Build multiple linear regression models by using R [100]

Data: AutoMPG, https://archive.ics.uci.edu/ml/datasets/auto+mpg, for your convenience, you can use the attached AutoMPG.csv in which the data have been formatted in csv, and the variable "name" has been removed.

Instructions: build multiple linear regression models on this data by using mpg as the dependent variable, and other variables as independent ones. You should build multiple models and evaluate them to tell which one(s) are the best.

Hints:

- There are some missing values, you should solve this issue before any experiments. Note: you cannot simply remove them in this assignment
- All the variables are numerical variables. However, the discrete variables in this data, including cylinders, year and origin, are better to be treated as categorical variables.
- How to convert these categorical variables to dummy variables in the linear regression models? Yes, you can create N-1 dummy variables if there are N values in such a variable. But you may obtain several dummy variables in the value of N is large. In the real-practice, you can figure out different ways to create groups. For example, take the variable year for example, you can simply calculate the median value, and use the median value to put all the values into two groups. Take cylinders for example, you can find q1, q2, q3 and they can help you create four groups: group1 (min to q1), group2 (q1 to q2), group3 (q2 to q3), group4 (q3 to max)

Submissions:

- You should run your R codes, paste the R codes and outputs (or snapshots) in this WORD document; explain your operations step by step (for example, why you want to conduct this specific step), interpret the outputs and provide necessary explanations
- Given the final conclusions at the end of the submission

Note: we are not going to give you more instructions. This is a real-world practice which is similar to your final project. You should use the data analytic skills to analyze the real-world data, build predictive models, seek the optimal solution, and explain your findings.

Step 1: Understand the data

https://archive.ics.uci.edu/ml/datasets/auto+mpg

Attribute Information:

- mpg: continuous
- cylinders: multi-valued discrete
- displacement: continuous
- horsepower: continuous
- weight: continuous
- acceleration: continuous
- model year: multi-valued discrete
- origin: multi-valued discrete

For discrete variables, you can directly consider them as numerical variables, or you can convert them into categories to be used in the linear regression. It depends on the data, the number of values in the variable, as well as the experiments. There are no fixed rules in the preprocessing. In the real-world practice, you should try different preprocessing to figure out which method is the best.

Step 2: Load data, prepare the data for model fitting

```
> dataset = read.csv('AutoMPG.csv')
> head(dataset)
  mpg cylinders displacement horsepower weight acceleration year origin
1  18         8          307        130   3504         12.0   70      1
2  15         8          350        165   3693         11.5   70      1
3  18         8          318        150   3436         11.0   70      1
4  16         8          304        150   3433         12.0   70      1
5  17         8          302        140   3449         10.5   70      1
6  15         8          429        198   4341         10.0   70      1
> mpg=dataset$mpg
> cyl=dataset$cylinders
> disp=dataset$displacement
> hp=dataset$horsepower
> wt=dataset$weight
> acc=dataset$acceleration
> yr=dataset$year
> org=dataset$origin
```

2.1) Deal with missing values

You can manually fill in the values or use Excel, or use R codes.

In this example, I use the variable mean value to fill in the missing values

#missing values replace by mean

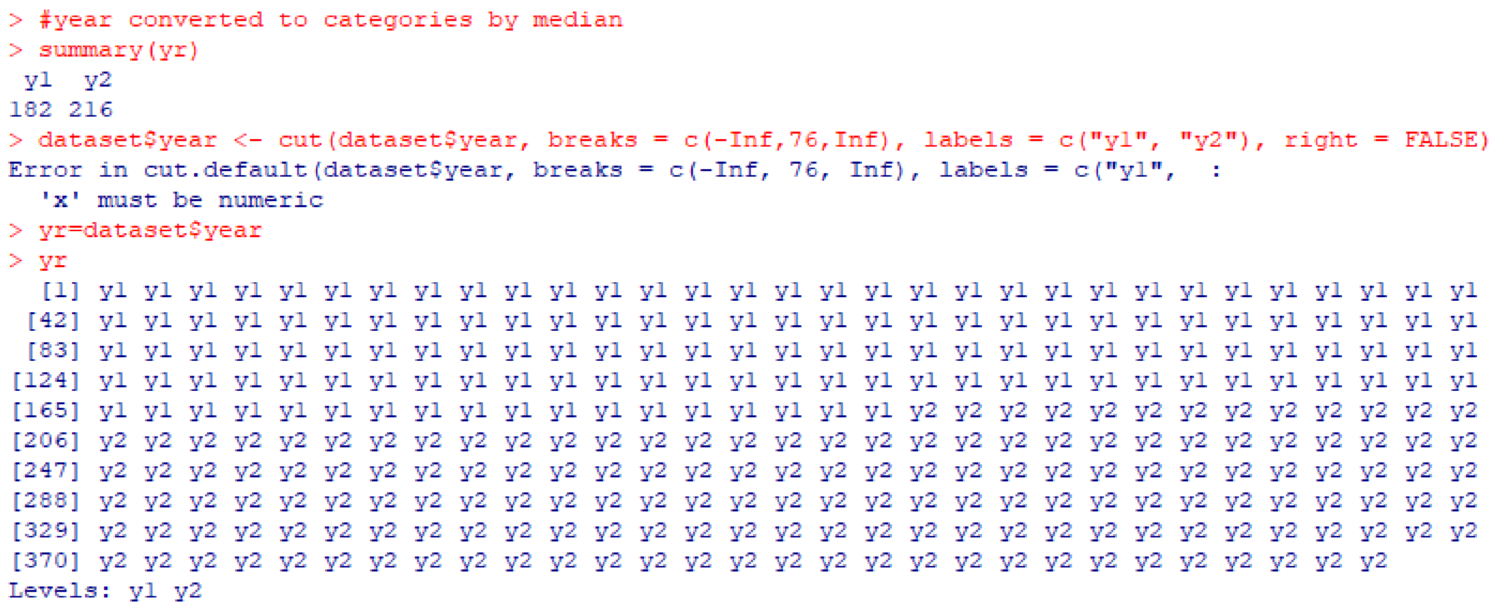
dataset$horsepower = ifelse(is.na(dataset$horsepower),ave(dataset$horsepower, FUN = function(x) mean(x, na.rm = TRUE)),dataset$horsepower)

