MSIA 401 - Hw6

Steven Lin

Setup

```
# My PC
main = "C:/Users/Steven/Documents/Academics/3_Graduate School/2014-2015
~ NU/"

# Aginity main = '\\\\nas1/labuser169'

course = "MSIA_401_Statistical Methods for Data Mining"
datafolder = "Data"
setwd(file.path(main, course, datafolder))

opts_knit$set(root.dir = getwd())
```

Problem 1

```
# Import data
filename = "P219.txt"
mydata = read.table(filename, header = T)
```

Part a

```
library(car)
fit = lm(H ~ P, data = mydata)
summary(fit)
```

```
##
## Call:
## lm(formula = H ~ P, data = mydata)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.008368 -0.002133  0.000525  0.002557  0.008075
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.06088    0.01042   -5.85  5.9e-06 ***
## P            0.07141    0.00423   16.87  1.9e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00408 on 23 degrees of freedom
## Multiple R-squared: 0.925,   Adjusted R-squared: 0.922
## F-statistic:  285 on 1 and 23 DF,  p-value: 1.91e-14
```

```
dw_positive = durbinWatsonTest(fit, alternative = "positive", data =
mydata)
dw_2sided = durbinWatsonTest(fit, alternative = "two.sided", data =
mydata)
dw_positive
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1          0.6511        0.6208       0
##  Alternative hypothesis: rho > 0
```

```
dw_2sided
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1          0.6511        0.6208       0
##  Alternative hypothesis: rho != 0
```

```


# alternatives library(car)
# durbinWatsonTest(fit_lagged,alternative='two.sided')
# durbinWatsonTest(fit_lagged,alternative='positive')

library(lmtest)
```

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following object(s) are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
dwtest(fit, alternative = "greater")
```

```
##
##  Durbin-Watson test
##
## data:  fit
## DW = 0.6208, p-value = 6.645e-06
## alternative hypothesis: true autocorrelation is greater than 0
```

```
dwtest(fit, alternative = "two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  fit
## DW = 0.6208, p-value = 1.329e-05
## alternative hypothesis: true autocorrelation is not 0
```

```

# Durbin-Watson statistic
dw_positive$dw
```

```
## [1] 0.6208
```

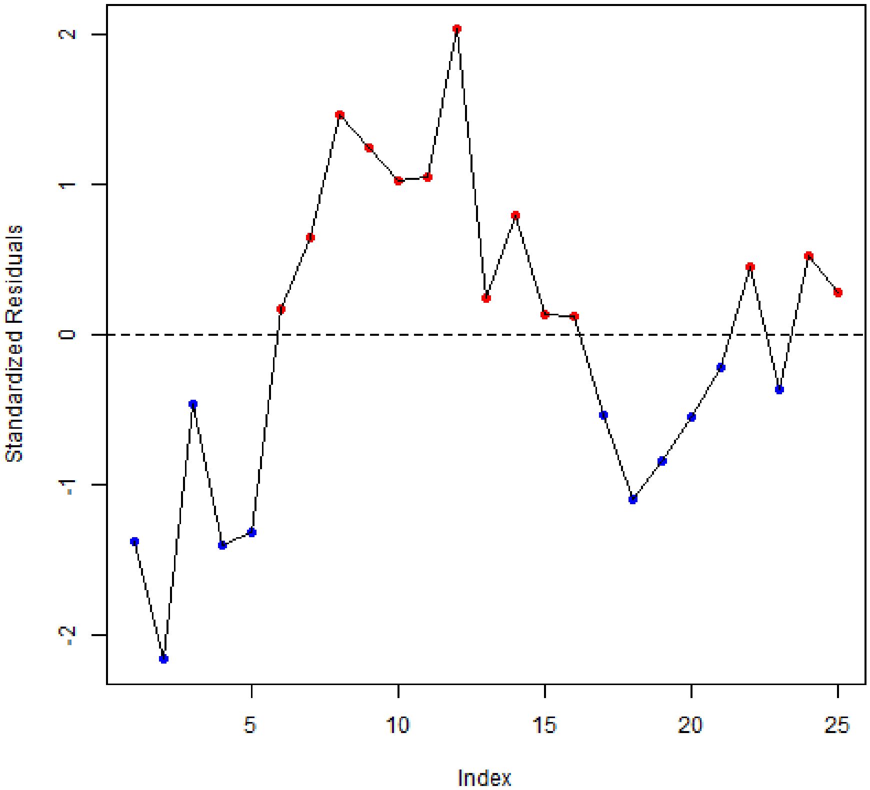
Evidence of autocorrelation is indicated by the deviation of d from 2

Durbin watson: d= 0.6208

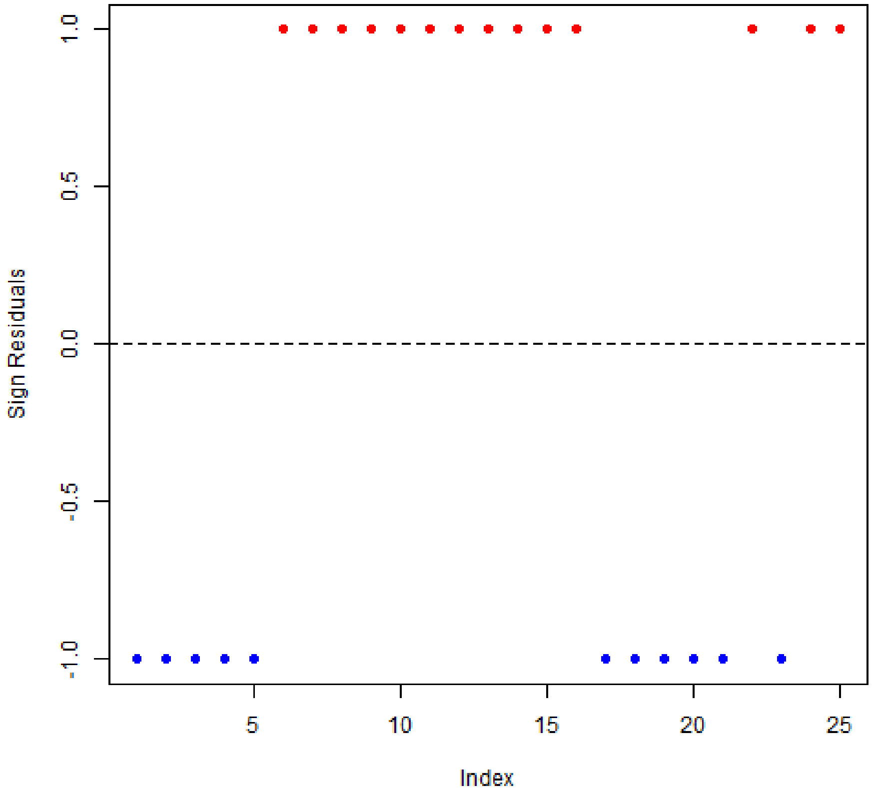
H0: correlation = 0, H1: correlation > 0 From talbe A.6, with n = 25, p = 1, and significance level 0.05, dL = 1.29, dU = 1.45. Since d = 0.62 < dL = 1.29, reject null, conclude that value of d is significant at 0.05 level, showing that positive autocorrelation is present. Similar conclusion is reached for two-sided hypothesis (H1: correlation diff 0)

Part b

```
# compute standard residuals
fit_stdres = rstandard(fit)
index_res = seq(1:length(fit_stdres))

# add standard residuals and color code +/- residuals
mydata$stdres = rstandard(fit)
mydata$res_sign[mydata$stdres > 0] = 1
mydata$res_sign[mydata$stdres < 0] = -1
mydata$color[mydata$stdres > 0] = "red"
mydata$color[mydata$stdres < 0] = "blue"

# plot results

plot(index_res, fit_stdres, ylab = "Standardized Residuals", xlab =
"Index",
    col = mydata$color, pch = 16)
abline(0, 0, lty = 2)
lines(index_res, fit_stdres)
```




```
plot(index_res, mydata$res_sign, ylab = "Sign Residuals", xlab =
"Index", col = mydata$color,
    pch = 16)
abline(0, 0, lty = 2)
```




```
# n1 = # of + res, n2 = # of - res
n1 = length(which(fit_stdres > 0))
n2 = length(which(fit_stdres < 0))

# expected value and standard deviation of number of runs
mu = 2 * n1 * n2/(n1 + n2) + 1
sigma = sqrt(2 * n1 * n2 * (2 * n1 * n2 - n1 - n2)/((n1 + n2 - 1) * (n1
+ n2)^2))

# number of runs
mydata$res_sign_lag = c(mydata$res_sign[1],
mydata$res_sign[1:length(mydata$res_sign) -
    1])

mydata$res_sign_change = mydata$res_sign != mydata$res_sign_lag
n_sign_changes = sum(mydata$res_sign_change)
n_runs = n_sign_changes + 1

# mu sigma n_runs
#
# mu-n_runs (mu-n_runs)/sigma
```

- Observed number of runs: 6.
- Expected number of runs: 13.32.
- Standard deviation: 2.4106.

The deviation of 7.32 from the expected number of runs is more than triple the standard deviation, indicating a significant departure from randomness.

```
# Using a statistical test:
z = (n_runs - mu)/sigma

# Compute critical value (two-sided)
z_crit = qnorm(0.05/2, lower.tail = FALSE)
z
```

```
## [1] -3.037
```

```
z_crit
```

```
## [1] 1.96
```

```
abs(z) > z_crit
```

```
## [1] TRUE
```

Reject null hypothesis that sequence is random and conclude that there is autocorrelation present

```
# Compute critical value (one-sided)
z_crit = qnorm(0.05, lower.tail = TRUE)
z
```

```
## [1] -3.037
```

```
z_crit
```

```
## [1] -1.645
```

```
z < z_crit
```

```
## [1] TRUE
```

Positive autocorrelation is manifested by Small values of number of runs and hence small negative values of Z. Reject null hypothesis that sequence is random and conclude that there is positive autocorrelation present

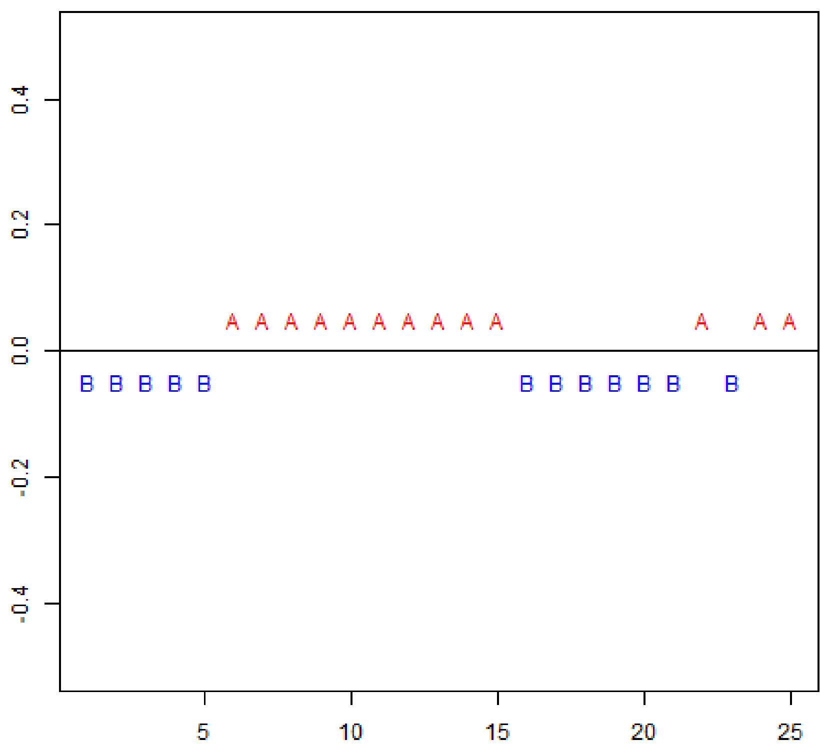
```
# use package source:
# http://www.itl.nist.gov/div898/handbook/eda/section3/eda35d.htm
library(lawstat)

# two.sided
run_test = runs.test(fit_stdres, plot.it = TRUE, alternative =
"two.sided")
run_test
```

```
##
##  Runs Test - Two sided
##
## data:  fit_stdres
## Standardized Runs Statistic = -3.062, p-value = 0.002203
```

```
# Compute critical value.
qnorm(0.05/2, lower.tail = FALSE)
```

```
## [1] 1.96
```

```
# positive.correlated
run_test = runs.test(fit_stdres, plot.it = TRUE, alternative =
"positive.correlated")
```




```
run_test
```

```
##
##  Runs Test - Positive Correlated
##
## data:  fit_stdres
## Standardized Runs Statistic = -3.062, p-value = 0.001101
```

```
# Compute critical value.
qnorm(0.05, lower.tail = TRUE)
```

```
## [1] -1.645
```

H0: the sequence was produced in a random manner Ha: the sequence was not produced in a random manner

Test statistic: Z = -3.0615 Significance level: alpha = 0.05 Critical value (upper tail): Z1-alpha/2 = 1.96 Critical region: Reject H0 if |Z| > 1.96

Since the test statistic is greater than the critical value (p-value < 0.05) we conclude that the sequence are not random at the 0.05 significance level, indicating error terms in the model are correlated and there is a pattern in the residuals present. This reconfrims earilier conclusion in (a).

Problem 2

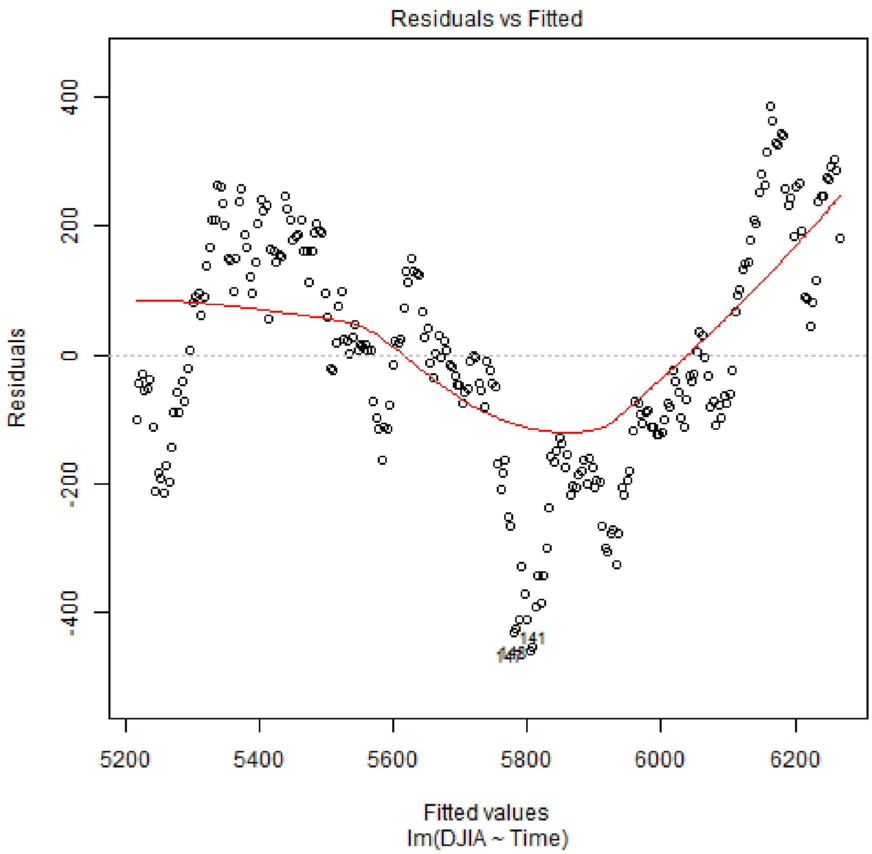
```
# Import data
filename = "P229-30.txt"
mydata = read.table(filename, header = T)
```

Part a

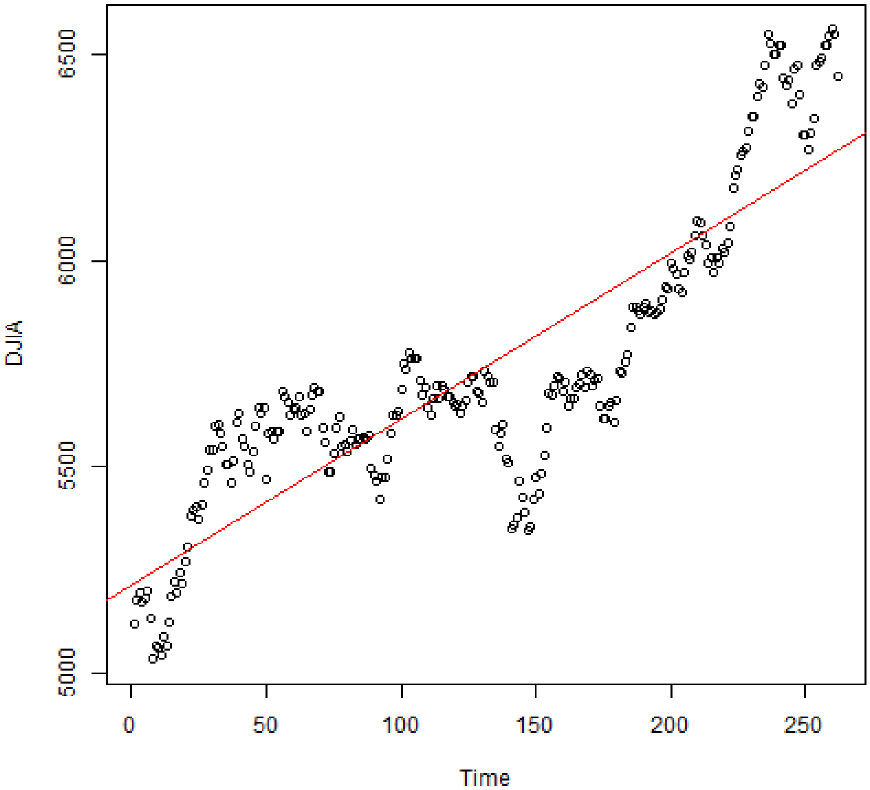
```
fit = lm(DJIA ~ Time, mydata)
summary(fit)
```

```
##
## Call:
## lm(formula = DJIA ~ Time, data = mydata)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -457.3 -111.6   -9.8  145.7  385.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5212.300     22.197   234.8   <2e-16 ***
## Time           4.024      0.146    27.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 179 on 260 degrees of freedom
## Multiple R-squared: 0.744,   Adjusted R-squared: 0.743
## F-statistic:  756 on 1 and 260 DF,  p-value: <2e-16
```

```
plot.lm(fit, which = 1)  # only get residuals vs fitted
```




```
plot(mydata$Time, mydata$DJIA, xlab = "Time", ylab = "DJIA")
abline(fit, col = "red")
```




