MATH3424 HW4

Chow, Hau Cheung Jasper (hcjchow / 20589533)

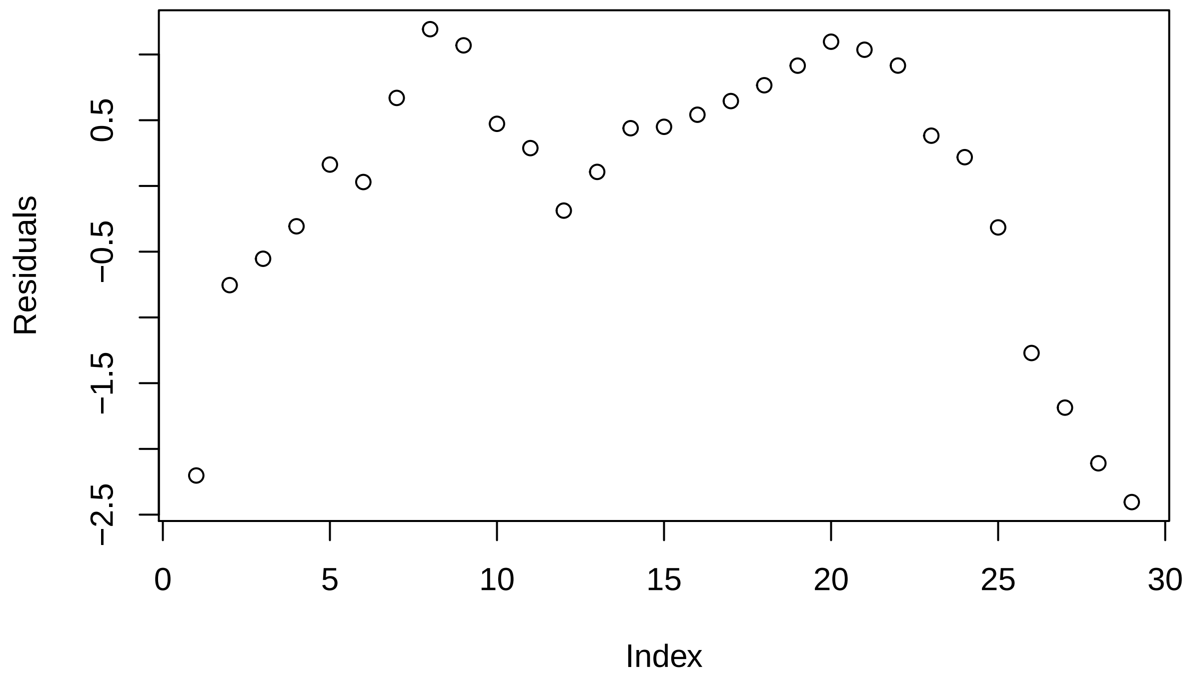
November 15, 2021

Q1

```r
setwd("/Users/jchow/Downloads/MATH3424 R") # need to set the starting directory as this
# since values are comma-separated
crude_oil_data <- read.csv(file="Crude-Oil-Production.txt",header=TRUE)
adv_data <- read.table(file="Advertising.txt",header=TRUE)
gasoline_data <- read.table(file="Gasoline-Consumption.txt",header=TRUE)

crude_oil_data$Barrels = log(crude_oil_data$Barrels) # nat log

crude_oil_model_1 = lm(Barrels ~ ., data = crude_oil_data)
summary(crude_oil_model_1)
```

```
##
## Call:
## lm(formula = Barrels ~ ., data = crude_oil_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.58858 -0.07806  0.07251  0.16624  0.29755
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.119e+02  2.997e+00  -37.33   <2e-16 ***
## Year         6.159e-02  1.538e-03   40.05   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2566 on 27 degrees of freedom
## Multiple R-squared:  0.9834, Adjusted R-squared:  0.9828
## F-statistic:  1604 on 1 and 27 DF,  p-value: < 2.2e-16
```

```r
crude_oil_model_1.residuals <- rstandard(crude_oil_model_1)

plot(seq(1,length(crude_oil_data$Year)), crude_oil_model_1.residuals, xlab="Index", ylab="Residuals")
```



```
library(snpar) # need to install this package
runs.test(crude_oil_model_1$residuals)
```

```
##
##  Approximate runs rest
##
## data:  crude_oil_model_1$residuals
## Runs = 5, p-value = 7.201e-05
## alternative hypothesis: two.sided
```

Q1a) There is a clear pattern in the residuals (close to quadratic) and we may expect this due to time series data usually being positively correlated. From the runs test conducted, we see that the p-value is very small, suggesting we reject the null hypothesis (that the residuals are uncorrelated and random) even at very low significance level values of α. Hence, the alternative hypothesis is true; the errors are clearly autocorrelated.

Q1b)

```
library(car)
```

```
## Loading required package: carData
```

```
durbinWatsonTest(crude_oil_model_1, alternative="positive") # positive, negative or two.sided
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.7337842     0.1945361       0
##  Alternative hypothesis: rho > 0
```

```
n = length(crude_oil_data$Year) # total number of datapoints
dw_manual <- sum((crude_oil_model_1$residuals[-1]-crude_oil_model_1$residuals[-n])^2)
dw_manual <- dw_manual/sum(crude_oil_model_1$residuals^2)
dw_manual
```

```
## [1] 0.1945361
```

The Durbin-Watson statistic is $d = 0.1945361$. As there are $p = 1$ predictor variable(s) and $n = 29$, the critical value is $d_L = 1.12$ at significance level $\alpha = 0.01$ and $d < d_L$. (Alternatively we examine the p-value and find that it is 0.) Therefore we reject $H_0 : \rho = 0$ where ρ is the correlation coefficient between successive error terms, so there IS autocorrelation.

Q1c)

```
n1 <- sum(crude_oil_model_1$residuals > 0)
n2 <- sum(crude_oil_model_1$residuals < 0)
mu <- 2*n1*n2/(n1+n2) + 1
sigma_sq <- 2*n1*n2*(2*n1*n2-n1-n2)/((n1+n2-1)*(n1+n2)^2)

mu
```

```
## [1] 14.10345
```

```
sigma_sq^0.5
```

```
## [1] 2.379953
```

```
rts <- (5-mu)/(sigma_sq^0.5)
rts
```

```
## [1] -3.825054
```

```
1-pnorm(rts, lower.tail=FALSE)
```

```
## [1] 6.537168e-05
```

We observe that $\mu = 14.10345, \sigma = 2.379953$ but the actual number of runs in the data is 5. Computing the run test statistic we get -3.825054, the absolute value of which is larger than the critical value of $z_{1-\alpha/2} = z_{0.975} = 1.96$ when $\alpha = 0.05$ which means we reject the H_0 at significance $\alpha = 0.05$ that the residuals are uncorrelated and random. This means autocorrelation is present. (Alternatively we check the p-value computed in 1a.)

