HW10

Answers are in black text !

Question 1) Model fit is often determined by R2 so let's dig into what this perspective of model fit is all about.Download demo_simple_regression_sq.R from Canvas – it has a function that runs a regression simulation.This week, the simulation also reports R2 along with the other metrics from last week.

Scenario 1: Consider a very narrowly dispersed set of points that have a negative or positive steep slope

Scenario 2: Consider a widely dispersed set of points that have a negative or positive steep slope

Scenario 3: Consider a very narrowly dispersed set of points that have a negative or positive shallow slope

Scenario 4: Consider a widely dispersed set of points that have a negative or positive shallow slope

a). Let's dig into what regression is doing to compute model fit:

i. Plot Scenario 2, storing the returned points: pts <- interactive_regression_rsq()
ii. Run a linear model of x and y points to confirm the R2 value reported by the simulation
iii. Add line segments to the plot to show the regression residuals (errors) Get the values of y(regression line's estimates of y, given x):

```
y_hat <- regr$fitted.values
Addsegments:segments(ptsx, ptsy,ptsx, y_hat, col="red", lty="dotted")
```

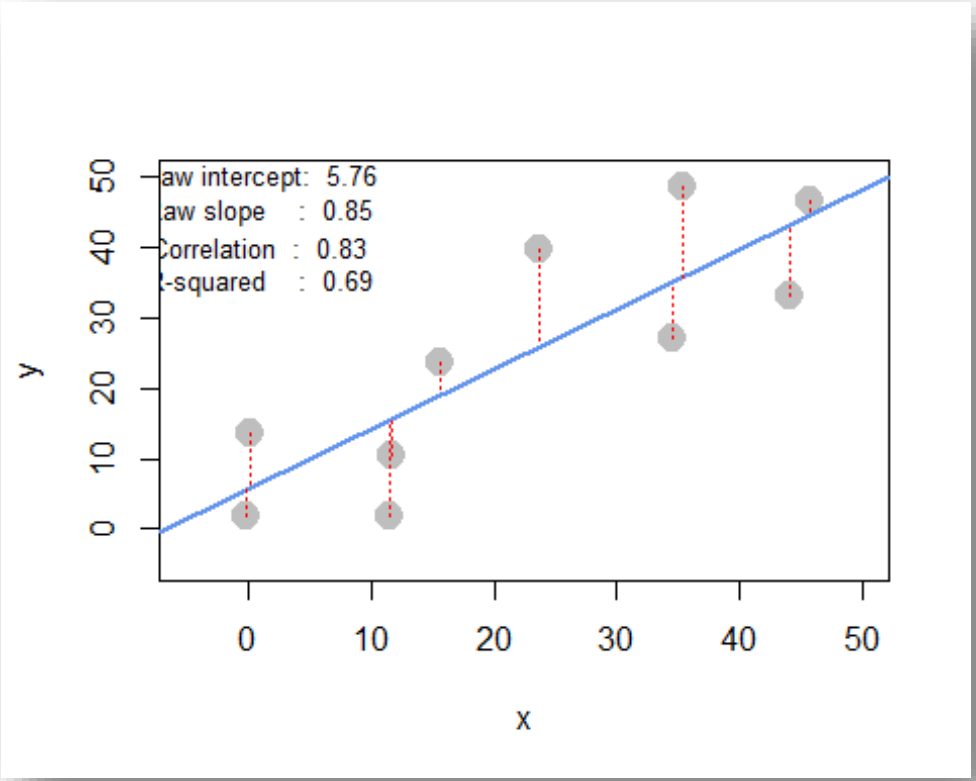
iv. Use only ptsx, ptsy, y_hat and mean(pts$y) to compute SSE, SSR and SST, and verify R2

```
#pts <- interactive_regression_rsq()
pts <- data.frame(x = c(-0.2042956,0.2491857,11.7373779,15.6675489,34.5626018,23.6790513,45.7484731, 35.318
4039,44.0857085,11.5862174), y=c(1.960052 ,13.654639 ,10.465206 ,23.648196 ,27.050258, 39.595361, 46.612113
 ,48.525773, 33.003866 , 1.747423))
regr <- lm(y~x ,data = pts)
```

```r
regr_summary <- summary(regr)
regr_summary
```

```
##
## Call:
## lm(formula = y ~ x, data = pts)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.8413  -7.3307  -0.7889   6.9099  13.7512
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.7629     5.5054   1.047  0.32580
## x             0.8481     0.2003   4.235  0.00286 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.23 on 8 degrees of freedom
## Multiple R-squared:  0.6915, Adjusted R-squared:  0.6529
## F-statistic: 17.93 on 1 and 8 DF,  p-value: 0.002859
```

```r
y_hat <- regr$fitted.values
plot(pts, xlim=c(-5,50), ylim=c(-5,50), pch=19, cex=2, col="gray")
abline(regr, lwd=2, col="cornflowerblue")
text(1, 50, paste(c("Raw intercept: ", round(regr$coefficients[1], 2)), collapse=" "), cex=0.80)
text(1, 45, paste(c("Raw slope    : ", round(regr$coefficients[2], 2)), collapse=" "), cex=0.80)
text(1, 40, paste(c("Correlation  : ", round(cor(pts$x, pts$y), 2)), collapse=" "), cex=0.80)
text(1, 35, paste(c("R-squared    : ", round(regr_summary$r.squared, 2)), collapse=" "), cex=0.80)
segments(pts$x, pts$y, pts$x, y_hat, col="red", lty="dotted")
```