```
library(lmtest)
dwtest(fit, alternative = "two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  fit
## DW = 0.0559, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is not 0
```

It is not clear what “linear trend model” refers to. If it refers to the linear regression model DJIA vs Time, then the plot DJIA vs Time clearly shows that the linear model is not adequate because of the cyclical behavior. Residual plot shows a trend, suggesting presence of auto- correlation in the residuals and, thus, the linear model does not seem to be adequate as the linear regression assumption of independent-errors does not hold. The presence of correlated errors have an impact on estimates, standard errors and statistical tests.

The graph of residuals show the presence of time dependence in the error term. Autocorrelation might suggest that a time-dependent variable is missing from the model.

The Durbin Watston test (two-sided) suggets that there is autocorrelation present (p-value<0.05)

Part b

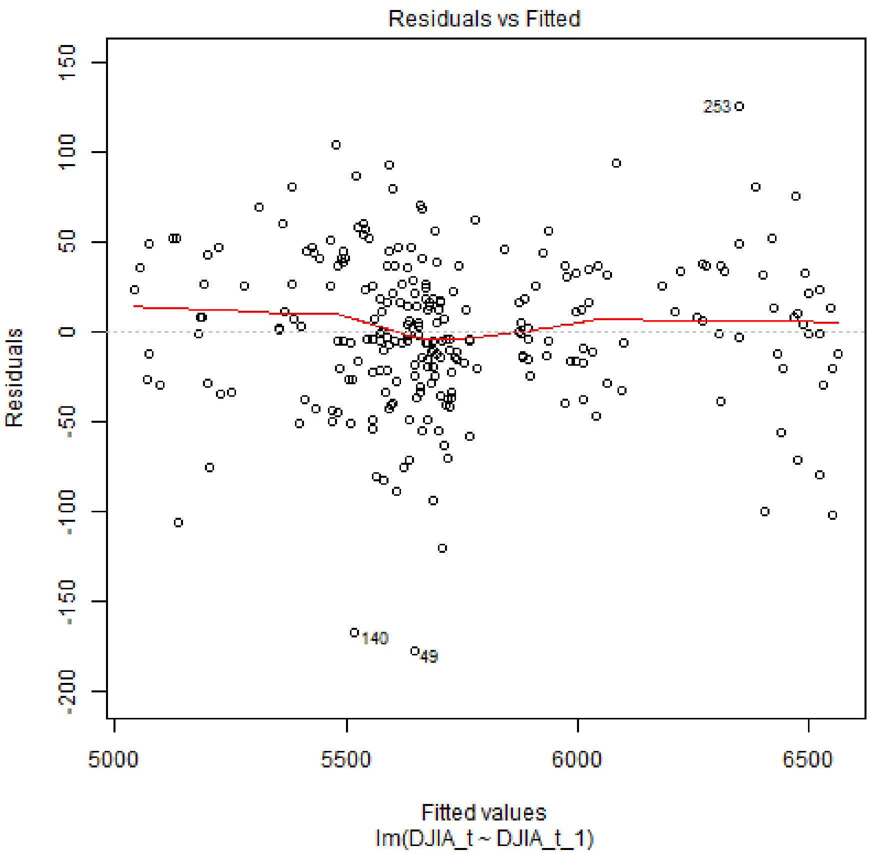
```
# Lag functions:
# http://heuristically.wordpress.com/2012/10/29/lag-function-for-data-
frames/
# http://ctszkin.com/2012/03/11/generating-a-laglead-variables/

# Create lag t-1
n = dim(mydata)[1]
mydata_lagged = data.frame(Time_t = mydata$Time[2:n], DJIA_t =
mydata$DJIA[2:n],
    Time_t_1 = mydata$Time[1:n - 1], DJIA_t_1 = mydata$DJIA[1:n - 1])
head(mydata_lagged)
```

```
##   Time_t DJIA_t Time_t_1 DJIA_t_1
## 1      2   5177        1     5117
## 2      3   5194        2     5177
## 3      4   5174        3     5194
## 4      5   5181        4     5174
## 5      6   5198        5     5181
## 6      7   5130        6     5198
```

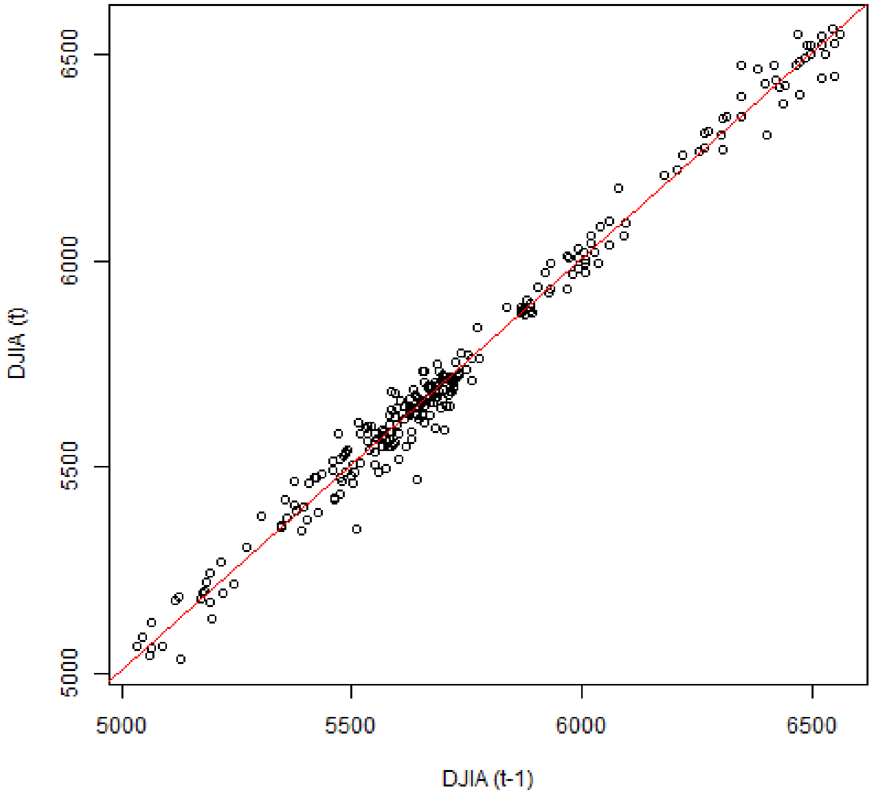
```
# Regress DJIA t vs DJIA t-1

fit_lagged = lm(DJIA_t ~ DJIA_t_1, mydata_lagged)
summary(fit_lagged)
```

```
##
## Call:
## lm(formula = DJIA_t ~ DJIA_t_1, data = mydata_lagged)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -176.88  -22.40   -0.64   26.48  125.14
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.89838   42.98910    0.86     0.39
## DJIA_t_1     0.99446    0.00748  133.00   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.4 on 259 degrees of freedom
## Multiple R-squared: 0.986,   Adjusted R-squared: 0.986
## F-statistic: 1.77e+04 on 1 and 259 DF,  p-value: <2e-16
```

```
plot.lm(fit_lagged, which = 1)  # only get residuals vs fitted
```




```
plot(mydata_lagged$DJIA_t_1, mydata_lagged$DJIA_t, xlab = "DJIA (t-1)",
ylab = "DJIA (t)")
abline(fit_lagged, col = "red")
```




```
library(lmtest)
dwtest(fit_lagged, alternative = "two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  fit_lagged
## DW = 1.759, p-value = 0.04345
## alternative hypothesis: true autocorrelation is not 0
```

The plot DJIA (t) vs DJIA (t-1) shows the linear model might be adequate

The residuals vs Fitted now appears not to show a trend, so there is no stong evidence of autocorrelation in the residuals, indicating that assumption of uncorrelated residuals (independent-errors assumption) might not be violated.

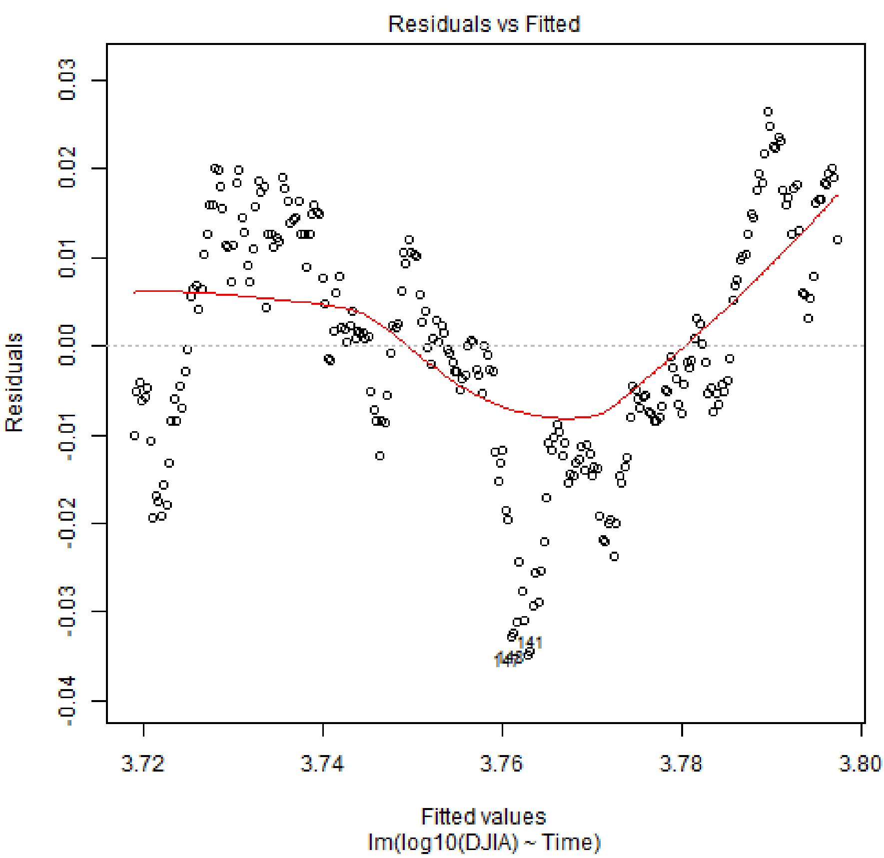
The Durbin Watston test (two-sided is more conservative) suggets that there still might be some autocorrelation (p-value borderline < 0.05), but there is not strong evidence for it. Compared to (a), it is clear that this model is more adequate for a linear regression.

Part c

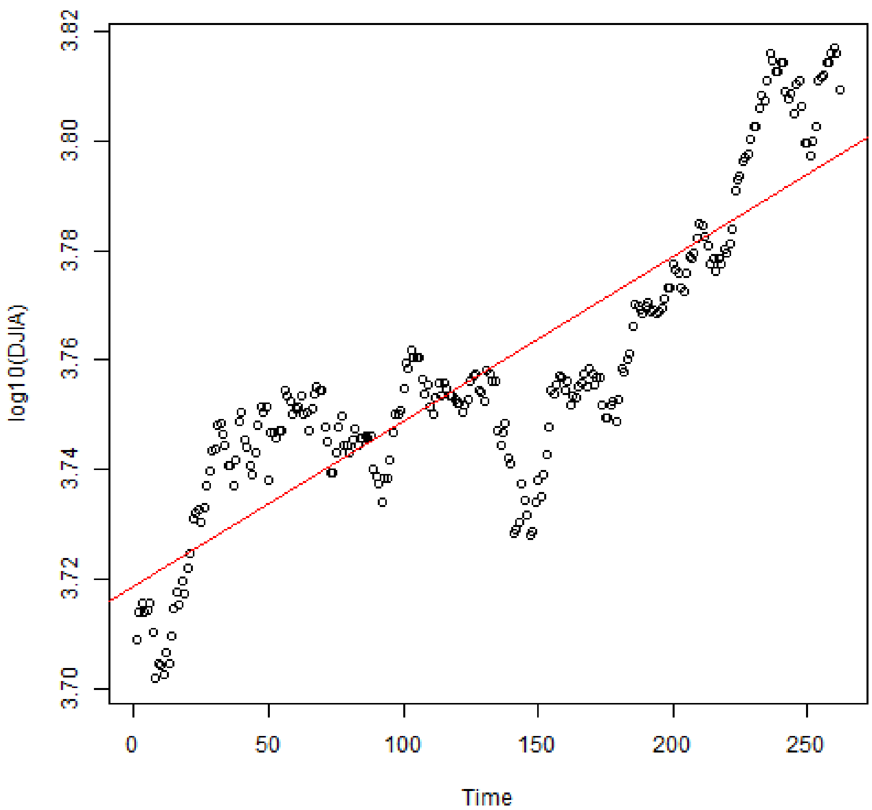
```
fit = lm(log10(DJIA) ~ Time, mydata)
summary(fit)
```

```
##
## Call:
## lm(formula = log10(DJIA) ~ Time, data = mydata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.03481 -0.00842 -0.00012  0.01111  0.02648
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.72e+00   1.64e-03  2271.4   <2e-16 ***
## Time        3.00e-04   1.08e-05    27.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0132 on 260 degrees of freedom
## Multiple R-squared: 0.749,   Adjusted R-squared: 0.748
## F-statistic:  775 on 1 and 260 DF,  p-value: <2e-16
```

```
plot.lm(fit, which = 1)  # only get residuals vs fitted
```




```
plot(mydata$Time, log10(mydata$DJIA), xlab = "Time", ylab =
"log10(DJIA)")
abline(fit, col = "red")
```




```
library(lmtest)
dwtest(fit, alternative = "two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  fit
## DW = 0.0601, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is not 0
```

Note, the problem does not specify what base to use for logarithm, so use base 10.

It is not clear what “linear trend model” refers to. If it refers to the linear regression model DJIA vs Time, then the plot DJIA vs Time clearly shows that the linear model is not adequate because of the cyclical behavior. Residual plot shows a trend, suggesting presence of auto- correlation in the residuals and, thus, the linear model does not seem to be adequate as the linear regression assumption of independent-errors does not hold. The presence of correlated errors have an impact on estimates, standard errors and statistical tests.

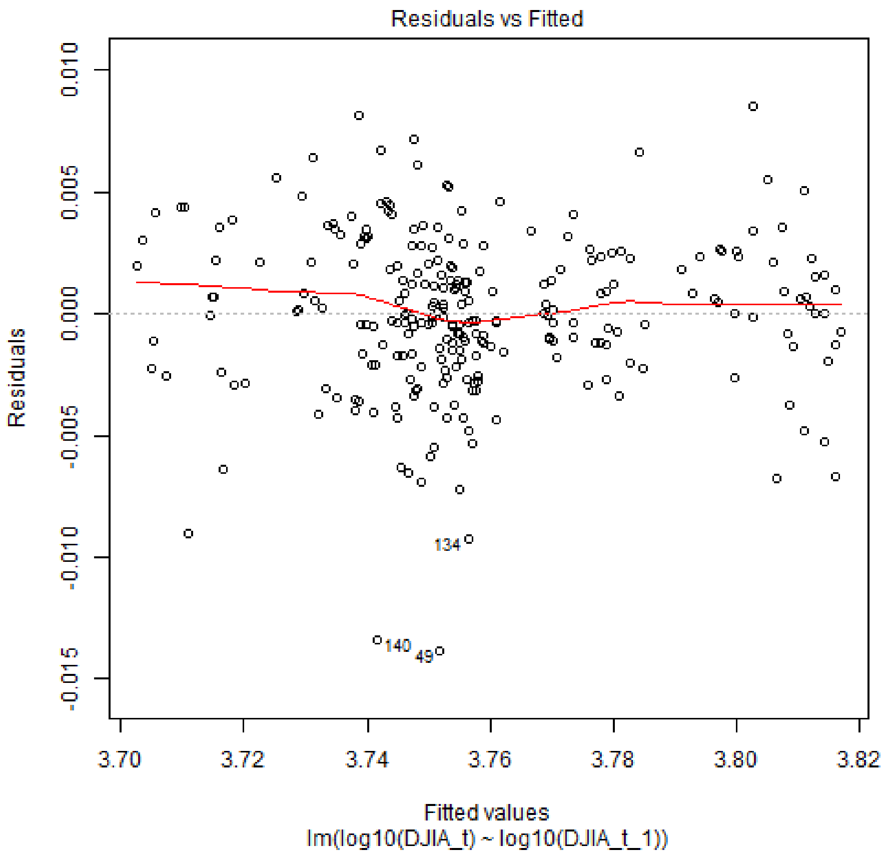
The graph of residuals show the presence of time dependence in the error term. Autocorrelation might suggest that a time-dependent variable is missing from the model.

The Durbin Watston test (two-sided) suggets that there is autocorrelation present (p-value<0.05)

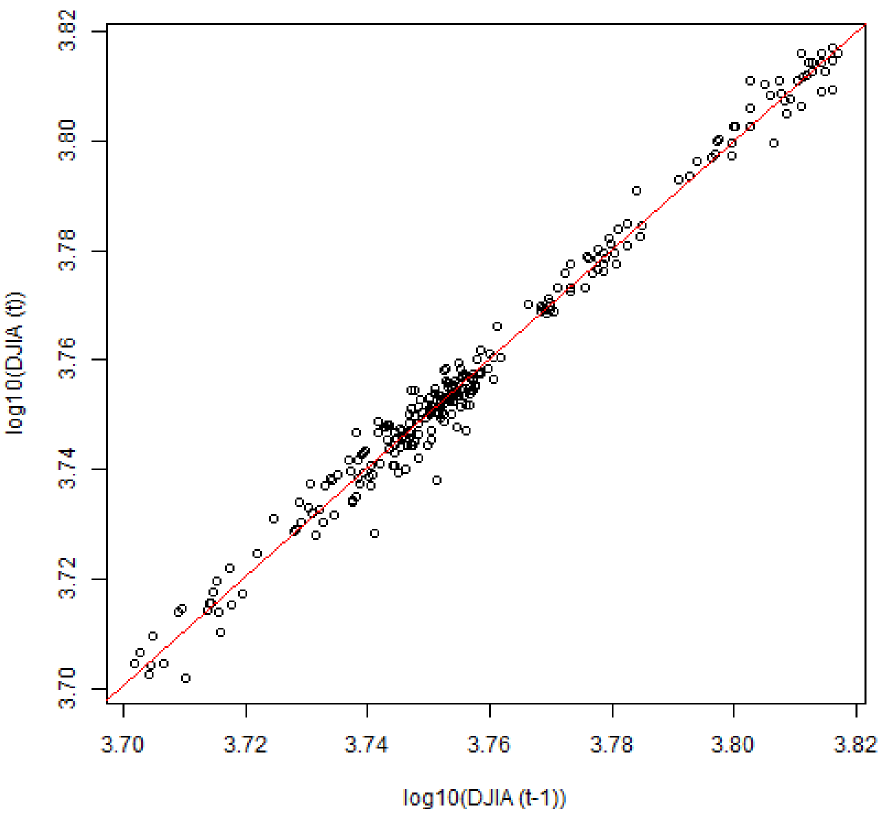
```
# Lag functions:
# http://heuristically.wordpress.com/2012/10/29/lag-function-for-data-
frames/
# http://ctszkin.com/2012/03/11/generating-a-laglead-variables/

# Regress log10(DJIA t) vs log10(DJIA t-1)

fit_lagged = lm(log10(DJIA_t) ~ log10(DJIA_t_1), mydata_lagged)
summary(fit_lagged)
```

```
##
## Call:
## lm(formula = log10(DJIA_t) ~ log10(DJIA_t_1), data = mydata_lagged)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.013818 -0.001715  0.000003  0.002126  0.008526
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.02698    0.02884    0.94     0.35
## log10(DJIA_t_1)  0.99292    0.00767  129.40   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00324 on 259 degrees of freedom
## Multiple R-squared: 0.985,   Adjusted R-squared: 0.985
## F-statistic: 1.67e+04 on 1 and 259 DF,  p-value: <2e-16
```

