HW11

108048110

4/27/2022

BACS HW - Week 11

Prerequisite

```
library(car)
library(ggplot2)
library(corrplot)
library(tidyverse)
require(gridExtra)

theme_set(theme_bw(base_size=16))
```

```
auto <- read.table("auto-data.txt", header=FALSE, na.strings = "?")
names(auto) <- c("mpg", "cylinders", "displacement", "horsepower", "weight",
                 "acceleration", "model_year", "origin", "car_name")
auto = auto[complete.cases(auto),]

# return regression model
LMOfCars <- function(data){
  lm(mpg~cylinders+
           displacement+
           horsepower+
           weight+
           acceleration+
           auto$model_year+
           factor(origin),
           data = data,
           na.action = na.exclude)
}
```

Observations from full regression model

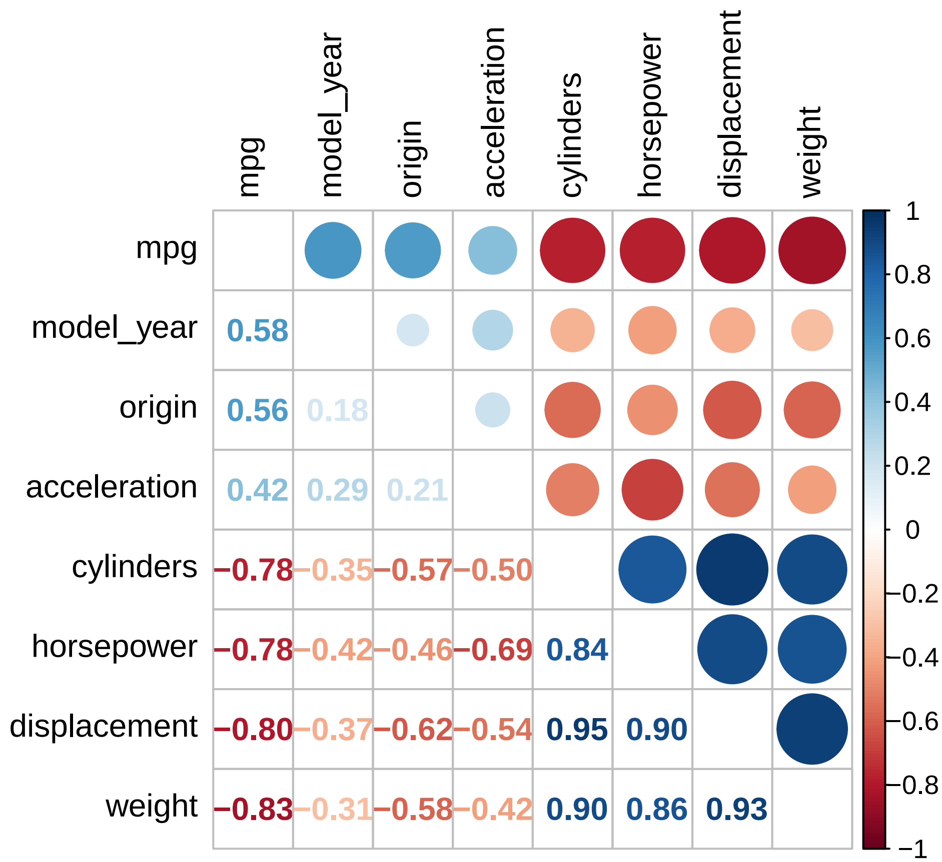
1. Only weight, year, and origin had significant effects.

```
summary(LMOfCars(auto))
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + auto$model_year + factor(origin), data = data,
##     na.action = na.exclude)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0095 -2.0785 -0.0982  1.9856 13.3608
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.795e+01  4.677e+00  -3.839 0.000145 ***
## cylinders       -4.897e-01  3.212e-01  -1.524 0.128215
## displacement     2.398e-02  7.653e-03   3.133 0.001863 **
## horsepower      -1.818e-02  1.371e-02  -1.326 0.185488
## weight          -6.710e-03  6.551e-04 -10.243  < 2e-16 ***
## acceleration     7.910e-02  9.822e-02   0.805 0.421101
## auto$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
## Multiple R-squared:  0.8242, Adjusted R-squared:  0.8205
## F-statistic: 224.5 on 8 and 383 DF,  p-value: < 2.2e-16
```

2. Non-significant factors were highly correlated with weight.

```
cor_plt <- function(data){
  cor_data <- round(cor(data[, 1:8], use='pairwise.complete.obs'), 3)
  cor_sorted_data <- names(sort(cor_data[, 'mpg'], decreasing = TRUE))
  cor_data <- cor_data[cor_sorted_data, cor_sorted_data]

  corrplot.mixed(cor_data, tl.col='black', tl.pos='lt')
}

cor_plt(auto)
```



3. Displacement has the opposite effect in the regression from its visualized effect.

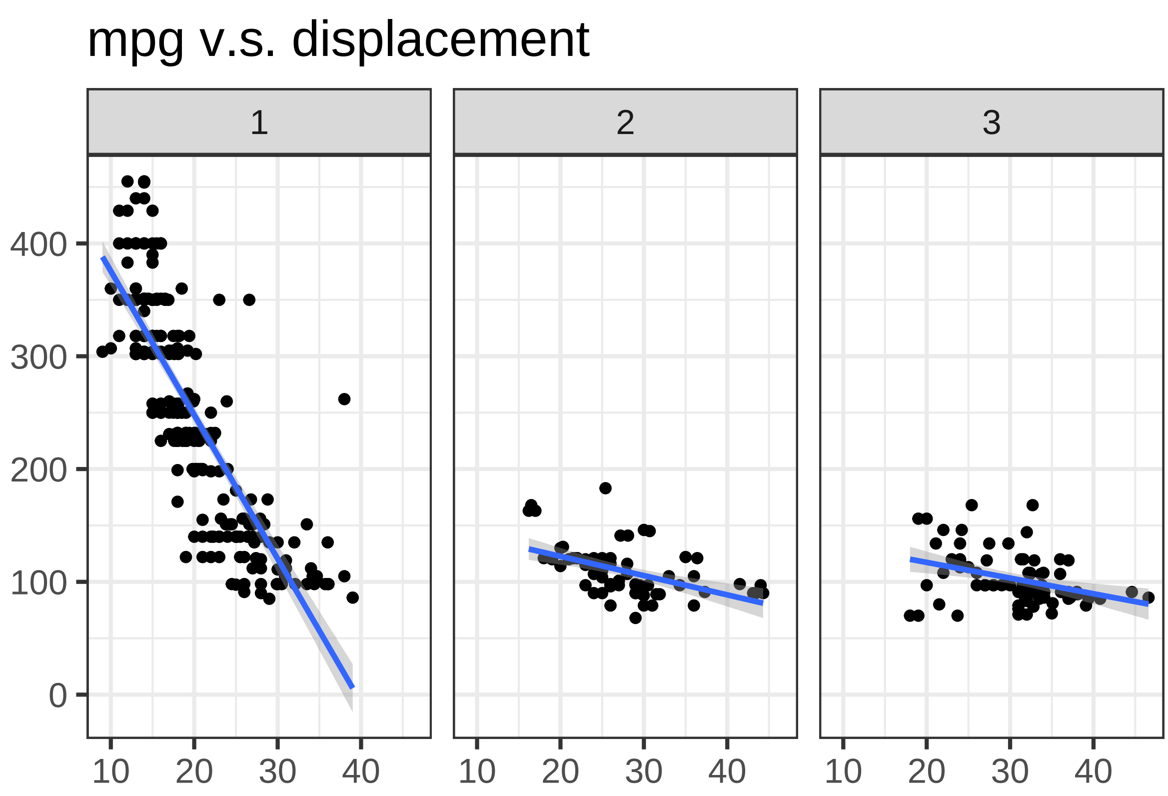
Note. Visualization is always right.

```
summary(LMOfCars(auto))
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + auto$model_year + factor(origin), data = data,
##     na.action = na.exclude)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0095 -2.0785 -0.0982  1.9856 13.3608
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.795e+01  4.677e+00  -3.839 0.000145 ***
## cylinders       -4.897e-01  3.212e-01  -1.524 0.128215
## displacement     2.398e-02  7.653e-03   3.133 0.001863 **
## horsepower      -1.818e-02  1.371e-02  -1.326 0.185488
## weight          -6.710e-03  6.551e-04 -10.243  < 2e-16 ***
## acceleration     7.910e-02  9.822e-02   0.805 0.421101
## auto$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
## Multiple R-squared:  0.8242, Adjusted R-squared:  0.8205
## F-statistic: 224.5 on 8 and 383 DF,  p-value: < 2.2e-16
```