Q1d)

1. Compute OLS estimates of $\hat{\beta}_0$, $\hat{\beta}_1$ by fitting them to data
2. Calculate residuals and from residuals estimate ρ:

```
rho_hat <- sum(crude_oil_model_1$residuals[-1]*crude_oil_model_1$residuals[-n])
rho_hat <- rho_hat/sum(crude_oil_model_1$residuals^2)
```

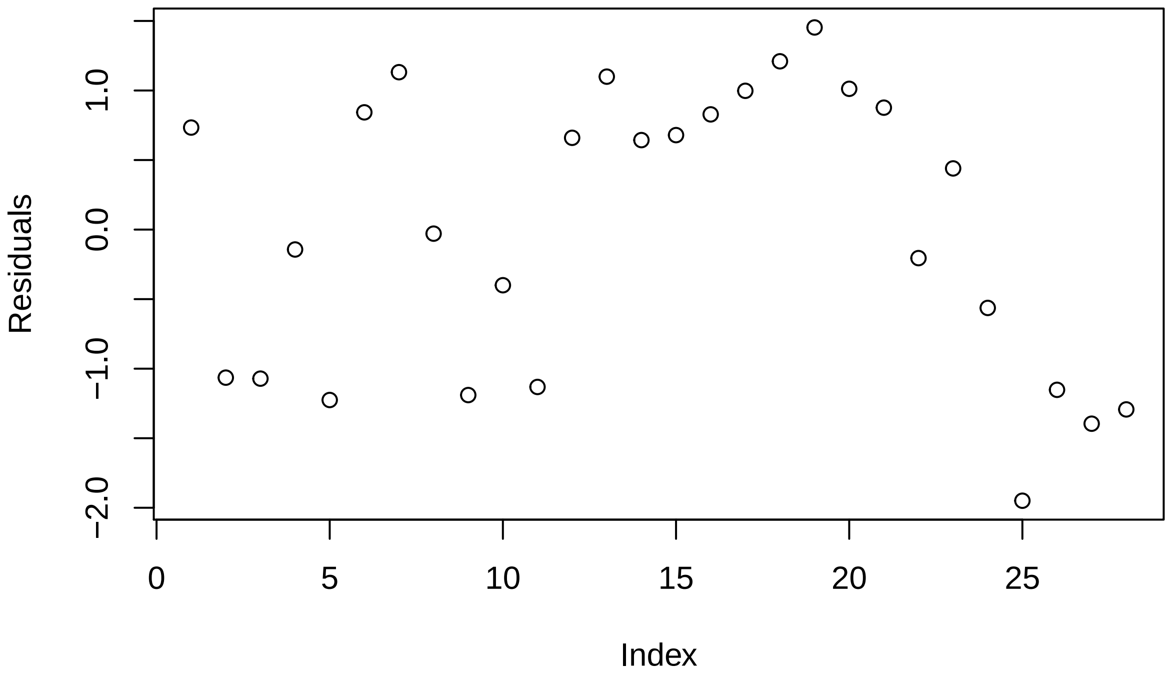
3. Fit $y_t^* = y_t - \hat{\rho} y_{t-1}$ to $x_t^* = x_t - \hat{\rho} x_{t-1}$.

```
x_t_star <- crude_oil_data$Year[-1]-crude_oil_data$Year[-n]*rho_hat
y_t_star <- crude_oil_data$Barrels[-1]-crude_oil_data$Barrels[-n]*rho_hat
crude_oil_model_2 <- lm(y_t_star ~ x_t_star)
summary(crude_oil_model_2)
```

```
##
## Call:
## lm(formula = y_t_star ~ x_t_star)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.18730 -0.10292  0.01986  0.08228  0.14196
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -24.32903    1.58253  -15.37 1.45e-14 ***
## x_t_star      0.05115    0.00303   16.88 1.58e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1001 on 26 degrees of freedom
## Multiple R-squared:  0.9164, Adjusted R-squared:  0.9132
## F-statistic: 284.9 on 1 and 26 DF,  p-value: 1.577e-15
```

4. Examine residuals of newly fitted equation.

```
crude_oil_model_2.residuals <- rstandard(crude_oil_model_2)

# need the -1 since Cochrane-Orcutt fits one less data point
plot(seq(1,length(crude_oil_data$Year)-1), crude_oil_model_2.residuals, xlab="Index", ylab="Residuals")
```



```
durbinWatsonTest(crude_oil_model_2, alternative="positive")
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.5615662     0.8017476       0
##  Alternative hypothesis: rho > 0
```

The residuals plot is slightly improved since there is no longer a pattern in indices 0-10, but there is still a significant pattern in indices 15-25 that indicates autocorrelation may still be present. Further use of the Cochrane-Orcutt procedure may be required.

Q2

```
library(regclass)
```

```
## Loading required package: bestglm

## Loading required package: leaps

## Loading required package: VGAM

## Loading required package: stats4
## Loading required package: splines

##
## Attaching package: 'VGAM'

## The following object is masked from 'package:car':
##
##     logit

## Loading required package: rpart

## Loading required package: randomForest

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

## Important regclass change from 1.3:
## All functions that had a . in the name now have an _
## all.correlations -> all_correlations, cor.demo -> cor_demo, etc.
```

```
adv_data_2 <- adv_data[-1,] # remove the 1st element
adv_data_2$S_.t.1. <- adv_data$S_t[-22] # number of datapoints is 22
# A_{t-1} gets rewritten to A_.t.1.
adv_model_1 = lm(S_t ~ E_t + P_t + A_.t.1. + S_.t.1., data=adv_data_2)
adv_model_2 = lm(S_t ~ E_t + A_t + A_.t.1. + S_.t.1., data=adv_data_2)
adv_model_3 = lm(S_t ~ E_t + A_t + P_t + S_.t.1., data=adv_data_2)
adv_model_4 = lm(S_t ~ E_t + A_t + P_t + A_.t.1., data=adv_data_2)
VIF(adv_model_1)
```

```
##      E_t      P_t  A_.t.1.  S_.t.1.
## 3.455814 1.309504 1.146808 3.490918
```

```
VIF(adv_model_2)
```

```
##      E_t      A_t  A_.t.1.  S_.t.1.
## 3.415835 1.320897 1.051241 3.829540
```

```
VIF(adv_model_3)
```

```
##      E_t      A_t      P_t  S_.t.1.
## 5.440452 2.239380 2.035060 6.644790
```

```
VIF(adv_model_4)
```

```
##      E_t      A_t      P_t  A_.t.1.
## 1.034196 1.316052 1.431257 1.282855
```

```
summary(adv_model_1)
```

```
##
## Call:
## lm(formula = S_t ~ E_t + P_t + A_.t.1. + S_.t.1., data = adv_data_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1683 -0.4420 -0.1603  0.6259  2.4908
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.6187     2.7699   3.834 0.001465 **
## E_t          26.7021     4.2319   6.310 1.04e-05 ***
## P_t           3.7682     0.7487   5.033 0.000123 ***
## A_.t.1.      -0.3630     0.7946  -0.457 0.653975
## S_.t.1.      -0.1617     0.1468  -1.101 0.286978
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.384 on 16 degrees of freedom
## Multiple R-squared:  0.9071, Adjusted R-squared:  0.8838
## F-statistic: 39.05 on 4 and 16 DF,  p-value: 4.589e-08
```

```
summary(adv_model_2)
```

```
##
## Call:
## lm(formula = S_t ~ E_t + A_t + A_.t.1. + S_.t.1., data = adv_data_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5228 -1.4566 -0.1493  1.4656  3.2228
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  26.4958     4.7226   5.610 3.91e-05 ***
## E_t          38.5091     5.7994   6.640 5.68e-06 ***
## A_t          -2.6584     1.1084  -2.398  0.02902 *
## A_.t.1.      -2.1034     1.0487  -2.006  0.06210 .
## S_.t.1.      -0.6685     0.2119  -3.154  0.00614 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.907 on 16 degrees of freedom
## Multiple R-squared:  0.8235, Adjusted R-squared:  0.7793
## F-statistic: 18.66 on 4 and 16 DF,  p-value: 7.162e-06
```

```
summary(adv_model_3)
```