```
plot.lm(fit_lagged, which = 1)  # only get residuals vs fitted
```




```
plot(log10(mydata_lagged$DJIA_t_1), log10(mydata_lagged$DJIA_t), xlab =
"log10(DJIA (t-1))",
    ylab = "log10(DJIA (t))")
abline(fit_lagged, col = "red")
```




```
library(lmtest)
dwtest(fit_lagged, alternative = "two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  fit_lagged
## DW = 1.774, p-value = 0.05824
## alternative hypothesis: true autocorrelation is not 0
```

The plot log DJIA (t) vs log DJIA (t-1) clearly shows the linear model is adequate

The residuals vs Fitted now do not show a trend, so there is no evidence of autocorrelation in the residuals, indicating that assumption of uncorrelated residuals (independent-errors assumption) is not violated.

The Durbin Watston test (two-sided is more conservative) suggets that there is no strong evidence for autocorrelation (p-value borderline > 0.05), although this is borderline. Compared to the previoius model, it is clear that this model is more adequate for a linear regression.

The conclusions reached in (a) and (b) are similar. No big differences are noticed. The coefficients estimates change, but the signficicant tests and R^2 remain almost the same. The plots also show the same patterns. It seems that there is only a change in the scale and decrease in the variability/volatility.

The main difference is the result of the Durbin Watson test, which shows the log model is slightly better than non-log model in reducing autocorrelation, in which the log model now has no strong evidence at a 0.05 signficance level for autocorrelation.

The non-log model has the advantage of keeping the same units that is easy to interpret. The log model might be preferred though because of the reduction in variability, symmetrization of the distribution and no strong evidence of autocorrelation.

Problem 3

Part a

```
mydata_lagged$log_DJIA_t = log10(mydata_lagged$DJIA_t)
mydata_lagged$log_DJIA_t_1 = log10(mydata_lagged$DJIA_t_1)
fit_lagged = lm(log_DJIA_t ~ log_DJIA_t_1, mydata_lagged[1:129, ])
summary(fit_lagged)
```

```
##
## Call:
## lm(formula = log_DJIA_t ~ log_DJIA_t_1, data = mydata_lagged[1:129,
##     ])
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.013329 -0.001812  0.000058  0.002033  0.008180
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.1587     0.0724    2.19     0.03 *
## log_DJIA_t_1   0.9577     0.0194   49.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00326 on 127 degrees of freedom
## Multiple R-squared: 0.951,   Adjusted R-squared: 0.95
## F-statistic: 2.45e+03 on 1 and 127 DF,  p-value: <2e-16
```

```
# Mean squared error (calcuate using # obs or df residuals?)
MSE_log = sum((fit_lagged$res)^2)/fit_lagged$df.res
MSE_log
```

```
## [1] 1.064e-05
```

```
# or summary(fit_lagged)$sigma^2 anova(fit_lagged)['Residuals','Mean
Sq']
```

For this question, answers are given for non-log model from the previous question for DJIA t vs DJIA t-1 models. See above for discussion on the adequacy of the models. Note the conclusions to the questions below do not change whether log or non-log model is used. Non-log model was chosen because it is simpler and keeps in the same units for the predictions and errors so easier to interpret.

Note that the training set goes from days 1 to 129 for the lagged values. Thus, DJIA(t-1) is for days 1 to 129, and DJIA (t) for days 2 to 130. This ensures only data from the first half of the (130 days) is used. Also note that MSE (using the book definition) is computed using the degrees of freedom of the residuals (not just the mean of the residuals squared).

- The residual mean square (log units): 1.0641×10^{-5}

Part b

First day of July 1996 = day 131, so start with DJIA_t_1 (130)

```
# option 1: use data (not predicted of t becomes t-1 etc)

first_day = 131
last_day = 131 + 15 - 1
newdata = data.frame(log_DJIA_t_1 =
mydata_lagged$log_DJIA_t_1[(first_day -
    1):(last_day - 1)])

predicted_log = predict(fit_lagged, newdata)
predicted = 10^predicted_log
actual_log = log10(mydata$DJIA[first_day:last_day])
actual = mydata$DJIA[first_day:last_day]

pred_error_log = actual_log - predicted_log
pred_error = actual - predicted

results = data.frame(date = mydata$Date[first_day:last_day], day =
mydata$Time[first_day:last_day],
    actual = actual, predicted = predicted, pred_error = pred_error,
actual_log = actual_log,
    predicted_log = predicted_log, pred_error_log = pred_error_log)

results
```

```
##       date day actual predicted pred_error actual_log predicted_log
## 1   7/1/96 131   5730      5653     76.646      3.758         3.752
## 2   7/2/96 132   5720      5725     -5.079      3.757         3.758
## 3   7/3/96 133   5703      5716    -13.252      3.756         3.757
## 4   7/4/96 134   5703      5700      3.362      3.756         3.756
## 5   7/5/96 135   5588      5700   -111.518      3.747         3.756
## 6   7/8/96 136   5551      5590    -38.827      3.744         3.747
## 7   7/9/96 137   5582      5554     27.949      3.747         3.745
## 8  7/10/96 138   5604      5584     20.009      3.748         3.747
## 9  7/11/96 139   5520      5605    -84.014      3.742         3.749
## 10 7/12/96 140   5511      5525    -14.285      3.741         3.742
## 11 7/15/96 141   5350      5515   -165.808      3.728         3.742
## 12 7/16/96 142   5359      5361     -2.094      3.729         3.729
## 13 7/17/96 143   5377      5370      7.149      3.731         3.730
## 14 7/18/96 144   5464      5387     77.062      3.738         3.731
## 15 7/19/96 145   5427      5471    -44.034      3.735         3.738
##    pred_error_log
## 1       0.0058484
## 2      -0.0003854
## 3      -0.0010080
## 4       0.0002561
## 5      -0.0085815
## 6      -0.0030272
## 7       0.0021800
## 8       0.0015535
## 9      -0.0065595
## 10     -0.0011244
## 11     -0.0132565
## 12     -0.0001696
## 13      0.0005779
## 14      0.0061685
## 15     -0.0035097
```

```
# plot(results$day,results$actual) lines(results$day,results$actual,
# col='red') par(new=TRUE)
#
plot(results$day,results$predicted,xlab='',ylab='',ylim=range(results$actual))

# lines(results$day,results$predicted, col='blue')

require(ggplot2)


plot1 = ggplot(results, aes(day)) + geom_point(aes(y = actual), size =
3, color = "red") +
    geom_line(aes(y = actual), colour = "red") + geom_point(aes(y =
predicted),
    size = 3, color = "blue") + geom_line(aes(y = predicted), colour =
"blue") +
    scale_colour_manual("Legend", breaks = c("Actual", "Predicted"),
values = c("red",
        "blue")) + ylab("Actual (red) vs Predicted (blue)")
```

Part c

```
### Part c
AVE_Sq_error15 = mean(pred_error^2)
AVE_Sq_error15
```

```
## [1] 4260
```

```
AVE_Sq_error15_log = mean(pred_error_log^2)
AVE_Sq_error15_log
```

```
## [1] 2.641e-05
```

```
```

- Average of the squared error (log units) = 2.6411 × 10^{-5}
- Average of the squared error (original units) = 4260.0055

As expected, average squared prediction errors are much higher than MSE in (a) since part © is testing data in a new period and in a smaller sample, while MSE (a) is for the data that the model was built on and over a longer time period.

Part d

First day of July 1996 = day 131, so start with DJIA_t_1 (130)

```
# Use to predict second half (132 days) First day of July 1996 = day
131,
# so start with DJIA_t_1 (130)

# option 1: use data

first_day = 131
last_day = dim(mydata)[1]

newdata = data.frame(log_DJIA_t_1 =
mydata_lagged$log_DJIA_t_1[(first_day -
    1):(last_day - 1)])

predicted_log = predict(fit_lagged, newdata)
predicted = 10^predicted_log
actual_log = log10(mydata$DJIA[first_day:last_day])
actual = mydata$DJIA[first_day:last_day]

pred_error_log = actual_log - predicted_log
pred_error = actual - predicted

results = data.frame(date = mydata$Date[first_day:last_day], day =
mydata$Time[first_day:last_day],
    actual = actual, predicted = predicted, pred_error = pred_error,
actual_log = actual_log,
    predicted_log = predicted_log, pred_error_log = pred_error_log)

results
```

```
##        date day actual predicted pred_error actual_log predicted_log
## 1    7/1/96 131   5730      5653   76.64567      3.758         3.752
## 2    7/2/96 132   5720      5725   -5.07885      3.757         3.758
## 3    7/3/96 133   5703      5716  -13.25204      3.756         3.757
```

```
## 4     7/4/96 134   5703        5700    3.36244       3.756            3.756
## 5     7/5/96 135   5588        5700 -111.51756       3.747            3.756
## 6     7/8/96 136   5551        5590  -38.82677       3.744            3.747
## 7     7/9/96 137   5582        5554   27.94909       3.747            3.745
## 8    7/10/96 138   5604        5584   20.00925       3.748            3.747
## 9    7/11/96 139   5520        5605  -84.01357       3.742            3.749
## 10   7/12/96 140   5511        5525  -14.28494       3.741            3.742
## 11   7/15/96 141   5350        5515 -165.80773       3.728            3.742
## 12   7/16/96 142   5359        5361   -2.09351       3.729            3.729
## 13   7/17/96 143   5377        5370    7.14948       3.731            3.730
## 14   7/18/96 144   5464        5387   77.06200       3.738            3.731
## 15   7/19/96 145   5427        5471  -44.03417       3.735            3.738
## 16   7/22/96 146   5391        5435  -44.08629       3.732            3.735
## 17   7/23/96 147   5347        5401  -54.05789       3.728            3.732
## 18   7/24/96 148   5355        5358   -3.32273       3.729            3.729
## 19   7/25/96 149   5422        5366   56.18528       3.734            3.730
## 20   7/26/96 150   5473        5430   42.64721       3.738            3.735
## 21   7/29/96 151   5435        5479  -44.77848       3.735            3.739
## 22   7/30/96 152   5482        5442   39.45150       3.739            3.736
## 23   7/31/96 153   5529        5488   41.03739       3.743            3.739
## 24    8/1/96 154   5595        5533   61.84488       3.748            3.743
## 25    8/2/96 155   5680        5596   83.84140       3.754            3.748
## 26    8/5/96 156   5674        5677   -3.18010       3.754            3.754
## 27    8/6/96 157   5696        5672   23.96292       3.756            3.754
## 28    8/7/96 158   5719        5693   25.62629       3.757            3.755
## 29    8/8/96 159   5713        5715   -1.14557       3.757            3.757
## 30    8/9/96 160   5681        5710  -28.36818       3.754            3.757
## 31   8/12/96 161   5705        5679   26.10313       3.756            3.754
## 32   8/13/96 162   5647        5702  -54.25349       3.752            3.756
## 33   8/14/96 163   5667        5646   20.58321       3.753            3.752
## 34   8/15/96 164   5666        5665    0.71730       3.753            3.753
## 35   8/16/96 165   5689        5664   25.44041       3.755            3.753
## 36   8/19/96 166   5699        5687   12.77117       3.756            3.755
## 37   8/20/96 167   5721        5696   25.02897       3.757            3.756
## 38   8/21/96 168   5690        5717  -27.29419       3.755            3.757
## 39   8/22/96 169   5733        5687   46.44701       3.758            3.755
## 40   8/23/96 170   5723        5729   -6.05848       3.758            3.758
## 41   8/26/96 171   5694        5719  -24.64052       3.755            3.757
## 42   8/27/96 172   5711        5691   20.35122       3.757            3.755
## 43   8/28/96 173   5712        5708    4.82647       3.757            3.756
## 44   8/29/96 174   5648        5709  -60.96586       3.752            3.757
## 45   8/30/96 175   5616        5647  -30.44107       3.749            3.752
## 46    9/2/96 176   5616        5617   -0.33333       3.749            3.749
## 47    9/3/96 177   5648        5617   31.84667       3.752            3.749
## 48    9/4/96 178   5657        5647    9.54037       3.753            3.752
## 49    9/5/96 179   5607        5656  -48.54775       3.749            3.752
## 50    9/6/96 180   5660        5608   52.17606       3.753            3.749
## 51    9/9/96 181   5734        5658   75.49824       3.758            3.753
## 52   9/10/96 182   5727        5729   -1.97253       3.758            3.758
## 53   9/11/96 183   5755        5723   32.14057       3.760            3.758
## 54   9/12/96 184   5772        5749   22.61761       3.761            3.760
## 55   9/13/96 185   5839        5766   72.91474       3.766            3.761
## 56   9/16/96 186   5889        5829   59.91777       3.770            3.766
## 57   9/17/96 187   5889        5878   11.09814       3.770            3.769
## 58   9/18/96 188   5877        5877   -0.01821       3.769            3.769
## 59   9/19/96 189   5868        5866    1.32549       3.768            3.768
## 60   9/20/96 190   5888        5857   31.24154       3.770            3.768
## 61   9/23/96 191   5895        5877   17.71544       3.770            3.769
## 62   9/24/96 192   5874        5883   -8.99698       3.769            3.770
## 63   9/25/96 193   5877        5863   14.12866       3.769            3.768
## 64   9/26/96 194   5869        5866    2.43549       3.769            3.768
## 65   9/27/96 195   5873        5858   14.64043       3.769            3.768
## 66   9/30/96 196   5882        5862   19.99974       3.770            3.768
## 67   10/1/96 197   5905        5871   33.88770       3.771            3.769
```

```
## 68   10/2/96 198   5934       5893   41.23266       3.773          3.770
## 69   10/3/96 199   5933       5921   12.33309       3.773          3.772
## 70   10/4/96 200   5993       5919   73.41327       3.778          3.772
## 71   10/7/96 201   5980       5977    3.03486       3.777          3.776
## 72   10/8/96 202   5967       5964    2.45963       3.776          3.776
## 73   10/9/96 203   5931       5952  -21.23399       3.773          3.775
## 74  10/10/96 204   5922       5917    4.35409       3.772          3.772
## 75  10/11/96 205   5969       5909   60.61638       3.776          3.771
## 76  10/14/96 206   6010       5954   55.65273       3.779          3.775
## 77  10/15/96 207   6005       5993   11.63525       3.778          3.778
## 78  10/16/96 208   6021       5988   32.65041       3.780          3.777
## 79  10/17/96 209   6059       6003   55.73213       3.782          3.778
## 80  10/18/96 210   6094       6040   54.10757       3.785          3.781
## 81  10/21/96 211   6091       6074   17.30970       3.785          3.783
## 82  10/22/96 212   6062       6070   -8.55337       3.783          3.783
## 83  10/23/96 213   6036       6043   -6.14454       3.781          3.781
## 84  10/24/96 214   5992       6018  -25.93159       3.778          3.779
## 85  10/25/96 215   6007       5976   30.60780       3.779          3.776
## 86  10/28/96 216   5973       5990  -17.56884       3.776          3.777
## 87  10/29/96 217   6007       5958   49.47262       3.779          3.775
## 88  10/30/96 218   5993       5990    2.93116       3.778          3.777
## 89  10/31/96 219   6029       5977   52.25147       3.780          3.776
## 90   11/1/96 220   6022       6012   10.27869       3.780          3.779
## 91   11/4/96 221   6042       6005   37.14262       3.781          3.778
## 92   11/5/96 222   6081       6023   57.78435       3.784          3.780
## 93   11/6/96 223   6178       6061  116.60562       3.791          3.783
## 94   11/7/96 224   6206       6153   52.82658       3.793          3.789
## 95   11/8/96 225   6220       6180   39.58566       3.794          3.791
## 96  11/11/96 226   6256       6193   62.22430       3.796          3.792
## 97  11/12/96 227   6266       6227   38.54829       3.797          3.794
## 98  11/13/96 228   6274       6237   36.79537       3.798          3.795
## 99  11/14/96 229   6313       6245   67.73843       3.800          3.796
## 100 11/15/96 230   6348       6282   65.82496       3.803          3.798
## 101 11/18/96 231   6347       6316   31.32493       3.803          3.800
## 102 11/19/96 232   6398       6315   83.08206       3.806          3.800
## 103 11/20/96 233   6430       6363   67.21307       3.808          3.804
## 104 11/21/96 234   6418       6394   24.78719       3.807          3.806
## 105 11/22/96 235   6472       6383   89.07633       3.811          3.805
## 106 11/25/96 236   6548       6433  114.36491       3.816          3.808
## 107 11/26/96 237   6528       6506   22.62160       3.815          3.813
## 108 11/27/96 238   6499       6487   11.99357       3.813          3.812
## 109 11/28/96 239   6499       6460   39.66087       3.813          3.810
## 110 11/29/96 240   6522       6460   62.02087       3.814          3.810
## 111  12/2/96 241   6522       6481   40.73933       3.814          3.812
## 112  12/3/96 242   6443       6481  -38.27067       3.809          3.812
## 113  12/4/96 243   6423       6406   17.19244       3.808          3.807
## 114  12/5/96 244   6437       6387   50.15942       3.809          3.805
## 115  12/6/96 245   6382       6400  -18.48472       3.805          3.806
## 116  12/9/96 246   6464       6348  116.04958       3.810          3.803
## 117 12/10/96 247   6473       6426   47.26981       3.811          3.808
## 118 12/11/96 248   6403       6435  -32.32357       3.806          3.809
## 119 12/12/96 249   6304       6367  -63.78302       3.800          3.804
## 120 12/13/96 250   6305       6273   31.51870       3.800          3.797
## 121 12/16/96 251   6268       6274   -6.10686       3.797          3.798
## 122 12/17/96 252   6308       6240   68.68324       3.800          3.795
## 123 12/18/96 253   6347       6278   69.01558       3.803          3.798
## 124 12/19/96 254   6474       6314  159.25545       3.811          3.800
## 125 12/20/96 255   6484       6435   49.18515       3.812          3.809
## 126 12/23/96 256   6489       6445   43.56202       3.812          3.809
## 127 12/24/96 257   6523       6450   72.99417       3.814          3.810
## 128 12/25/96 258   6523       6482   40.79488       3.814          3.812
## 129 12/26/96 259   6547       6482   64.62488       3.816          3.812
## 130 12/27/96 260   6561       6505   56.17781       3.817          3.813
## 131 12/30/96 261   6549       6518   31.09795       3.816          3.814
```

```
## 132 12/31/96 262   6448      6507  -59.02182      3.809           3.813
##     pred_error_log
## 1        5.848e-03
## 2       -3.854e-04
## 3       -1.008e-03
## 4        2.561e-04
## 5       -8.581e-03
## 6       -3.027e-03
## 7        2.180e-03
## 8        1.554e-03
## 9       -6.560e-03
## 10      -1.124e-03
## 11      -1.326e-02
## 12      -1.696e-04
## 13       5.779e-04
## 14       6.169e-03
## 15      -3.510e-03
## 16      -3.537e-03
## 17      -4.369e-03
## 18      -2.694e-04
## 19       4.524e-03
## 20       3.397e-03
## 21      -3.564e-03
## 22       3.137e-03
## 23       3.235e-03
## 24       4.827e-03
## 25       6.459e-03
## 26      -2.433e-04
## 27       1.831e-03
## 28       1.951e-03
## 29      -8.707e-05
## 30      -2.163e-03
## 31       1.992e-03
## 32      -4.152e-03
## 33       1.580e-03
## 34       5.499e-05
## 35       1.946e-03
## 36       9.742e-04
## 37       1.904e-03
## 38      -2.078e-03
## 39       3.533e-03
## 40      -4.595e-04
## 41      -1.875e-03
## 42       1.550e-03
## 43       3.671e-04
## 44      -4.663e-03
## 45      -2.348e-03
## 46      -2.578e-05
## 47       2.456e-03
## 48       7.331e-04
## 49      -3.744e-03
## 50       4.022e-03
## 51       5.756e-03
## 52      -1.496e-04
## 53       2.432e-03
## 54       1.705e-03
## 55       5.458e-03
## 56       4.441e-03
## 57       8.192e-04
## 58      -1.346e-06
## 59       9.812e-05
## 60       2.310e-03
## 61       1.307e-03
## 62      -6.647e-04
```

