HW12

108048110

5/6/2022

BACS HW - Week 12

Prerequisite

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

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

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

cars_log <- with(auto, data.frame(log(mpg),
                                  log(weight),
                                  log(acceleration),
                                  auto$model_year,
                                  auto$origin))

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

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

mpg	weight	acceleration	model_year	origin
2.890372	8.161660	2.484907	70	1
2.708050	8.214194	2.442347	70	1
2.890372	8.142063	2.397895	70	1
2.772589	8.141190	2.484907	70	1

mpg	weight	acceleration	model_year	origin
2.833213	8.145840	2.351375	70	1
2.708050	8.375860	2.302585	70	1

Question 1) Visualize how `weight` and `acceleration` are related to `mpg`.

a. Visualize how `weight` might moderate the relationship between `acceleration` and `mpg`.

- ***i.*** Create two subsets of your data, one for *light-weight cars*, and one for *heavy car*.

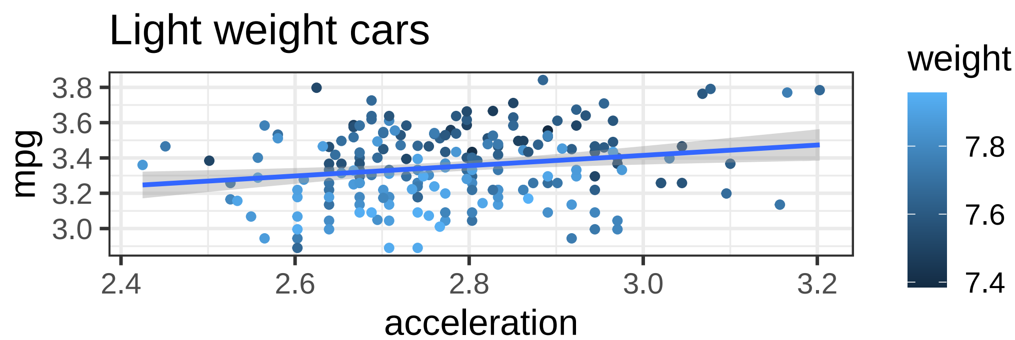
```
# Moderate
light_wg_cars <- cars_log %>% subset(weight<mean(weight))
heavy_wg_cars <- cars_log %>% subset(weight>=mean(weight))
```

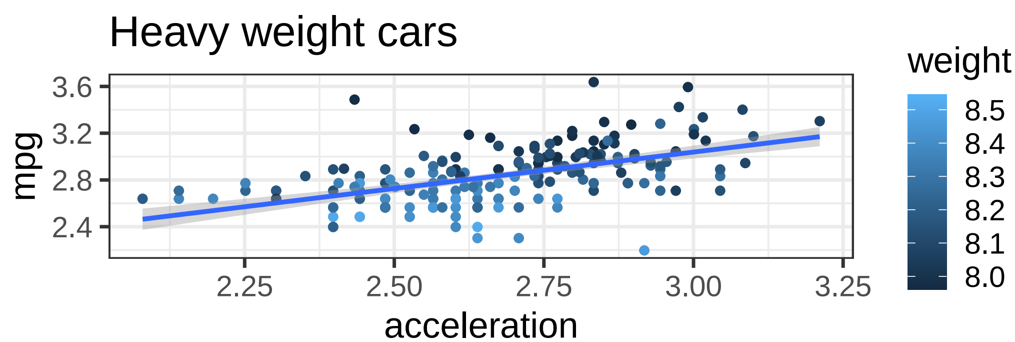
  Note. Simple example for doing the log transformation on the original data first.

  - Say you have 3 measurements with values of 1, 10, and 100.
    Their mean value is 111/3=37. The base 10 logarithm of 37 is 1.57, which is the log of their mean value in the original scale.
    With the base 10 logarithms of the original data are 0, 1, and 2; the mean of the logarithms is 1, corresponding to a value of 10 in the original scale.
    As a result, mean of the log does not equal the log of the mean.
    So if the log transformation of the data is appropriate, you should always do the transformation on the original data first.

