STA206 Assignment 6

Zhen Zhang

November 11, 2015

Question 1

(a)

```
property <- read.table("~/Github/UCDavis/STA206/hw/hw4/STA206_hw4_data_property.txt")
colnames(property) <- c("Y", paste0("X", 1:4))
```

linear regression:

```
fit1 <- lm(Y ~ X1 + X2 + X4 + X3, data = property)
summary(fit1)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2 + X4 + X3, data = property)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1872 -0.5911 -0.0910  0.5579  2.9441
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.220e+01  5.780e-01  21.110  < 2e-16 ***
## X1          -1.420e-01  2.134e-02  -6.655 3.89e-09 ***
## X2           2.820e-01  6.317e-02   4.464 2.75e-05 ***
## X4           7.924e-06  1.385e-06   5.722 1.98e-07 ***
## X3           6.193e-01  1.087e+00   0.570     0.57
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.137 on 76 degrees of freedom
## Multiple R-squared:  0.5847, Adjusted R-squared:  0.5629
## F-statistic: 26.76 on 4 and 76 DF,  p-value: 7.272e-14
```

(b)

The coefficient of X_3 is 0.6193435:

```
fit1$coefficients[5]
```

```
##        X3
## 0.6193435
```

The coefficient of partial determination $R^2_{Y3|124} = \frac{SSR(X_3|X_1,X_2,X_4)}{SSE(X_1,X_2,X_4)}$

```r
anova(fit1)
```

```
## Analysis of Variance Table
##
## Response: Y
##           Df Sum Sq Mean Sq F value    Pr(>F)    
## X1         1 14.819  14.819 11.4649  0.001125 ** 
## X2         1 72.802  72.802 56.3262 9.699e-11 ***
## X4         1 50.287  50.287 38.9062 2.306e-08 ***
## X3         1  0.420   0.420  0.3248  0.570446    
## Residuals 76 98.231   1.293                      
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

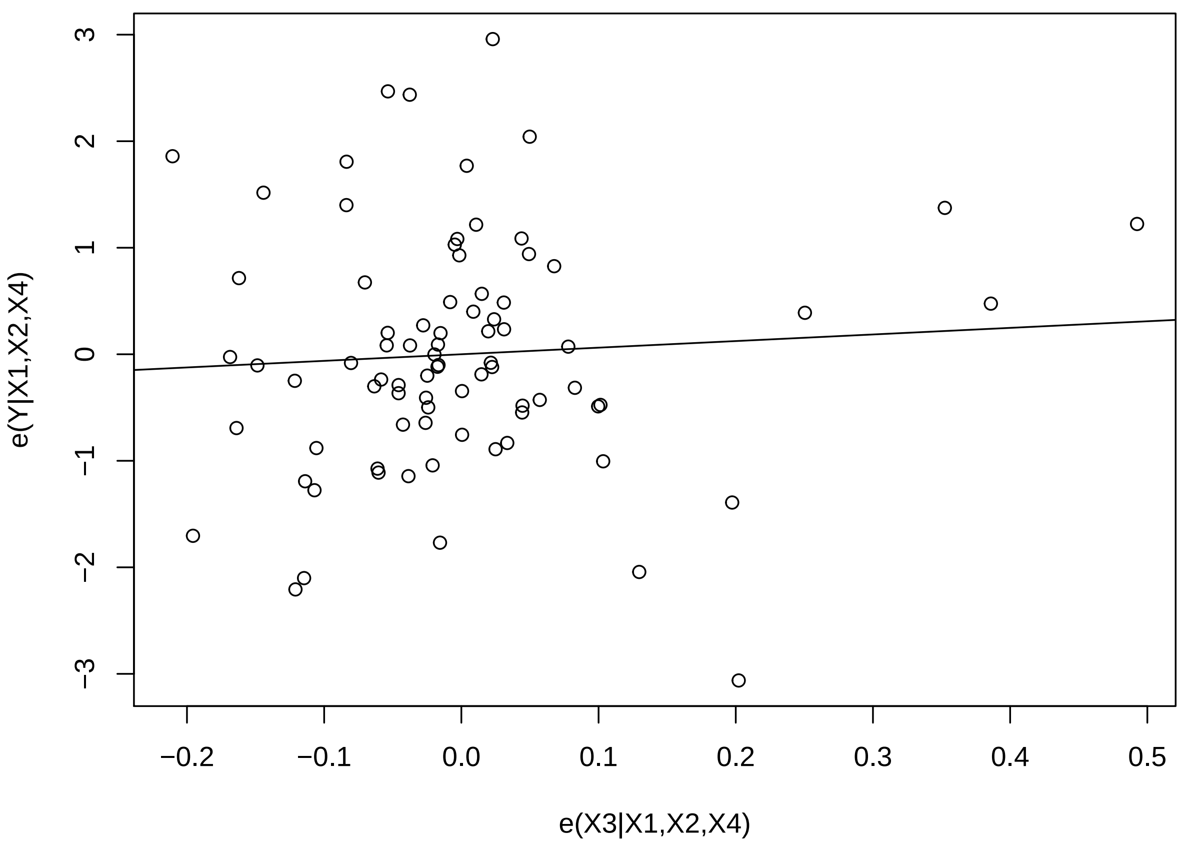
So here, $SSR(X_3|X_1, X_2, X_4) = 0.420$, $SSE(X_1, X_2, X_4) = 0.420 + 98.231 = 98.651$, so $R^2_{Y3|124} = \frac{0.420}{98.651} = 0.004257433$.

Since the coefficient of X_3 is larger than 0, so $r_{Y3|124} = \sqrt{0.004257433.} = 0.06524901$.

$R^2_{Y3|124}$ measures the marginal contribution in proportional reduction in SSE by adding X_3 into the model. The result shows there is almost no marginal contribution in proportional reduction in SSE by adding X_3 into the model.

(c)

```r
fit2 <- lm(Y ~ . - X3, data = property)
fit3 <- lm(X3 ~ . - Y, data = property)
fit4 <- lm(fit2$residuals ~ fit3$residuals)
plot(fit3$residuals, fit2$residuals, xlab = "e(X3|X1,X2,X4)", ylab = "e(Y|X1,X2,X4)")
abline(fit4)
```



The points scatter evenly on both sides of the regression line, and the regression line is almost horizontal, which means there is almost no marginal importance of X_3 in reducing the residual variability in Y after accounting for the linear effects in the rest of the X variables.

(d)

```
fit2 <- lm(Y ~ . - X3, data = property)
fit3 <- lm(X3 ~ . - Y, data = property)
fit4 <- lm(fit2$residuals ~ fit3$residuals)
summary(fit4)
```

```
##
## Call:
## lm(formula = fit2$residuals ~ fit3$residuals)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1872 -0.5911 -0.0910  0.5579  2.9441
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.467e-17  1.239e-01   0.000    1.000
## fit3$residuals 6.193e-01  1.066e+00   0.581    0.563
##
## Residual standard error: 1.115 on 79 degrees of freedom
## Multiple R-squared:  0.004255,   Adjusted R-squared:  -0.008349
## F-statistic: 0.3376 on 1 and 79 DF,  p-value: 0.5629
```

```r
fit4$coefficients[2]
```

```
## fit3$residuals
##      0.6193435
```

So the regression slope from this regression is 0.6193435, the same as the coefficient of X_3 in part (b). This means that X_3 is uncorrelated with X_1, X_2, X_4.

(e)

```r
anova(fit4)
```

```
## Analysis of Variance Table
##
## Response: fit2$residuals
##                Df Sum Sq Mean Sq F value Pr(>F)
## fit3$residuals  1  0.420 0.41975  0.3376 0.5629
## Residuals      79 98.231 1.24343
```

The regression sum of squares is 0.420, identical with extra sum of squares $SSR(X_3|X_1, X_2, X_4)$ from the R output of Model 1.

(f)

The correlation coefficient r between the two sets of residuals $e(Y|X_1, X_2, X_4)$ and $e(X_3|X_1, X_2, X_4)$ is: $\frac{0.420}{98.231}$ = 0.004255, almost identical with $r_{Y3|124}$. r^2 is also 0.004255.

(g)

```r
fit5 <- lm(property$Y ~ fit3$residuals)
summary(fit5)
```

```
##
## Call:
## lm(formula = property$Y ~ fit3$residuals)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7641 -1.1392 -0.1056  1.1221  4.1630
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     15.1389     0.1921  78.807   <2e-16 ***
## fit3$residuals   0.6193     1.6528   0.375    0.709
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.729 on 79 degrees of freedom
## Multiple R-squared:  0.001774,   Adjusted R-squared:  -0.01086
## F-statistic: 0.1404 on 1 and 79 DF,  p-value: 0.7089
```