```
## 63       1.045e-03
## 64       1.803e-04
## 65       1.084e-03
## 66       1.479e-03
## 67       2.500e-03
## 68       3.028e-03
## 69       9.037e-04
## 70       5.353e-03
## 71       2.205e-04
## 72       1.791e-04
## 73      -1.552e-03
## 74       3.194e-04
## 75       4.433e-03
## 76       4.040e-03
## 77       8.423e-04
## 78       2.362e-03
## 79       4.013e-03
## 80       3.873e-03
## 81       1.236e-03
## 82      -6.124e-04
## 83      -4.418e-04
## 84      -1.875e-03
## 85       2.219e-03
## 86      -1.276e-03
## 87       3.592e-03
## 88       2.125e-04
## 89       3.780e-03
## 90       7.419e-04
## 91       2.678e-03
## 92       4.146e-03
## 93       8.276e-03
## 94       3.713e-03
## 95       2.773e-03
## 96       4.342e-03
## 97       2.680e-03
## 98       2.554e-03
## 99       4.685e-03
## 100      4.527e-03
## 101      2.149e-03
## 102      5.677e-03
## 103      4.564e-03
## 104      1.680e-03
## 105      6.019e-03
## 106      7.652e-03
## 107      1.507e-03
## 108      8.022e-04
## 109      2.658e-03
## 110      4.150e-03
## 111      2.721e-03
## 112     -2.572e-03
## 113      1.164e-03
## 114      3.397e-03
## 115     -1.256e-03
## 116      7.868e-03
## 117      3.183e-03
## 118     -2.187e-03
## 119     -4.372e-03
## 120      2.177e-03
## 121     -4.229e-04
## 122      4.754e-03
## 123      4.748e-03
## 124      1.082e-02
## 125      3.307e-03
## 126      2.925e-03
```

```
## 127      4.887e-03
## 128      2.725e-03
## 129      4.308e-03
## 130      3.735e-03
## 131      2.067e-03
## 132     -3.957e-03
```

```
# plot(results$day,results$actual) lines(results$day,results$actual,
# col='red') par(new=TRUE)
#
plot(results$day,results$predicted,xlab='',ylab='',ylim=range(results$actual))

# lines(results$day,results$predicted, col='blue')

require(ggplot2)


plot2 = ggplot(results, aes(day)) + geom_point(aes(y = actual), size =
3, color = "red") +
    geom_line(aes(y = actual), colour = "red") + geom_point(aes(y =
predicted),
    size = 3, color = "blue") + geom_line(aes(y = predicted), colour =
"blue") +
    scale_colour_manual("Legend", breaks = c("Actual", "Predicted"),
values = c("red",
        "blue")) + ylab("Actual (red) vs Predicted (blue)")

AVE_Sq_error132 = mean(pred_error^2)
AVE_Sq_error132
```

```
## [1] 2467
```

```
AVE_Sq_error132_log = mean(pred_error_log^2)
AVE_Sq_error132_log
```

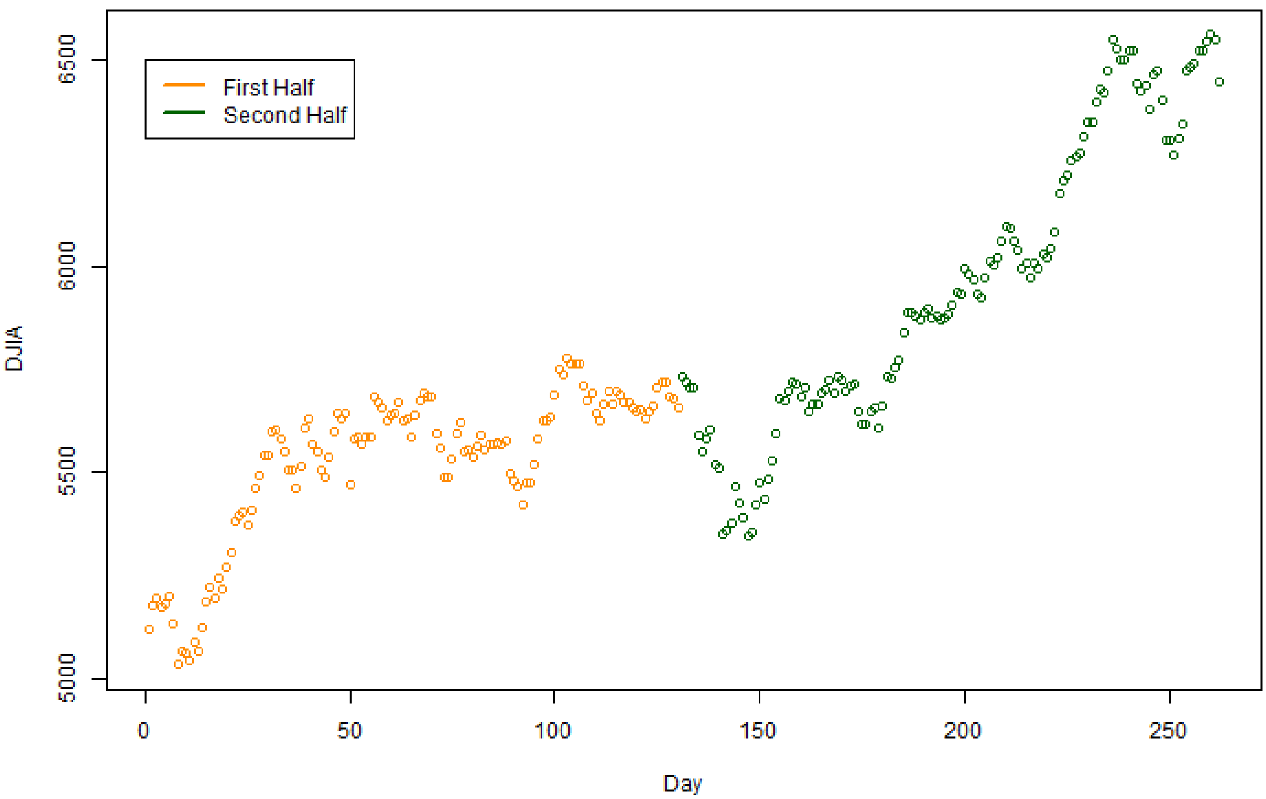
```
## [1] 1.303e-05
```

- Average of the squared error (log units) = 1.3034×10^{-5}
- Average of the squared error (original units) = 2466.9671

Average squared prediction errors are higher than MSE (a), but lower than average squared prediction error for first 15 days of second half of year (part c)

Part e

```
mydata$color[mydata$Time<131]="darkorange"
mydata$color[mydata$Time>=131]="darkgreen"

plot(mydata$Time,mydata$DJIA, xlab="Day", ylab="DJIA", col=mydata$color)
legend(0,6500, c("First Half","Second Half"), # puts text in the legend
in the appropriate place
      lty=c(1,1), # gives the legend appropriate symbols (lines)
      lwd=c(2.5,2.5), # gives the legend lines the correct color and
width
      col=c("darkorange","darkgreen"))
```




```
require(ggplot2)
require(grid)
require(gridExtra)

grid.arrange(plot1, plot2, ncol=2, main = "Second half first 15 days vs
Second Half")
```

From the scater plot we clearly see a difference between the first half of the year and the second half. Because the model used the training data for the first half of the year, then the prediction error is larger in (c) and (d) than (a) because (c) and (d) are based on the second part of the year, which is different data than the training set.The expected error the model exhibits on new data will always be higher than that it exhibits on the training data (http://scott.fortmann-roe.com/docs/MeasuringError.html)

Now usually one would expect that the error in a closer time to the training to be smaller than the error in a further time out. However, this is not the case here because the prediction for period t is a function of the previous period t-1. One can explain the results of the average squared predicion error in 15 days © being larger than average squared predicion error for the entire second half of the year (d) as follows:

- The behavior for the entire second half of the year is similar to the behavior of the entire first half of the year. Because the model was based on the entire first half of the year, then the predictions errors will be smaller for the entire second half of the year rather than a small portion (e.g. 15 days). In addition we see that for the first 15 days the DJIA decreases while it increases in a stable manner afterwards, similary to the first half of the year.

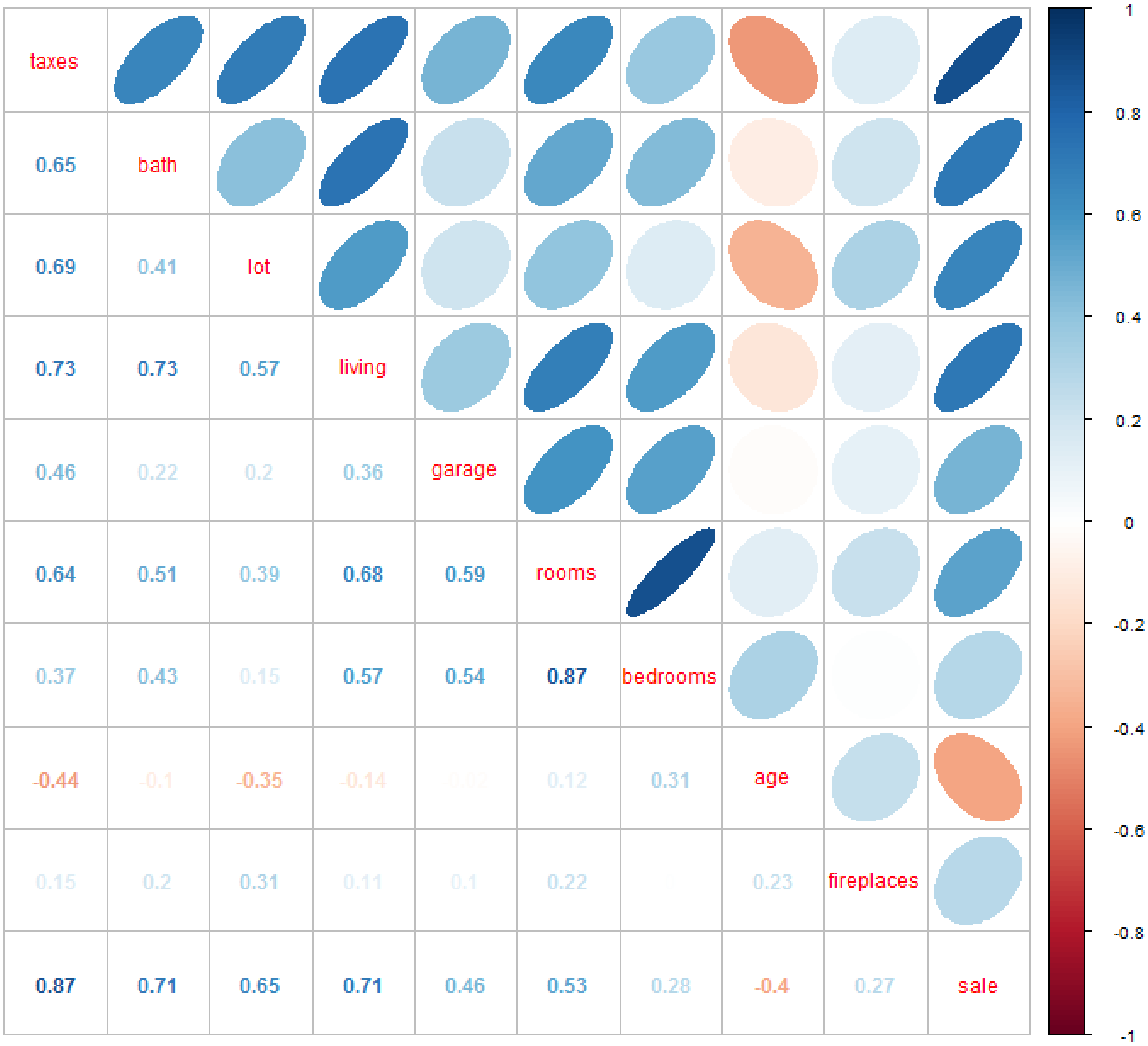
- The error of a small sample is also larger than a bigger sample. When we look at the prediction vs actual in the first 15 days, we see day-today big changes in the actual values for some days. Thus, since the prediction of the next day is based on the previous period, then the error will be substantial. It is as if the prediction is “catching up” the actual value. In the second half of the year, we see that this happens too, but there are more days in which the day-to-day changes are small, which translates on an smaller prediction average error for the entire second half of the year vs the first 15 days.

Problem 4

```
# Import data
filename = "P329.txt"
mydata = read.table(filename, header = T)

var_names = c("taxes", "bath", "lot", "living", "garage", "rooms",
"bedrooms",
    "age", "fireplaces", "sale")

predictor_names = c("intercept", "taxes", "bath", "lot", "living",
"garage",
    "rooms", "bedrooms", "age", "fireplaces")

colnames(mydata) = var_names
```

Part a

```
# Correlation matrix

corr = round(cor(mydata), 2)

library(corrplot)
corrplot.mixed(corr, upper = "ellipse", lower = "number")
```




```
# pairs(mydata[,-1], main = 'Correlation coeffficients matrix and
scatter
# plot', pch = 21, lower.panel = NULL, panel = panel.smooth,
cex.labels=2)
```

The pairwise correlation coefficeints of the predictor vairables and the corresponding scatter plots show strong linear relationships among some pairs of predictors variables, suggesting collinearity. (look at high magnitudes for correlation coefficient in conjuction for a trend in the scatter plot)

In particular, rooms (X6) and bedrooms (X7) are strongly correlated, which makes sense since a bedroom is a room too. Thus, both variables cannot be in the model since might cause the non-collinearity assumption to be violated.

```
fit = lm(sale ~ ., mydata)
summary(fit)
```

```
##
## Call:
## lm(formula = sale ~ ., data = mydata)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.773 -1.980 -0.087  1.662  4.262
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  15.3104     5.9609    2.57    0.022 *
## taxes         1.9541     1.0383    1.88    0.081 .
## bath          6.8455     4.3353    1.58    0.137
## lot           0.1376     0.4944    0.28    0.785
## living        2.7814     4.3948    0.63    0.537
## garage        2.0508     1.3846    1.48    0.161
## rooms        -0.5559     2.3979   -0.23    0.820
## bedrooms     -1.2452     3.4229   -0.36    0.721
## age          -0.0380     0.0673   -0.57    0.581
## fireplaces    1.7045     1.9532    0.87    0.398
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.97 on 14 degrees of freedom
## Multiple R-squared: 0.851,   Adjusted R-squared: 0.756
## F-statistic:  8.9 on 9 and 14 DF,  p-value: 0.000202
```

```
# Compute VIF
library(car)
vif(fit)
```

```
##      taxes       bath        lot     living     garage      rooms
##      7.022      2.835      2.455      3.836      1.823     11.711
##   bedrooms        age fireplaces
##      9.722      2.321      1.942
```

```
# Determine VIF > 10
names(vif(fit))[vif(fit) > 10]
```

```
## [1] "rooms"
```

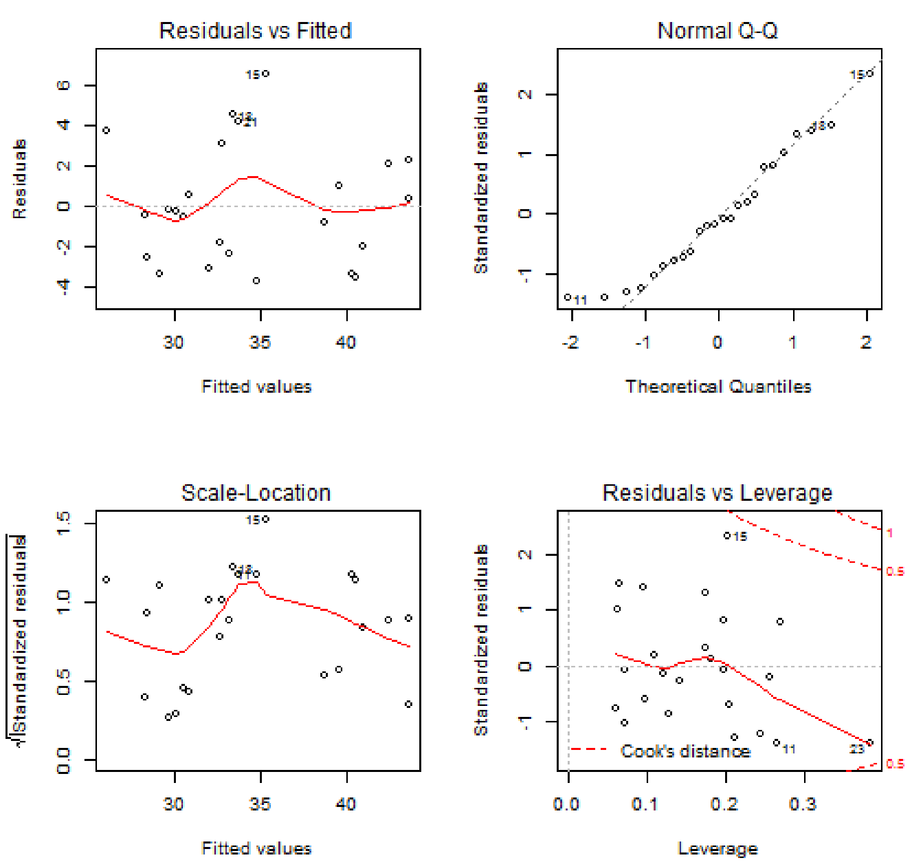
Fitting a linear model with all predictors and computing VIF confirms our suspicion. It appears that rooms (X6) is affected by the presence of collinearity because VIF > 10. Thus,there is a multicollinearity problem.Do not include all of them because of multicollinearity. In addition, if all variables in the model are included, none of the variables are significant (p-value > 0.05)

Part b

```
fit = lm(sale ~ taxes + rooms + age, mydata)
summary(fit)
```

```
##
## Call:
## lm(formula = sale ~ taxes + rooms + age, data = mydata)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.749 -2.408 -0.359  2.138  6.535
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.79601    4.97110    2.98  0.00746 **
## taxes        3.48946    0.72937    4.78  0.00011 ***
## rooms       -0.41551    1.18226   -0.35  0.72892
## age          0.00492    0.06360    0.08  0.93906
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.12 on 20 degrees of freedom
## Multiple R-squared: 0.765,   Adjusted R-squared: 0.73
## F-statistic: 21.8 on 3 and 20 DF,  p-value: 1.65e-06
```

```
par(mfrow = c(2, 2))
plot(fit)
```



```
par(mfrow = c(1, 1))

