Predictions using Statistical Models

Nelson Dsouza

December 14, 2015

There are 4 researches and corresponding predictions which I have worked on based on the topics listed below:

1. Prediction of having marriage affair using Fair's Affairs dataset.
2. Prediction of murder rates across the USA using internal R dataset for States.
3. Prediction of getting Breast Cancer using Wisconsin Breast Cancer Dataset.
4. Prediction of car milage using internal R dataset for Auto.

I have created and compared multiple statistical models in order to arrive at the best statistical model for prediction. I have also done data exploration and answered many research questions along the way.

Note -

1. These predictions are ONLY valid in the context of the datasets I have analyzed here and SHOULD NOT be extended the entire domain.
2. I have done the research and predictions listed here as part of course work for Data Science (Autumn 2015) at the University of Washington.

To start off, let us load the required libraries:

```
# Loading standard libraries
library(dplyr)
library(ggplot2)
library(grid)
```

Research 1 - Marriage Affairs

We will use the infidelity data, known as the Fair's Affairs dataset. The 'Affairs' dataset is available as part of the AER package in R. This data comes from a survey conducted by Psychology Today in 1969, see Greene (2003) and Fair (1978) for more information. The dataset contains various self-reported characteristics of 601 participants, including how often the respondent engaged in extramarital sexual intercourse during the past year, as well as their gender, age, year married, whether they had children, their religiousness (on a 5-point scale, from 1=anti to 5=very), education, occupation (Hillinghead 7-point classification with reverse numbering), and a numeric self-rating of their marriage (from 1=very unhappy to 5=very happy).

```
# Loading the library and dataset
library(AER)
data("Affairs")

# Creating local dataframe for Affairs data
affairs <- tbl_df(Affairs)

# Viewing structure of our data
str(affairs)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    601 obs. of  9 variables:
##  $ affairs      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gender       : Factor w/ 2 levels "female","male": 2 1 1 2 2 1 1 2 1 2 ...
##  $ age          : num  37 27 32 57 22 32 22 57 32 22 ...
##  $ yearsmarried : num  10 4 15 15 0.75 1.5 0.75 15 15 1.5 ...
##  $ children     : Factor w/ 2 levels "no","yes": 1 1 2 2 1 1 1 2 2 1 ...
##  $ religiousness: int  3 4 1 5 2 2 2 2 4 4 ...
##  $ education    : num  18 14 12 18 17 17 12 14 16 14 ...
##  $ occupation   : int  7 6 1 6 6 5 1 4 1 4 ...
##  $ rating       : int  4 4 4 5 3 5 3 4 2 5 ...
```

Research Research Question 1(a)

Describe the participants in the study.

Solution 1(a)

Let us look at the number of participants of each gender

```
# Checking the gender ratio
affairs %>%
  mutate(total = n()) %>%  # Creating temporary variable to hold total
  group_by(gender, total) %>%
  summarise(count = n()) %>%
  mutate(percent = paste(round(count/total*100,2),'%')) %>%
  select(gender,count,percent)
```

```
## Source: local data frame [2 x 3]
## Groups: gender [2]
##
##   gender count percent
##   (fctr) (int)   (chr)
## 1 female   315 52.41 %
## 2   male   286 47.59 %
```

We can see that there are a total of 601 participants in this study with 315 or approximately 52.41% females and 286 or approximately 47.59% males.

Next let us see the summary of age

```
# summary function gives a statistical summary of the variable
summary(affairs$age)
```

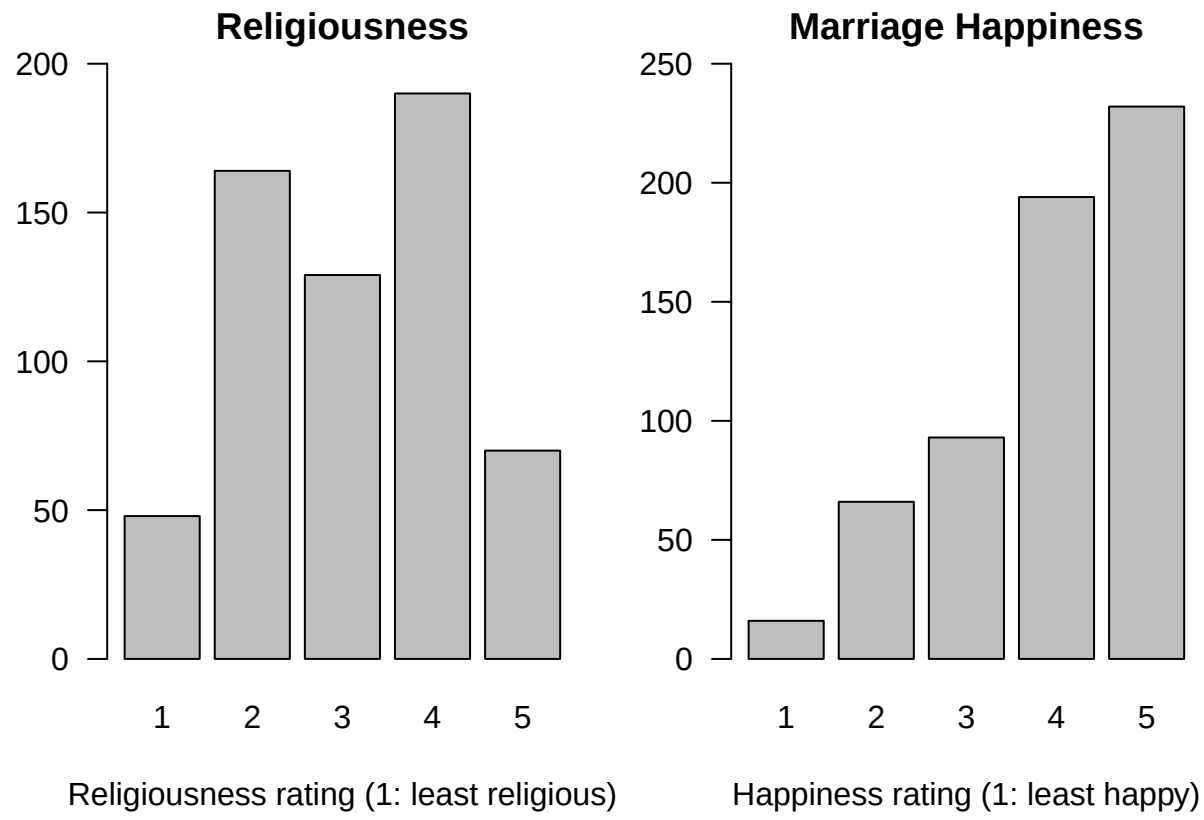
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   17.50   27.00   32.00   32.49   37.00   57.00
```

We have to remember that minimum age 17.5 is a code for all ages below 20. We see that the mean and median age are around 32 (age range 30 - 34) and the maximum age is 57 (age of 55 and over)

Reference - http://www.doviak.net/courses/statistics/Fair-JPE-1978_data-descriptions.pdf

Let us now visually explore the abstract self reported attributes such as their religiousness and happiness.

```r
# Dividing canvas into 2 parts to display graphs
par(mfrow=c(1,2), mar=c(5,3,2,1), las=1)

# using barplot to visualize religiousness in marriage
barplot(table(affairs$religiousness), main = "Religiousness",
          xlab = "Religiousness rating (1: least religious)",
          ylab = "Participants", ylim = c(0,200))

# using barplot to visualize happiness in marriage
barplot(table(affairs$rating), main = "Marriage Happiness",
          xlab = "Happiness rating (1: least happy)",
          ylab = "Participants", ylim = c(0,250))
```



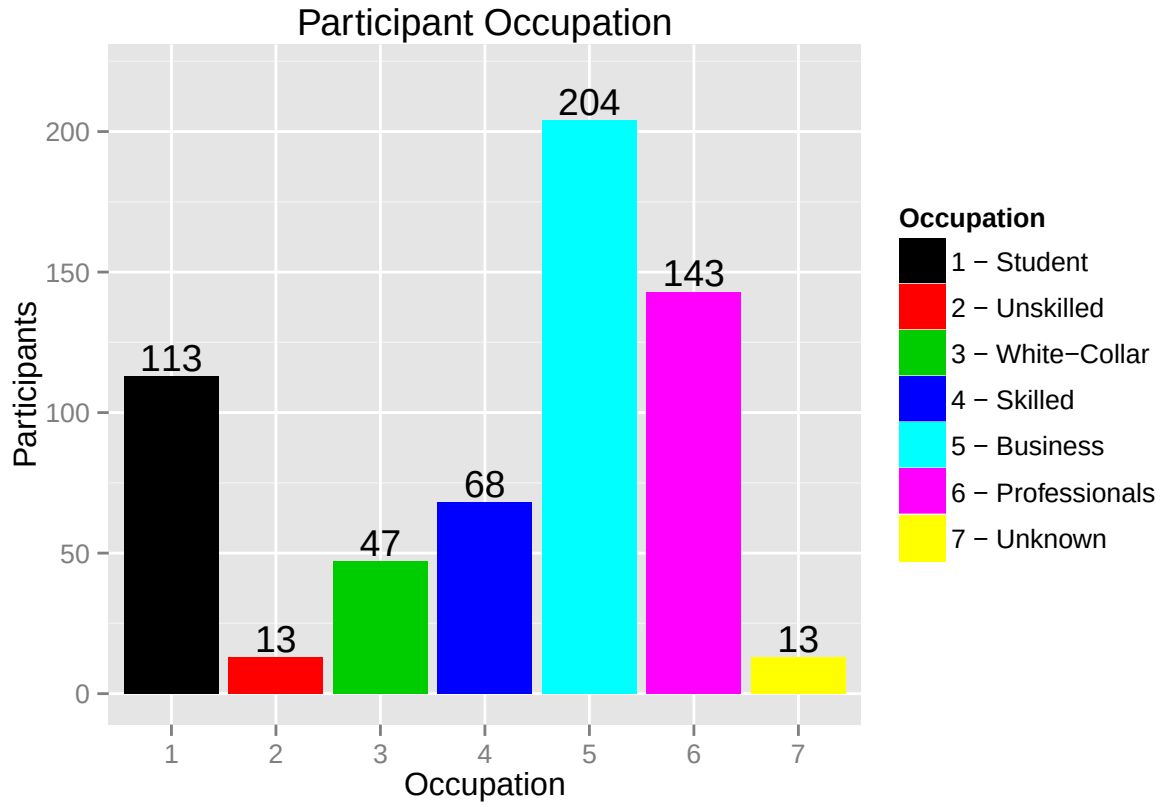

