Jin Kweon (3032235207) Lab4

Jin Kweon

9/23/2017

fitted values = predicted values lm(response $\sim$ expression) corresponds to a linear model: response = β_0 + $\beta_1$expression + ϵ:

$$\begin{pmatrix} y1 \\ y2 \\ y3 \\ . \\ . \\ . \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} \\ 1 & x_{21} \\ 1 & x_{31} \\ . & \\ . & \\ . & \\ 1 & x_{n1} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ . \\ . \\ . \\ \epsilon_n \end{pmatrix}$$

$y - \bar{y} = \hat{\beta_{0mc}} + \hat{\beta_{1mc}}(x - \bar{x})$ is a mean centered regression. So, $y = (\bar{y} + \hat{\beta_{0mc}} - \hat{\beta_{1mc}}\bar{x}) + \hat{\beta_{1mc}}x$

$\frac{y - \bar{y}}{se(y)} = \hat{\beta_{0mc}} + \hat{\beta_{1mc}}(\frac{x-\bar{x}}{se(x)})$ and then solve it.

```
#Q. when I use "-" in the lm function reg, it does not show its coefficient... why????
#Q. What are the best ways to recover from mean-centered and standardized data? Can we just literally c

reg <- lm(mpg ~ disp - hp, data = mtcars)


reg1 <- lm(mpg ~ disp, data = mtcars)
summary(reg1)
```

```
##
## Call:
## lm(formula = mpg ~ disp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8922 -2.2022 -0.9631  1.6272  7.2305
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29.599855   1.229720  24.070  < 2e-16 ***
## disp        -0.041215   0.004712  -8.747 9.38e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.251 on 30 degrees of freedom
## Multiple R-squared:  0.7183, Adjusted R-squared:  0.709
## F-statistic: 76.51 on 1 and 30 DF,  p-value: 9.38e-10
```

```
reg1
```

```
##
## Call:
## lm(formula = mpg ~ disp, data = mtcars)
##
## Coefficients:
```

```
## (Intercept)         disp
##    29.59985     -0.04122
```

```r
names(reg1)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```r
scale_mt <- as.data.frame(scale(mtcars, T, F))
reg2 <- lm(mpg ~ disp, data = scale_mt)
#Since it is mean-centered, the intercept should be approsimately zero.

#Recover intercept!!!
apply(as.matrix(mtcars$mpg), 2, mean) + reg2$coefficients[1] - reg2$coefficients[2] * apply(as.matrix(m
```

```
## (Intercept)
##    29.59985
```

```r
scale_mt2 <- as.data.frame(scale(mtcars, T, T))
reg3 <- lm(mpg ~ disp, data = scale_mt2)


#Get OLS with no intercept is different with getting zero intercept by doing mean-centered.
lm(mpg ~ disp -1, data = mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ disp - 1, data = mtcars)
##
## Coefficients:
##    disp
## 0.05905
```

```r
lm(mpg ~ disp +0, data = mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ disp + 0, data = mtcars)
##
## Coefficients:
##    disp
## 0.05905
```

```r
lm(mpg ~ disp, data = mtcars, subset = am == 0)
```

```
##
## Call:
## lm(formula = mpg ~ disp, data = mtcars, subset = am == 0)
##
## Coefficients:
## (Intercept)         disp
##    25.15706     -0.02758
```

```r
#Same as the one below:
new <- mtcars %>% filter(am == 0)
lm(mpg ~ disp, new)
```

```
##
## Call:
## lm(formula = mpg ~ disp, data = new)
##
## Coefficients:
## (Intercept)         disp
##    25.15706     -0.02758
```

```r
reg0 <- lm(mpg ~., data = mtcars)
#use all variables. When you want to exclude just one variable, you might be able to modify data frame,


reg_sum <- summary(reg1)
reg_sum
```

```
##
## Call:
## lm(formula = mpg ~ disp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8922 -2.2022 -0.9631  1.6272  7.2305
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29.599855   1.229720  24.070  < 2e-16 ***
## disp        -0.041215   0.004712  -8.747 9.38e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.251 on 30 degrees of freedom
## Multiple R-squared:  0.7183, Adjusted R-squared:  0.709
## F-statistic: 76.51 on 1 and 30 DF,  p-value: 9.38e-10
```