# Compute VIF
library(car)
vif(fit)
```

```
## taxes rooms   age
## 3.140 2.580 1.881
```

```

# Determine VIF > 10
names(vif(fit))[vif(fit) > 10]
```

```
## character(0)
```

The residuals diagnostics and VIF show no problems. The R^2 of 0.77 is OK but not great, and the two predictors in the model (rooms and age) are far from being signficant (p-value > 0.05). Thus, this model would NOT adequately describe the sale price.

Part c

```
# Stepwise regression to determine best model, start with all variables
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object(s) are masked from 'package:VGAM':
##
##     huber
```

```
fit = lm(sale ~ ., data = mydata)
fit_stepAIC = step(object = fit, direction = "both")  # AIC
```

```
## Start:  AIC=59.36
## sale ~ taxes + bath + lot + living + garage + rooms + bedrooms +
##     age + fireplaces
##
##              Df Sum of Sq RSS  AIC
## - rooms       1      0.47 124 57.5
## - lot         1      0.68 124 57.5
## - bedrooms    1      1.17 125 57.6
## - age         1      2.82 127 57.9
## - living      1      3.54 127 58.0
## - fireplaces  1      6.73 130 58.6
## <none>                    124 59.4
## - garage      1     19.39 143 60.9
## - bath        1     22.04 146 61.3
## - taxes       1     31.30 155 62.8
##
## Step:  AIC=57.45
## sale ~ taxes + bath + lot + living + garage + bedrooms + age +
##     fireplaces
```

```
##
##              Df Sum of Sq RSS  AIC
## - lot         1       0.8 125 55.6
## - age         1       2.9 127 56.0
## - living      1       3.4 128 56.1
## - fireplaces  1       6.7 131 56.7
## - bedrooms    1       9.4 134 57.2
## <none>                    124 57.5
## - garage      1      19.7 144 59.0
## + rooms       1       0.5 124 59.4
## - bath        1      26.2 150 60.0
## - taxes       1      40.2 164 62.2
##
## Step:  AIC=55.61
## sale ~ taxes + bath + living + garage + bedrooms + age + fireplaces
##
##              Df Sum of Sq RSS  AIC
## - age         1       3.4 128 54.2
## - living      1       4.8 130 54.5
## - fireplaces  1       9.6 135 55.4
## - bedrooms    1       9.7 135 55.4
## <none>                    125 55.6
## - garage      1      19.0 144 57.0
## + lot         1       0.8 124 57.5
## + rooms       1       0.6 124 57.5
## - bath        1      25.5 150 58.1
## - taxes       1      53.2 178 62.1
##
## Step:  AIC=54.24
## sale ~ taxes + bath + living + garage + bedrooms + fireplaces
##
##              Df Sum of Sq RSS  AIC
## - living      1       5.0 133 53.2
## - fireplaces  1       6.5 135 53.4
## <none>                    128 54.2
## + age         1       3.4 125 55.6
## - garage      1      20.0 148 55.7
## + lot         1       1.3 127 56.0
## + rooms       1       0.7 128 56.1
## - bedrooms    1      22.8 151 56.2
## - bath        1      24.3 153 56.4
## - taxes       1      95.1 224 65.6
##
## Step:  AIC=53.16
## sale ~ taxes + bath + garage + bedrooms + fireplaces
##
##              Df Sum of Sq RSS  AIC
## - fireplaces  1       6.2 140 52.3
## <none>                    133 53.2
## - garage      1      17.8 151 54.2
## - bedrooms    1      17.9 151 54.2
## + living      1       5.0 128 54.2
## + age         1       3.6 130 54.5
## + lot         1       3.0 130 54.6
## + rooms       1       0.6 133 55.1
## - bath        1      39.3 173 57.4
## - taxes       1     158.0 291 69.9
##
## Step:  AIC=52.26
## sale ~ taxes + bath + garage + bedrooms
##
##              Df Sum of Sq RSS  AIC
## <none>                    140 52.3
## + fireplaces  1       6.2 133 53.2
```

```
## + lot          1       5.8 134 53.2
## + living       1       4.7 135 53.4
## - garage       1      20.8 160 53.6
## - bedrooms     1      21.7 161 53.7
## + rooms        1       0.4 139 54.2
## + age          1       0.4 139 54.2
## - bath         1      47.4 187 57.3
## - taxes        1     156.6 296 68.3
```

```
summary(fit_stepAIC)
```

```
## 
## Call:
## lm(formula = sale ~ taxes + bath + garage + bedrooms, data = mydata)
## 
## Residuals:
##    Min     1Q Median     3Q    Max 
## -4.560 -2.086  0.024  1.858  3.898 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)   13.621      3.673    3.71  0.00149 ** 
## taxes          2.412      0.523    4.62  0.00019 ***
## bath           8.459      3.330    2.54  0.01997 *  
## garage         2.060      1.223    1.68  0.10854    
## bedrooms      -2.215      1.290   -1.72  0.10218    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 2.71 on 19 degrees of freedom
## Multiple R-squared: 0.832,   Adjusted R-squared: 0.797 
## F-statistic: 23.5 on 4 and 19 DF,  p-value: 3.87e-07
```

```

x = mydata[, 1:9]  # design matrix or use model.matrix(fullfit)
y = mydata[, 10]  # response vector

# function returns best subset function with different criteria
modelSelection = function(x, y) {
    # Inputs: x = design matrix y = response vector
    n = length(y)  # number of observations
    p = dim(x)[2]  # number of predictors

    # Variable Selection Using Package
    library(leaps)

    # find the best subset
    reg_exh = regsubsets(x, y, nbest = 1, nvmax = n, method = 
"exhaustive")
    # summary(reg_exh,matrix.logical=TRUE) names(reg_exh)
    # names(summary(reg_exh))

    # get matrix with models
    models = summary(reg_exh)$which  # T/F -> multiply by 1 to get 1/0
(not needed)
    msize = as.numeric(apply(models, 1, sum))  # model size

    # compute criteria
    cp = summary(reg_exh)$cp
    cp = round(cp, 3)
    adjr2 = summary(reg_exh)$adjr2
```

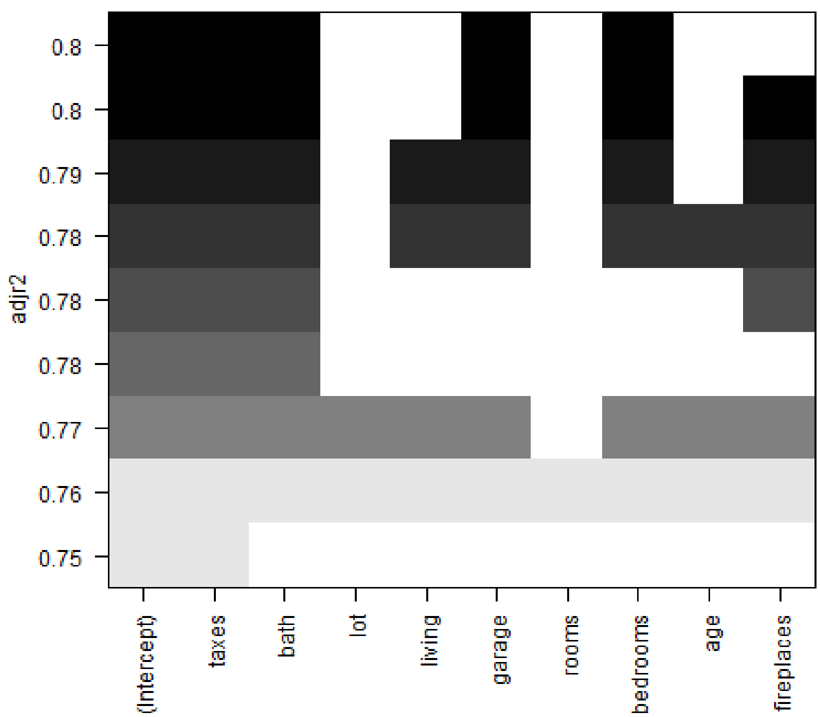
```
    adjr2 = round(adjr2, 3)
    aic = n * log(summary(reg_exh)$rss/n) + 2 * msize
    aic = round(aic, 3)
    bic = n * log(summary(reg_exh)$rss/n) + msize * log(n)
    bic = round(bic, 3)
    # different from regsubsets, just differ by constant bic =
    # summary(reg_exh)$bic; bic = round(bic,3)

    # alternative optimizing various criteria
leaps(x,y,nbest=1,method='Cp')
    # leaps(x,y,nbest=1,method='adjr2')

    # rank by criteria
    rk_cp = as.numeric(factor(cp))
    rk_adjr2 = vector(length = length(adjr2))
    rk_adjr2[order(adjr2, decreasing = TRUE)] = 1:length(adjr2)  #
highest is better
    rk_aic = as.numeric(factor(aic))
    rk_bic = as.numeric(factor(bic))

    rk_tot = rk_cp + rk_adjr2 + rk_aic + rk_bic

    # create matrix and data frame of results
    results = cbind(msize, models, cp, adjr2, aic, bic, rk_cp, rk_adjr2,
rk_aic,
        rk_bic, rk_tot)

    colnames(results)[2] = "Int"

    results_df = data.frame(results)

    # display results
    results

    # alternative x1 = vector(length=length(cp)) x1[order(cp)] =
1:length(cp)

    # Models
    cp_model = c("intercept", colnames(x)[models[order(cp)[1], ][-1]])
    adjr2_model = c("intercept", colnames(x)[models[order(adjr2,
decreasing = TRUE)[1],
        ][-1]])
    aic_model = c("intercept", colnames(x)[models[order(aic)[1], ][-1]])
    bic_model = c("intercept", colnames(x)[models[order(bic)[1], ][-1]])

    cat("best cp model:\n", cp_model, "\n")
    cat("best adjr2 model:\n", adjr2_model, "\n")
    cat("best aic model:\n", aic_model, "\n")
    cat("best bic model:\n", bic_model, "\n")

    # Order results results[order(cp),]; # order by Cp
    # results[order(adjr2,decreasing=TRUE),]; # order by adjr2
    # results[order(aic),]; # order by BIC results[order(bic),]; # order
by
    # BIC

    # alternative sort(cp, decreasing = FALSE,index.return=TRUE)$ix <->
    # order(cp)

    # plots

    plot(reg_exh, scale = "adjr2")
    plot(reg_exh, scale = "bic")
    plot(reg_exh, scale = "Cp")
```

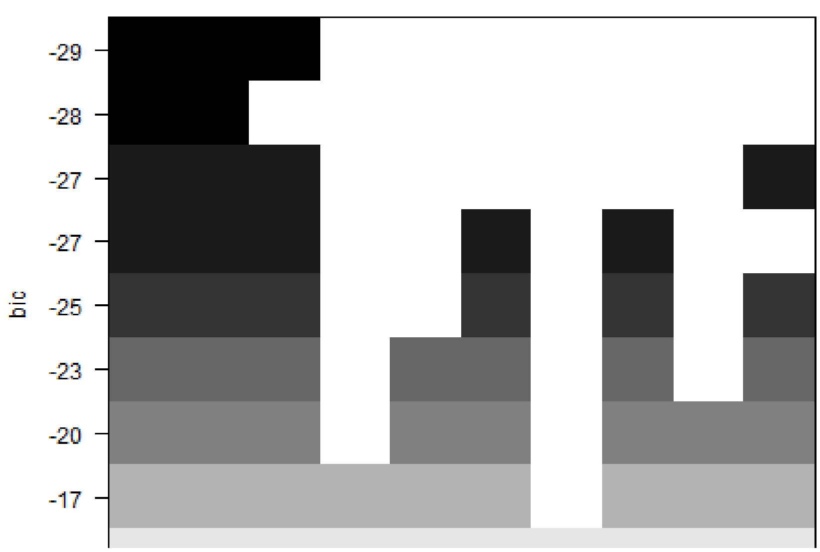
```
    localenv = environment()

    require(ggplot2)
    require(grid)
    require(gridExtra)


    plot_vector = vector(mode = "list", length = 4)

    plot_vector[[1]] = ggplot(results_df, aes(x = results_df[[1]], y =
results_df[[p +
        3]]), environment = localenv) + geom_point(size = 4) +
geom_line(aes(y = results_df[[p +
        3]]), colour = "blue") + labs(x = colnames(results_df[1]), y =
colnames(results_df[p +
        3])) + scale_x_continuous(breaks = msize) + geom_point(data =
results_df[order(cp)[1],
        ], aes(x = msize, y = cp), colour = "red", size = 5)


    plot_vector[[2]] = ggplot(results_df, aes(x = results_df[[1]], y =
results_df[[p +
        3 + 1]]), environment = localenv) + geom_point(size = 4) +
geom_line(aes(y = results_df[[p +
        3 + 1]]), colour = "blue") + labs(x = colnames(results_df[1]), y
= colnames(results_df[p +
        3 + 1])) + scale_x_continuous(breaks = msize) + geom_point(data
= results_df[order(adjr2,
        decreasing = TRUE)[1], ], aes(x = msize, y = adjr2), colour =
"red",
        size = 5)

    plot_vector[[3]] = ggplot(results_df, aes(x = results_df[[1]], y =
results_df[[p +
        3 + 2]]), environment = localenv) + geom_point(size = 4) +
geom_line(aes(y = results_df[[p +
        3 + 2]]), colour = "blue") + labs(x = colnames(results_df[1]), y
= colnames(results_df[p +
        3 + 2])) + scale_x_continuous(breaks = msize) + geom_point(data
= results_df[order(aic)[1],
        ], aes(x = msize, y = aic), colour = "red", size = 5)

    plot_vector[[4]] = ggplot(results_df, aes(x = results_df[[1]], y =
results_df[[p +
        3 + 3]]), environment = localenv) + geom_point(size = 4) +
geom_line(aes(y = results_df[[p +
        3 + 3]]), colour = "blue") + labs(x = colnames(results_df[1]), y
= colnames(results_df[p +
        3 + 3])) + scale_x_continuous(breaks = msize) + geom_point(data
= results_df[order(bic)[1],
        ], aes(x = msize, y = bic), colour = "red", size = 5)


    grid.arrange(plot_vector[[1]], plot_vector[[2]], plot_vector[[3]],
plot_vector[[4]],
        ncol = 2, main = "Model Selection")



    return(results_df)

}
```

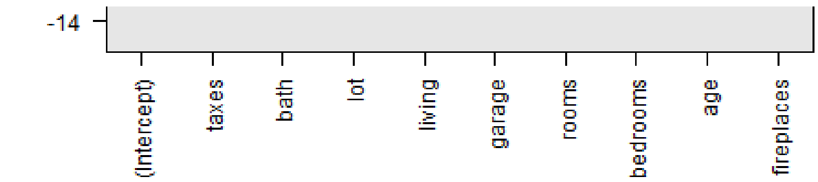
```
bestSubset = modelSelection(x, y)
```

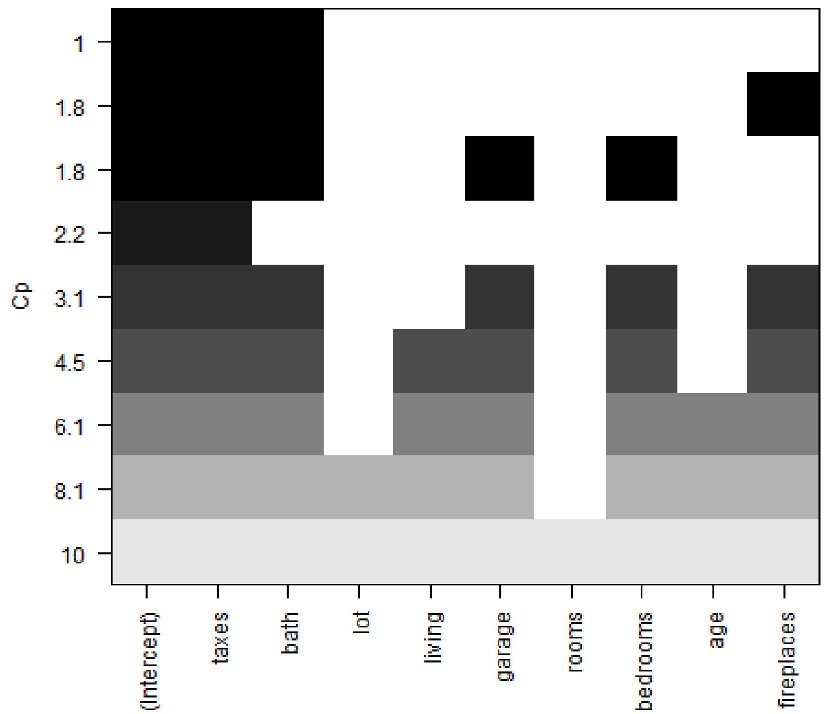
```
## Warning: package 'leaps' was built under R version 2.15.3
```

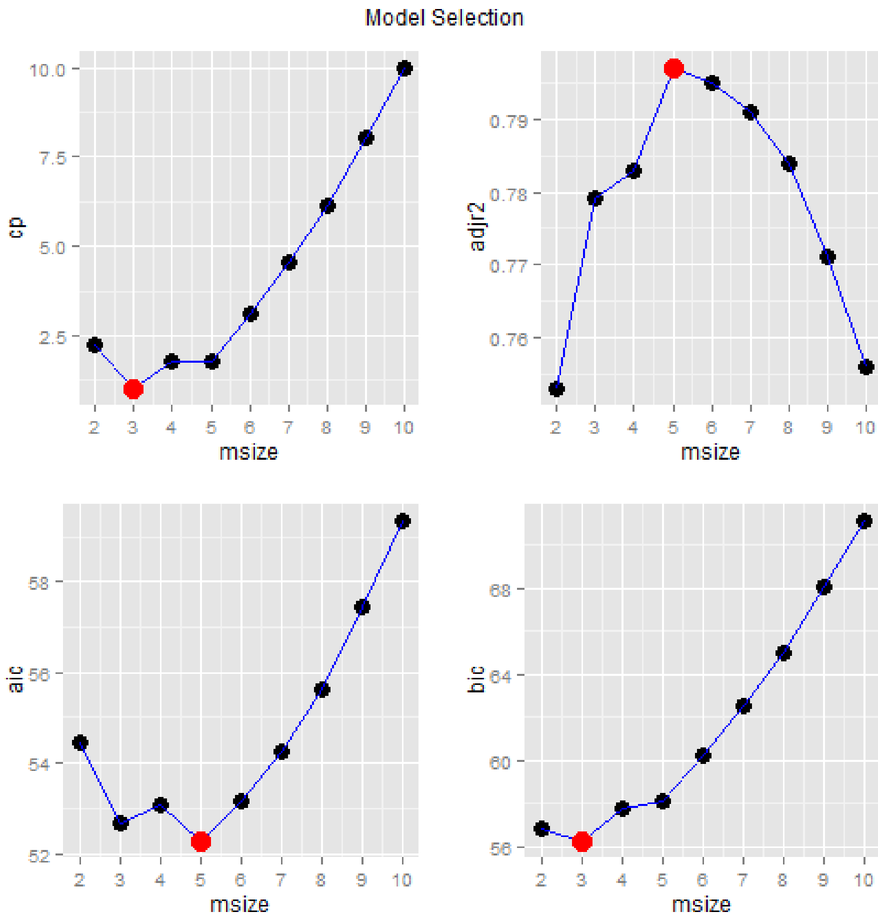
```
## best cp model:
##  intercept taxes bath
## best adjr2 model:
##  intercept taxes bath garage bedrooms
## best aic model:
##  intercept taxes bath garage bedrooms
## best bic model:
##  intercept taxes bath
```


adjr2
0.8
0.8
0.79
0.78
0.78
0.78
0.77
0.76
0.75
(Intercept)
taxes
bath
lot
living
garage
rooms
bedrooms
age
fireplaces


bic
-29
-28
-27
-27
-25
-23
-20
-17


-14
(Intercept)
taxes
bath
lot
living
garage
rooms
bedrooms
age
fireplaces


Cp
1
1.8
1.8
2.2
3.1
4.5
6.1
8.1
10
(Intercept)
taxes
bath
lot
living
garage
rooms
bedrooms
age
fireplaces


Model Selection
cp
10.0
7.5
5.0
2.5
adjr2
0.79
0.78
0.77
0.76
aic
58
56
54
52
bic
68
64
60
56
msize
2
3
4
5
6
7
8
9
10