```r
fit5$coefficients[2]
```

```
## fit3$residuals
##      0.6193435
```

The fitted regression slope from this regression is 0.6193435, identical with the fitted regression coefficient of X_3 from part (b). It means Y_3 is indeed uncorrelated with Y.

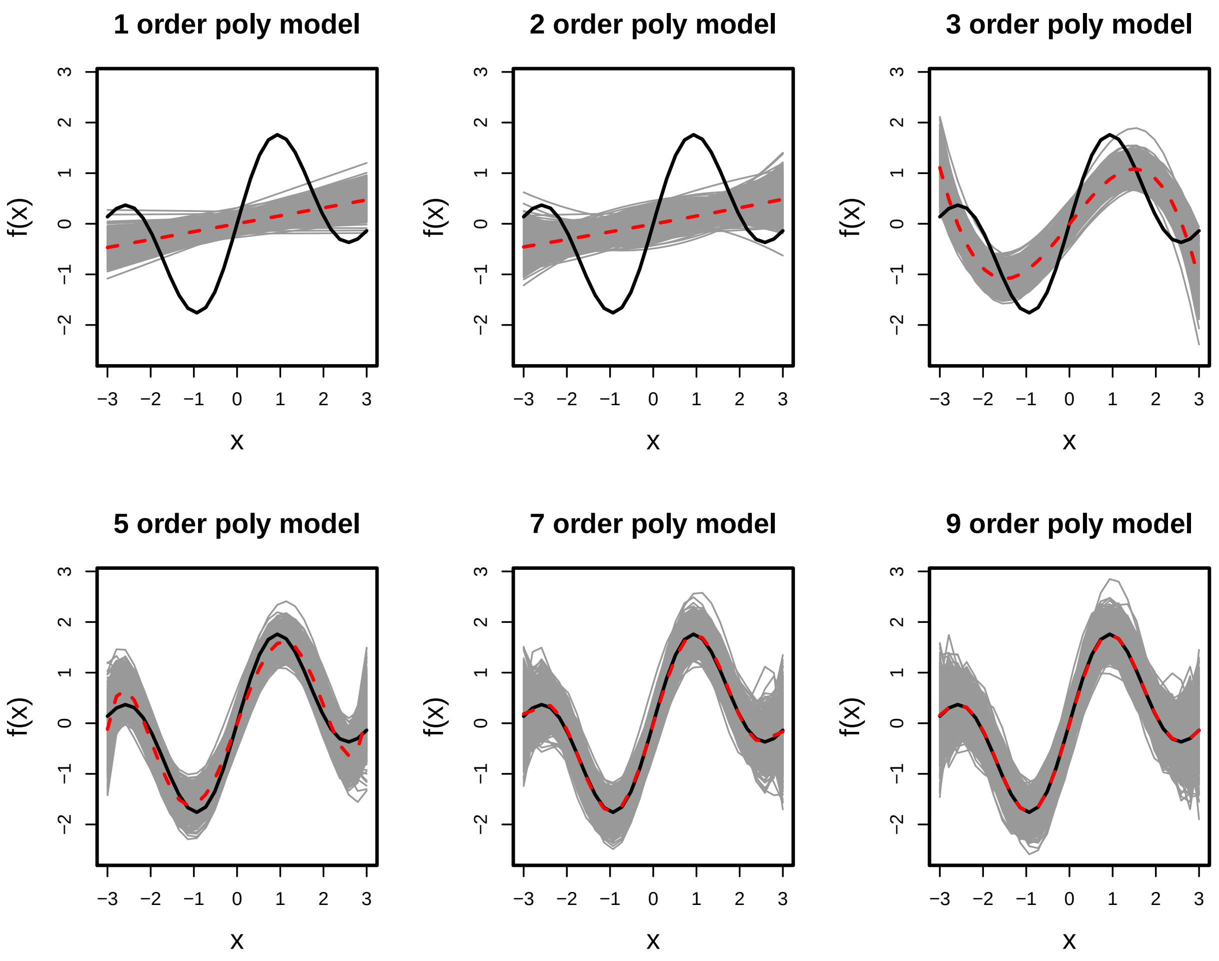
Question 2

The code is:

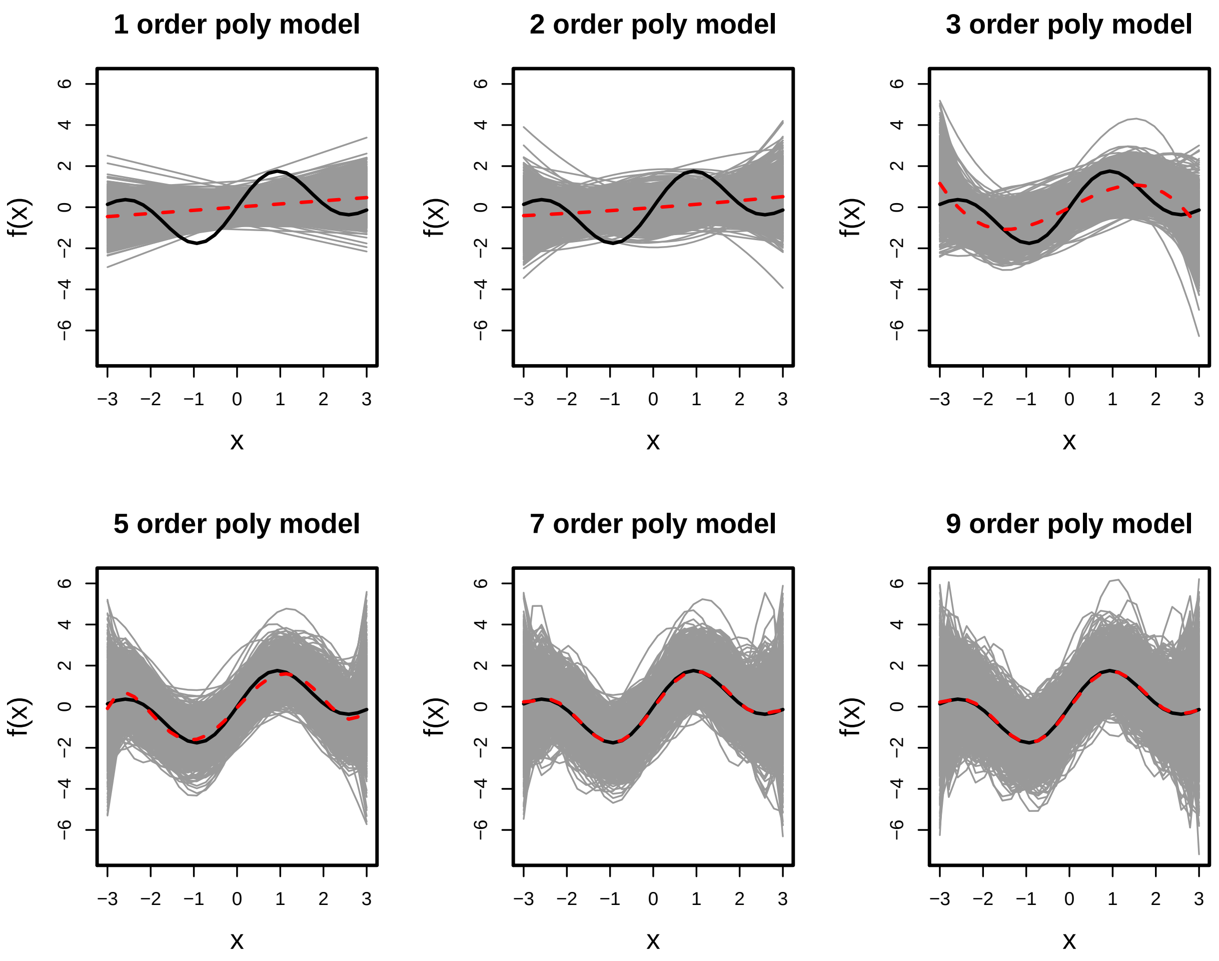
```r
sigma_for_each <- function(sigma.e) {
    n = 30  #sample size
    X = seq(-3, 3, length.out = n)  # design points: fixed throughout the simulation
    f.X = sin(X) + sin(2 * X)
    # the values of the true regression function on the design points.

    par(lwd = 2, cex.lab = 1.5, cex.main = 1.5)
    # customize features of the graph in this R session

    # Observations: Y_i=f(x_i)+e_i, e_i ~ i.i.d. N(0, sigma.e), i=1,..., n
    rep = 1000  # number of independent data sets (replicates) to be generated

    Y = matrix(0, n, rep)
    # matrix to record the observations; each column corresponds to one
    # replicate

    for (k in 1:rep) {
        set.seed(1234 + k * 56)
        # set seed for the random number generator; for reproducibility of the
        # result
        e.c = rnorm(n, 0, sigma.e)  #generate the random errors
        Y.c = f.X + e.c
        # generate observations for kth replicate: true mean response + error
        Y[, k] = Y.c
    }

    l.order = c(1, 2, 3, 5, 7, 9)  # order of the polynomial models to be fitted
    Y.fit = array(0, dim = c(n, rep, length(l.order)))
    # record the fitted values; 1st index corresponds to cases; 2nd index
    # corresponds to replicates, 3rd index corresponds to models
```

```r
for (k in 1:rep) {
    Y.c = Y[, k]  #observations of the kth replicate

    for (l in 1:length(l.order)) {
        fit.c = lm(Y.c ~ poly(X, l.order[l], raw = TRUE))
        # fit a polynomial model with order l.order[l]; raw=TRUE means raw
        # polynomial is used; raw= FALSE mean orthogonal polynomial is used
        Y.fit[, k, l] = fitted(fit.c)
    }  # end of l loop

}  # end of k loop

Y.fit.ave = apply(Y.fit, c(1, 3), mean)  # average across  replicates (2nd index)

par(mfrow = c(2, 3))

for (l in 1:length(l.order)) {
    plot(X, f.X, type = "n", xlab = "x", ylab = "f(x)", ylim = range(Y.fit),
        main = paste(l.order[l], "order poly model"))
    # set plot axis label/limit, title, etc.

