Setting up the workspace, reading data and loading the packages which will use this the next

```
library(knitr) # for changing workspace
library(stringr) # for using some regex functions
opts_chunk$set(root.dir = 'I:/R Data/141')
options(width = 110)
load("vehicles.rda")
vehicle = vposts     # make a copy of the orignial data
body = vehicle$body
```

For the Part 1 of this homework, we need to extract some different variables from the "body" column, this is the function that can extract different patterns from the data we want to. then from each questions, I don't need to write the same code again. This function will return a list that contains three lists, the values that matched pattern, the total number of the matched pattern and the posts that don't match the pattern. the matched values is what we want to get, the number of the matched pattern is I want to know how many matched pattern I get for the regular expression that I write, the last part is the posts that don't match the patter, I want to look at the posts that don't match pattern to check if there any other partterns exist but I don't include in my regular expression, then I can add it to reular expression to get more matched values. For dealing multiple return values, I decide to choose the first one, Because most time people will post year and price, at the first sentence, the first return value is correct for most time

```
value = function(regex,data)
{
  regex_fun = regex

  # It will store all the matched values in the posts return as a list
  result_fun= str_extract_all(data, ignore.case(regex_fun))

  # If there are multiple values in the list, get the first one out, if there is no
 value, give it NA
  value_fun = sapply(result_fun,function(x) {
       if (length(x)>0) x[1]
       else NA
       })

  # how many posts contain the pattern
  number = as.numeric(table(is.na(value_fun))[1])

  # the posts that don't have that pattern
  check = data[is.na(value_fun)]
  return(list(value_fun,number,check))

}
```

Extract the price being asked for the vehicle from the body column, if it is present, and check if it agrees with the actual price in the price column. This regular expression try to match the price that start with $, then in the parenthesis is any 0 to 3 digits end with "," "." or

nothing, and this pattern can appear 0 to any times, if apear 0 times that mean the price is under 1000, if appear once, that means the price is between $999,9999 to $1,000 or price under $1,000 with decimal,like $123.00, and so on. The last part is 1 to 3 digits.

```
# regular expression for price
price = "\\$?(\\d{0,3}[,.]?)*\\d{1,3}"
price_value = value(price,body)

# how many posts have price in body
price_value[2]

## [[1]]
## [1] 33565

head(price_value[[1]])

## [1] "2012" "2013" "2013" "2009" "2013" "2012"

# remove "$", "," or "." and then transform to numeric
price_value = as.numeric(gsub("\\$|,|.", "", price_value[[1]]))

# compare the price from "body" and from price column
table(price_value == vposts$price)

## < table of extent 0 >
```

I get 14410 posts have price in the body and only 8644 match the actual price.

Extract a Vehicle Identication Number (VIN) from the body, if it is present. We could use this to both identify details of the car (year it was built, type and model of the car, safety features, body style, engine type, etc.) and also use it to get historical information about the particular car. Add the VIN, if available,to the data frame. How many postings include the VIN?

Before I work on this question, I go to google to search the vin number pattern, This is the website that I follow with http://www.autocheck.com/vehiclehistory/autocheck/en/vinbasics The regular expression starts at VIN then follow with space or ":" or "-". The base on the website, the vin number has five parts, they are number, charactor, number, charactor and number again. I also limited the times that they can appear.

```
# regular expression for vin number
vin = "VIN[ :-]? ?\\d*\\w{1,7}\\d{1,3}\\w{1,5}\\d{4,8}"
vin_num = value(vin,body)

# how many posts have price in body
vin_num[2]

## [[1]]
## [1] 5622

head(vin_num[[1]])
```

```
## [1] "VIN: 2G1FT1EW1C9106920" "VIN: 2GNFLNEK7D6324351" "VIN: 1N4AL3AP5DN569028" "
VIN: JNKCY01F29M851648"
## [5] "VIN: JN1CV6AR2DM763007" "VIN: 2HNYD2H20CH537485"
```

I only get 5622 matchs, but on the Piazz, some students got around 8000 matches, rough way to get the vin number, which is any string that has charactors and numbers but the length is 17, but not contains I, O.

```
# rough way to get the vin number
vin = "[A-HJ-NPR-Z0-9]{17}"
vin_num1 = value(vin,body)

# how many posts have price in body
vin_num1[2]
```

```
## [[1]]
## [1] 8924
```

```
head(vin_num1[[1]])
```

```
## [1] "2G1FT1EW1C9106920" "2GNFLNEK7D6324351" "1N4AL3AP5DN569028" "JNKCY01F29M8516
48" "JN1CV6AR2DM763007"
## [6] "2HNYD2H20CH537485"
```

```
# look it the value
 VIN = data.frame(vin_num1[[1]])
#View(VIN)
 vposts$vin = vin_num[[1]]
```

now I get 8924 matchs. Since it is a rough match, so I looked at the first 1000 values that I get, there are some wrong values, like "CEEEEEEEEEEEEEEE", "peterfullerusedau", or "bestusedcarmarket" etc. So even this one gives me more match, but it also contains some wrong values, so I decide to use the first match

Extract phone numbers from the body column, and again add these as a new column. How many posts include a phone number? This regular expression matchs any 3 digits that in the parenthese or not, then between the first three number to the next three number, it can be space, "-" or double space or nothing, after that is anther three numbers conenct with "-", space or nothing with rest four numbers.

```
# regular expression for vin number
phone = "\\(?\\d{3}\\)?[ ]?[ -]?\\d{3}-? ?\\d{4}"
phone_num = value(phone,body)

# how many posts have price in body
phone_num[2]
```

```
## [[1]]
## [1] 16236
```

```
phone_num[[1]][200:205]
```

```
## [1] "(508) 205-1046" NA                "978-228-6000"   "(508) 205-1046" "(508) 2
13-4680" "508-844-8124"
```

I get 16236 matches.