The R2 reported by simulation is the same as regression summary (red text).

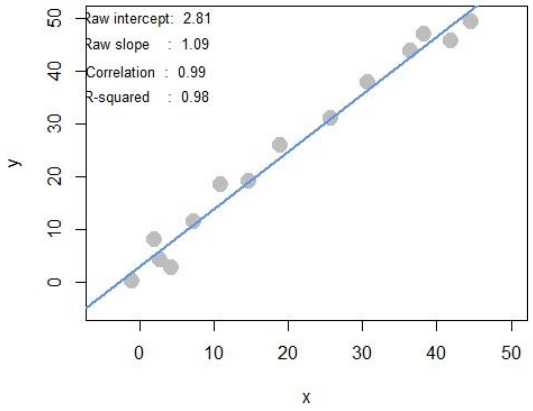
```
model_fit_error <- function(pts){
  regr <- lm(y~x ,data = pts)
  y_hat <- regr$fitted.values
  SSE <- sum((pts$y - y_hat)^2)
  SSR <- sum((y_hat - mean(pts$y))^2)
```

```
  SST <- sum((pts$y-mean(pts$y))^2)
  R_sq <- SSR/SST
  return(data.frame(SSE=SSE,SSR=SSR,SST=SST,R_square=R_sq))
}
model_fit_error(pts)
```

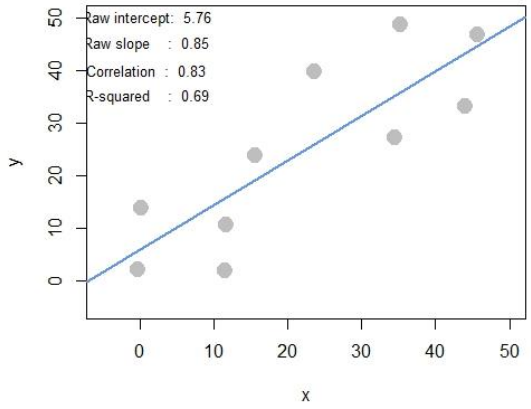
SSE <dbl>	SSR <dbl>	SST <dbl>	R_square <dbl>
837.2189	1876.546	2713.765	0.6914918

b). Comparing scenarios 1 and 2, which do we expect to have a stronger R2 ?

Scenarios 1 might have a stronger R2.