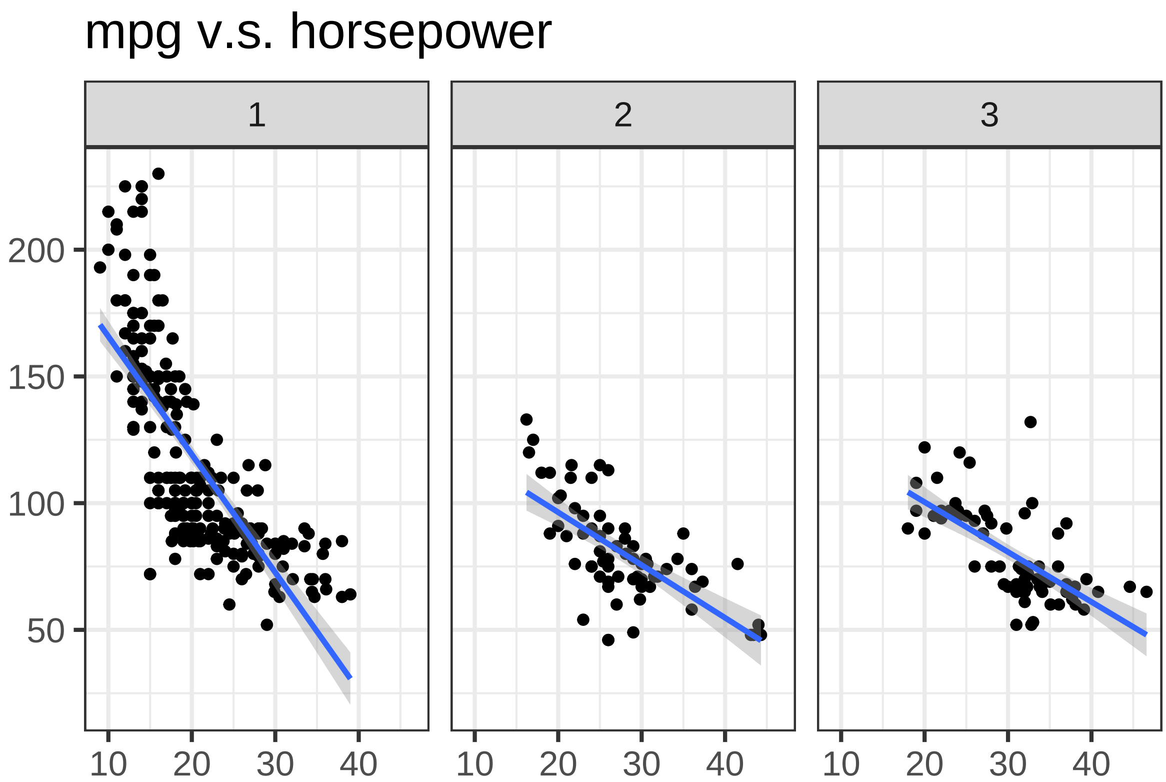
```r
plt <- function(a, b, title=''){
  ggplot(auto, aes(x=a, y=b))+
    geom_point()+
    facet_wrap(~factor(origin))+
    stat_smooth(method=lm)+
    labs(x='', y='')+
    ggtitle(title)
}

plt(auto$mpg, auto$displacement, 'mpg v.s. displacement')
```



