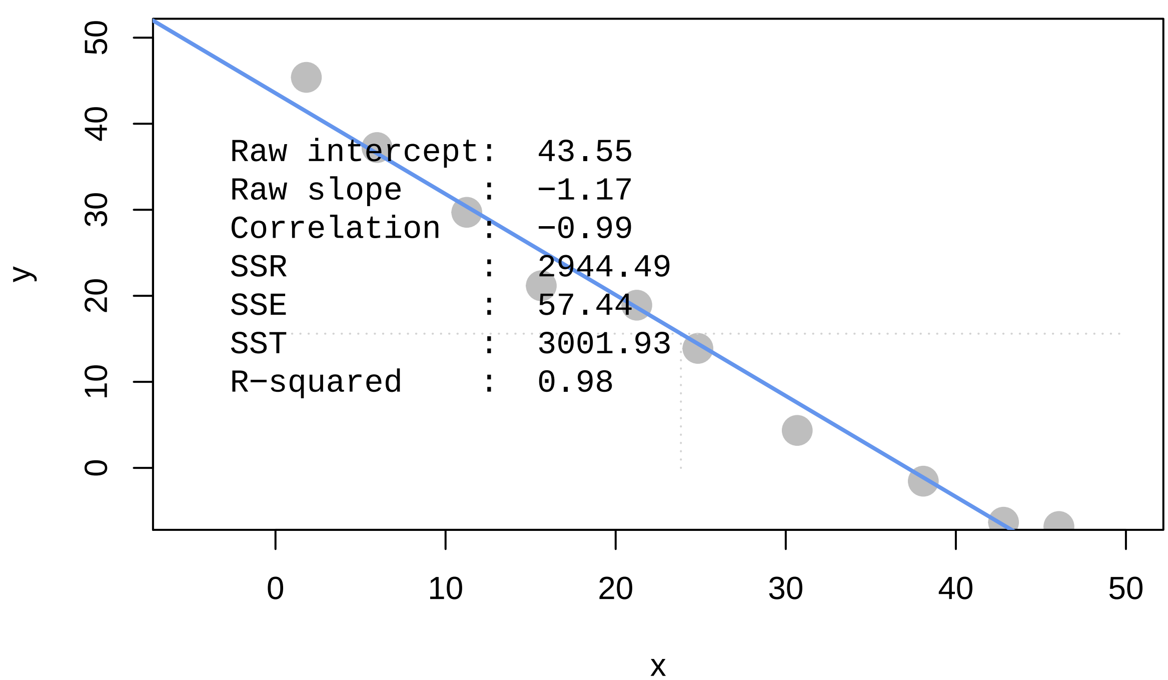
HW10

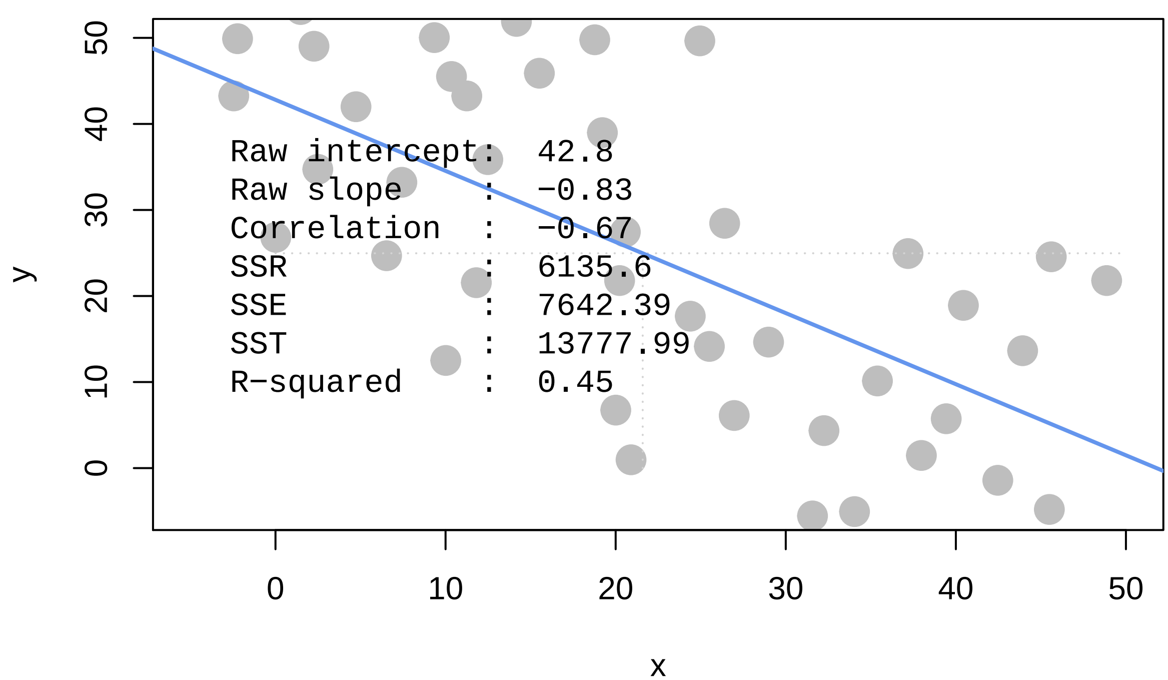
107070008

Question 1) Download demo_simple_regression_rsq.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.