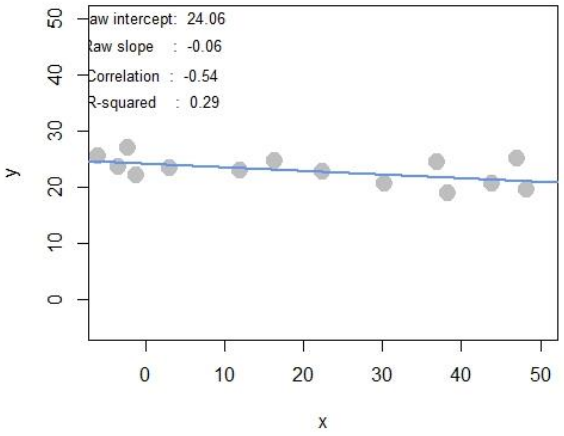
Scenarios 1



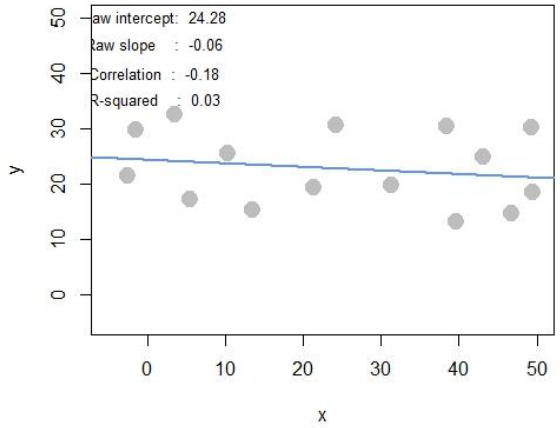
Scenarios 2

c). Comparing scenarios 3 and 4, which do we expect to have a stronger R2 ?

Scenarios 3 might have a stronger R2.



Scenarios3



Scenarios 4

d). Comparing scenarios 1 and 2, which do we expect has bigger/smaller SSE, SSR, and SST?

Scenarios 2 has bigger SSE, smaller SSR (depends on the slope) and bigger SST.

e). Comparing scenarios 3 and 4, which do we expect has bigger/smaller SSE, SSR, and SST?

Scenarios 4 has bigger SSE, smaller SSR (depends on the slope) and bigger SST.

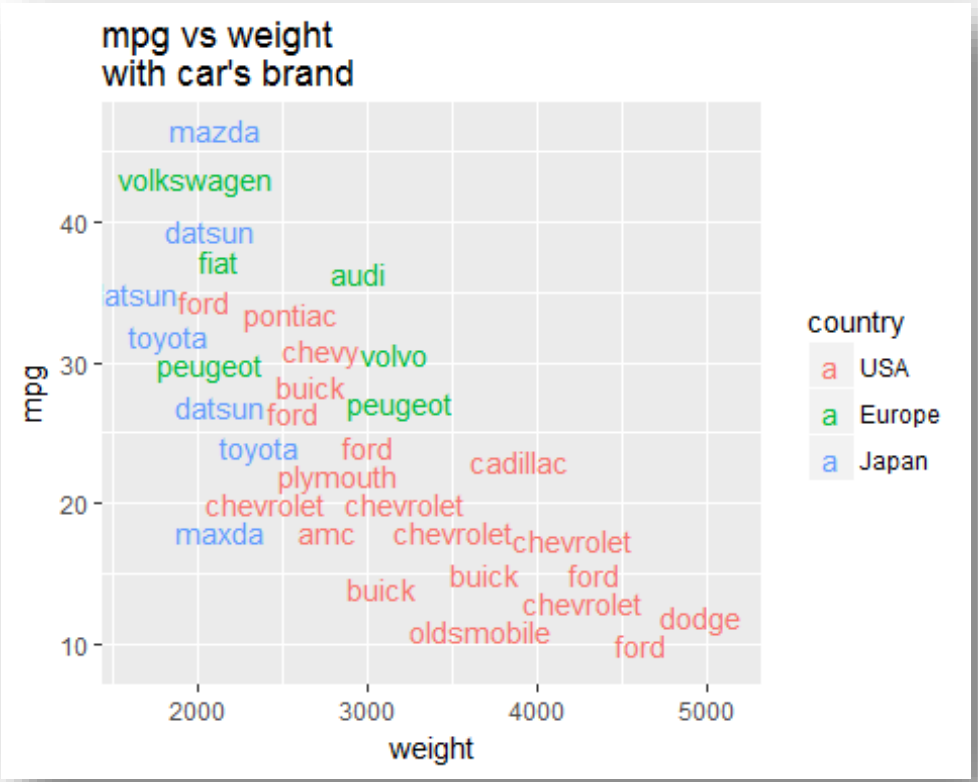
Question 2) We're going to take a look back at the heady days of car manufacturing, when American, Japanese, and European cars competed to rule the world. Take a look at a data set (auto-data.txt). We are interested in explaining what kind of cars have higher fuel efficiency (measured by mpg).