    for (k in 1:rep) {
        points(X, Y.fit[, k, l], type = "l", lwd = 1, col = grey(0.6))
        # fitted response curves of lth model: grey
    }  # end of k loop

    points(X, f.X, type = "l", col = 1)  # true mean response: solid black
    points(X, Y.fit.ave[, l], type = "l", col = 2, lty = 2)
    # averaged (across replicates) fitted mean reponse of the lth model:
    # broken red

}  #end l loop
par(mfrow = c(1, 1))

# compare SSE; variance, bias^2 and mean-squared-estimation-error =
# variance+bias^2 across models
SSE = matrix(0, rep, length(l.order))  # record SSE for each model on each replicate
resi = array(0, dim = c(n, rep, length(l.order)))
# record residuals : residual := obs-fitted
error.fit = array(0, dim = c(n, rep, length(l.order)))
# record estimation errors in the fitted values: error := fitted value
# - true mean response

for (l in 1:length(l.order)) {
    temp = Y - Y.fit[, , l]
    resi[, , l] = temp  # residuals
    SSE[, l] = apply(temp^2, 2, sum)  # SSE=sum of squared residuals across cases
    error.fit[, , l] = Y.fit[, , l] - matrix(f.X, n, rep, byrow = FALSE)
    # estimation error = fitted value - true mean response
}

SSE.mean = apply(SSE, 2, mean)
# mean SSE (averaged over the replicates); this is the empirical
```

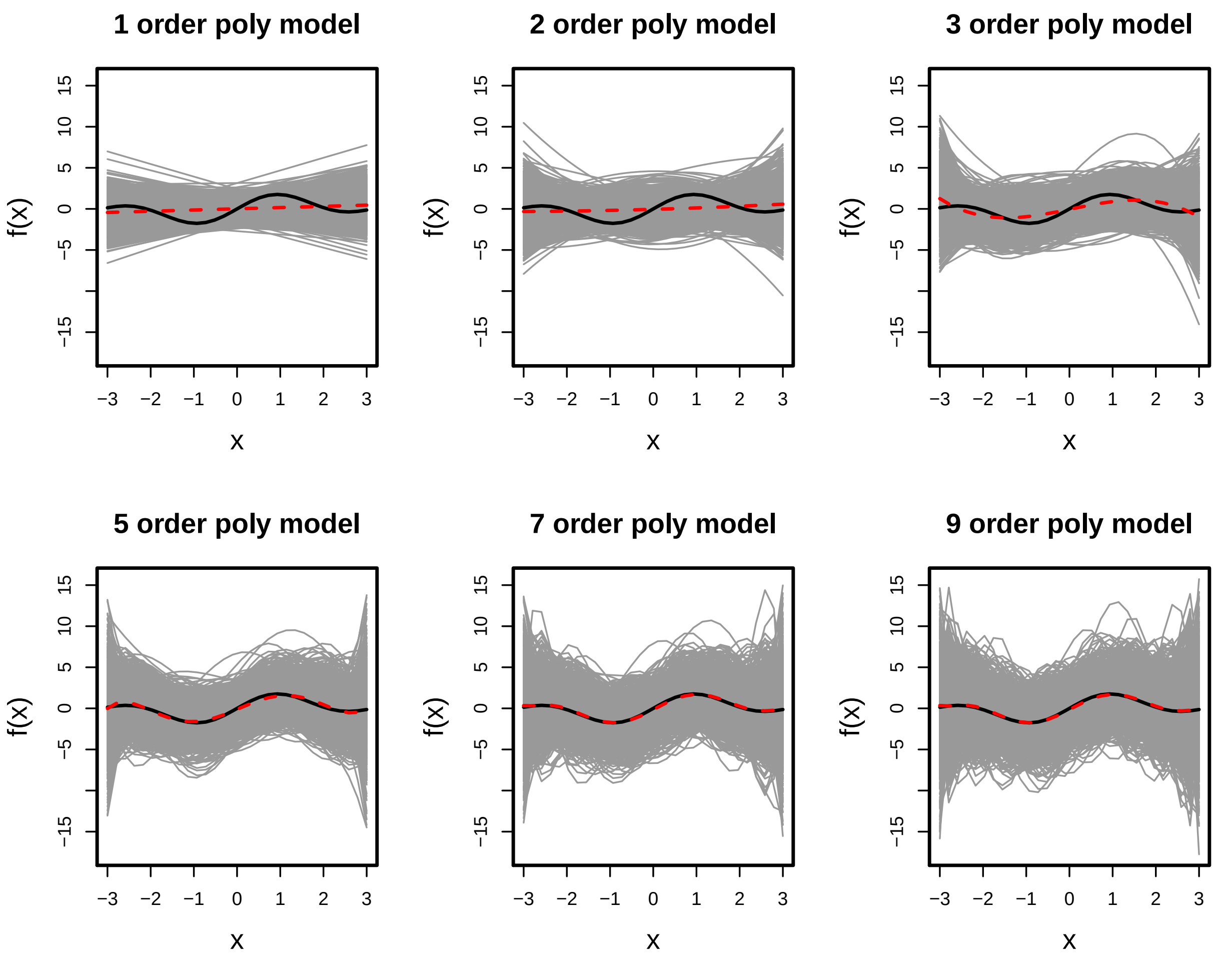
```
    # version of E(SSE)
    bias = apply(error.fit, c(1, 3), mean)
    # bias= mean (averaged across replicates) errors in the fitted values
    variance = apply(Y.fit, c(1, 3), var)
    # variance (across replicates) of the fitted values
    err2.mean = apply(error.fit^2, c(1, 3), mean)
    # mean-squared-estimation errors: squared estimation errors of the
    # fitted values averaged across replicates

    # bias, variance, err2.mean are calculated on each design point/case
    # for each model to facilitate comparison among models, we sum them
    # across the design points/cases to produce an overall quantity (each)
    # for each model
    bias2.ave = apply(bias^2, 2, mean)
    # average bias^2 across design points for each model: overall in-sample
    # bias
    variance.ave = apply(variance, 2, mean)
    # average variance across design points for each model: overall
    # in-sample variance
    err2.mean.ave = apply(err2.mean, 2, mean)
    # average mean-squared-estimation-error across design points for each
    # model: over-all in-sample msee

    cat("\n")
    cat(sprintf("This is the model for error variance %g", sigma.e))
    cat("\n")
    print_matrix <- data.frame(order = l.order, sse = SSE.mean, `bias square` = bias2.ave,
        variance = variance.ave, msee = err2.mean.ave)
    print(print_matrix)


    return(print_matrix)
}
sigma_total <- c(`0.5` = 0.5, `2` = 2, `5` = 5)
print_matrix_sigma_each <- invisible(lapply(sigma_total, sigma_for_each))
```



```
##
## This is the model for error variance 0.5
##   order       sse  bias.square   variance        msee
## 1     1 34.876531 0.9327759976 0.01715738 0.94991622
## 2     2 34.621515 0.9328097760 0.02563261 0.95841676
## 3     3 18.712425 0.4107060098 0.03338640 0.44405902
## 4     5  7.229235 0.0396581174 0.04909157 0.08870060
## 5     7  5.559057 0.0010606068 0.06626084 0.06725519
## 6     9  5.013139 0.0001106624 0.08371268 0.08373963
```



```
##
## This is the model for error variance 2
##   order       sse bias.square  variance      msee
## 1     1 139.67949 0.932831618 0.2745181 1.2070752
## 2     2 135.59924 0.933372072 0.4101218 1.3430838
## 3     3 116.16362 0.411269780 0.5341824 0.9449180
## 4     5  97.96621 0.040819145 0.7854652 0.8254989
## 5     7  88.58094 0.002420521 1.0601735 1.0615338
## 6     9  80.20314 0.001650710 1.3394029 1.3397142
```



