# SDM_Assignment2_1

Sri Balaji Muruganandam

02/10/2021

## Setting Working Directory

```
rm(list = ls())
setwd("G:\\SDM_Sem01\\Assignment2")
```

## Importing necessary libraries

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.1.1
```

## Importing Cereal data from local path

```
load("cereal_clean_data.RData")
#c_data <- read.delim("cereal.csv",sep = ",")
dim(c_data)
```

```
## [1] 75 16
```

## Viewing the Sample data

```
head(c_data, 5)
```

```
##                         name mfr type calories protein fat sodium fiber carbo
## 1                  100% Bran   4    1       70       4   1    130  10.0   5.0
## 2          100% Natural Bran   6    1      120       3   5     15   2.0   8.0
## 3                   All-Bran   3    1       70       4   1    260   9.0   7.0
## 4 All-Bran with Extra Fiber   3    1       50       4   0    140  14.0   8.0
## 6   Apple Cinnamon Cheerios   2    1      110       2   2    180   1.5  10.5
##   sugars   potass vitamins shelf weight cups    rating
## 1      6 2.447158     0.25     3      1 0.33 0.6840297
## 2      8 2.130334     0.00     3      1 1.00 0.3398368
## 3      5 2.505150     0.25     3      1 0.33 0.5942551
## 4      0 2.518514     0.25     3      1 0.50 0.9370491
## 6     10 1.845098     0.25     1      1 0.75 0.2950954
```

```
tail(c_data, 5)
```

```
##                   name mfr type calories protein fat sodium fiber carbo sugars
## 73             Triples   2    1      110       2   1    250     0    21      3
## 74                Trix   2    1      110       1   1    140     0    13     12
## 75          Wheat Chex   7    1      100       3   1    230     3    17      3
## 76            Wheaties   2    1      100       3   1    200     3    17      3
## 77 Wheaties Honey Gold   2    1      110       2   1    200     1    16      8
##      potass vitamins shelf weight cups    rating
## 73 1.778151     0.25     3      1 0.75 0.3910617
## 74 1.397940     0.25     2      1 1.00 0.2775330
## 75 2.060698     0.25     1      1 0.67 0.4978744
## 76 2.041393     0.25     1      1 1.00 0.5159219
## 77 1.778151     0.25     1      1 0.75 0.3618756
```

## Splitting 80% data for training and 20% for the test data

```
#new_data = sort(sample(nrow(c_data), nrow(c_data)*.8))
set.seed(23)
new_data = sample(c(1:length(c_data[,1])), size = round(8/10 * length(c_data[,1])), replace = FALSE)
train_data <- c_data[new_data,]
test_data <- c_data[-new_data,]
y_train_data <- train_data$rating
y_test_data <- test_data$rating
```

# Performing Linear Regression

## Fitting a linear model and predicting the Mean Square Error(MSE) for the test data

```
model_fit <- lm(rating~., data = train_data[,2:16])
train_MSE <- mean(model_fit$residuals^2)
names(model_fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

## Summary of Regression model

```
summary(model_fit)
```

```
##
## Call:
## lm(formula = rating ~ ., data = train_data[, 2:16])
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0119251 -0.0025788  0.0002787  0.0028211  0.0140271
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.068e-01  1.197e-02  50.685  < 2e-16 ***
## mfr         -3.277e-04  4.717e-04  -0.695  0.49082
## type         1.169e-02  6.281e-03   1.862  0.06921 .
## calories    -1.923e-03  1.461e-04 -13.161  < 2e-16 ***
## protein      3.187e-02  1.214e-03  26.252  < 2e-16 ***
## fat         -1.933e-02  1.527e-03 -12.662  < 2e-16 ***
## sodium      -5.585e-04  1.039e-05 -53.764  < 2e-16 ***
## fiber        2.850e-02  8.482e-04  33.599  < 2e-16 ***
## carbo        9.997e-03  6.816e-04  14.666  < 2e-16 ***
## sugars      -8.616e-03  6.236e-04 -13.815  < 2e-16 ***
## potass      -3.369e-02  5.558e-03  -6.061 2.53e-07 ***
## vitamins    -4.482e-02  3.678e-03 -12.186 7.49e-16 ***
## shelf       -3.003e-03  1.058e-03  -2.840  0.00676 **
## weight      -1.562e-02  1.213e-02  -1.288  0.20441
## cups        -8.378e-03  3.914e-03  -2.140  0.03779 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00529 on 45 degrees of freedom
## Multiple R-squared:  0.9988, Adjusted R-squared:  0.9984
## F-statistic:  2607 on 14 and 45 DF,  p-value: < 2.2e-16
```

