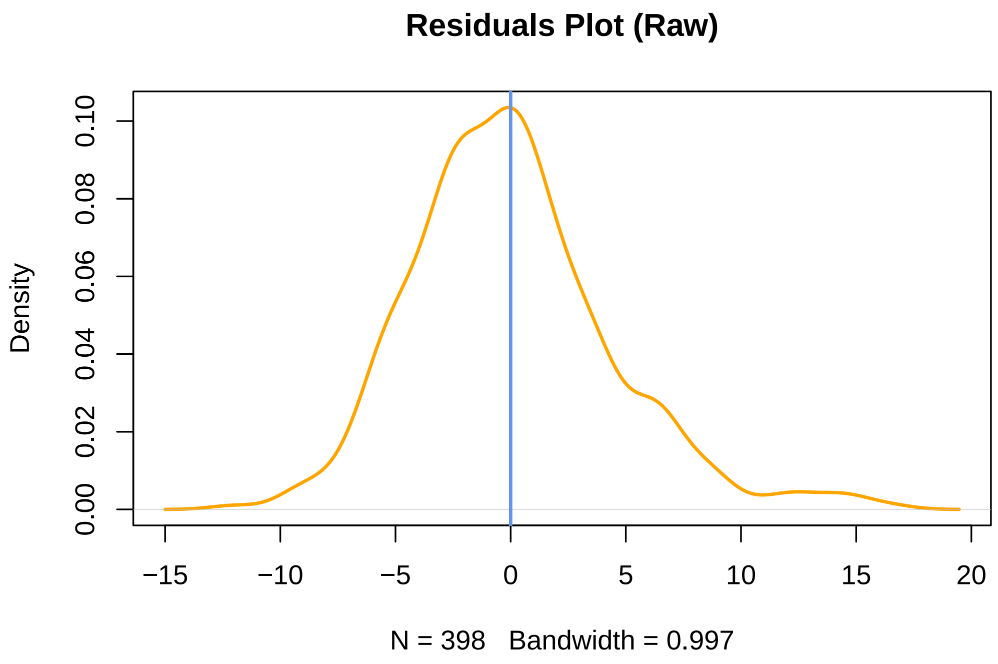
(ii) Create a regression (call it regr_wt_log) of log.mpg. on log.weight. from cars_log

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

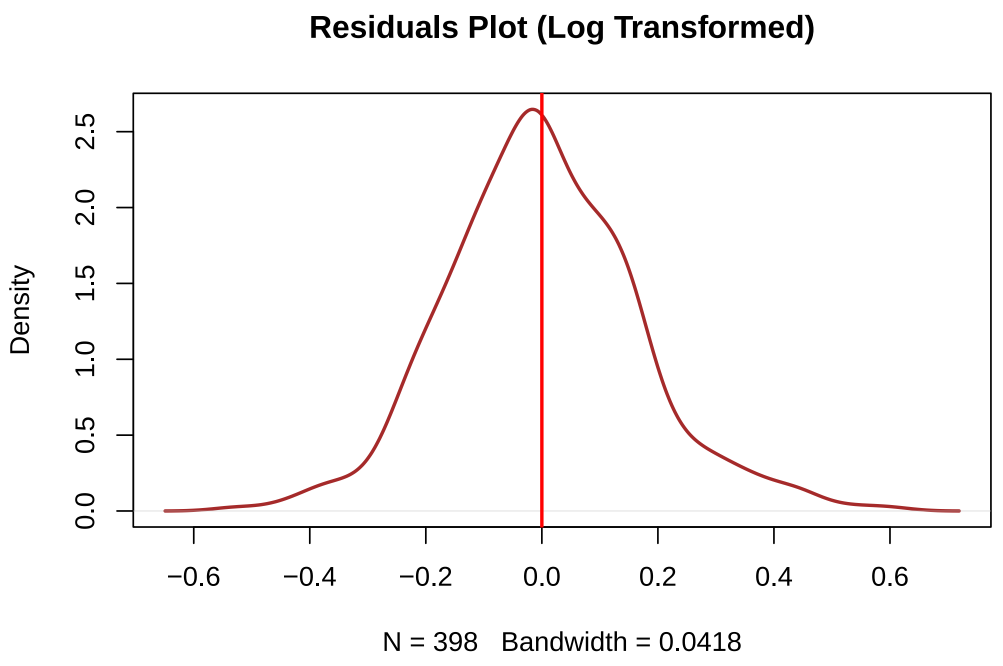
```
##
## Call:
## lm(formula = log.mpg. ~ log.weight., data = cars_log, na.action = na.exclude)
##
## Coefficients:
## (Intercept)  log.weight.
##      11.522       -1.058
```