Extract email addresses from the body column, and again add these as a new column. How many posts include an email address?

Since there no pattern for email, the only pattern that we can use is @ and it end with com, net, org, edu or gov, so I use a rough way to match any alphanumeric characters with any punctuations then it follows by @ and any alphanumeric characters with any punctuations.

```
# regular expression for vin number
email = "[[:alnum:]|[:punct:]]+@[[:alnum:]|[:punct:]]+?\\.(com|net|org|edu|gov){1}"

email_add = value(email,body)

# how many posts have price in body
email_add[2]

## [[1]]
## [1] 107

head(email_add[[1]])

## [1] NA NA NA NA NA NA

# grep the posts have "email"
email = body[grepl("email", body, ignore.case = TRUE)]

# how many posrs have email
length(email)

## [1] 2174

# look at some posts have "email"
email[1]

## [1] "\n        WOW!! Only 48,873 miles on this economical Saturn L200. Nicely eq
uipped including power windows, mirrors and locks this sedan is a lot of car for $3
880. Thoroughly inspected and serviced you'll never have to worry about tires, brak
es or other maintenance items when you purchase a vehicle from us. Low mileage beau
ties like this don't last. Call or email today to set a time for a showing.\n    "
```

I only get 107 matches, so few people wrote email in their post. Then I look at how many posts mention the email, I get 1161 of them mention it, still less compare to the price, phone number and vin number that I get from previous questions, then I look at one post which has "email" in the body. It has "email" and "call", but I don't see the email address and phone number. So I go to craigslist and open one post, I see on the left corner has a reply link, the email is in the reply, so now it makes sense that I get few email address

Find the year in the description or body and compare it with the value in the year column.
This regular expression matchs all the 4 digits that start with 19xx or 20xx.

```
# regular expression for year
year = " ?(19|20)\\d{2} ?"
year_b = value(year,body)

# how many posrs have year
year_b[2]

## [[1]]
## [1] 25429

head(year_b[[1]])

## [1] "2012 " "2013 " "2013 " "2009 " "2013 " "2012 "

# compare to the year column
year_num = as.numeric(year_b[[1]])
table(year_num == vposts$year)

##
## FALSE  TRUE
##  2579 22850
```

I get 25429 matches and 22850 is same as the value in the year column

Determine the model of the car, e.g., S60, Boxter, Cayman, 911, Jetta. This includes correcting mis-spelled or abbreviated model names. You may find the agrep() function useful. You should also use statistics, i.e., counts to see how often a word occurs in other posts and if such a spelling is reasonable, and whether this model name has been seen with that maker often. When doing these questions, you will very likely have to iterate by developing a regular expression, and seeing what results it gives you and adapting it. Furthermore, you will probably have to use two or more strategies when looing for a particular piece of information. This is expected; the data are not nice and regularly formatted.

I look at he vposts data, I feel like find model from title is easier from body column, because body column contains a lot of other informations. In the title column, it starts with the year, and then is the maker and model. So I grep the third position from the title column. For correcting mis-spelled or abbreviated model names, I use Google refine, because I saw someone asked on Piazz, can we use google refine to correct the mi-spelling, and Ducan said "Yep"(post @1104), so I save the data as .csv and load in to Google refine to correct them.

```
# get the model from the "title" column
title = "\\d{2,4} ([A-z]+[:punct:]?[A-z]+) ([A-z0-9]+)"
new_title  = value(title,vposts$title)
model = gsub(".*\\d{2,4} ([A-z]+) ([A-z0-9]+).*" , "\\2", new_title[[1]])
model = casefold(model, upper = F)

# give the frequncy for all spelling
```

```
frq = table(model)

# save it as csv (won't do it again, since I already saved it)
#vposts$model = model
#write.csv(vposts[,c(1,28)], file = "model.csv")

# read the sort data from google refine
model_google = read.csv("model-google.csv")

# redefine model column
vposts$model = model_google[,3]
```

Pick two models of cars, each for a different car maker, e.g., Toyota or Volvo. For each of these, separately explore the relationship between the price being asked for the vehicle, the number of miles (odometer), age of the car and condition. Does location (city) have an effect on this? Use a statistical model to be able to suggest the appropriate price for such a car given its age, mileage, and condition. You might consider a linear model, k-nearest neighbors, or a regression tree. You need to describe why the method you chose is appropriate? what assumptions are needed and how reasonable they are? and how well if performs and how you determined this? Would you use it if you were buying or selling this type of car?

For this question, I use linear regression to model the data. Linear regression model is a simple and onvenience way to model the data and do the prediction. For the data we have, the response variable (price) is numeric, the explortory variables(odometer, age and condition) are mixed with numeric and categorical data, so multiple linear regression might be a good model for this data. I also plot the price with other three variables separately to see the relation between them. After the log transfermation. The relations between price and odometer, and price with age are very linear, so I decide to use multiple linear regression. The assumptions for multiple linear regression are linearity, normality, equal variance. The first assumption is done by plot the price with other three variables separately. for the second one I did QQ plot, the last one I did scale-location plot. Before I used the regression model for the data, I checked those three assumptions, the plots showed the model satisfied those assumptions. For determined how well it performs, first I use cross validation to get training data and test data. I plot the true price for test data and predicted price for test data in the same plot to see how good the model predicts price. From the plot I think the model doesn't predict perfectly, but most of predict price are very close to the true price, so I think the model performs good. I probabily won't use this model for buying a car. Bacause this model might violate the regression assumption by omitting related varibles. In this model, we only contain the odometer, age and condition, Like "title status", salvage car always cheaper than clean tital car. I also searched on Google "what will affect used car price". There are some varibales affect the price rather than odometer, age and condition.