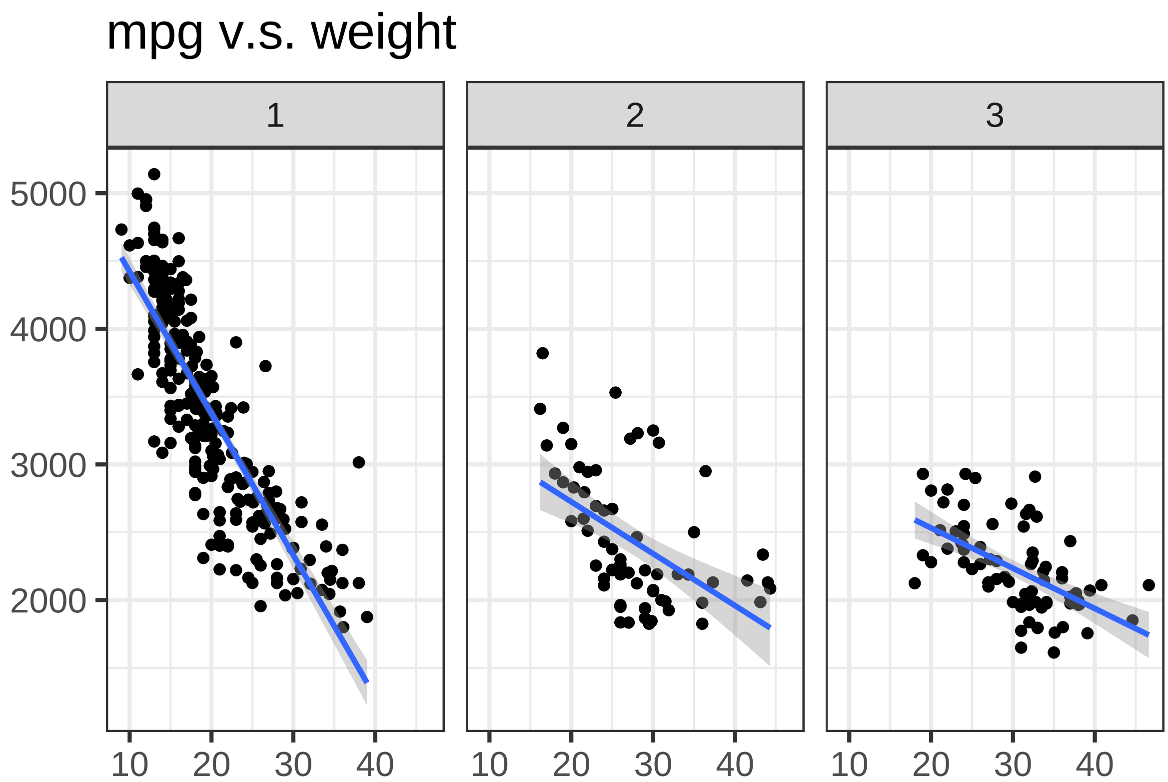
- Displacement in regression: -0.02398

4. Factors like horsepower and weight, seem to have a nonlinear relationship with mpg

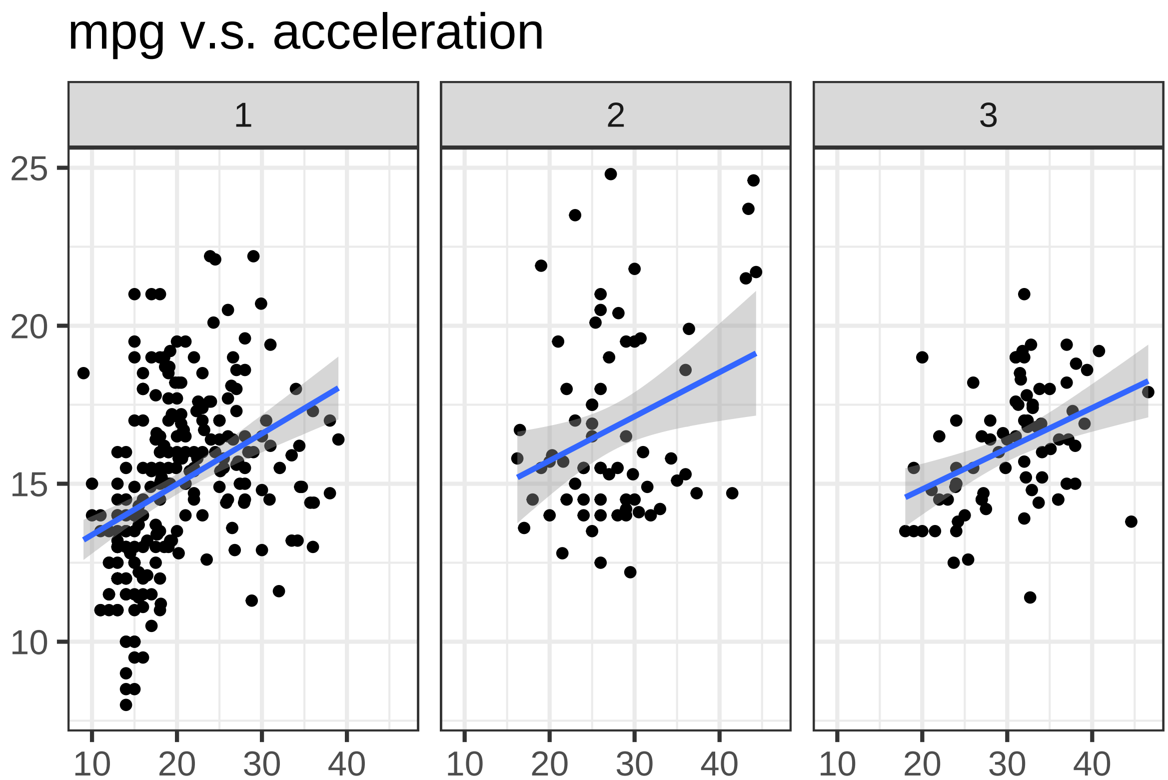
```
plt(auto$mpg, auto$horsepower, 'mpg v.s. horsepower')
```




```
plt(auto$mpg, auto$weight, 'mpg v.s. weight')
```




```
plt(auto$mpg, auto$acceleration, 'mpg v.s. acceleration')
```



Question 1) Nonlinearity

```
cars <- auto
cars_log <- with(cars, data.frame(log(mpg),
                                  log(cylinders),
                                  log(displacement),
                                  log(horsepower),
                                  log(weight),
                                  log(acceleration),
                                  model_year,
                                  origin))

names(cars_log) <- c("mpg", "cylinders", "displacement", "horsepower", "weight",
                 "acceleration", "model_year", "origin")

knitr::kable(head(cars_log))
```

mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
2.890372	2.079442	5.726848	4.867534	8.161660	2.484907	70	1
2.708050	2.079442	5.857933	5.105945	8.214194	2.442347	70	1

mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
2.890372	2.079442	5.762051	5.010635	8.142063	2.397895	70	1
2.772589	2.079442	5.717028	5.010635	8.141190	2.484907	70	1
2.833213	2.079442	5.710427	4.941642	8.145840	2.351375	70	1
2.708050	2.079442	6.061457	5.288267	8.375860	2.302585	70	1

a. Run a new regression on the `cars_log` dataset.

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

```
summary(LMOfCars(cars_log))
```

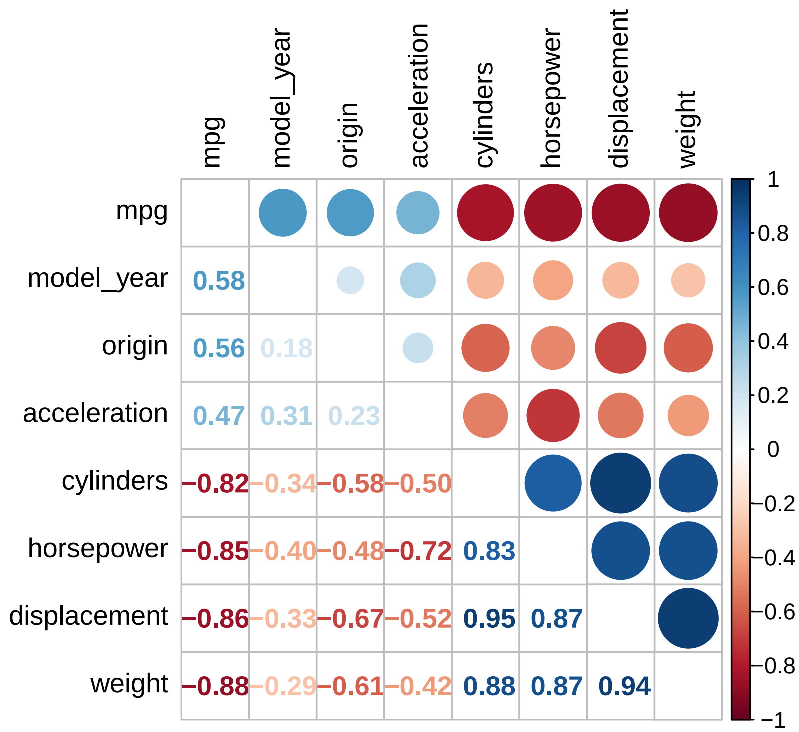
```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + auto$model_year + factor(origin), data = data,
##     na.action = na.exclude)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39727 -0.06880  0.00450  0.06356  0.38542
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       7.301938   0.361777  20.184  < 2e-16 ***
## cylinders        -0.081915   0.061116  -1.340  0.18094
## displacement      0.020387   0.058369   0.349  0.72707
## horsepower       -0.284751   0.057945  -4.914 1.32e-06 ***
## weight           -0.592955   0.085165  -6.962 1.46e-11 ***
## acceleration     -0.169673   0.059649  -2.845  0.00469 **
## auto$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
## Multiple R-squared:  0.8919,   Adjusted R-squared:  0.8897
## F-statistic:   395 on 8 and 383 DF,  p-value: < 2.2e-16
```

  - **Ans.** horsepower, weight, model_year

- ***ii.*** Do some new factors now have effects on `mpg`? Why?
  - **Ans.** Yes, new factors like horsepower, weight, have effects on mpg.
  - we are able to get linear relationships by applying log transformation on every values in the dataset.