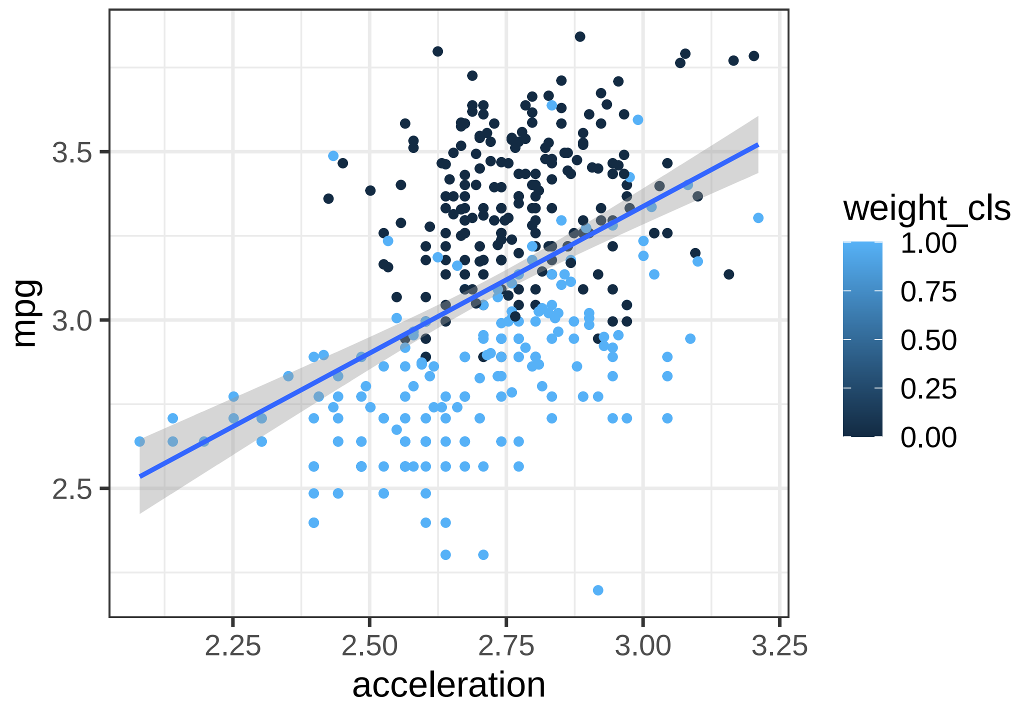
- ***ii.*** Create a single scatter plot of `acceleration` vs. `mpg`, with different colors and/or shapes for *light* vs. *heavy cars*.

```
p1 <- ggplot(light_wg_cars)+
  geom_point(aes(acceleration, mpg, color=weight))+
  stat_smooth(aes(acceleration, mpg), method=lm)+
  ggtitle('Light weight cars')

p2 <- ggplot(heavy_wg_cars)+
  geom_point(aes(acceleration, mpg, color=weight))+
  stat_smooth(aes(acceleration, mpg), method=lm)+
  ggtitle('Heavy weight cars')

grid.arrange(p1, p2)
```



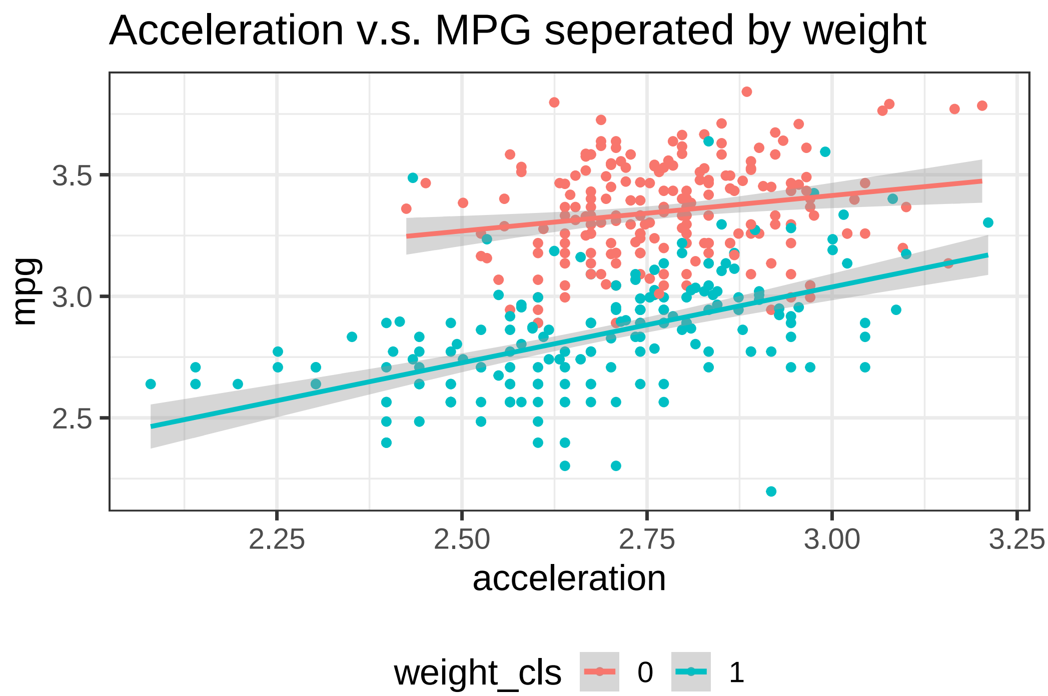



```
# putting them together
weight_cls <- as.numeric(cars_log[,'weight']>mean(cars_log$weight))
cars_log <- cbind(cars_log, weight_cls)
# Heavy:1, light:0

ggplot(cars_log)+
  geom_point(aes(x=acceleration, y=mpg, color=weight_cls))+
  stat_smooth(aes(y=mpg, x=acceleration), method=lm)
```



- ***iii.*** Draw two slopes of *acceleration-vs-mpg* over the scatter plot.