Clean and sort data, when I did the homework1, I remember in the data, it has a lot of different conditions, so I use the code from the homework1 solution for question 16 to sort the condition, named it new_cond.

```
# add age columns to the data
vposts$age = 2015- vposts$year
```

```
# Print out conditions so we can cut and paste them into smaller
conditions = levels(vposts$condition)
conditions = sprintf('"%s",\n', conditions)

# Define new categories. (code from homework1 solution for question 16)
new_cats = list(
  excellent = c("excellent"),
  good = c("good", "very good"),
  "like new" = c("like new", "mint", "new", "pre owned", "pre-owned", "preowned", "
preownes"),
  used = c("0used", "used"),
  fair = c("fair", "nice", "nice teuck"),
  salvage = c("complete parts car, blown engine", "front side damage", "hit and run
 :( gently",
              "muscle car restore", "needs bodywork", "needs restoration!", "needs
restored",
              "needs total restore", "needs work", "needs work/for parts", "nice ro
lling restoration",
              "not running", "parts", "project", "project car", "rebuildable projec
t", "restoration",
              "restoration project", "restore", "restored", "salvage", "rough but r
uns"),
  other = c("207,400", "ac/heater", "carfax guarantee!!", "certified", "honnda", "s
uperb original" )
)

# Convert conditions to new categories.
vposts$new_cond = vposts$condition
levels(vposts$new_cond) = c(levels(vposts$new_cond), "other")

for (i in seq_along(new_cats)) {
  new_cat = names(new_cats)[[i]]
  vposts$new_cond[vposts$new_cond %in% new_cats[[i]]] = new_cat
}

vposts$new_cond = factor(vposts$new_cond)
```

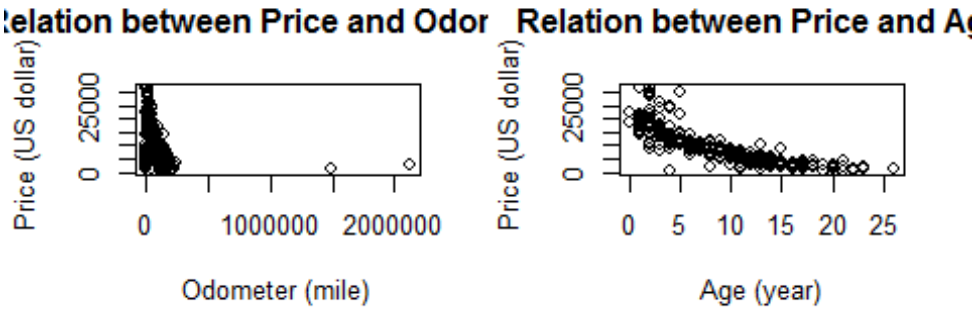
For the two models, I decide to choose two models that have more obervations

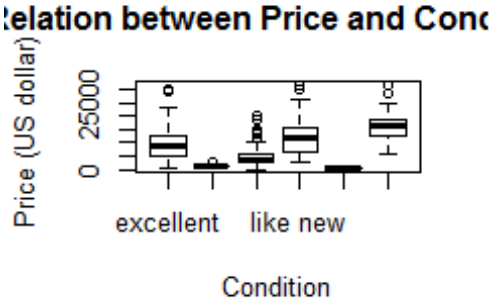
```
sort(table(vposts$model), decreasing = TRUE)[1:5]

##
## accord  camry  civic  grand altima
##    735    721    715    702    570
```

I decide to do "civic" and "camry"

Start with "camry", I first take out any NA and plot the price with other three variables separately to see the relation between them

```r
# get the  camry posts
camry = vposts[vposts$model == "camry",]
camry = camry[!is.na(camry$model),]

# take out the NAs
remove_na = function(data)
{
  data = data[!is.na(data$odometer),]
  data = data[!is.na(data$price),]
  data = data[!is.na(data$new_cond),]
}

camry = remove_na(camry)

# plot the price with odometer, age and condition separately
plot_relation = function(data)
{
plot(data$odometer, data$price, ylab= "Price (US dollar) ", xlab = "Odometer (mile)
", main = "Relation between Price and Odometer")
plot(data$age, data$price, ylab = "Price (US dollar) ", xlab = "Age (year)", main =
 "Relation between Price and Age")
plot(factor(data$new_cond), data$price, ylab= "Price (US dollar) ", xlab = "Conditi
on", main = "Relation between Price and Condition")
}

par(mfrow=c(2,2))
plot_relation(camry)
```