```r
# Setting canvas to default value
grid.newpage()
```

We observe for the participants

1. There are not many who are extremely religious or completely non-religious
2. Most participants are very happy with their marriage

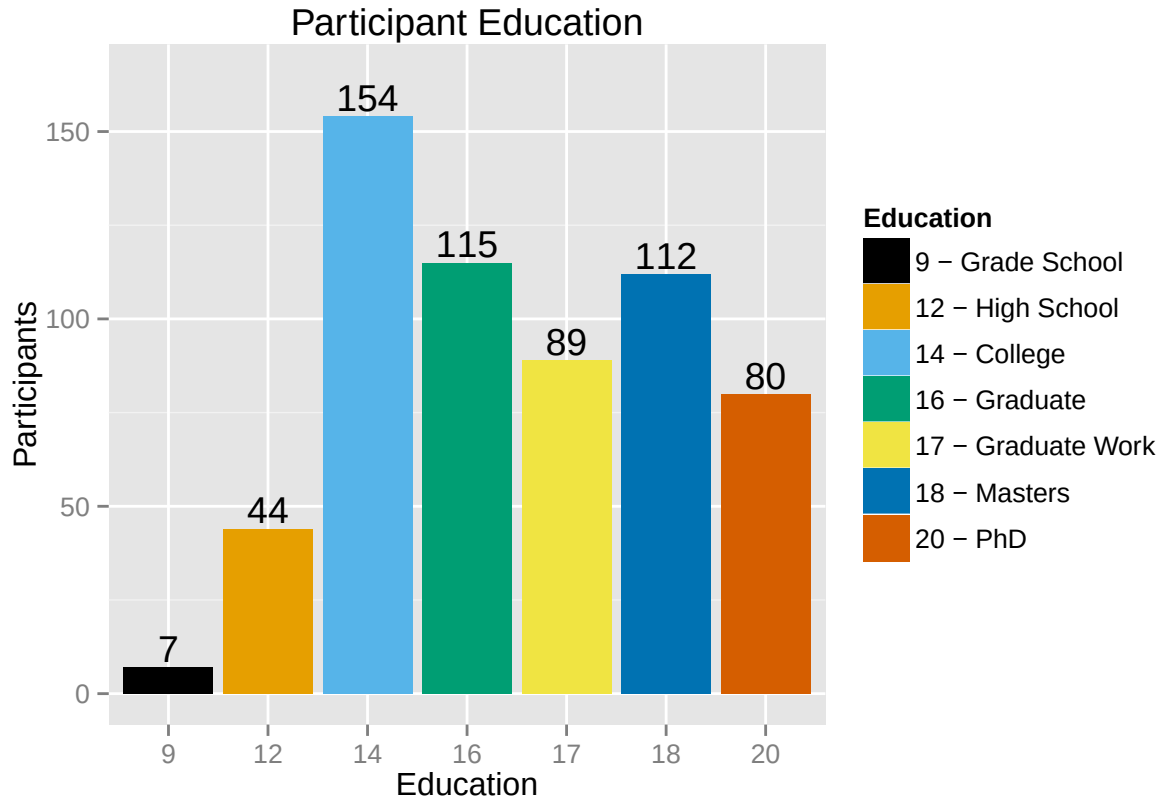
Let us also visually explore non abstract attributes such as education and occupation of the participants.

```r
# For the legend values of occupation and education, the below source was referred
# Source - http://www.doviak.net/courses/statistics/Fair-JPE-1978_data-descriptions.pdf

occupation.legend <-  c("1 - Student", "2 - Unskilled","3 - White-Collar", "4 - Skilled",
                        "5 - Business", "6 - Professionals", "7 - Unknown")

education.legend <- c("9 - Grade School", "12 - High School", "14 - College",
                      "16 - Graduate", "17 - Graduate Work", "18 - Masters", "20 - PhD")

# Function to create layout for ggplot
# plot.layout <- function(x, y) viewport(layout.pos.row = x, layout.pos.col = y)

# Using barplots to visualize occupation of participants
plot.occupation <-
  ggplot(affairs, aes(x = as.factor(occupation), fill = as.factor(occupation))) +
  geom_bar(stat = "bin") +
  scale_fill_identity(name = "Occupation", guide = "legend",labels = occupation.legend) +
  labs(title = "Participant Occupation", x = "Occupation",
       y = "Participants") +
  stat_bin(aes(ymax=max(..count..), label=..count..), geom="text", vjust = -0.2) +
  ylim(0,220)

# For next graph, lets set color manually
colors <- c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00",
            "#CC79A7")

# Using barplots to visualize education of participants
plot.education <-
  ggplot(affairs, aes(x = as.factor(education), fill = as.factor(education))) +
  geom_bar(stat = "bin") +
  scale_fill_manual(values = colors, name = "Education", guide = "legend",
                    labels = education.legend) +
  labs(title = "Participant Education", x = "Education",
       y = "Participants") +
  stat_bin(aes(ymax=max(..count..), label=..count..), geom="text", vjust = -0.2) +
  ylim(0,165)

# Plotting graph for occupation
plot.occupation
```



We observe that for the participants the most common occupation is Business

```
# Plotting graph for education
plot.education
```



We observe that for the participants, the most common education level is a college degree

---

Research Question 1(b)

Explore the characteristics of participants who engage in extramarital sexual intercourse

Solution 1(b)

Let's create a new variable to hold TRUE value if the participant had at least 1 affair

```
# creating new variable hadaffair to hold the binary value
affairs <- affairs %>%
  mutate(hadaffair = affairs > 0)

# changing datatype of new variable to factor for ease of use
affairs$hadaffair <- as.factor(affairs$hadaffair)
```

Let us check the summary of this new information

```
# Checking affairs ratio
affairs %>%
  mutate(total = n()) %>%  # Creating temporary variable to hold total
  group_by(hadaffair, total, gender) %>%
```

```
  summarise(count = n()) %>%
  mutate(percent = paste(round(count/total*100,2),'%')) %>%
  select(hadaffair,gender, count,percent)
```

```
## Source: local data frame [4 x 5]
## Groups: hadaffair, total [2]
##
##   total hadaffair gender count percent
##   (int)    (fctr) (fctr) (int)   (chr)
## 1   601     FALSE female   243 40.43 %
## 2   601     FALSE   male   208 34.61 %
## 3   601      TRUE female    72 11.98 %
## 4   601      TRUE   male    78 12.98 %
```

We see that approximately 3/4 th participants did not have an extramarital affair.

Research Question 1(c)

Explore the relationship between having an affair and other personal characteristics.

Solution 1(c)

Since we are trying to predict binomial data, a logistic regression model would be appropriate for our analysis.

```
# fitting a logistic regression model for having an affair
glm.affairs <- glm(hadaffair~ age + yearsmarried + religiousness + education + occupation
                   + rating + children + gender, family = "binomial", data=affairs)

# generating summary for our logistic model
summary(glm.affairs)
```

```
##
## Call:
## glm(formula = hadaffair ~ age + yearsmarried + religiousness +
##     education + occupation + rating + children + gender, family = "binomial",
##     data = affairs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5713  -0.7499  -0.5690  -0.2539   2.5191
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.37726    0.88776   1.551 0.120807
## age           -0.04426    0.01825  -2.425 0.015301 *
## yearsmarried   0.09477    0.03221   2.942 0.003262 **
## religiousness -0.32472    0.08975  -3.618 0.000297 ***
## education      0.02105    0.05051   0.417 0.676851
```

```
## occupation     0.03092    0.07178   0.431 0.666630
## rating        -0.46845    0.09091  -5.153 2.56e-07 ***
## childrenyes    0.39767    0.29151   1.364 0.172508
## gendermale     0.28029    0.23909   1.172 0.241083
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 675.38  on 600  degrees of freedom
## Residual deviance: 609.51  on 592  degrees of freedom
## AIC: 627.51
##
## Number of Fisher Scoring iterations: 4
```

From the summary we can see that the p values for education, occupation, children and gender are not significant (not less than 0.05) and hence these independent variables are not predictive of whether a person will have an affair.

On the flip side, we see that age, yearsmarried, religiousness and rating have a significant p value (greater than 0.05). Hence we can conclude that these 4 independent variables are predictive of having an affair. Furthermore, since the p value of religiousness and rating is highly significant, we can infer that these 2 covariates are highly predictive compared to age or yearmarried.

It is also worth noting that the AIC value 627.51.

Research Question 1(d)

Obtain a best fit model

Solution 1(d)

```
# Loading the bestglm library
library(bestglm)
```

```
## Loading required package: leaps
```

```
# Creating new columns to hold values for success of affair and failure of affair
# since this is required by the stepwise glm function - bestglm
affairs.matrix <- affairs %>%
  mutate(successaffair = hadaffair, failureaffair = as.factor(!as.logical(hadaffair)))

# Converting values to binary since it is required by bestglm function
levels(affairs.matrix$successaffair) <- c("0","1")
levels(affairs.matrix$failureaffair) <- c("0","1")

affairs.matrix$successaffair <- as.numeric(as.character(affairs.matrix$successaffair))
affairs.matrix$failureaffair <- as.numeric(as.character(affairs.matrix$failureaffair))
```

```
# Taking only the required variables
affairs.matrix <- affairs.matrix[,c("gender", "age","yearsmarried", "children",
                                    "religiousness", "education", "occupation",
                                    "rating", "successaffair", "failureaffair")]

bestglm.affairs <- bestglm(affairs.matrix, family = binomial, IC="AIC",
                           method = "exhaustive")
```

```
## Morgan-Tatar search since family is non-gaussian.
```

```
# Let us see the predictors suggested as bestglm for the best fit model
bestglm.affairs$BestModels[1,1:9]
```

```
##   gender  age yearsmarried children religiousness education occupation
## 1   TRUE TRUE         TRUE     TRUE          TRUE     FALSE      FALSE
##   rating Criterion
## 1   TRUE  614.1529
```

The variables included in the best fit model are: gender, age, yearsmarried, children, religiousness and rating.

If we compare this with our earlier model from part (c) we realize that the significant predictors suggested by the full model (age, yearsmarried, religiousness and rating) are subset of the predictors suggested by bestglm() for the best fit model. In bestglm, we are getting additional predictors, children and gender.

Research Question 1(e)

Interpret the model parameters using the model from part (d).

Solution 1(e)