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



b. Report the full summaries of two separate regressions for *light* and *heavy cars* where `log.mpg.` is dependent on `log.weight.`, `log.acceleration.`, `model_year` and `origin`.

```
# light
summary(with(light_wg_cars, lm(mpg~weight+acceleration+model_year+factor(origin))))
```

```
##
## Call:
## lm(formula = mpg ~ weight + acceleration + model_year + factor(origin))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.36590 -0.06612  0.00637  0.06333  0.31513
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.809014   0.598446  11.378   <2e-16 ***
## weight           -0.821951   0.065769 -12.497   <2e-16 ***
## acceleration      0.111137   0.058297   1.906   0.0580 .
## model_year        0.033344   0.002049  16.270   <2e-16 ***
## factor(origin)2   0.042309   0.020926   2.022   0.0445 *
## factor(origin)3   0.020923   0.019210   1.089   0.2774
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1102 on 199 degrees of freedom
## Multiple R-squared:  0.7093,   Adjusted R-squared:  0.702
## F-statistic:  97.1 on 5 and 199 DF,  p-value: < 2.2e-16
```

```r
# heavy
summary(with(heavy_wg_cars, lm(mpg~weight+acceleration+model_year+factor(origin))))
```

```
##
## Call:
## lm(formula = mpg ~ weight + acceleration + model_year + factor(origin))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37099 -0.07224  0.00150  0.06704  0.42751
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       7.132892   0.677740  10.525  < 2e-16 ***
## weight           -0.825517   0.068101 -12.122  < 2e-16 ***
## acceleration      0.031221   0.055465   0.563  0.57418
## model_year        0.031735   0.003254   9.752  < 2e-16 ***
## factor(origin)2   0.099027   0.033840   2.926  0.00386 **
## factor(origin)3   0.063148   0.065535   0.964  0.33650
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1212 on 187 degrees of freedom
## Multiple R-squared:  0.7585,	Adjusted R-squared:  0.752
## F-statistic: 117.4 on 5 and 187 DF,  p-value: < 2.2e-16
```

c. What do you observe about *light vs. heavy cars* so far?

Ans. By observing the R^2 values, I deducted that vehicles that weigh heavier tend to be more related to the target explanatory variable mpg.

Question 2) Using the fully transformed dataset from `cars_log`, to test whether we have moderation.

a. Between `weight` and `acceleration`, use your intuition and experience to state which variable might be a moderating versus independent variable, in affecting mileage.

- **Ans.** Guessing from my inexperienced intuition, I state that the weight might be a moderator of accerleration affecting mileage.

b. Use various regression models to model the possible moderation on `log.mpg`.

```r
# drop the weight class
cars_log = cars_log[,-length(cars_log)]

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

mpg	weight	acceleration	model_year	origin
2.890372	8.161660	2.484907	70	1
2.708050	8.214194	2.442347	70	1
2.890372	8.142063	2.397895	70	1
2.772589	8.141190	2.484907	70	1
2.833213	8.145840	2.351375	70	1
2.708050	8.375860	2.302585	70	1

- **Identifying symptoms of multicollinearity!**
- – ***i.*** Report a regression without any interaction terms.

```
summary(lm(mpg~
             weight+
             acceleration+
             model_year+
             factor(origin),
           data = cars_log))
```

```
##
## Call:
## lm(formula = mpg ~ weight + 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 ***
## weight          -0.876608   0.028697 -30.547  < 2e-16 ***
## 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
```

- – ***ii.*** Report a regression with an interaction between `weight` and `acceleration`.

```
interaction_term <- cars_log$weight*cars_log$acceleration
summary(lm(mpg~
            weight+
            acceleration+
            model_year+
            factor(origin)+
            interaction_term,
          data=cars_log))
```

```
##
## Call:
## lm(formula = mpg ~ weight + acceleration + model_year + factor(origin) +
##     interaction_term, data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37807 -0.06868  0.00463  0.06891  0.39857
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.089642   2.752872   0.396  0.69245
## weight           -0.096632   0.337637  -0.286  0.77488
## acceleration      2.357574   0.995349   2.369  0.01834 *
## model_year        0.033685   0.001735  19.411  < 2e-16 ***
## factor(origin)2   0.058737   0.017789   3.302  0.00105 **
## factor(origin)3   0.028179   0.018266   1.543  0.12370
## interaction_term -0.287170   0.123866  -2.318  0.02094 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.115 on 391 degrees of freedom
## Multiple R-squared:  0.8871,   Adjusted R-squared:  0.8854
## F-statistic: 512.2 on 6 and 391 DF,  p-value: < 2.2e-16
```

- – ***iii.*** Report a regression with a **mean-centered interaction term**.

