BACS HW (Week12)

106070038

2021-05-16

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):

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

```
# install.packages("logr")
library(logr)

cars <- read.table("auto-data.txt",)
names(cars) <- c("mpg", "cylinders", "displacement", "horsepower", "weight", "acceleration",
                 "model_year","origin", "car_name")
cars$horsepower <- as.numeric(cars$horsepower)
```

```
## Warning: NAs introduced by coercion
```

```
cars <- na.omit(cars)
head(cars, 6)
```

```
##   mpg cylinders displacement horsepower weight acceleration model_year origin
## 1  18         8          307        130   3504         12.0         70      1
## 2  15         8          350        165   3693         11.5         70      1
## 3  18         8          318        150   3436         11.0         70      1
## 4  16         8          304        150   3433         12.0         70      1
## 5  17         8          302        140   3449         10.5         70      1
## 6  15         8          429        198   4341         10.0         70      1
##                    car_name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

```
cars_log <- with(cars, data.frame(log(mpg), log(cylinders), log(displacement),
                                  log(horsepower), log(weight), log(acceleration), model_year, origin))
regr <- lm(log.mpg. ~ ., data = cars_log)
summary(regr)
```

```
##
```

```
## Call:
## lm(formula = log.mpg. ~ ., data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.41449 -0.06967  0.00040  0.06035  0.39298
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         7.252158   0.363468  19.953  < 2e-16 ***
## log.cylinders.     -0.074879   0.061060  -1.226  0.22083
## log.displacement. -0.008015   0.055532  -0.144  0.88532
## log.horsepower.   -0.296585   0.057548  -5.154 4.09e-07 ***
## log.weight.       -0.554906   0.081716  -6.791 4.26e-11 ***
## log.acceleration. -0.182062   0.059222  -3.074  0.00226 **
## model_year         0.029608   0.001726  17.149  < 2e-16 ***
## origin             0.022419   0.010301   2.176  0.03014 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1132 on 384 degrees of freedom
## Multiple R-squared:  0.8912, Adjusted R-squared:  0.8892
## F-statistic: 449.5 on 7 and 384 DF,  p-value: < 2.2e-16
```

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

- Answer:
  - **< 10% significance**: log.horsepower. , log.weight. , log.acceleration. , model_year , origin

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

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

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

```
##     Min      1Q  Median      3Q     Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.218435   4.644294  -3.707  0.00024 ***
## cylinders     -0.493376   0.323282  -1.526  0.12780
## displacement   0.019896   0.007515   2.647  0.00844 **
## horsepower    -0.016951   0.013787  -1.230  0.21963
## weight        -0.006474   0.000652  -9.929  < 2e-16 ***
## acceleration   0.080576   0.098845   0.815  0.41548
## model_year     0.750773   0.050973  14.729  < 2e-16 ***
## origin         1.426141   0.278136   5.127 4.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.328 on 384 degrees of freedom
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

- Answer: Comparing raw regression and log-transformed regression, after log-transformed, **horsepower and acceleration** become significant affecting mpg. However, in contrast, **displacement** becomes not significant affecting.

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

```
cars <- cars[-9] ## drop car_name
cor(cars)
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## model_year    0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
##              acceleration model_year     origin
## mpg             0.4233285  0.5805410  0.5652088
## cylinders      -0.5046834 -0.3456474 -0.5689316
## displacement   -0.5438005 -0.3698552 -0.6145351
## horsepower     -0.6891955 -0.4163615 -0.4551715
## weight         -0.4168392 -0.3091199 -0.5850054
```

```
## acceleration    1.0000000  0.2903161  0.2127458
## model_year      0.2903161  1.0000000  0.1815277
## origin          0.2127458  0.1815277  1.0000000
```

```r
## raw data
cor(cars$mpg, cars)
```

```
##      mpg  cylinders displacement horsepower     weight acceleration model_year
## [1,]   1 -0.7776175   -0.8051269 -0.7784268 -0.8322442    0.4233285   0.580541
##         origin
## [1,] 0.5652088
```

```r
## log-transformed data
cor(cars_log$log.mpg., cars_log)
```

```
##      log.mpg. log.cylinders. log.displacement. log.horsepower. log.weight.
## [1,]        1      -0.821506         -0.8600904      -0.8501157   -0.874511
##      log.acceleration. model_year    origin
## [1,]         0.4652735  0.5772748 0.5605076
```

- Answer:
  - still insignificant: **displacement**
  - opposite effects (cor < -0.7):
    - raw: cylinders, displacement, horsepower, weight
    - log-transformed: log.cylinders., log.displacement., log.horsepower., log.weight.
  - According to the correlation matrix, 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 on weight from the original cars dataset