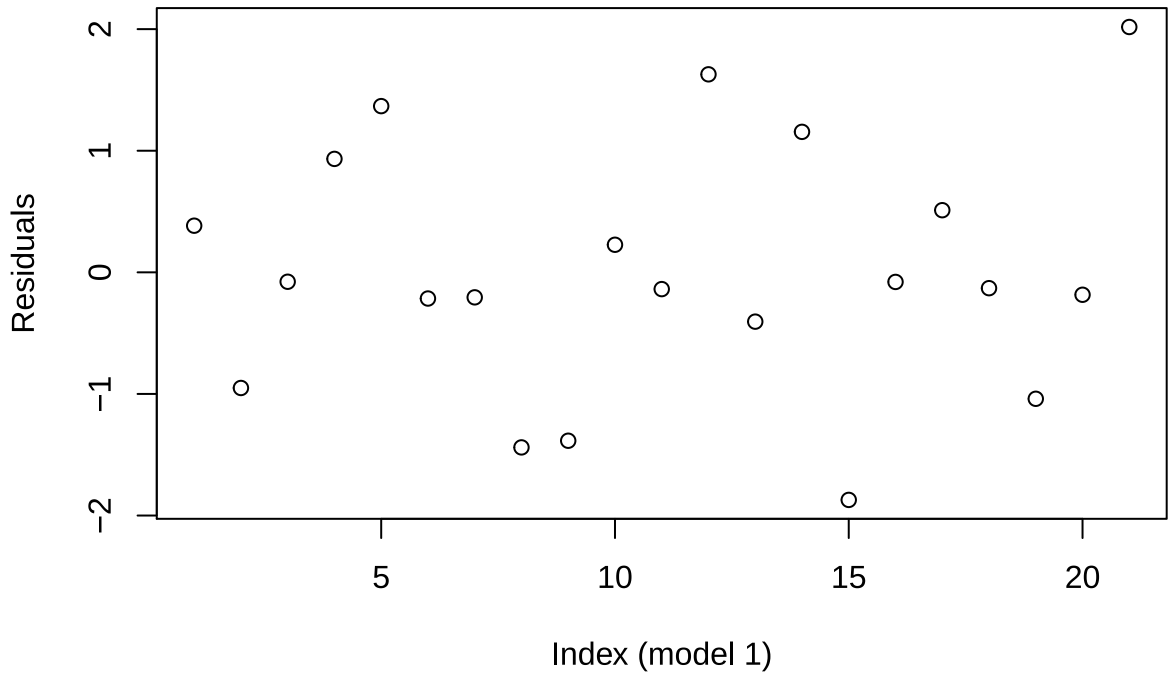
```
##
## Call:
## lm(formula = S_t ~ E_t + A_t + P_t + S_.t.1., data = adv_data_2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.09254 -0.72869  0.03372  0.53785  2.57620
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.27295    4.21999   1.486 0.156590
## E_t         23.71767    5.20043   4.561 0.000321 ***
## A_t          0.97134    1.02549   0.947 0.357625
## P_t          4.44661    0.91416   4.864 0.000172 ***
## S_.t.1.     -0.02675    0.19835  -0.135 0.894390
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 1.355 on 16 degrees of freedom
## Multiple R-squared:  0.9109, Adjusted R-squared:  0.8886
## F-statistic: 40.88 on 4 and 16 DF,  p-value: 3.302e-08
```

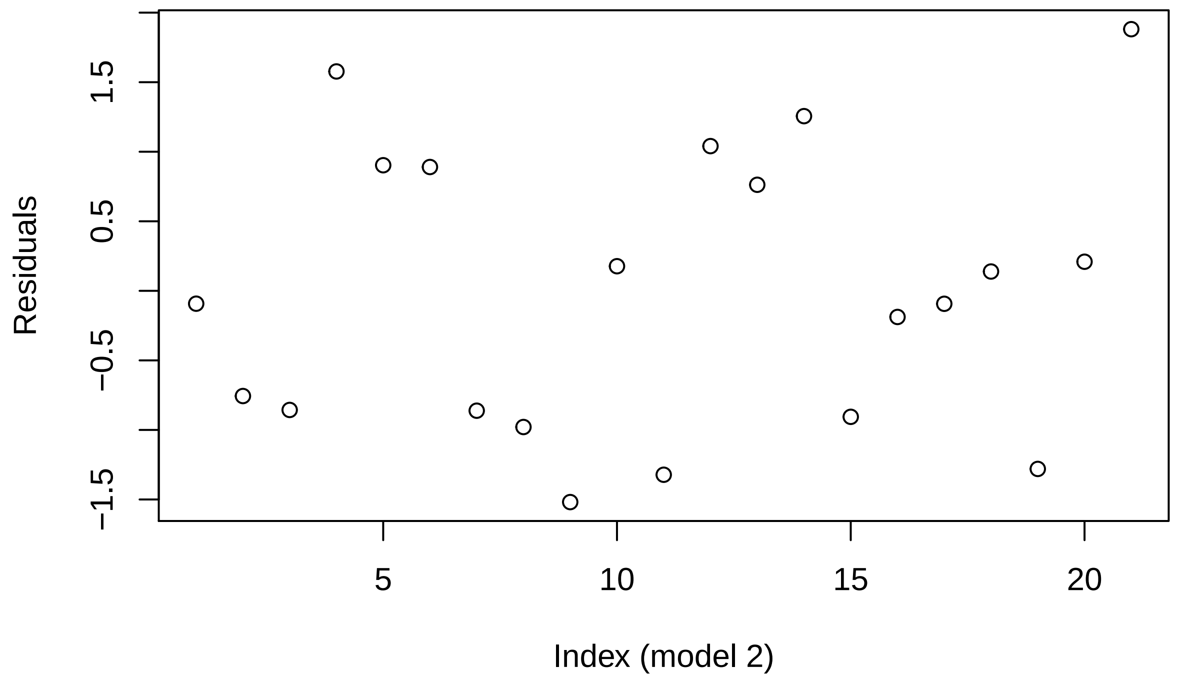
```
summary(adv_model_4)
```

```
##
## Call:
## lm(formula = S_t ~ E_t + A_t + P_t + A_.t.1., data = adv_data_2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.12547 -0.79269  0.05501  0.56819  2.54779
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.52829    3.04705   1.814   0.0884 .
## E_t         23.11710    2.26790  10.193 2.10e-08 ***
## A_t          1.09415    0.78633   1.391   0.1831
## P_t          4.56082    0.76682   5.948 2.05e-05 ***
## A_.t.1.      0.08546    0.82333   0.104   0.9186
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.355 on 16 degrees of freedom
## Multiple R-squared:  0.9108, Adjusted R-squared:  0.8885
## F-statistic: 40.86 on 4 and 16 DF,  p-value: 3.314e-08
```

