HDS Exercise set 3

Shabbeer Hassan

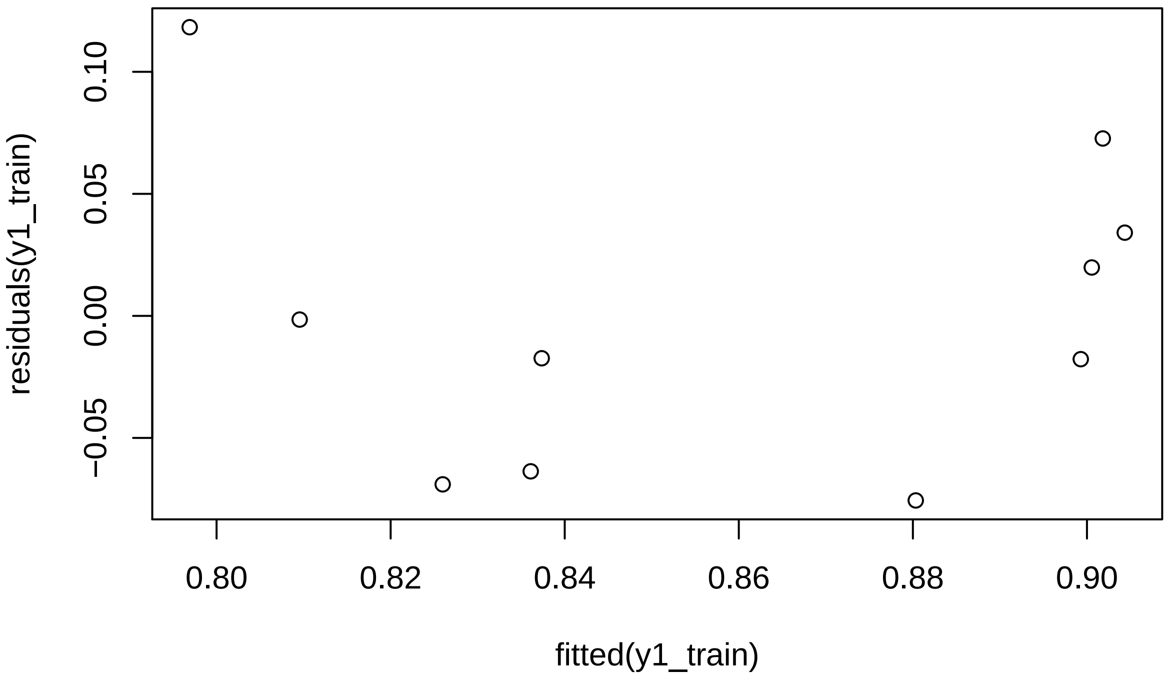
Problem 1 – Solution

(a)

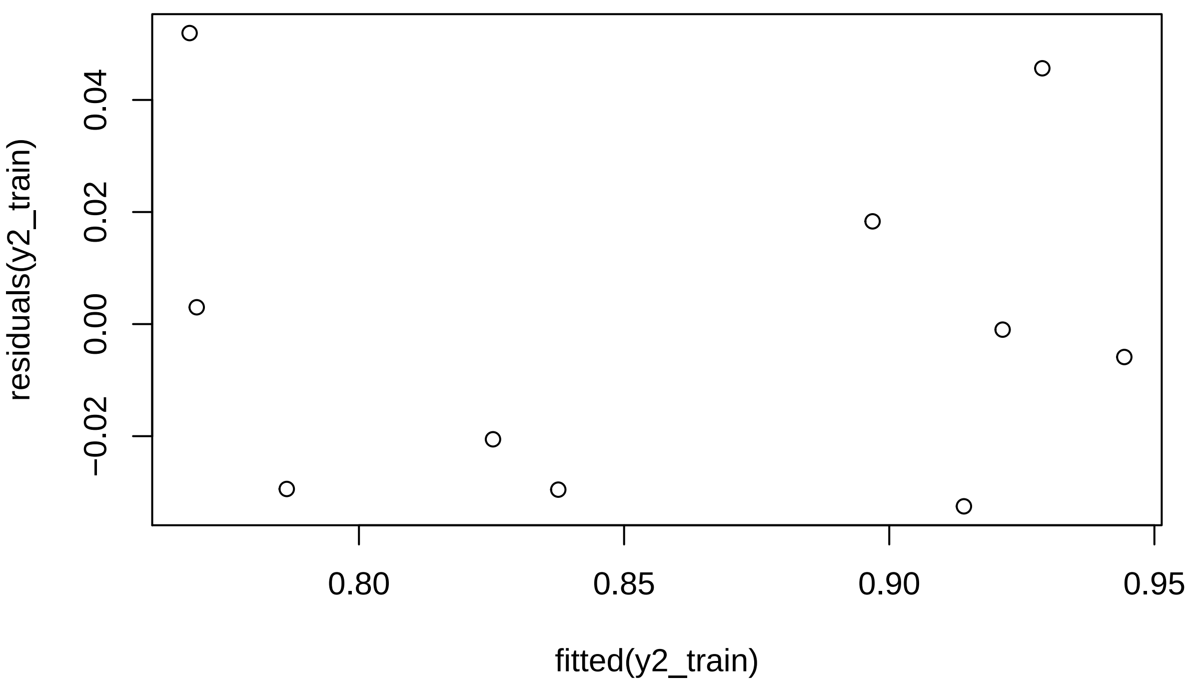
```
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------------------- tidyverse 1.2.1 --
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## -- Conflicts -------------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

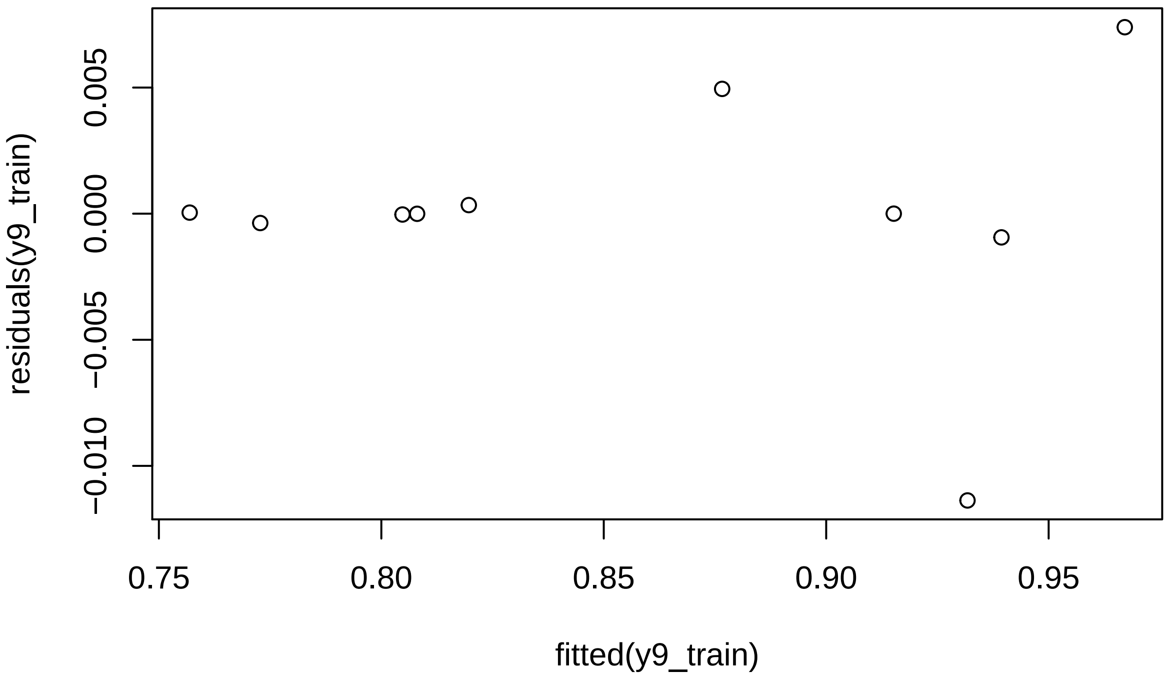
```
library(ggplot2)

# Dataset
HDS_ex3 <- read.csv("~/Dropbox/Important_Documents/Doctoral_Work/Courses/High Dimensional Stats/2019/We

train <- HDS_ex3[ c(1:10), ]
test <- HDS_ex3[ -c(1:10), ]

### Models using TRAIN data

# 1st order LM
y1_train <- lm(y ~ x, data = train)
sm_y1_train <- summary( y1_train )
plot(fitted(y1_train), residuals(y1_train)) # Fitted vs Residuals
```




```r
# 2nd order LM
y2_train <- lm(y ~ poly(x, 2, raw = T), data = train)
sm_y2_train <- summary( y2_train )
plot(fitted(y2_train), residuals(y2_train)) # Fitted vs Residuals
```