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

```
##
## Call:
## lm(formula = mpg ~ weight, data = cars)
##
## Coefficients:
## (Intercept)       weight
##   46.216525    -0.007647
```

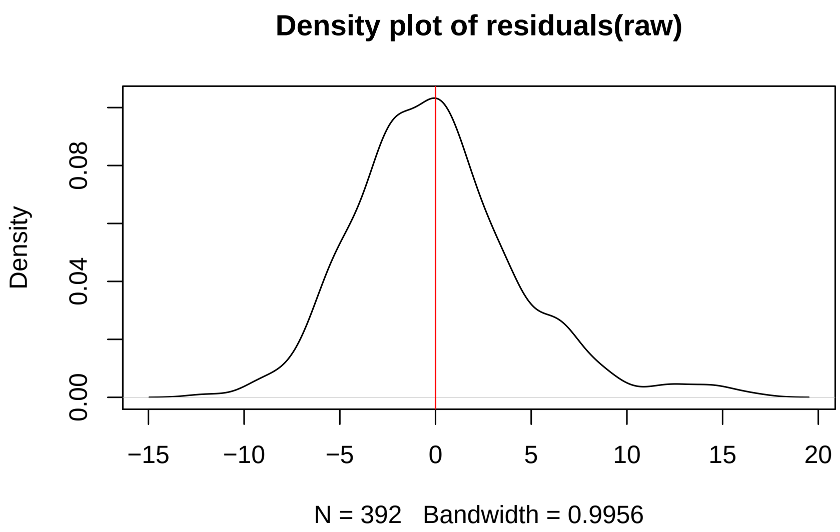
ii. Create a regression (call it regr_wt_log) of log.mpg. on log.weight. from cars_log

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

```
##
## Call:
## lm(formula = log.mpg. ~ log.weight., data = cars_log)
##
## Coefficients:
## (Intercept)  log.weight.
##      11.515       -1.058
```

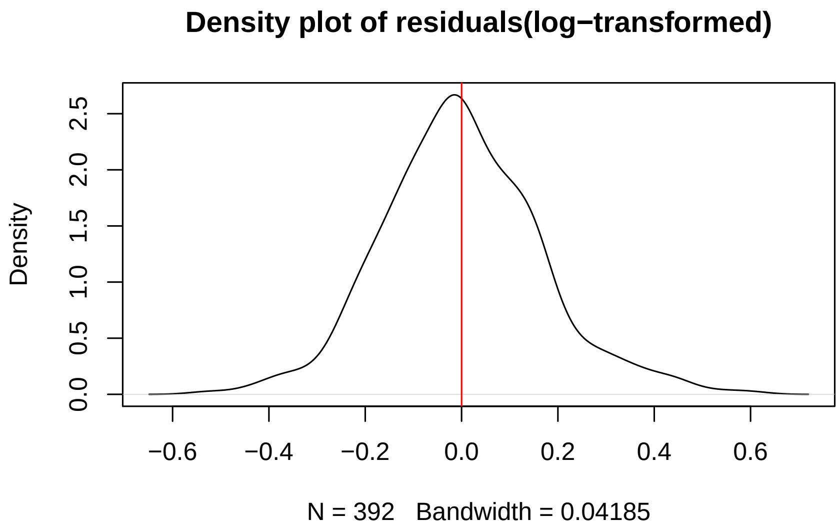
iii. Visualize the residuals of both regression models (raw and log-transformed):

```
## 1. density plots of residuals
plot(density(regr_wt$residuals), main="Density plot of residuals(raw)") + abline(v=mean(regr_wt$residual
```



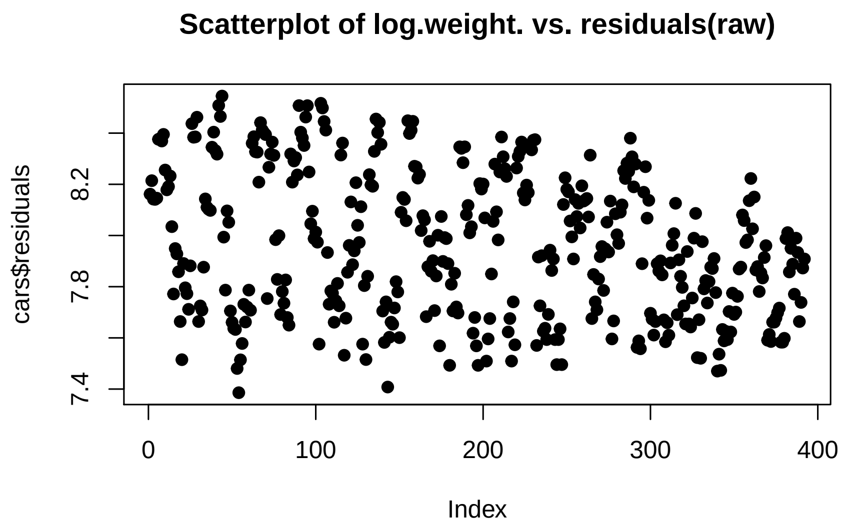

```
## integer(0)
```

```
plot(density(regr_wt_log$residuals),
     main="Density plot of residuals(log-transformed)") + abline(v=mean(regr_wt_log$residuals), col="re
```