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

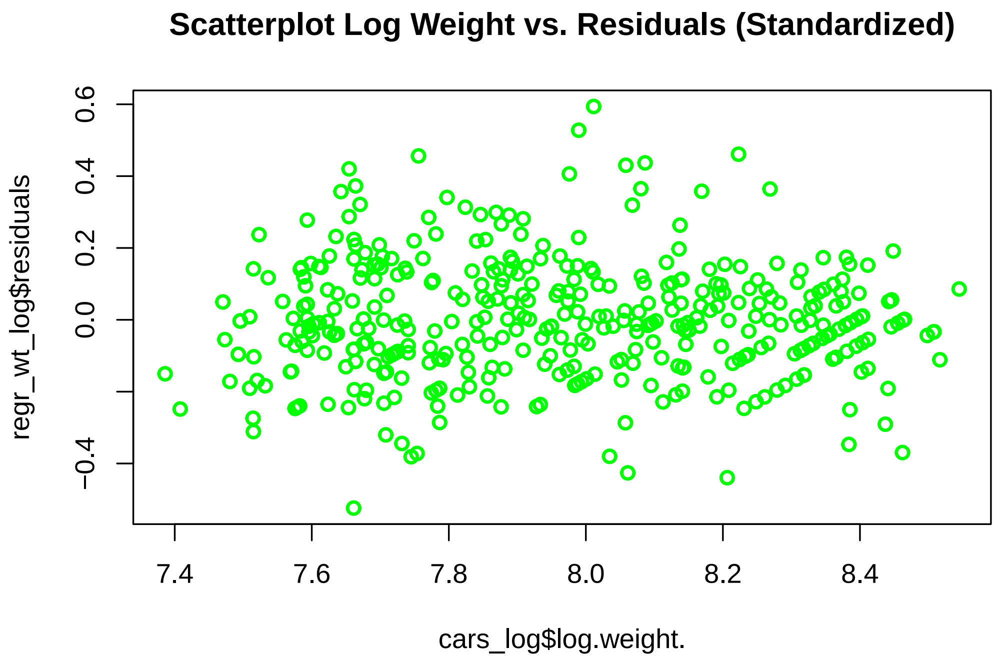
```r
plot(density(regr_wt$residuals),
     col="orange", lwd=2,
     main="Residuals Plot (Raw)")
abline(v=mean(regr_wt$residuals), col="cornflowerblue", lwd=2)
```



```r
plot(density(regr_wt_log$residuals),
     col="brown", lwd=2,
     main="Residuals Plot (Log Transformed)")
abline(v=mean(regr_wt_log$residuals), col="red", lwd=2)
```