hp=dataset$horsepower

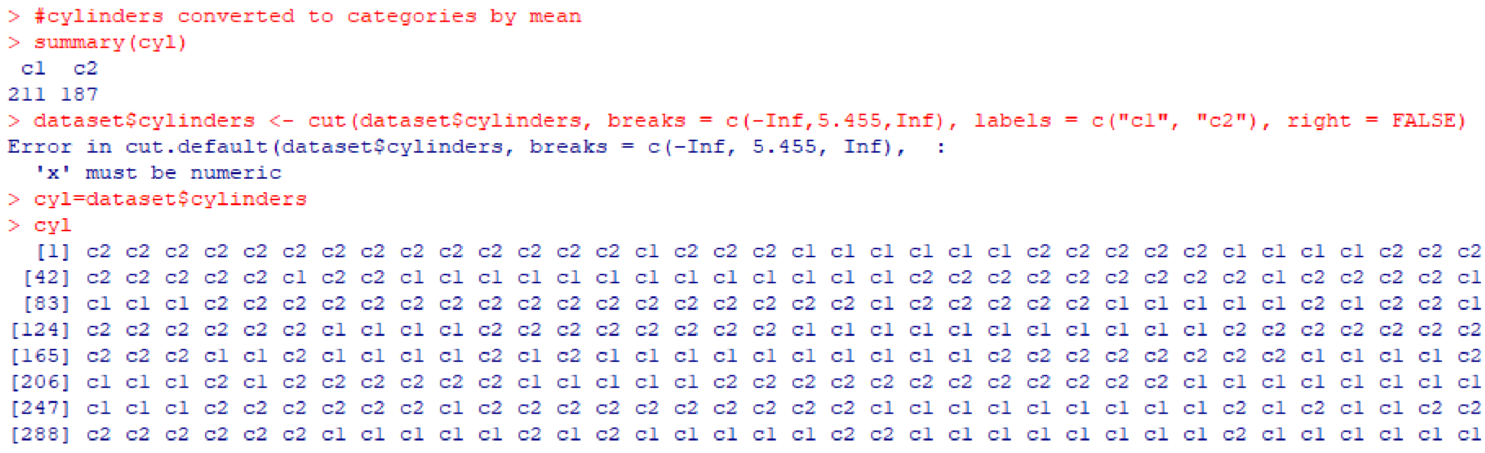
2.2) Deal with Discrete variables

- Cylinders ➔ values in scale [3, 8]
- Year ➔ scale [70, 82]
- Original ➔ scale [1, 3]