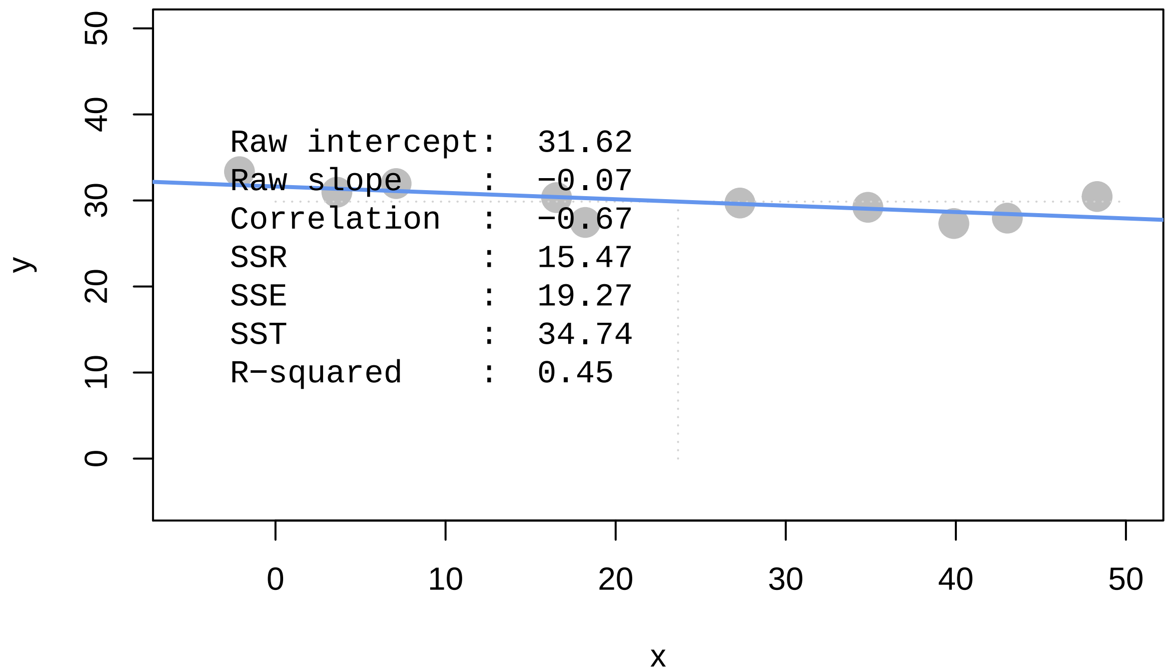
```
source("demo_simple_regression_rsq.r")
s1 <- read.table("./s1.txt")
plot_regr_rsq(s1)
```



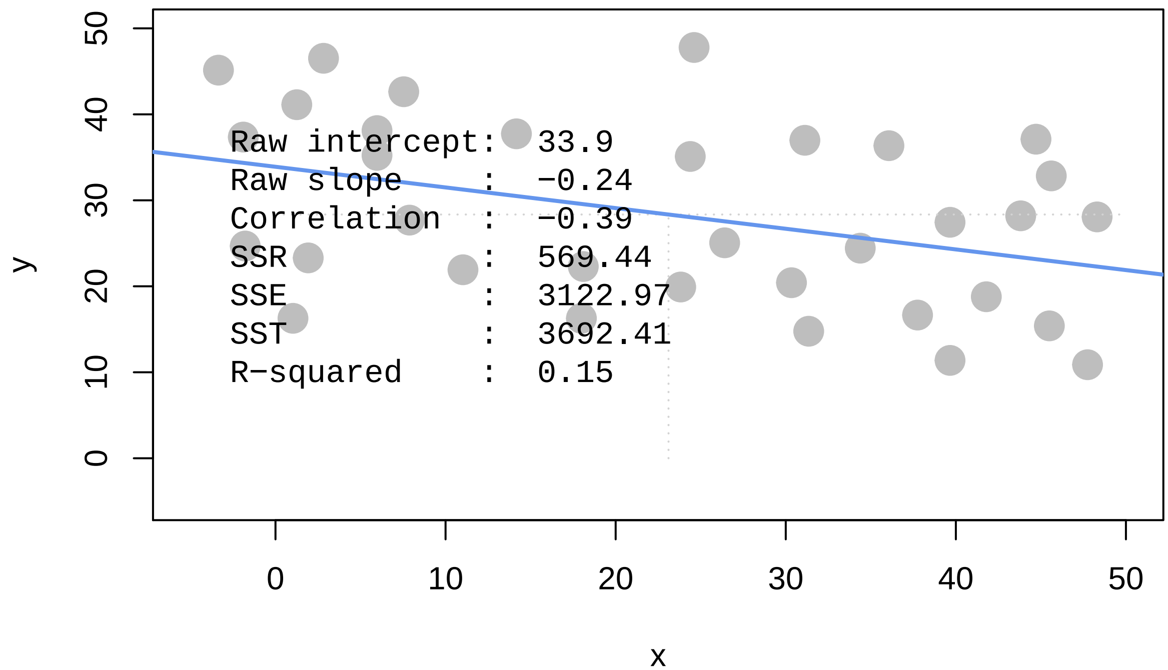

```
s2 <- read.table("./s2.txt")
plot_regr_rsq(s2)
```




```
s3 <- read.table("./s3.txt")
plot_regr_rsq(s3)
```




```r
s4 <- read.table("./s4.txt")
plot_regr_rsq(s4)
```



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

s1 has a stronger R squared.

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

s3 has a stronger R squared.

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

s2 has a larger SSR, larger SSE, larger SST.

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

s4 has a larger SSR, larger SSE, larger SST.

Question 2) Let's perform regression ourselves on the programmer_salaries.txt dataset we saw in class.

a. First, use the lm() function to estimate the model Salary ~ Experience + Score + Degree

```
salaries <- read.csv("programmer_salaries.txt", sep="\t")
salaries_reg <- lm(Salary ~ Experience + Score + Degree, data = salaries)
summary(salaries_reg)
```

```
##
## Call:
## lm(formula = Salary ~ Experience + Score + Degree, data = salaries)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8963 -1.7290 -0.3375  1.9699  5.0480
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.9448     7.3808   1.076   0.2977
## Experience    1.1476     0.2976   3.856   0.0014 **
## Score         0.1969     0.0899   2.191   0.0436 *
## Degree        2.2804     1.9866   1.148   0.2679
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.396 on 16 degrees of freedom
## Multiple R-squared:  0.8468, Adjusted R-squared:  0.8181
## F-statistic: 29.48 on 3 and 16 DF,  p-value: 9.417e-07
```

```
salaries_reg$fitted.values[1:5]
```

```
##        1        2        3        4        5
## 27.89626 37.95204 26.02901 32.11201 36.34251
```

```
salaries_reg$residuals[1:5]
```

```
##          1          2          3          4          5
## -3.8962605  5.0479568 -2.3290112  2.1879860 -0.5425072
```