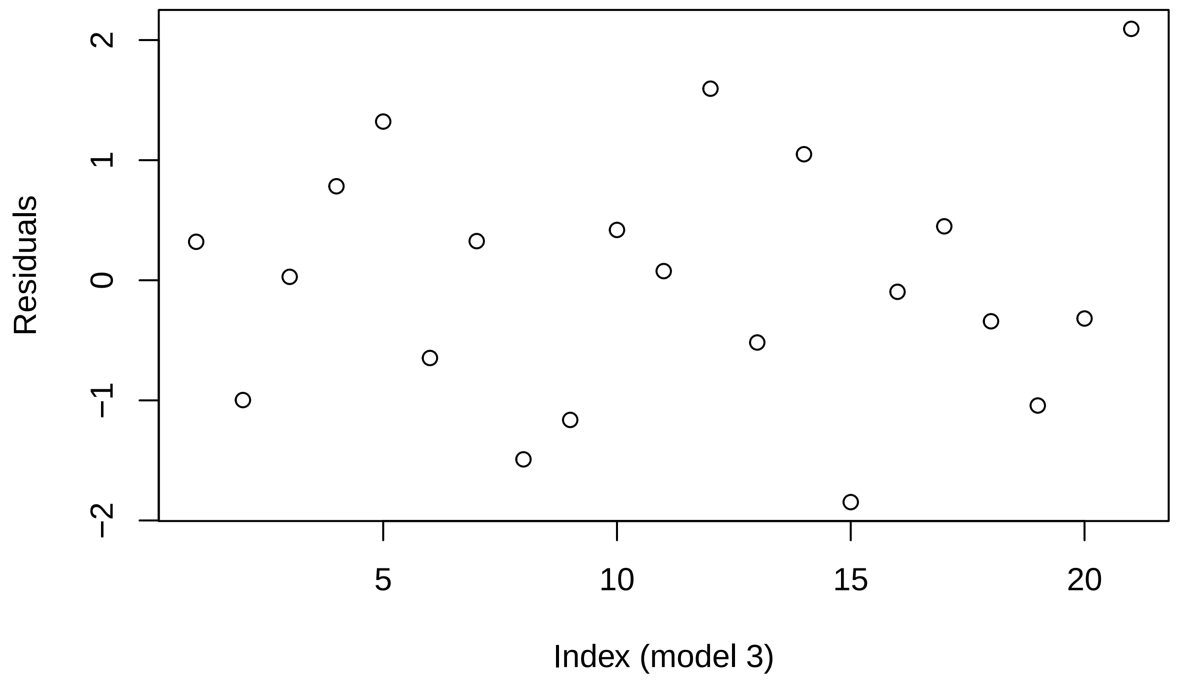
```
plot(seq(1,length(adv_data_2$S_t)), rstandard(adv_model_1), xlab="Index (model 1)", ylab="Residuals")
```



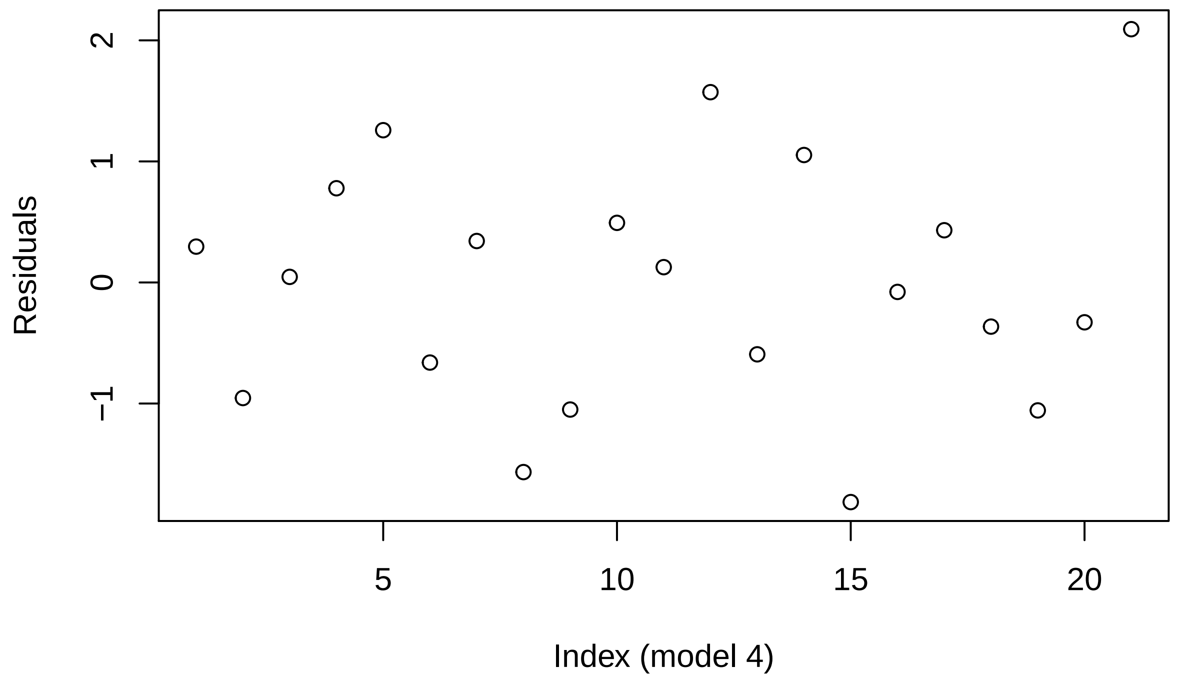
```
plot(seq(1,length(adv_data_2$S_t)), rstandard(adv_model_2), xlab="Index (model 2)", ylab="Residuals")
```



```
plot(seq(1,length(adv_data_2$S_t)), rstandard(adv_model_3), xlab="Index (model 3)", ylab="Residuals")
```



```
plot(seq(1,length(adv_data_2$S_t)), rstandard(adv_model_4), xlab="Index (model 4)", ylab="Residuals")
```

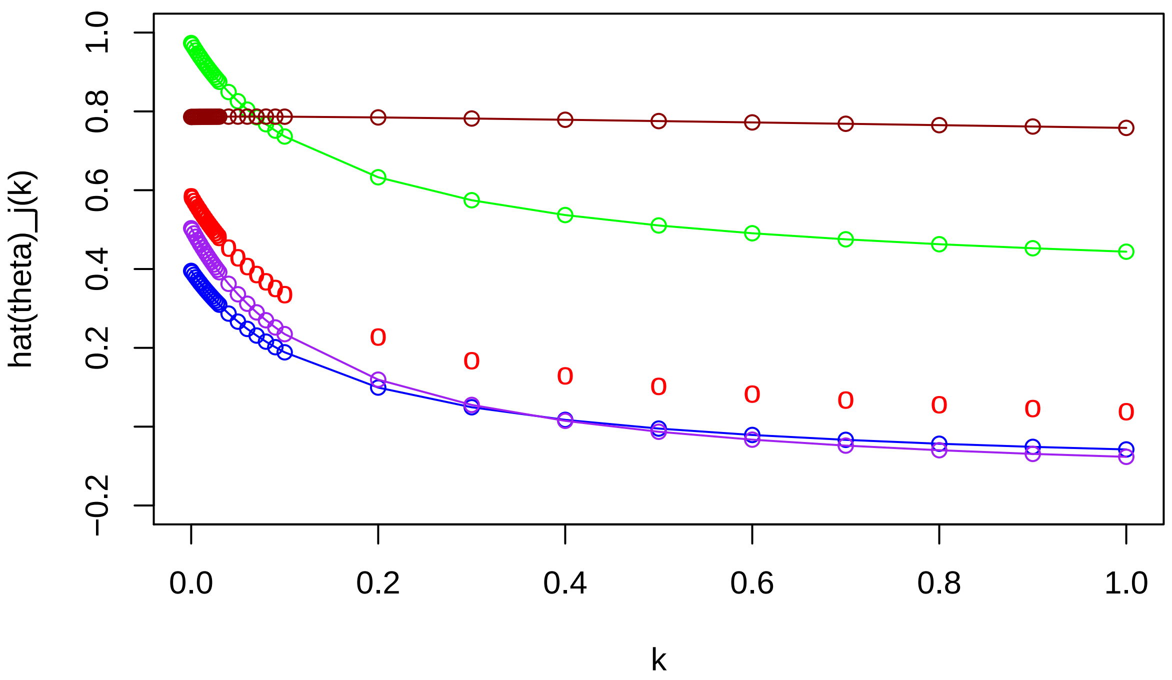


We observe that in all cases, the models have high R^2 and high F statistic and the index vs standard residuals plot are good (show no pattern). Since the VIF for each predictor in each of the 4 models are now all less than 10, we can say that collinearity has been removed in each case. Overall we observe model `adv_model_4` seems to be the best since all of its VIFs are very close to 1.