```r
# 9th order LM
y9_train <- lm(y ~ poly(x, 9, raw = T), data = train)
sm_y9_train <- summary( y9_train )
plot( fitted(y9_train), residuals(y9_train) ) # Fitted vs Residuals
```

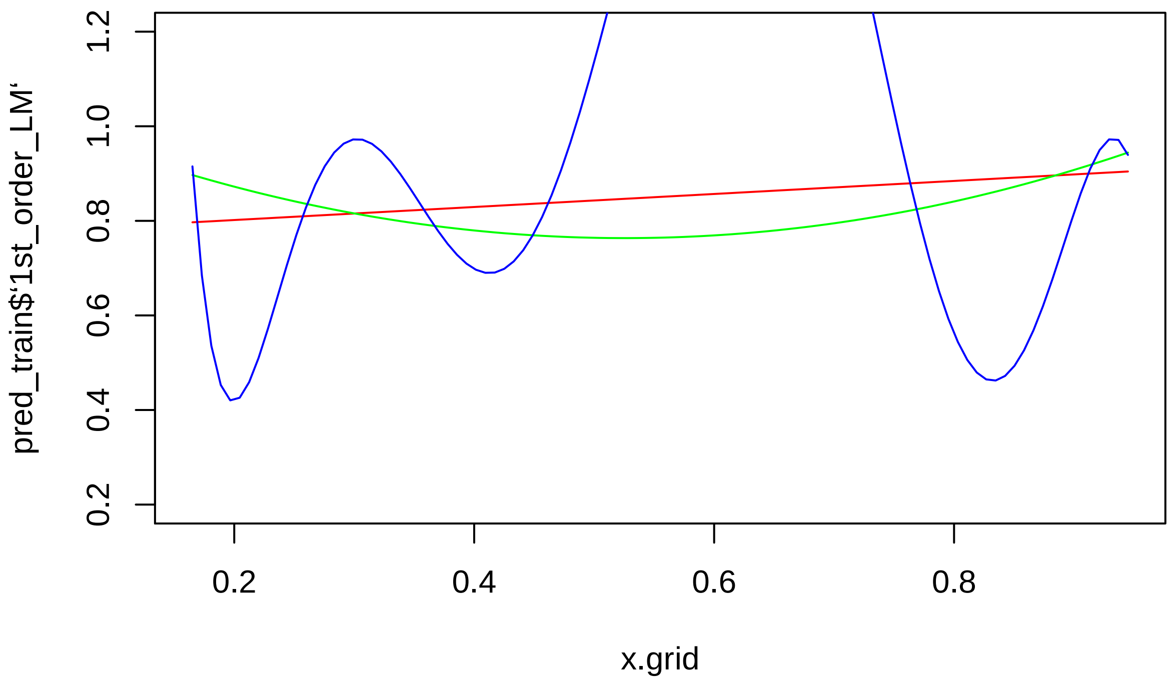


(b)

```
##### Plot training data for all models

x.grid = (seq(min(train$x),max(train$x),length=100))

# Adding predicted values to a dataset
pred_train <- as.data.frame(cbind( x.grid,
                    predict(y1_train, data.frame(x = x.grid)),
                    predict( y2_train, data.frame(x = x.grid)),
                  predict( y9_train, data.frame(x = x.grid))))
```

```
## Warning in predict.lm(y9_train, data.frame(x = x.grid)): prediction from a
## rank-deficient fit may be misleading
```

```
colnames( pred_train ) <- c( "x", "1st_order_LM", "2nd_order_LM", "9th_order_LM")

plot(x.grid, pred_train$`1st_order_LM`, type="l", col="red", ylim=c(0.2,1.2))
lines(x.grid, pred_train$`2nd_order_LM`, col="green", ylim=c(0.2,1.2))
lines(x.grid, pred_train$`9th_order_LM`, col="blue", ylim=c(0.2,1.2))
```



(c)

```r
### Models using test

# 1st order LM & mse
y1_test <- predict(y1_train, newdata = test)
y1_test_mse <- mean((test$y - y1_test)^ 2)

# 2nd order LM
y2_test <- predict(y2_train, newdata = test)
y2_test_mse <- mean((test$y - y2_test)^ 2)

# 9th order LM
y9_test <- predict(y9_train, newdata = test)
```

```
## Warning in predict.lm(y9_train, newdata = test): prediction from a rank-
## deficient fit may be misleading
```

```r
y9_test_mse <- mean((test$y - y9_test)^ 2)


# Obtain TEST MSE's
test_mse.df <- as.data.frame(cbind( "Test_MSE", y1_test_mse, y2_test_mse, y9_test_mse ))
colnames(test_mse.df) <- c("Data", "1st_order", "2nd order", "9th_order")

# Obtain TRAIN MSE's
y1_train_mse <- mean(y1_train$residuals^2)
y2_train_mse <- mean(y2_train$residuals^2)
y9_train_mse <- mean(y9_train$residuals^2)
train_mse.df <- as.data.frame(cbind( "Train_MSE", y1_train_mse, y2_train_mse, y9_train_mse ))
colnames(train_mse.df) <- c("Data", "1st_order", "2nd order", "9th_order")

# Get a df to showcase diff bet train and test MSE
diff_mse <- rbind(test_mse.df, train_mse.df)
```

```r
diff_mse
```

```
##        Data           1st_order            2nd order            9th_order
## 1  Test_MSE  0.0115978575797133  0.00305041209504852     270.842259445527
## 2 Train_MSE 0.00359849684471529 0.000838066404415258 2.09569420646987e-05
```

Training MSE's for the 1st, 2nd and 9th order models are lower than the test MSE's. However, when using test data, we see that MSE for 9th order is quite high whereas 2nd order model is the lowest.

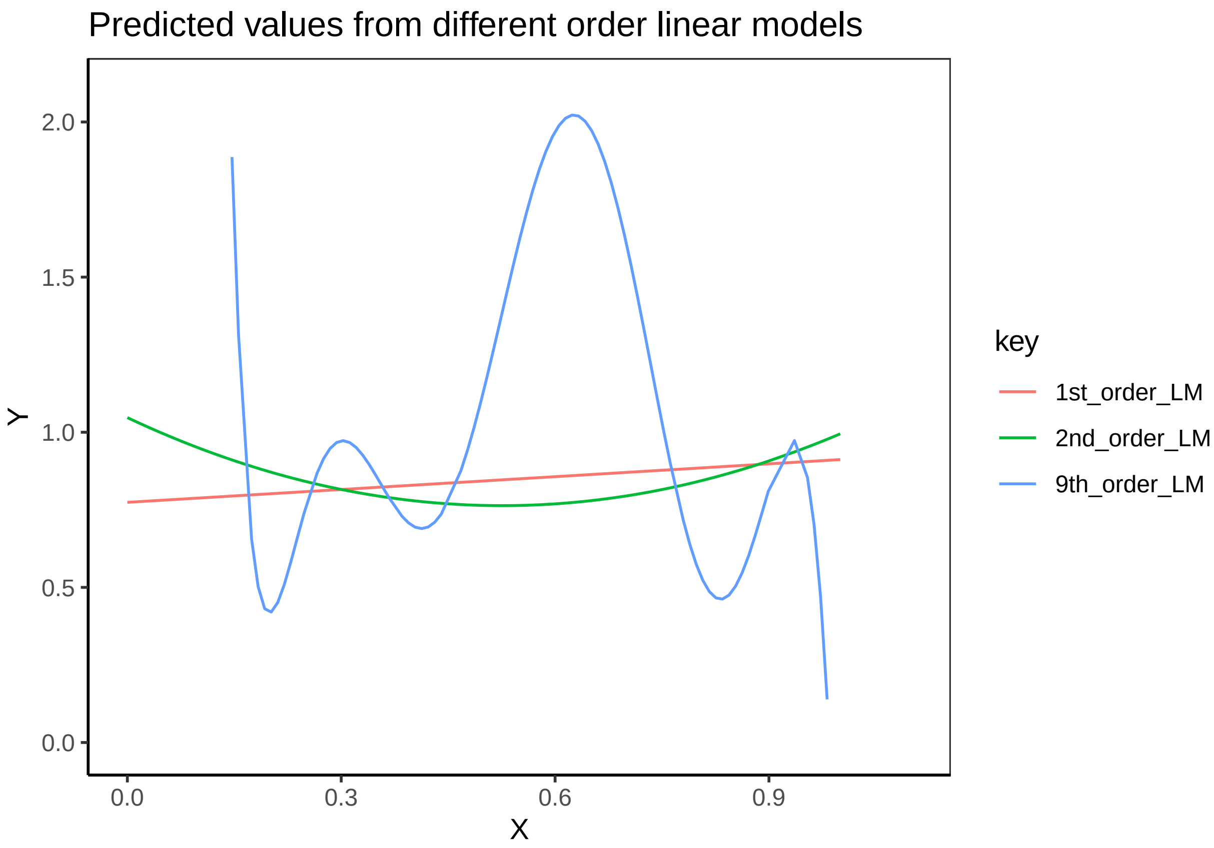
(d)