```
# Evaluating the best glm model
bestglm.affairs$BestModel
```

```
##
## Call:  glm(formula = y ~ ., family = family, data = Xi, weights = weights)
##
## Coefficients:
##   (Intercept)     gendermale            age   yearsmarried    childrenyes
##       1.75357        0.37366       -0.04285        0.09518        0.37307
## religiousness         rating
##      -0.32900       -0.46015
##
## Degrees of Freedom: 600 Total (i.e. Null);  594 Residual
## Null Deviance:       675.4
## Residual Deviance: 610.2     AIC: 624.2
```

In the best fit model, we see that the predictors are age, yearsmarried, religiousness and rating. From the coefficients, we can gauge that:

1. As age increases by one unit, the chance of having an affair decreases by approximately 4.2%.

2. As years married increases by one unit, the chance of having an affair increases by approximately 10% .
3. As religiousness increases by a factor of 1, the chance of having an affair decreases by approximately 33%.
4. As happiness rating increases by a factor of 1, the chance of having an affair decreases by approximately 46%.
5. If the person has children, then the chance of having an affair increases by approximately 37%
6. If the person is a male, then the chance of having an affair increase by approximately 37%.

Further, we also see that the AIC is 624.2. This is less than the AIC we had obtained in our earlier full-fit model (627.51). This corroborates the fact that our bestglm model is indeed the best.

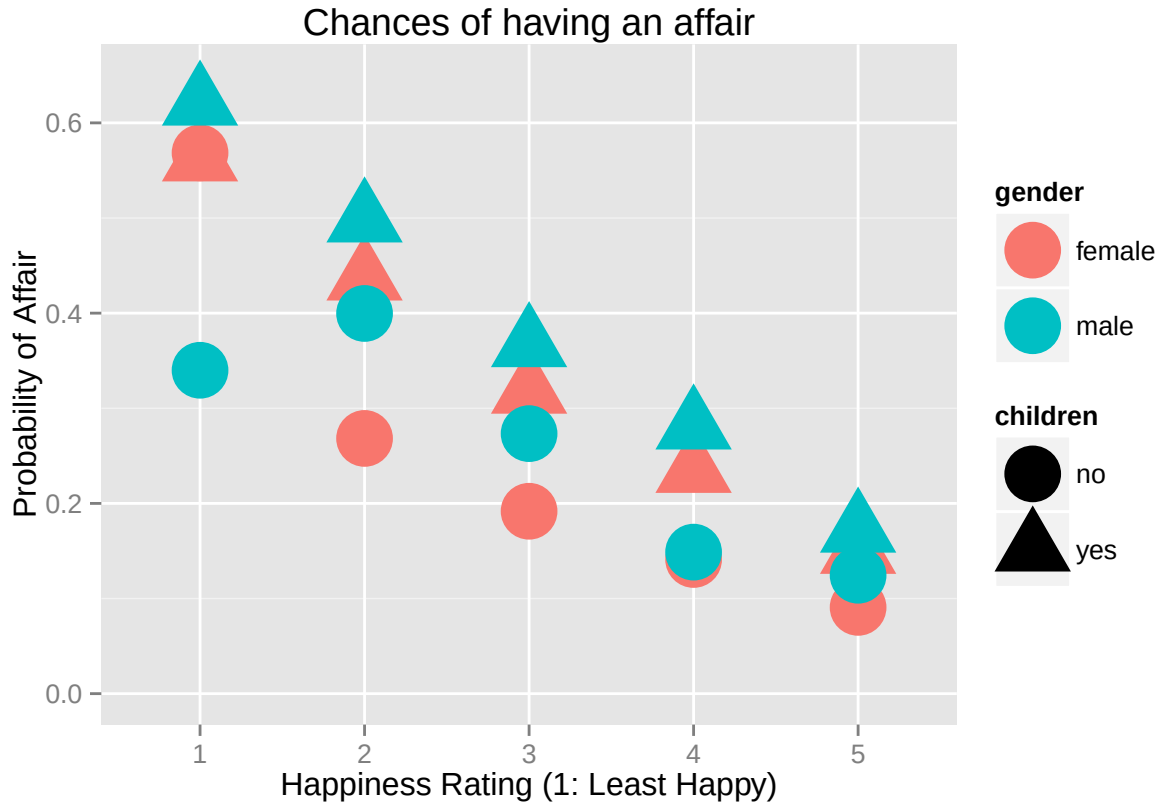
Research Question 1(f)

(f) Predict probabilities of having an affair

Solution 1(f)

Creating an artificial test dataset with mean of numeric values summarised on the factors (gender, children) and the rating.

```
# Creating test dataset with all numeric values set to mean based on factors and rating
affairs.summarised <- Affairs %>%
  group_by(rating, gender, children) %>%
  summarise_each(funs(mean(., na.rm = TRUE)), age, yearsmarried,
                 religiousness, education, occupation)

# Predicting on the test dataset based on our bestglm model
affairs.prediction <- predict(bestglm.affairs$BestModel , affairs.summarised,
                              type ="response")

# Creating new dataset with the predictor information and the predicted values
affairs.result <- affairs.summarised
affairs.summarised$affairchance <- affairs.prediction

# Plotting our predictions
plot.affairs.pred <-
  ggplot(affairs.summarised, aes(x = as.factor(rating), y = affairchance)) +
  geom_point(aes(shape=children, colour=gender), size= 10)+
  labs(title = "Chances of having an affair", x = "Happiness Rating (1: Least Happy)",
       y = "Probability of Affair") +
  ylim(0,0.65)

plot.affairs.pred
```



From our plot, we can see various patterns. These are as follows:

1. As happiness rating in an marriage increases, the chances of having an affair decreases.
2. People having children have an tendency to cheat more.
3. Generally males cheat more than females for all ratings of marriage happiness except a rating on 1

Research 2 - USA States

Explore the relationship between a state's Murder rate and other characteristics of the state, for example population, illiteracy rate, and more.

Prerequisite task: Getting the data ready

If we query ?state we can find that there are the following datasets for the 50 US states: state.abb, state.area, state.center, state.division, state.name, state.region, state.x77, Population, Income, Illiteracy, LifeExp, Murder, HSGrad, Frost, Area

We will merge the individual datasetsets into a single dataframe.

```
# We wont take state.area data since we are already getting it from state.x77
# data. We must note that the area in both the datasets do not match with each other but
# they are approximately close.

states <- data.frame(state.abb, state.name, state.region, state.division,
                         state.center, state.x77)

# Let us rename the columns of few data sets to have consistent naming convention

states <- states %>%
  rename(Abb = state.abb, Name = state.name, Region = state.region,
         Division = state.division, Longitude = x, Latitude = y)

# It is important to note that these datasets are ordered which is why we can merge them
# in a single data frame just based on order.
```

Before we proceed with the analysis, we must note that the Murder data was collected for 1976 whereas the data for other variables are census data for different years, such as Illiteracy is for 1970, Population is for 1975 etc.

Hence we will be comparing data collected in different years. However since these years are close to each other we can proceed with our analysis.

Research Question 2(a)

Examine the bivariate relationships present in the data.

Solution 2(a)

Let us examine the bivariate relationships present in the data.

First let us find the correlation between Murder and all other variables

```
# Creating temporary dataset containing subset of states without abbreviation, name,
# region, division, latitude and longitude since these are non numeric
states.multi <- states[,7:14]
cor(x = states.multi, y = states.multi$Murder)
```

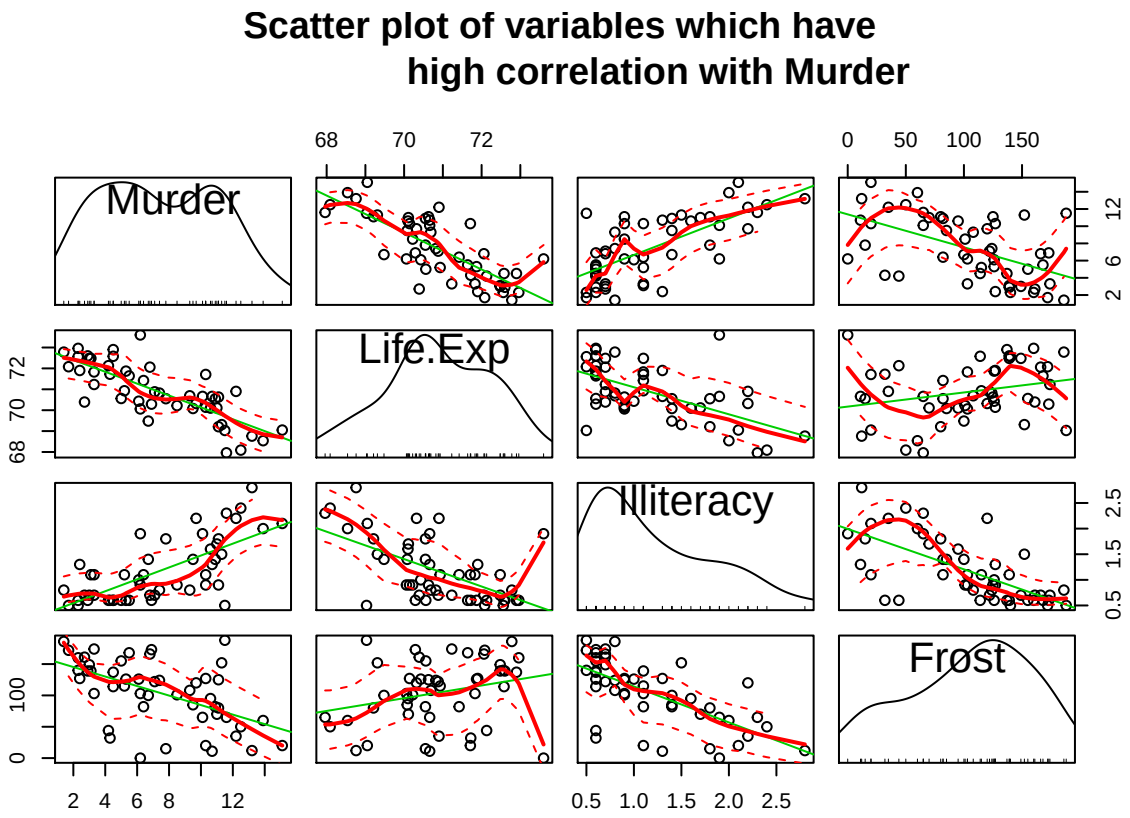
```
##                  [,1]
## Population  0.3436428
## Income     -0.2300776
## Illiteracy  0.7029752
## Life.Exp   -0.7808458
## Murder      1.0000000
## HS.Grad    -0.4879710
## Frost      -0.5388834
## Area        0.2283902
```

We see that the correlation is high for Life Expectancy, Illiteracy and Frost.

Let us also visualize all bivariate relationships to see if we can find any correlations.

We will first check the relationship for which we got a high correlation score

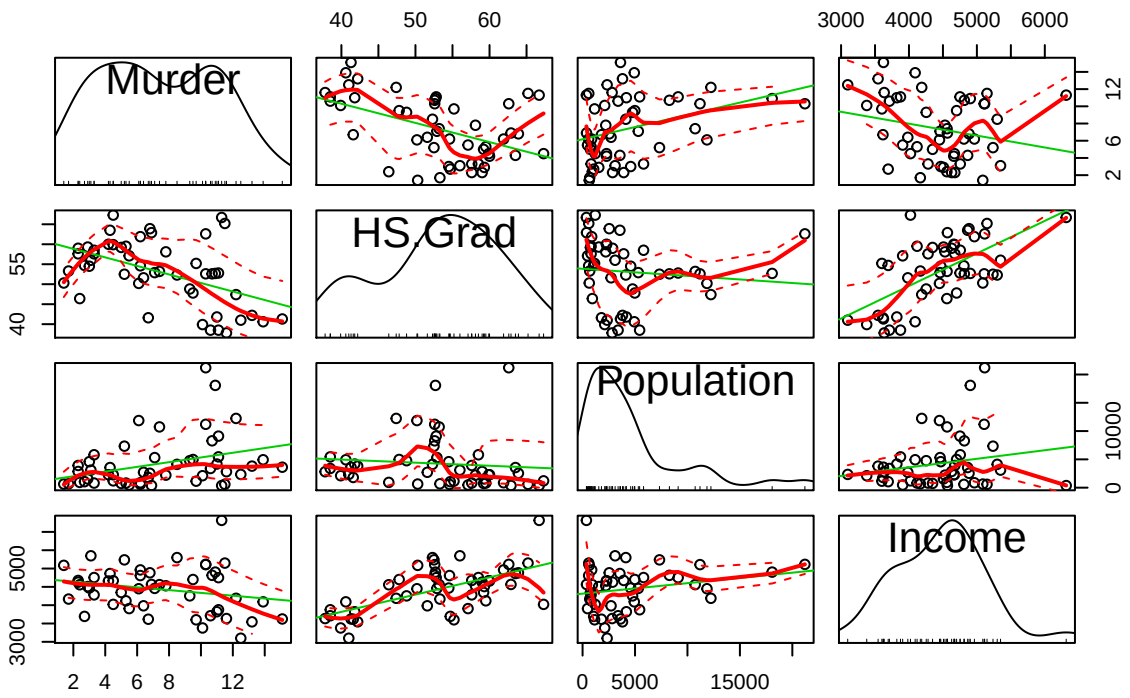
```r
# scatterplotMatrix function to plot variables which have high correlation with Murder
scatterplotMatrix(~ Murder + Life.Exp + Illiteracy + Frost, data = states,
                  main = "Scatter plot of variables which have
                  high correlation with Murder")
```



From the visualization, we can see that murder has a strong direct linear correlation with Illiteracy and strong indirect linear correlation with Frost

Now, let us check the relationship for which we got a low correlation score

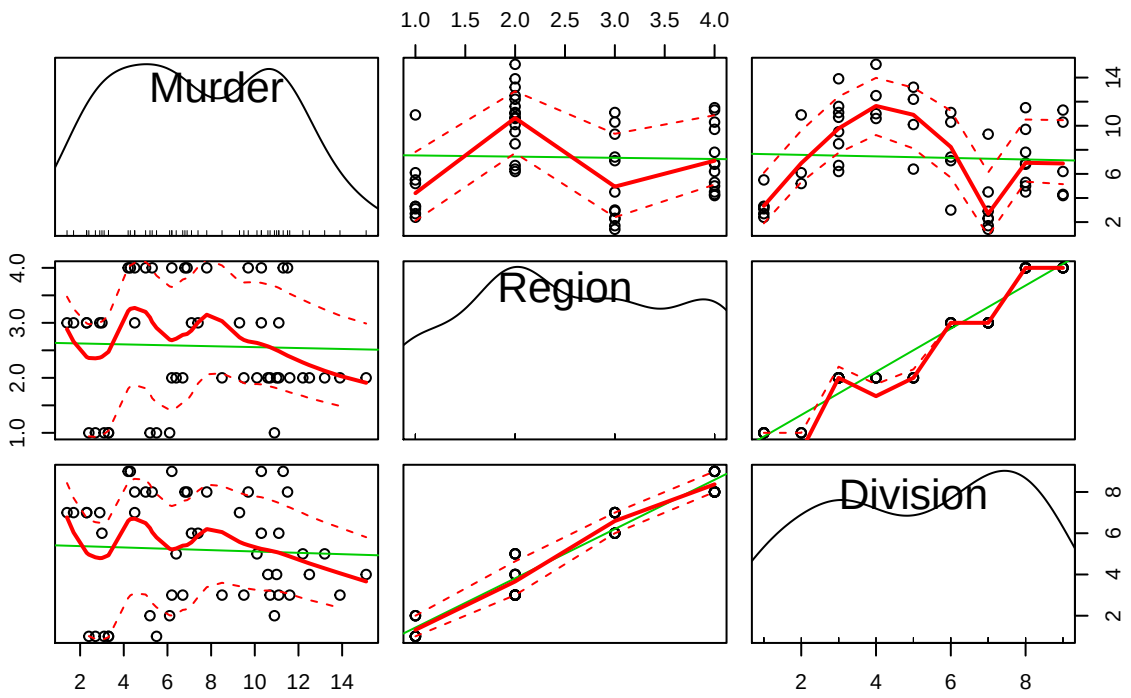
```r
# scatterplotMatrix function to plot variables which have low correlation with Murder
scatterplotMatrix(~ Murder + HS.Grad + Population + Income, data = states,
                  main = "Scatterplot of variables which have
                  low correlation with Murder")
```



We see that population gives a notable linear relationship trend.

Finally let us check relationships for non-numeric predictors