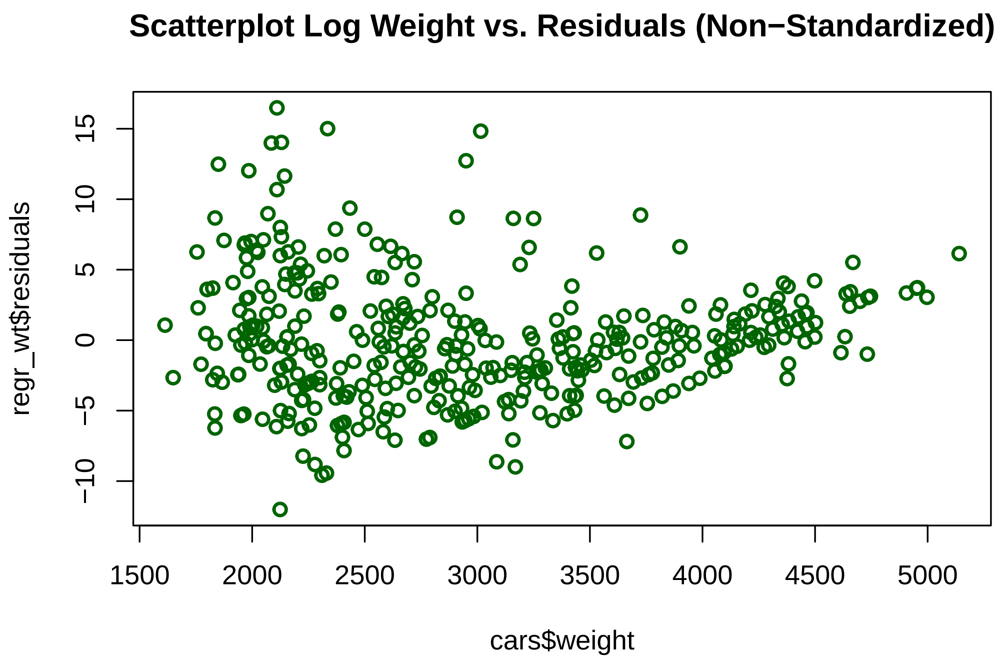


```
plot(cars_log$log.weight., regr_wt_log$residuals,
     col="green", lwd=2,
     main="Scatterplot Log Weight vs. Residuals (Standardized)")
```



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

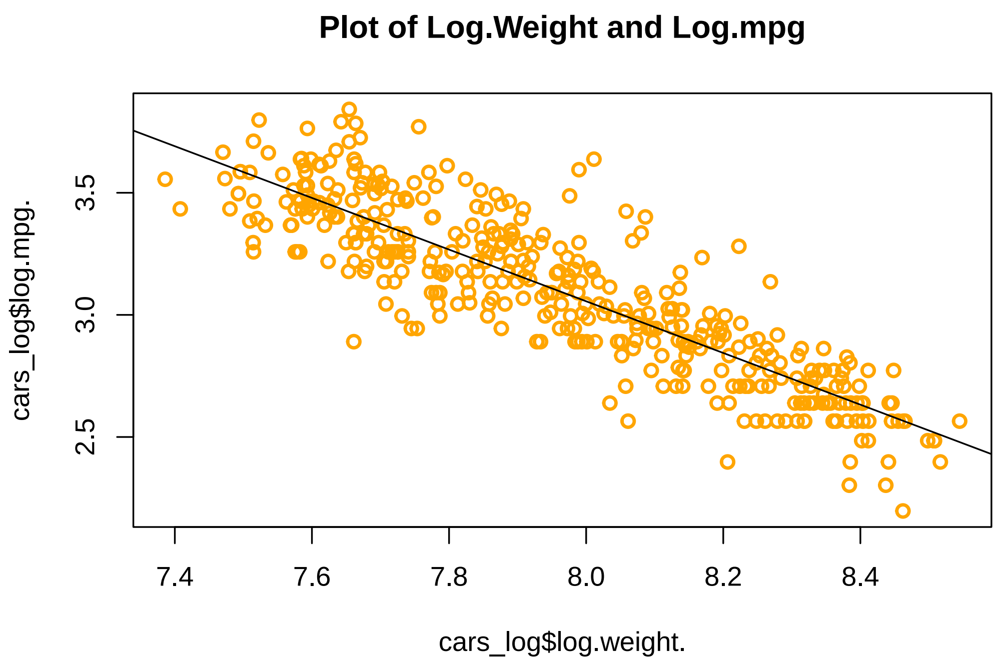
```
plot(cars$weight, regr_wt$residuals,
     col="darkgreen", lwd=2,
     main="Scatterplot Log Weight vs. Residuals (Non-Standardized)")
```



The residuals of standardized plot is better as the data is centralized in the middle.