```r
### Plot predicted x-values against actual x from train data

## Prediction for each models
predict_y1 <- predict(y1_train, newdata = test)

predict_y2 <- predict(y2_train, newdata = test)

predict_y9 <- predict(y9_train, newdata = test)
```

```
## Warning in predict.lm(y9_train, newdata = test): prediction from a rank-
## deficient fit may be misleading
```

```r
# Adding all predicted values in a df
predicted_df <- as.data.frame(cbind( test$x, predict_y1, predict_y2, predict_y9 ))
colnames(predicted_df) <- c( "x", "1st_order_LM", "2nd_order_LM", "9th_order_LM")


# Convert from wide to long
predicted_test.df <- reshape2::melt(predicted_df,
                      id = "x")

# Plotting
predicted_df %>%
     gather(key,value, "1st_order_LM", "2nd_order_LM", "9th_order_LM") %>%
    ggplot(aes(x=x, y=value, colour=key)) +
    geom_line() +
  #theme with white background
     theme_bw() +
  #eliminates background, gridlines, and chart border
     theme(
       plot.background = element_blank()
      ,panel.grid.major = element_blank()
      ,panel.grid.minor = element_blank()
      ) +
  #draws x and y axis line
     theme(axis.line = element_line(color = 'black')) +
  scale_x_continuous( "X", limits = c(0, 1.1) ) +
  scale_y_continuous( "Y", limits = c(0, 2.1)) +
  labs(title = "Predicted values from different order linear models")
```

```
## Warning: Removed 18 rows containing missing values (geom_path).
```



The first degree model is reasonable, but we can see that the second degree model fits much better. The ninth degree model seem rather wild.

(e).

```
# Get truth "y" into predicted dataframe
pred_truth.df <- as.data.frame( cbind(test$y, predicted_df) )
colnames(pred_truth.df) <- c("truth", "x", "1st_order_LM", "2nd_order_LM", "9th_order_LM")

## Bias-Variance tradeoff
get_bias = function(estimate, truth) {
  mean(estimate) - truth
}

get_var = function(estimate) {
  mean((estimate - mean(estimate)) ^ 2)
}

get_mse = function(truth, estimate) {
  mean((estimate - truth) ^ 2)
}


# Bias from 3 models
bias_1st <- get_bias(pred_truth.df$`1st_order_LM`, pred_truth.df$truth)
bias_2nd <- get_bias(pred_truth.df$`2nd_order_LM`, pred_truth.df$truth)
bias_9th <- get_bias(pred_truth.df$`9th_order_LM`, pred_truth.df$truth)
```

```r
bias_df <- as.data.frame( cbind(bias_1st, bias_2nd, bias_9th) )
bias <- c(mean(bias_df$bias_1st), mean(bias_df$bias_2nd), mean(bias_df$bias_9th))

# Variance from 3 models
var_1st <- get_var(pred_truth.df$`1st_order_LM`)
var_2nd <- get_var(pred_truth.df$`2nd_order_LM`)
var_9th <- get_var(pred_truth.df$`9th_order_LM`)

var <- as.data.frame( rbind(var_1st, var_2nd, var_9th) )

# MSE from 3 models
mse_1st <- get_mse(pred_truth.df$`1st_order_LM`, pred_truth.df$truth)
mse_2nd <- get_mse(pred_truth.df$`2nd_order_LM`, pred_truth.df$truth)
mse_9th <- get_mse(pred_truth.df$`9th_order_LM`, pred_truth.df$truth)

mse <- as.data.frame( rbind(mse_1st, mse_2nd, mse_9th) )

# Summarize these above results in the following table
results <- data.frame (
  cbind(poly_degree = c(1, 2, 9),
        round(bias^2, 5),
        round(mse, 5),
        round(var, 5))
)
colnames(results) = c("Degree", "Mean Squared Error", "Bias Squared", "Variance")
rownames(results) = NULL
knitr::kable(results, booktabs = TRUE, escape = TRUE, align = "c")
```

Degree	Mean Squared Error	Bias Squared	Variance
1	0.00006	0.01160	0.00158
2	0.00028	0.00305	0.00650
9	25.35862	270.84226	246.74820

```r
# Bias-Variance Tradeoff
# Defined as, bias ^ 2 + variance == mse
```

We see that the 9th order model has the highest variance and 2nd order model has the lowest bias.

We see that the 2nd order model gets the best bias-variance tradeoff here

Problem 2 – Solution

(a)

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
data(Boston)

#Using all variables as predictors

lm.fit <- lm(medv ~ ., data = Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

```r
AIC(lm.fit)
```

```
## [1] 3027.609
```

```r
BIC(lm.fit)
```

```
## [1] 3091.007
```

```r
# Using all but age
lm.fit1 <- lm(medv ~ .-age , data = Boston )
summary (lm.fit1)
```

```
##
## Call:
## lm(formula = medv ~ . - age, data = Boston)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.6054  -2.7313  -0.5188   1.7601  26.2243
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.436927   5.080119   7.172 2.72e-12 ***
## crim         -0.108006   0.032832  -3.290 0.001075 **
## zn            0.046334   0.013613   3.404 0.000719 ***
## indus         0.020562   0.061433   0.335 0.737989
## chas          2.689026   0.859598   3.128 0.001863 **
## nox         -17.713540   3.679308  -4.814 1.97e-06 ***
## rm            3.814394   0.408480   9.338  < 2e-16 ***
## dis          -1.478612   0.190611  -7.757 5.03e-14 ***
## rad           0.305786   0.066089   4.627 4.75e-06 ***
## tax          -0.012329   0.003755  -3.283 0.001099 **
## ptratio      -0.952211   0.130294  -7.308 1.10e-12 ***
## black         0.009321   0.002678   3.481 0.000544 ***
## lstat        -0.523852   0.047625 -10.999  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.74 on 493 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7343
## F-statistic: 117.3 on 12 and 493 DF,  p-value: < 2.2e-16
```

```
AIC(lm.fit1)
```

```
## [1] 3025.611
```

```
BIC(lm.fit1)
```

```
## [1] 3084.783
```

```
# Using all but rm
lm.fit2 <- lm(medv ~ .-rm , data = Boston )
summary (lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ . - rm, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.8415  -2.9471  -0.5922   1.7921  23.1236
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  69.467163   3.884501  17.883  < 2e-16 ***
## crim         -0.115899   0.035484  -3.266 0.001166 **
## zn            0.065917   0.014645   4.501 8.45e-06 ***
## indus        -0.030978   0.066138  -0.468 0.639713
## chas          2.931577   0.930110   3.152 0.001721 **
## nox         -21.021201   4.107510  -5.118 4.44e-07 ***
## age           0.025592   0.013959   1.833 0.067351 .
```

```
## dis          -1.718414   0.213494  -8.049 6.29e-15 ***
## rad           0.401657   0.070757   5.677 2.35e-08 ***
## tax          -0.014881   0.004050  -3.674 0.000265 ***
## ptratio      -1.142943   0.139493  -8.194 2.20e-15 ***
## black         0.006747   0.002885   2.339 0.019752 *
## lstat        -0.772070   0.046280 -16.683  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.125 on 493 degrees of freedom
## Multiple R-squared:  0.6968, Adjusted R-squared:  0.6895
## F-statistic: 94.43 on 12 and 493 DF,  p-value: < 2.2e-16
```

```r
AIC(lm.fit2)
```

```
## [1] 3104.581
```

```r
BIC(lm.fit2)
```

```
## [1] 3163.753
```

```r
### Comparing AIC & BIC
AIC(lm.fit, lm.fit1, lm.fit2)
```

```
##         df      AIC
## lm.fit  15 3027.609
## lm.fit1 14 3025.611
## lm.fit2 14 3104.581
```

```r
BIC(lm.fit, lm.fit1, lm.fit2)
```

```
##         df      BIC
## lm.fit  15 3091.007
## lm.fit1 14 3084.783
## lm.fit2 14 3163.753
```

Both AIC and BIC are minimized in the second model with just the age removed.

(b)

```r
# Get log-lk values from AIC = 2k - 2loglk
# Here
loglk_aic_full <- (-AIC(lm.fit) + 2*15)/2
loglk_aic_reduced <- (-AIC(lm.fit1) + 2*14)/2

loglk_aic_full
```

```
## [1] -1498.804
```

```r
loglk_aic_reduced
```

```
## [1] -1498.806
```

```r
# The Likelihood ratio should be converted to -2*difference in log likelihoods: -2ln(Likelihood_reduced
# And the above can be written as:
# -2ln(Likelihood_reduced)- -2ln(Likelihood_full)
# LRT then should be approximately Chi-squared distributed with df equal to the number of fixed dimensi