```
# scatterplotMatrix function to check correlation of non-numeric variables with Murder
scatterplotMatrix(~ Murder + Region + Division, data = states,
                  main = "Scatterplot of non-numeric variables with Murder")
```



We see that one region and a few divisions show strong direct correlation to Murder.

Research Question 2(b)

How much variance in the murder rate across states?

Solution 2(b)

```
# Fitting a multiple linear regression predicting Murder using other Independent
# Variables
multi.lm.states <- lm(Murder ~ ., data = states.multi)

# Checking the summary of our model
summary(multi.lm.states)
```

```
##
## Call:
## lm(formula = Murder ~ ., data = states.multi)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -3.4452 -1.1016 -0.0598  1.1758  3.2355
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.222e+02  1.789e+01   6.831 2.54e-08 ***
## Population   1.880e-04  6.474e-05   2.905  0.00584 **
## Income      -1.592e-04  5.725e-04  -0.278  0.78232
## Illiteracy   1.373e+00  8.322e-01   1.650  0.10641
## Life.Exp    -1.655e+00  2.562e-01  -6.459 8.68e-08 ***
## HS.Grad      3.234e-02  5.725e-02   0.565  0.57519
## Frost       -1.288e-02  7.392e-03  -1.743  0.08867 .
## Area         5.967e-06  3.801e-06   1.570  0.12391
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.746 on 42 degrees of freedom
## Multiple R-squared:  0.8083, Adjusted R-squared:  0.7763
## F-statistic: 25.29 on 7 and 42 DF,  p-value: 3.872e-13
```

As we can see from the model, the significant predictors (with p value less than 0.05) are Population and Life Expectancy.

The variance is explained by the multiple R-sqared value which in this case is approximately 81%. Thus we can say that the variance explained by our model or the accuracy of our model is 81%.

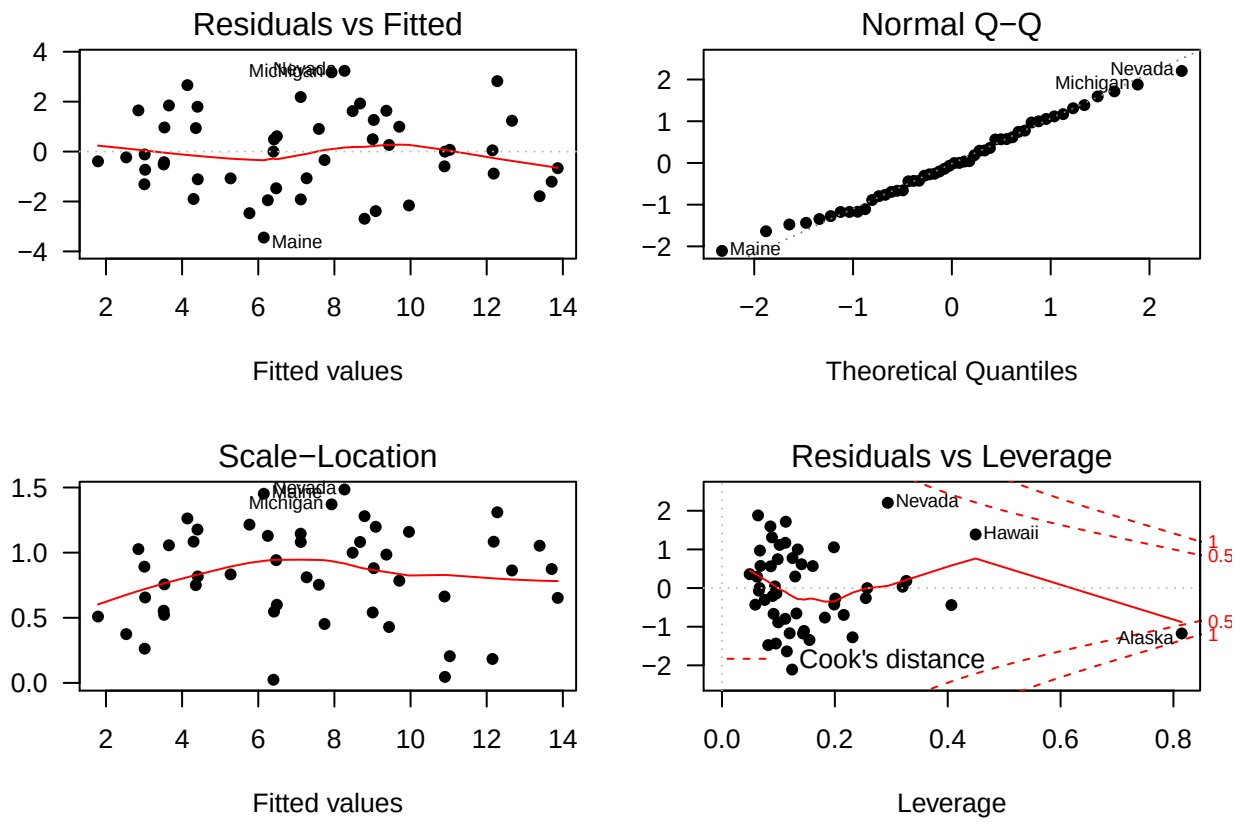
Research Question 2(c)

Analyze the model.

Solution 2(c)

Let us perform a basic analysis of our model residuals.

```
par(mfrow =c(2,2), mar=c(5,3,2,1), las=1, pch=16)

plot(multi.lm.states)
```



We see that the residual vs fitted graph is random for Predictors vs Murder i.e. it does not have any notable pattern. The variables seem to be distributed equally on either side of the mean zero line. Also there are not too many extreme outliers which distort our graph. Hence we can infer that our model is good one.

The statistical assumptions in the previous regression analysis are -

1. The R-squared or accuracy will always increase with the addition of predictable variables even if those variables are weakly associated with the variables.

Research Question 2(d)

Explore different models to obtain a best fit model which predicts the Murder rate. Which variables are the significant predictors?

Solution 2(d)

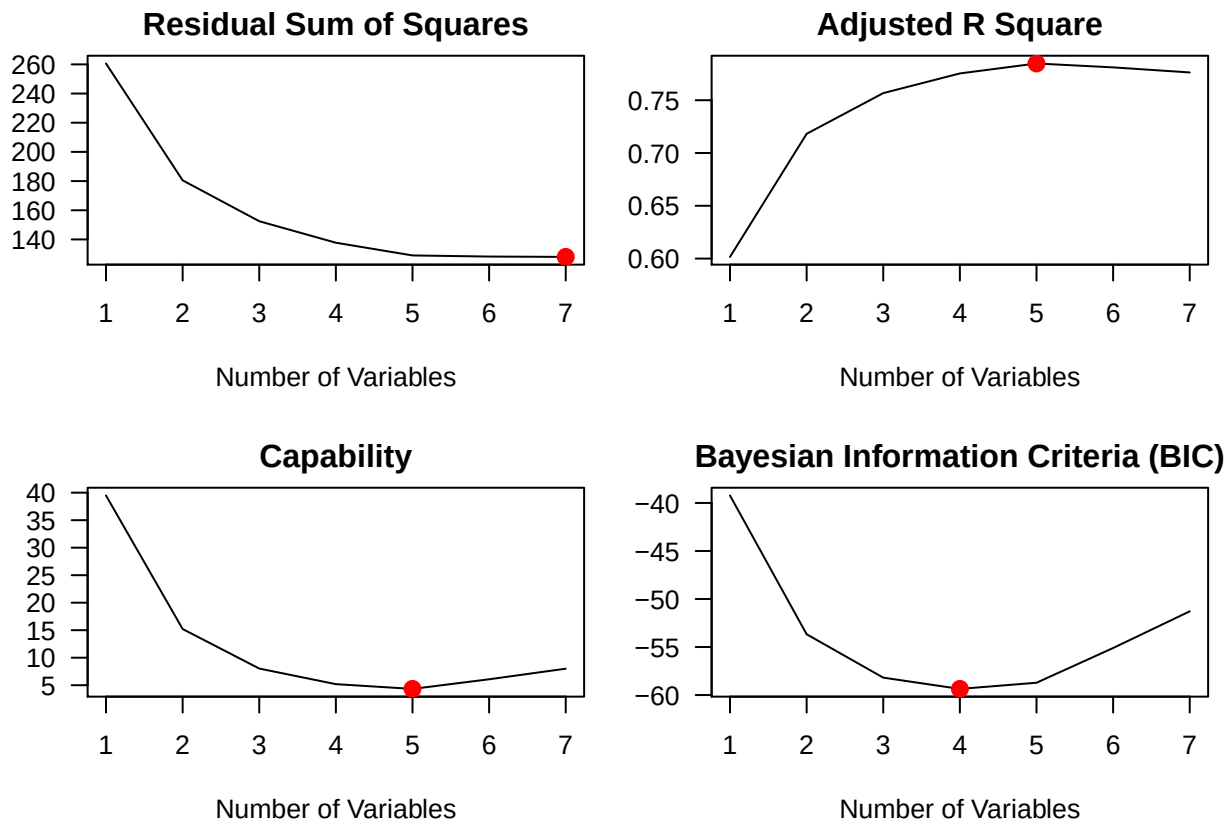
Let us use the stepwise selection model - 'best subset selection' to obtain a best fit model.