Variable description: 1. mpg: miles-per-gallon (dependent variable) 2. cylinders: cylinders in engine 3. displacement: size of engine 4. horsepower: power of engine 5. weight: weight of car 6. acceleration: acceleration ability of car 7. model_year: year model was released 8. origin: place car was designed (1: USA, 2: Europe, 3: Japan) 9. car_name: make and model names

a). Let's first try exploring this data and problem:

i. Visualize the data in any way you feel relevant (report only relevant/interesting ones)

```
#load daat
library(dplyr)

auto <- read.table("auto-data.txt", header=FALSE, na.strings = "?", stringsAsFactors = F)
names(auto) <- c("mpg", "cylinders", "displacement", "horsepower", "weight",
"acceleration", "model_year", "origin", "car_name")
auto <- auto %>% mutate(brand = unlist(strsplit(car_name," "))[1])
auto$brand <- as.character(lapply(auto$car_name,function(x){unlist(strsplit(x," "))[1]}))
auto$country <- factor(auto$origin,labels=c("USA","Europe","Japan"))
#visualization
library(ggplot2)
ggplot(data = auto,aes(x = weight,y=mpg, label = brand , color =country))+
  geom_text(check_overlap = TRUE)+
  ggtitle("mpg vs weight \nwith car's brand")
```



Here I used scatter plot to discover the relationship between mpg and weight. The colored words represent the car brands and where it was made. We can easily find out that american cars tend to be heavier and lower feul efficiency. The most important is there exists negative relationship between mpg and weight!

ii. Report a correlation table of all variables, rounding to two decimal places (in the cor(...) function, set use="pairwise.complete.obs" to handle missing values)

```r
cor_table <- cor(auto[1:8], use = "pairwise.complete.obs") %>% round(2)
cor_table
```