b. Use only linear algebra (and the geometric view of regression) to estimate the regression yourself:

i. Create an X matrix that has a first column of 1s followed by columns of the independent variables (only show the code)

```r
X_1 <- t(matrix(1, ncol = 20))
X <- cbind(X_1, salaries$Experience, salaries$Score, salaries$Degree)
X
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]    1    4   78    0
##  [2,]    1    7  100    1
##  [3,]    1    1   86    0
##  [4,]    1    5   82    1
##  [5,]    1    8   86    1
##  [6,]    1   10   84    1
##  [7,]    1    0   75    0
##  [8,]    1    1   80    0
##  [9,]    1    6   83    0
## [10,]    1    6   91    1
## [11,]    1    9   88    1
## [12,]    1    2   73    0
## [13,]    1   10   75    1
## [14,]    1    5   81    0
## [15,]    1    6   74    0
## [16,]    1    8   87    1
## [17,]    1    4   79    0
## [18,]    1    6   94    1
## [19,]    1    3   70    0
## [20,]    1    3   89    0
```

ii. Create a y vector with the Salary values (only show the code)

```r
y <- salaries$Salary
y
```

```
##  [1] 24.0 43.0 23.7 34.3 35.8 38.0 22.2 23.1 30.0 33.0 38.0 26.6 36.2 31.6 29.0
## [16] 34.0 30.1 33.9 28.2 30.0
```

iii. Compute the beta_hat vector of estimated regression coefficients (show the code and values)

```r
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y
beta_hat
```

```
##          [,1]
## [1,] 7.944849
## [2,] 1.147582
## [3,] 0.196937
## [4,] 2.280424
```

iv. Compute a y_hat vector of estimated y values, and a res vector of residuals (show the code and the first 5 values of y_hat and res)

```r
y_hat <- X %*% beta_hat
y_hat
```

```
##           [,1]
##  [1,] 27.89626
##  [2,] 37.95204
##  [3,] 26.02901
##  [4,] 32.11201
##  [5,] 36.34251
##  [6,] 38.24380
##  [7,] 22.71512
##  [8,] 24.84739
##  [9,] 31.17611
## [10,] 35.03203
## [11,] 37.88396
## [12,] 24.61641
## [13,] 36.47136
## [14,] 29.63465
## [15,] 29.40368
## [16,] 36.53944
## [17,] 28.09320
## [18,] 35.62284
## [19,] 25.17318
## [20,] 28.91499
```

```
res <- y - y_hat
res
```

```
##             [,1]
##  [1,] -3.8962605
##  [2,]  5.0479568
##  [3,] -2.3290112
##  [4,]  2.1879860
##  [5,] -0.5425072
##  [6,] -0.2437966
##  [7,] -0.5151227
##  [8,] -1.7473893
##  [9,] -1.1761089
## [10,] -2.0320286
## [11,]  0.1160371
## [12,]  1.9835879
## [13,] -0.2713638
## [14,]  1.9653468
## [15,] -0.4036760
## [16,] -2.5394441
## [17,]  2.0068025
## [18,] -1.7228396
## [19,]  3.0268171
## [20,]  1.0850144
```

v. Using only the results from (i) – (iv), compute SSR, SSE and SST (show the code and values)

```
SSE <- sum((y - y_hat)^2)
SSE
```

```
## [1] 91.88949
```

```r
RSquared <- cor(y, y_hat)^2
SST <- SSE/(1-RSquared)
SST
```

```
##          [,1]
## [1,] 599.7855
```

```r
SSR <- SST - SSE
SSR
```

```
##         [,1]
## [1,] 507.896
```

c. Compute R2 for in two ways, and confirm you get the same results (show code and values):

i. Use any combination of SSR, SSE, and SST

```r
RSquared_i <- SSR/SST
RSquared_i
```

```
##           [,1]
## [1,] 0.8467961
```

ii. Use the squared correlation of vectors y and y hat

```r
RSquared_ii <- cor(y, y_hat)^2
RSquared_ii
```

```
##           [,1]
## [1,] 0.8467961
```

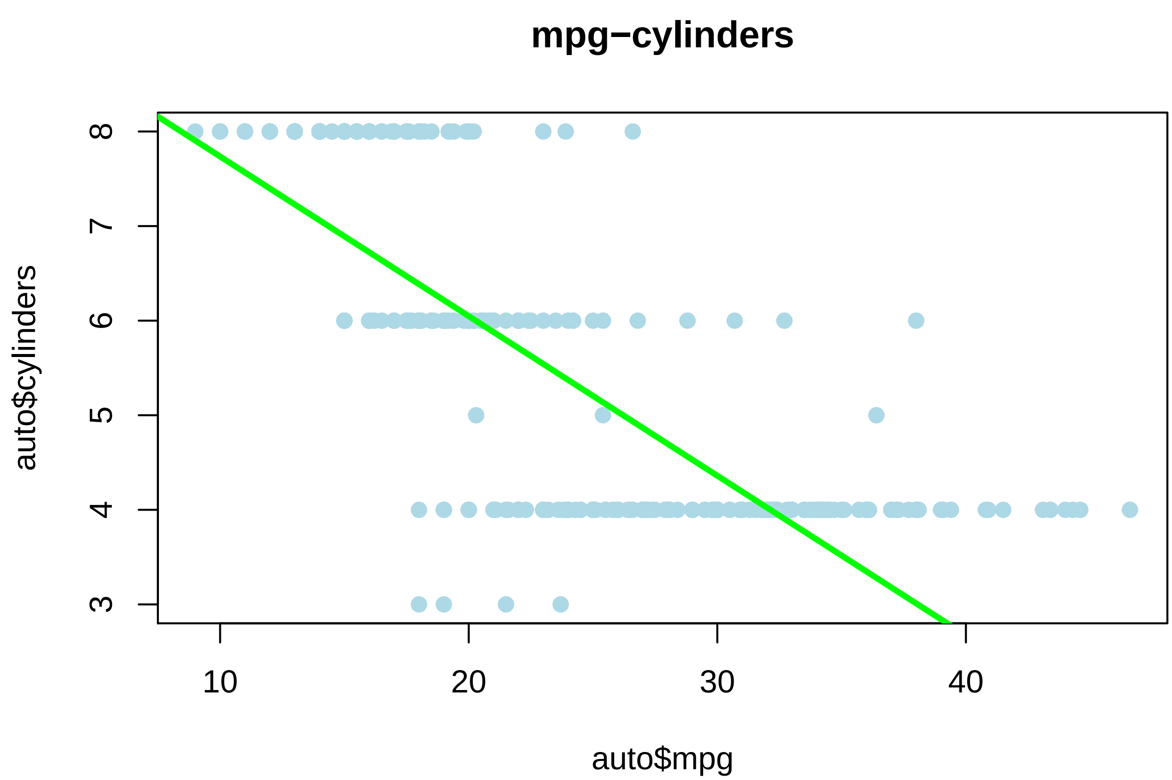
Question 3) We're going to take a look back at the early heady days of global car manufacturing, when American, Japanese, and European cars competed to rule the world.