## Fitting the test data and calculating the MSE

```
model_predict_test <- predict.lm(model_fit, test_data[,2:15])
test_MSE <- mean((y_test_data - model_predict_test)^2)
print(test_MSE)
```

```
## [1] 4.315096e-05
```

```
print(round(test_MSE, digit = 7))
```

```
## [1] 4.32e-05
```

# Performing Forward Subset Selection

## Creating objects to store errors

```
train_error_st <- matrix(rep(NA,14))
test_error_st <- matrix(rep(NA,14))
```

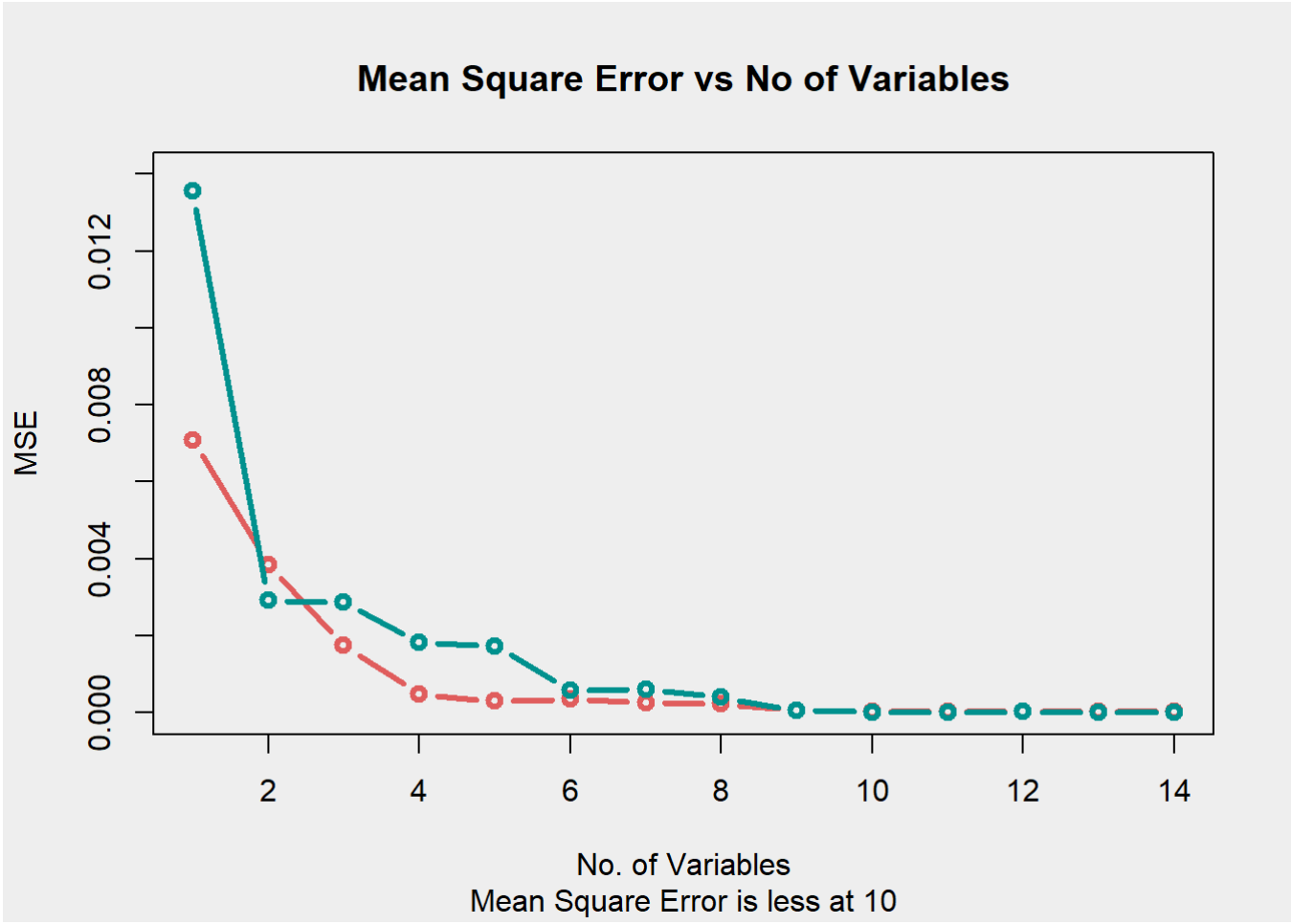
```
predict.regsubsets = function(object, newdata, id){
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, newdata)
    coefi = coef(object,id=id)
    xvars=names(coefi)
    mat[,xvars]%*%coefi
}
```

```
fwd_subset <- regsubsets(rating~., data = c_data[,2:16], nbest = 1, nvmax = 15, method = "for
ward")
```

```
for(i in 1:14){
  y_hat_train = predict(fwd_subset, newdata = train_data[,2:16], id = i)
  y_hat_test = predict(fwd_subset, newdata = test_data[,2:16], id = i)

  train_error_st[i] = (1/length(y_train_data))*sum((y_train_data-y_hat_train)^2)
  test_error_st[i] = (1/length(y_test_data))*sum((y_test_data-y_hat_test)^2)
}
```

```
par(bg = '#EEEEEE')
plot(train_error_st, col="#E05D5D", type = "b", xlab = "No. of Variables", ylab = "MSE", ylim
= c(0,0.014), main = "Mean Square Error vs No of Variables", sub="Mean Square Error is less a
t 10", lwd = 3.0)
lines(test_error_st, col = "#00918E", type = "b", lwd = 3.0)
```