```
bestSubset
```

```
##   msize Int taxes bath lot living garage rooms bedrooms age
fireplaces
## 1     2   1     1    0   0      0      0     0        0   0
0
## 2     3   1     1    1   0      0      0     0        0   0
0
## 3     4   1     1    1   0      0      0     0        0   0
1
## 4     5   1     1    1   0      0      1     0        1   0
0
## 5     6   1     1    1   0      0      1     0        1   0
1
## 6     7   1     1    1   0      1      1     0        1   0
1
## 7     8   1     1    1   0      1      1     0        1   1
1
## 8     9   1     1    1   1      1      1     0        1   1
1
## 9    10   1     1    1   1      1      1     1        1   1
1
##        cp adjr2   aic   bic rk_cp rk_adjr2 rk_aic rk_bic rk_tot
## 1  2.230 0.753 54.46 56.81     4        9      6      2     21
## 2  0.999 0.779 52.69 56.22     1        6      2      1     10
## 3  1.763 0.783 53.08 57.79     2        5      3      3     13
## 4  1.795 0.797 52.26 58.15     3        1      1      4      9
## 5  3.091 0.795 53.16 60.23     5        2      4      5     16
## 6  4.524 0.791 54.24 62.49     6        3      5      6     20
## 7  6.143 0.784 55.60 65.03     7        4      7      7     25
## 8  8.054 0.771 57.45 68.06     8        7      8      8     31
## 9 10.000 0.756 59.36 71.14     9        8      9      9     35
```

Stepwise regression shows that the best model is: X1 (taxes), X2 (bathrooms), X5 (garage) and X7 (bedrooms).

The model by the expert(sales ~ taxes (x1)) is ranked 4th in cp, last in adjusted $r^{2,}$ sixth in aic and 2nd in BIC. Thus, comparing it with other models, the assertion by the expert that the building characteristics are redundant does not hold. For instance, the most adequate models (include intercept) seem to be:

- 1) X1 (taxes) and X2 (bathrooms), -> best in cp and bic criteria
- 2) X1 (taxes), X2 (bathrooms), X5 (garage) and X7 (bedrooms) -> best in adjr^2 and aic criteria

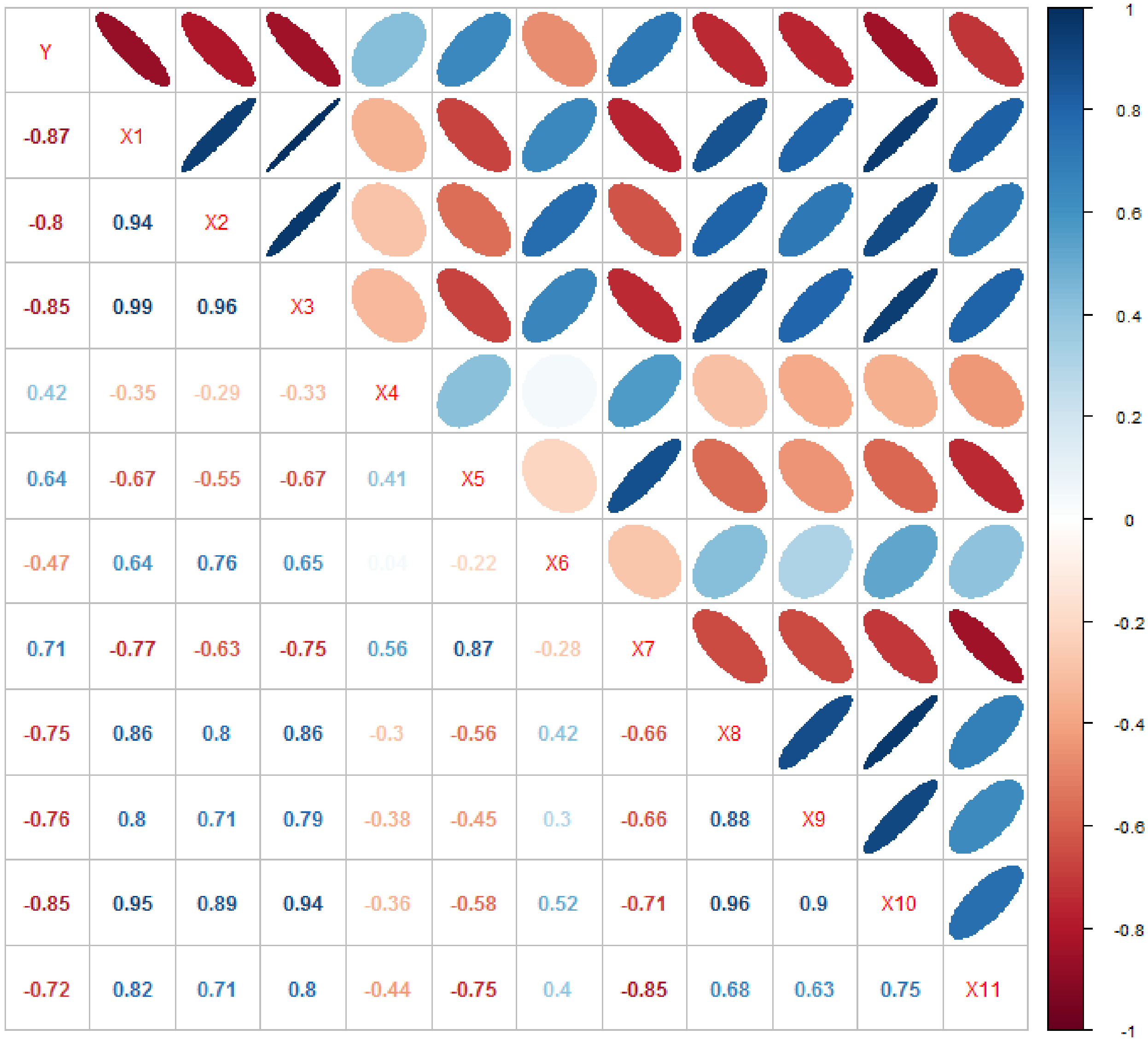
In addition to taxes, the two models include bathroom as a signficant predictor of price, and one model includes garage and bedroom as well. Model X1 (taxes), X2 (bathrooms), X5 (garage) and X7 (bedrooms) indicates that garage and bedrooms are not significant. Thus, the most adequate model for predicting sales price seems to be X1 (taxes) and X2 (bathrooms).

Problem 5

```
# Import data
filename = "P256.txt"
mydata = read.table(filename, header = T)

# Look at data
names(mydata)
head(mydata)
nrow(mydata)
summary(mydata)

# Fix names
names(mydata)[-1] = sapply(1:11, function(i) paste("X", i, sep = ""))
```

Part a

```
corr = round(cor(mydata), 2)

library(corrplot)
corrplot.mixed(corr, upper = "ellipse", lower = "number")
```




```
# pairs(mydata[,-1], main = 'Correlation coeffficients matrix and
scatter
# plot', pch = 21, lower.panel = NULL, panel = panel.smooth,
cex.labels=2)
```

The pairwise correlation coefficeints of the predictor vairables and the corresponding scatter plots show strong linear relationships among some pairs of predictors variables, suggesting collinearity.(look at high magnitudes for correlation coefficient in conjuction for a trend in the scatter plot)

For example, X1 is strongly correlated with X2,X3, X8, X9, X10 and X11. If all of these are included in the model, the the non-collinearity assumption of the predictors might be violated.

```
fit = lm(Y ~ ., mydata)
summary(fit)
```

```
##
## Call:
## lm(formula = Y ~ ., data = mydata)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -5.35  -1.62  -0.60   1.52   5.28
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 17.77320   30.50877    0.58    0.567
## X1          -0.07795    0.05861   -1.33    0.200
## X2          -0.07340    0.08892   -0.83    0.420
## X3           0.12111    0.09135    1.33    0.201
## X4           1.32903    3.09954    0.43    0.673
## X5           5.97599    3.15865    1.89    0.075 .
## X6           0.30418    1.28909    0.24    0.816
## X7          -3.19858    3.10544   -1.03    0.317
## X8           0.18536    0.12925    1.43    0.169
## X9          -0.39915    0.32381   -1.23    0.234
## X10         -0.00519    0.00589   -0.88    0.390
## X11          0.59865    3.02068    0.20    0.845
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.23 on 18 degrees of freedom
## Multiple R-squared: 0.835,   Adjusted R-squared: 0.735
## F-statistic:  8.3 on 11 and 18 DF,  p-value: 5.29e-05
```

```
# Compute VIF
library(car)
vif(fit)
```

```
##      X1      X2      X3      X4      X5      X6      X7      X8
X9
## 128.835  43.921 160.436   2.058   7.781   5.327  11.735  20.586
9.419
##     X10     X11
##  85.676   5.143
```

```
# Determine VIF > 10
names(vif(fit))[vif(fit) > 10]
```

```
## [1] "X1"  "X2"  "X3"  "X7"  "X8"  "X10"
```

Fitting a linear model with all predictors and computing VIF confirms our suspicion. It appears that X1, X2, X3, X7, X8 and X10 are affected by the presence of collinearity because VIF > 10. Thus, there is a multicollinearity problem if all variables are included. Thus, do not include all of them because of multicollinearity In addition, if all variables in the model are included,none of the variables are significant (p-value > 0.05)

Part b

```
x = mydata[, 2:12]  # design matrix or use model.matrix(fullfit)
y = mydata[, 1]  # response vector
# x$X12=x$X2*x$X10 x$X13=x$X8/x$X10

n = length(y)  # number of observations
p = dim(x)[2]  # number of predictors


models = vector(mode = "list", length = 6)
models[[1]] = lm(Y ~ X1, mydata)
models[[2]] = lm(Y ~ X10, mydata)
models[[3]] = lm(Y ~ X1 + X10, mydata)
models[[4]] = lm(Y ~ X2 + X10, mydata)
models[[5]] = lm(Y ~ X8 + X10, mydata)
models[[6]] = lm(Y ~ X8 + X5 + X10, mydata)

full = lm(Y ~ ., mydata)

# compute the selection model criteria input = lm object for desired
model
# and full model
computeCriteria = function(fit, full) {
    n = length(summary(fit)$res)
    msize = dim(summary(fit)$coeff)[1]
    RSS = sum(summary(fit)$residuals^2)
    cp = RSS/summary(full)$sigma^2 + 2 * msize - n
    adjr2 = summary(fit)$adj.r
    aic = n * log(RSS/n) + 2 * msize
    bic = n * log(RSS/n) + msize * log(n)

    return(round(c(cp, adjr2, aic, bic), 3))

}

results = matrix(, nrow = 6, ncol = 4)

for (i in 1:6) {
    results[i, ] = computeCriteria(models[[i]], full)
}

colnames(results) = c("cp", "adjr2", "aic", "bic")
rownames(results) = 1:6
results
```

```
##       cp adjr2   aic   bic
## 1  0.218 0.751 70.24 73.04
## 2  3.844 0.717 74.12 76.93
## 3  1.660 0.748 71.59 75.80
## 4  5.036 0.715 75.30 79.50
## 5  0.992 0.754 70.80 75.00
## 6 -0.524 0.781 68.25 73.86
```

```
# get best model from six
rownames(results)[order(results[, "cp"])[1]]
```

```
## [1] "6"
```

```
rownames(results)[order(results[, "adjr2"], decreasing = TRUE)[1]]
```

```
## [1] "6"
```

```
rownames(results)[order(results[, "aic"])[1]]
```

```
## [1] "6"
```

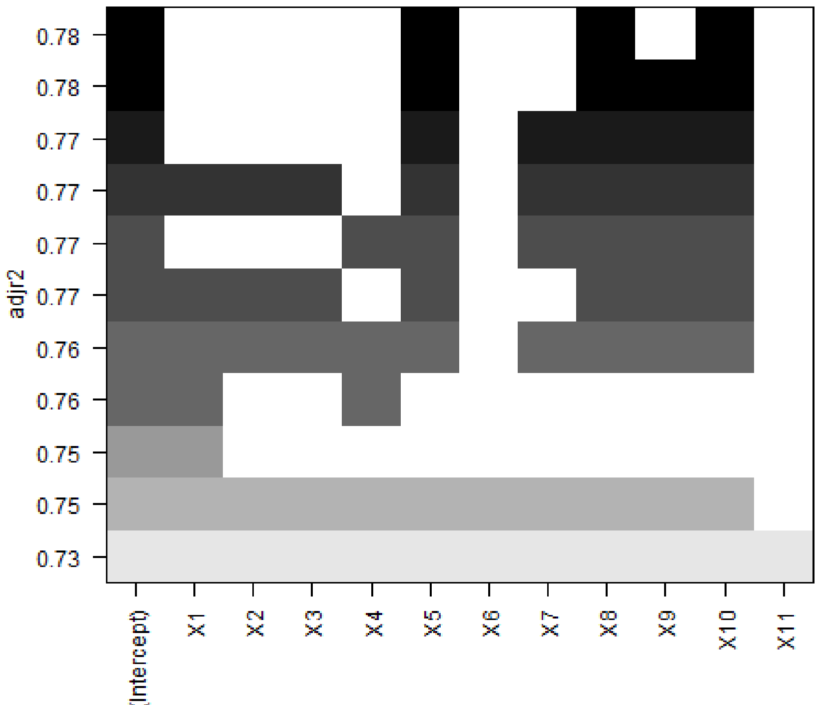
```
rownames(results)[order(results[, "bic"])[1]]
```

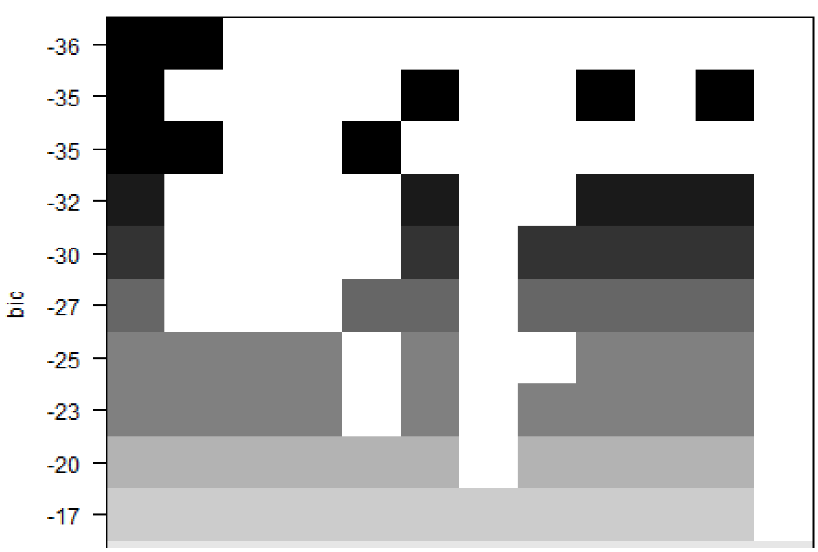
```
## [1] "1"
```

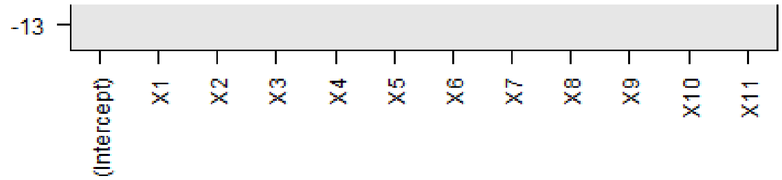
Among the six regression models, model 6 (X8,X5,X10) is the best in predicting Y because it has the highest adjr2, lowest cp, lowest aic and second-to-lowest bic (very close to the lowest bic).

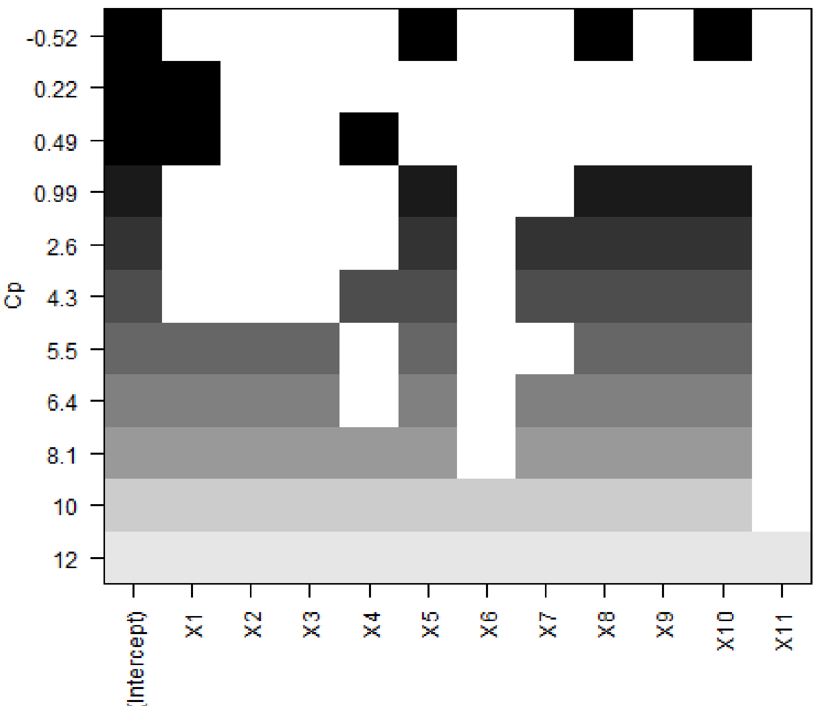
```
# find a better model
bestSubset = modelSelection(x, y)
```

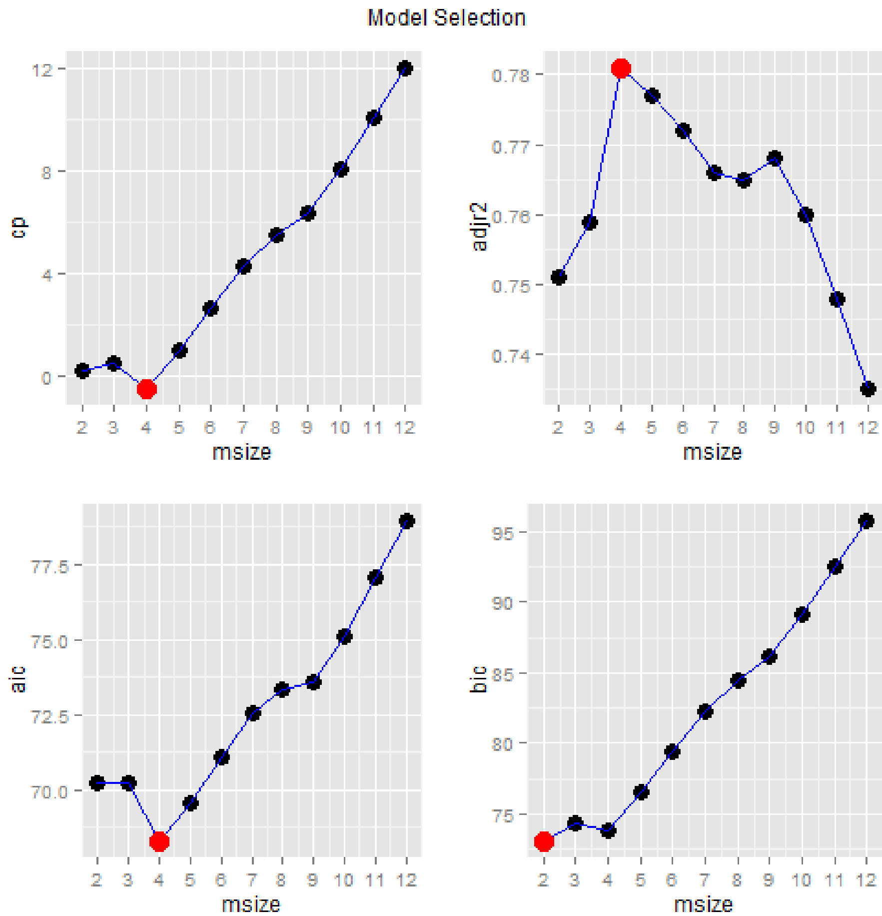
```
## best cp model:
##  intercept X5 X8 X10
## best adjr2 model:
##  intercept X5 X8 X10
## best aic model:
##  intercept X5 X8 X10
## best bic model:
##  intercept X1
```


adjr2
0.78
0.78
0.77
0.77
0.77
0.77
0.76
0.76
0.75
0.75
0.73
(Intercept)
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11


bic
-36
-35
-35
-32
-30
-27
-25
-23
-20
-17


-13
(Intercept)
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11


Cp
-0.52
0.22
0.49
0.99
2.6
4.3
5.5
6.4
8.1
10
12
(Intercept)
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11