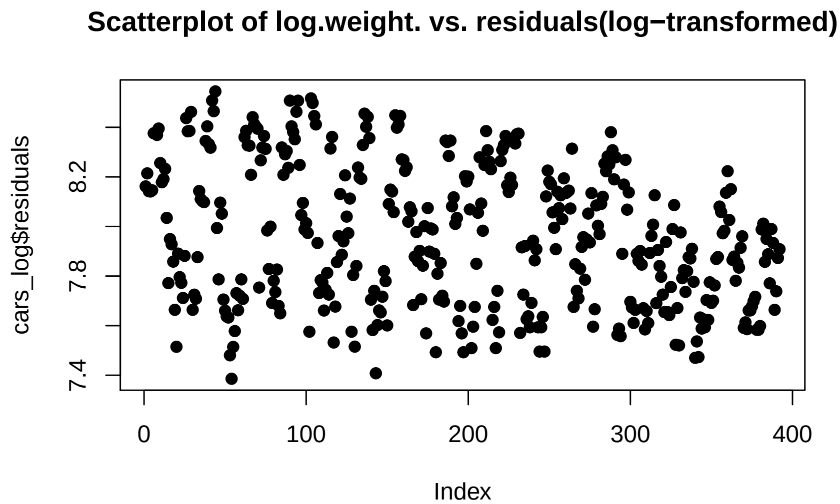



```
## integer(0)
```

```r
## 2. scatterplot of log.weight. vs. residuals
plot(cars_log$log.weight., cars$residuals, pch=19,
     main = "Scatterplot of log.weight. vs. residuals(raw)")
```




```r
plot(cars_log$log.weight., cars_log$residuals, pch=19,
     main = "Scatterplot of log.weight. vs. residuals(log-transformed)")
```



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

- Answer: The residuals of **Log—transformed** is better for the assumptions of regression. (randomly distributed around zero)

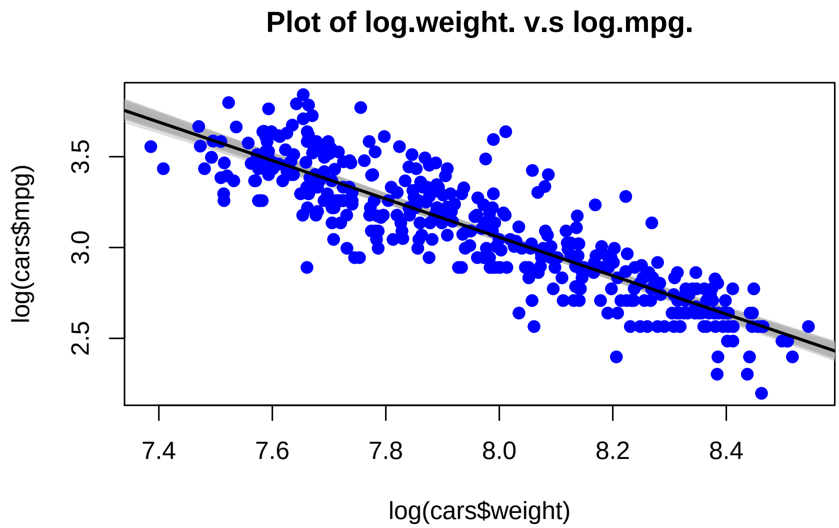
v. How would you interpret the slope of log.weight. vs log.mpg. in simple words?

- Answer: The slope of log.weight. vs log.mpg. is **-1.058**, which means it's **negative slope**.