```
##
## This is the model for error variance 5
##   order      sse bias.square variance     msee
## 1     1 727.8384  0.93314309 1.715738 2.647165
## 2     2 702.3368  0.93652093 2.563261 3.497219
## 3     3 663.2994  0.41442689 3.338640 3.749728
## 4     5 606.1921  0.04732090 4.909157 4.951569
## 5     7 553.5687  0.01003604 6.626084 6.629494
## 6     9 501.2640  0.01027498 8.371268 8.373172
```

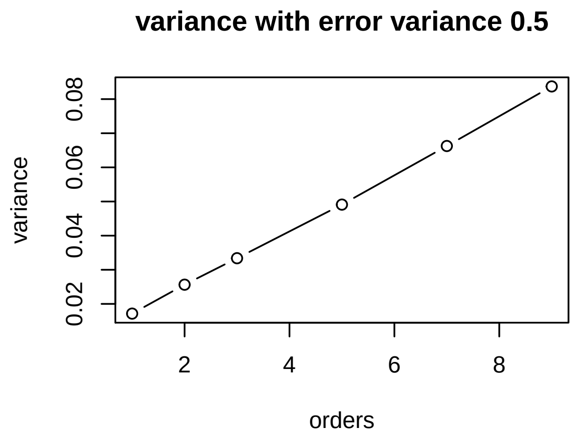
Now answer the questions:

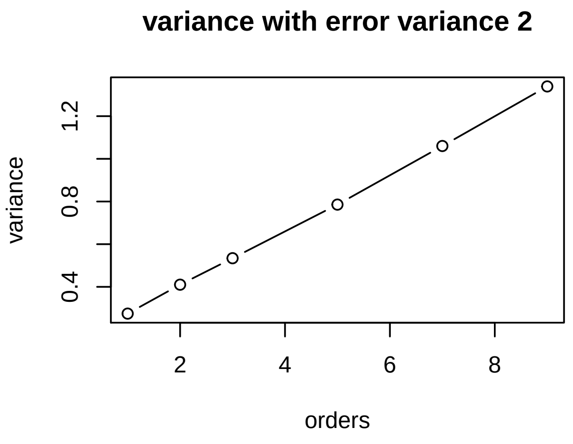
(a) I think there is no correct model, since under different noise levels, I will get different best model, which means best model is closely related to the noise level.

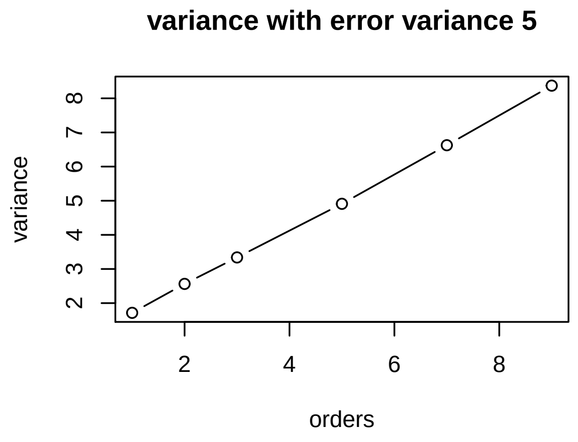
(b) The (in-sample) model variance for each of these models are:

```
analysis <- function(name) {
    print_matrix_order_name <- sapply(print_matrix_sigma_each, "[[", name)
    indices <- c(1, 2, 3, 5, 7, 9)
    row.names(print_matrix_order_name) <- indices
    print(print_matrix_order_name)
    opar <- par()
    par(mfrow = c(2, 2))
    cat("\n")
    cat(sprintf("This is the plot for %s", name))
    cat("\n")

    invisible(lapply(1:3, function(i) {
        plot(indices, print_matrix_order_name[, i], type = "b", xlab = "orders",
            ylab = name, main = paste(name, "with error variance", sigma_total[i]))
    }))

    invisible(print_matrix_order_name)
}

analysis("variance")
```

```
##          0.5         2        5
## 1 0.01715738 0.2745181 1.715738
## 2 0.02563261 0.4101218 2.563261
## 3 0.03338640 0.5341824 3.338640
## 5 0.04909157 0.7854652 4.909157
## 7 0.06626084 1.0601735 6.626084
## 9 0.08371268 1.3394029 8.371268
##
## This is the plot for variance
```





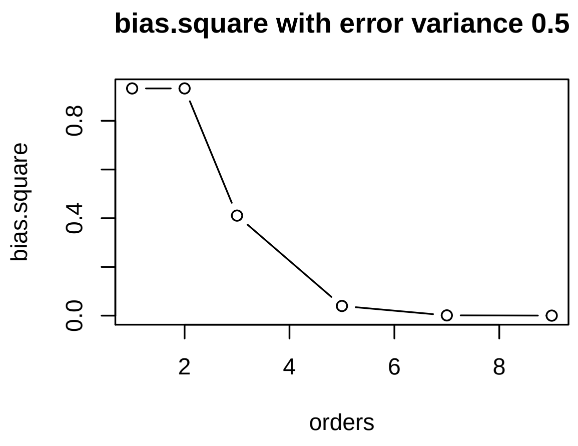


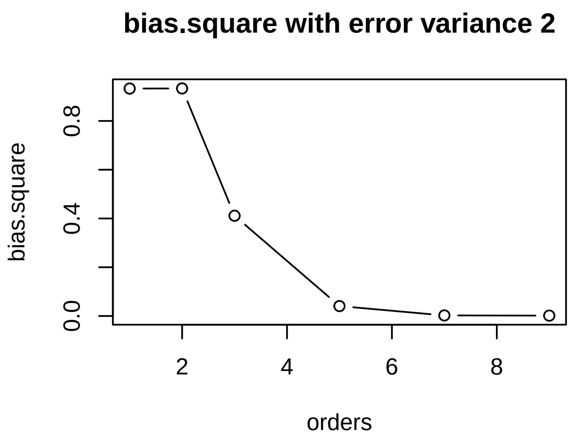
It is clear that the (in-sample) model variance changes with the error variance.

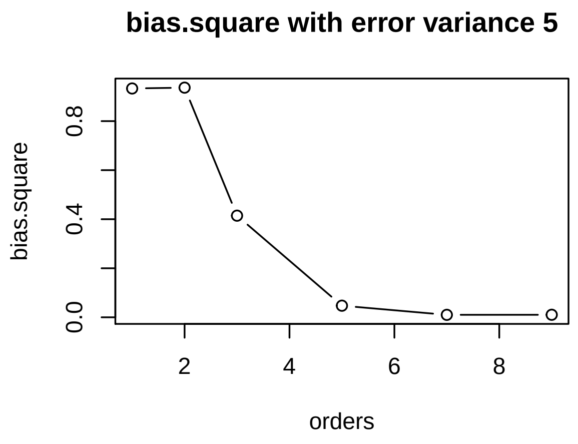
(c) The (in-sample) model bias for each of these models are:

```
analysis("bias.square")
```

```
##             0.5            2          5
## 1 0.9327759976 0.932831618 0.93314309
## 2 0.9328097760 0.933372072 0.93652093
## 3 0.4107060098 0.411269780 0.41442689
## 5 0.0396581174 0.040819145 0.04732090
## 7 0.0010606068 0.002420521 0.01003604
## 9 0.0001106624 0.001650710 0.01027498
##
## This is the plot for bias.square
```







The (in-sample) model bias almost does not change with the error variance.

(d)

```
print_matrix_sigma_each
```

```
## $`0.5`
##   order       sse  bias.square   variance       msee
## 1     1 34.876531 0.9327759976 0.01715738 0.94991622
## 2     2 34.621515 0.9328097760 0.02563261 0.95841676
## 3     3 18.712425 0.4107060098 0.03338640 0.44405902
## 4     5  7.229235 0.0396581174 0.04909157 0.08870060
## 5     7  5.559057 0.0010606068 0.06626084 0.06725519
## 6     9  5.013139 0.0001106624 0.08371268 0.08373963
##
## $`2`
##   order       sse bias.square  variance      msee
## 1     1 139.67949 0.932831618 0.2745181 1.2070752
## 2     2 135.59924 0.933372072 0.4101218 1.3430838
## 3     3 116.16362 0.411269780 0.5341824 0.9449180
## 4     5  97.96621 0.040819145 0.7854652 0.8254989
## 5     7  88.58094 0.002420521 1.0601735 1.0615338
## 6     9  80.20314 0.001650710 1.3394029 1.3397142
##
## $`5`
##   order      sse bias.square variance     msee
## 1     1 727.8384  0.93314309 1.715738 2.647165
## 2     2 702.3368  0.93652093 2.563261 3.497219
## 3     3 663.2994  0.41442689 3.338640 3.749728
## 4     5 606.1921  0.04732090 4.909157 4.951569
## 5     7 553.5687  0.01003604 6.626084 6.629494
## 6     9 501.2640  0.01027498 8.371268 8.373172
```

When error variance is 0.5, bias plays a dominant role in the (in-sample) mean-squared-estimation-error, with error variance 2, bias is still larger than variance, but not as dominant as that of error variance 0.5. While error variance is 5, variance instead plays a dominant role in the (in-sample) mean-squared-estimation-error.

The reason is, witht a low error variance, the bias is larger than variance, then bias will play a dominant role. With a high error variance, the variance in a model increases very fast. So with the increase of polynomial order, the increase in variance is much larger than the decrease in bias, which means variance will play a dominant role.

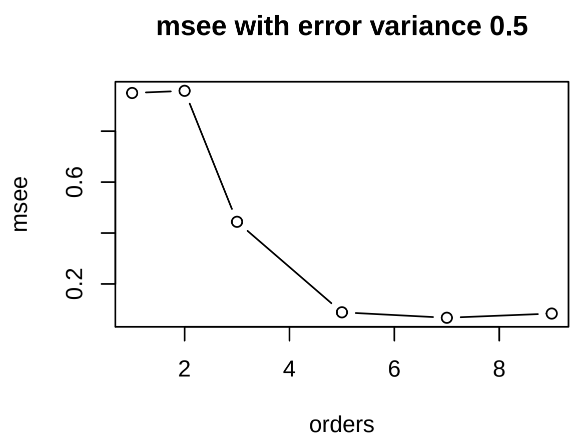
(e)

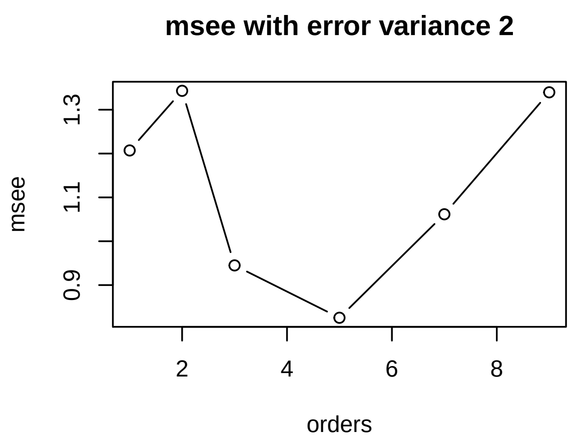
```r
print_matrix_order_msee <- analysis("msee")
```

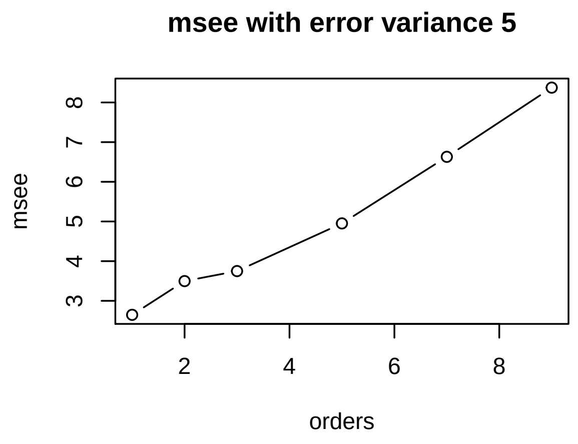
```
##          0.5         2        5
## 1 0.94991622 1.2070752 2.647165
## 2 0.95841676 1.3430838 3.497219
## 3 0.44405902 0.9449180 3.749728
## 5 0.08870060 0.8254989 4.951569
## 7 0.06725519 1.0615338 6.629494
## 9 0.08373963 1.3397142 8.373172
##
## This is the plot for msee
```