# Likelihood-Ratio test (frequentist)
```

```r
Deviance <- (-2 * loglk_aic_reduced) - (-2 * loglk_aic_full)
Deviance
```

```
## [1] 0.002824145
```

```r
Chisq.crit <- qchisq(0.95,1)
Chisq.crit
```

```
## [1] 3.841459
```

```r
# LRT
Deviance >= Chisq.crit   # perform the LRT
```

```
## [1] FALSE
```

```r
# p-value
options(digits=10)
1-pchisq(Deviance,1)
```

```
## [1] 0.9576182206
```

The p-values for LET calculation based on AIC agrees with the one from full model for variable age (0.95 vs 0.9576)

(c)

```r
# Get log-lk values from BIC = ln(n)*k - 2ln(L)
loglk_bic_full <- (-BIC(lm.fit) + log(nrow(Boston))*15)/2
loglk_bic_reduced <- (-BIC(lm.fit2) + log(nrow(Boston))*14)/2

loglk_bic_full
```

```
## [1] -1498.804297
```

```r
loglk_bic_reduced
```

```
## [1] -1538.290563
```

```r
# The Likelihood ratio should be converted to -2*difference in log likelihoods: -2ln(Likelihood_reduced
# And the above can be written as:
# -2ln(Likelihood_reduced)- -2ln(Likelihood_full)
# LRT then should be approximately Chi-squared distributed with df equal to the number of fixed dimensi

# Likelihood-Ratio test (frequentist)
Deviance <- (-2*loglk_bic_reduced) - (-2*loglk_bic_full)
Deviance
```

```
## [1] 78.97253251
```

```r
Chisq.crit <- qchisq(0.95,1)
Chisq.crit
```

```
## [1] 3.841458821
```

```r
# LRT
Deviance >= Chisq.crit   # perform the LRT
```

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

```r
# p-value
options(digits = 22)
1-pchisq(Deviance,1)
```

```
## [1] 0
```

BIC based LRT gets a highly significant p-value as Deviance is very, very hugh compared to critical value of chi.square distribution at 1 degrees of freedom.

Problem 3 – Solution

(a)

```r
library(MASS)
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
data(Boston)

tr.ind = 1:350

# Train & Test dataset
train <- Boston[tr.ind, ]
test <- Boston[-tr.ind, ]

# Using all variables as predictors
train_lm.fit <- lm(medv ~ ., train)

# MSE for train and test data based models
mse_train <- mean(train_lm.fit$residuals^2)
mse_train
```

```
## [1] 8.987147271551950922223
```

```r
mse_test <- mean((test$medv - predict.lm(train_lm.fit, test)) ^ 2)
mse_test
```

```
## [1] 545.9435298634641640092
```

The test data model has a higher RMSE than the train dataset. This could be an indication that your model is overfitting

(b)

```r
# 10-fold cross-validation (CV) within training data
modelcv <- train(
  medv ~ .,
  data = train,
  method = "lm",
  trControl = trainControl(
    method = "cv", number = 10
  )
)

RMSE_Modelcv <- modelcv$results$RMSE
RMSE_Modelcv
```

```
## [1] 3.132821204347347343599
```

```r
pcv <- predict(modelcv, test)
errorcv <- (pcv- test$medv)
RMSE_NewDatacv <- sqrt(mean(errorcv^2))
RMSE_NewDatacv
```

```
## [1] 23.36543451047859676351
```

The cross validation didnt work that well at all. Here, we arent randomly dividing the dataset into test and train sets which increases overfitting error, violating the assumption of the data being iid.

(c)

```r
tr.ind <- read.csv("~/Dropbox/Important_Documents/Doctoral_Work/Courses/High Dimensional Stats/2019/Weel
data(Boston)

# Train & Test dataset

id <- which(rownames(Boston) %in% tr.ind$X1 )
train <- Boston[id, ]
test <- Boston[-id, ]

# Using all variables as predictors
train_lm.fit <- lm(medv ~ ., train)

# MSE for train and test data based models
mse_train <- mean(train_lm.fit$residuals^2)
mse_train
```

```
## [1] 21.81514031532957531567
```

```r
mse_test <- mean((test$medv - predict.lm(train_lm.fit, test)) ^ 2)
mse_test
```

```
## [1] 23.21951643873919124417
```

```r
# 10-fold cross-validation (CV) within training data
modelcv <- train(
  medv ~ .,
  data = train,
  method = "lm",
```

```
  trControl = trainControl(
    method = "cv", number = 10
  )
)

RMSE_Modelcv <- modelcv$results$RMSE
RMSE_Modelcv
```

```
## [1] 4.860076572546001116848
```

```
pcv <- predict(modelcv, test)
errorcv <- (pcv- test$medv)
RMSE_NewDatacv <- sqrt(mean(errorcv^2))
RMSE_NewDatacv
```

```
## [1] 4.818663345652940854791
```

We see that randomly splitting did massively improve th CV based rmse because it remóved underlying bias associated in separating our datasets non-randomly, thus restring the assumption of iid for inference.

Problem 4 – Solution

(a)

```
tr.ind <- fread("~/Dropbox/Important_Documents/Doctoral_Work/Courses/High Dimensional Stats/2019/Week 3/
```

```
## Error in fread("~/Dropbox/Important_Documents/Doctoral_Work/Courses/High Dimensional Stats/2019/Week
```

```
# Train & Test dataset
data(Boston)
train <- Boston[tr.ind$X1, ]
test <- Boston[-tr.ind$X1, ]

# Create null model
glm.null <- glm(medv ~ 1, family = gaussian,data = train)

# Using all variables as predictors
train_glm.fit <- glm(medv ~ ., family = gaussian, train)

# AIC-based forward selection
model.aic.forward <- step(glm.null, direction = "forward", k = 2, trace = FALSE,
                          scope = list(lower=glm.null, upper=train_glm.fit))
summary(model.aic.forward)
```

```
##
## Call:
## glm(formula = medv ~ lstat + rm + ptratio + chas + black + dis +
##     nox + zn + rad + tax, family = gaussian, data = train)
##
## Deviance Residuals:
##                     Min                        1Q                     Median
```

```
## -14.1293554337183167036   -2.6513550596041568497   -0.5952357583240264205
##                      3Q                      Max
##   1.6193941324851266472   25.9798227277326603257
##
## Coefficients:
##                          Estimate                Std. Error
## (Intercept)  34.9887004829515220194448   6.1356533039967660059 23
## lstat        -0.5569561299375765939 25   0.0585660239536474752 64
## rm            3.3757033411210084494 77   0.5214018498255817979 37
## ptratio      -0.7360907759021545482 27   0.1557924507449606754 68
## chas          2.6683664802955235906 27   0.9469719809794322484 82
## black         0.0088168161368044270 51   0.0033315043364909608 52
## dis          -1.4218797164143106570 58   0.2234711236037421611 12
## nox         -15.9943551335776241728 58   4.1264515371982248836 96
## zn            0.0538081001483871612 66   0.0157307839151380746 13
## rad           0.2504043871251128705 60   0.0751447901808746931 97
## tax          -0.0126044465983806835 97   0.0041974962859067897 76
##                         t value   Pr(>|t|)
## (Intercept)  5.702519999999999811 2.5769e-08 ***
## lstat       -9.509880000000000777 < 2.22e-16 ***
## rm           6.474280000000000257 3.3552e-10 ***
## ptratio     -4.724820000000000242 3.3834e-06 ***
## chas         2.817790000000000017 0.00512008 **
## black        2.646500000000000075 0.00851352 **
## dis         -6.362700000000000244 6.4471e-10 ***
## nox         -3.876059999999999839 0.00012754 ***
## zn           3.420560000000000045 0.00070148 ***
## rad          3.332289999999999974 0.00095669 ***
## tax         -3.002850000000000019 0.00287398 **
## ---
## Signif. codes:
##   0 '***' 0.0010000000000000000208 17 '**' 0.0100000000000000002081 7 '*'
##   0.0500000000000000002775 56 '.' 0.1000000000000000005551 1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 22.670496706416148668 95)
##
##     Null deviance: 26803.9758739255012188  on 348  degrees of freedom
## Residual deviance:  7662.6278867686587546  on 338  degrees of freedom
## AIC: 2092.4934795786962241
##
## Number of Fisher Scoring iterations: 2
```

```r
# BIC-based forward selection
model.bic.forward <- step(glm.null, direction = "forward", k = log(nrow(train)), trace = FALSE,
                          scope = list(lower=glm.null, upper=train_glm.fit))
summary(model.bic.forward)
```

```
##
## Call:
## glm(formula = medv ~ lstat + rm + ptratio + chas + black + dis +
##     nox + zn, family = gaussian, data = train)
##
## Deviance Residuals:
##                    Min                     1Q                  Median
## -14.714173723614315747   -2.734711628867813715   -0.566886100198601639
```