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

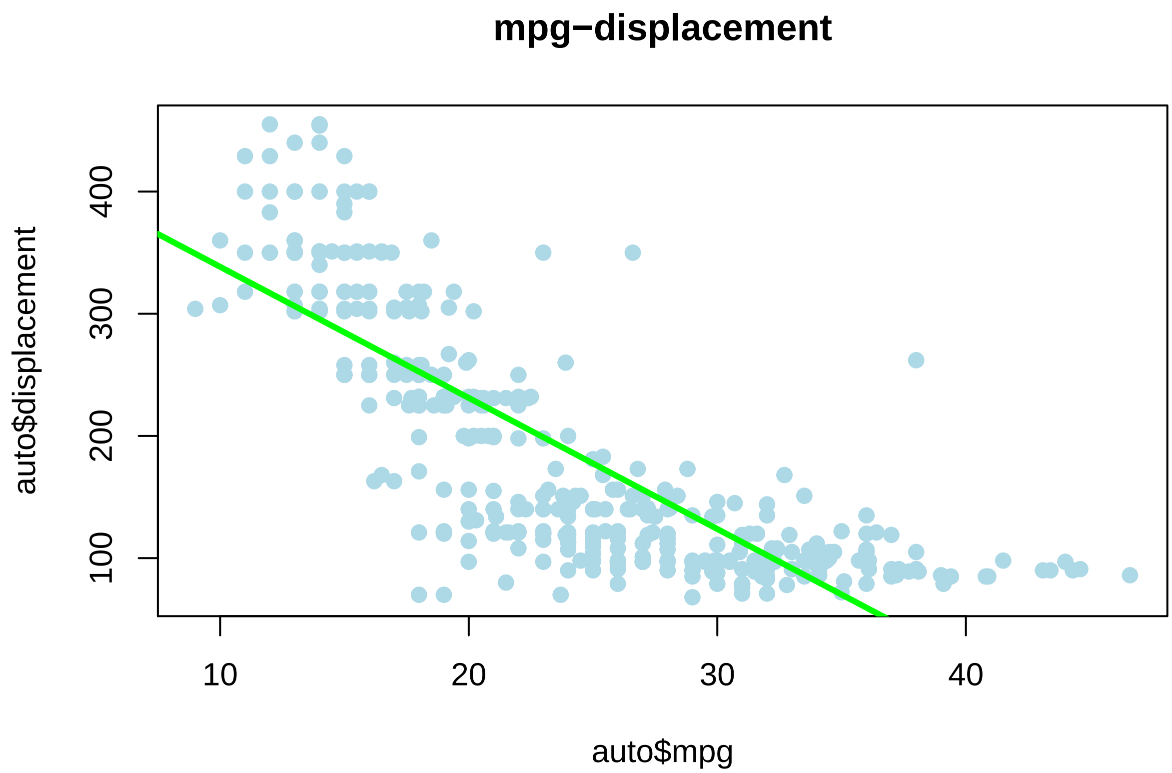
a. Let's first try exploring this data and problem:

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

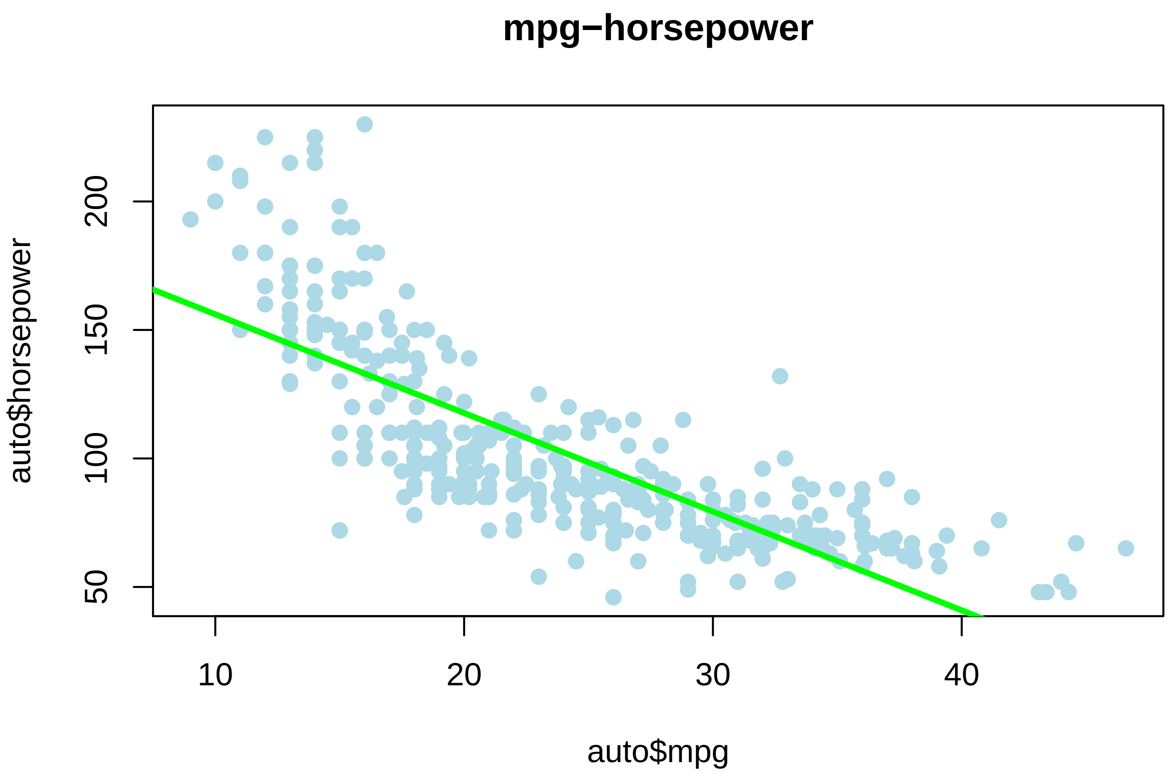
```r
plot(auto$mpg, auto$cylinders, pch = 19, col = "lightblue", main = "mpg-cylinders")
abline(lm(auto$cylinders ~ auto$mpg), col = "green", lwd = 3)
```