Q3

```r
p = 5 # number of predictors
standardize <- function(column){
  return ((column-mean(column))/sd(column))
}
adv_data_stand <- apply(adv_data, 2, function(x) (x-mean(x))/sd(x))
#test_data_stand <- apply(adv_data, 2, standardize)
y_stand <- adv_data_stand[,1]
x_stand <- adv_data_stand[,-1]
I_p <- diag(p)

k_1_vec <- c(0.000, 0.001, 0.003, 0.005, 0.007, 0.009)
k_2_vec <- seq(from=0.01, to=0.03, by=0.002)
k_3_vec <- seq(from=0.04, to=0.09, by=0.01)
k_4_vec <- seq(from=0.1, to=1, by=0.1)
k_vec <- c(k_1_vec, k_2_vec, k_3_vec, k_4_vec)

# create a matrix 'df_theta' to hold the regression parameters for each value of k
df_theta <- matrix(data=NA, nrow=length(k_vec), ncol=p)
for (index in (1:length(k_vec)))
{
  k = k_vec[index]
  theta_hat <- solve(t(x_stand) %*% x_stand + k*I_p) %*% t(x_stand) %*% y_stand
  df_theta[index,] = theta_hat
}
```

```
#for (col_index in (1:p))
#{ # separate plots
#  plot(k_vec, df_theta[,col_index], xlab="bias parameter k", #ylab="hat(theta)_j(k)")
#}

# set ylimit to (-0.2,1) interval. red=theta_1(k), green=theta_2(k), etc.
colours = c("red", "green", "dark red", "blue", "purple")
plot(k_vec, df_theta[,1], col=colours[1], pch="o", xlab="k", ylab="hat(theta)_j(k)", ylim=c(-0.2,1))
for (col_index in (2:p))
{
  points(k_vec, df_theta[,col_index], col=colours[col_index], lty=1)
  lines(k_vec, df_theta[,col_index], col=colours[col_index], lty=1)
}
```



We want to select the smallest value of k such that ALL the regression coefficients are beginning to stabilise. From the ridge trace, we can see this occurs at approximately $k = 0.65$.

Q3b)

```
df_theta[1,]
```

```
## [1] 0.5830280 0.9734160 0.7858832 0.3952872 0.5034937
```

```
adv_model_5 <- lm(S_t ~ ., data=as.data.frame(adv_data_stand))
summary(adv_model_5)
```

```
##
## Call:
## lm(formula = S_t ~ ., data = as.data.frame(adv_data_stand))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.46540 -0.24639  0.03309  0.17557  0.55161
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.720e-16  7.041e-02   0.000   1.0000
## A_t          5.830e-01  4.380e-01   1.331   0.2019
## P_t          9.734e-01  4.170e-01   2.334   0.0329 *
## E_t          7.859e-01  7.476e-02  10.512 1.36e-08 ***
## A_.t.1.      3.953e-01  3.669e-01   1.077   0.2973
## P_.t.1.      5.035e-01  4.755e-01   1.059   0.3053
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3303 on 16 degrees of freedom
## Multiple R-squared:  0.9169, Adjusted R-squared:  0.8909
## F-statistic:  35.3 on 5 and 16 DF,  p-value: 4.289e-08
```

```r
n <- length(y_stand)
epsilon = 0.00001 # user defined threshold

# (fitted-actual)^2/16, remember to use standardised data!
sigmahatsq_0 = sum((adv_model_5$fitted.values-y_stand)^2)/(n-p-1)
numerator = p*sigmahatsq_0
denom = sum((df_theta[1,])^2)

k_0 = numerator/denom
last_k_i = k_0+1 # ensures first iteration never converges
k_i = k_0
count = 0

while (TRUE)
{
  theta_hat <- solve(t(x_stand) %*% x_stand + k_i*I_p) %*% t(x_stand) %*% y_stand
  denom = sum((theta_hat)^2)
  k_i = numerator/denom
  if (abs(k_i-last_k_i) <= epsilon){
    break
  }
  last_k_i = k_i
  count <- count+1
}


theta_hat_iterative_method_k <- solve(t(x_stand) %*% x_stand + k_i*I_p) %*% t(x_stand) %*% y_stand

k_i
```

```
## [1] 0.6544365
```

```r
count
```

```
## [1] 7
```

```r
sd_all <- apply(adv_data[,-1], 2, sd)
mean_all <- apply(adv_data[,-1], 2, mean)
beta_1_to_j <- theta_hat_iterative_method_k*sd(adv_data$S_t)/sd_all # beta_j = theta_j*(sy/sj)
beta_0 <- mean(adv_data$S_t) - sum(mean_all*beta_1_to_j) # beta_0 = mean(y) - sum(mean(x_j)*beta_j)
beta_original <- rbind(beta_0, beta_1_to_j)
```

```
beta_original
```

```
##                [,1]
## beta_0   8.0130695
## A_t      0.6772603
## P_t      4.1451377
## E_t     22.0706034
## A_.t.1. -0.2752962
## P_.t.1. -0.3420478
```

```
summary(lm(S_t ~ ., data=adv_data))
```

```
##
## Call:
## lm(formula = S_t ~ ., data = adv_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8601 -0.9848  0.1323  0.7017  2.2046
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -14.194     18.715  -0.758   0.4592
## A_t            5.361      4.028   1.331   0.2019
## P_t            8.372      3.586   2.334   0.0329 *
## E_t           22.521      2.142  10.512 1.36e-08 ***
## A_.t.1.        3.855      3.578   1.077   0.2973
## P_.t.1.        4.125      3.895   1.059   0.3053
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.32 on 16 degrees of freedom
## Multiple R-squared:  0.9169, Adjusted R-squared:  0.8909
## F-statistic:  35.3 on 5 and 16 DF,  p-value: 4.289e-08
```

Compared to the OLS estimates, we see that the coefficient for E_t is largely the same, but the coefficients for $A_t, P_t, A_{t-1}, P_{t-1}$ are significantly different. This matches our conclusion with the ridge trace approach; only the collinear variables have unstable regression coefficients and in Q2 and in the lectures we had determined that E_t was not collinear to any other variable but the other variables were, so its coefficient would not change much when k changed, but the other variables' coefficients would change significantly when k changed.