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

```
plot(cars_log$log.weight.,cars_log$log.mpg.,
     col="orange", lwd=2,
    main="Plot of Log.Weight and Log.mpg")
abline(regr_wt_log)
```



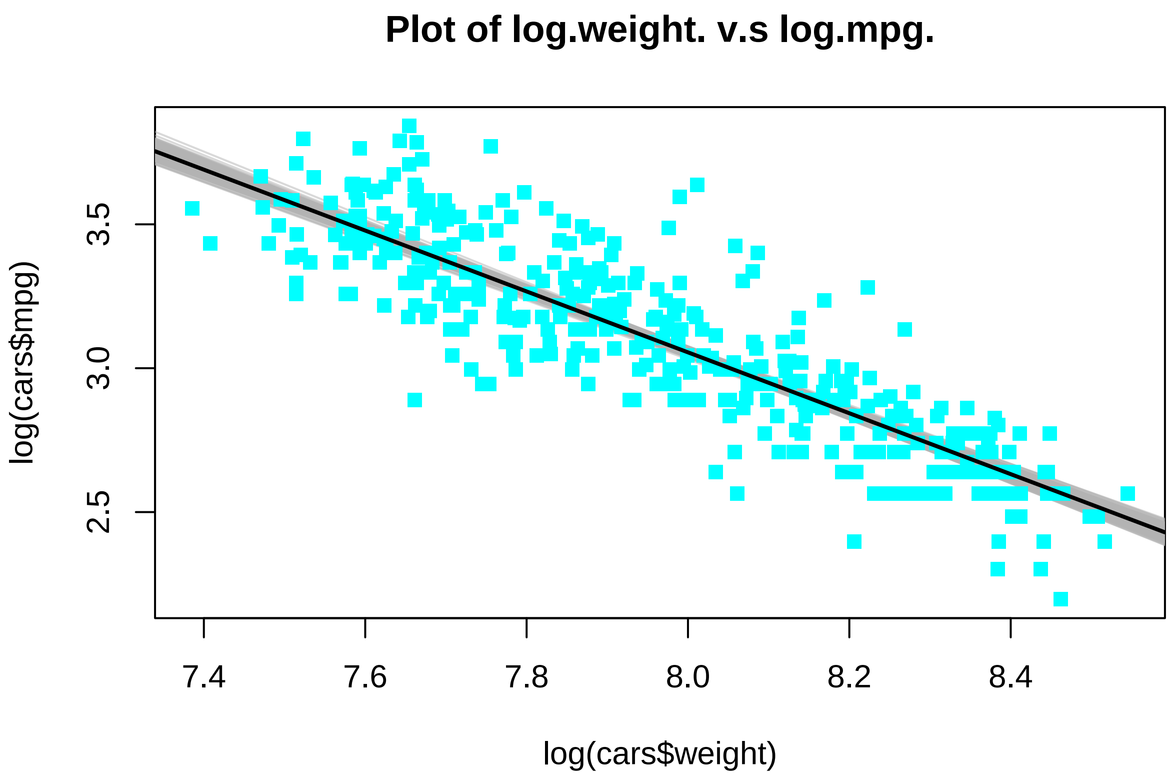
The slope is a negative slope and we can see that 1% change in weight, which leads to -1.058% change in MPG.