Simply, I decide to put them into different groups by using the cut function



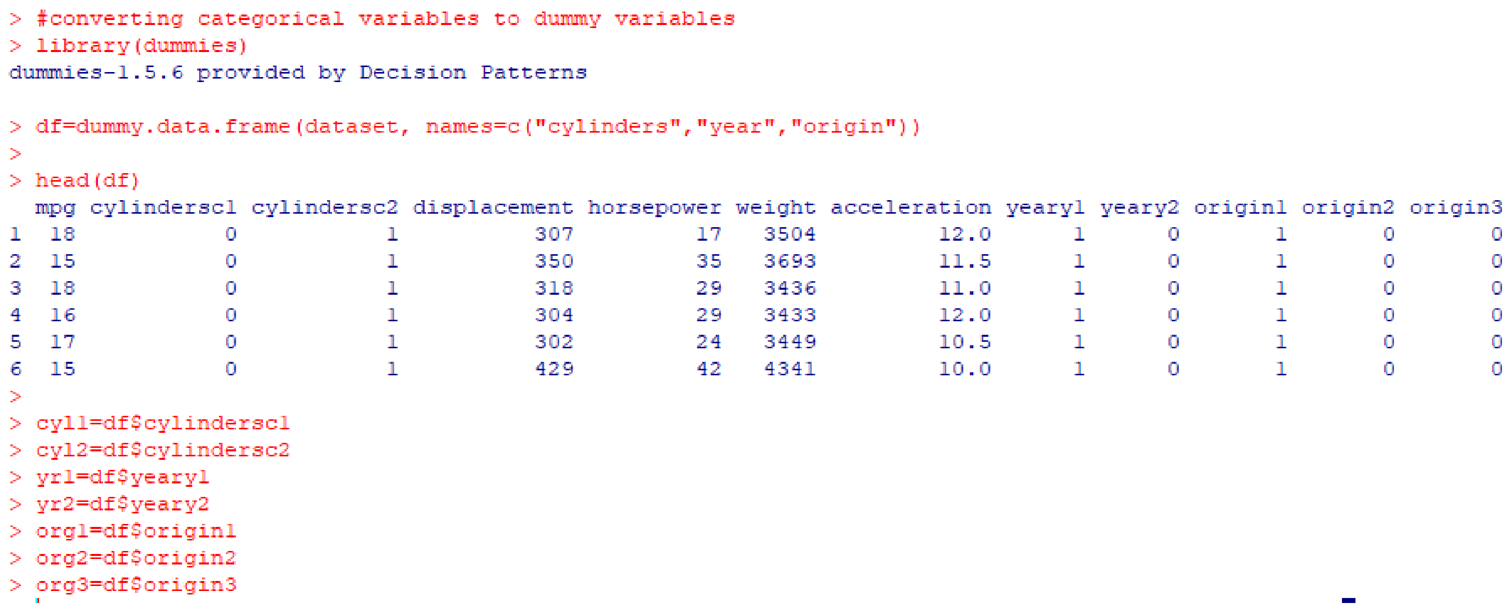
```
> #year converted to categories by median
> summary(yr)
 y1  y2 
182 216 
> dataset$year <- cut(dataset$year, breaks = c(-Inf,76,Inf), labels = c("y1", "y2"), right = FALSE)
Error in cut.default(dataset$year, breaks = c(-Inf, 76, Inf), labels = c("y1",  : 
  'x' must be numeric
> yr=dataset$year
> yr
  [1] y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1
 [42] y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1
 [83] y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1
[124] y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1
[165] y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y1 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2
[206] y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2
[247] y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2
[288] y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2
[329] y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2
[370] y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2 y2
Levels: y1 y2
```



```
> #cylinders converted to categories by mean
> summary(cyl)
 c1  c2
211 187
> dataset$cylinders <- cut(dataset$cylinders, breaks = c(-Inf,5.455,Inf), labels = c("c1", "c2"), right = FALSE)
Error in cut.default(dataset$cylinders, breaks = c(-Inf, 5.455, Inf),  :
  'x' must be numeric
> cyl=dataset$cylinders
> cyl
  [1] c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c1 c2 c2 c2 c1 c1 c1 c1 c1 c1 c2 c2 c2 c2 c2 c1 c1 c1 c1 c2 c2 c2
 [42] c2 c2 c2 c2 c2 c1 c2 c2 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c2 c2 c2 c2 c2 c2 c2 c2 c2 c1 c2 c2 c2 c2 c1
 [83] c1 c1 c1 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c1 c2 c2 c2 c2 c2 c1 c1 c1 c1 c1 c2 c1 c2 c2 c1
[124] c2 c2 c2 c2 c2 c2 c1 c1 c1 c1 c2 c2 c2 c2 c2 c2 c2 c2 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c2 c2 c2 c2 c2 c2 c2
[165] c2 c2 c2 c1 c1 c2 c1 c1 c1 c1 c2 c1 c2 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c2 c2 c2 c2 c2 c2 c2 c2 c1 c1 c1 c1 c2
[206] c1 c1 c1 c2 c1 c2 c2 c2 c2 c2 c2 c1 c1 c1 c1 c1 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c1 c1 c1 c1 c1 c1 c1 c1
[247] c1 c1 c1 c2 c2 c2 c2 c2 c2 c1 c2 c2 c2 c2 c2 c2 c2 c2 c2 c2 c1 c1 c1 c1 c1 c1 c1 c1 c1 c2 c1 c2 c1 c1 c2 c2
[288] c2 c2 c2 c2 c2 c2 c1 c1 c1 c1 c1 c2 c1 c2 c1 c1 c1 c1 c1 c2 c2 c1 c1 c1 c1 c1 c1 c1 c1 c2 c1 c1 c1 c1 c1 c1
```

2.3) Deal with Nominal variables

For the purpose of linear regression, we should convert them into dummy variables



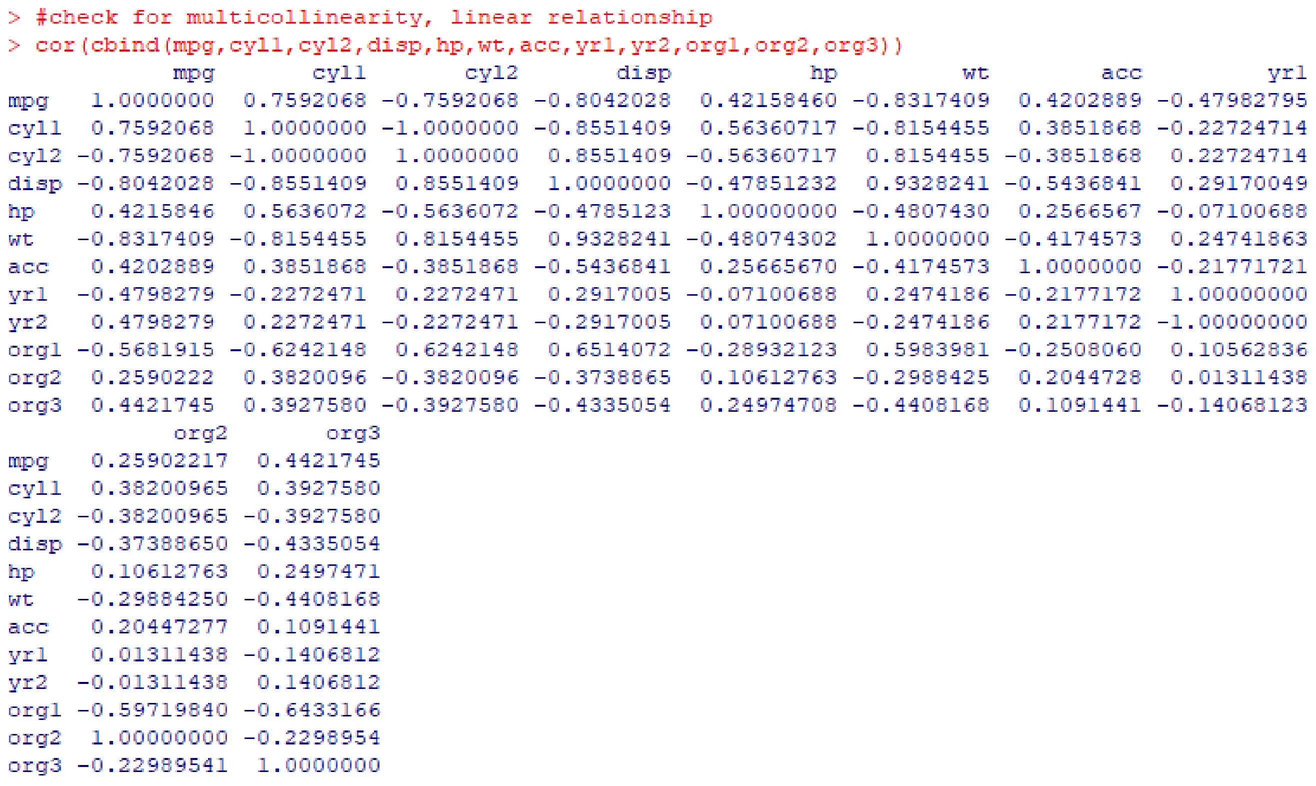
```
> #converting categorical variables to dummy variables
> library(dummies)
dummies-1.5.6 provided by Decision Patterns

> df=dummy.data.frame(dataset, names=c("cylinders","year","origin"))
>
> head(df)
  mpg cylindersc1 cylindersc2 displacement horsepower weight acceleration year1 year2 origin1 origin2 origin3
1  18           0           1          307         17   3504         12.0     1     0       1       0       0
2  15           0           1          350         35   3693         11.5     1     0       1       0       0
3  18           0           1          318         29   3436         11.0     1     0       1       0       0
4  16           0           1          304         29   3433         12.0     1     0       1       0       0
5  17           0           1          302         24   3449         10.5     1     0       1       0       0
6  15           0           1          429         42   4341         10.0     1     0       1       0       0
>
> cyl1=df$cylindersc1
> cyl2=df$cylindersc2
> yr1=df$year1
> yr2=df$year2
> org1=df$origin1
> org2=df$origin2
> org3=df$origin3
```