```
# The regsubsets() function performs best subregsubsets() set selection by identifying
# the best model that contains a given number of predictors.

# Fitting model using stepwise selection function - regsubsets
regfit.states=regsubsets(Murder ~ ., data = states.multi)
```

```r
# Deriving summary of the stepwise selection model
regfit.summary <- summary(regfit.states)

# Storing the minimum or maximum values for regression sum of squares, adjusted r square,
# process capability and the Bayesian Information Criteria (BIC) based on which is
# statistically significant
min.rss <- which.min(regfit.summary$rss)
max.adjr2 <- which.max(regfit.summary$adjr2)
min.cp <- which.min(regfit.summary$cp)
min.bic <- which.min(regfit.summary$bic)
```

Plotting RSS, adjusted R2, Cp, and BIC for all of the models at once as it will help us decide which model to select.

```r
# Dividing canvas into 4 parts
par(mfrow =c(2,2), mar=c(5,3,2,1), las=1)

# Plotting RSS and highlighting lowest point
plot(regfit.summary$rss ,main = "Residual Sum of Squares",
     xlab=" Number of Variables ",ylab=" RSS", type="l")
points(min.rss, regfit.summary$rss[min.rss], col ="red",cex =2, pch =20)

# Plotting Adjusted R square and highlighting highest point
plot(regfit.summary$adjr2 ,main = "Adjusted R Square",
     xlab =" Number of Variables ", ylab=" Adjusted RSq",type="l")
points(max.adjr2, regfit.summary$adjr[max.adjr2], col ="red",cex =2, pch =20)

# Plotting RSS and highlighting lowest point
plot(regfit.summary$cp ,main = "Capability",
     xlab =" Number of Variables ", ylab=" Adjusted RSq",type="l")
points(min.cp, regfit.summary$cp[min.cp], col ="red",cex =2, pch =20)

# Plotting RSS and highlighting lowest point
plot(regfit.summary$bic ,main = "Bayesian Information Criteria (BIC)",
     xlab =" Number of Variables ", ylab=" Adjusted RSq",type="l")
points(min.bic, regfit.summary$bic[min.bic], col ="red",cex =2, pch =20)
```



As we can observe from the graphs, our model performs best when we chose 4 predictors because:

1. The residual sum of squares is low
2. The adjusted R square is high
3. The capability is low
4. The BIC is the lowest

Hence our best fit model is the model which has 4 predictors.

Finally, let us see which are the 4 predictors which we should choose based on stepwise selection model.

```
# Checking the predictors in the 4-variable model which is our best fit model
coef(regfit.states ,4)
```

```
##   (Intercept)    Population      Life.Exp         Frost          Area
##  1.387215e+02  1.581235e-04 -1.837437e+00 -2.204187e-02  7.387061e-06
```

We see that our step-wise selection process has suggested us 4 predictors - Population, Life Expectancy, Frost and Area.

Let us build a model considering these 4 predictors and test the model on a subset of our data. We will also compute the mean square estimate.

```
# Creating a best fit lm with the 4-variables suggested above on complete dataset
# Creating separately for lm since we will use cv.lm function later
best.step.lm <- lm(Murder ~ Frost + Area + Population + Life.Exp, data = states)
```

```r
# Creating a best fit model with the 4-variables on complete dataset
# Creating separately for glm since we will use cv.glm function later
best.step.glm <- glm(Murder ~ Frost + Area + Population + Life.Exp, data = states)

# Setting the value of seed so that results are reproducible
set.seed(1)

# Creating a new training set with 80% observations to train our model
train.states <- sample(nrow(states), nrow(states)*0.8)

# Creating a best fit glm with the 4-variables suggested above on training dataset
# Creating to compute the mean square estimate and compare later
# glm will give same results as lm since we are using its default behaviour
best.step.glm.train <- glm(Murder ~ Frost + Area + Population + Life.Exp,
                     data = states, subset = train.states)

# Testing our model using test dataset
predict.best.step <- predict(best.step.glm.train, states)[-train.states]

# Computing the mean square error for our prediction
mse.best.step <- mean(((states$Murder)[-train.states] - predict.best.step)^2)
mse.best.step
```

```
## [1] 4.733718
```

As we see the mean square error for our glm model after procuring parameters from step wise selection procedure is 4.73.

After fitting the complete model earlier in part (b), we had come to the conclusion that Population and Life Expectancy are significant predictors.

Thus our full model and our best models are different. Our bestglm model gave us 2 additional predictors - Frost and Area.

In conclusion, according to the stepwise selection model - 'best subset selection', the best fit model for predicting Murder in states is a 4 variable model with the predictors - Frost, Area, Population and Life Expectancy.

Research Question 2(e)

Perform a 10-fold cross validation to estimate model performance.

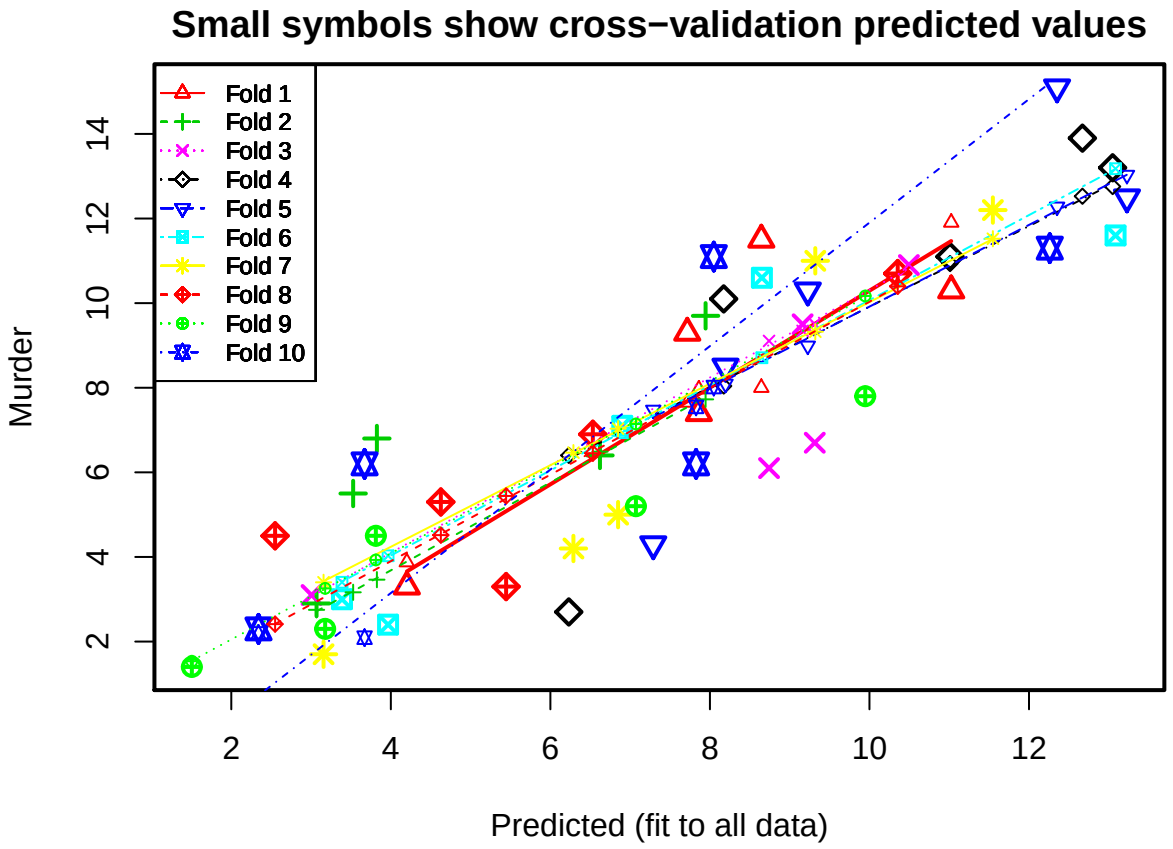
Solution 2(e)

We will check the generalizability of the model by performing a 10 fold cross validation. What this means is that we will divide the data into 10 approximately equal subsets and perform a validation each time by training the data on 9 subsets and testing it on the remaining 1 subset. We will do this 10 times, each time testing on a different subset.

This will give us a good idea if our model is generalizable and gives a good performance with change in dataset.

```r
# Setting canvas to default value
grid.newpage()