```r
# class of origin is originally 'factor'
# thus I transform it into numeric data type in order to scale it and develop a linear model.

# Mean-centered
log_weight_mc = scale(cars_log$weight, scale=FALSE)
log_acc_mc = scale(cars_log$acceleration, scale=FALSE)
interaction_term = log_weight_mc*log_acc_mc

with(cars_log, cor(log_weight_mc, interaction_term))
```

```
##              [,1]
## [1,] -0.2026948
```

```r
with(cars_log, cor(log_acc_mc, interaction_term))
```

```
##             [,1]
## [1,] 0.3512271
```

```r
summary(lm(mpg~
             weight+
             acceleration+
             model_year+
             factor(origin)+
             interaction_term,
           data = cars_log))
```

```
##
## Call:
```

```
## lm(formula = mpg ~ weight + acceleration + model_year + factor(origin) +
##     interaction_term, data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37807 -0.06868  0.00463  0.06891  0.39857
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       7.325939   0.313819  23.344  < 2e-16 ***
## weight           -0.880393   0.028585 -30.799  < 2e-16 ***
## acceleration      0.072596   0.037567   1.932  0.05403 .
## model_year        0.033685   0.001735  19.411  < 2e-16 ***
## factor(origin)2   0.058737   0.017789   3.302  0.00105 **
## factor(origin)3   0.028179   0.018266   1.543  0.12370
## interaction_term -0.287170   0.123866  -2.318  0.02094 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.115 on 391 degrees of freedom
## Multiple R-squared:  0.8871,   Adjusted R-squared:  0.8854
## F-statistic: 512.2 on 6 and 391 DF,  p-value: < 2.2e-16
```

  - As we can observe from the summary table, there is no change in significance or R^2 value; thus, we can only improve interpretability of the coefficients.

- - ***iv.*** Report a regression with an **orthogonalized interaction term**.

```r
# predict weight by cylinders
interaction_term = cars_log$weight*cars_log$acceleration

interaction_pred = lm(interaction_term~
                         cars_log$weight+
                         cars_log$acceleration)
interaction_ortho = interaction_pred$residuals

summary(lm(mpg~
             weight+
             acceleration+
             model_year+
             factor(origin)+
             interaction_ortho,
           data=cars_log))
```

```
##
## Call:
## lm(formula = mpg ~ weight + acceleration + model_year + factor(origin) +
##     interaction_ortho, data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37807 -0.06868  0.00463  0.06891  0.39857
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        7.377176   0.311392  23.691  < 2e-16 ***
```

```
## weight             -0.876967   0.028539 -30.729  < 2e-16 ***
## acceleration        0.046100   0.036524   1.262  0.20764
## model_year          0.033685   0.001735  19.411  < 2e-16 ***
## factor(origin)2     0.058737   0.017789   3.302  0.00105 **
## factor(origin)3     0.028179   0.018266   1.543  0.12370
## interaction_ortho -0.287170   0.123866  -2.318  0.02094 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.115 on 391 degrees of freedom
## Multiple R-squared:  0.8871,   Adjusted R-squared:  0.8854
## F-statistic: 512.2 on 6 and 391 DF,  p-value: < 2.2e-16
```

```
# Compare to the Original dataset
summary(with(cars_log, lm(mpg~
                            weight+
                            acceleration+
                            model_year+
                            factor(origin)
                          )
             )
)
```

```
##
## Call:
## lm(formula = mpg ~ weight + acceleration + model_year + factor(origin))
##
## 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 ***
## weight          -0.876608   0.028697 -30.547  < 2e-16 ***
## 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
```

- **Note.** We still can not statistically remove multicollinearity from the model.
- Despite the fact that the estimating value looks almost the same as the original data, as we can observe from the summary table, the standard error did slightly decrease.
- As a result, we can conclude that orthogonalization gives us the most interpretable coefficients; however, it still does not statistically remove multicollinearity from our explanatory model.

c. For each of the interaction term strategies above, what is the correlation between that interaction term and the two variables that you multiplied together?

```r
# correlation plot function
cor_plt <- function(data){
  cor_data <- round(cor(data[, 1:length(data)], 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')
}
```

- – it seems that acceleration has a higher correlation with the interactive term.