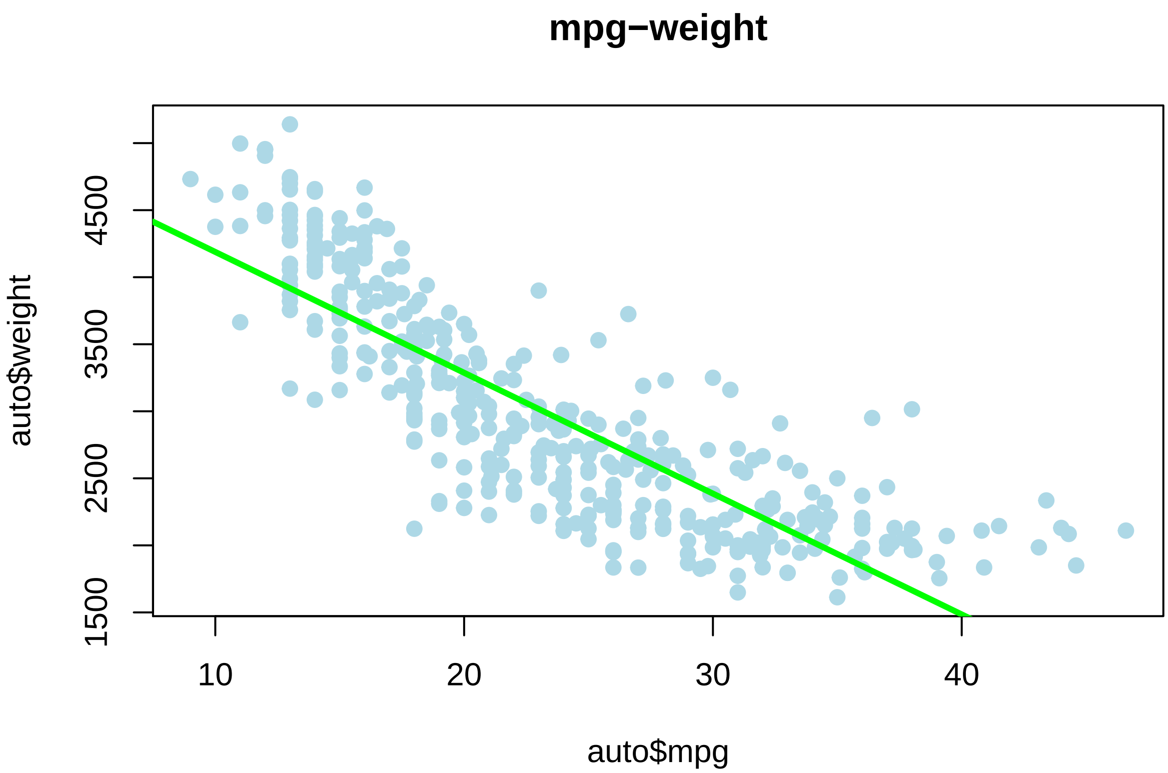

```r
plot(auto$mpg, auto$displacement, pch = 19, col = "lightblue", main = "mpg-displacement")
abline(lm(auto$displacement ~ auto$mpg), col = "green", lwd = 3)
```