2.4) Make a decision on the model evaluations

This data set is small, and we should use N-fold cross validation in this case

Step 3: Build Models

3.1). Check linear relationships



```
> #check for multicollinearity, linear relationship
> cor(cbind(mpg,cyl1,cyl2,disp,hp,wt,acc,yr1,yr2,org1,org2,org3))
            mpg       cyl1       cyl2       disp          hp         wt        acc         yr1
mpg   1.0000000  0.7592068 -0.7592068 -0.8042028  0.42158460 -0.8317409  0.4202889 -0.47982795
cyl1  0.7592068  1.0000000 -1.0000000 -0.8551409  0.56360717 -0.8154455  0.3851868 -0.22724714
cyl2 -0.7592068 -1.0000000  1.0000000  0.8551409 -0.56360717  0.8154455 -0.3851868  0.22724714
disp -0.8042028 -0.8551409  0.8551409  1.0000000 -0.47851232  0.9328241 -0.5436841  0.29170049
hp    0.4215846  0.5636072 -0.5636072 -0.4785123  1.00000000 -0.4807430  0.2566567 -0.07100688
wt   -0.8317409 -0.8154455  0.8154455  0.9328241 -0.48074302  1.0000000 -0.4174573  0.24741863
acc   0.4202889  0.3851868 -0.3851868 -0.5436841  0.25665670 -0.4174573  1.0000000 -0.21771721
yr1  -0.4798279 -0.2272471  0.2272471  0.2917005 -0.07100688  0.2474186 -0.2177172  1.00000000
yr2   0.4798279  0.2272471 -0.2272471 -0.2917005  0.07100688 -0.2474186  0.2177172 -1.00000000
org1 -0.5681915 -0.6242148  0.6242148  0.6514072 -0.28932123  0.5983981 -0.2508060  0.10562836
org2  0.2590222  0.3820096 -0.3820096 -0.3738865  0.10612763 -0.2988425  0.2044728  0.01311438
org3  0.4421745  0.3927580 -0.3927580 -0.4335054  0.24974708 -0.4408168  0.1091441 -0.14068123
            org2       org3
mpg   0.25902217  0.4421745
cyl1  0.38200965  0.3927580
cyl2 -0.38200965 -0.3927580
disp -0.37388650 -0.4335054
hp    0.10612763  0.2497471
wt   -0.29884250 -0.4408168
acc   0.20447277  0.1091441
yr1   0.01311438 -0.1406812
yr2  -0.01311438  0.1406812
org1 -0.59719840 -0.6433166
org2  1.00000000 -0.2298954
org3 -0.22989541  1.0000000
```

Usually, if the correlation value is larger than 0.7, we say there is a strong linear relationship; they still have linear relationship if the correlation value is larger than 0.3. Otherwise, it is weak-linear relationship, or even no linear relationship.