# Setting the value of seed so that results are reproducible
set.seed(1)

library(DAAG)  # Loading library containing function for cv.lm

# Performing 10 fold cross validation
cv.lm(data = states, best.step.lm, m=10, seed = 1)
```

```
## Analysis of Variance Table
##
## Response: Murder
##            Df Sum Sq Mean Sq F value  Pr(>F)
## Frost       1  193.9   193.9    63.3   4e-10 ***
## Area        1   45.4    45.4    14.8 0.00037 ***
## Population  1   17.7    17.7     5.8 0.02021 *
## Life.Exp    1  272.9   272.9    89.2   3e-12 ***
## Residuals  45  137.8     3.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```




```
##
```

```
## fold 1
## Observations in test set: 5
##             California Missouri Nevada New Hampshire   Ohio
## Predicted        11.02     7.72   8.64         4.201  7.861
## cvpred           11.91     7.65   8.00         3.877  7.962
## Murder           10.30     9.30  11.50         3.300  7.400
## CV residual      -1.61     1.65   3.50        -0.577 -0.562
##
## Sum of squares = 18.2    Mean square = 3.64    n = 5
##
## fold 2
## Observations in test set: 5
##             Colorado Nebraska New Mexico Oklahoma Vermont
## Predicted       3.83     3.07       7.95    6.622    3.53
## cvpred          3.46     2.75       7.73    6.489    3.16
## Murder          6.80     2.90       9.70    6.400    5.50
## CV residual     3.34     0.15       1.97   -0.089    2.34
##
## Sum of squares = 20.5    Mean square = 4.1    n = 5
##
## fold 3
## Observations in test set: 5
##             Connecticut New York Pennsylvania Virginia West Virginia
## Predicted        3.0063  10.4944         8.74    9.162          9.31
## cvpred           3.0807  10.9109         9.11    9.344          9.51
## Murder           3.1000  10.9000         6.10    9.500          6.70
## CV residual      0.0193  -0.0109        -3.01    0.156         -2.81
##
## Sum of squares = 17    Mean square = 3.39    n = 5
##
## fold 4
## Observations in test set: 5
##             Arkansas Georgia Louisiana Maine North Carolina
## Predicted       8.17   12.67    13.049  6.23         11.010
## cvpred          8.04   12.53    12.756  6.40         10.936
## Murder         10.10   13.90    13.200  2.70         11.100
## CV residual     2.06    1.37     0.444 -3.70          0.164
##
## Sum of squares = 20.1    Mean square = 4.01    n = 5
##
## fold 5
## Observations in test set: 5
##             Alabama Illinois Maryland Mississippi Washington
## Predicted     12.35     9.23    8.195      13.228       7.29
## cvpred        12.28     8.99    8.091      13.029       7.48
## Murder        15.10    10.30    8.500      12.500       4.30
## CV residual    2.82     1.31    0.409      -0.529      -3.18
##
## Sum of squares = 20.2    Mean square = 4.04    n = 5
##
## fold 6
## Observations in test set: 5
##             Indiana Kentucky Rhode Island South Carolina Wisconsin
## Predicted     6.902     8.65         3.97          13.09     3.388
```

```
## cvpred        6.938     8.71         4.03          13.18     3.401
## Murder        7.100    10.60         2.40          11.60     3.000
## CV residual   0.162     1.89        -1.63          -1.58    -0.401
##
## Sum of squares = 8.9    Mean square = 1.78    n = 5
##
## fold 7
## Observations in test set: 5
##             Montana Oregon South Dakota Tennessee  Texas
## Predicted      6.85   6.29         3.16      9.32 11.547
## cvpred         7.03   6.47         3.40      9.32 11.522
## Murder         5.00   4.20         1.70     11.00 12.200
## CV residual   -2.03  -2.27        -1.70      1.68  0.678
##
## Sum of squares = 15.4    Mean square = 3.08    n = 5
##
## fold 8
## Observations in test set: 5
##             Florida Idaho Massachusetts Utah Wyoming
## Predicted    10.354 4.627          5.45 2.55   6.532
## cvpred       10.397 4.516          5.44 2.41   6.452
## Murder       10.700 5.300          3.30 4.50   6.900
## CV residual   0.303 0.784         -2.14 2.09   0.448
##
## Sum of squares = 9.85    Mean square = 1.97    n = 5
##
## fold 9
## Observations in test set: 5
##             Arizona   Iowa Kansas New Jersey North Dakota
## Predicted      9.95  3.177   3.81       7.07        1.505
## cvpred        10.17  3.257   3.93       7.14        1.535
## Murder         7.80  2.300   4.50       5.20        1.400
## CV residual   -2.37 -0.957   0.57      -1.94       -0.135
##
## Sum of squares = 10.6    Mean square = 2.13    n = 5
##
## fold 10
## Observations in test set: 5
##             Alaska Delaware Hawaii Michigan Minnesota
## Predicted    12.26     7.83   3.67     8.05     2.341
## cvpred       16.53     7.55   2.09     8.02     2.192
## Murder       11.30     6.20   6.20    11.10     2.300
## CV residual  -5.23    -1.35   4.11     3.08     0.108
##
## Sum of squares = 55.6    Mean square = 11.1    n = 5
##
## Overall (Sum over all 5 folds)
##   ms
## 3.93
```

We can see from the 10 fold cross validation that all the 10 folds fit the data in approximately the same pattern. We also see that for each of the folds that the actual value for Murder and the predicted value for Murder are close to each other.

From this we can infer that our model gives same results on different subsets and hence it is generalizable.

Let us further corroborate this with cv.glm

```
library(boot) # Loading library containing function for cv.glm

# Setting the value of seed so that results are reproducible
set.seed(1)

# Calculating k-fold Cross Validation for our best glm model
cv.states <- cv.glm(states, best.step.glm, K=10)

# Checking the mean square error for our k-fold Cross Validation
mse.cv <- cv.states$delta[2]
mse.cv
```

```
## [1] 3.84
```

We see that the mean square error for one of our prediction on a random test set above (4.73) and the mean square error we got from k-fold Cross Validation (3.84) are close to each other. Since Murder varies from 1.4 to 15.1, this variance is not bad.

This means that our model is generalizable and works well on different subsets of data.

Research Question 2(f)

Fit a regression tree using the same covariates in the best fit model from part (d).

Solution 2(f)

Let us fit a regression tree using same 4 covariates as in our best fir model from part (d).

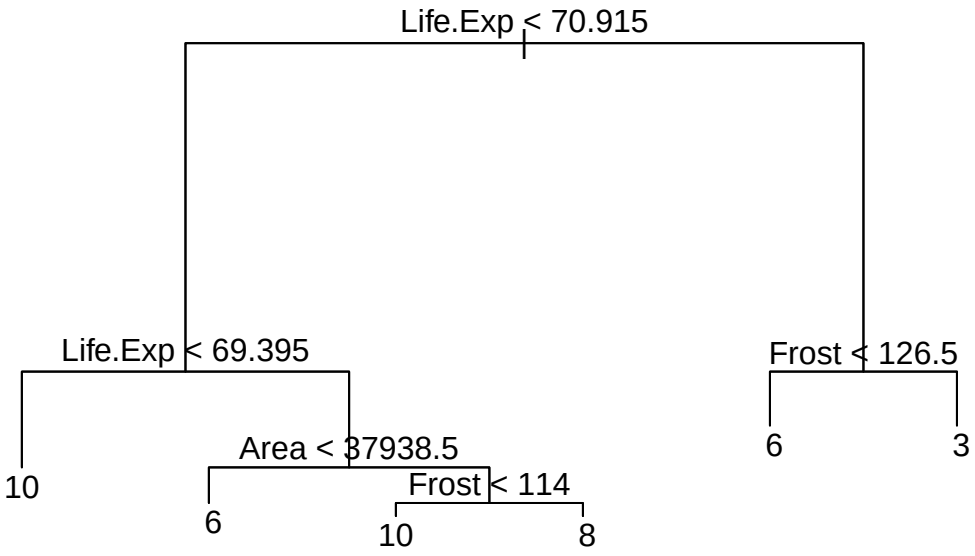
```
library(tree)  # Loading library for tree

# Fitting a regression tree model using covariates found from best fit model
tree.states = tree(Murder ~ Frost + Area + Population + Life.Exp, data = states,
                   subset = train.states)

# Let us look at the performance on the training set
summary(tree.states)
```

```
##
## Regression tree:
## tree(formula = Murder ~ Frost + Area + Population + Life.Exp,
##     data = states, subset = train.states)
## Variables actually used in tree construction:
## [1] "Life.Exp" "Area"     "Frost"
## Number of terminal nodes:  6
## Residual mean deviance:  2.57 = 87.4 / 34
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -3.54   -1.00   -0.06    0.00    0.72    4.02
```

We see in the training set the value of the Residual mean deviance which is equivalent to the mean square error.

let us visualize the tree

```
# Visualizing our tree
plot(tree.states)
text(tree.states ,pretty =0)
```

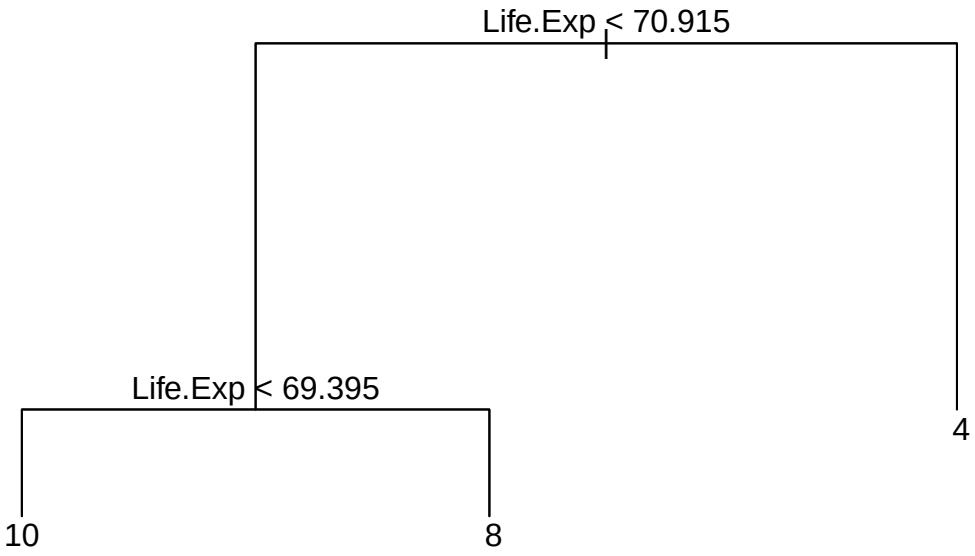


Next let us see if we can build a better tree by pruning it.

```
# Setting the value of seed so that results are reproducible
set.seed(1)

# Now we use the cv.tree() function to see whether pruning the tree will improve
# the performance.
cv.t =cv.tree(tree.states)

# Visualizing the size of the tree vs the Deviance
plot(cv.t$size ,cv.t$dev ,type='b', main = "Size vs Deviance for tree",
     xlab = "Size", ylab = "Deviance")
```

We see that we get less deviance when tree is pruned to 3 leaf nodes or more. Further, since the complexity of our model will increase with the increase in leaf nodes, for the sake of simplicity and performance, we will prune the tree to 3 leaf nodes.

```
# Pruning the tree to 3 leaf nodes
prune.states = prune.tree(tree.states ,best = 3)

# Visualizing our tree
plot(prune.states)
text(prune.states ,pretty =0)
```



The pruned tree suggests that only Life Expectancy should be taken as a predictor for the best tree.

Let us fit a new tree with only the one predictor.