```r
# raw:
# calculate cor between weight, acceleration with interactive term respectively.
cor(cars_log)
```

```
##                     mpg     weight acceleration model_year     origin
## mpg           1.0000000 -0.8744686    0.4640533  0.5763423  0.5583293
## weight       -0.8744686  1.0000000   -0.4256194 -0.2840090 -0.6048831
## acceleration  0.4640533 -0.4256194    1.0000000  0.3107471  0.2210906
## model_year    0.5763423 -0.2840090    0.3107471  1.0000000  0.1806622
## origin        0.5583293 -0.6048831    0.2210906  0.1806622  1.0000000
```

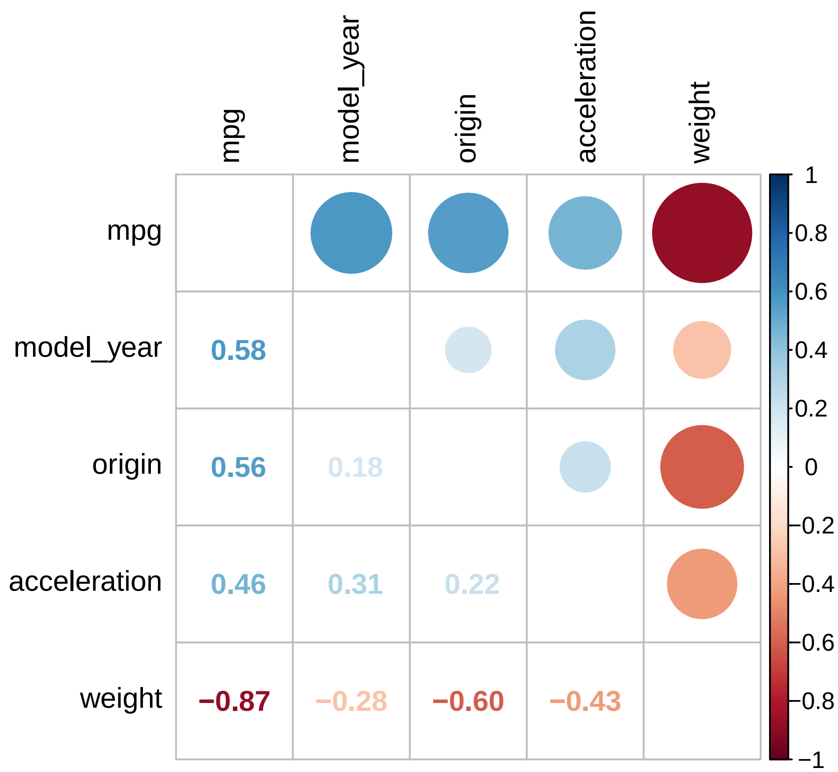
```r
interaction_term = cars_log$weight*cars_log$acceleration
cor(cars_log$weight, interaction_term)
```

```
## [1] 0.1083055
```

```r
cor(cars_log$acceleration, interaction_term)
```

```
## [1] 0.852881
```

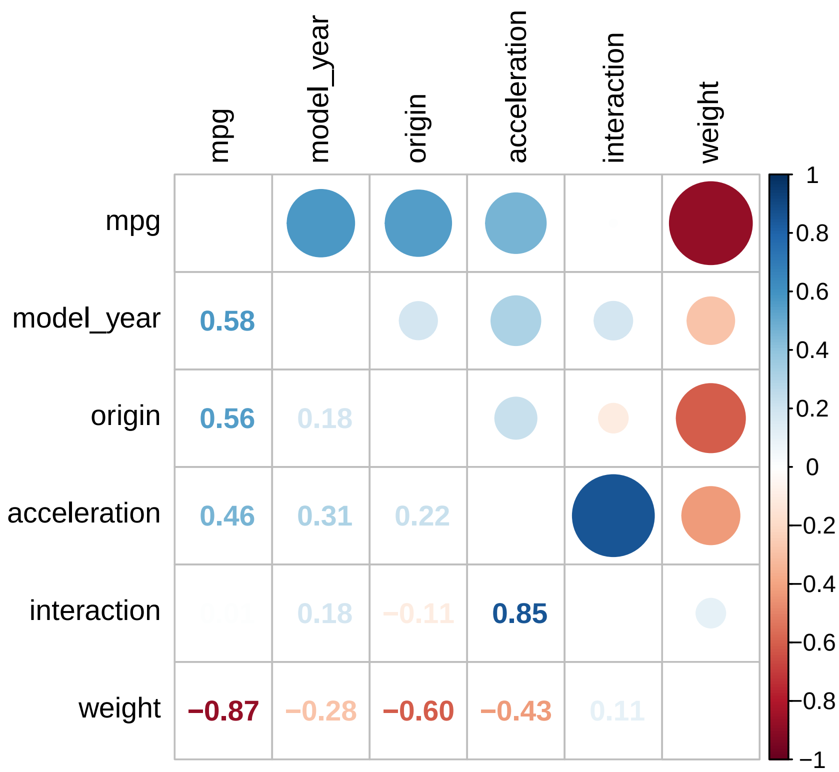
```r
cor_plt(cars_log)
```




```r
# interaction(weight*acc):
with_interaction <- cbind(cars_log, 'interaction'=cars_log$weight*cars_log$acceleration)
cor(with_interaction)
```