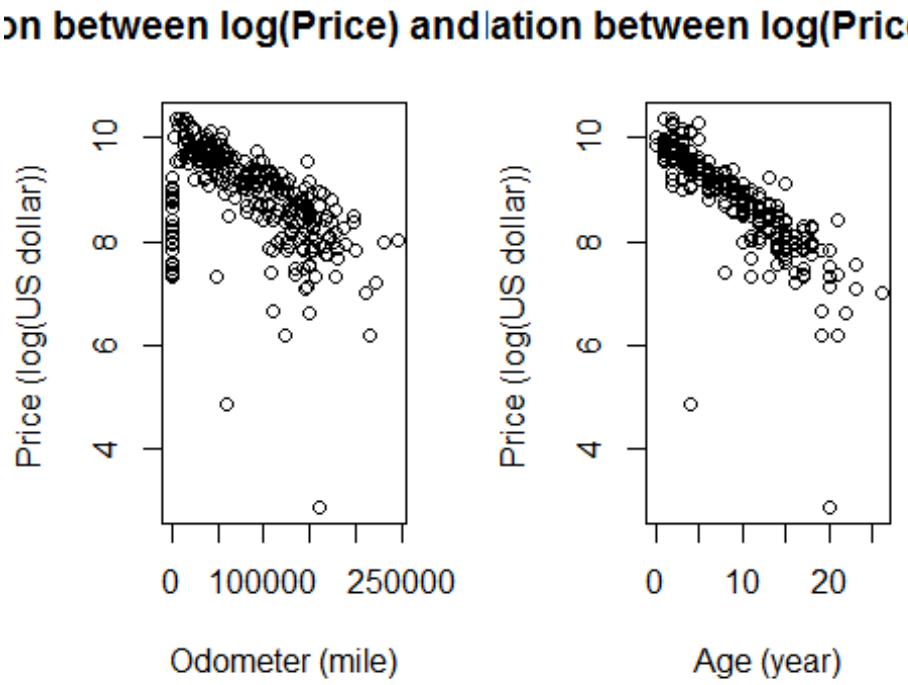
From the plot, I think there is expinatial relation between price and odometer, price and age. I do a log transfermation. Also from the odometer plot, I saw there are two outliers and some value looks like zero, so I take off the outliers, zero value and plot them again

```
# take out the outliers
# see the 50 largest odometer
tail( sort(camry$odometer), 50 )
```

```
##  [1]  150000  150000  150000  150000  151200  152000  152110  153000  154000  15
4000  154000  154000  154194
## [14]  155000  155675  157000  160000  160000  161000  161334  164150  167500  16
8000  168000  170000  170000
## [27]  172000  172144  173000  173425  174753  175000  176807  177000  178000  18
0000  180000  180000  187398
## [40]  189000  196438  197767  200000  211000  215022  221158  230000  244976 149
0000 2120000
```

```
# I take odometer that no more than 30,000 and zero value
camry = camry[camry$odometer<300000 & camry$odometer != 0,]

#log trasformation
log_plot = function(data)
{
  plot(data$odometer, log(data$price), ylab= "Price (log(US dollar)) ", xlab = "Odo
meter (mile)", main = "Relation between log(Price) and Odometer")
  plot(data$age, log(data$price), ylab= "Price (log(US dollar)) ", xlab = "Age (yea
r)", main = "Relation between log(Price) and Age")
```

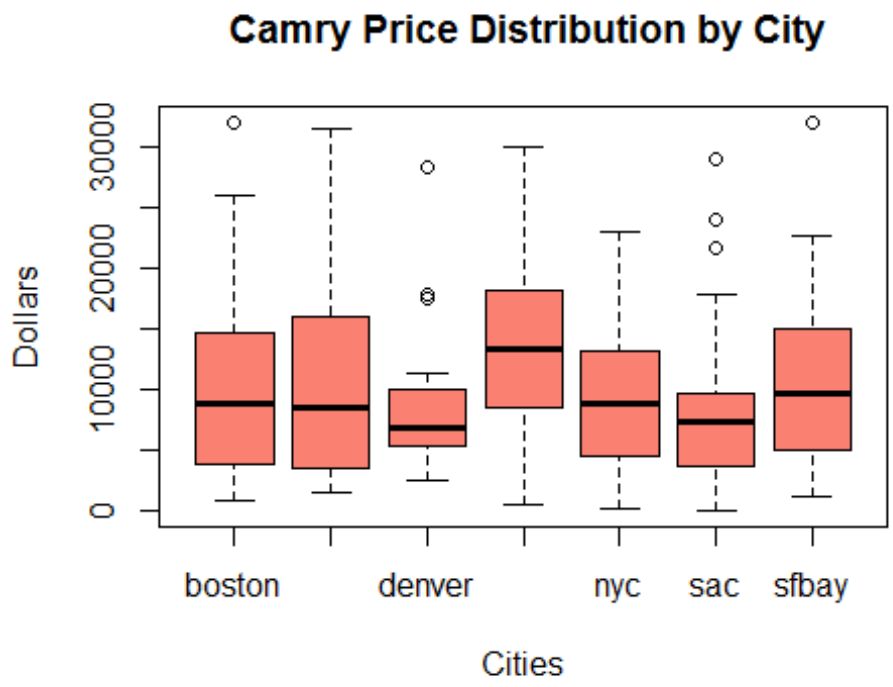
```
}
par(mfrow=c(1,2))
log_plot(camry)
```



After lot transfermation, the relation between price and other two variables look like linear relation. so I decide to fit the data with linear regression. Since condition is not a numeric data, I set it as a dummy varible.