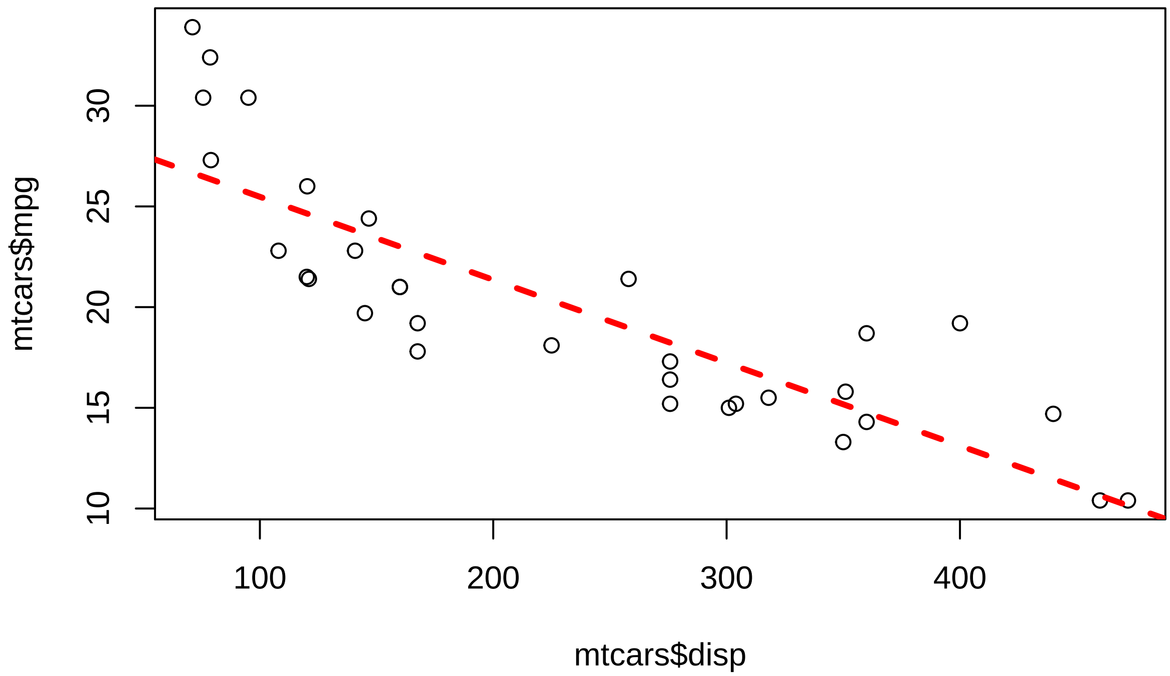
```r
class(reg_sum)
```

```
## [1] "summary.lm"
```