```r
msee_min <- apply(print_matrix_order_msee, 2, function(i) {
    as.numeric(rownames(print_matrix_order_msee)[order(i)[1]])
})
print(msee_min)
```

```
## 0.5   2   5
##   7   5   1
```







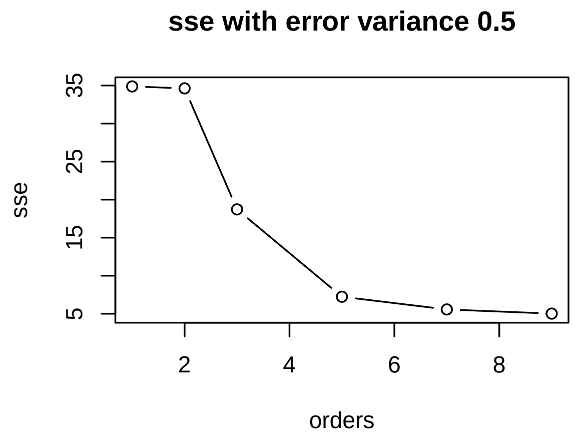
At error variance 0.5, the best model is order 7, At error variance 2, the best model is order 5, At error variance 5, the best model is order 1.

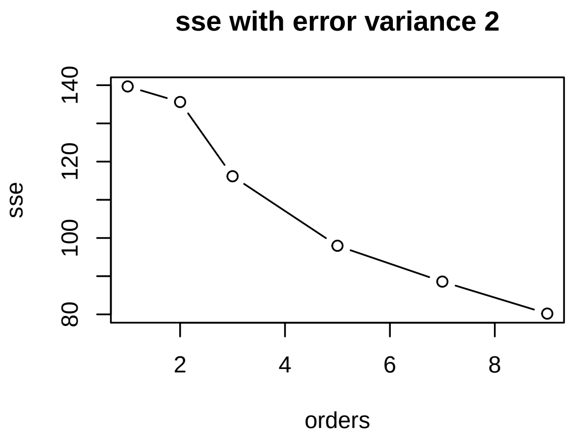
The selection of best model is related with who plays a dominant role in msee. When bias plays a dominant role, we should select the model with smaller bias, which is order 7. When variance plays a dominant role, we should select the model with smaller variance, which is order 1. For error variance 2, we should balance the bias and variance, which is order 5.

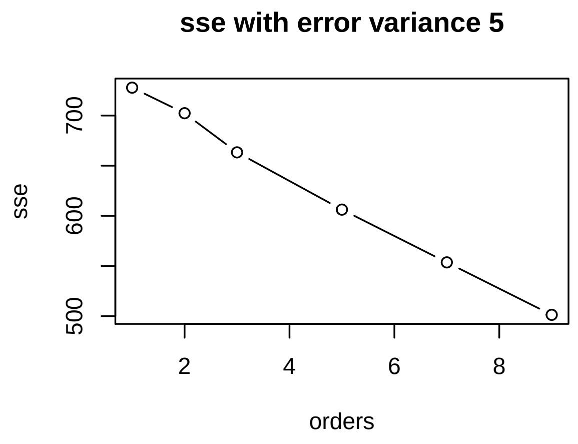
(f)

```
analysis("sse")
```

```
##         0.5         2        5
## 1 34.876531 139.67949 727.8384
## 2 34.621515 135.59924 702.3368
## 3 18.712425 116.16362 663.2994
## 5  7.229235  97.96621 606.1921
## 7  5.559057  88.58094 553.5687
## 9  5.013139  80.20314 501.2640
##
## This is the plot for sse
```







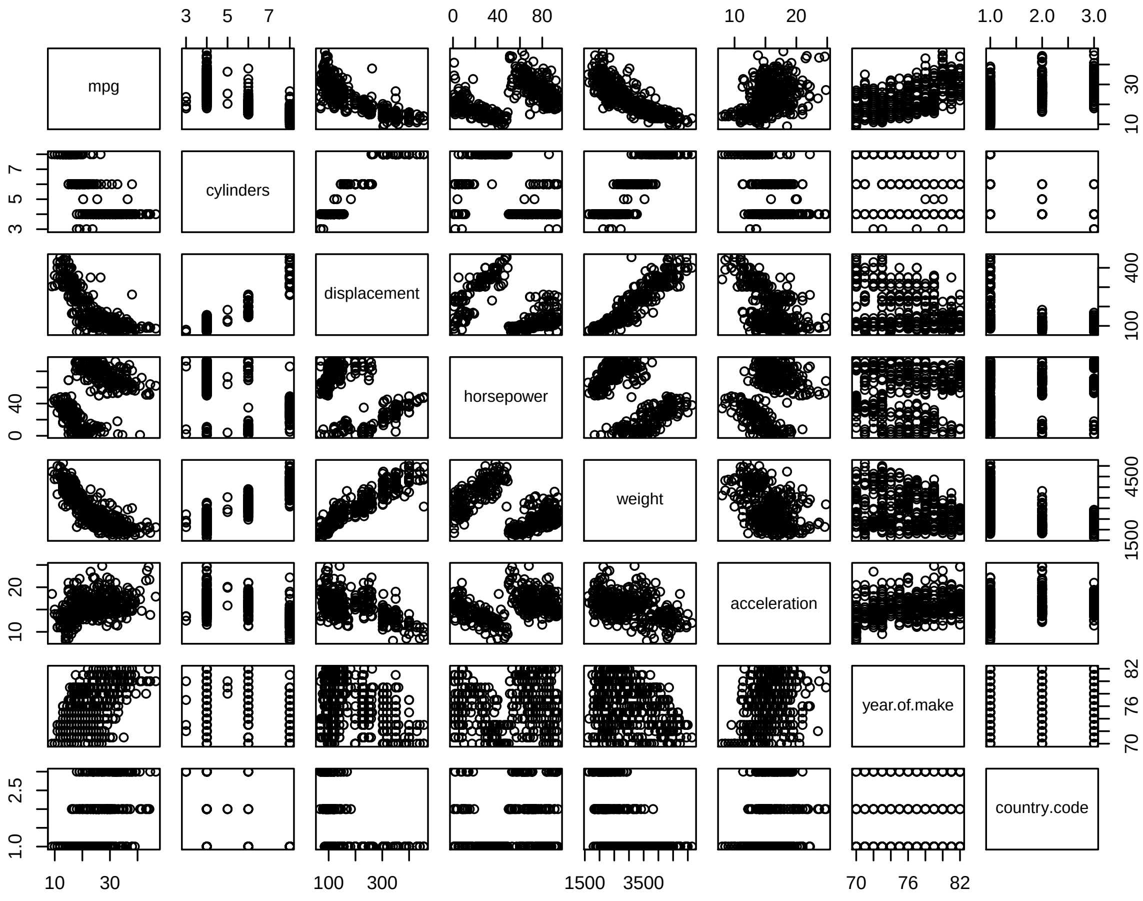
Yes, I observe different patterns. Although all of them are monotonically decreasing, when the noise level is lower, at the beginning and the end, it decrease very slowly, and in the middle, it decrease very quickly. In contrast, when the noise level is higher, it decrease at a constant rate.

The reason, I think, is when noise level is small, the variance does not increase to much, so sse will not increase very much at the beginnning. When the noise level is high, the variance will increase more at the beginning.

Question 3

(a)

```
cars <- read.csv("~/Github/UCDavis/STA206/hw/hw6/Cars.csv", header = T)
pairs(cars)
```



The variable cylinders, year.of.make and country.code are all points on some particular lines, which show strong pattern of categorical variable.

(b)