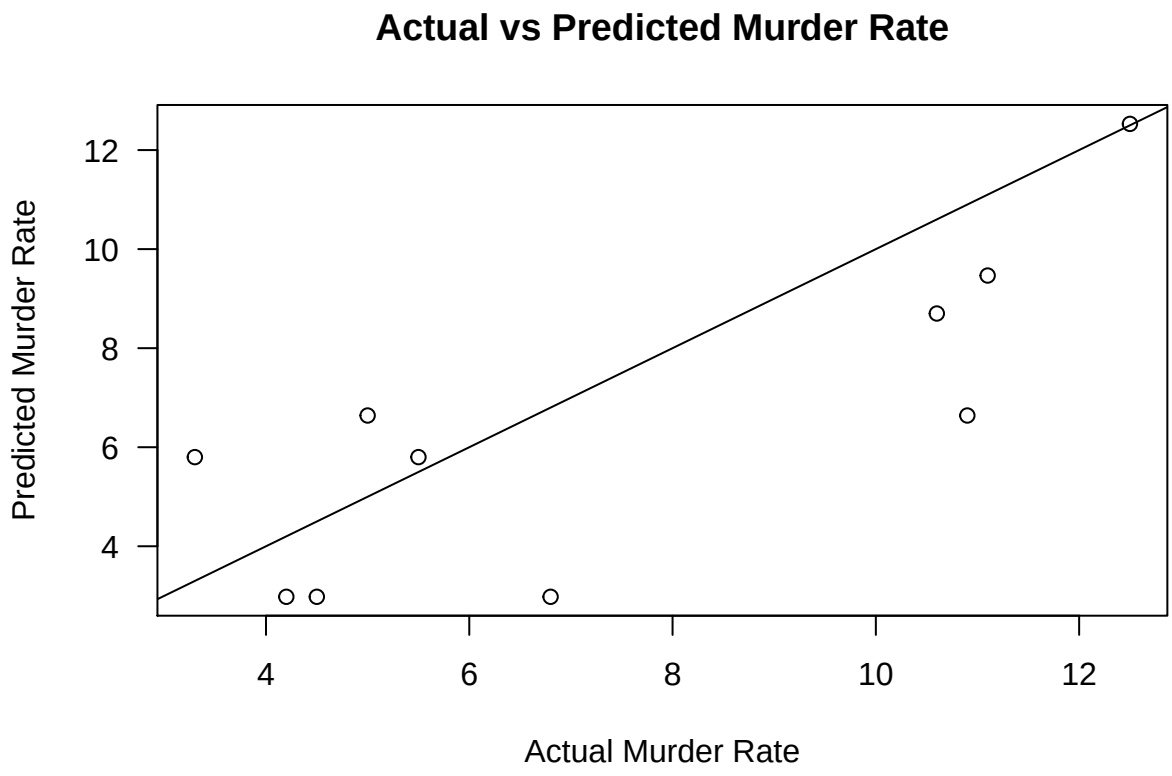
```
# Fitting a regression tree model using covariates found from best fit model
tree.pruned.states = tree(Murder ~ Life.Exp, data = states,
                  subset = train.states)

# Let us look at the performance on the training set
summary(tree.pruned.states)
```

```
##
## Regression tree:
## tree(formula = Murder ~ Life.Exp, data = states, subset = train.states)
## Number of terminal nodes:  6
## Residual mean deviance:  3.73 = 127 / 34
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -3.94   -1.24   -0.18    0.00    1.18    4.50
```

We see in the training set the value of the Residual mean deviance which is equivalent to the mean square error.

Now let us predict Murder rate using test set

```r
# Creating vector to store actual Murder rates for test data
tree.test = states[-train.states,"Murder"]

# Predicting Murder rate using pruned tree model
predict.best.tree = predict(tree.pruned.states ,newdata = states[-train.states,])

# Plotting the predicted murder rate against the actual murder rate
plot(tree.test, predict.best.tree, main = "Actual vs Predicted Murder Rate",
     xlab = "Actual Murder Rate", ylab = "Predicted Murder Rate", las = 1)
abline (0,1)
```



Now let us check the mean square error for our prediction

```r
mse.best.tree <- mean((predict.best.tree - tree.test)^2)
mse.best.tree
```

```
## [1] 5.18
```

Thus, our mean square error for our pruned tree is 5.18.

Research Question 2(g)

Which model is better?

Solution 2(g)

We can compare the models based on the mean squared error. The lesser the mean square error, the better our model is.

Let us recall the values of mean square errors for our linear regression model and tree regression model

```
mse.best.step  # Mean square error of best glm model after stepwise selection
```

```
## [1] 4.73
```

```
mse.best.tree  # Mean square error of best tree model after pruning
```

```
## [1] 5.18
```

We see that the mean square error for the best glm model after step wise selection is less than that of the best tree model after pruning.

Hence I would prefer the linear regression model over the tree model.

Research 3 - Breast Cancer

The Wisconsin Breast Cancer dataset is available as a comma-delimited text file on the UCI Machine Learning Repository http://archive.ics.uci.edu/ml. Our goal in this problem will be to predict whether observations (i.e. tumors) are malignant or benign.

Research Question 3(a)

Describe the data

Solution 3(a)

```
# Reading the data using read.csv function
wbcd <- tbl_df(read.csv(
  "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisc
```

This breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg as of 15'th July 1992.

The data consists of 698 observations of patients. The columns consist of various medical measurements and also let us know if the patient in concern is classified as having malignant cell growth (Cancer causing) or benign one (non Cancer causing)

Reference - https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.names

---

Research Question 3(b)

Tidy the data.

Solution 3(b)

```
# The columns need to be renamed. The reference used was -
# https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/
# breast-cancer-wisconsin.names

colnames(wbcd)[1:11] <- c("code.number", "clump.thickness", "uniformity.cell.size",
                          "uniformity.cell.shape","marginal.adhesion",
                          "epithelial.cell.size", "bare.nuclei", "bland.chromatin",
                          "normal.nucleoli", "mitosis", "class")

# Removing NA values from the column bare.nuclei
wbcd$bare.nuclei[wbcd$bare.nuclei == "?" ] = NA

# Converting data type of bare.nuclei to integer to keep consistent format
wbcd$bare.nuclei <- as.integer(wbcd$bare.nuclei)

# Converting data type of class to factor and renaming the values appropriately
wbcd$class <- as.factor(wbcd$class)
levels(wbcd$class) <- c("Benign","Malignant")

# Let us look at our data
head(wbcd)
```

```
## Source: local data frame [6 x 11]
##
##   code.number clump.thickness uniformity.cell.size uniformity.cell.shape
##         (int)           (int)                (int)                 (int)
## 1     1000025               5                    1                     1
## 2     1002945               5                    4                     4
## 3     1015425               3                    1                     1
## 4     1016277               6                    8                     8
## 5     1017023               4                    1                     1
## 6     1017122               8                   10                    10
## Variables not shown: marginal.adhesion (int), epithelial.cell.size (int),
##   bare.nuclei (int), bland.chromatin (int), normal.nucleoli (int), mitosis
##   (int), class (fctr)
```

Research Question 3(c)

Split the data into a training and validation set.

Solution 3(c)

```
# Setting the value of seed so that results are reproducible
set.seed(1)

# Creating vector of indices which will hold training sample
# Such that training data has 70% of the overall data
train <- sample(nrow(wbcd),nrow(wbcd)*0.7)

# Creating datasets for training and testing based on the sampling indices created above
wbcd.train <- wbcd[train,]
wbcd.test <- wbcd[-train,]
```

Research Question 3(d)

Fit a regression model to predict whether tissue samples are malignant or benign i.e predict whether a patient has cancer or not.

Solution 3(d)

```
# Training logistic regression model on train data using glm function
glm.wbcd.train <- glm(class ~ ., family = "binomial", wbcd.train, na.action = na.omit)

# Getting predicted probablities
predict.glm.wbcd <- predict(glm.wbcd.train, newdata = wbcd.test, type="response")

# Classifying cases in the validation set and constructing the confusion matrix
conf.matrix.glm <- table(wbcd.test$class, predict.glm.wbcd > 0.5)
conf.matrix.glm
```

```
##
##              FALSE TRUE
##   Benign       134    3
##   Malignant      7   64
```

From the confusion matrix we see that our model predicts correctly 198 times out of 208 times.

```
# Computing accuracy of confusion matrix
(conf.matrix.glm[1] + conf.matrix.glm[4]) / sum(conf.matrix.glm) * 100
```

```
## [1] 95.2
```

That is 95.2% accuracy.

We also see that the false positives are 7 and the false negatives are 3. Ideally, we should not have any false positives or false negatives. But even if we do, in this domain of medical prediction, false negative should be as low as possible.

Research Question 3(e)

(e) Fit a random forest model to predict whether tissue samples are malignant or benign.

Solution 3(e)

```
library(randomForest) # R package to fit a random forest model

# Setting the value of seed so that results are reproducible
set.seed(1)

# creating test dataset for Random Forest
rf.test <- wbcd[-train,]

# Fitting the Random Forest model
rf.wbcd <- randomForest(class ~ ., data = wbcd.train, na.action = na.omit)

# Predicting survival using our random tree model
predict.rf.wbcd <- predict(rf.wbcd, newdata = rf.test)

# Constructing confusion matrix
conf.matrix.rf <- table(rf.test$class, predict.rf.wbcd)
conf.matrix.rf
```

```
##            predict.rf.wbcd
##             Benign Malignant
##   Benign       134         3
##   Malignant      6        65
```

From the confusion matrix we see that our model predicts correctly 199 times out of 208 times.

```
# Computing accuracy of confusion matrix
(conf.matrix.rf[1] + conf.matrix.rf[4]) / sum(conf.matrix.rf) * 100
```

```
## [1] 95.7
```

That is approximately 95.7% accuracy.

We also see that the false positives are 6 and the false negatives are 3. Ideally, we should not have any false positives or false negatives. But even if we do, in this domain of medical prediction, false negative should be as low as possible.

Research Question 3(f)

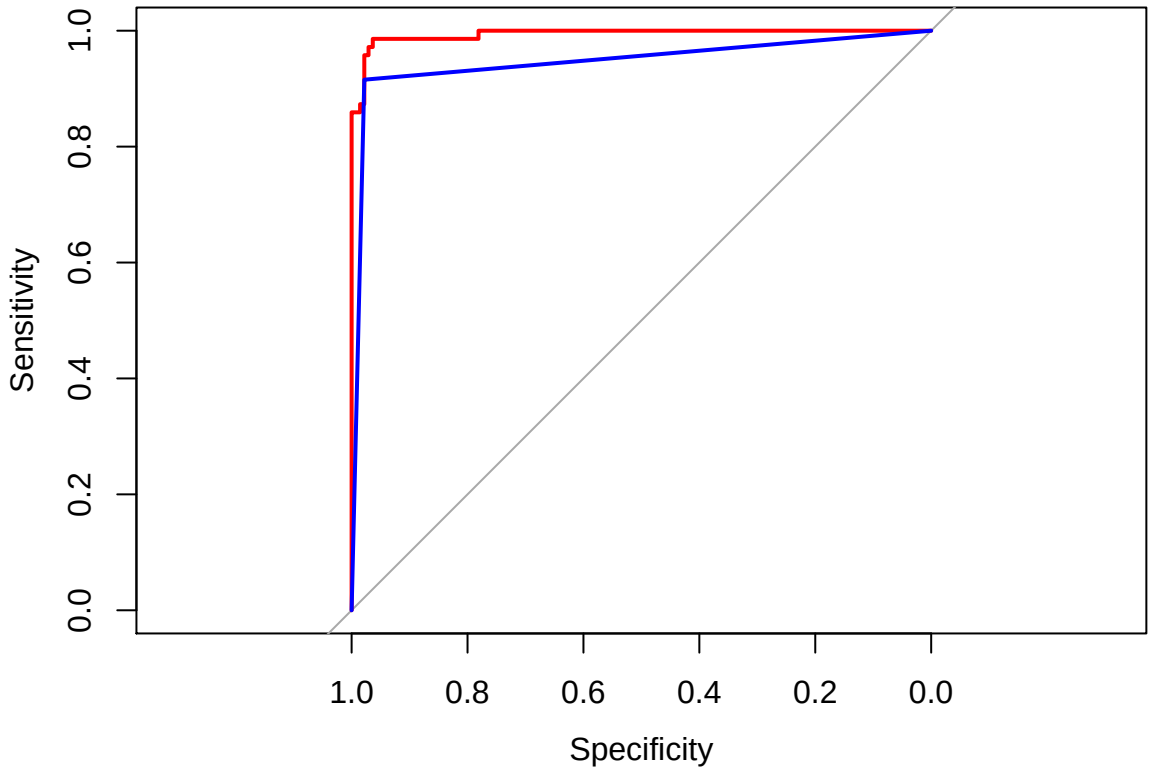
(f) Compare the models.