In order to see if the location has affect on the price, I draw a boxplot to how different the price cross the different cities

```
# split price by city
pri_by_city = split(camry$price, camry$city)

# draw boxplot
par(mfrow=c(1,1))
boxplot(pri_by_city, col = "salmon")
title(" Camry Price Distribution by City", ylab = "Dollars" , xlab = "Cities")
```



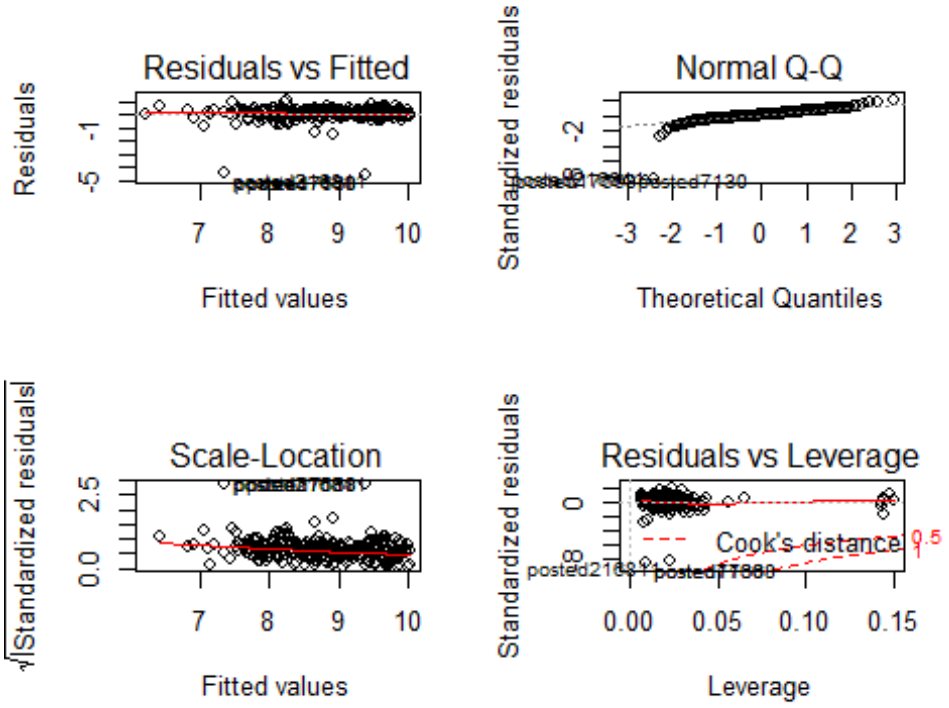
From the plot, the price for different cities is different, but not too much, the only main difference is the price in Las Vegas, the price is higher than all of other cities

Before using the model, I chcek the assumptions.

```
# fit a regression model
fit_model = lm(log(price)~odometer+age+factor(new_cond), data = camry)

# diagnostics for model
par(mfrow=c(2,2))
plot(fit_model)
```

```
## Warning: not plotting observations with leverage one:
##   252
```

```
## Warning: not plotting observations with leverage one:
##   252
```



For Residuals vs Fitted plot, the line is horizontial, so the assumption hold. For the Q-Q plot, most potins are on the straight line, the data satisfied the normal assumption. For Scale-location plot, the line is kind of horizontial, so the assumption hold. For Residuals vs Leverage are the points fall inside the red dotted line, so the assumption hold.

Now look at the significant of coefficients, whole regression line, and R^2

```
summary(fit_model)
```

```
##
## Call:
## lm(formula = log(price) ~ odometer + age + factor(new_cond),
##     data = camry)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5404 -0.1226  0.0494  0.2191  1.0165
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 9.899e+00  7.369e-02 134.330   <2e-16 ***
## odometer                   -1.160e-06  6.284e-07  -1.846   0.0659 .
## age                        -1.074e-01  7.102e-03 -15.123   <2e-16 ***
## factor(new_cond)fair       -4.408e-01  2.239e-01  -1.968   0.0499 *
## factor(new_cond)good       -2.165e-01  8.613e-02  -2.514   0.0124 *
## factor(new_cond)like new    1.433e-01  8.329e-02   1.721   0.0862 .
## factor(new_cond)salvage    -1.394e+00  5.482e-01  -2.543   0.0115 *
```

```
## factor(new_cond)used      2.237e-01  1.034e-01   2.164   0.0312 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5397 on 311 degrees of freedom
## Multiple R-squared:  0.6742, Adjusted R-squared:  0.6669
## F-statistic: 91.95 on 7 and 311 DF,  p-value: < 2.2e-16
```

```
anova(fit_model)
```