```
##                    3Q                    Max
##   1.618281616094495945   26.934057843555464729
##
## Coefficients:
##                              Estimate                Std. Error
## (Intercept)  29.712092788472745041872   5.828956898004529207924
## lstat        -0.547990065293597061746   0.058956915194980259731
## rm            3.690250165390836833978   0.518140603855466008731
## ptratio      -0.680050580896372047768   0.137831852650530523041
## chas          3.034969065099948792863   0.950917551576890507370
## black         0.008133546199970299179   0.003294112655736606589
## dis          -1.384211247676725653477   0.224615385729099975576
## nox         -16.914536273439679803232   3.801752095608633474910
## zn            0.049483385598185117282   0.015397577395027426533
##                           t value   Pr(>|t|)
## (Intercept)  5.097330000000000361 5.7294e-07 ***
## lstat       -9.294750000000000512 < 2.22e-16 ***
## rm           7.122099999999999653 6.3755e-12 ***
## ptratio     -4.933910000000000018 1.2641e-06 ***
## chas         3.191619999999999902  0.0015467 **
## black        2.469120000000000203  0.0140357 *
## dis         -6.162580000000000169 2.0247e-09 ***
## nox         -4.449139999999999873 1.1698e-05 ***
## zn           3.213709999999999845  0.0014358 **
## ---
## Signif. codes:
##   0 '***' 0.0010000000000000000020817 '**' 0.0100000000000000000020817 '*'
##   0.0500000000000000000277556 '.' 0.1000000000000000055511 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 23.294121820743743711155)
##
##     Null deviance: 26803.975873925501219  on 348  degrees of freedom
## Residual deviance:  7920.001419052872734  on 340  degrees of freedom
## AIC: 2100.0231810578620752
##
## Number of Fisher Scoring iterations: 2
```

(b)

```r
# AIC-based backward selection
model.aic.backward <- step(train_glm.fit,  data = train, direction = "backward", k = 2, trace = FALSE)
summary(model.aic.backward)
```

```
##
## Call:
## glm(formula = medv ~ zn + chas + nox + rm + dis + rad + tax +
##     ptratio + black + lstat, family = gaussian, data = train)
##
## Deviance Residuals:
##                    Min                     1Q                 Median
## -14.129355433718302492 7   -2.651355059604142638 8   -0.595235758324026420 5
##                     3Q                    Max
##   1.619394132485130199 9   25.979822727732670983 9
##
## Coefficients:
```

```
##                            Estimate               Std. Error
## (Intercept)  34.98870048295149359773 9   6.13565330399676689410 2
## zn            0.05380810014838746657 7   0.01573078391513814400 2
## chas          2.66836648029551515293 2   0.94697198097942947292 4
## nox         -15.99435513357763127828 5   4.12645153719822488369 6
## rm            3.37570334112101155810 1   0.52140184982558213100 4
## dis          -1.42187971641431198932 5   0.22347112360374221662 3
## rad           0.25040438712511292607 1   0.07514479018087469319 7
## tax          -0.01260444659838069921 0   0.00419749628590679064 3
## ptratio      -0.73609077590215388209 3   0.15579245074496075873 5
## black         0.00881681613680442011 2   0.00333150433649096041 8
## lstat        -0.55695612993757614983 5   0.05856602395364746138 6
##                        t value   Pr(>|t|)
## (Intercept)  5.702519999999999811 2.5769e-08 ***
## zn           3.420560000000000045 0.00070148 ***
## chas         2.817790000000000017 0.00512008 **
## nox         -3.876059999999999839 0.00012754 ***
## rm           6.474280000000000257 3.3552e-10 ***
## dis         -6.362700000000000244 6.4471e-10 ***
## rad          3.332289999999999974 0.00095669 ***
## tax         -3.002850000000000019 0.00287398 **
## ptratio     -4.724820000000000242 3.3834e-06 ***
## black        2.646500000000000075 0.00851352 **
## lstat       -9.509880000000000777 < 2.22e-16 ***
## ---
## Signif. codes:
##   0 '***' 0.001000000000000000020817 '**' 0.01000000000000000020817 '*'
##   0.05000000000000000277556 '.' 0.1000000000000000055511 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 22.670496706416152221 66)
##
##     Null deviance: 26803.9758739255012188  on 348  degrees of freedom
## Residual deviance:  7662.6278867686596641  on 338  degrees of freedom
## AIC: 2092.4934795786962241
##
## Number of Fisher Scoring iterations: 2
```

```r
# BIC-based backward selection
model.bic.backward <- step(train_glm.fit, direction = "backward", k = log(nrow(train)), trace = FALSE)
summary(model.bic.backward)
```

```
##
## Call:
## glm(formula = medv ~ zn + chas + nox + rm + dis + rad + tax +
##     ptratio + black + lstat, family = gaussian, data = train)
##
## Deviance Residuals:
##                     Min                       1Q                  Median
## -14.1293554337183024927   -2.6513550596041426388   -0.5952357583240264205
##                      3Q                      Max
##   1.6193941324851301999   25.9798227277326709839
##
## Coefficients:
##                            Estimate               Std. Error
## (Intercept)  34.98870048295149359773 9   6.13565330399676689410 2
```

```
## zn            0.053808100148387466577   0.015730783915138144002
## chas          2.668366480295515152932   0.946971980979429472924
## nox         -15.994355133577631278285   4.126451537198224883696
## rm            3.375703341121011558101   0.521401849825582131004
## dis          -1.421879716414311989325   0.223471123603742216623
## rad           0.250404387125112926071   0.075144790180874693197
## tax          -0.012604446598380699210   0.004197496285906790643
## ptratio      -0.736090775902153882093   0.155792450744960758735
## black         0.008816816136804420112   0.003331504336490960418
## lstat        -0.556956129937576149835   0.058566023953647461386
##                           t value   Pr(>|t|)
## (Intercept)  5.702519999999999811 2.5769e-08 ***
## zn           3.420560000000000045 0.00070148 ***
## chas         2.817790000000000017 0.00512008 **
## nox         -3.876059999999999839 0.00012754 ***
## rm           6.474280000000000257 3.3552e-10 ***
## dis         -6.362700000000000244 6.4471e-10 ***
## rad          3.332289999999999974 0.00095669 ***
## tax         -3.002850000000000019 0.00287398 **
## ptratio     -4.724820000000000242 3.3834e-06 ***
## black        2.646500000000000075 0.00851352 **
## lstat       -9.509880000000000777 < 2.22e-16 ***
## ---
## Signif. codes:
##   0 '***' 0.00100000000000000020817 '**' 0.01000000000000000020817 '*'
##   0.05000000000000000277556 '.' 0.10000000000000000055511 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 22.67049670641615222166)
##
##     Null deviance: 26803.9758739255012188  on 348  degrees of freedom
## Residual deviance:  7662.6278867686596641  on 338  degrees of freedom
## AIC: 2092.4934795786962241
##
## Number of Fisher Scoring iterations: 2
```

(c)

```r
# Use AIC for interaction terms in the model
# Limiting to 2nd order interactions only

# AIC-based forward selection
model.aic.interaction.forward <- step(train_glm.fit, direction = "forward", k = 2, trace = FALSE,
                                      scope = . ~ .^2)
summary(model.aic.interaction.forward)
```

```
##
## Call:
## glm(formula = medv ~ crim + zn + indus + chas + nox + rm + age +
##     dis + rad + tax + ptratio + black + lstat + rm:tax + tax:lstat +
##     rm:lstat + dis:rad + black:lstat + crim:rm + age:tax + nox:rad +
##     crim:chas + chas:nox + crim:lstat + indus:chas + rm:ptratio +
##     chas:rm + chas:ptratio + rm:black + age:lstat + age:black +
##     rm:age + indus:tax + zn:tax + dis:black + nox:ptratio + dis:tax +
##     indus:age + nox:age + zn:lstat + tax:ptratio + indus:rm +
##     indus:nox + rm:rad + age:rad + crim:nox, family = gaussian,
```