- ***iii.*** Which factors still have insignificant effects on mpg?
  Why?

```
cor_plt(cars_log)
```




```
sort(cor(cars_log)[, 'mpg'], decreasing=TRUE)
```

```
##          mpg   model_year       origin acceleration    cylinders   horsepower
##    1.0000000    0.5772748    0.5605076    0.4652735   -0.8215060   -0.8501157
## displacement       weight
##   -0.8600904   -0.8745110
```

- **Ans.** Cylinders and displacement still have insignificant effects on mpg.
- This might be result from the nonlinearity present in the data.

b. Take a closer look at weight.

- *i.* Create a regression of `mpg` over `weight` from the original `cars` dataset.

```
regr_cars <- summary(LMOfCars(cars))
regr_wt <- lm(mpg~weight, data=cars, na.action=na.exclude)
regr_wt$coefficients
```

```
##  (Intercept)       weight
## 46.216524549 -0.007647343
```

- *ii.* Create a regression of `log(mpg)` on `log(weight)` from `cars_log`.

```
regr_cars_log <- summary(LMOfCars(cars_log))
regr_wt_log <- lm(mpg~weight, data=cars_log, na.action=na.exclude)
regr_wt_log$coefficients
```

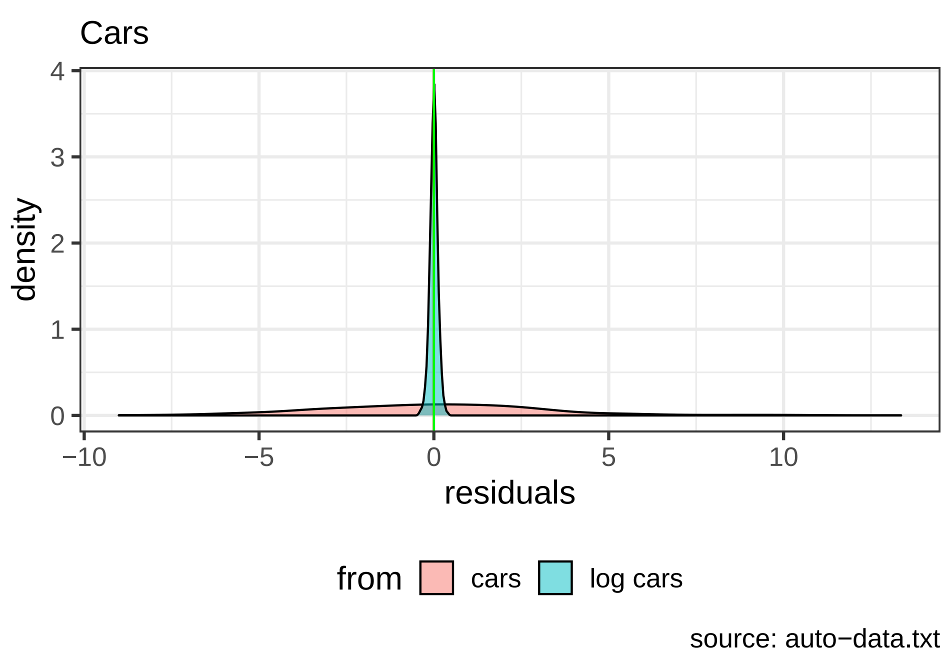
```
## (Intercept)      weight
##   11.515197   -1.057506
```

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

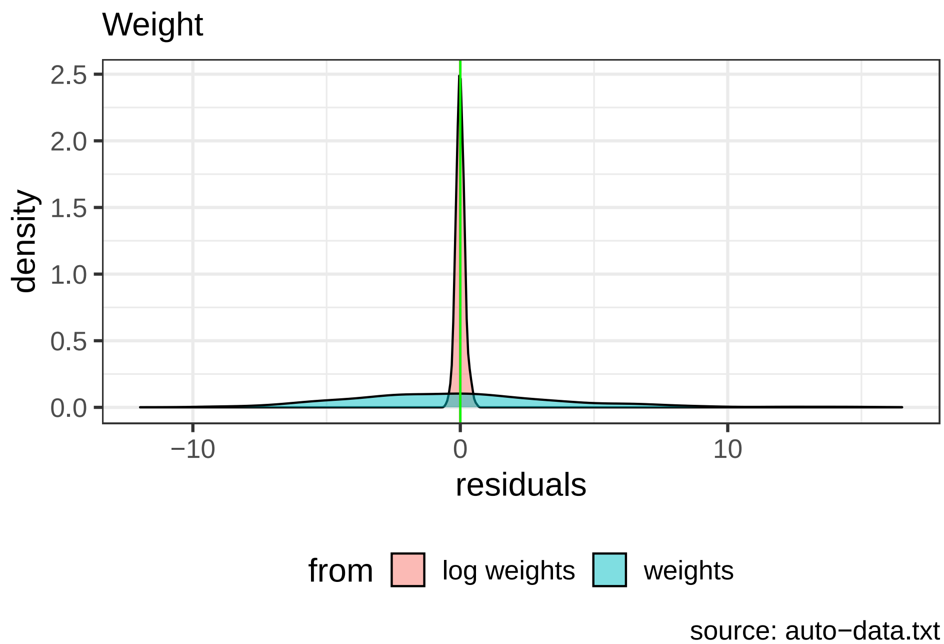
```
tibble_density_plot <- function(data1, data2, title='', t1='', t2=''){
  ToDF <- function(x, name=''){
    temp <- list()
    temp$residuals <- x$residuals
    temp$from <- rep(name, length(x$residuals))
    return(data.frame(temp))
  }

  d1 <- ToDF(data1, t1)
  d2 <- ToDF(data2, t2)
  temp <- as.tibble(rbind(d1, d2))

  temp %>%
    ggplot(aes(x=residuals, fill=from))+
    geom_density(alpha=0.5)+
    geom_vline(xintercept = c(mean(d1$residuals),
                              mean(d2$residuals)),
               col=c('red', 'green'))+
    labs(x='residuals',
         subtitle=title,
         caption='source: auto-data.txt')+
    theme(legend.position = 'bottom')
}
```

– 1. Density plots of residuals.