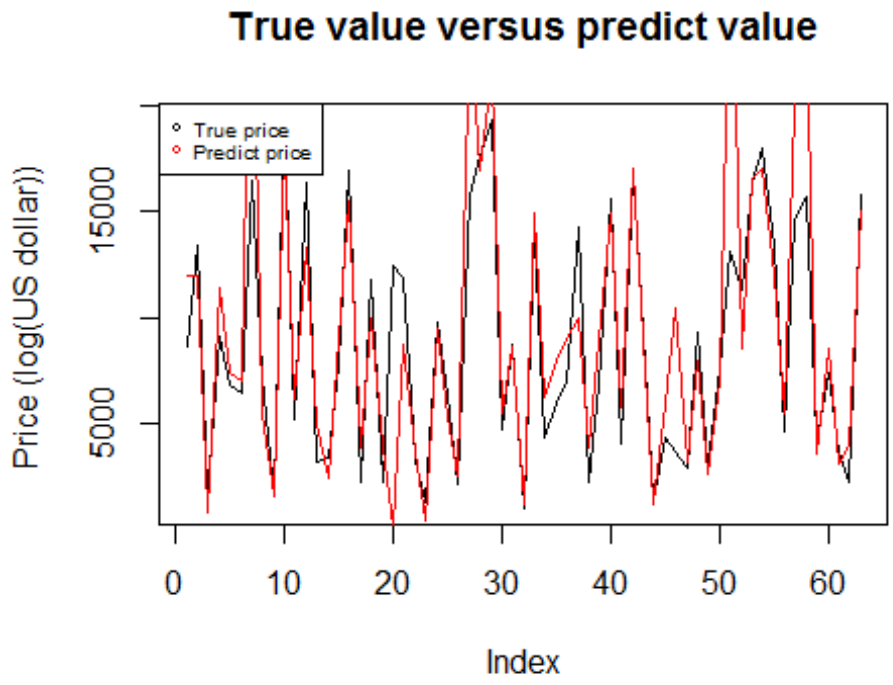
```
## Analysis of Variance Table
##
## Response: log(price)
##                   Df  Sum Sq Mean Sq  F value    Pr(>F)
## odometer           1  67.793  67.793 232.7174 < 2.2e-16 ***
## age                1 113.164 113.164 388.4669 < 2.2e-16 ***
## factor(new_cond)   5   6.547   1.309   4.4946 0.0005709 ***
## Residuals        311  90.597   0.291
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

All the coefficients and whole regression line significant at 0.1 significant level, the R^2 is 0.6742 which is acceptable.

Use cross validation to test the model. I split data into two groups, one is test dataset, one is train dataset. the test dataset contain one fifth data, the train dataset contains four fifth data. I use train dataset to fit the linear model, and use that model to predict the price for test dataset. In order to see how good the model predicts price, First I use MSE to see how good it is, but the value is very big, because the most price are higher than $8000, so the MSE is very large, it seems like the fit is not good. Then I draw a plot, the black line is the true price, and the red line is the predict price.

```
# use cross validation to test the model
test_model = function(data)
{
  # split train and test data set
  index = sample(1:nrow(data), nrow(data)/5)
  train = data[-index,]
  test = data[index,]

  # fit the model
  fit_model = lm(log(price)~odometer+age+factor(new_cond), data = train)

  # format test data for prediction
  testdata = data.frame(with(test,cbind(odometer, age)))
  testdata$new_cond = test$new_cond

  # get the predict price
  pred_price = exp(predict(fit_model, testdata, type="response") )

  # plot the true and predict price
```

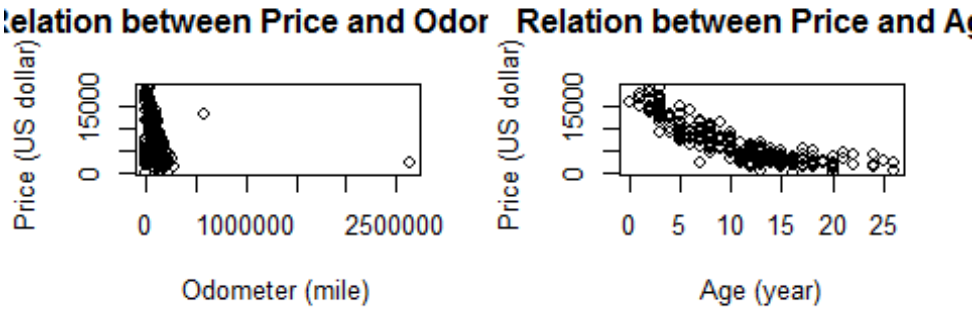
```
  plot(pred_price, type = "l", ylab = "Price (log(US dollar)) ", xlab = "Index", ma
in = "True value versus predict value")
  points(test$price, col = "red", type = "l")
  legend("topleft", legend = c("True price", "Predict price"), col= c("black", "red
"), pch=1, cex = 0.6)

}


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

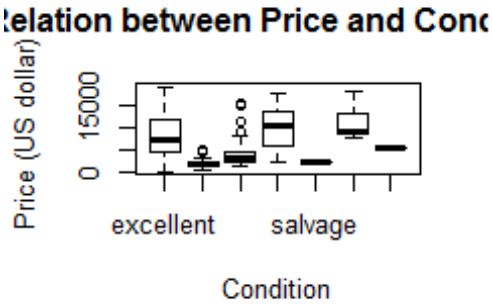


From the plot we can see two different lines very close to each other, so the predict values are very close to the true values. The model does a good job.

Do the same thing again for civic

```
# get the  camry posts
civic= vposts[vposts$model == "civic",]

# take out the NAs
civic = remove_na(civic)

# plot the price with odometer, age and condition separately
par(mfrow=c(2,2))
plot_relation(civic)
```