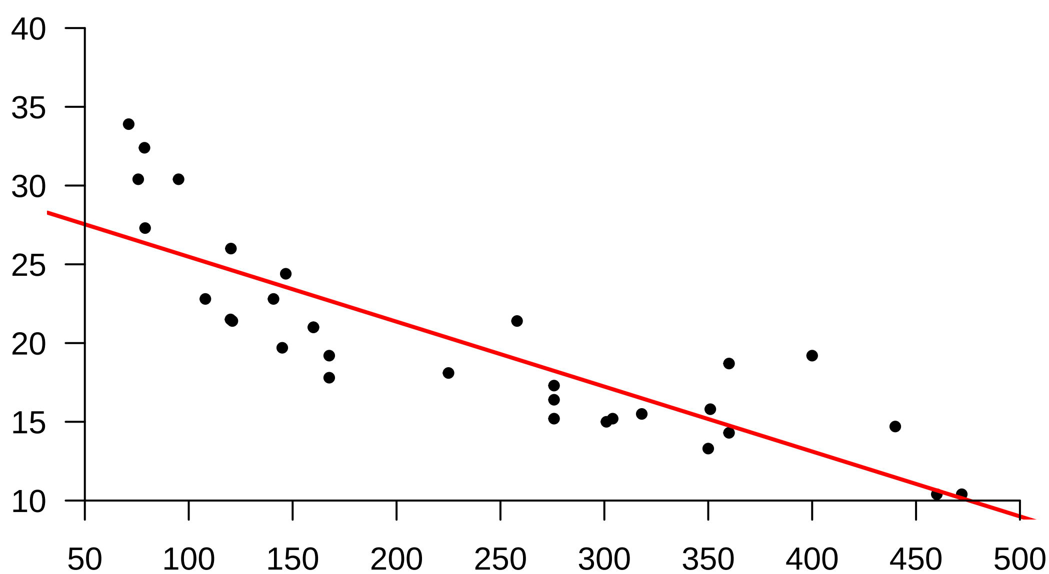
```r
names(reg_sum) #contain different things with reg1
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

```r
plot(mtcars$disp, mtcars$mpg)
abline(reg1, col = "Red", lty = 2, lwd = 3) #lty is a line type and lwd is line width
```



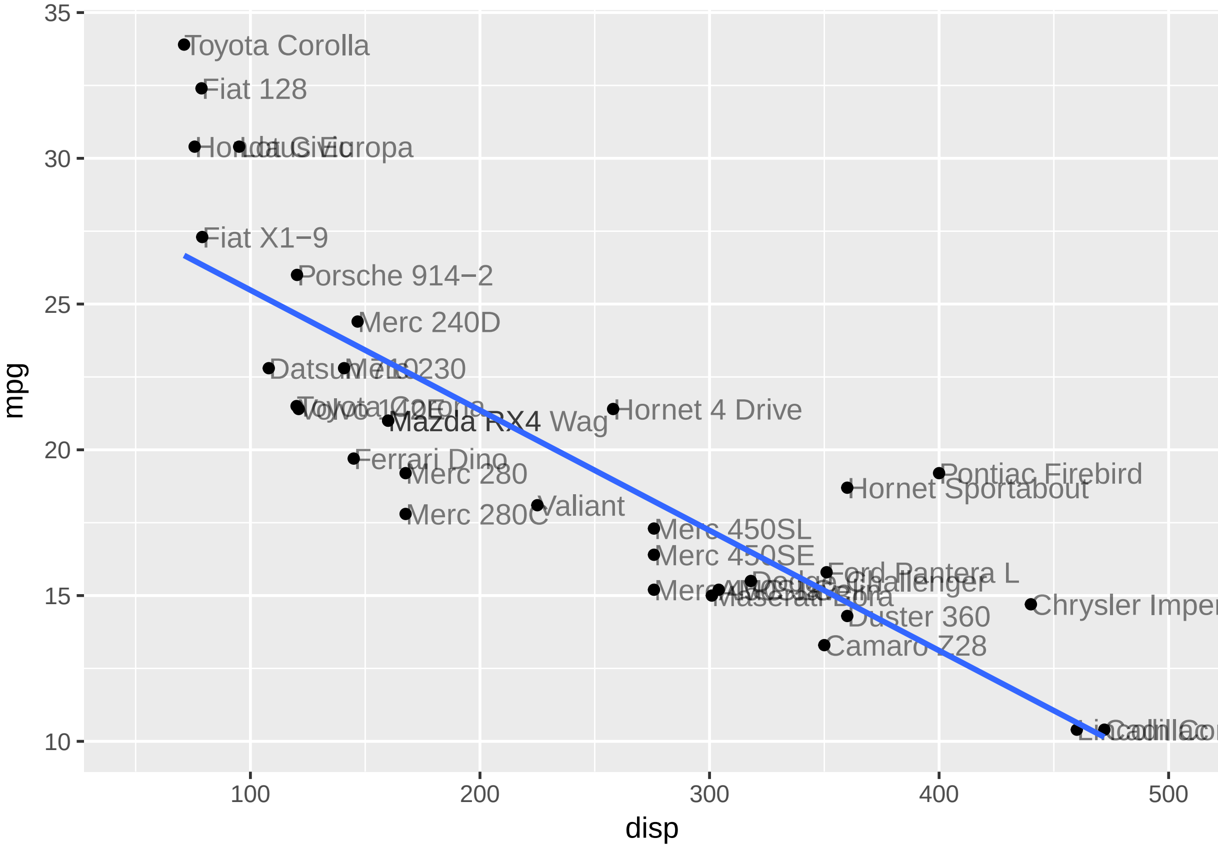