(vi) From its standard error, what is the 95% confidence interval of the slope of log.weight. vs log.mpg.?

```
plot(log(cars$weight), log(cars$mpg),
    col=NA, pch=15,
    main="Plot of log.weight. v.s log.mpg.")

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

points(log(cars$weight), log(cars$mpg), col="cyan", pch=15)
abline(a=mean(coeffs["(Intercept)",]),
    b=mean(coeffs["log(weight)",]), lwd=2)
```



```
quantile(coeffs["log(weight)",], c(0.025, 0.975))
```

```
##      2.5%     97.5%
## -1.112479 -1.008793
```

Problem 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.acceleration. + model_year + factor(origin),
        data = cars_log, na.action = na.exclude)
r2 <- summary(weight_regr)$r.squared
vif <- sqrt(1 / (1 - r2))
```

```
## [1] "VIF:  4.19226874712011"
```

(b) Let's try a procedure called Stepwise VIF Selection to remove highly collinear predictors.
(i) Use vif(regr_log) to compute VIF of the all the independent variables

```r
regr_log_vif <- vif(regr_log)
regr_log_vif
```

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

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

From result above, we remove displacement

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

```
##                          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
```

From result above, we remove horsepower

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

```
##                       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
```

From result above, we remove cylinder

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

```
##                       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

```
regr_log
```

```
##
## Call:
## lm(formula = log.mpg. ~ log.weight. + log.acceleration. + model_year +
##     factor(origin), data = cars_log)
##
## Coefficients:
##       (Intercept)        log.weight.  log.acceleration.         model_year
##           7.43116           -0.87661            0.05151            0.03273
##   factor(origin)2    factor(origin)3
##           0.05799            0.03233
```

```
summary(regr_log)
```

```
##
## 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?
`horsepower` is removed, in which it was significant before.

(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?
VIF = 1 due to r^2 being 0
(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?

```r
vif<-5
rsquared<-sqrt(1-(1/vif))
rsquared
```

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

If we want to get the VIF scores to be 5 or higher: r^2 > 0.8.

```r
vif<-10
rsquared<-sqrt(1-(1/vif))
rsquared
```

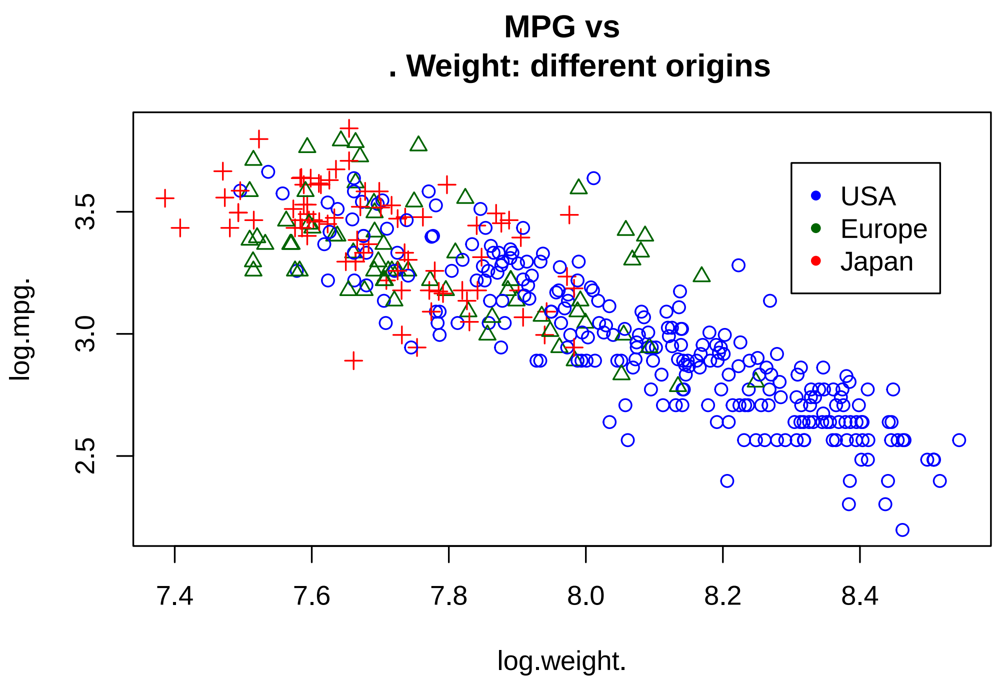
```
## [1] 0.9486833
```

Meanwhile, if we want to get the VIF to be 10 or higher: r^2 > 0.9.

Problem 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:

```r
origin_colors = c("blue", "darkgreen", "red")
with(cars_log,
    plot(log.weight., log.mpg., pch=origin,
    main = "MPG vs
    . Weight: different origins",
    col=origin_colors[origin]))
legend(8.3, 3.7, c("USA", "Europe", "Japan"),
    col = c("blue", "darkgreen", "red"),
    pch = c(20,20,20))
```



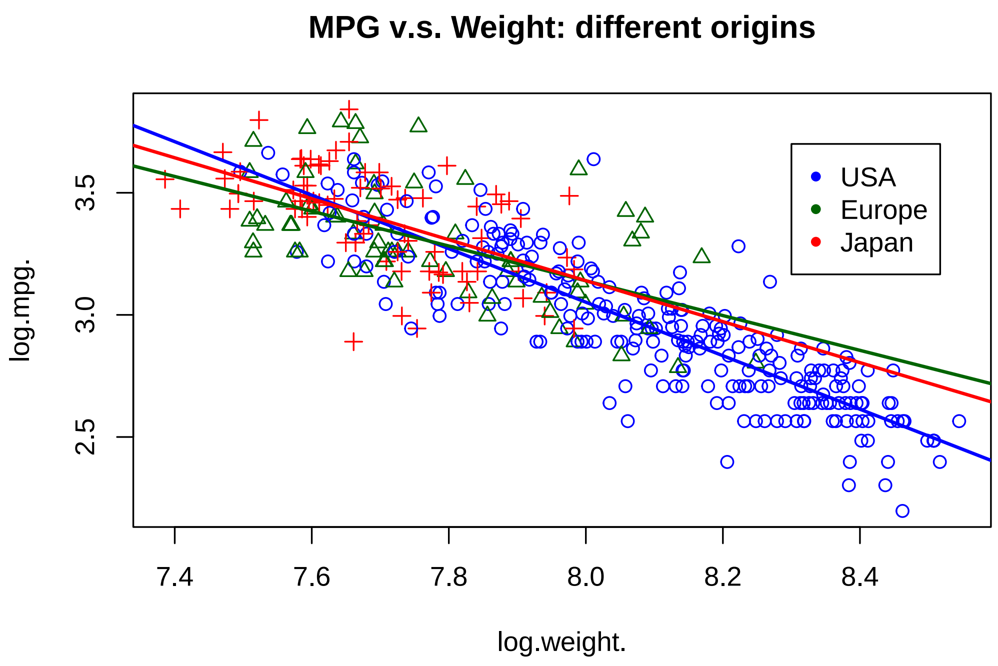
(a) Let's add three separate regression lines on the scatterplot, one for each of the origins.

```
origin_colors = c("blue", "darkgreen", "red")
with(cars_log,
    plot(log.weight., log.mpg., pch=origin,
    main = "MPG v.s. Weight: different origins",
    col=origin_colors[origin]))
legend(8.3, 3.7, c("USA", "Europe", "Japan"),
    col = c("blue", "darkgreen", "red"),
    pch = c(20,20,20))

USA <- subset(cars_log, origin==1)
reg_USA <- lm(log.mpg. ~ log.weight., data=USA)
abline(reg_USA, col=origin_colors[1], lwd=2)

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

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



(b) Do cars from different origins appear to have different weight vs. mpg relationships?
For all cars produced in USA, Europe and Japan, we can see that if the cars are lighter in weight, they are able to carry more fuel, hence the further the distance can be travelled.