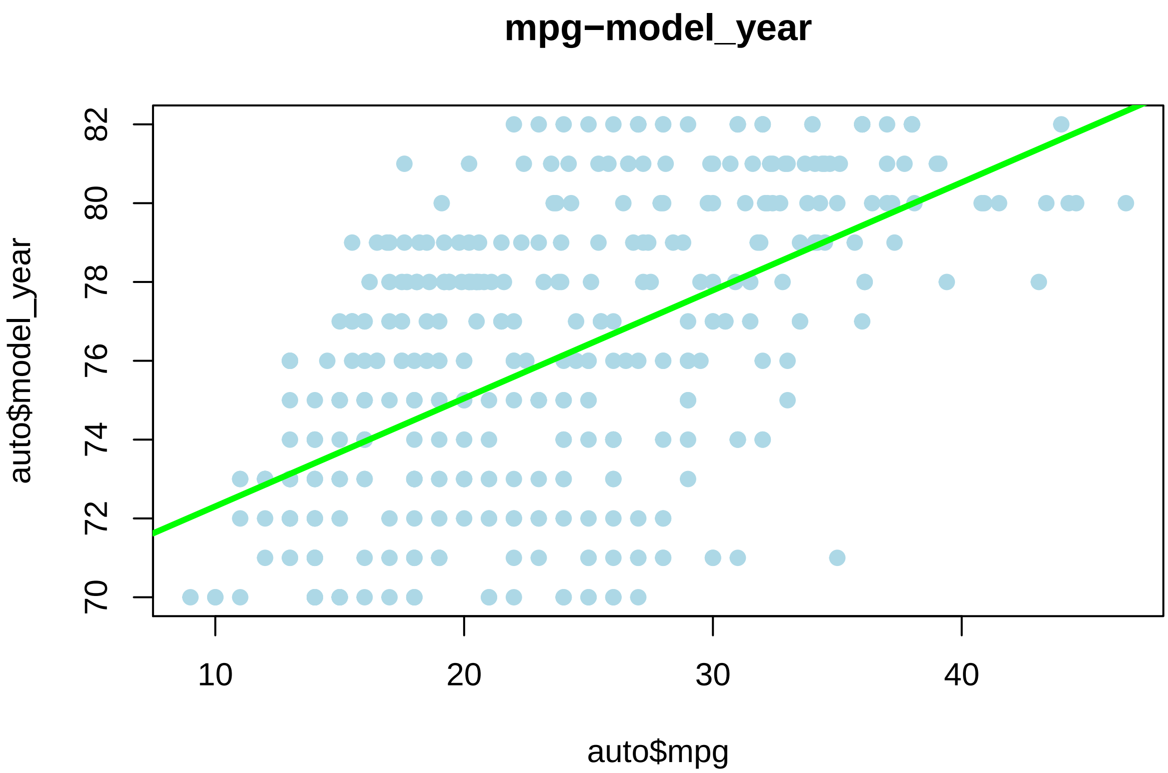

```
plot(auto$mpg, auto$horsepower, pch = 19, col = "lightblue", main = "mpg-horsepower")
abline(lm(auto$horsepower ~ auto$mpg), col = "green", lwd = 3)
```



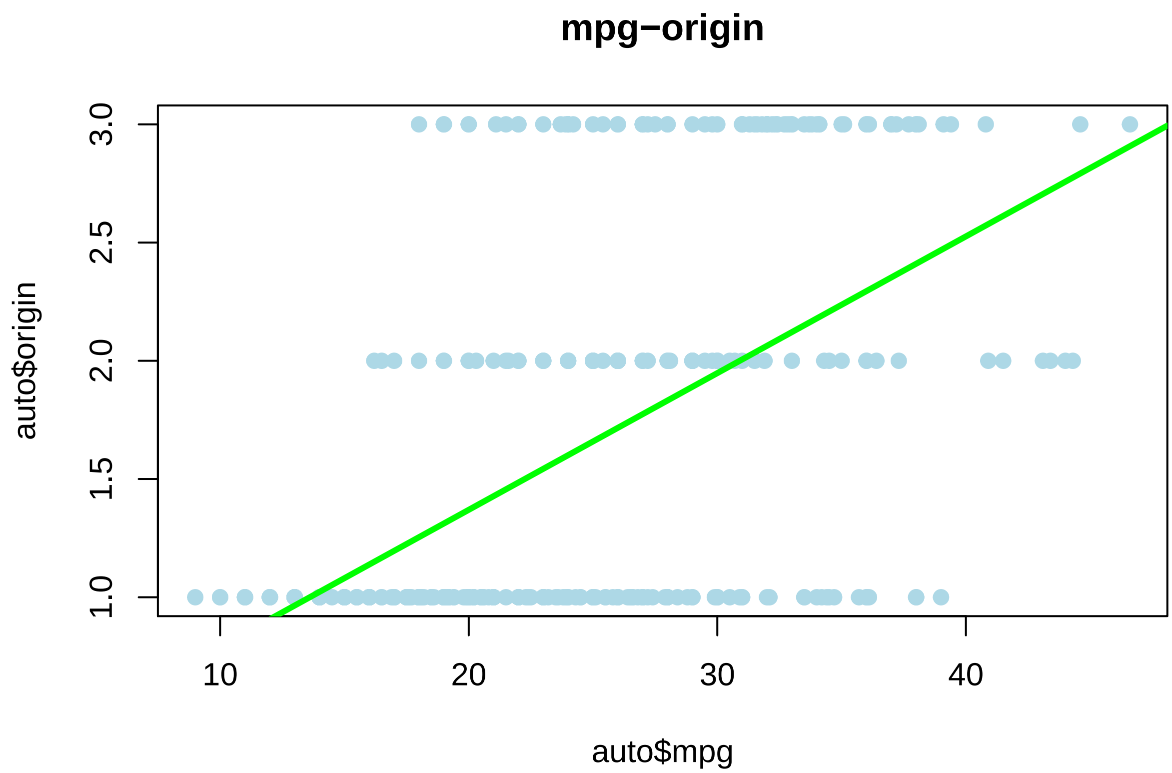

```
plot(auto$mpg, auto$weight, pch = 19, col = "lightblue", main = "mpg-weight")
abline(lm(auto$weight ~ auto$mpg), col = "green", lwd = 3)
```




```
plot(auto$mpg, auto$model_year, pch = 19, col = "lightblue", main = "mpg-model_year")
abline(lm(auto$model_year ~ auto$mpg), col = "green", lwd = 3)
```




```r
plot(auto$mpg, auto$origin, pch = 19, col = "lightblue", main = "mpg-origin")
abline(lm(auto$origin ~ auto$mpg), col = "green", lwd = 3)
```



ii. Report a correlation table of all variables, rounding to two decimal places

```
auto_m <- data.matrix(auto)
res <- cor(auto_m, use="pairwise.complete.obs")
round(res, 2)
```