```
bestSubset
```

```
##    msize Int X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11     cp adjr2   aic
bic
## 1      2   1  1  0  0  0  0  0  0  0  0   0   0  0.218 0.751 70.24
73.04
## 2      3   1  1  0  0  1  0  0  0  0  0   0   0  0.495 0.759 70.20
74.40
## 3      4   1  0  0  0  0  1  0  0  1  0   1   0 -0.524 0.781 68.25
73.86
## 4      5   1  0  0  0  0  1  0  0  1  1   1   0  0.992 0.777 69.57
76.58
## 5      6   1  0  0  0  0  1  0  1  1  1   1   0  2.633 0.772 71.05
79.46
## 6      7   1  0  0  0  1  1  0  1  1  1   1   0  4.268 0.766 72.52
82.33
## 7      8   1  1  1  1  0  1  0  0  1  1   1   0  5.469 0.765 73.31
84.52
## 8      9   1  1  1  1  0  1  0  1  1  1   1   0  6.362 0.768 73.55
86.17
## 9     10   1  1  1  1  1  1  0  1  1  1   1   0  8.087 0.760 75.10
89.11
## 10    11   1  1  1  1  1  1  1  1  1  1   1   0 10.039 0.748 77.02
92.44
## 11    12   1  1  1  1  1  1  1  1  1  1   1   1 12.000 0.735 78.96
95.77
##    rk_cp rk_adjr2 rk_aic rk_bic rk_tot
## 1      2        9      4      1     16
## 2      3        8      3      3     17
## 3      1        1      1      2      5
## 4      4        2      2      4     12
## 5      5        3      5      5     18
## 6      6        5      6      6     23
## 7      7        6      7      7     27
## 8      8        4      8      8     28
## 9      9        7      9      9     34
## 10    10       10     10     10     40
## 11    11       11     11     11     44
```

Comparing the best models of each model size, we see that the same model (X8,X5,X10) is the best model in terms of adjr2, cp and aic.The bic is also very close to the best model. Thus, no other better models can be suggested (this is if assuming no transformation or higher order terms or interactions terms are considered). If for example an interaction is allowed (e.g consider x12=x2*x10), then the best model in terms of cp would be intercept X2 X8 X10 X11 X12. There are many more interactions and transformations (e.g. x8/x10) that can be tested if one wants to reallly find the best model.

Stepwise regression to determine best model, start with all variables

```
library(MASS)
fit_stepAIC = step(object = full, direction = "both")  # AIC
```

```
## Start:  AIC=78.96
## Y ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 + X10 + X11
##
##        Df Sum of Sq RSS  AIC
## - X11   1       0.4 188 77.0
## - X6    1       0.6 188 77.0
## - X4    1       1.9 189 77.3
## - X2    1       7.1 194 78.1
## - X10   1       8.1 196 78.2
## - X7    1      11.0 198 78.7
## <none>              187 79.0
```

```
## - X9    1      15.8 203 79.4
## - X3    1      18.3 206 79.8
## - X1    1      18.4 206 79.8
## - X8    1      21.4 209 80.2
## - X5    1      37.3 225 82.4
##
## Step:  AIC=77.02
## Y ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 + X10
##
##        Df Sum of Sq RSS  AIC
## - X6    1       0.5 188 75.1
## - X4    1       1.9 190 75.3
## - X2    1       7.2 195 76.2
## - X10   1       8.2 196 76.3
## <none>              188 77.0
## - X7    1      14.0 202 77.2
## - X9    1      16.9 205 77.6
## - X1    1      18.0 206 77.8
## - X3    1      18.3 206 77.8
## - X8    1      21.6 209 78.3
## + X11   1       0.4 187 79.0
## - X5    1      37.7 226 80.5
##
## Step:  AIC=75.1
## Y ~ X1 + X2 + X3 + X4 + X5 + X7 + X8 + X9 + X10
##
##        Df Sum of Sq RSS  AIC
## - X4    1       2.9 191 73.6
## - X2    1       7.9 196 74.3
## - X10   1       9.6 198 74.6
## <none>              188 75.1
## - X7    1      14.4 203 75.3
## - X9    1      18.1 206 75.9
## - X3    1      18.5 207 75.9
## - X1    1      18.6 207 75.9
## - X8    1      22.5 211 76.5
## + X6    1       0.5 188 77.0
## + X11   1       0.3 188 77.0
## - X5    1      40.4 229 78.9
##
## Step:  AIC=73.55
## Y ~ X1 + X2 + X3 + X5 + X7 + X8 + X9 + X10
##
##        Df Sum of Sq RSS  AIC
## - X7    1      11.5 203 73.3
## - X2    1      11.6 203 73.3
## <none>              191 73.6
## - X10   1      14.4 206 73.7
## - X1    1      17.0 208 74.1
## - X9    1      18.2 209 74.3
## - X3    1      22.1 213 74.8
## + X4    1       2.9 188 75.1
## + X6    1       1.5 190 75.3
## + X11   1       0.2 191 75.5
## - X8    1      29.3 220 75.8
## - X5    1      41.5 233 77.4
##
## Step:  AIC=73.31
## Y ~ X1 + X2 + X3 + X5 + X8 + X9 + X10
##
##        Df Sum of Sq RSS  AIC
## - X1    1       8.7 211 72.6
## - X9    1       9.1 212 72.6
## - X2    1      10.6 213 72.8
```

```
## <none>                   203 73.3
## - X3    1       15.1 218 73.5
## + X7    1       11.5 191 73.6
## - X10   1       19.5 222 74.1
## + X11   1        2.9 200 74.9
## - X8    1       27.6 230 75.1
## + X6    1        0.9 202 75.2
## + X4    1        0.0 203 75.3
## - X5    1       34.5 237 76.0
##
## Step:  AIC=72.57
## Y ~ X2 + X3 + X5 + X8 + X9 + X10
##
##         Df Sum of Sq RSS  AIC
## - X9    1        6.1 217 71.4
## - X3    1        6.5 218 71.5
## - X2    1        6.9 218 71.5
## <none>                   211 72.6
## + X1    1        8.7 203 73.3
## + X7    1        3.2 208 74.1
## + X11   1        0.5 211 74.5
## + X6    1        0.1 211 74.5
## + X4    1        0.1 211 74.5
## - X5    1       38.9 250 75.6
## - X10   1       54.5 266 77.5
## - X8    1       55.8 267 77.6
##
## Step:  AIC=71.42
## Y ~ X2 + X3 + X5 + X8 + X10
##
##         Df Sum of Sq RSS  AIC
## - X2    1        4.5 222 70.0
## - X3    1        6.1 224 70.3
## <none>                   217 71.4
## + X9    1        6.1 211 72.6
## + X1    1        5.7 212 72.6
## + X4    1        0.8 217 73.3
## + X7    1        0.5 217 73.3
## + X11   1        0.5 217 73.3
## + X6    1        0.1 217 73.4
## - X5    1       35.4 253 73.9
## - X8    1       53.7 271 76.0
## - X10   1       87.5 305 79.6
##
## Step:  AIC=70.03
## Y ~ X3 + X5 + X8 + X10
##
##         Df Sum of Sq RSS  AIC
## - X3    1        1.7 224 68.3
## <none>                   222 70.0
## + X2    1        4.5 217 71.4
## + X9    1        3.7 218 71.5
## + X1    1        3.6 218 71.5
## + X4    1        1.7 220 71.8
## + X11   1        1.1 221 71.9
## + X7    1        1.1 221 71.9
## + X6    1        0.5 221 72.0
## - X5    1       34.2 256 72.3
## - X8    1       50.4 272 74.2
## - X10   1       84.7 306 77.7
##
## Step:  AIC=68.25
## Y ~ X5 + X8 + X10
##
```

```
##        Df Sum of Sq RSS  AIC
## <none>              224 68.3
## + X9    1       5.0 219 69.6
## + X4    1       2.7 221 69.9
## + X3    1       1.7 222 70.0
## + X11   1       1.3 222 70.1
## + X7    1       0.6 223 70.2
## + X6    1       0.1 223 70.2
## + X2    1       0.0 224 70.3
## + X1    1       0.0 224 70.3
## - X5    1      36.6 260 70.8
## - X8    1      53.1 277 72.6
## - X10   1     194.7 418 85.0
```

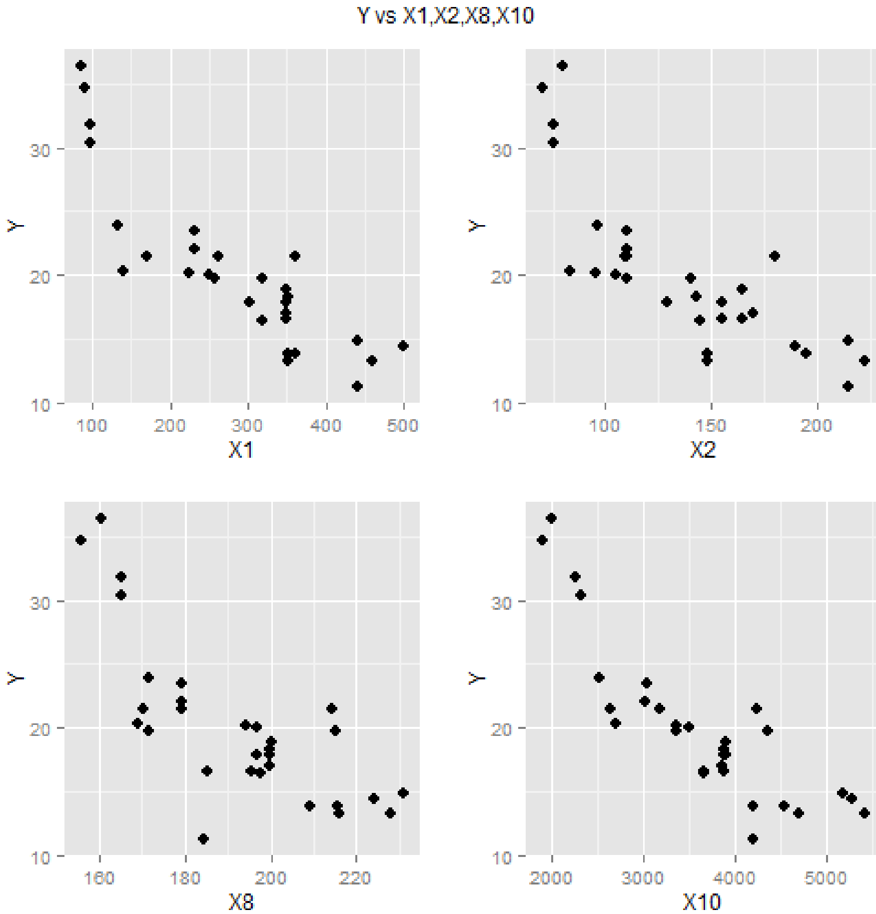
```
summary(fit_stepAIC)
```

```
##
## Call:
## lm(formula = Y ~ X5 + X8 + X10, data = mydata)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.593 -1.967 -0.644  2.031  5.882
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.49497   11.76475    0.38    0.706
## X5           2.60734    1.26379    2.06    0.049 *
## X8           0.21812    0.08776    2.49    0.020 *
## X10         -0.00948    0.00199   -4.76  6.4e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.93 on 26 degrees of freedom
## Multiple R-squared: 0.803,   Adjusted R-squared: 0.781
## F-statistic: 35.4 on 3 and 26 DF,  p-value: 2.47e-09
```

Using a stepwise regression confirms the conclusions reached above

Part c

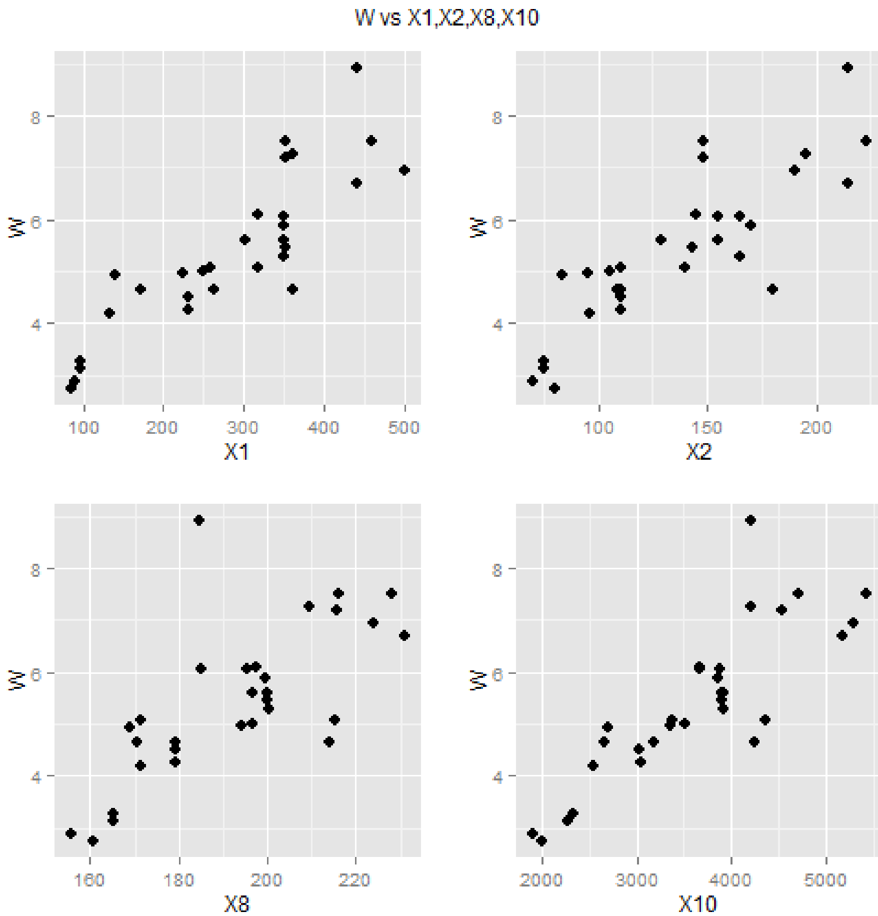
```
plot1 = ggplot(mydata, aes(x = X1, y = Y)) + geom_point(size = 3)

plot2 = ggplot(mydata, aes(x = X2, y = Y)) + geom_point(size = 3)

plot3 = ggplot(mydata, aes(x = X8, y = Y)) + geom_point(size = 3)

plot4 = ggplot(mydata, aes(x = X10, y = Y)) + geom_point(size = 3)

grid.arrange(plot1, plot2, plot3, plot4, ncol = 2, main = "Y vs
X1,X2,X8,X10")
```



The plots suggets that the relationship between Y and X1,X2,X8 and X10 (individually) is not linear. It seems that the relationship is hyperbolic (i.e. 1/x)

Part d

```
mydata$W = 100/mydata$Y

plot1 = ggplot(mydata, aes(x = X1, y = W)) + geom_point(size = 3)
plot2 = ggplot(mydata, aes(x = X2, y = W)) + geom_point(size = 3)
plot3 = ggplot(mydata, aes(x = X8, y = W)) + geom_point(size = 3)
plot4 = ggplot(mydata, aes(x = X10, y = W)) + geom_point(size = 3)
grid.arrange(plot1, plot2, plot3, plot4, ncol = 2, main = "W vs
X1,X2,X8,X10")
```



The plots now suggets that the relationship between W and X1,X2,X8 and X10 (individually) is more linear than that between Y and the variables.

Part e

```
w = mydata$W  # response vector

models = vector(mode = "list", length = 6)
models[[1]] = lm(W ~ X1, mydata)
models[[2]] = lm(W ~ X10, mydata)
models[[3]] = lm(W ~ X1 + X10, mydata)
models[[4]] = lm(W ~ X2 + X10, mydata)
models[[5]] = lm(W ~ X8 + X10, mydata)
models[[6]] = lm(W ~ X8 + X5 + X10, mydata)

full = lm(W ~ ., mydata)

# compute the selection model criteria input = lm object for desired
model
# and full model

results = matrix(, nrow = 6, ncol = 4)

for (i in 1:6) {
    results[i, ] = computeCriteria(models[[i]], full)
}

colnames(results) = c("cp", "adjr2", "aic", "bic")
rownames(results) = 1:6
results
```

```
##      cp adjr2    aic     bic
## 1 156.8 0.743 -15.63 -12.833
## 2 183.9 0.705 -11.49  -8.686
## 3 155.9 0.738 -14.13  -9.923
## 4 154.8 0.740 -14.31 -10.107
## 5 113.2 0.800 -22.25 -18.042
## 6 114.8 0.793 -20.33 -14.725
```