Solution 3(f)

```
library(pROC)  # R package to plot a ROC curve

# Creating the ROC curves
roc.glm <- roc(wbcd.test$class, predict.glm.wbcd)
roc.rf <- roc(rf.test$class, as.ordered(predict.rf.wbcd))

# Plotting the ROC curves
plot(roc.glm, col='red')
```

```
##
## Call:
## roc.default(response = wbcd.test$class, predictor = predict.glm.wbcd)
##
## Data: predict.glm.wbcd in 137 controls (wbcd.test$class Benign) < 71 cases (wbcd.test$class Malignan
## Area under the curve: 0.994
```

```
lines(roc.rf, col='blue')
```




```
# Calculating the percentage of Area Under Curve (AUC)
roc.glm$auc*100  # ROC for Logistic model
```

```
## [1] 99.4
```

```
roc.rf$auc*100  # ROC for Random Forest model
```

```
## [1] 94.7
```

As we see, even though the predictions were better for our random tree model, the area under the ROC curve is greater for our logistic model.

We are getting a very high accuracy for both the ROC curves (>94%). This might indicate overfitting of the data.

Since area under ROC curve is indicative of the approximate accuracy, we can conclude that the learning method using logistic regression is better for predicting cancer compared to random forest method.

Hence I would prefer the logistic regression model to the random forest model.

Research 4 - Auto Mileage

Apply boosting, bagging and random forests to a fit prediction model and evaluate their performance.

Solution 6

We will apply the following techniques on the Auto dataset to fit a model which predicts the miles per gallons. We will evaluate the performances of all the techniques.

1. Random Forest
2. Bagging
3. Boosting
4. Linear Regression

The Auto dataset we are using contains data for 392 observations. There are various parameters for a auto such as miles per gallon(mpg), cylinders, displacement, horsepower, weight, acceleration, year and origin.

```
library(ISLR)  # Loading library containing Auto data

# Setting the value of seed so that results are reproducible
set.seed(1)

# Loading the auto data except name of the auto since we do not need it
auto <- Auto[, 1:8]

# Creating a training subset which contains 70% of the data
train.auto <- sample(nrow(auto), nrow(auto)*0.7)

# 1. Random Forest
# Fitting the Random Forest model
rf.auto <- randomForest(mpg ~ ., data = auto, subset = train.auto)
```

```
# Predicting mpg using our random tree model
predict.rf.auto <- predict(rf.auto, newdata = auto[-train.auto,])


# 2. Bagging
# Bagging is a special case of a random forest with m = p. Here our m = p = 10 predictors
# Fitting the Bagging model
bag.auto <- randomForest(mpg ~ .,data = auto ,subset = train.auto, mtry=7,
                         importance =TRUE)
bag.auto
```

```
##
## Call:
##  randomForest(formula = mpg ~ ., data = auto, mtry = 7, importance = TRUE,      subset = train.auto)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##           Mean of squared residuals: 7.25
##                     % Var explained: 88
```

```
# Predicting mpg using our bagging model
predict.bag.auto <- predict (bag.auto ,newdata = auto[-train.auto,])


# 3. Boosting
library(gbm)  # Loading library containing gbm funtion for boosting
# Fitting the Boosting model
boost.auto = gbm(mpg ~.,data= auto[train.auto,], distribution="gaussian",
                 n.trees =5000 , interaction.depth =4)

# Predicting mpg using our boosting model
predict.boost.auto  <- predict(boost.auto ,newdata = auto[-train.auto,], n.trees =5000)
```

Research Question 6(a)

How accurate are the results compared to simple methods like linear or logistic regression?

Solution 6(a)

Let us fit another model for the above, this time using linear regression

```
lm.auto <- lm(mpg ~ ., data = auto, subset =train.auto)
summary(lm.auto)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = auto, subset = train.auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -9.621 -2.259 -0.076  2.003 13.338
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.68e+01   5.67e+00   -2.96   0.0034 **
## cylinders    -6.47e-01   3.90e-01   -1.66   0.0984 .
## displacement  1.80e-02   9.16e-03    1.96   0.0507 .
## horsepower   -2.77e-02   1.63e-02   -1.70   0.0901 .
## weight       -5.54e-03   7.67e-04   -7.22  5.3e-12 ***
## acceleration -1.35e-02   1.17e-01   -0.12   0.9084
## year          7.56e-01   6.15e-02   12.28  < 2e-16 ***
## origin        1.45e+00   3.44e-01    4.22  3.3e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.34 on 266 degrees of freedom
## Multiple R-squared:  0.819,  Adjusted R-squared:  0.815
## F-statistic:  172 on 7 and 266 DF,  p-value: <2e-16
```

We see that the significant predictors (p value greater than 0.05) are weight, year and origin. Let us create a new model using only the significant predictors

```
# Creating linear model with only significant predictors
lm.auto.significant <- lm(mpg ~ weight + year + origin, data = auto, subset = train.auto)

# Predicting mpg using our linear model
predict.lm.auto <- predict(lm.auto.significant, newdata = auto[-train.auto,])
```

Now let us see how our linear model compares to the other models

```
# Creating vector which will store actual mpg value
test.auto <- auto[-train.auto ,"mpg"]

# Calculating the mean square error for each model
mse.rf.auto <- mean((predict.rf.auto - test.auto)^2)
mse.bag.auto <- mean((predict.bag.auto - test.auto)^2)
mse.boost.auto <- mean((predict.boost.auto - test.auto)^2)
mse.lm.auto <- mean((predict.lm.auto - test.auto)^2)

# Creating data frame to store all mean square error  values and dispay them
model.name <- c("Random Forest", "Bagging", "Boosting", "Linear")
model.mse <- c(mse.rf.auto, mse.bag.auto, mse.boost.auto, mse.lm.auto)
model.all.mse <- data_frame(model.name, model.mse)
model.all.mse
```

```
## Source: local data frame [4 x 2]
##
##      model.name model.mse
##           (chr)     (dbl)
## 1 Random Forest      7.93
## 2       Bagging      8.44
## 3      Boosting      8.30
## 4        Linear     11.15
```

We see from the mean square error that our linear model has the highest mean square error and hence we can conclude that it is not as accurate as the other models.

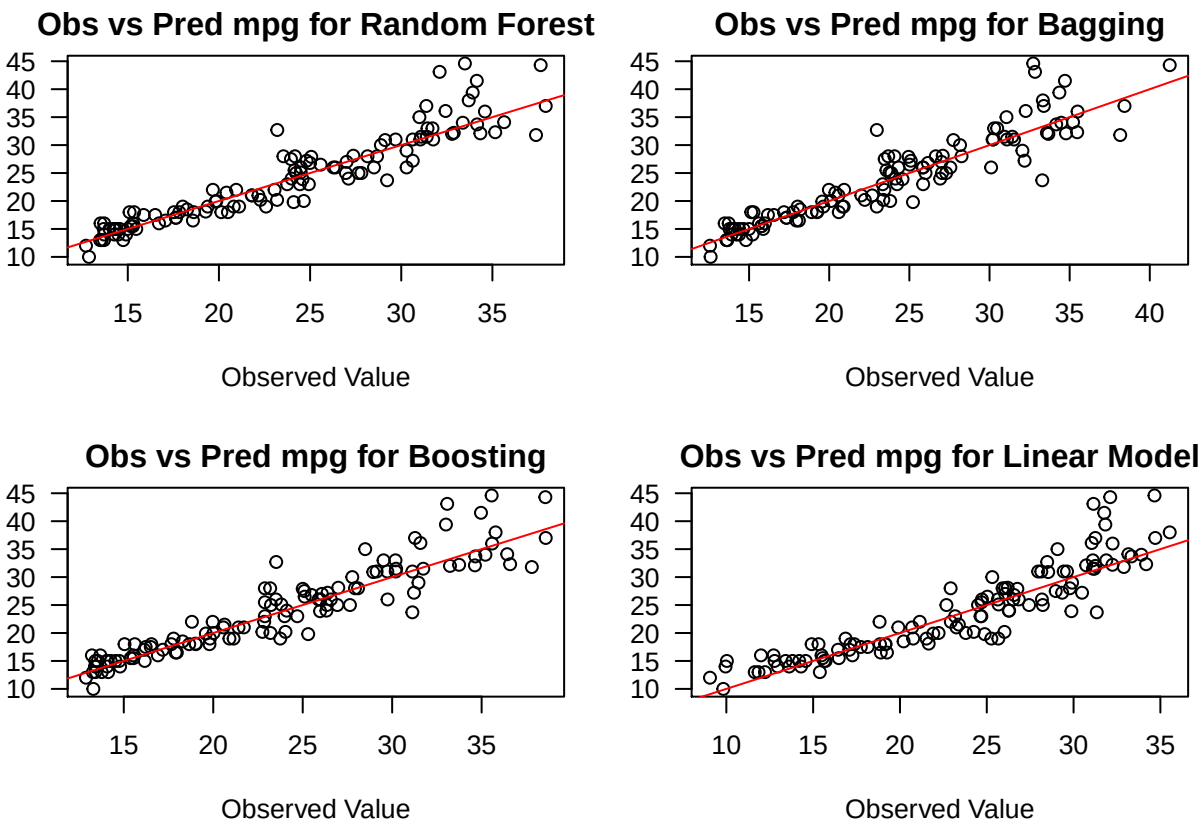
Research Question 6(b)

Which of the approaches yields the best performance?

Solution 6(b)

Let us first visualise how all our models fit the data

```
# Dividing our canvas into 4 parts
par(mfrow=c(2,2), mar=c(5,3,2,1), las=1)

# Plotting observed value versus predicted value for random forest model
plot(predict.rf.auto, test.auto, main = "Obs vs Pred mpg for Random Forest",
     xlab = "Observed Value", ylab = "Predicted Value")
abline (0,1, col = "red")

# Plotting observed value versus predicted value for bagging model
plot(predict.bag.auto, test.auto, main = "Obs vs Pred mpg for Bagging",
     xlab = "Observed Value", ylab = "Predicted Value")
abline (0,1, col = "red")

# Plotting observed value versus predicted value for boosting model
plot(predict.boost.auto, test.auto, main = "Obs vs Pred mpg for Boosting",
     xlab = "Observed Value", ylab = "Predicted Value")
abline (0,1, col = "red")

# Plotting observed value versus predicted values
plot(predict.lm.auto, test.auto, main = "Obs vs Pred mpg for Linear Model",
     xlab = "Observed Value", ylab = "Predicted Value")
abline (0,1, col = "red")
```



We see from the plot that the predicted value against the actual value lies approximately on the 45 degree line for all our models. This means that all our models are fairly accurate.

Now let us evaluate the performance of the models by looking at the mean square error for each model again.

```
model.all.mse
```

```
## Source: local data frame [4 x 2]
##
##      model.name model.mse
##           (chr)     (dbl)
## 1 Random Forest      7.93
## 2       Bagging      8.44
## 3      Boosting      8.30
## 4        Linear     11.15
```

The best model is the one that has the least mean square error. In this case, we see that the Random Forest model has the least mean square error and hence we can say that it yields the best performance.

References:

1. A Handbook of Statistical Analysis Using R - Everitt, B. and Hothorn, T.
2. An Introduction to Statistical Learning - James, G. et all
3. OpenIntro Statistics - Diez, D. et all

4. Canvas lecture slides for INFX 573 (University of Washington - Autumn 2015)

Nelson Dsouza 12/14/2015