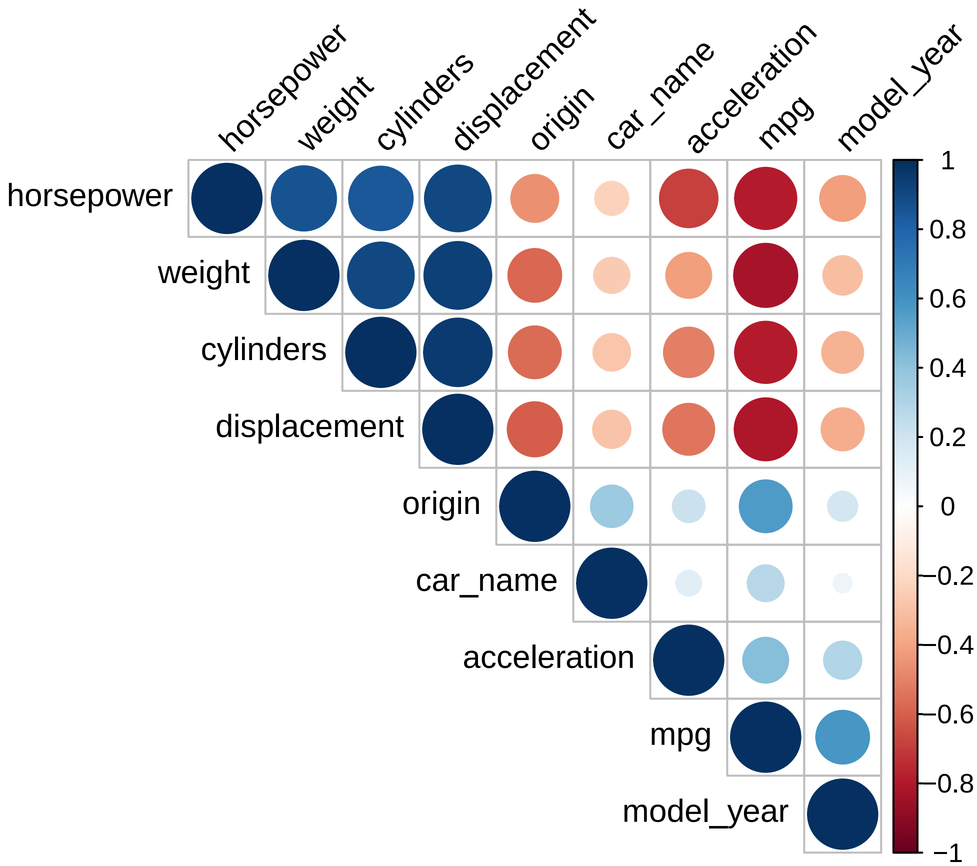
```
##                mpg cylinders displacement horsepower weight acceleration
## mpg           1.00     -0.78        -0.80      -0.78  -0.83         0.42
## cylinders    -0.78      1.00         0.95       0.84   0.90        -0.51
## displacement -0.80      0.95         1.00       0.90   0.93        -0.54
## horsepower   -0.78      0.84         0.90       1.00   0.86        -0.69
## weight       -0.83      0.90         0.93       0.86   1.00        -0.42
## acceleration  0.42     -0.51        -0.54      -0.69  -0.42         1.00
## model_year    0.58     -0.35        -0.37      -0.42  -0.31         0.29
## origin        0.56     -0.56        -0.61      -0.46  -0.58         0.21
## car_name      0.27     -0.28        -0.29      -0.23  -0.26         0.13
##              model_year origin car_name
## mpg                0.58   0.56     0.27
## cylinders         -0.35  -0.56    -0.28
## displacement      -0.37  -0.61    -0.29
## horsepower        -0.42  -0.46    -0.23
## weight            -0.31  -0.58    -0.26
## acceleration       0.29   0.21     0.13
## model_year         1.00   0.18     0.07
## origin             0.18   1.00     0.36
## car_name           0.07   0.36     1.00
```

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

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
corrplot(round(res, 2), type = "upper", order = "hclust", tl.col = "black", tl.srt = 45)
```




```
displacement, weight, cylinders, horsepower
```

iv. Which relationships might not be linear? `Name`

v. Are there any pairs of independent variables that are highly correlated ($r > 0.7$)? `(horsepower, weight) (horsepower, cylinders) (horsepower, displacement) (weight, cylinders) (weight, displacement) (cylinders, displacement)`

b. Let's create a linear regression model where mpg is dependent upon all other suitable variables

i. Which independent variables have a 'significant' relationship with mpg at 1% significance?

```
auto_m2 <- as.data.frame(auto_m)
mpg_rgm <- lm(mpg ~ cylinders + displacement + horsepower +
                weight + acceleration + model_year +
                factor(origin), data = auto_m2)
summary(mpg_rgm)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + model_year + factor(origin), data = auto_m2)
##
## 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
```

```
displacement
```

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?

```
mpg_rgm$coefficients
```

```
##     (Intercept)       cylinders    displacement      horsepower          weight
##    -17.954602067    -0.489709424     0.023978644    -0.018183464    -0.006710384
##    acceleration      model_year factor(origin)2 factor(origin)3
##     0.079103036     0.777026939     2.630002360     2.853228228
```

```
The abs number of cylinders's coefficient is the biggest, so it will efficiently increase
mpg the most.
```

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

i. Create fully standardized regression results: are these slopes easier to compare?

```
auto_ <- auto_m[complete.cases(auto_m),]
auto_ <- auto_[, -8]
col_mean <- apply(auto_, 2, mean)
mean_matrix <- t(replicate(nrow(auto_), col_mean))
```

```r
col_sd <- apply(auto_, 2, sd)
sd_matrix <- t(replicate(nrow(auto_), col_sd))
auto_std <- (auto_ - mean_matrix)/sd_matrix

mpg_reg_all <- lm(mpg ~ cylinders + displacement + horsepower +
                    weight + acceleration + model_year + car_name, as.data.frame(auto_std))
summary(mpg_reg_all)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + model_year + car_name, data = as.data.frame(auto_std))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.09971 -0.29948 -0.01151  0.24776  1.83602
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.044e-15  2.203e-02   0.000  1.00000
## cylinders    -7.130e-02  7.192e-02  -0.991  0.32219
## displacement  1.377e-01  9.852e-02   1.398  0.16294
## horsepower   -5.748e-03  6.763e-02  -0.085  0.93231
## weight       -7.511e-01  7.237e-02 -10.378  < 2e-16 ***
## acceleration  3.241e-02  3.575e-02   0.906  0.36525
## model_year    3.579e-01  2.462e-02  14.534  < 2e-16 ***
## car_name      6.573e-02  2.314e-02   2.840  0.00475 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4361 on 384 degrees of freedom
## Multiple R-squared:  0.8132, Adjusted R-squared:  0.8098
## F-statistic: 238.8 on 7 and 384 DF,  p-value: < 2.2e-16
```

Yes, in standardize type, it is easier to compare.

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