In our case, hp, and acc have small correlation values, but still larger than 0.4. We tried different transformations, no much improvements. So we directly keep hp and acc

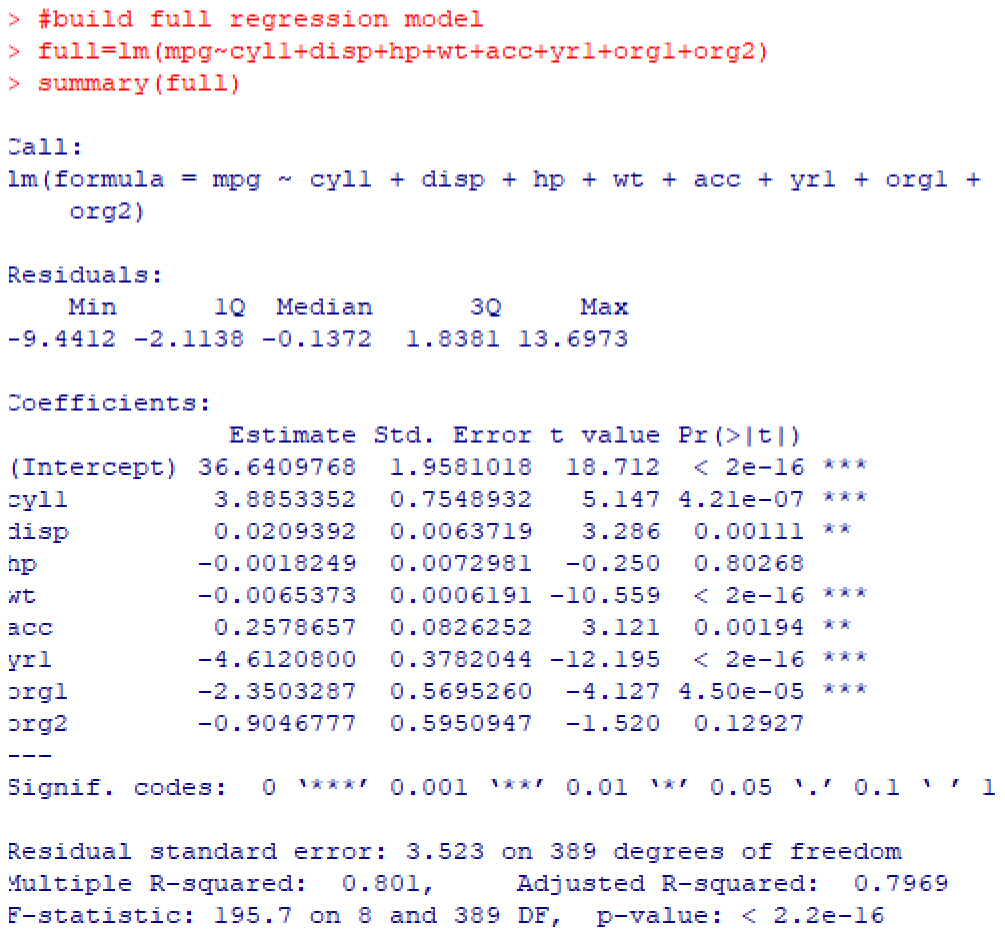
In terms of the dummy variables, it is difficult to interpret its relationship with mpg, we just keep it

3.2). Build models by feature selection

The models I planned to build

- A full model with all x variables
- A simple model with only one x variable
- Backward elimination by using p-value as metrics
- Backward elimination by using ACI as metrics
- Stepwise regression

For each model, we should examine goodness of fit test and the residuals to make sure they are qualified