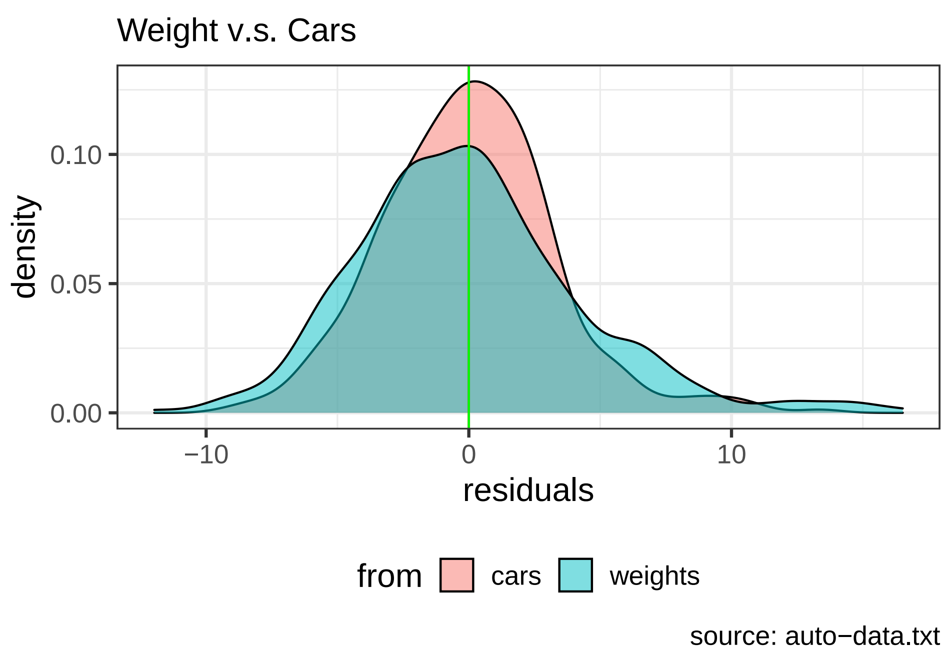
```
tibble_density_plot(regr_cars,
                    regr_cars_log,
                    'Cars',
                    'cars',
                    'log cars')
```



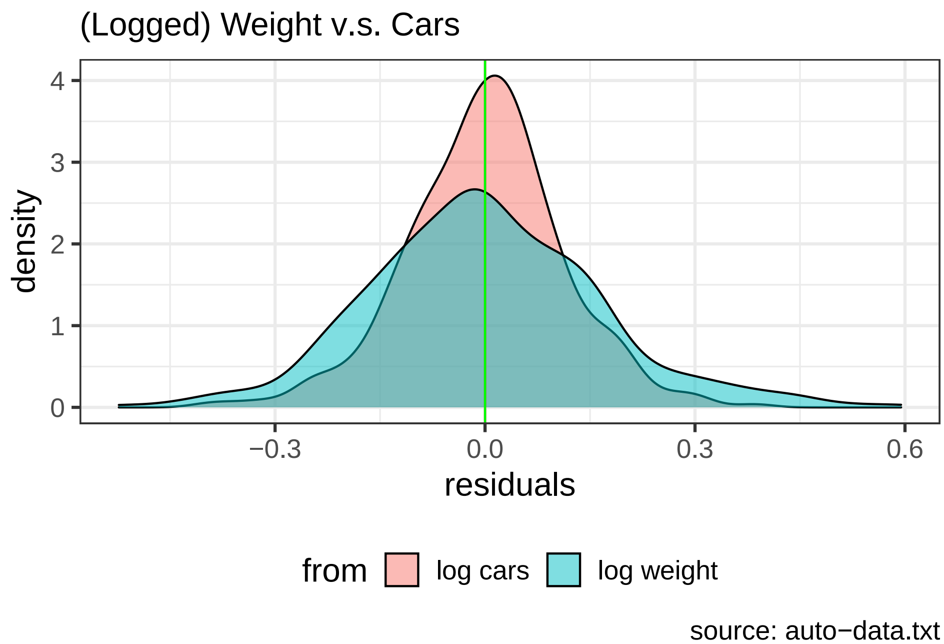
```r
tibble_density_plot(regr_wt,
                    regr_wt_log,
                    'Weight',
                    'weights',
                    'log weights')
```




```r
tibble_density_plot(regr_wt,
                    regr_cars,
                    'Weight v.s. Cars',
                    'weights',
                    'cars')
```

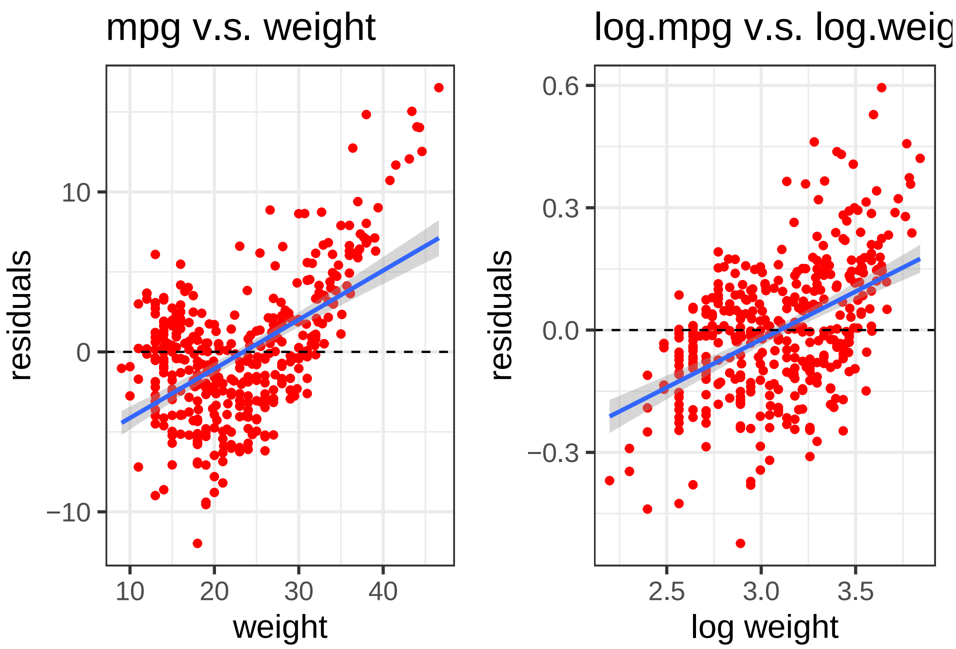


```
tibble_density_plot(regr_wt_log,
                    regr_cars_log,
                    '(Logged) Weight v.s. Cars',
                    'log weight',
                    'log cars')
```



2. Scatterplot of `log(weight)` vs. `residuals`

```
p1 <- ggplot()+
  aes(cars$mpg, resid(regr_wt))+
  geom_point(col="red")+
  stat_smooth(method=lm)+
  geom_hline(yintercept=0, lty='dashed')+
  labs(x='weight', y='residuals')+
  ggtitle('mpg v.s. weight')

p2 <- ggplot()+
  aes(cars_log$mpg, resid(regr_wt_log))+
  geom_point(col="red")+
  stat_smooth(method=lm)+
  geom_hline(yintercept=0, lty='dashed')+
  labs(x='log weight', y='residuals')+
  ggtitle('log.mpg v.s. log.weight')

grid.arrange(p1, p2, ncol=2)
```



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

Table 2: Regression Assumptions

Requirements	Implications
1. random, normally distributed error terms, with mean(ϵ)=0.	values of $\bar{y}$ are on the regression line. $\hat{\beta}$ are symmetrically distributed.
2. $var(\epsilon)$ is the same for all values of x.	The distribution of y is the same across values of x.
3. ϵ are independent.	The values of y are independent.

Ans. As we can observe from the plots above, the log-transformed regression produces a better distributed residuals that looks less likely to a curve.