```r
summary(lm(mpg ~ factor(origin) , as.data.frame(auto_m)))
```

```
##
## Call:
## lm(formula = mpg ~ factor(origin), data = as.data.frame(auto_m))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.451  -5.083  -1.034   3.649  18.916
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      20.0835     0.4056  49.517   <2e-16 ***
```

```
## factor(origin)2   7.8079     0.8658   9.018   <2e-16 ***
## factor(origin)3  10.3671     0.8264  12.544   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.4 on 395 degrees of freedom
## Multiple R-squared:  0.3329, Adjusted R-squared:  0.3295
## F-statistic: 98.54 on 2 and 395 DF,  p-value: < 2.2e-16
```

```
summary(lm(mpg ~ car_name , as.data.frame(auto_m)))
```

```
##
## Call:
## lm(formula = mpg ~ car_name, data = as.data.frame(auto_m))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.8603  -5.9061  -0.9038   5.5417  22.4287
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.936764   0.735372  27.111  < 2e-16 ***
## car_name     0.023924   0.004221   5.668 2.79e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.526 on 396 degrees of freedom
## Multiple R-squared:  0.07504,    Adjusted R-squared:  0.07271
## F-statistic: 32.13 on 1 and 396 DF,  p-value: 2.785e-08
```

```
summary(lm(mpg ~ acceleration , as.data.frame(auto_m)))
```

```
##
## Call:
## lm(formula = mpg ~ acceleration, data = as.data.frame(auto_m))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.007  -5.636  -1.242   4.758  23.192
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.9698     2.0432   2.432   0.0154 *
## acceleration   1.1912     0.1292   9.217   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.101 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
```

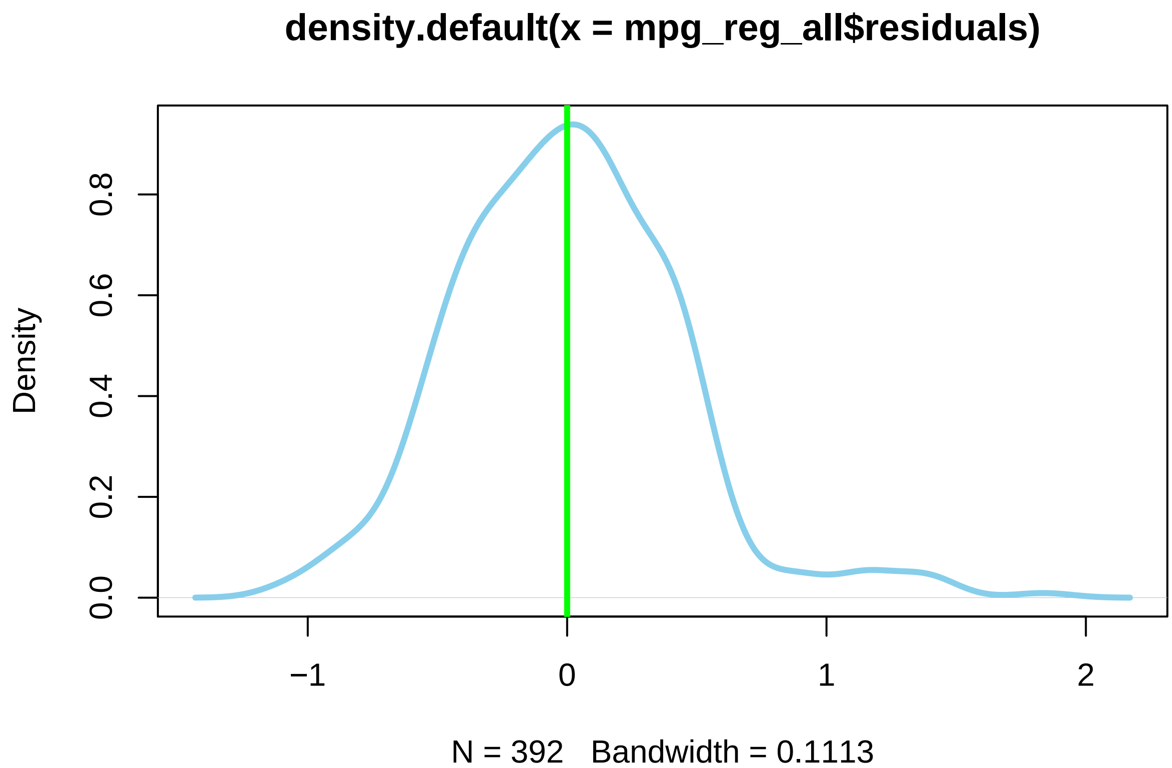
```
summary(lm(mpg ~ model_year , as.data.frame(auto_m)))
```

```
##
## Call:
## lm(formula = mpg ~ model_year, data = as.data.frame(auto_m))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.024  -5.451  -0.390   4.947  18.200
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -69.55560    6.58911  -10.56   <2e-16 ***
## model_year    1.22445    0.08659   14.14   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.379 on 396 degrees of freedom
## Multiple R-squared:  0.3356, Adjusted R-squared:  0.3339
## F-statistic:   200 on 1 and 396 DF,  p-value: < 2.2e-16
```

```
origin is the most significant.
```

iii. Plot the density of the residuals: are they normally distributed and centered around zero?

```
plot(density(mpg_reg_all$residuals), col = "skyblue", lwd = 3)
abline(v=mean(mpg_reg_all$residuals), col = "green", lwd = 3)
```




```
Yes, they are normally distributed and centered around zero
```