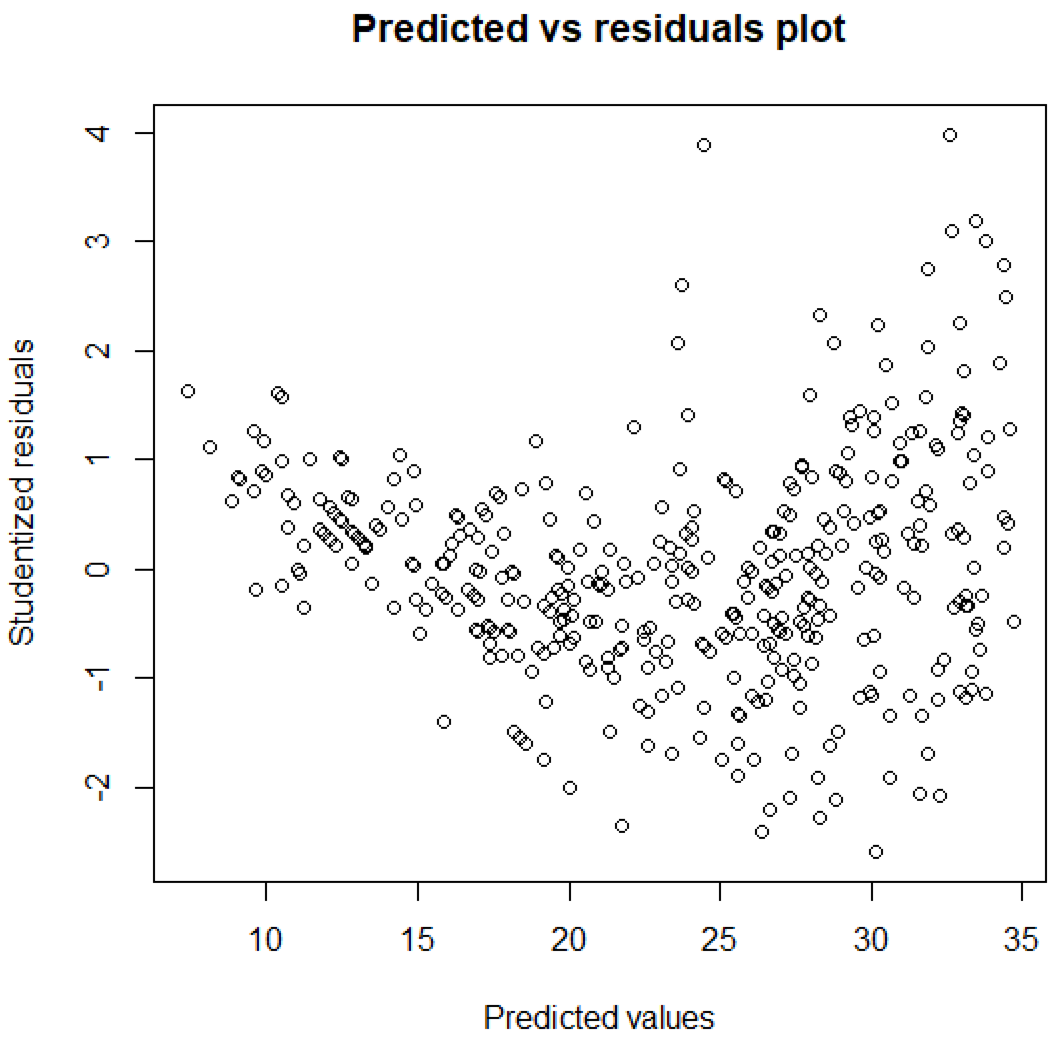


```
> #build full regression model
> full=lm(mpg~cyl1+disp+hp+wt+acc+yr1+org1+org2)
> summary(full)

Call:
lm(formula = mpg ~ cyl1 + disp + hp + wt + acc + yr1 + org1 +
    org2)

Residuals:
    Min      1Q  Median      3Q     Max
-9.4412 -2.1138 -0.1372  1.8381 13.6973

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 36.6409768  1.9581018  18.712  < 2e-16 ***
cyl1         3.8853352  0.7548932   5.147 4.21e-07 ***
disp         0.0209392  0.0063719   3.286  0.00111 **
hp          -0.0018249  0.0072981  -0.250  0.80268
wt          -0.0065373  0.0006191 -10.559  < 2e-16 ***
acc          0.2578657  0.0826252   3.121  0.00194 **
yr1         -4.6120800  0.3782044 -12.195  < 2e-16 ***
org1        -2.3503287  0.5695260  -4.127 4.50e-05 ***
org2        -0.9046777  0.5950947  -1.520  0.12927
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.523 on 389 degrees of freedom
Multiple R-squared:  0.801,	Adjusted R-squared:  0.7969
F-statistic: 195.7 on 8 and 389 DF,  p-value: < 2.2e-16
```

```
> #Backward Elimination by individual parameter test
> #eliminate hp
> m1=lm(mpg~cyl1+disp+wt+acc+yr1+org1+org2)
> summary(m1)

Call:
lm(formula = mpg ~ cyl1 + disp + wt + acc + yr1 + org1 + org2)

Residuals:
   Min     1Q Median     3Q    Max
-9.487 -2.080 -0.124  1.857 13.616

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 36.5977247  1.9481012  18.786  < 2e-16 ***
cyl1         3.8177874  0.7040568   5.423 1.03e-07 ***
disp         0.0208452  0.0063531   3.281  0.00113 **
wt          -0.0065257  0.0006166 -10.583  < 2e-16 ***
acc          0.2560796  0.0822169   3.115  0.00198 **
yr1         -4.6207867  0.3761453 -12.285  < 2e-16 ***
org1        -2.3538734  0.5686648  -4.139 4.27e-05 ***
org2        -0.8878561  0.5905690  -1.503  0.13355
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.519 on 390 degrees of freedom
Multiple R-squared:  0.8009,    Adjusted R-squared:  0.7973
F-statistic: 224.1 on 7 and 390 DF,  p-value: < 2.2e-16

> #eliminate org2
> m2=lm(mpg~cyl1+disp+wt+acc+yr1+org1)
> summary(m2)

Call:
lm(formula = mpg ~ cyl1 + disp + wt + acc + yr1 + org1)

Residuals:
    Min      1Q  Median      3Q     Max
-9.0922 -2.0770 -0.1044  1.8592 13.9726

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 36.6573338  1.9508338  18.791  < 2e-16 ***
cyl1         3.7380991  0.7031891   5.316 1.79e-07 ***
disp         0.0214339  0.0063512   3.375 0.000813 ***
wt          -0.0066569  0.0006114 -10.888  < 2e-16 ***
acc          0.2477451  0.0821619   3.015 0.002735 **
yr1         -4.6812451  0.3745915 -12.497  < 2e-16 ***
org1        -1.9386801  0.4979012  -3.894 0.000116 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.524 on 391 degrees of freedom
Multiple R-squared:  0.7998,    Adjusted R-squared:  0.7967
F-statistic: 260.3 on 6 and 391 DF,  p-value: < 2.2e-16
```