- ***v.*** Interpret the slope of `log(weight)` vs `log(mpg)` in simple words.

```
summary(regr_wt)
```

```
##
## Call:
## lm(formula = mpg ~ weight, data = cars, na.action = na.exclude)
##
## 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
```

```
summary(regr_wt_log)
```

```
##
## Call:
## lm(formula = mpg ~ weight, data = cars_log, na.action = na.exclude)
##
## 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 ***
## 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
```

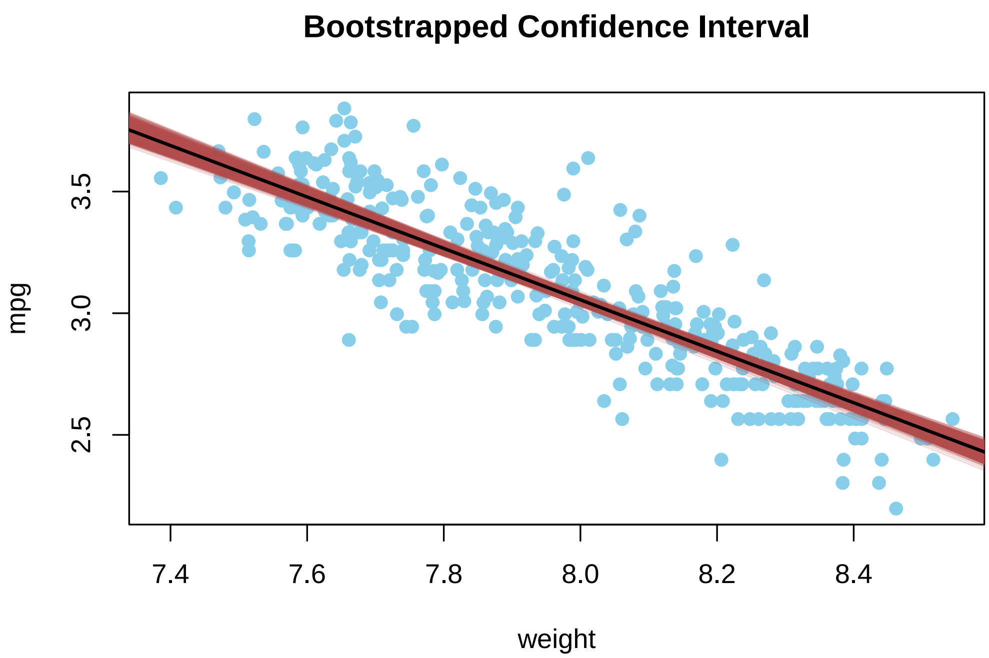
- **Ans.** The slope of `log(mpg)` vs. `log(weight)` is much flatter then the `mpg` vs. `weight` due to the log transformation.

c. Examine the 95% confidence interval of the slope of `log(weight)` vs. `log(mpg)`.

- ***i.*** Create a bootstrapped confidence interval.

```
# empty canvas
plot(log(cars$weight),
     log(cars$mpg),
     type="n",
     xlab='weight',
     ylab='mpg',
     main='Bootstrapped Confidence Interval')
# function for single resampleed reggresion line
points(log(cars$weight), log(cars$mpg), col="skyblue", pch=19)

boot_regr <- function(model, dataset){
  # random row index number
  boot_index <- sample(1:nrow(dataset), replace=TRUE)
  data_boot <- dataset[boot_index, ] # picking the rows
  regr_boot <- lm(model, data=data_boot) # run regression model
  abline(regr_boot, lwd=1, col=rgb(0.7, 0.3, 0.3, 0.2), alpha=0.05)
  regr_boot$coefficients
}
coeffs <- replicate(3000, boot_regr(log(mpg)~log(weight), cars))

abline(a = mean(coeffs["(Intercept)", ]),
```

```
       b = mean(coeffs["log(weight)", ]),
       lwd=2)
```




```
# Confidence interval values
round(quantile(coeffs["log(weight)", ], c(0.025, 0.975)), 3)
```

```
##   2.5%  97.5%
## -1.110 -1.004
```

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

```
# traditional way
regression_model <- lm(log(mpg)~log(weight), data=cars)
summary(regression_model)
```

```
##
## Call:
## lm(formula = log(mpg) ~ log(weight), data = cars)
##
## 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
```

```
# estimate +- 1.96*stderr
round(regression_model$coefficients['log(weight)']+c(-1.96, 1.96)*0.0297, 3)
```

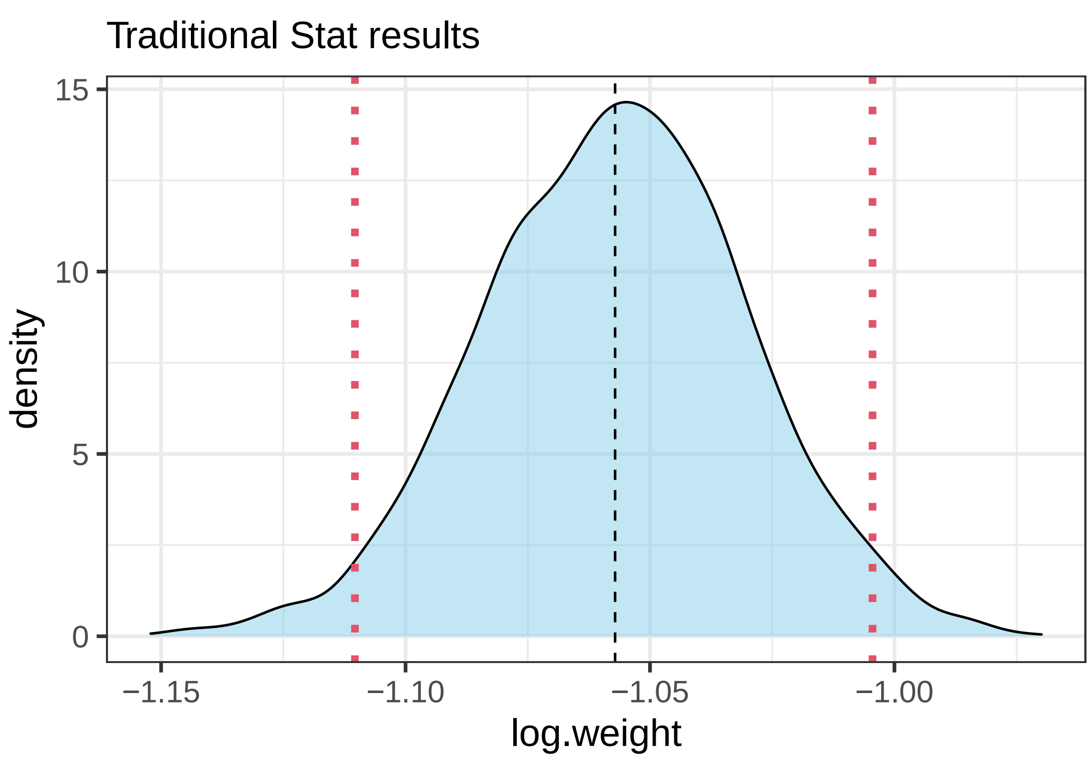
```
## [1] -1.116 -0.999
```

```
round(confint(regression_model,'log(weight)', level=0.95), 3)
```

```
##              2.5 % 97.5 %
## log(weight) -1.116 -0.999
```

- **Note.** Slightly different results were presented with different computing methods.
- When your are not having a decent dataset, bootstrapping will be a more reliable way to compute the regression.

```
ggplot()+
  aes(coeffs['log(weight)', ])+
  geom_density(fill='skyblue', alpha=0.5)+
  geom_vline(xintercept=mean(coeffs['log(weight)',]), lty='dashed')+
  geom_vline(xintercept=quantile(coeffs['log(weight)', ],c(0.025, 0.975)),
             col=2,
             lwd=1.5,
             lty='dotted')+
  labs(x='log.weight', subtitle = 'Traditional Stat results')
```



Note. By finding the slope we get an estimate of the slope by which the dependent variable(mpg) is expected to increase or decrease.