```r
str(cars)
```

```
## 'data.frame':    397 obs. of  8 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : Factor w/ 94 levels "?","100","102",..: 17 35 29 29 24 42 47 46 48 40 ...
##  $ weight      : int  3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year.of.make: int  70 70 70 70 70 70 70 70 70 70 ...
##  $ country.code: int  1 1 1 1 1 1 1 1 1 1 ...
```

horsepower has been treated as qualitative, but country code has been treated quantitative. In addition, cylinders should also be categorized.

To summary, the variables should be encoded as:

quantitative: mpg, displacement, horsepower, weight, acceleration

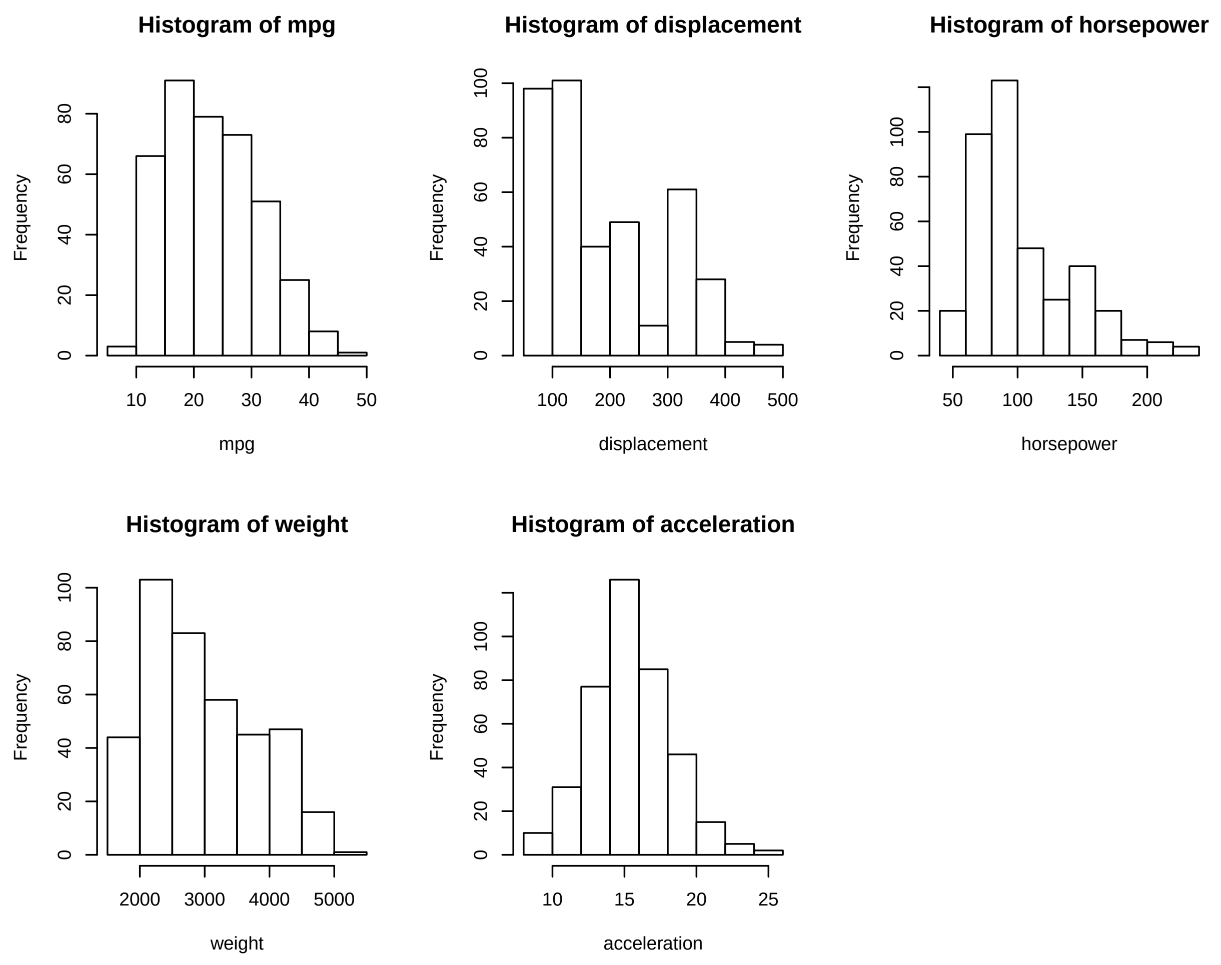
qualitative: cylinders, year.of.make, country.code

(c)

```r
cars <- read.csv("~/Github/UCDavis/STA206/hw/hw6/Cars.csv", header = T,
    na.strings = "?")
cars$country.code <- as.factor(cars$country.code)
cars$cylinders <- as.factor(cars$cylinders)
cars$year.of.make <- as.factor(cars$year.of.make)
```

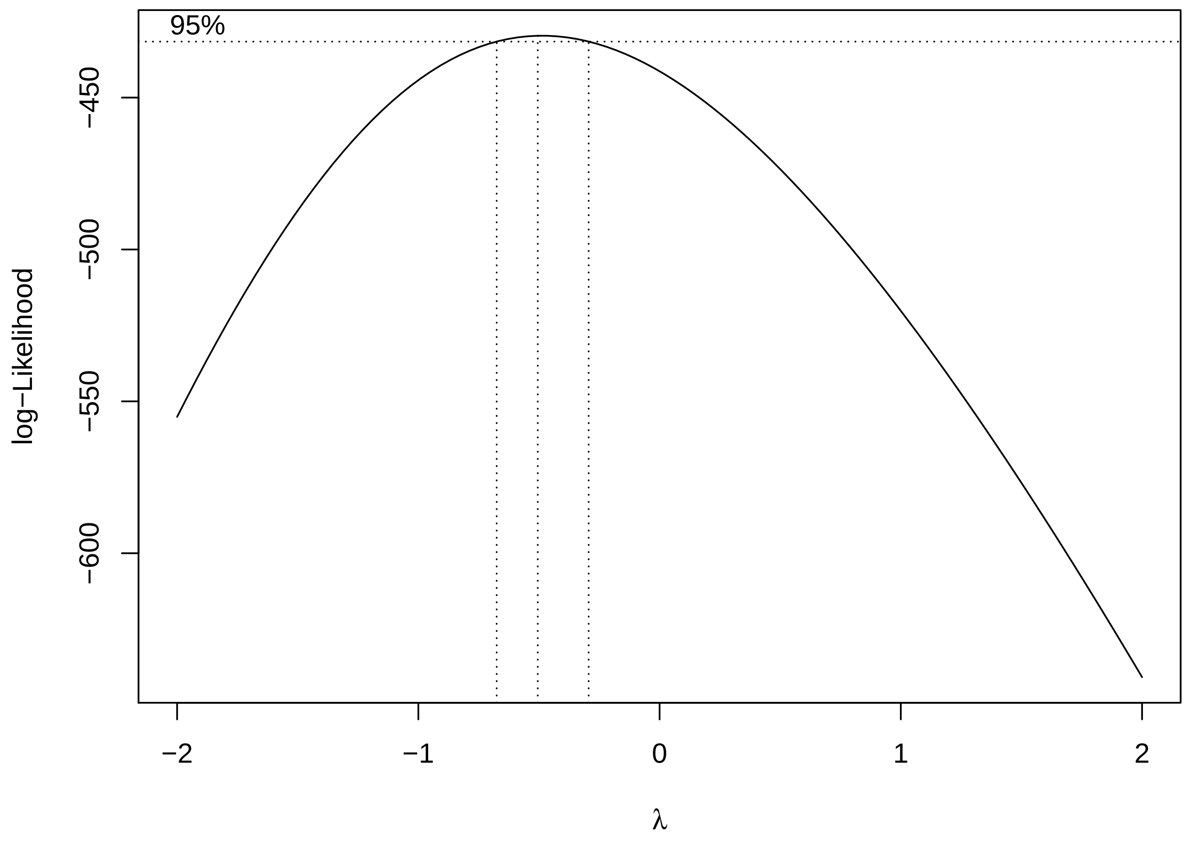
(d)

```r
quantitative_vars <- c("mpg", "displacement", "horsepower", "weight", "acceleration")
qualitative_vars <- c("cylinders", "year.of.make", "country.code")

par(mfrow = c(2, 3))
invisible(lapply(quantitative_vars, function(x) hist(cars[[x]], xlab = x,
    main = paste("Histogram of", x))))
```



Since the data is right skewed, so we want to use log transformation. To verify this, I use boxcox plot:

```r
library(MASS)
boxcox(mpg ~ ., data = cars[, quantitative_vars])
```

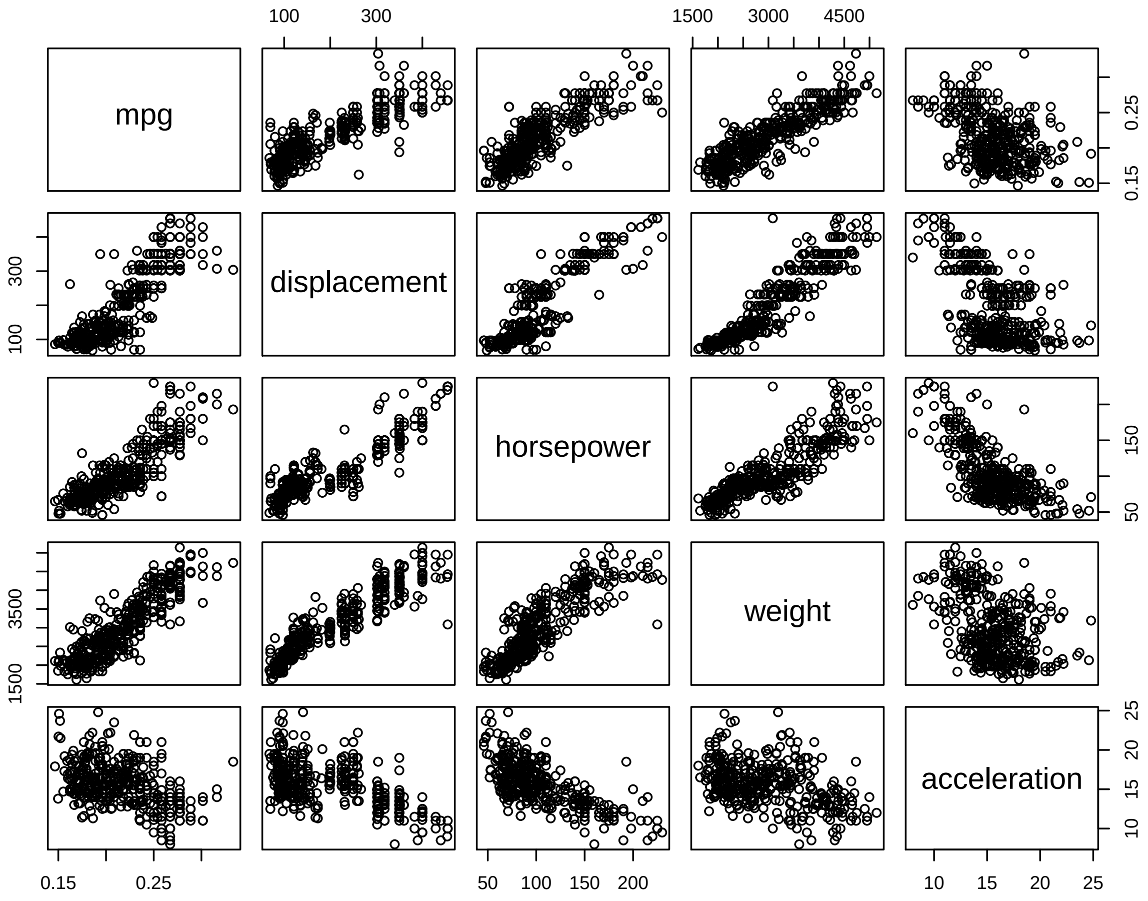


The best order indicated by the plot is -0.5, so negative square root transformation is reasonable.

```r
cars_trans <- cars
cars_trans$mpg <- (cars_trans$mpg)^(-0.5)
```

(e)

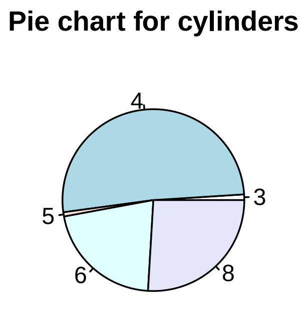
```r
pairs(cars_trans[, quantitative_vars])
```

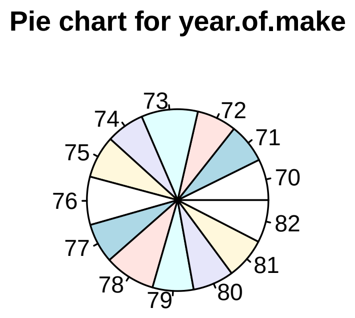


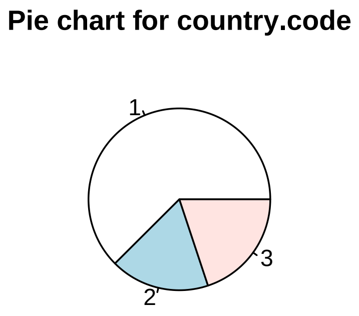
Acceleration is not linear.

(f)

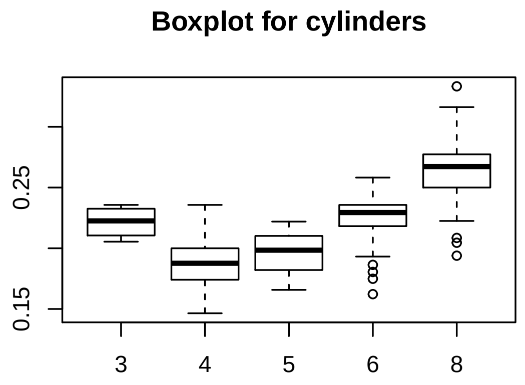
```r
par(mfrow = c(2, 2))
invisible(lapply(qualitative_vars, function(x) pie(table(cars[, x]), main = paste("Pie chart for",
    x))))
```

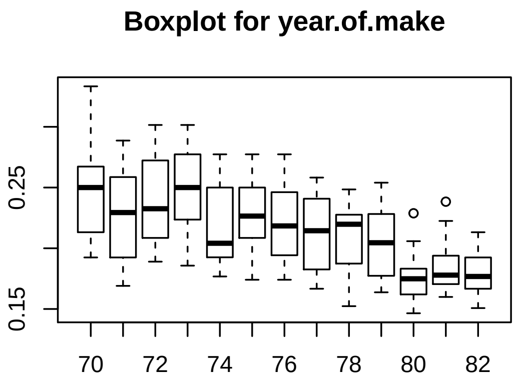


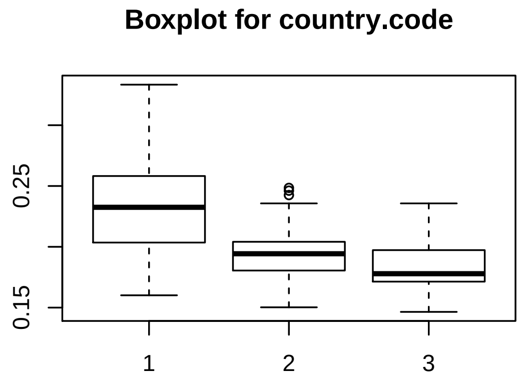




```r
par(mfrow = c(2, 2))
invisible(lapply(qualitative_vars, function(x) boxplot(cars_trans$mpg ~
    cars_trans[, x], main = paste("Boxplot for", x))))
```







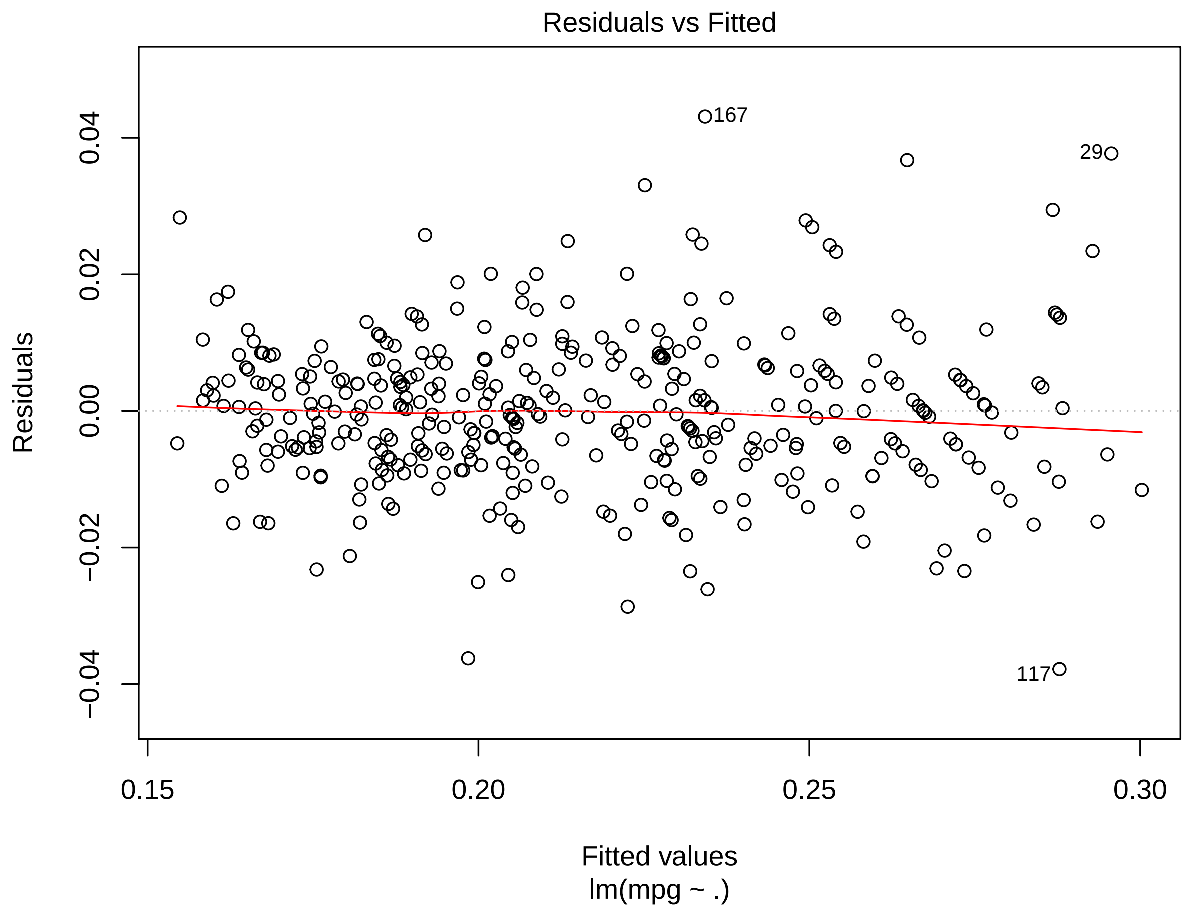
I observe that, the trend for cylinders in downloading, while the trends for year.of.make and country.code is increasing.

(g)