Next, residual analysis

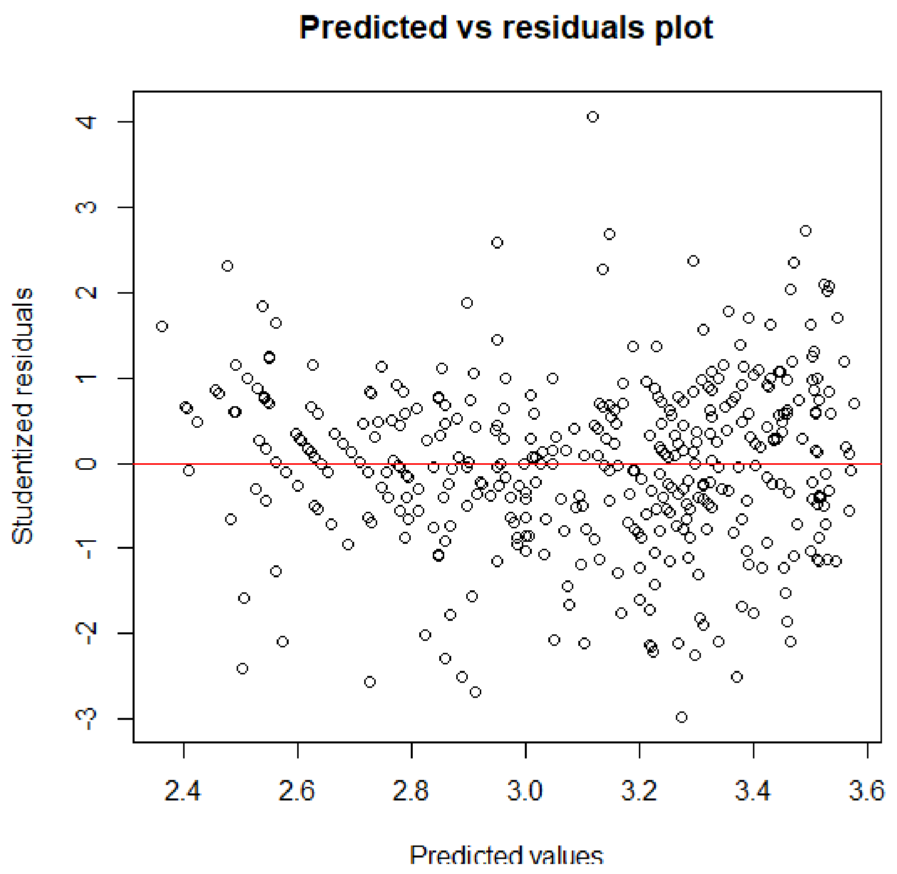
```
#Residual Analysis

plot( fitted(m2), rstandard(m2), xlab="Predicted values",ylab="Studentized residuals",main="Predicted vs residuals plot")

abline(a=0, b=0, col='red') #add zero line
```



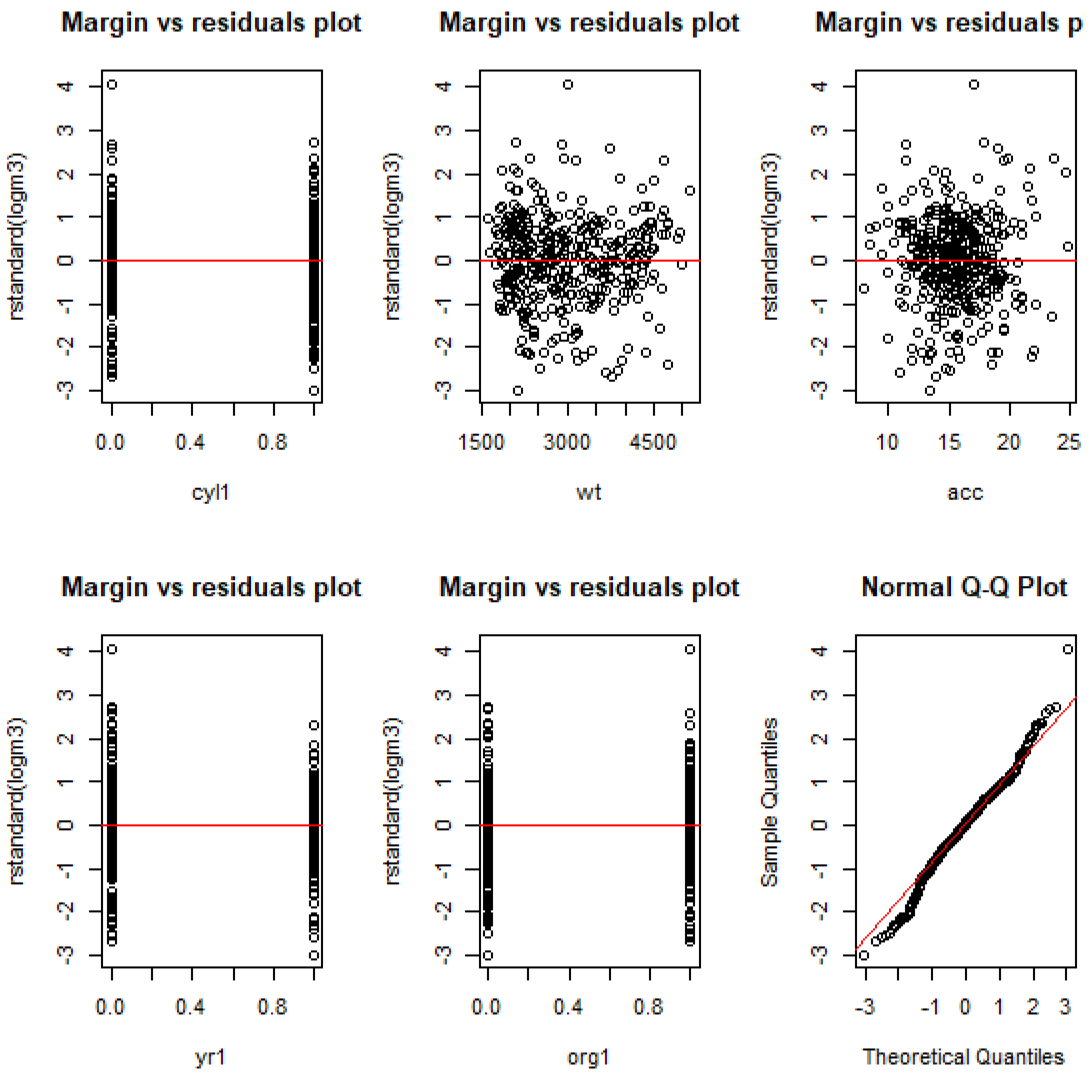
No constant variable! We should apply transformations on the y variable.