```
##     data = train)
##
## Deviance Residuals:
##                    Min                      1Q                  Median
## -8.36191453513221816252  -1.48387745823181305127  -0.04506097996837965525
##                     3Q                     Max
##  1.30647991624691073298  17.14516279669210518932
##
## Coefficients:
##                              Estimate               Std. Error
## (Intercept)  -2.764656683014658256e+02  3.099285221797312317e+01
## crim         -1.396650522310709741e+00  7.613746037412615353e-01
## zn           -1.239538881052500707e-01  4.001624795462360717e-02
## indus        -2.786835390545370661e+00  7.824395453034850290e-01
## chas          6.521841141111552531e+01  1.163164957396033117e+01
## nox           1.057109031485380797e+02  2.482298188561901853e+01
## rm            3.528913205053627422e+01  3.466455514739979815e+00
## age           8.331826311187912060e-01  1.646506315920409635e-01
## dis           6.672617531454471340e+00  1.766067656525021734e+00
## rad          -1.106218880946331140e+00  8.545624824131926589e-01
## tax           2.318534389338405000e-01  5.500016350212019040e-02
## ptratio       3.851245211223220721e+00  1.408843525399480434e+00
## black         1.873872970979600927e-01  3.318628917406287598e-02
## lstat         3.641217384150063907e+00  3.747108910375596125e-01
## rm:tax       -4.142222436446049705e-02  6.764517426931998707e-03
## tax:lstat    -3.051656490596716710e-03  3.134052359627326960e-04
## rm:lstat     -2.819695265633577197e-01  4.929451119297676570e-02
## dis:rad      -6.214195306218942699e-02  4.716157435352843347e-02
## black:lstat  -1.903661763694611741e-03  4.165976196156629188e-04
## crim:rm       2.658209589216991020e-01  8.146103565486899345e-02
## age:tax      -1.631825816626490330e-04  1.819072503604225299e-04
## nox:rad      -6.056017903176419415e-01  5.896145924780528125e-01
## crim:chas     2.605125693916086949e+00  3.901834065217794079e-01
## chas:nox     -5.226641800773642643e+01  8.709257219453771626e+00
## crim:lstat    3.181759086955792543e-02  7.822536722718263433e-03
## indus:chas    3.068300653814132750e-01  2.122592807877582144e-01
## rm:ptratio   -5.009374045687918775e-01  1.514748950061289012e-01
## chas:rm      -3.543549279100450811e+00  8.535329022315761849e-01
## chas:ptratio -1.104822856559980471e+00  3.880213489937475724e-01
## rm:black     -9.648690559464680877e-03  4.121532383189240555e-03
## age:lstat    -7.066623636186998222e-03  1.730148505995588891e-03
## age:black    -6.703246636447737121e-04  2.050302879744211448e-04
## rm:age       -5.283396976230293751e-02  1.614119347472250943e-02
## indus:tax    -8.025642136804199249e-05  4.249186028467852273e-04
## zn:tax        6.403571229472971270e-04  1.503325969657462143e-04
## dis:black    -1.271543984436804102e-02  4.002171361476621753e-03
## nox:ptratio  -5.042549853015561467e+00  1.250802882088529744e+00
## dis:tax      -7.605411695025130620e-03  2.037190744946314191e-03
## indus:age     4.381518232110723915e-03  2.475524763757829330e-03
## nox:age      -4.947682246157291686e-01  1.589312366743856342e-01
## zn:lstat     -9.480520546068571530e-03  3.017668549246235125e-03
## tax:ptratio   4.516282670086391691e-03  1.663897154486750985e-03
## indus:rm      2.851481738018857848e-01  1.021290744836650249e-01
## indus:nox     1.586215240493120016e+00  6.369878087127706090e-01
```

```
## rm:rad        2.330094292503021858e-01  1.020265122917953377e-01
## age:rad       6.885221624572052460e-03  3.626216172694303984e-03
## crim:nox     -1.534444859162441110e+00  9.962766452015711094e-01
##                            t value   Pr(>|t|)
## (Intercept)  -8.920299999999992298 < 2.22e-16 ***
## crim         -1.834379999999998995 0.06758126 .
## zn           -3.097589999999998435 0.00213441 **
## indus        -3.561729999999998406 0.00042791 ***
## chas          5.606980000000000750 4.6484e-08 ***
## nox           4.258589999999998755 2.7505e-05 ***
## rm           10.180180000000000068 < 2.22e-16 ***
## age           5.060310000000003078 7.2843e-07 ***
## dis           3.778230000000001994 0.00019026 ***
## rad          -1.294489999999999185 0.19648663
## tax           4.215499999999995808 3.2962e-05 ***
## ptratio       2.733620000000001609 0.00663377 **
## black         5.646530000000002711 3.7777e-08 ***
## lstat         9.717399999999995936 < 2.22e-16 ***
## rm:tax       -6.123459999999996811 2.8447e-09 ***
## tax:lstat    -9.737090000000002450 < 2.22e-16 ***
## rm:lstat     -5.720100000000004064 2.5610e-08 ***
## dis:rad      -1.317639999999999224 0.18862305
## black:lstat  -4.569549999999995566 7.1327e-06 ***
## crim:rm       3.263170000000001261 0.00122811 **
## age:tax      -0.897059999999999685 0.37039892
## nox:rad      -1.027109999999999675 0.30518843
## crim:chas     6.676669999999996606 1.1718e-10 ***
## chas:nox     -6.001249999999997513 5.6011e-09 ***
## crim:lstat    4.067429999999998789 6.0716e-05 ***
## indus:chas    1.445540000000000471 0.14934212
## rm:ptratio   -3.307069999999999541 0.00105664 **
## chas:rm      -4.151629999999999315 4.2992e-05 ***
## chas:ptratio -2.847319999999998511 0.00471107 **
## rm:black     -2.341040000000000098 0.01988021 *
## age:lstat    -4.084399999999995865 5.6660e-05 ***
## age:black    -3.269390000000000183 0.00120231 **
## rm:age       -3.273239999999999274 0.00118663 **
## indus:tax    -0.188870000000000101 0.85031780
## zn:tax        4.259599999999998309 2.7387e-05 ***
## dis:black    -3.177140000000000752 0.00164135 **
## nox:ptratio  -4.031450000000004221 7.0244e-05 ***
## dis:tax      -3.733280000000001539 0.00022584 ***
## indus:age     1.769940000000000688 0.07774692 .
## nox:age      -3.113100000000002004 0.00202871 **
## zn:lstat     -3.141669999999999626 0.00184648 **
## tax:ptratio   2.714280000000000260 0.00702381 **
## indus:rm      2.792040000000000773 0.00557171 **
## indus:nox     2.490180000000000601 0.01330526 *
## rm:rad        2.283809999999998957 0.02307661 *
## age:rad       1.898730000000000288 0.05855240 .
## crim:nox     -1.540180000000001045 0.12456372
## ---
## Signif. codes:
##   0 '***' 0.00100000000000000020817 '**' 0.01000000000000000020817 '*'
```

```
##   0.05000000000000000277556 '.' 0.10000000000000000555511 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 7.37692757943339838533)
##
##     Null deviance: 26803.9758739255012188  on 348  degrees of freedom
## Residual deviance:  2227.8321289888863248  on 302  degrees of freedom
## AIC: 1733.3646993297024892
##
## Number of Fisher Scoring iterations: 2
```

```r
# BIC-based forward selection
model.bic.interaction.forward <- step(train_glm.fit, direction = "forward", k = log(nrow(train)),
                                      trace = FALSE, scope = . ~ .^2)
summary(model.bic.interaction.forward)
```

```
##
## Call:
## glm(formula = medv ~ crim + zn + indus + chas + nox + rm + age +
##     dis + rad + tax + ptratio + black + lstat + rm:tax + tax:lstat +
##     rm:lstat + dis:rad + black:lstat + crim:rm + age:tax + nox:rad +
##     crim:chas + chas:nox + crim:lstat + indus:chas + rm:ptratio +
##     chas:rm + chas:ptratio + rm:black + age:lstat, family = gaussian,
##     data = train)
##
## Deviance Residuals:
##                    Min                      1Q                  Median
## -8.9584734029085844043  -1.3686066551287137116  -0.1707027924997888135
##                     3Q                     Max
##  1.4076263808047535520  17.5539910539074597295
##
## Coefficients:
##                              Estimate               Std. Error
## (Intercept)  -1.753821193183746914e+02  2.065012501202262030e+01
## crim         -2.429663684336619944e+00  5.953249788167160883e-01
## zn            1.843841944997085674e-02  1.078834124712413182e-02
## indus         3.082323732418955700e-02  5.548439251683835033e-02
## chas          6.303123099516884054e+01  1.212904445722971758e+01
## nox           1.477821044833492081e+01  4.842649189992489056e+00
## rm            2.918228846536947074e+01  2.956865531014346793e+00
## age          -1.286989727608114920e-01  3.061158750981189669e-02
## dis           5.404854154522686048e-03  2.364223895119365870e-01
## rad           1.890042659269945080e+00  2.598848099452226168e-01
## tax           1.644121279503556465e-01  2.183141815100938729e-02
## ptratio       1.910981672091115469e+00  8.958528934426160939e-01
## black         1.142901124420816761e-01  2.894285353357905002e-02
## lstat         3.891124276766534162e+00  3.738099515319451838e-01
## rm:tax       -2.608117249246676497e-02  3.106080638344956533e-03
## tax:lstat    -3.067823841447844032e-03  3.023810150241406270e-04
## rm:lstat     -3.632070368712051467e-01  4.751754114739078355e-02
## dis:rad      -1.433034431184866120e-01  2.974325557454528951e-02
## black:lstat  -2.123211832503334780e-03  3.646542692426392603e-04
## crim:rm       2.922928223217445276e-01  8.200388670272751312e-02
## age:tax       3.710606329469607278e-04  7.923460104306478610e-05
## nox:rad      -1.967127720292835447e+00  3.391415692035700813e-01
## crim:chas     2.338226300384794065e+00  4.139882412025911451e-01
```