```
plot.new()
plot.window(xlim = c(50, 500), ylim = c(10, 40))
points(mtcars$disp, mtcars$mpg, pch = 20, cex = 1) #pch = dot type. cex = size
abline(reg1, col = "red", lwd = 2)
axis(side = 1, pos = 10, at = seq(50, 500, 50)) #side=1 means x axis with pos=10 starting point
axis(side = 2, las = 1, pos = 50, at = seq(10, 40, 5)) #side=2 means y axis
```




```
#see warning if i do xlim(100,500) because there are some data outside of x < 100.

graph <- ggplot(data = mtcars, aes(x = disp, y = mpg))
graph <- graph + geom_point() + geom_text(aes(label = rownames(mtcars)), hjust = 0, alpha = 0.5) +
  xlim(50,500) +
  geom_smooth(method = "lm", se = FALSE)
#or, I can do this below:
```

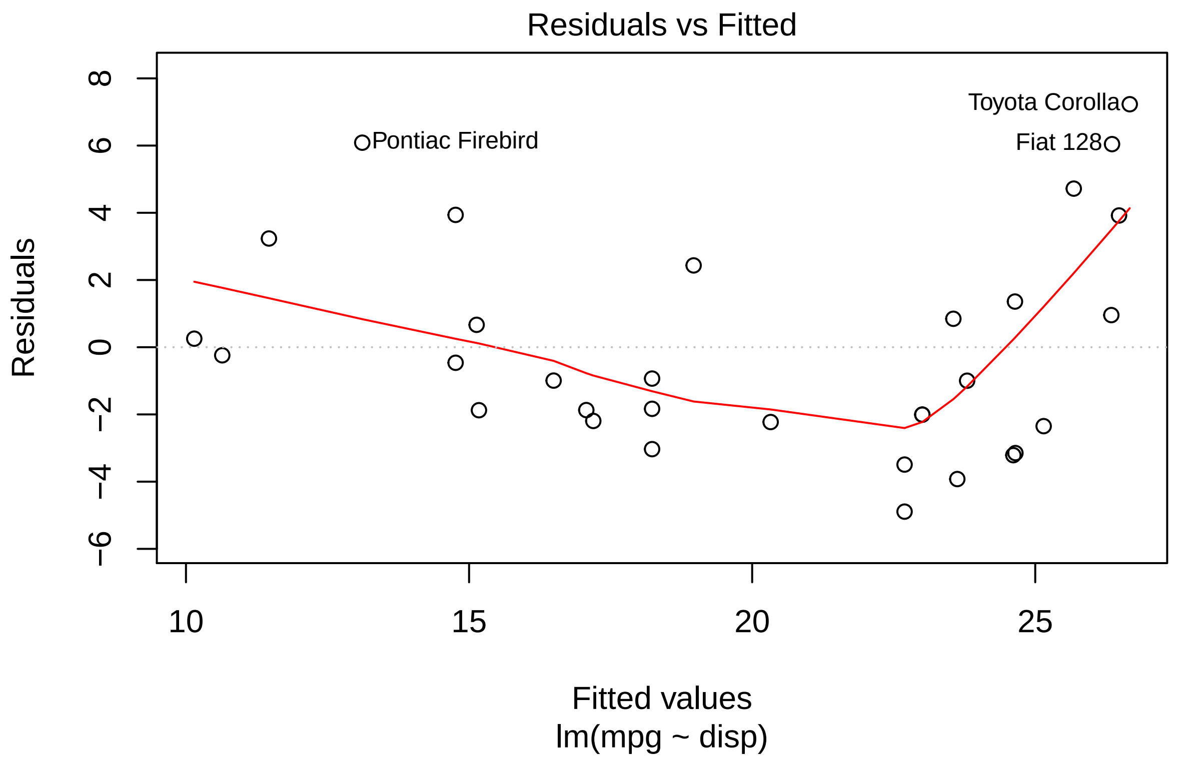
```r
#+ geom_abline(intercept = reg1$coefficients[1], slope = reg1$coefficients[2], col = "blue", lwd = 1.2)
graph
```



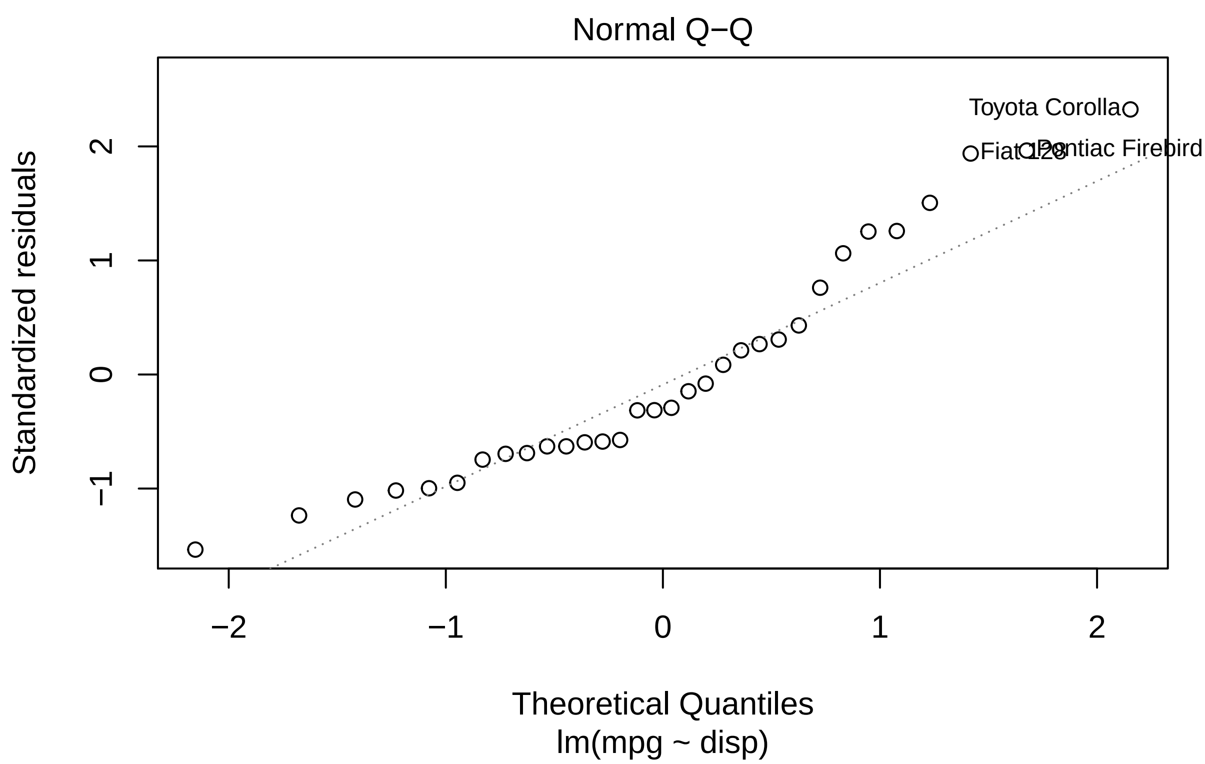
• a plot of residuals versus fitted values: for example there may be a pattern in the residuals that suggests that we should be fitting a curve rather than a line; • a normal probability: if residuals are from a normal distribution points should lie, to within statistical error, close to a line.

Keep in mind that these diagnostic plots are not definitive. Rather, they draw attention to points that require further investigation.