Q4

Q4a) No, as there is multicollinearity present. An examination of the correlation matrix (conducted below) between predictors reveals some variables have extremely high correlation magnitudes (such as X_3 and X_2, X_3 and X_1, X_{10} and X_1 etc.) Overall, X_1, X_2, X_3 seem to be strongly related to each other, and X_{10} appears to be strongly related to both X_8 and X_1.

```
cor(gasoline_data)
```

```
##                Y        X_1        X_2        X_3         X_4        X_5
## Y      1.0000000 -0.8718188 -0.7965605 -0.8493416  0.42241460  0.6352323
## X_1   -0.8718188  1.0000000  0.9406456  0.9895851 -0.34958682 -0.6714311
## X_2   -0.7965605  0.9406456  1.0000000  0.9643592 -0.28989951 -0.5509642
## X_3   -0.8493416  0.9895851  0.9643592  1.0000000 -0.32599915 -0.6728661
## X_4    0.4224146 -0.3495868 -0.2898995 -0.3259992  1.00000000  0.4137808
## X_5     0.6352323 -0.6714311 -0.5509642 -0.6728661  0.41378081  1.0000000
## X_6    -0.4719210  0.6399642  0.7614190  0.6531263  0.03748643 -0.2195283
## X_7     0.7078714 -0.7717815 -0.6259445 -0.7461800  0.55823570  0.8717662
## X_8    -0.7523967  0.8649023  0.8027387  0.8641224 -0.30415026 -0.5613315
## X_9    -0.7624550  0.8001582  0.7105117  0.7881284 -0.37817358 -0.4534470
## X_.10. -0.8525706  0.9531271  0.8878810  0.9434871 -0.35845879 -0.5798617
## X_.11. -0.7216882  0.8241409  0.7086735  0.8012765 -0.44054570 -0.7546650
##               X_6        X_7        X_8        X_9     X_.10.     X_.11.
## Y      -0.47192100  0.7078714 -0.7523967 -0.7624550 -0.8525706 -0.7216882
## X_1     0.63996417 -0.7717815  0.8649023  0.8001582  0.9531271  0.8241409
## X_2     0.76141897 -0.6259445  0.8027387  0.7105117  0.8878810  0.7086735
## X_3     0.65312630 -0.7461800  0.8641224  0.7881284  0.9434871  0.8012765
## X_4     0.03748643  0.5582357 -0.3041503 -0.3781736 -0.3584588 -0.4405457
## X_5    -0.21952829  0.8717662 -0.5613315 -0.4534470 -0.5798617 -0.7546650
## X_6     1.00000000 -0.2756386  0.4220680  0.3003862  0.5203669  0.3954893
## X_7    -0.27563863  1.0000000 -0.6552065 -0.6551300 -0.7058126 -0.8506963
## X_8     0.42206800 -0.6552065  1.0000000  0.8831512  0.9554541  0.6824919
## X_9     0.30038618 -0.6551300  0.8831512  1.0000000  0.8994711  0.6326677
## X_.10.  0.52036693 -0.7058126  0.9554541  0.8994711  1.0000000  0.7530353
## X_.11.  0.39548928 -0.8506963  0.6824919  0.6326677  0.7530353  1.0000000
```

Q4b)

For Mallow's C_p, we need to choose a full model such that all proposed models are reduced forms of it. We will use the full model defined as follows: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_5 + \beta_4 X_8 + \beta_5 X_{10} + \epsilon$.

```r
gasoline_model_1 <- lm(Y ~ X_1, data=gasoline_data)
gasoline_model_2 <- lm(Y ~ X_.10., data=gasoline_data)
gasoline_model_3 <- lm(Y ~ X_1 + X_.10., data=gasoline_data)
gasoline_model_4 <- lm(Y ~ X_2 + X_.10., data=gasoline_data)
gasoline_model_5 <- lm(Y ~ X_8 + X_.10., data=gasoline_data)
gasoline_model_6 <- lm(Y ~ X_8 + X_5 + X_.10., data=gasoline_data)
```

```r
library(stats)
library(olsrr)
```

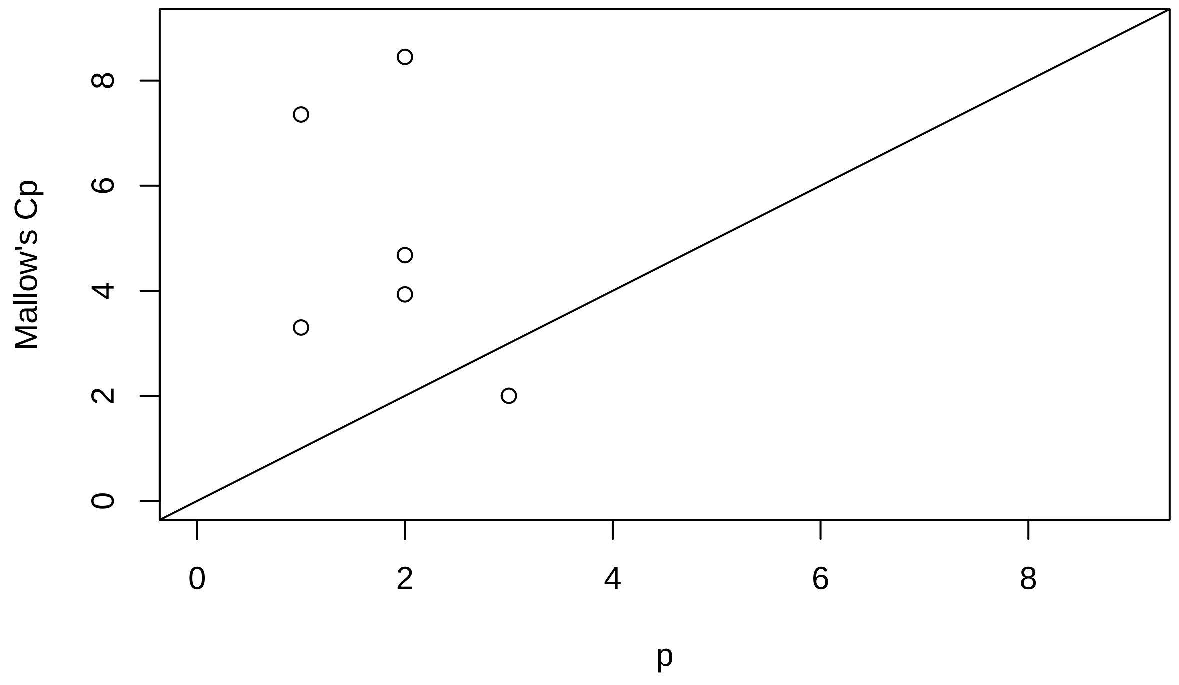
```
##
## Attaching package: 'olsrr'

## The following object is masked from 'package:datasets':
##
##     rivers
```

```r
gasoline_model_full <- lm(Y ~ X_1+X_2+X_5+X_8+X_.10., data=gasoline_data)
crit <- data.frame(model_a=rep(0,4),
                   model_b=rep(0,4),
                   model_c=rep(0,4),
                   model_d=rep(0,4),
                   model_e=rep(0,4),
                   model_f=rep(0,4))
row.names(crit) <- c("Adjusted R^2", "Mallow's Cp", "AIC", "BIC")
crit[1,] <- c(summary(gasoline_model_1)$adj.r.squared,
              summary(gasoline_model_2)$adj.r.squared,
              summary(gasoline_model_3)$adj.r.squared,
              summary(gasoline_model_4)$adj.r.squared,
              summary(gasoline_model_5)$adj.r.squared,
              summary(gasoline_model_6)$adj.r.squared)
```

```r
crit[2,] <- c(ols_mallows_cp(gasoline_model_1,gasoline_model_full),
              ols_mallows_cp(gasoline_model_2,gasoline_model_full),
              ols_mallows_cp(gasoline_model_3,gasoline_model_full),
              ols_mallows_cp(gasoline_model_4,gasoline_model_full),
              ols_mallows_cp(gasoline_model_5,gasoline_model_full),
              ols_mallows_cp(gasoline_model_6,gasoline_model_full))
crit[3,] <- c(AIC(gasoline_model_1),
              AIC(gasoline_model_2),
              AIC(gasoline_model_3),
              AIC(gasoline_model_4),
              AIC(gasoline_model_5),
              AIC(gasoline_model_6))
crit[4,] <- c(BIC(gasoline_model_1),
              BIC(gasoline_model_2),
              BIC(gasoline_model_3),
              BIC(gasoline_model_4),
              BIC(gasoline_model_5),
              BIC(gasoline_model_6))
crit
```

```
##                 model_a     model_b     model_c     model_d     model_e
## Adjusted R^2   0.751499   0.7171223   0.7477775   0.7145943   0.7543429
## Mallow's Cp    3.301341   7.3547842   4.6780096   8.4509688   3.9315085
## AIC          157.374293 161.2613314 158.7292132 162.4372038 157.9379565
## BIC          161.577886 165.4649235 164.3340028 168.0419934 163.5427461
##                 model_f
## Adjusted R^2   0.7807823
## Mallow's Cp    2.0021557
## AIC          155.3896042
## BIC          162.3955911
```