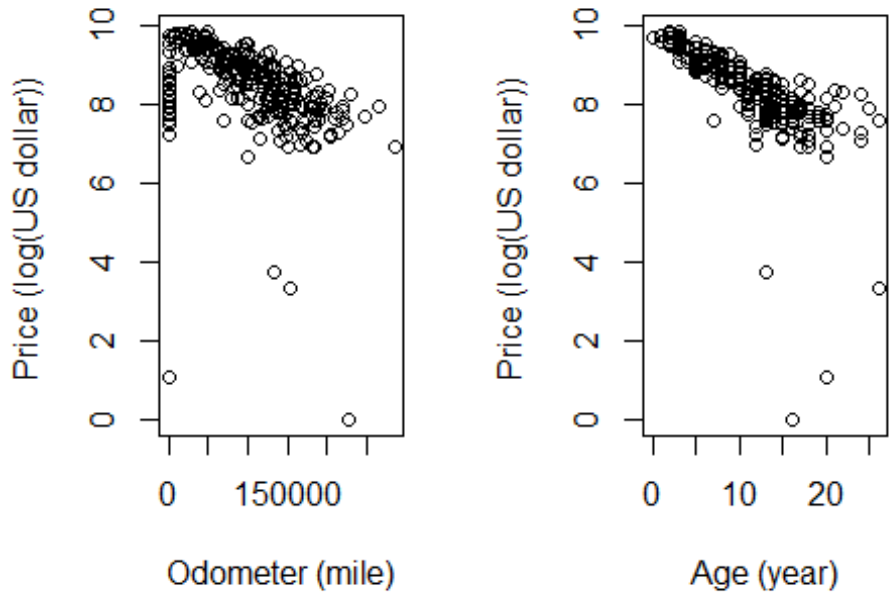




The relations between price, odometer and age are also looked like exponential, and also I can see there are two outliers from odometer plot. So I take out the outliers and make a log transfermation.

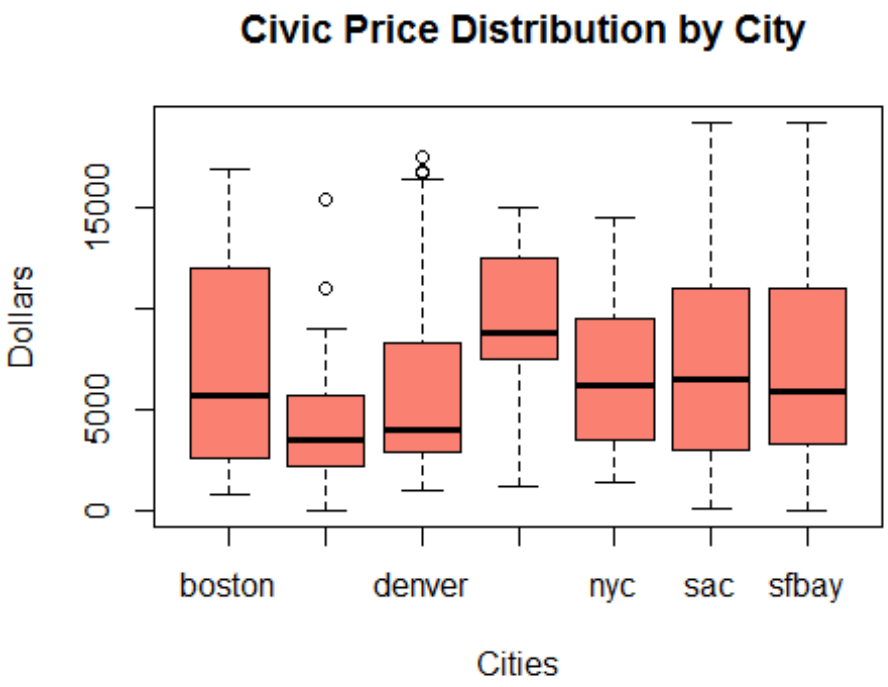
```
# take out the outliers
# I take odometer that no more than 50,000
civic = civic[civic$odometer<500000 | civic$odometer == 0,]

#log trasformation
par(mfrow=c(1,2))
log_plot(civic)
```

ɔn between log(Price) and lation between log(Price) a



After lot transfermation, the relation between price and other two variables look like linear relation. so I decide to fit the data with linear regression. Since condition is not a numeric data, I set it as a dummy varible.

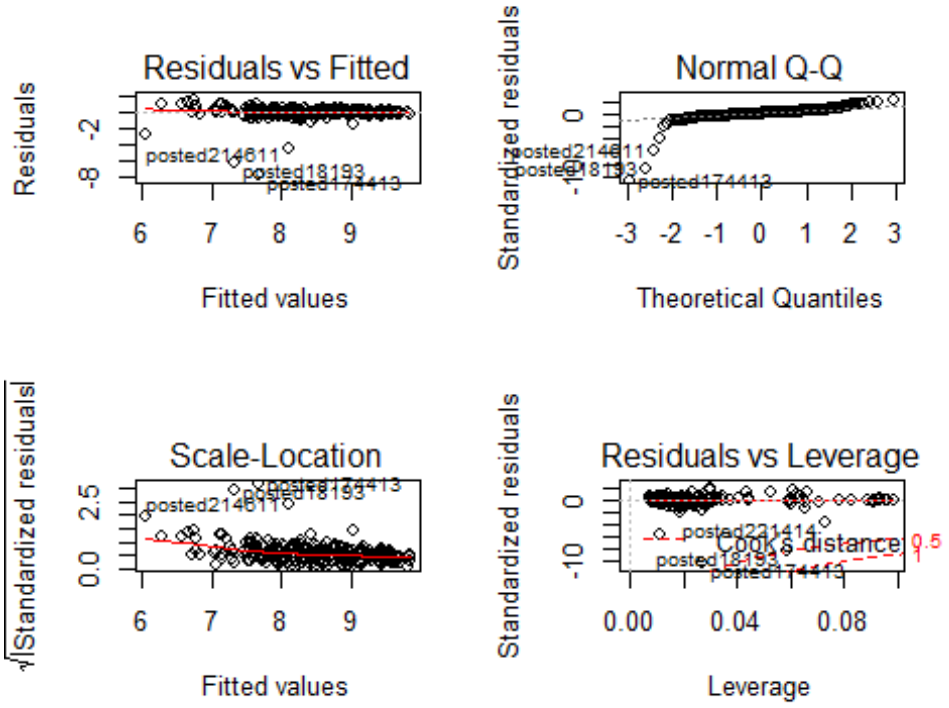
```
# split price by city
pri_by_city = split(civic$price, civic$city)

# draw boxplot
par(mfrow=c(1,1))
boxplot(pri_by_city, col = "salmon")
title(" Civic Price Distribution by City", ylab = "Dollars" , xlab = "Cities")
```