```
##                       mpg     weight acceleration model_year     origin
## mpg           1.000000000 -0.8744686    0.4640533  0.5763423  0.5583293
## weight       -0.874468594  1.0000000   -0.4256194 -0.2840090 -0.6048831
## acceleration  0.464053310 -0.4256194    1.0000000  0.3107471  0.2210906
## model_year    0.576342261 -0.2840090    0.3107471  1.0000000  0.1806622
## origin        0.558329285 -0.6048831    0.2210906  0.1806622  1.0000000
## interaction   0.007445392  0.1083055    0.8528810  0.1853457 -0.1078488
##               interaction
## mpg           0.007445392
## weight        0.108305532
## acceleration  0.852881042
## model_year    0.185345672
## origin       -0.107848822
## interaction   1.000000000
```

```r
cor_plt(with_interaction)
```




```
# mean-centered:
scaled_w <- scale(cars_log$weight, center=TRUE, scale=FALSE)
scaled_a <- scale(cars_log$weight, center=TRUE, scale=FALSE)
interaction_term_mc <- scaled_w * scaled_a
cor(scaled_w, interaction_term_mc) # == cor()
```

```
##           [,1]
## [1,] 0.1631556
```

```
cor(scaled_a, interaction_term_mc)
```

```
##           [,1]
## [1,] 0.1631556
```

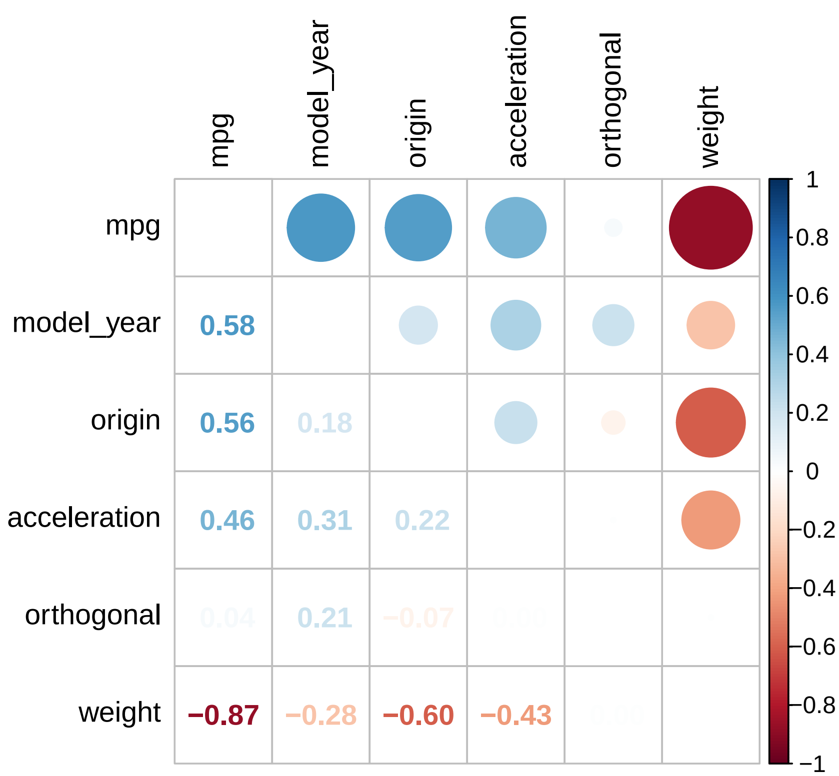
- As we can observe from the above summary table, it does not matter whether you scale the data or not you get the same R^2 value; however, compare to the original data, mean-centered method does decrease the standard error value dramatically.

```
# !!!!! MIND THIS ONE !!!!!!
# orthogonal
interaction_term = cars_log$weight * cars_log$acceleration

# In order to calculate the orthogoanl interaction
# We should predict the interaction term by factors!!!!
car_log_regr <- lm(interaction_term~
                     cars_log$weight+
                     cars_log$acceleration)
```

```r
# And then calculate the residuals
interaction_ortho = car_log_regr$residuals

with_orthogonal <- cbind(cars_log, 'orthogonal'= interaction_ortho)
cor(with_orthogonal)
```

```
##                      mpg        weight  acceleration model_year      origin
## mpg           1.00000000 -8.744686e-01  4.640533e-01  0.5763423  0.55832929
## weight       -0.87446859  1.000000e+00 -4.256194e-01 -0.2840090 -0.60488314
## acceleration  0.46405331 -4.256194e-01  1.000000e+00  0.3107471  0.22109064
## model_year    0.57634226 -2.840090e-01  3.107471e-01  1.0000000  0.18066220
## origin        0.55832929 -6.048831e-01  2.210906e-01  0.1806622  1.00000000
## orthogonal    0.03586215  2.468461e-17 -6.804111e-17  0.2107232 -0.06656733
##                  orthogonal
## mpg           3.586215e-02
## weight        2.468461e-17
## acceleration -6.804111e-17
## model_year    2.107232e-01
## origin       -6.656733e-02
## orthogonal    1.000000e+00
```