```
## chas:nox     -5.117428497527086506e+01  8.890704544122554509e+00
## crim:lstat    2.471445788282258804e-02  7.399940244240762847e-03
## indus:chas    3.710035945667873869e-01  2.105913090689877643e-01
## rm:ptratio   -3.560178181375316941e-01  1.401328244820214397e-01
## chas:rm      -3.209659549865007566e+00  8.492687644633943878e-01
## chas:ptratio -1.116015264430078924e+00  4.019387349786481267e-01
## rm:black     -1.146917169537272144e-02  4.119426367947163006e-03
## age:lstat    -3.382424696172292751e-03  1.306143278808277524e-03
##                              t value   Pr(>|t|)
## (Intercept)   -8.49302999999999919112 7.8431e-16 ***
## crim          -4.08124000000000020094 5.6708e-05 ***
## zn             1.70910999999999990706 0.08840694 .
## indus          0.55552999999999996827 0.57892289
## chas           5.19672000000000000597 3.6336e-07 ***
## nox            3.05168000000000017025 0.00246742 **
## rm             9.86932999999999971408 < 2.22e-16 ***
## age           -4.20425999999999966406 3.4088e-05 ***
## dis            0.02285999999999999838 0.98177548
## rad            7.27261999999999986244 2.7623e-12 ***
## tax            7.53099000000000007304 5.2527e-13 ***
## ptratio        2.13314000000000003610 0.03367837 *
## black          3.94881999999999999673 9.6749e-05 ***
## lstat         10.40936999999999912347 < 2.22e-16 ***
## rm:tax        -8.39681000000000032912 1.5369e-15 ***
## tax:lstat    -10.14555999999999968963 < 2.22e-16 ***
## rm:lstat      -7.64364000000000043400 2.5166e-13 ***
## dis:rad       -4.81801000000000012591 2.2485e-06 ***
## black:lstat   -5.82253000000000042746 1.4172e-08 ***
## crim:rm        3.56437999999999988177 0.00042067 ***
## age:tax        4.68306000000000022254 4.1925e-06 ***
## nox:rad       -5.80030999999999963279 1.5979e-08 ***
## crim:chas      5.64804999999999957083 3.6023e-08 ***
## chas:nox      -5.75593000000000021288 2.0286e-08 ***
## crim:lstat     3.33982000000000001094 0.00093804 ***
## indus:chas     1.76171999999999995268 0.07907693 .
## rm:ptratio    -2.54057000000000021700 0.01154223 *
## chas:rm       -3.77931999999999979067 0.00018775 ***
## chas:ptratio  -2.77658000000000004803 0.00581909 **
## rm:black      -2.78417000000000003368 0.00568791 **
## age:lstat     -2.58963000000000009848 0.01004990 *
## ---
## Signif. codes:
##   0 '***' 0.00100000000000000020817 '**' 0.01000000000000000020817 '*'
##   0.05000000000000000277556 '.' 0.10000000000000000055511 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 8.660684943529627588532)
##
##     Null deviance: 26803.9758739255012188  on 348  degrees of freedom
## Residual deviance:  2754.0978120424215376  on 318  degrees of freedom
## AIC: 1775.3739683878152391
##
## Number of Fisher Scoring iterations: 2
```

(d)

```r
##################### Getting predictors + MSE from previous models

##### AIC_forward model
AIC_forward <- attr(model.aic.forward$terms , "term.labels")
AIC_forward
```

```
##  [1] "lstat"   "rm"      "ptratio" "chas"    "black"   "dis"     "nox"
##  [8] "zn"      "rad"     "tax"
```

```r
# Train
model.aic.forward.train <- step(glm.null, data = train, direction = "forward", k = 2, trace = FALSE,
                           scope = list(lower=glm.null, upper=train_glm.fit))
AIC_forward_mse_train <- mean(model.aic.forward.train$residuals^2)

# Test
model.aic.forward.test <- predict(model.aic.forward.train, newdata = test)
AIC_forward_mse_test <- mean((test$medv - model.aic.forward.test)^ 2)




##### BIC_forward model
BIC_forward <- attr(model.bic.forward$terms , "term.labels")
BIC_forward
```

```
## [1] "lstat"   "rm"      "ptratio" "chas"    "black"   "dis"     "nox"
## [8] "zn"
```

```r
# Train
model.bic.forward.train <- step(glm.null, data = train, direction = "forward", k = log(nrow(train)), tra
                           scope = list(lower=glm.null, upper=train_glm.fit))
BIC_forward_mse_train <- mean(model.bic.forward.train$residuals^2)

# Test
model.bic.forward.test <- predict(model.bic.forward.train, newdata = test)
BIC_forward_mse_test <- mean((test$medv - model.bic.forward.test)^ 2)




##### AIC_backward model
AIC_backward <- attr(model.aic.backward$terms , "term.labels")
AIC_backward
```

```
##  [1] "zn"      "chas"    "nox"     "rm"      "dis"     "rad"     "tax"
##  [8] "ptratio" "black"   "lstat"
```

```r
# Train
model.aic.backward.train <- step(train_glm.fit,  data = train, direction = "backward", k = 2, trace = F
AIC_backward_mse_train <- mean(model.aic.backward.train$residuals^2)

# Test
model.aic.backward.test <- predict(model.aic.backward.train, newdata = test)
AIC_backward_mse_test <- mean((test$medv - model.aic.backward.test)^ 2)
```

```r
##### BIC_backward model
BIC_backward <- attr(model.bic.backward$terms , "term.labels")
BIC_backward
```

```
##  [1] "zn"      "chas"    "nox"     "rm"      "dis"     "rad"     "tax"
##  [8] "ptratio" "black"   "lstat"
```

```r
# Train
model.bic.backward.train <- step(train_glm.fit, data = train, direction = "backward", k = log(nrow(trai
BIC_backward_mse_train <- mean(model.bic.backward.train$residuals^2)

# Test
model.bic.backward.test <- predict(model.bic.backward.train, newdata = test)
BIC_backward_mse_test <- mean((test$medv - model.bic.backward.test)^ 2)



##### AIC-based forward selection + interaction
AIC_forward_interaction <- attr(model.aic.interaction.forward$terms , "term.labels")
AIC_forward_interaction
```

```
##  [1] "crim"          "zn"            "indus"         "chas"
##  [5] "nox"           "rm"            "age"           "dis"
##  [9] "rad"           "tax"           "ptratio"       "black"
## [13] "lstat"         "rm:tax"        "tax:lstat"     "rm:lstat"
## [17] "dis:rad"       "black:lstat"   "crim:rm"       "age:tax"
## [21] "nox:rad"       "crim:chas"     "chas:nox"      "crim:lstat"
## [25] "indus:chas"    "rm:ptratio"    "chas:rm"       "chas:ptratio"
## [29] "rm:black"      "age:lstat"     "age:black"     "rm:age"
## [33] "indus:tax"     "zn:tax"        "dis:black"     "nox:ptratio"
## [37] "dis:tax"       "indus:age"     "nox:age"       "zn:lstat"
## [41] "tax:ptratio"   "indus:rm"      "indus:nox"     "rm:rad"
## [45] "age:rad"       "crim:nox"
```

```r
# Train
model.aic.interaction.forward.train <- step(train_glm.fit,  data = train, direction = "forward", k = 2,
                                      scope = . ~ .^2)
AIC_forward_interaction_mse_train <- mean(model.aic.interaction.forward.train$residuals^2)

# Test
model.aic.interaction.forward.test <- predict(model.aic.interaction.forward.train, newdata = test)
AIC_forward_interaction_mse_test <- mean((test$medv - model.aic.interaction.forward.test)^ 2)



##### BIC-based forward selection + interaction
BIC_forward_interaction <- attr(model.bic.interaction.forward$terms , "term.labels")
BIC_forward_interaction
```

```
##  [1] "crim"          "zn"            "indus"         "chas"
##  [5] "nox"           "rm"            "age"           "dis"
##  [9] "rad"           "tax"           "ptratio"       "black"
## [13] "lstat"         "rm:tax"        "tax:lstat"     "rm:lstat"
```

```
## [17] "dis:rad"      "black:lstat"  "crim:rm"      "age:tax"
## [21] "nox:rad"      "crim:chas"    "chas:nox"     "crim:lstat"
## [25] "indus:chas"   "rm:ptratio"   "chas:rm"      "chas:ptratio"
## [29] "rm:black"     "age:lstat"
```

```
# Train
model.bic.interaction.forward.train <- step(train_glm.fit, direction = "forward", k = log(nrow(train)),
                                             trace = FALSE, scope = . ~ .^2)
BIC_forward_interaction_mse_train <- mean(model.bic.interaction.forward.train$residuals^2)

# Test
model.bic.interaction.forward.test <- predict(model.bic.interaction.forward.train, newdata = test)
BIC_forward_interaction_mse_test <- mean((test$medv - model.bic.interaction.forward.test)^ 2)