```
# get best model from six
rownames(results)[order(results[, "cp"])[1]]
```

```
## [1] "5"
```

```
rownames(results)[order(results[, "adjr2"], decreasing = TRUE)[1]]
```

```
## [1] "5"
```

```
rownames(results)[order(results[, "aic"])[1]]
```

```
## [1] "5"
```

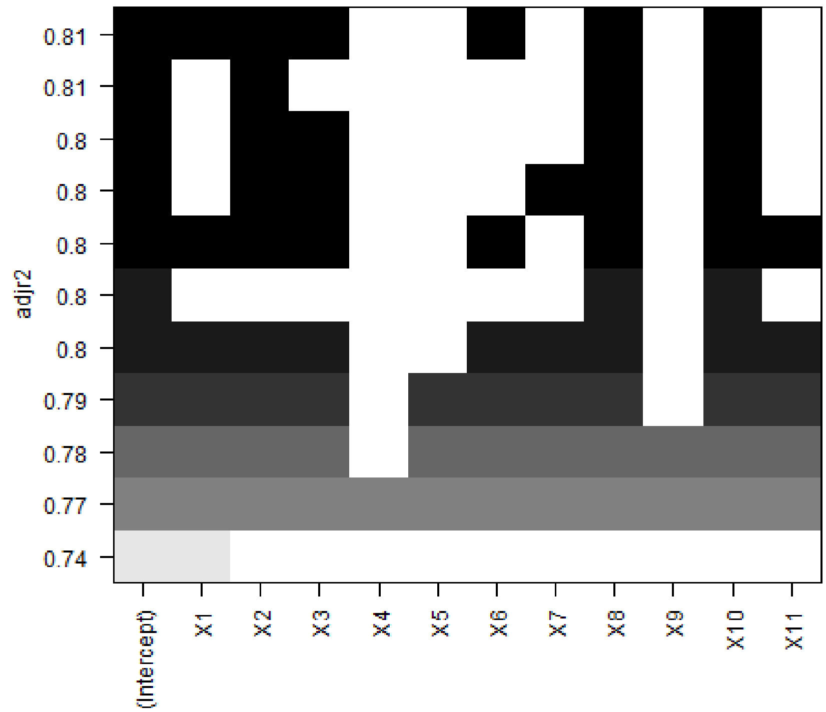
```
rownames(results)[order(results[, "bic"])[1]]
```

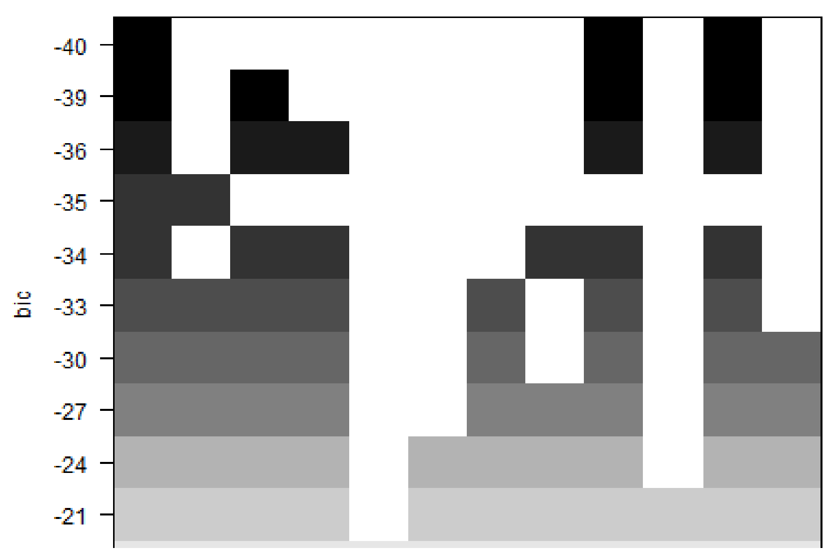
```
## [1] "5"
```

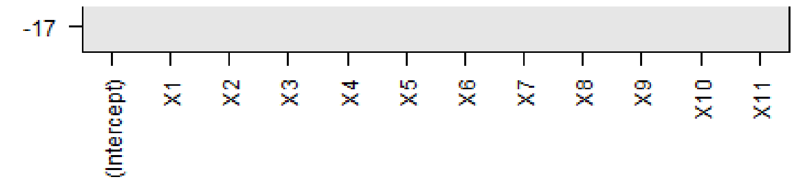
Among the six regression models, model 5 (x8,10) is the best in predicting W because it has the highest adjr2, lowest cp,lowest aic and lowest bic. This answer is different from (b).

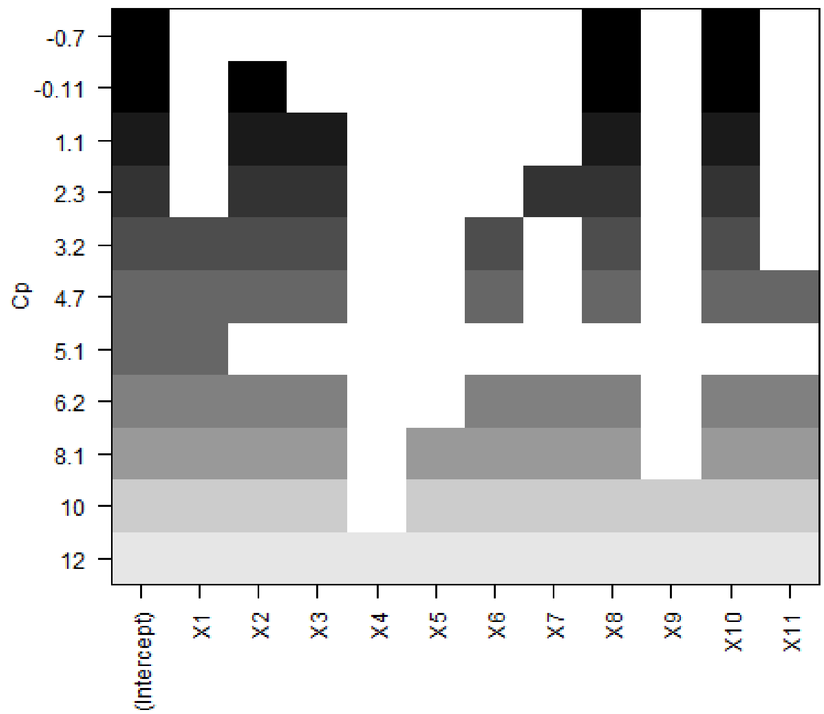
```
# find a better model
bestSubset = modelSelection(x, w)
```

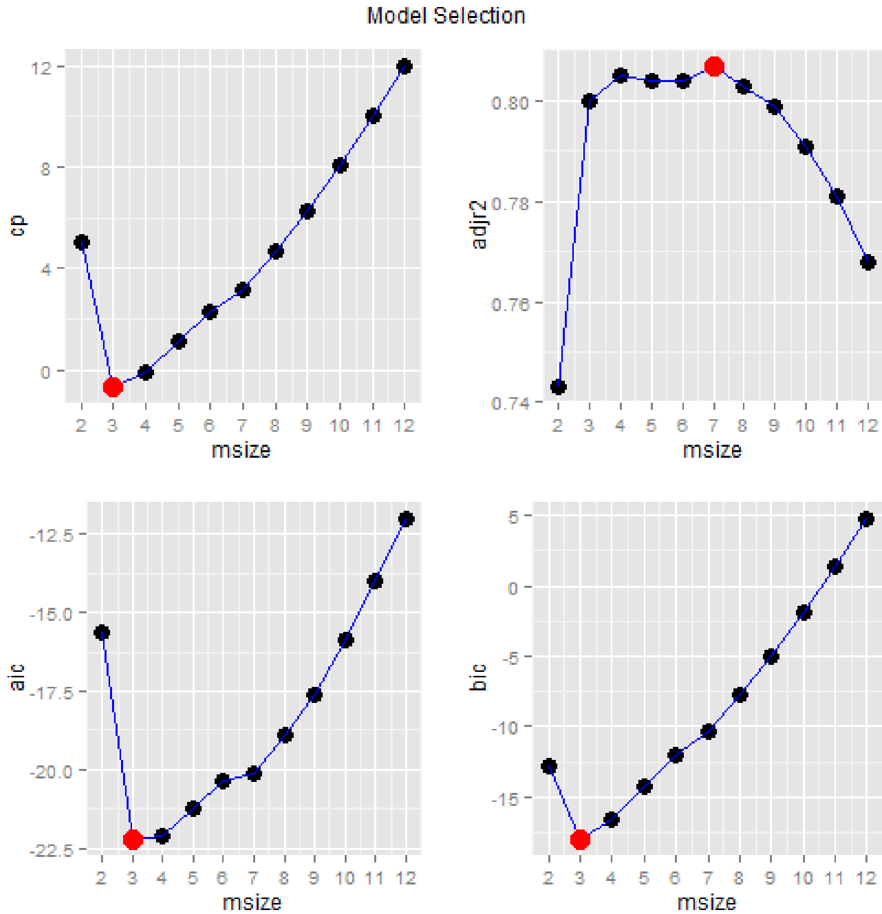
```
## best cp model:
##  intercept X8 X10
## best adjr2 model:
##  intercept X1 X2 X3 X6 X8 X10
## best aic model:
##  intercept X8 X10
## best bic model:
##  intercept X8 X10
```


adjr2
0.81
0.81
0.8
0.8
0.8
0.8
0.8
0.79
0.78
0.77
0.74
(Intercept)
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11


bic
-40
-39
-36
-35
-34
-33
-30
-27
-24
-21


-17
(Intercept)
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11


Cp
-0.7
-0.11
1.1
2.3
3.2
4.7
5.1
6.2
8.1
10
12
(Intercept)
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11




```
bestSubset
```

```
##    msize Int X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11     cp adjr2    aic
## 1      2   1  1  0  0  0  0  0  0  0  0   0   0  5.051 0.743 -15.63
## 2      3   1  0  0  0  0  0  0  0  1  0   1   0 -0.696 0.800 -22.25
## 3      4   1  0  1  0  0  0  0  0  1  0   1   0 -0.112 0.805 -22.13
## 4      5   1  0  1  1  0  0  0  0  1  0   1   0  1.109 0.804 -21.21
## 5      6   1  0  1  1  0  0  0  1  1  0   1   0  2.311 0.804 -20.37
## 6      7   1  1  1  1  0  0  1  0  1  0   1   0  3.159 0.807 -20.12
## 7      8   1  1  1  1  0  0  1  0  1  0   1   1  4.682 0.803 -18.88
## 8      9   1  1  1  1  0  0  1  1  1  0   1   1  6.231 0.799 -17.61
## 9     10   1  1  1  1  0  1  1  1  1  0   1   1  8.076 0.791 -15.87
## 10    11   1  1  1  1  0  1  1  1  1  1   1   1 10.002 0.781 -13.99
## 11    12   1  1  1  1  1  1  1  1  1  1   1   1 12.000 0.768 -11.99
##         bic rk_cp rk_adjr2 rk_aic rk_bic rk_tot
## 1   -12.833     7       11      9      4     31
## 2   -18.042     1        6      1      1      9
## 3   -16.521     2        2      2      2      8
## 4   -14.207     3        3      3      3     12
## 5   -11.963     4        4      4      5     17
## 6   -10.313     5        1      5      6     17
## 7    -7.667     6        5      6      7     24
## 8    -5.000     8        7      7      8     30
## 9    -1.854     9        8      8      9     34
## 10    1.423    10        9     10     10     39
## 11    4.821    11       10     11     11     43
```

Comparing the best models of each model size, we see that the same model (X8,X10) is the best model in terms of bic, cp and aic. The adjr2 is also very close to the best. Thus, no other better models can be suggested (assuming no transformation or higher order terms or interactions terms are considered).

```
# Stepwise regression to determine best model, start with all variables
library(MASS)
fit_stepAIC = step(object = full, direction = "both")  # AIC
```

```
## Start:  AIC=-64.9
## W ~ Y + X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 + X10 + X11
##
##         Df Sum of Sq  RSS   AIC
## - X3     1      0.00 1.45 -66.9
## - X1     1      0.02 1.47 -66.6
## - X4     1      0.06 1.51 -65.7
## <none>               1.45 -64.9
## - X8     1      0.16 1.61 -63.7
## - X10    1      0.24 1.69 -62.4
## - X11    1      0.29 1.74 -61.5
## - X6     1      0.33 1.78 -60.7
## - X2     1      0.33 1.78 -60.7
## - X9     1      0.91 2.36 -52.3
## - X7     1      1.34 2.79 -47.3
## - X5     1      2.04 3.49 -40.6
## - Y      1      7.59 9.04 -12.0
##
## Step:  AIC=-66.88
## W ~ Y + X1 + X2 + X4 + X5 + X6 + X7 + X8 + X9 + X10 + X11
##
##         Df Sum of Sq  RSS   AIC
## - X1     1      0.03 1.48 -68.3
## - X4     1      0.07 1.52 -67.5
## <none>               1.45 -66.9
## - X8     1      0.17 1.62 -65.6
## + X3     1      0.00 1.45 -64.9
## - X10    1      0.24 1.69 -64.4
## - X11    1      0.29 1.74 -63.5
## - X6     1      0.39 1.84 -61.8
## - X2     1      0.94 2.39 -53.9
## - X9     1      0.97 2.42 -53.6
## - X7     1      1.39 2.84 -48.8
## - X5     1      2.57 4.02 -38.3
## - Y      1      8.29 9.74 -11.8
##
## Step:  AIC=-68.33
## W ~ Y + X2 + X4 + X5 + X6 + X7 + X8 + X9 + X10 + X11
##
##         Df Sum of Sq  RSS   AIC
## - X4     1      0.05 1.53 -69.4
## <none>               1.48 -68.3
## - X8     1      0.14 1.62 -67.6
## + X1     1      0.03 1.45 -66.9
## + X3     1      0.01 1.47 -66.6
## - X10    1      0.28 1.75 -65.2
## - X11    1      0.36 1.83 -63.8
## - X6     1      0.46 1.94 -62.2
## - X9     1      0.94 2.42 -55.6
## - X2     1      1.01 2.49 -54.7
## - X7     1      1.40 2.88 -50.3
## - X5     1      2.72 4.20 -39.0
## - Y      1      8.29 9.77 -13.7
```

```
##
## Step:  AIC=-69.36
## W ~ Y + X2 + X5 + X6 + X7 + X8 + X9 + X10 + X11
##
##        Df Sum of Sq  RSS   AIC
## <none>              1.53 -69.4
## - X8    1      0.13 1.66 -68.9
## + X4    1      0.05 1.48 -68.3
## + X1    1      0.01 1.52 -67.5
## + X3    1      0.00 1.53 -67.4
## - X10   1      0.28 1.81 -66.3
## - X11   1      0.34 1.87 -65.3
## - X6    1      0.41 1.94 -64.2
## - X9    1      0.91 2.43 -57.3
## - X2    1      0.96 2.49 -56.7
## - X7    1      1.42 2.95 -51.6
## - X5    1      2.71 4.24 -40.7
## - Y     1      8.30 9.82 -15.5
```

```
summary(fit_stepAIC)
```

```
##
## Call:
## lm(formula = W ~ Y + X2 + X5 + X6 + X7 + X8 + X9 + X10 + X11,
##     data = mydata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.4722 -0.1843  0.0068  0.1592  0.4520
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.942140   1.863459    7.48  3.2e-07 ***
## Y           -0.196935   0.018882  -10.43  1.6e-09 ***
## X2           0.014732   0.004147    3.55  0.00200 **
## X5           1.490785   0.250047    5.96  7.9e-06 ***
## X6          -0.219383   0.094632   -2.32  0.03114 *
## X7          -1.012612   0.234490   -4.32  0.00033 ***
## X8          -0.013644   0.010361   -1.32  0.20277
## X9          -0.094223   0.027306   -3.45  0.00253 **
## X10          0.000693   0.000361    1.92  0.06885 .
## X11         -0.530714   0.249935   -2.12  0.04639 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.276 on 20 degrees of freedom
## Multiple R-squared: 0.976,   Adjusted R-squared: 0.965
## F-statistic: 89.4 on 9 and 20 DF,  p-value: 3.58e-14
```

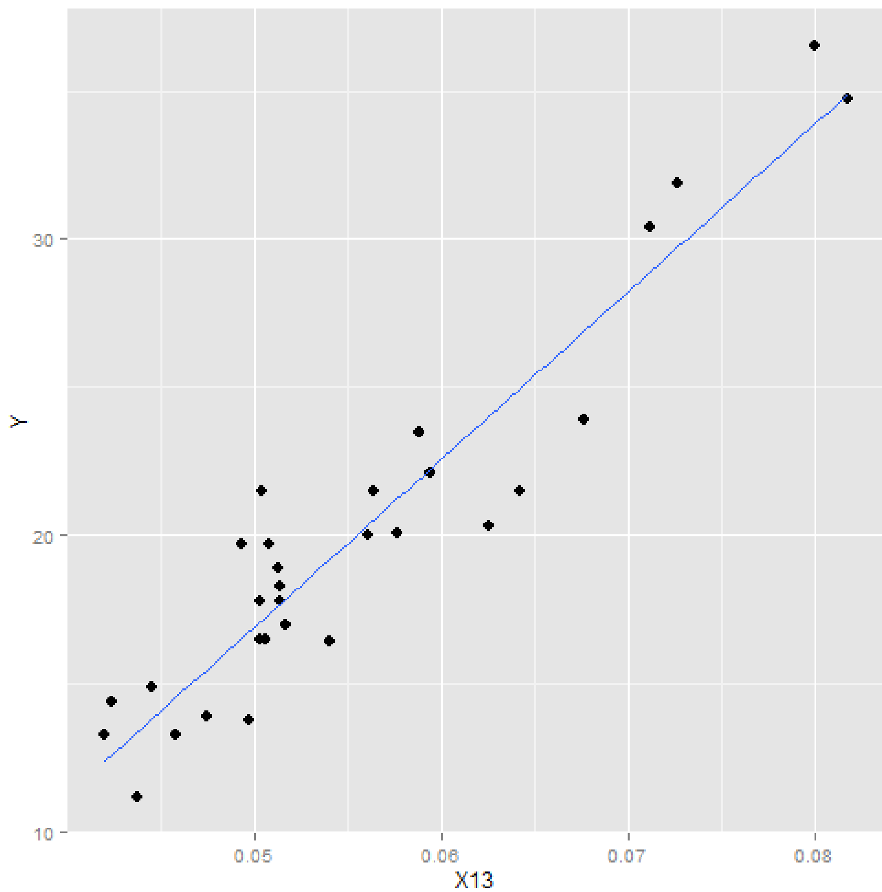
Using a stepwise regression shows a different model, however this model has insignificant terms and also ranks lower in other criteria and also has too many predictors.So the (X8,X10) model is preferred. In conclusion, the transformation of the variable makes a difference in variable selection so it should be examined carefully.

Part f

```
mydata$X13 = mydata$X8/mydata$X10

fit = lm(Y ~ X13, mydata)
summary(fit)
```

```
##
## Call:
## lm(formula = Y ~ X13, data = mydata)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.713 -1.246 -0.023  1.421  4.346
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -11.4        2.1   -5.41  8.9e-06 ***
## X13            566.0       37.2   15.21  4.6e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.09 on 28 degrees of freedom
## Multiple R-squared: 0.892,   Adjusted R-squared: 0.888
## F-statistic:  231 on 1 and 28 DF,  p-value: 4.59e-15
```

```
ggplot(mydata, aes(x = X13, y = Y)) + geom_point(size = 3) +
stat_smooth(method = "lm",
    se = FALSE)
```



The model seems to be very good in terms of R^2 and fit to the data.