```
#2.11

# Get Data
data = read.csv("ToyotaCorolla.csv")
head(data)
summary(data)
str(data)#structure of data

numeric_data <- data[,-c(1,2,8)]#get numeric variables only
str(numeric_data)
dim(numeric_data)

#a. Explore the data using the data visualization capabilities of R. Which
of the pairs among the variables seem to be correlated?
summary(numeric_data)
correlations <- cor(numeric_data)
correlations <- data.frame(correlations)
sum(is.na(correlations))#sum of NAs

correlations[is.na(correlations)]<-0#Replace NA with 0
sum(is.na(correlations))

out <- which(abs(correlations) > 0.80 & abs(correlations) != 1,
arr.ind=TRUE)
out

cols <- rownames(correlations)

out.df <- data.frame(out)

cbind(rownames(out.df),cols[out.df$col])#Highly Correlated

#Scatterplot the first two
plot(numeric_data$Age_08_04, numeric_data$Price)
plot(numeric_data$Mfg_Year, numeric_data$Price)


#b. We plan to analyze the data using various data mining techniques
described in future chapters. Prepare the data for use as follows:

#i. the dataset has two categorical attributes: Fuel Type and Metallic
# DDescribe how you would convert these to binary variables.

data$Fuel_Type
data$Metallic_Rim#already binary

library(caret)
to.one.hot <- data[,c('Fuel_Type','Doors')]

dummy <- dummyVars(" ~ .", data=to.one.hot)
final_df <- data.frame(predict(dummy, newdata=to.one.hot))
head(final_df)


#ii. Prepare the dataset (as factored into dummies) for data mining
techniques of supervised learning by creating partitions in R. Select all
the variables and use default values for random seed and partitioning
percentages for training (50%), validation (30%), test (20%) sets.
```

```
# (1) Encode Categorical Variables
data = read.csv("ToyotaCorolla.csv")
dummy <- dummyVars(" ~ .", data=data)
final_df <- data.frame(predict(dummy, newdata=data))

#training set
training.rows <- sample(row.names(final_df), dim(final_df)[1]*0.5)
train.df <- final_df[training.rows,]

#validation set
validation.rows <-
sample(setdiff(row.names(final_df),training.rows),dim(final_df)[1]*0.3)
valid.df <- final_df[validation.rows,]

#test set
test.rows <- setdiff(row.names(final_df), union(training.rows,
validation.rows))
test.df <- final_df[test.rows,]

length(training.rows) + length(validation.rows) + length(test.rows) ==
dim(final_df)[1]#확인차

#Describe the roles that these partitions will play in modeling.

#Training Partition: Used to build the models we are examining.

#Validation Partition: Used to assess the predictive performance of each
model so that you can compare models and choose the best one. In some
algorithms, the validation partition may be used in an automated fashion to
tune and improve the model.

#Test Partition(Holdout Partition): Used to assess the performance of the
chosen model with new data.
```