```r
fit1 <- lm(mpg ~ ., data = cars_trans)
summary(fit1)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = cars_trans)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.037803 -0.006423 -0.000168  0.006303  0.043109
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.537e-01  9.356e-03  16.431  < 2e-16 ***
## cylinders4     -2.479e-02  6.089e-03  -4.071 5.74e-05 ***
## cylinders5     -2.546e-02  9.262e-03  -2.749  0.00628 **
## cylinders6     -1.429e-02  6.760e-03  -2.114  0.03517 *
## cylinders8     -1.209e-02  7.802e-03  -1.549  0.12212
## displacement   -4.552e-05  2.685e-05  -1.695  0.09084 .
## horsepower      2.504e-04  5.166e-05   4.847 1.85e-06 ***
## weight          2.251e-05  2.473e-06   9.102  < 2e-16 ***
## acceleration    7.019e-04  3.444e-04   2.038  0.04223 *
## year.of.make71 -3.676e-03  3.232e-03  -1.137  0.25608
## year.of.make72  1.196e-03  3.186e-03   0.375  0.70764
## year.of.make73  3.803e-03  2.859e-03   1.330  0.18428
## year.of.make74 -6.431e-03  3.387e-03  -1.899  0.05839 .
## year.of.make75 -5.717e-03  3.319e-03  -1.723  0.08578 .
## year.of.make76 -8.644e-03  3.178e-03  -2.720  0.00684 **
## year.of.make77 -1.604e-02  3.249e-03  -4.937 1.20e-06 ***
## year.of.make78 -1.548e-02  3.088e-03  -5.014 8.31e-07 ***
## year.of.make79 -2.502e-02  3.269e-03  -7.656 1.70e-13 ***
## year.of.make80 -3.221e-02  3.468e-03  -9.286  < 2e-16 ***
## year.of.make81 -2.664e-02  3.423e-03  -7.783 7.19e-14 ***
## year.of.make82 -3.016e-02  3.366e-03  -8.959  < 2e-16 ***
## country.code2  -5.618e-03  2.046e-03  -2.746  0.00633 **
## country.code3  -6.216e-03  1.969e-03  -3.157  0.00172 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01128 on 369 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.9125, Adjusted R-squared:  0.9073
## F-statistic: 174.9 on 22 and 369 DF,  p-value: < 2.2e-16
```

```
plot(fit1, which = 1)
```

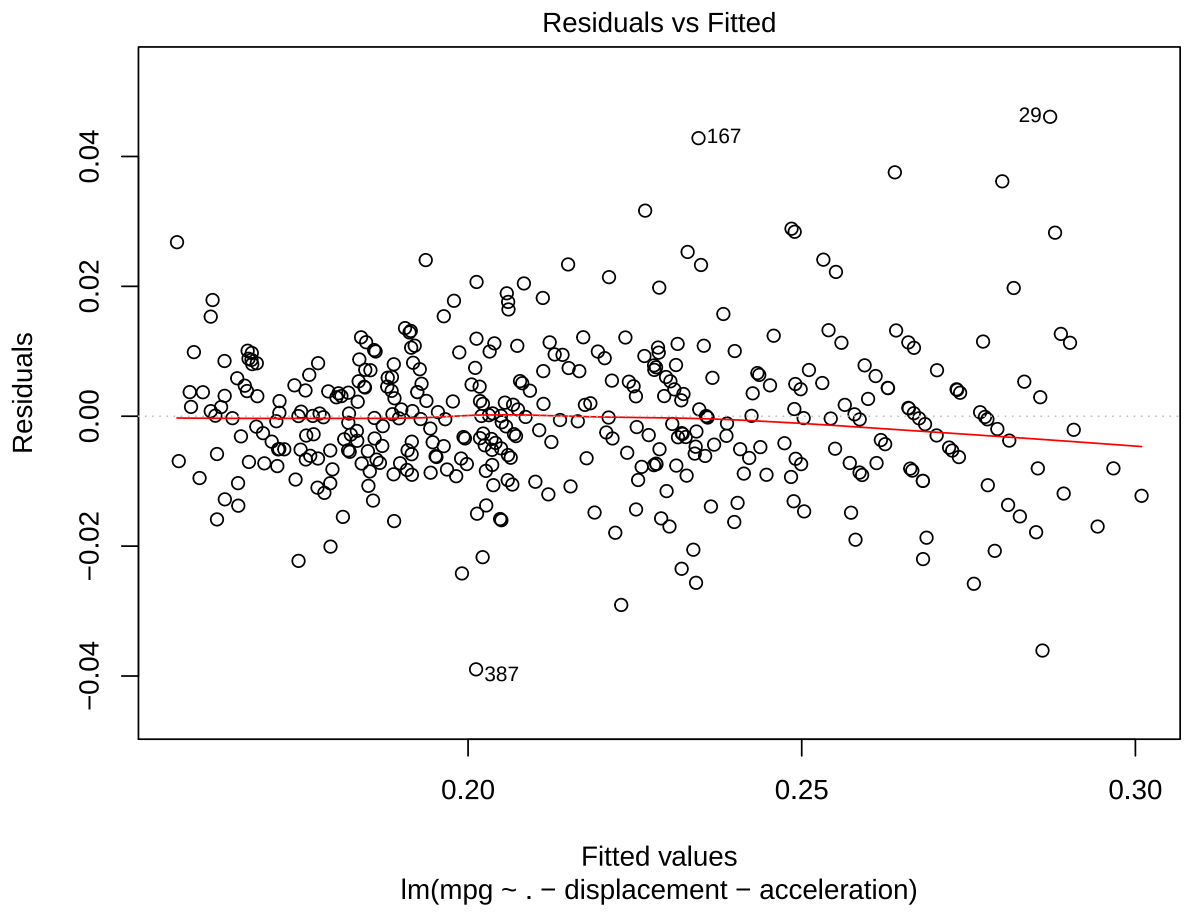


I see that acceleration and displacement has little impact on mpg. So remove it from the model:

```
fit2 <- lm(mpg ~ . - displacement - acceleration, data = cars_trans)
summary(fit2)
```

```
##
## Call:
## lm(formula = mpg ~ . - displacement - acceleration, data = cars_trans)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.038976 -0.006579 -0.000278  0.005918  0.046109
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1.660e-01  7.823e-03  21.226  < 2e-16 ***
## cylinders4      -2.545e-02  5.943e-03  -4.282 2.37e-05 ***
## cylinders5      -2.704e-02  9.132e-03  -2.961  0.00326 **
## cylinders6      -1.786e-02  6.237e-03  -2.863  0.00443 **
## cylinders8      -1.908e-02  6.629e-03  -2.879  0.00423 **
## horsepower       1.572e-04  3.871e-05   4.061 5.96e-05 ***
## weight           2.323e-05  1.977e-06  11.750  < 2e-16 ***
## year.of.make71 -4.616e-03  3.226e-03  -1.431  0.15336
## year.of.make72  1.551e-03  3.184e-03   0.487  0.62653
## year.of.make73  3.584e-03  2.872e-03   1.248  0.21288
## year.of.make74 -6.491e-03  3.363e-03  -1.930  0.05435 .
## year.of.make75 -6.324e-03  3.305e-03  -1.913  0.05647 .
## year.of.make76 -8.823e-03  3.159e-03  -2.793  0.00549 **
## year.of.make77 -1.609e-02  3.234e-03  -4.974 1.00e-06 ***
## year.of.make78 -1.535e-02  3.066e-03  -5.007 8.58e-07 ***
## year.of.make79 -2.520e-02  3.259e-03  -7.733 1.00e-13 ***
## year.of.make80 -3.250e-02  3.471e-03  -9.363  < 2e-16 ***
## year.of.make81 -2.705e-02  3.393e-03  -7.972 1.95e-14 ***
## year.of.make82 -3.039e-02  3.360e-03  -9.044  < 2e-16 ***
## country.code2  -4.309e-03  1.943e-03  -2.218  0.02715 *
## country.code3  -4.949e-03  1.895e-03  -2.612  0.00938 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01137 on 371 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.9106, Adjusted R-squared:  0.9058
## F-statistic:   189 on 20 and 371 DF,  p-value: < 2.2e-16
```

```
plot(fit2, which = 1)
```



The adjusted r^2 decreases only a little bit, so fit2 is preferred than fit1.

The residual plot seems to indicate the model is fitting well now, except the line is a little downwards sloping.

The next step is to consider other models, for instance, regression tree or k-nearest-neighbour. Also, cross validation should be performed to test which model is the best.