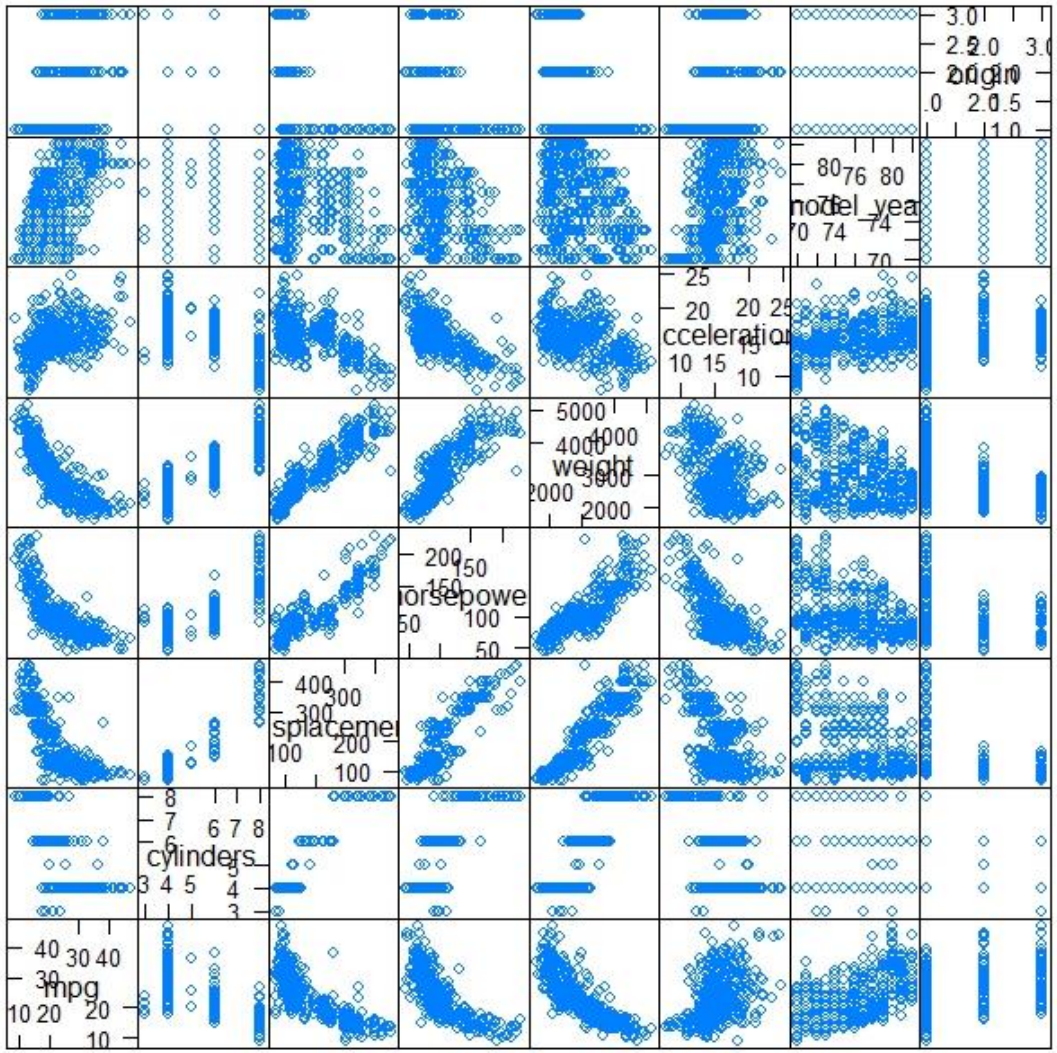
	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
mpg	1.00	-0.78	-0.80	-0.78	-0.83	0.42	0.58	0.56
cylinders	-0.78	1.00	0.95	0.84	0.90	-0.51	-0.35	-0.56
displacement	-0.80	0.95	1.00	0.90	0.93	-0.54	-0.37	-0.61
horsepower	-0.78	0.84	0.90	1.00	0.86	-0.69	-0.42	-0.46
weight	-0.83	0.90	0.93	0.86	1.00	-0.42	-0.31	-0.58
acceleration	0.42	-0.51	-0.54	-0.69	-0.42	1.00	0.29	0.21
model_year	0.58	-0.35	-0.37	-0.42	-0.31	0.29	1.00	0.18
origin	0.56	-0.56	-0.61	-0.46	-0.58	0.21	0.18	1.00

iii. From the visualizations and correlations, which variables seem to relate to mpg ?

According to the correlation table, **cylinders, displacement, horsepower, weight, model_year** seem to relate to mpg. Among them, weight might be the most related variables. Notice that most of the variables tend to have negative relationship with mpg!

iv. Which relationships might not be linear? (don't worry about linearity for rest of this HW)

```r
library(lattice)
splom(~auto[1:8])
```



Scatter Plot Matrix

Among those **relationship, displacement, horsepower, weight with mpg** might not be linear. From the above plots matrix, we can see that those variables construct a moon-liked shape insteat of straight line shape.

v. Are any of the independent variables highly correlated (r > 0.7) with others?

```
library(reshape2)
diag(cor_table) <- 0
cor_melt <- melt(cor_table)
hight_cor <- cor_melt[order(abs(cor_melt$value),decreasing = T) & abs(cor_melt$value) >0.7,]

#### eliminate the same combination of variable
#sort two variable by first character order
hight_cor[1:2] <- t( apply(hight_cor[1:2], 1, sort) )
#eliminate the same variable combination
hight_cor[!duplicated(hight_cor[1:2]),]
```