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

i. Create a bootstrapped confidence interval

```
# Empty plot canvas
plot(log(cars$weight), log(cars$mpg), col=NA,
     pch=19, main="Plot of log.weight. v.s log.mpg.")
# 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$coefficients
}
# Bootstrapping for confidence interval
coeffs <- replicate(300, boot_regr(log(mpg) ~ log(weight), cars))
# Plot points and regression line
points(log(cars$weight), log(cars$mpg), col="blue", pch=19)
abline(a=mean(coeffs["(Intercept)",]),b=mean(coeffs["log(weight)",]), lwd=2)
```




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

```
##      2.5%     97.5%
## -1.110347 -1.001026
```

ii. 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), cars)
confint(hp_regr_log)
```

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

Question 2 Let's tackle multicollinearity next. Consider the regression model:

a. Using regression and R2, compute the VIF of log.weight. using the approach shown in class

```
regr_log <- lm(log.mpg. ~ log.cylinders. + log.displacement. + log.horsepower. +
                              log.weight. + log.acceleration. + model_year +
                              factor(origin),  data=cars_log)
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_weight <- summary(weight_regr)$r.squared
vif_weight <- 1 / (1 - r2_weight)
vif_weight
```

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

```r
sqrt(vif_weight)
```

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

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
# install.packages("car")
library(car)
```

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

```r
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
```

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

- Eliminate **log.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
```

- Eliminate **log.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.    5.427610  1        2.329723
## log.weight.       4.871730  1        2.207200
## log.acceleration. 1.401202  1        1.183724
## model_year        1.206351  1        1.098340
## factor(origin)    1.821167  2        1.161682
```

- Eliminate **log.cylinders.**

```r
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.       1.933208  1        1.390398
## log.acceleration. 1.304761  1        1.142261
## model_year        1.175545  1        1.084225
## factor(origin)    1.710178  2        1.143564
```

iv. Report the final regression model and its summary statistics

```r
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
##           7.41097           -0.87550            0.05438            0.03279
##   factor(origin)2    factor(origin)3
##           0.05611            0.03194
```

```r
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
## -0.38259 -0.07054  0.00401  0.06696  0.39798
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         7.410974   0.316806  23.393  < 2e-16 ***
## log.weight.        -0.875499   0.029086 -30.101  < 2e-16 ***
## log.acceleration.  0.054377   0.037132   1.464  0.14389
## model_year          0.032787   0.001731  18.937  < 2e-16 ***
## factor(origin)2     0.056111   0.018241   3.076  0.00225 **
## factor(origin)3     0.031937   0.018506   1.726  0.08519 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1163 on 386 degrees of freedom
## Multiple R-squared:  0.8845, Adjusted R-squared:  0.883
## F-statistic: 591.1 on 5 and 386 DF,  p-value: < 2.2e-16
```

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

- Answer: We lost **log.horsepower.** which was previously significant.

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: **VIF = 1, because r_squared = 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?

- Answer: To get VIF scores of 5 or higher - **r_squared > 0.8**
- To get VIF scores of 10 or higher - **r_squared > 0.9**

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

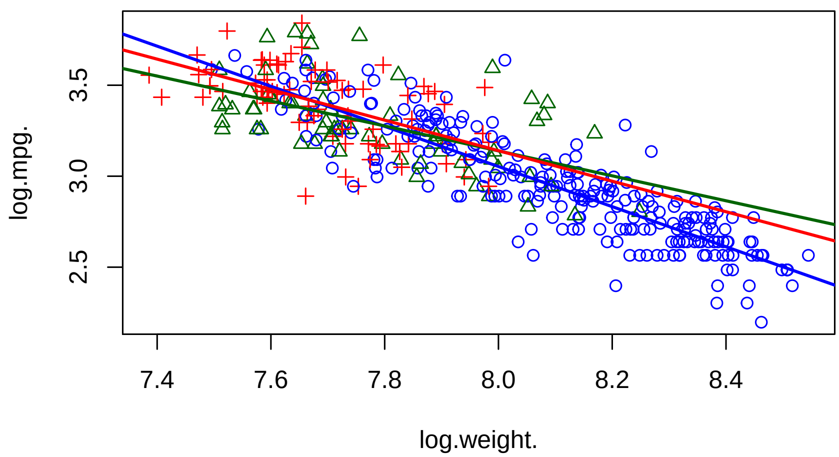
a. Let's add three separate regression lines on the scatterplot, one for each of the origins:

```
origin_colors = c("blue", "darkgreen", "red")
with(cars_log, plot(log.weight., log.mpg., pch=origin, 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_us <- subset(cars_log, origin==2)
wt_regr_us <- lm(log.mpg. ~ log.weight., data=cars_us)
abline(wt_regr_us, col=origin_colors[2], lwd=2)

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



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

- Answer: Yes