The Confident interval provides the range of the slope values that we expect 95% of the times when the sample size is the same.

Since obviously neither of the two results includes 0 in their confident interval, we can conclude that there is a significant linear relationship between weight and mpg.

Question 2) Multicollinearity

```
regr_log <- LMOfCars(cars_log)
# diagnosing multicollinearity

weight_regr <- lm(weight~
                    cylinders+
                    displacement+
                    horsepower+
                    acceleration+
                    cars$model_year+
                    factor(origin),
                  data=cars_log,
                  na.action = na.exclude)

r2_log_weight <- summary(weight_regr)$r.squared
r2_log_weight
```

```
## [1] 0.9431014
```

a. Compute the VIF of `log(weight)`.

```
vif_log_weight <- 1/(1-r2_log_weight)
vif_log_weight
```

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

- Multicollinearity inflates the variance of the weights by more than 17 times.

```
sqrt(vif_log_weight) >2
```

```
## [1] TRUE
```

- The high multicollinearity implies that "log weight" shares more than half of its variance with other independent variables.

b. Use Stepwise VIF Selection to remove highly collinear predictors.

- ***i.*** Use `vif(regr_log)` to compute VIF.

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

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

```r
sort(vif_log_cars[,'GVIF'], decreasing=TRUE) # generalized VIF
```

```
##    displacement          weight      horsepower       cylinders    acceleration
##       29.625732       17.575117       12.132057       10.456738        3.570357
##  factor(origin) auto$model_year
##        2.656795        1.303738
```

```r
# displacement should be removed
multicollinearity <- function(model, data){
  LM <- lm(model, data)
  sort_order <- sort(vif(LM)[, 'GVIF'], decreasing=TRUE)
  if (unname(sort_order[1] >5 )==TRUE){
    print('Variable you should remove next: ')
    names(sort_order)[1]
  }else{
    print('No more vif of variable is larger than 5.')
    return(LM)
  }

}

multicollinearity(mpg~cylinders+
                    displacement+
                    horsepower+
                    weight+
                    acceleration+
                    cars$model_year+
                    factor(origin),
                  cars_log)
```

```
## [1] "Variable you should remove next: "
## [1] "displacement"
```

- ***iii.*** Repeat steps (i) and (ii)

```r
multicollinearity(mpg~cylinders+
                    horsepower+
                    weight+
                    acceleration+
                    cars$model_year+
                    factor(origin), cars_log)
```

```
## [1] "Variable you should remove next: "
## [1] "horsepower"
```

```r
multicollinearity(mpg~cylinders+
                    weight+
                    acceleration+
                    cars$model_year+
                    factor(origin), cars_log)
```

```
## [1] "Variable you should remove next: "
```

```
## [1] "cylinders"
```

```
multicollinearity(mpg~weight+
                    acceleration+
                    cars$model_year+
                    factor(origin), cars_log)
```

```
## [1] "No more vif of variable is larger than 5."
```

```
##
## Call:
## lm(formula = model, data = data)
##
## Coefficients:
##     (Intercept)           weight     acceleration  cars$model_year
##         7.41097         -0.87550          0.05438          0.03279
## factor(origin)2  factor(origin)3
##         0.05611          0.03194
```

```
final_Regression_model <- multicollinearity(mpg~weight+
                                              acceleration+
                                              cars$model_year+
                                              factor(origin), cars_log)
```

```
## [1] "No more vif of variable is larger than 5."
```

- ***iv.*** Report the final regression model and its summary statistics.

```
summary(final_Regression_model)
```

```
##
## Call:
## lm(formula = model, data = data)
##
## 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 ***
## weight          -0.875499   0.029086 -30.101  < 2e-16 ***
## acceleration     0.054377   0.037132   1.464  0.14389
## cars$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
```

```r
# to compare
summary(LMOfCars(cars_log))
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + auto$model_year + factor(origin), data = data,
##     na.action = na.exclude)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39727 -0.06880  0.00450  0.06356  0.38542
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       7.301938   0.361777  20.184  < 2e-16 ***
## cylinders        -0.081915   0.061116  -1.340  0.18094
## displacement      0.020387   0.058369   0.349  0.72707
## horsepower       -0.284751   0.057945  -4.914 1.32e-06 ***
## weight           -0.592955   0.085165  -6.962 1.46e-11 ***
## acceleration     -0.169673   0.059649  -2.845  0.00469 **
## auto$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
## Multiple R-squared:  0.8919,   Adjusted R-squared:  0.8897
## F-statistic:   395 on 8 and 383 DF,  p-value: < 2.2e-16
```

c. Have we lost any variables that were previously significant? If so, how much did we hurt our explanation?

```r
model_fit <- function(y, yhat){
  variances <- list()
  variances$SSE <- sum((yhat-y)^2)
  variances$SSR <- sum((yhat-mean(y))^2)
  variances$SST <- sum((y-mean(y))^2)
  variances$Rsq <- sum((yhat-mean(y))^2)/sum((y-mean(y))^2)
  return(variances)
}

model_fit(cars_log$mpg, regr_log$fitted.values)$Rsq
```

```
## [1] 0.89191
```

```r
model_fit(cars_log$mpg, final_Regression_model$fitted.values)$Rsq
```

```
## [1] 0.8844856
```

```
summary(regr_log)$r.squared-summary(final_Regression_model)$r.squared
```

```
## [1] 0.007424334
```

- **Ans.** Yes, we lose horsepower and weight that were previously significant.
  - About 0.74%

d. From only the formula for VIF...

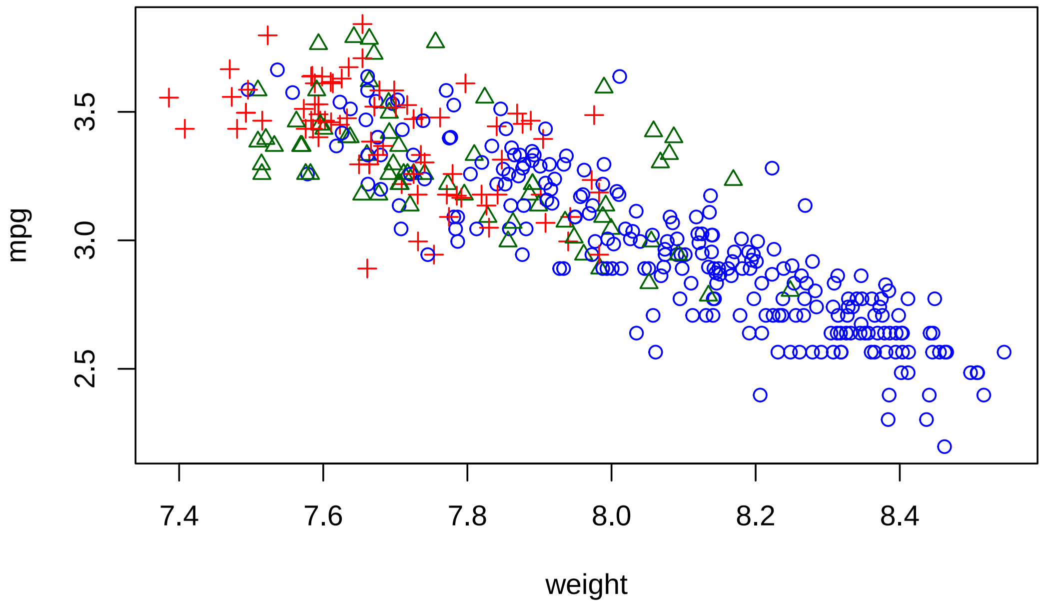
- ***i.*** If an independent variable has no correlation with other independent variables, what would its VIF score be?
  - **Ans.** 1, when multicollinearity does not exist, standard error of an independent variable would not inflated.
- ***ii.*** Given a regression with only two independent variables (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?
  - **Ans.**
    $vif = [\frac{1}{(1-r^2)} > 5] = [r > \sqrt{\frac{4}{5}}]$
    indicating that the coefficient of multiple correlation between dependent variable and independent variables should be greater that 80%
    $vif = [\frac{1}{(1-r^2)} > 10] = [r > \sqrt{\frac{9}{10}}]$
    This indicates that the coefficient of multiple correlation between dependent variable and independent variables should be greater that 90%