### Get results from above together

results <- data.frame(rbind(
                  c(AIC_forward_mse_train, AIC_forward_mse_test),
                  c(BIC_forward_mse_train, BIC_forward_mse_test),
                  c(AIC_backward_mse_train, AIC_backward_mse_test),
                  c(BIC_backward_mse_train, BIC_backward_mse_test),
                  c(AIC_forward_interaction_mse_train, AIC_forward_interaction_mse_test),
                  c(BIC_forward_interaction_mse_train, BIC_forward_interaction_mse_test)))
colnames(results) <- c("Training_MSE", "Test_MSE")
results <- results %>% mutate(Model = c("AIC_forward", "BIC_forward", "AIC_backward",
                                         "BIC_backward", "AIC_forward_interaction", "BIC_forward_interac
results
```

```
##               Training_MSE                Test_MSE                   Model
## 1 21.955953830282691541242 24.294631938157376538 36             AIC_forward
## 2 22.693413808174419443731 24.974373100081503906 72             BIC_forward
## 3 21.955953830282691541242 24.294631938157397854 64            AIC_backward
## 4 21.955953830282691541242 24.294631938157397854 64            BIC_backward
## 5  6.383473148965290278056 13.853919909447377278 87 AIC_forward_interaction
## 6  7.891397742241895052473 15.121609327752937090 85 BIC_forward_interaction
```

The results suggest that stepwise AIC-forward regression model with 2nd order interaction terms has the lowest RMSE indictaing it to be the better model. The variance not explained is given by MSE/Variance(Y) and hence 1-Variance Unexplained will give us the explained variance

```
## Variance explained
(1 - AIC_forward_interaction_mse_test/var(test$medv)) * 100
```

```
## [1] 86.40015334529469726021
```

Problem 5 – Solution

(a)

```
# Dataset
library(data.table)
```

```
##
## Attaching package: 'data.table'
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
HDS_ex3 <- fread("~/Dropbox/Important_Documents/Doctoral_Work/Courses/High Dimensional Stats/2019/Week 3

# Null Model
glm.null <- glm(z ~ 1, data = HDS_ex3)

# Log Reg
glm.fit <- glm(z ~ x + I(x^2), family = "binomial", data = HDS_ex3)

# Forward stepwise selection with AIC
model.aic.forward <- step(glm.null, data = HDS_ex3, direction = "forward", k = 2, trace = TRUE,
                          scope = list(lower=glm.null, upper=glm.fit))
```

```
## Start:  AIC=162.3600000000000136424
## z ~ 1
##
##          Df               Deviance                   AIC
## <none>      27.1727272727271952 162.35715468192961453
## + x       1 27.098879629500945754 164.05779941194552407
## + I(x^2)  1 27.117723729974805735 164.13426495966163543
```

```
summary(model.aic.forward)
```

```
##
## Call:
## glm(formula = z ~ 1, data = HDS_ex3)
##
## Deviance Residuals:
##                    Min                        1Q                   Median
## -0.5545454545454547857  -0.5545454545454547857   0.4454545454545452143
##                     3Q                       Max
##  0.4454545454545452143   0.4454545454545452143
##
## Coefficients:
##                           Estimate              Std. Error
## (Intercept) 0.55454545454545478567 0.04760548821460328789
##                          t value   Pr(>|t|)
## (Intercept) 11.64877000000000073 < 2.22e-16 ***
## ---
## Signif. codes:
##   0 '***' 0.00100000000000000020817 '**' 0.01000000000000000020817 '*'
##   0.05000000000000000277556 '.' 0.1000000000000000055511 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.249291075896580477389)
##
##     Null deviance: 27.1727272727271952  on 109  degrees of freedom
```

```
## Residual deviance: 27.172727272727271952  on 109  degrees of freedom
## AIC: 162.3571546819295861
##
## Number of Fisher Scoring iterations: 2
```

```r
### THe intercept model is the one chosen by the stepwise method
new.model <- glm(z ~ 1 + x + I(x^2), family = "binomial", data = HDS_ex3)
AIC(new.model)
```

```
## [1] 139.725896997331659577
```

We may not necessarily always get the highest variance explained and lower AIC because we only compare a subset of possible models and might miss the one with the highest adjR2/lowest AIC which would include all the variables, like the above case. Given a set of predictors, there is no guarantee that stepwise will find the "best" combination of predictors (defined as, say, the highest adjusted R^2); it can get stuck in local optima´and never reach the so-called global optima which might be the desired solution

(b)

```r
# Logistic reg - 1
log.reg1 <- glm(z ~ 1 + x + I(x^2), family = "binomial", data = HDS_ex3)
summary(log.reg1)
```

```
##
## Call:
## glm(formula = z ~ 1 + x + I(x^2), family = "binomial", data = HDS_ex3)
##
## Deviance Residuals:
##                    Min                      1Q                 Median
## -2.1300539131789357761  -1.0151568747577928153   0.5240391151288268379
##                     3Q                     Max
##  1.0014015912025853172   1.4891145212415386467
##
## Coefficients:
##                             Estimate              Std. Error
## (Intercept)    2.4880356645983900954   0.7475189900256100639
## x            -12.2658935497350292110   3.3066288184383378912
## I(x^2)        11.7626242308449153740   3.1476694202173804982
##                            z value   Pr(>|z|)
## (Intercept)   3.328390000000000182 0.00087349 ***
## x            -3.709490000000000176 0.00020768 ***
## I(x^2)        3.736930000000000085 0.00018628 ***
## ---
## Signif. codes:
##   0 '***' 0.001000000000000000020817 '**' 0.01000000000000000020817 '*'
##   0.05000000000000000277556 '.' 0.1000000000000000055511 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 151.18067983034620738  on 109  degrees of freedom
## Residual deviance: 133.72589699733165958  on 107  degrees of freedom
## AIC: 139.72589699733165958
##
## Number of Fisher Scoring iterations: 4
```

```r
# Logistic reg - 2
log.reg2 <- glm(z ~ 1+ x + I(x^2) + y, family = "binomial", data = HDS_ex3)
summary(log.reg2)
```

```
##
## Call:
## glm(formula = z ~ 1 + x + I(x^2) + y, family = "binomial", data = HDS_ex3)
##
## Deviance Residuals:
##                    Min                      1Q                  Median
## -2.0623470538137560482  -0.9985004187637595008   0.4939246844396286140
##                     3Q                     Max
##  0.9802582814127576150   1.8278084823511471235
##
## Coefficients:
##                          Estimate            Std. Error
## (Intercept) -6.267048163668172300  4.395718860100193304
## x           -3.692198554016479228  5.245681650757693326
## I(x^2)       3.539967893354933004  5.026152989658693393
## y            8.648104430866997205  4.325755915372162086
##                            z value Pr(>|z|)
## (Intercept) -1.425720000000000983 0.153950
## x           -0.703849999999999757 0.481523
## I(x^2)       0.704309999999999916 0.481240
## y            1.999209999999999316 0.045585 *
## ---
## Signif. codes:
##   0 '***' 0.001000000000000000020817 '**' 0.01000000000000000020817 '*'
##   0.05000000000000000277556 '.' 0.1000000000000000055511 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 151.18067983034620738  on 109  degrees of freedom
## Residual deviance: 129.48590151209737087  on 106  degrees of freedom
## AIC: 137.48590151209737087
##
## Number of Fisher Scoring iterations: 4
```

The above results suggest that there is a problem of multicollinearity which exists when related variables appear as predictors violating the assumption of independence in regression.

(c)

```r
# Backward Selection
model.aic.backward <- step(log.reg2,  data = HDS_ex3, direction = "backward", k = 2, trace = FALSE)
summary(model.aic.backward)
```

```
##
## Call:
## glm(formula = z ~ y, family = "binomial", data = HDS_ex3)
##
## Deviance Residuals:
##                    Min                      1Q                  Median
## -2.0331310910528812563  -1.0154754701591115484   0.5088006797286009908
```

```
##                     3Q                    Max
##  0.9902892118952659750   1.9030261630043476817
##
## Coefficients:
##                          Estimate            Std. Error
## (Intercept) -8.961319509947299977  2.261552006770386036
## y           11.065177139438622689  2.736863694132543046
##                           z value   Pr(>|z|)
## (Intercept) -3.962460000000000093 7.4180e-05 ***
## y            4.043009999999999771 5.2769e-05 ***
## ---
## Signif. codes:
##   0 '***' 0.001000000000000000020817 '**' 0.01000000000000000020817 '*'
##   0.05000000000000000277556 '.' 0.1000000000000000055511 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 151.18067983034620738  on 109  degrees of freedom
## Residual deviance: 129.98763286639479020  on 108  degrees of freedom
## AIC: 133.9876328663947902
##
## Number of Fisher Scoring iterations: 3
```

Based on this the choice is the model with just "y" (formula = z ~ y) instead of "x" and its quadratic term.