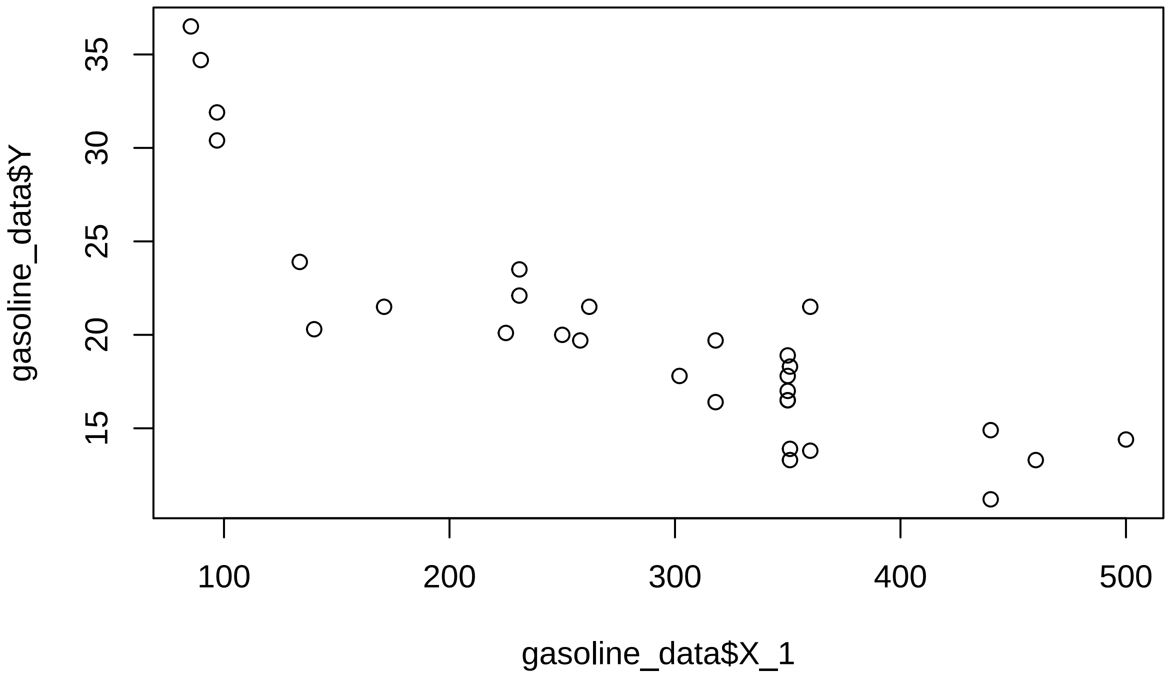
```r
plot(c(1,1,2,2,2,3),crit[2,],xlim=c(0,9),ylim=c(0,9),xlab="p",ylab="Mallow's Cp")
abline(a=0,b=1)
```



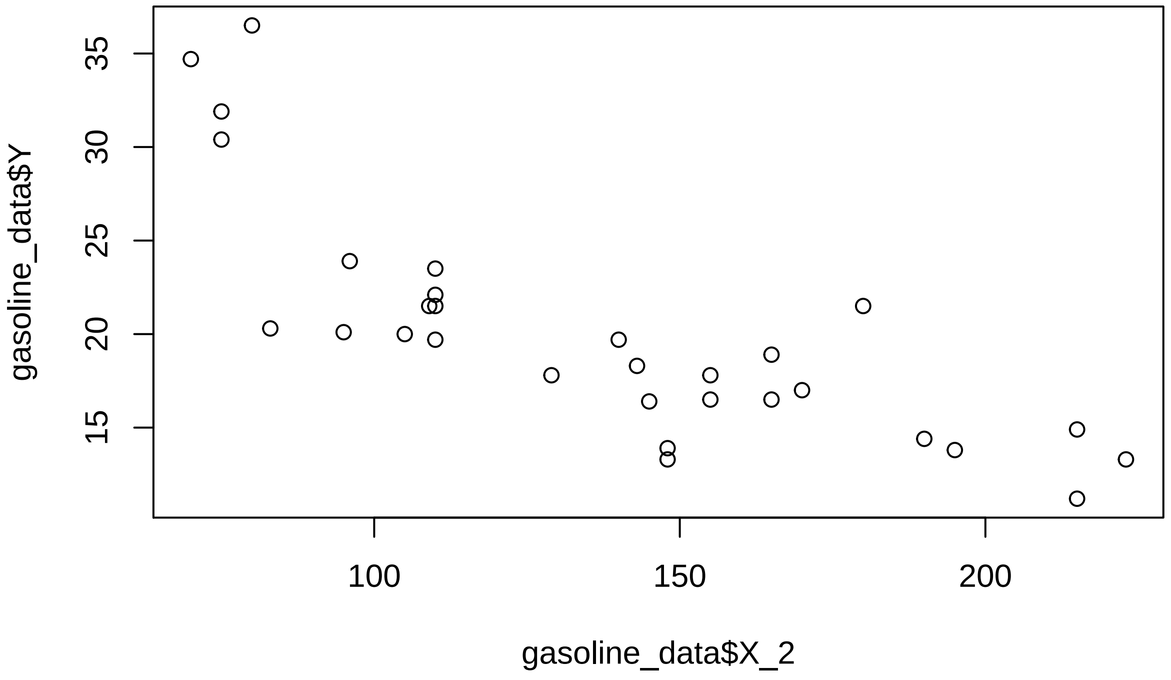
The model with the highest adjusted R-squared is model (f). The model with the smallest Mallow's C_p is preferred, and that is also model (f) (we also see from the plot that model (f) is the model whose (p, C_p) point is closest to the line with intercept 0 and slope 1 (the desirable criteria.)) Models with smaller AIC and BIC are preferred. The model with the smallest AIC is model (f) and the model with the smallest BIC is (a), although model (f) has the 2nd smallest BIC. In conclusion, the best model is likely model (f).

Q4c)

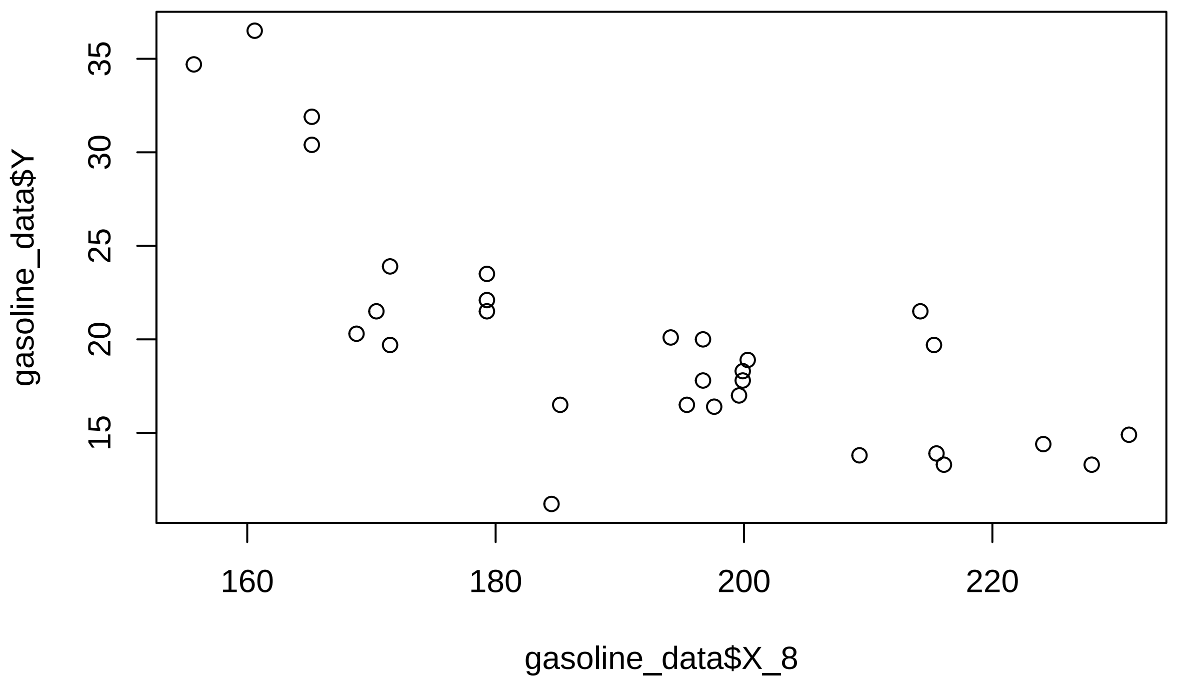
```
plot(gasoline_data$X_1, gasoline_data$Y)
```