```r
cor_plt(with_orthogonal)
```




```r
cor(cars_log$weight, interaction_ortho)
```

```
## [1] 2.468461e-17
```

```r
cor(cars_log$acceleration, interaction_ortho)
```

```
## [1] -6.804111e-17
```

Question 3) Might cylinders have an indirect relationship with mpg through its weight?

a. Try computing the direct effects first.

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

# Add cylinders column into our estimation
cars_log <- with(auto, data.frame(log(mpg),
                                  log(weight),
                                  log(acceleration),
                                  auto$model_year,
                                  auto$origin,
                                  log(cylinders)))

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

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

mpg	weight	acceleration	model_year	origin	cylinders
2.890372	8.161660	2.484907	70	1	2.079442
2.708050	8.214194	2.442347	70	1	2.079442
2.890372	8.142063	2.397895	70	1	2.079442
2.772589	8.141190	2.484907	70	1	2.079442
2.833213	8.145840	2.351375	70	1	2.079442
2.708050	8.375860	2.302585	70	1	2.079442

```r
cor(cars_log)
```

```
##                     mpg     weight acceleration model_year     origin
## mpg           1.0000000 -0.8744686    0.4640533  0.5763423  0.5583293
## weight       -0.8744686  1.0000000   -0.4256194 -0.2840090 -0.6048831
## acceleration  0.4640533 -0.4256194    1.0000000  0.3107471  0.2210906
## model_year    0.5763423 -0.2840090    0.3107471  1.0000000  0.1806622
## origin        0.5583293 -0.6048831    0.2210906  0.1806622  1.0000000
## cylinders    -0.8204483  0.8810356   -0.5047899 -0.3403128 -0.5757918
##               cylinders
```

```
## mpg          -0.8204483
## weight        0.8810356
## acceleration -0.5047899
## model_year   -0.3403128
## origin       -0.5757918
## cylinders     1.0000000
```

```
# We can easily tell from the correlation matrix that cylinders and weight are
# highly correlated.
```

- ***i.*** Model 1: Regress `log.weight.` over `log.cylinders` only.

```
# predict weight by cylinders
model_1 <- lm(weight~cylinders, data=cars_log)
summary(model_1)
```

```
##
## Call:
## lm(formula = weight ~ cylinders, data = cars_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.35473 -0.09076 -0.00147  0.09316  0.40374
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.60365    0.03712  177.92   <2e-16 ***
## cylinders    0.82012    0.02213   37.06   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1329 on 396 degrees of freedom
## Multiple R-squared:  0.7762,   Adjusted R-squared:  0.7757
## F-statistic:  1374 on 1 and 396 DF,  p-value: < 2.2e-16
```

- ***ii.*** Model 2: Regress `log.mpg.`over `log.weight.` and all control variables.

```
# we should include all factors instead of "Cylinders" to calculate the direct effects on the 

model_2 <- lm(mpg~
                weight+
                acceleration+
                model_year+
                factor(origin)
              , data=cars_log)
summary(model_2)
```

```
##
## Call:
## lm(formula = mpg ~ weight + 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 ***
## weight           -0.876608   0.028697 -30.547  < 2e-16 ***
## 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
```

b. What is the indirect effect of cylinders on mpg?

```
model_1$coefficients[2] * model_2$coefficients[2]
```

```
##  cylinders
## -0.7189275
```

c. Bootstrap for the confidence interval of the indirect effect of `cylinders` on `mpg`.

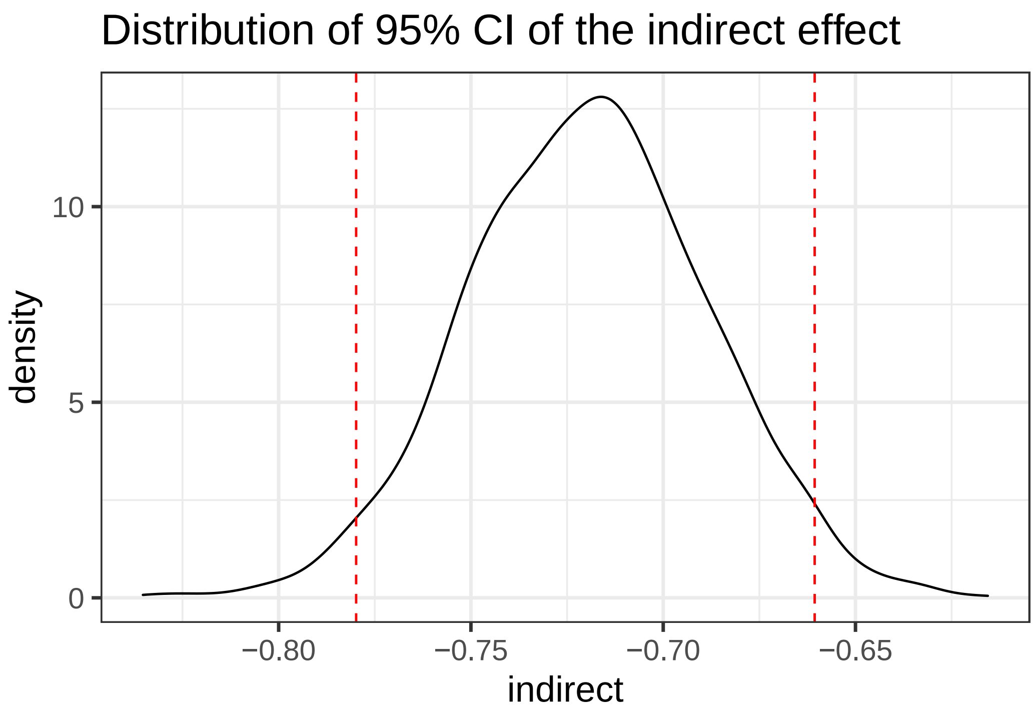
- ***i.*** Bootstrap regression models 1 & 2, and compute the indirect effect each time. What is its 95% CI of the indirect effect of `log.cylinders.` on `log.mpg.`?