Question 3) Visualization

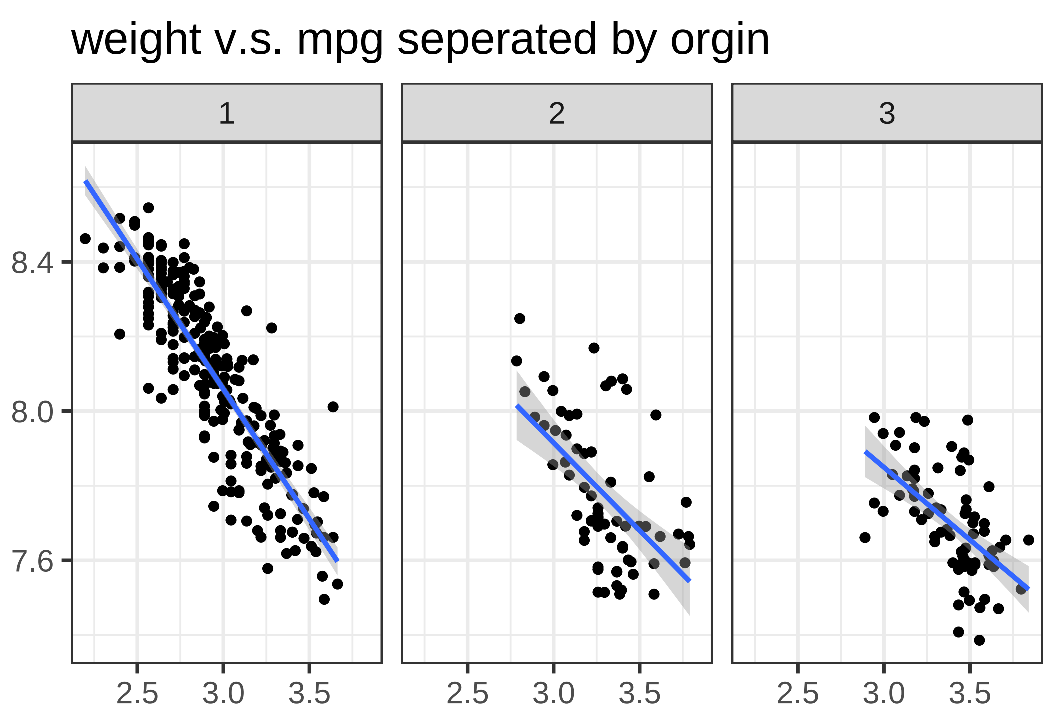
Might the relationship of weight on mpg be different for cars from different origins?

a. Add three separate regression lines on the scatterplot.

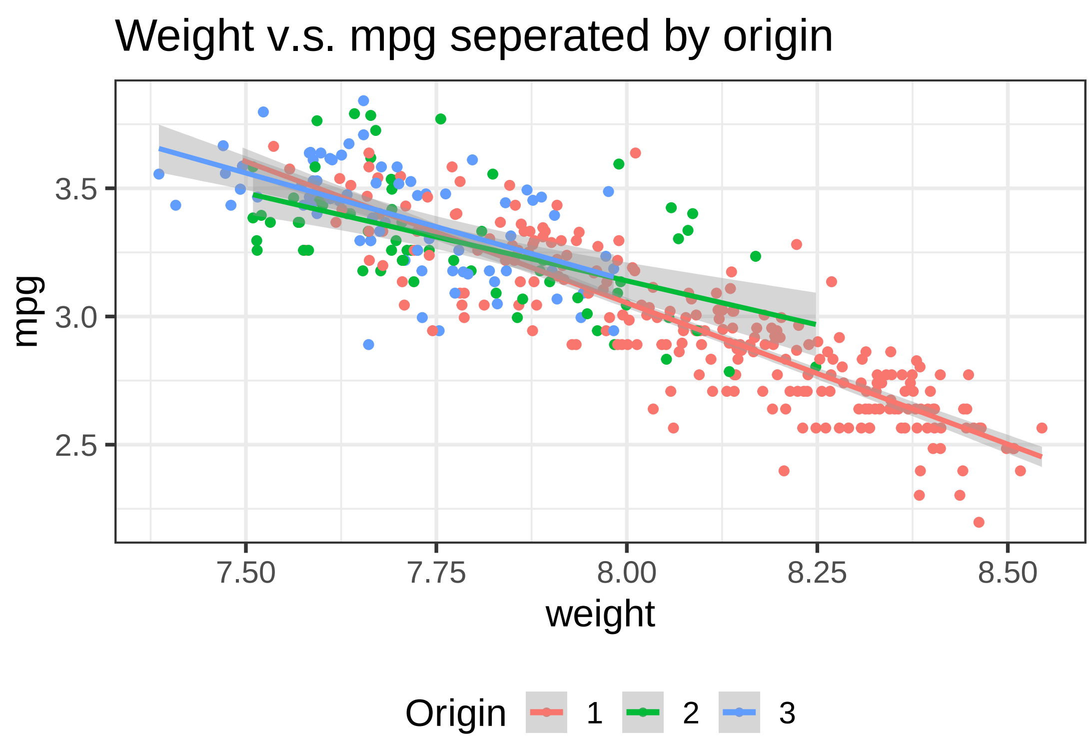
```
origin_colors = c("blue", "darkgreen", "red")
with(cars_log, plot(weight, mpg, pch=origin, col=origin_colors[origin]))
```




```
plt(cars_log$mpg, cars_log$weight, 'weight v.s. mpg seperated by orgin')
```



```r
ggplot(cars_log, aes(weight, mpg, color=factor(origin)))+
  geom_point(size=1.5)+
  stat_smooth(method=lm)+
  labs(color='Origin')+
  theme(legend.position = 'bottom')+
  ggtitle('Weight v.s. mpg seperated by origin')+
  guides(color=guide_legend(override.aes = list(size=1.2)))
```



b. Do cars from different origins appear to have different weight vs. mpg relationships?

Ans. I think so, yes.