Var1 <chr>	Var2 <chr>	value <dbl>
cylinders	mpg	-0.78
displacement	mpg	-0.80
horsepower	mpg	-0.78
mpg	weight	-0.83
cylinders	displacement	0.95
cylinders	horsepower	0.84
cylinders	weight	0.90
displacement	horsepower	0.90
displacement	weight	0.93
horsepower	weight	0.86

The above table shows the high correlated variables pair.

b). Let's try an ordinary linear regression, where mpg is dependent upon all other suitable variables (Note: origin is categorical with three levels, so use factor(origin) in lm(...) to split it into two dummy variables)

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

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + model_year + factor(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 ***
## 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
```

i. Which factors have a 'significant' effect on mpg at 1% significance?

Intercept, displacement, weight, model_year, and origin have significant effects on mpg at 1% significance.

ii. Looking at the coefficients, is it possible to determine which independent variables are the most effective at increasing mpg? If so, which ones, and if not, why not? (hint: units!)

Nope! Because those variable are in different scales, it's unreasonable to compare their magnitude directly. We should standardize them before constructing the regression model.

c). Let's try to resolve some of the issues with our regression model above.

i. Create fully standardized regression results: are these values easier to interpret? (note: consider if you should standardize origin)

```
auto_sd <- cbind(scale(auto[1:7]),auto$origin)
colnames(auto_sd) <- colnames(auto[1:8])
auto_sd <- as.data.frame(auto_sd)
regr_sd <- lm(mpg ~ cylinders+displacement+horsepower+weight+acceleration+model_year+factor(origin)
           ,data = auto_sd)
summary(regr_sd)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + model_year + factor(origin), data = auto_sd)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.15270 -0.26593 -0.01257  0.25404  1.70942
##
## Coefficients:
```

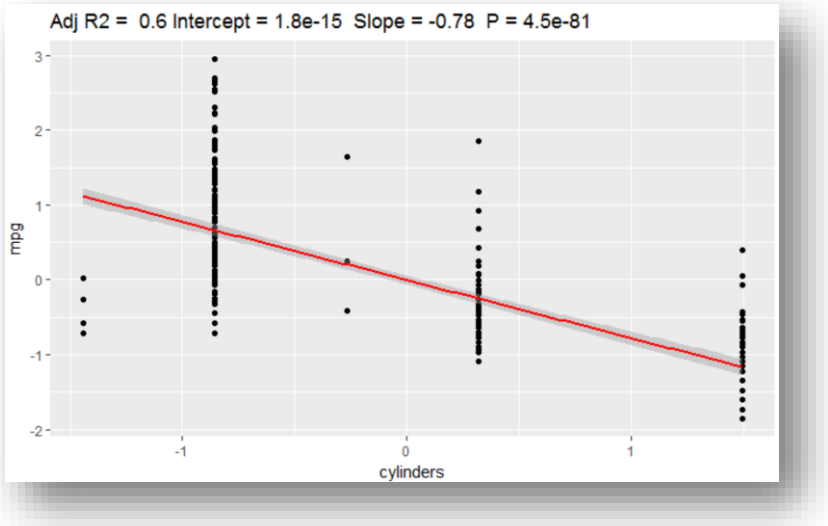
```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -0.13323    0.03174  -4.198 3.35e-05 ***
## cylinders       -0.10658    0.06991  -1.524  0.12821
## displacement     0.31989    0.10210   3.133  0.00186 **
## horsepower      -0.08955    0.06751  -1.326  0.18549
## weight          -0.72705    0.07098 -10.243  < 2e-16 ***
## acceleration     0.02791    0.03465   0.805  0.42110
## model_year       0.36760    0.02450  15.005  < 2e-16 ***
## factor(origin)2  0.33649    0.07247   4.643 4.72e-06 ***
## factor(origin)3  0.36505    0.07072   5.162 3.93e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 0.423 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
```

After standarization, it make it earilier to interpret the coefficient. According the summary report, we can find out that **weight** is the most effective at increasing mpg. (e.g., by decreasing the **weight** of car, **mpg** might greatly increase.)

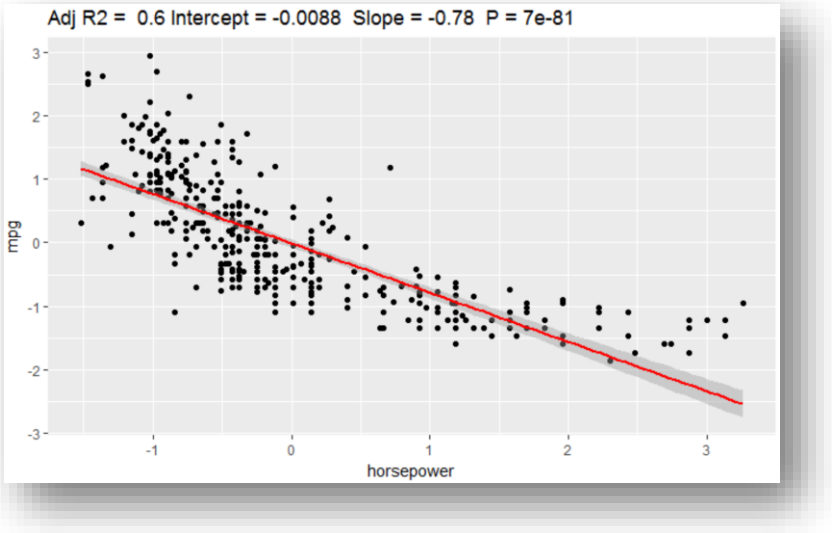
ii. Regress mpg over each nonsignificant independent variable, individually. Which ones are significant if we regress mpg over them individually?