```
boot_model <- function(model1, model2, dataset){
  boot_index <- sample(1:nrow(dataset), replace=TRUE)
  new_data <- dataset[boot_index,]
  regr1 <- lm(model1, new_data)
  regr2 <- lm(model2, new_data)
  return(regr1$coefficients[2]*regr2$coefficients[2])
}

indirect <- replicate(2000, boot_model(model_1, model_2, cars_log))
quantile(indirect, probs=c(0.025, 0.975))
```

```
##       2.5%      97.5%
## -0.7798486 -0.6606180
```

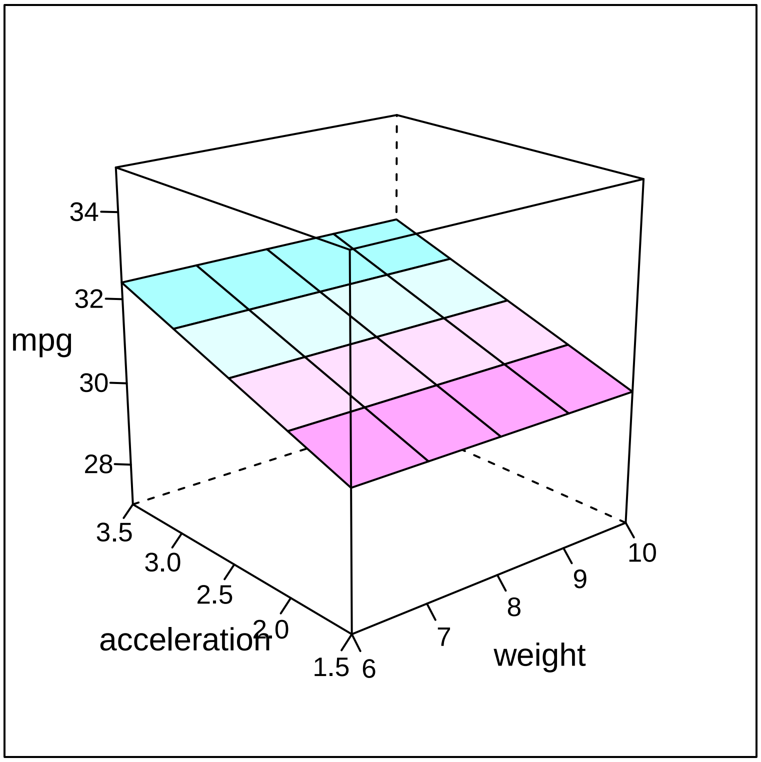
- ***ii.*** Show a density plot of the distribution of the 95% CI of the indirect effect.

```
ggplot()+
  aes(indirect)+
  geom_density()+
  geom_vline(xintercept=quantile(indirect, c(0.025, 0.975)), col='red', lty=2)+
  ggtitle('Distribution of 95% CI of the indirect effect')
```



Visualization

```
library(lattice)

# creating x, y grid of all possible points
g <- expand.grid(weight=6:10, acceleration = seq(1.5, 3.5, 0.5))
auto_regr <- lm(mpg~weight+acceleration, data=auto)
g$mpg <- predict(auto_regr, g)
auto_regr_intxn <- lm(mpg~
                        weight+
                        acceleration+
                        weight*acceleration,
                      data=auto)
g$mpg <- predict(auto_regr_intxn, g)

wireframe(mpg~weight*acceleration, data=g, zlim=c(27, 35),
          scales=list(arrows=FALSE), drape=TRUE, colorkey=FALSE,
          screen=list(z=40, x=-70))
```


34
32
30
28
mpg
3.5
3.0
2.5
2.0
1.5
acceleration
6
7
8
9
10
weight