From the plot, the price for different cities are almost same, the only main difference is the price in Las Vegas, same like camry

Before using the model, I chcek the assumptions.

```
# fit a regression model
fit_model = lm(log(price)~odometer+age+factor(new_cond), data = civic)

# diagnostics for model
par(mfrow=c(2,2))
plot(fit_model)

## Warning: not plotting observations with leverage one:
##   192, 285

## Warning: not plotting observations with leverage one:
##   192, 285
```



For Residuals vs Fitted plot, the line is horizontial, so the assumption hold. For the Q-Q plot, most potins are on the straight line, the data satisfied the normal assumption. For Scale-location plot, the line is kind of horizontial, so the assumption hold. For Residuals vs Leverage there is one outlers which is posted 18193.

Now look at the significant of coefficients, whole regression line, and R^2

```
# remove outlier
civic = civic[!rownames(civic) == "posted18193",]

# fit regression model agian
fit_model = lm(log(price)~odometer+age+factor(new_cond), data = civic)

summary(fit_model)
```

```
##
## Call:
## lm(formula = log(price) ~ odometer + age + factor(new_cond),
##     data = civic)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.6771 -0.1132  0.0403  0.2294  1.4443
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               9.831e+00  9.242e-02 106.362  < 2e-16 ***
```

```
## odometer                 -2.013e-06  7.487e-07  -2.689  0.00758 **
## age                      -1.060e-01  9.192e-03 -11.535  < 2e-16 ***
## factor(new_cond)fair     -5.399e-01  1.814e-01  -2.976  0.00316 **
## factor(new_cond)good     -2.438e-02  1.000e-01  -0.244  0.80764
## factor(new_cond)like new  8.282e-02  1.062e-01   0.780  0.43606
## factor(new_cond)salvage   3.754e-01  6.609e-01   0.568  0.57045
## factor(new_cond)used      2.132e-01  2.048e-01   1.041  0.29889
## factor(new_cond)other     4.945e-01  6.540e-01   0.756  0.45017
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6495 on 292 degrees of freedom
## Multiple R-squared:  0.5812, Adjusted R-squared:  0.5697
## F-statistic: 50.65 on 8 and 292 DF,  p-value: < 2.2e-16
```

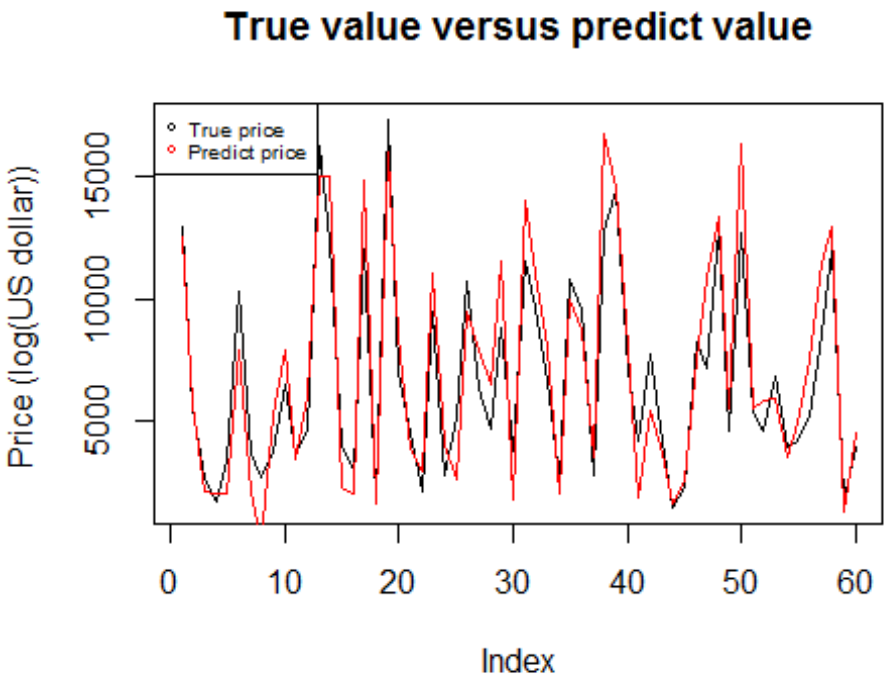
```
anova(fit_model)
```

```
## Analysis of Variance Table
##
## Response: log(price)
##                   Df  Sum Sq Mean Sq  F value Pr(>F)
## odometer           1  79.283  79.283 187.9609 <2e-16 ***
## age                1  86.254  86.254 204.4888 <2e-16 ***
## factor(new_cond)   6   5.383   0.897   2.1269 0.0503 .
## Residuals        292 123.167   0.422
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Like new, salvage, used, other, good, are not significant at 0.1 siginificant level, whole regression line significant at 0.1 significant level. R^2 is 0.5812 also acceptable.

Now the relations between price, odometer and age are also looked like linear. Fit the model and test it

```
# use cross validation to test the model do it twice
par(mfrow=c(1,1))
test_model(civic)
```



From the plot we can see two different lines very close to each other, so the predict values are very close to the true values. The model does a good job.