```r
#function to create regression plot
ggplotRegression <- function (fit) {
require(ggplot2)
ggplot(fit$model, aes_string(x = names(fit$model)[2], y = names(fit$model)[1])) +
  geom_point() +
  stat_smooth(method = "lm", col = "red") +
  labs(title = paste("Adj R2 = ",signif(summary(fit)$adj.r.squared, 2),
                     "Intercept =",signif(fit$coef[[1]],2 ),
                     " Slope =",signif(fit$coef[[2]], 2),
                     " P =",signif(summary(fit)$coef[2,4], 2)))
```

```
}
ggplotRegression(lm(mpg ~ cylinders,data = auto_sd))
```



```r
ggplotRegression(lm(mpg ~ horsepower,data = auto_sd))
```

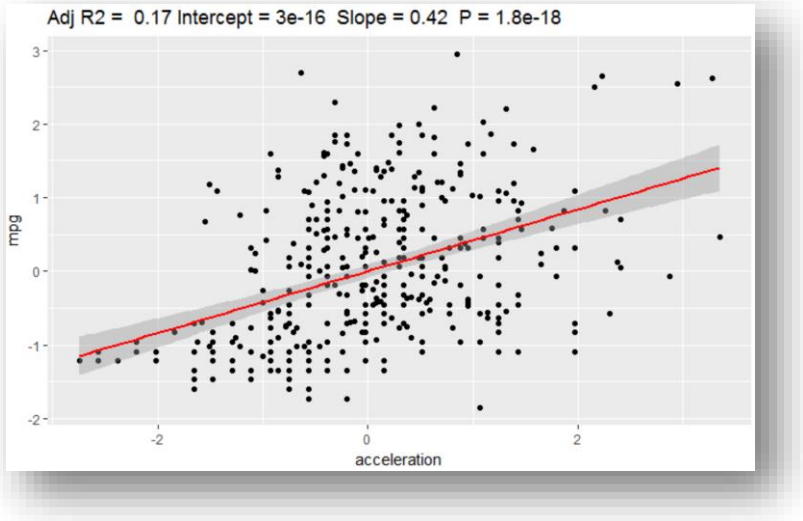