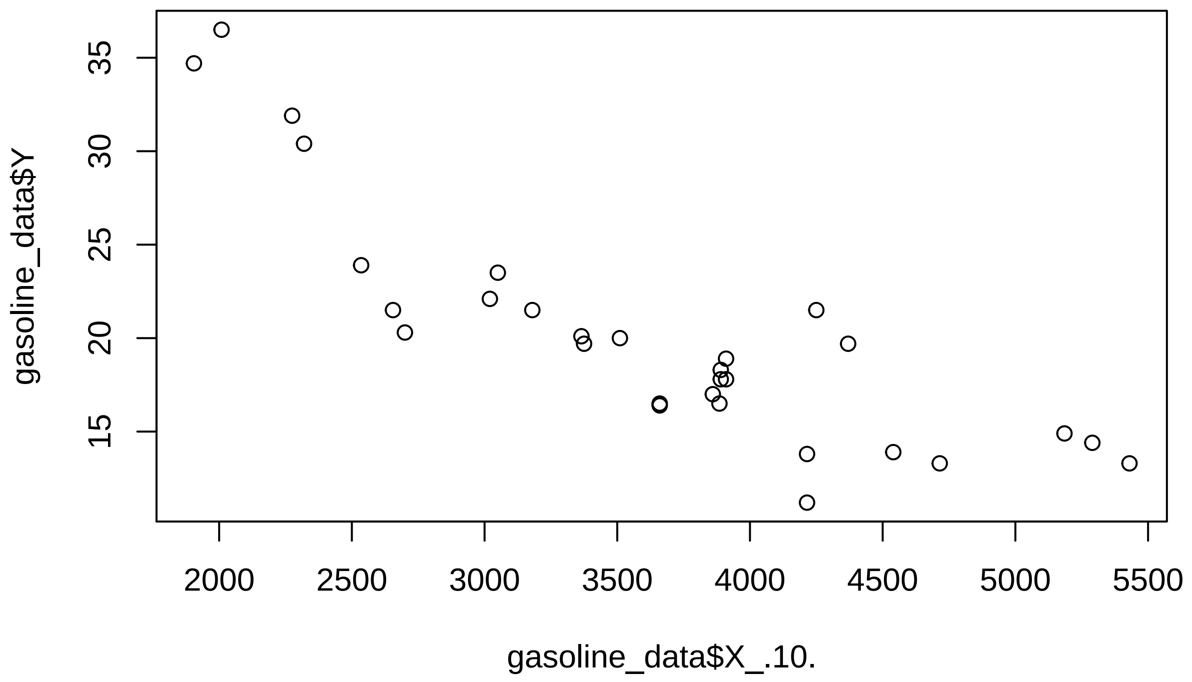
```
plot(gasoline_data$X_2, gasoline_data$Y)
```



```
plot(gasoline_data$X_8, gasoline_data$Y)
```



```
plot(gasoline_data$X_.10., gasoline_data$Y)
```



The plots show that Y tends to decrease as the values of the predictors increase but the rate at which they decrease gets smaller the larger the values of the predictor. It may suggest that the relationship between Y and these predictors is not necessarily linear.

Q4d)

1. Start with intercept-only model.

```r
t_cutoff_df = length(gasoline_data$Y)-2
t_cutoff <- qt(0.025, df=t_cutoff_df, lower.tail=FALSE)
t_cutoff
```

```
## [1] 2.048407
```

```r
gasoline_data_4d = subset(gasoline_data, select=c("Y","X_1","X_2","X_5","X_8","X_.10."))
gasoline_forward_0 = lm(Y ~ 1, data=gasoline_data)

# ignore correlation with itself, choose predictor with highest magnitude of correlation with response
cor(gasoline_data_4d)
```

```
##                 Y        X_1        X_2        X_5        X_8     X_.10.
## Y       1.0000000 -0.8718188 -0.7965605  0.6352323 -0.7523967 -0.8525706
## X_1    -0.8718188  1.0000000  0.9406456 -0.6714311  0.8649023  0.9531271
## X_2    -0.7965605  0.9406456  1.0000000 -0.5509642  0.8027387  0.8878810
## X_5     0.6352323 -0.6714311 -0.5509642  1.0000000 -0.5613315 -0.5798617
## X_8    -0.7523967  0.8649023  0.8027387 -0.5613315  1.0000000  0.9554541
## X_.10. -0.8525706  0.9531271  0.8878810 -0.5798617  0.9554541  1.0000000
```

```r
names(which.max(abs(cor(gasoline_data_4d)[1,-1])))
```

```
## [1] "X_1"
```

```r
gasoline_forward_1 = lm(Y ~ X_1, data=gasoline_data)

# get t-value of added coefficient. Here it is significant so we continue.
summary(gasoline_model_1)$coefficients[,"t value"][2]
```

```
##       X_1
## -9.418054
```

2. Since the t-value of the added predictor was significant and exceeded `t_cutoff`, we continue and try to select the predictor which has the highest correlation with residuals of the previous model

```r
t_cutoff_df <- t_cutoff_df - 1 # add another predictor
t_cutoff <- qt(0.025, df=t_cutoff_df, lower.tail=FALSE)
t_cutoff
```

```
## [1] 2.051831
```

```r
gasoline_data_4d <- subset(gasoline_data_4d, select = -c(X_1))

names(which.max(abs(cor(residuals(gasoline_forward_1),
                        gasoline_data_4d)[1,-1])))
```

```
## [1] "X_5"
```

```r
gasoline_forward_2 = lm(Y ~ X_1 + X_5, data=gasoline_data)

# get t-value of added coefficient. Here it is insignificant so we stop and return gasoline_forward_1
summary(gasoline_forward_2)$coefficients[,"t value"][3]
```

```
##      X_5
## 0.720647
```

```r
summary(gasoline_forward_1)
```

```
##
```

```
## Call:
## lm(formula = Y ~ X_1, data = gasoline_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6000 -2.0240 -0.2681  1.4684  7.0261
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.487803   1.537109  21.786  < 2e-16 ***
## X_1         -0.047056   0.004996  -9.418 3.55e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.122 on 28 degrees of freedom
## Multiple R-squared:  0.7601, Adjusted R-squared:  0.7515
## F-statistic:  88.7 on 1 and 28 DF,  p-value: 3.555e-10
```

Since the coefficient for the new predictor is insignificant, we ignore it and return to the model with 1 predictor. Hence, after applying forward selection, our final fitted model is $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1$.