```r
plot(reg1, which = 1) #seems like there is a little bit of pattern. Curve might be better.
```




```r
plot(reg1, which = 2) # follows normal pretty well.
```



ANOVA test gives you: $\sum(y_i - \bar{y})^2 = \sum(y_i - \hat{y}_i)^2 + \sum(\hat{y}_i - \bar{y})^2$. Below test outputs RSS = 317.16, Regss = 808.89, and TSS = 808.89 + 317.76. As explanatory variables go up, RSS goes down (as R^2 goes up), and Regss goes up. Tss stays the same although the number of explanatory variable changes. So, it is about changes between RSS and Regss.

```r
reg_anova <- anova(reg1)
reg_anova #have the same F value as I got from lm summary.
```

```
## Analysis of Variance Table
##
## Response: mpg
##           Df Sum Sq Mean Sq F value   Pr(>F)
## disp       1 808.89  808.89  76.513 9.38e-10 ***
## Residuals 30 317.16   10.57
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#We reject the null. So, disp should be kept for linear regression.
#Residuals Sum Sq = RSS
#explantory varaible Sum Sq = Regss
#sum of RSS and Regss = TSS
#Mean sq = Sum Sq / DF

anova(lm(mpg ~ disp + hp, data = mtcars))
```

```
## Analysis of Variance Table
##
## Response: mpg
##           Df Sum Sq Mean Sq F value    Pr(>F)
## disp       1 808.89  808.89 82.7454 5.406e-10 ***
## hp         1  33.67   33.67  3.4438   0.07368 .
## Residuals 29 283.49    9.78
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
R2 <- (reg_anova$`Sum Sq`[1] / sum(reg_anova$`Sum Sq`))
R2
```

```
## [1] 0.7183433
```

```r
summary(reg1)$r.squared
```

```
## [1] 0.7183433
```

Although this works, computationally it is not the best way to compute b. Why? Because it is inefficient and be very inaccurate when the predictors are strongly correlated.

Always include intercept term!!!

```r
int <- rep(1, nrow(reg1$model))

x <- cbind(int, reg1$model[,2])
y <- reg1$model[,1]

solve(t(x) %*% x) %*% t(x) %*% y
```

```
##            [,1]
## int 29.59985476
##     -0.04121512
```

```r
solve(crossprod(x, x), crossprod(x,y))
```

```
##            [,1]
## int 29.59985476
```

```
##     -0.04121512
```

```r
coef <- reg1$coefficients
coef
```

```
## (Intercept)        disp
## 29.59985476 -0.04121512
```

$$y = X\hat{\beta} = (QR)\hat{\beta}$$

, so,

$$\hat{\beta} = (QR)^{-1}y = R^{-1}Q^{-1}y = R^{-1}Q^{T}y, \ as\ Q^{T}Q = I$$

. And, to be unique, this works only if X is linerly independent meaning that X has full rank, and this is the same as beta's least square estimator.

Professor made a mistake on f <- t(Q %*% y).

```r
#Q. what is QR$qr??? ===> QR includes the R (upper triangular) on top, and Q in compact form.

QR <- qr(x) #same as reg1$qr

Q <- qr.Q(QR)
R <- qr.R(QR)

backsolve(R, crossprod(Q, y)) #Better way than solve when we RX = y.
```

```
##             [,1]
## [1,] 29.59985476
## [2,] -0.04121512
```

```r
#forwardsolve(R, crossprod(Q, y)) --> This will work on lower triangular matrix application LX = y.
solve(R, crossprod(Q, y))
```

```
##            [,1]
## int 29.59985476
##     -0.04121512
```