```r
summary(lm(mpg ~ acceleration,data = auto_sd))
```

```
##
## Call:
## lm(formula = mpg ~ acceleration, data = auto_sd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3039 -0.7210 -0.1589  0.6087  2.9672
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.004e-16  4.554e-02   0.000        1
## acceleration 4.203e-01  4.560e-02   9.217   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9085 on 396 degrees of freedom
## Multiple R-squared:  0.1766, Adjusted R-squared:  0.1746
## F-statistic: 84.96 on 1 and 396 DF,  p-value: < 2.2e-16
```

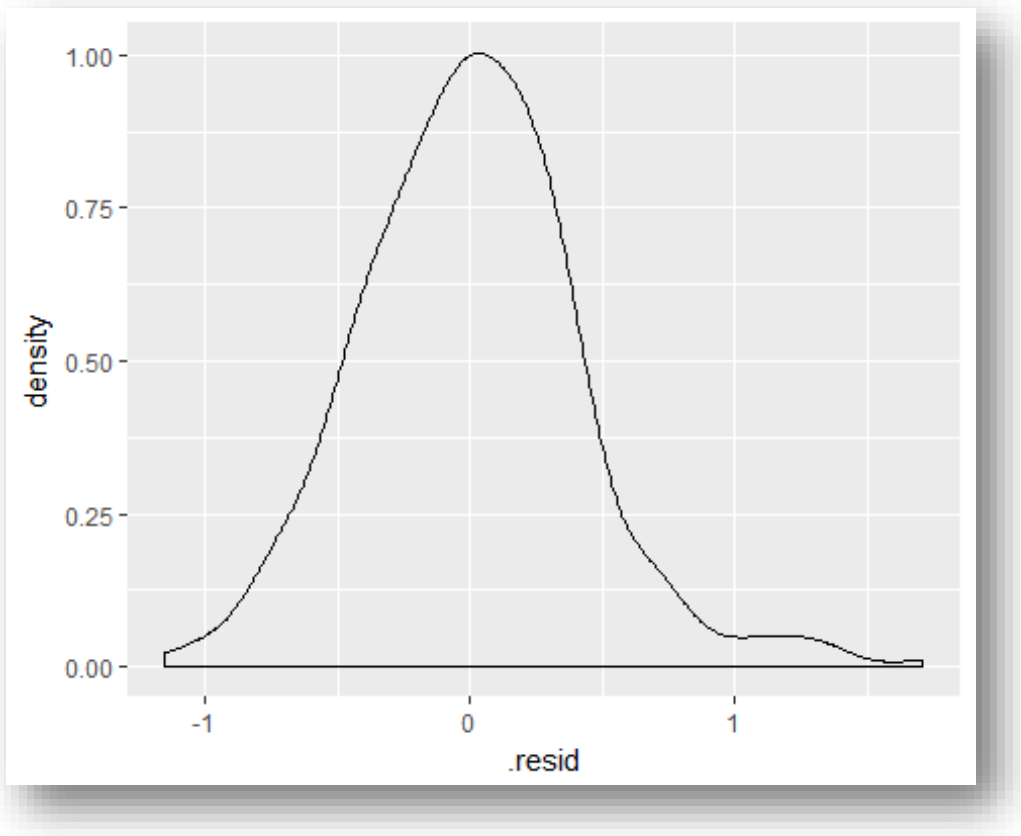
```
ggplotRegression(lm(mpg ~ acceleration,data = auto_sd))
```



The above figures show the regression line for three nonsignificant independent variable, **cylinders, horsepower, acceleration**. According to the p-value of each regression, all of them are significant if we regress mpg over them individually.

iii. Plot the density of the residuals: are they normally distributed and centered around zero? (hint: get the residuals of a linear model, e.g. regr <- lm(...) , using regr$residuals

```
regr_sdf <- fortify(regr_sd)
ggplot(regr_sdf,aes(.resid))+
  geom_density()
```



Yes, they're normally distributed and centered around zero.