## Subset with minimum error

```
which(test_error_st == min(test_error_st))
```

```
## [1] 10
```

```
print(min(test_error_st))
```

```
## [1] 1.420306e-05
```

# Performing Exhaustive Selection

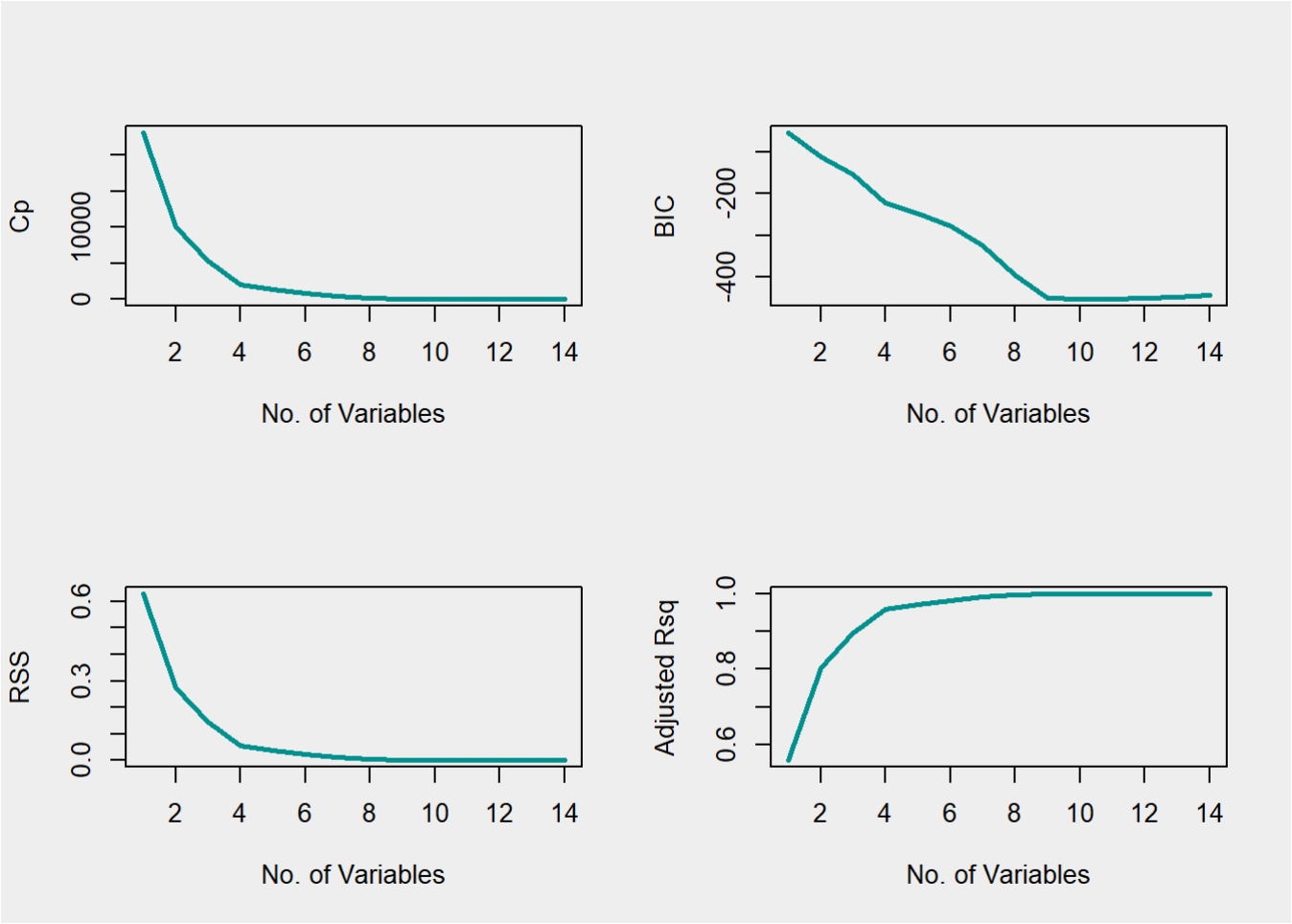
## Defining the model

```
ex_subset <- regsubsets(rating~., data = c_data[,2:16], nbest = 1, nvmax = 15, method = "exha
ustive")
ex_summary <- summary(ex_subset)
names(ex_summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

## Plotting Exhaustive Selection Measurements

```
par(mfrow = c(2,2),bg = '#EEEEEE')
plot(ex_summary$cp, xlab = "No. of Variables", ylab = "Cp", type = "l",col = "#00918E",lwd =
2.5)
plot(ex_summary$bic, xlab = "No. of Variables", ylab = "BIC", type = "l",col = "#00918E",lwd
 = 2.5)
plot(ex_summary$rss, xlab = "No. of Variables", ylab = "RSS", type = "l",col = "#00918E",lwd
 = 2.5)
plot(ex_summary$adjr2, xlab = "No. of Variables", ylab = "Adjusted Rsq", type = "l",col = "#0
0918E",lwd = 2.5)
```




# Finding the optimal model measures selection

```
which(ex_summary$cp == min(ex_summary$cp))
```

```
## [1] 12
```

```
which(ex_summary$bic == min(ex_summary$bic))
```

```
## [1] 11
```

```
which(ex_summary$rss == min(ex_summary$rss))
```

```
## [1] 14
```

```
which(ex_summary$adjr2 == max(ex_summary$adjr2))
```

```
## [1] 13
```

```
print(min(ex_summary$rss))
```

```
## [1] 0.001629464
```

```
print(max(ex_summary$adjr2))
```

```
## [1] 0.9986239
```

Linear model provides a MSE of 0.0000432 and R2 value of 0.9984. Whereas Forward Selection provides a MSE of 0.0000142 and Exhaustive Subset selection has a R2 of 0.9986.

Exhaustive subset selection is better than the Linear model and the forward selection. Exhaustive model has a R2 value of 99.86% which is close to 1. Almost 99.8% of data fit the model well.