Here we tried log transformation, and then go through the previous steps again

```
#Non constant variance, applying transformation and rebuild model

logfull=lm(log(mpg)~cyl1+disp+hp+wt+acc+yr1+org1+org2)

summary(full)

#eliminate hp

logm1=lm(log(mpg)~cyl1+disp+wt+acc+yr1+org1+org2)

summary(logm1)

#eliminate org2

logm2=lm(log(mpg)~cyl1+disp+wt+acc+yr1+org1)

summary(logm2)

#eliminate disp

logm3=lm(log(mpg)~cyl1+wt+acc+yr1+org1)

summary(logm3)
```



Validate linear relationship between residual and x variables, and normality

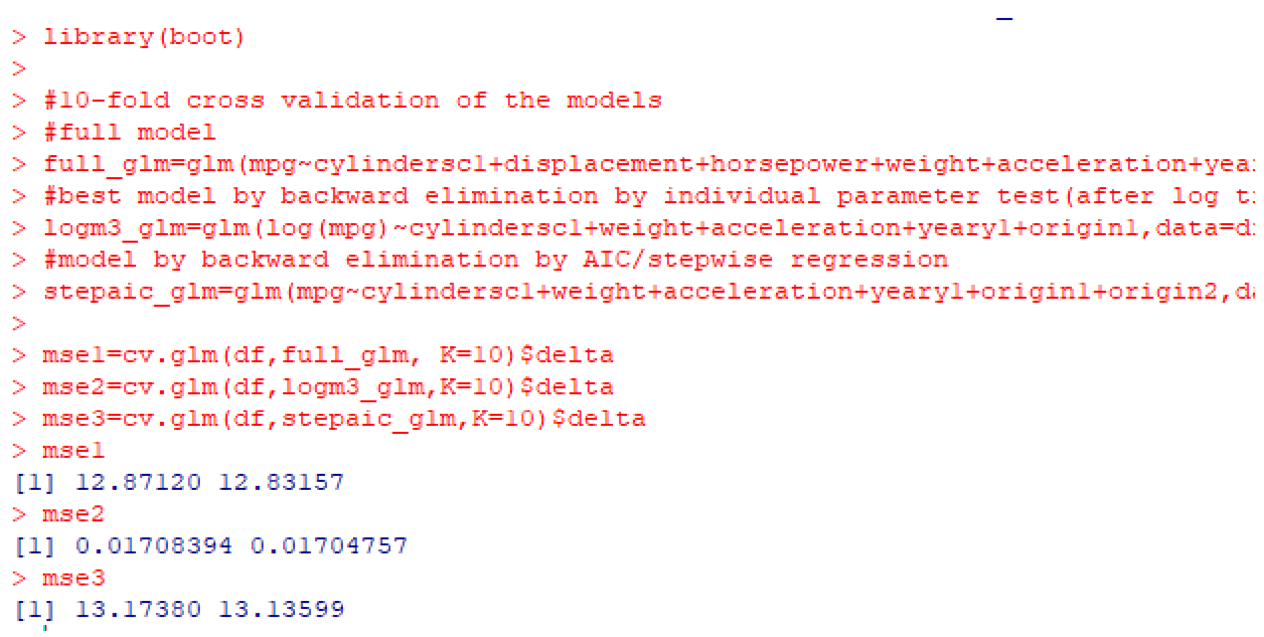


OK. Currently, we get the first qualified model by using backward elimination with p-value as metrics

Next, repeat the process, but use other feature selection methods

Step 4: Model Evaluations by N-folds Cross validations

Before the N-folds, you can take a look at the adj-R2 of the models you got, try to figure out whether there are big differences in the adj-R2. Here, we use 10-folds cross validations



```
> library(boot)
>
> #10-fold cross validation of the models
> #full model
> full_glm=glm(mpg~cylinderscl+displacement+horsepower+weight+acceleration+yea
> #best model by backward elimination by individual parameter test(after log t
> logm3_glm=glm(log(mpg)~cylinderscl+weight+acceleration+yearyl+originl,data=d
> #model by backward elimination by AIC/stepwise regression
> stepaic_glm=glm(mpg~cylinderscl+weight+acceleration+yearyl+originl+origin2,d
>
> msel=cv.glm(df,full_glm, K=10)$delta
> mse2=cv.glm(df,logm3_glm,K=10)$delta
> mse3=cv.glm(df,stepaic_glm,K=10)$delta
> msel
[1] 12.87120 12.83157
> mse2
[1] 0.01708394 0.01704757
> mse3
[1] 13.17380 13.13599
```

The MSE by using backward elimination with p-value as metrics is significantly smaller than other ones.

But note that, we applied the log transformation on the MPG variable in the 2nd model.

We should calculate the prediction errors (MAE, RMSE, MSE) at the same level for comparison purpose

So, how to calculate prediction error based on the level of y, instead of log(y)?????????

It is a big challenge, since cv.glm function does not provide a way to do so

Method – 1: can only be applied if you have log(y)

After searching online, I found that cv.lm is able to perform the N-fold cross validations for linear regression, and it does provide an option for log(y)!!

For this purpose, I need to build models by using lm, and perform N-fold cross validation by cv.lm

library(lmvar)

#full model

full_lm=lm(mpg~cylindersc1+displacement+horsepower+weight+acceleration+yeary1+origin1+origin2,data=df,**x=T,y=T**)

#model by backward elimination by AIC/stepwise regression

stepaic_lm=lm(mpg~cylindersc1+weight+acceleration+yeary1+origin1+origin2,data=df,**x=T,y=T**)

#best model by backward elimination by individual parameter test(after log transformation)

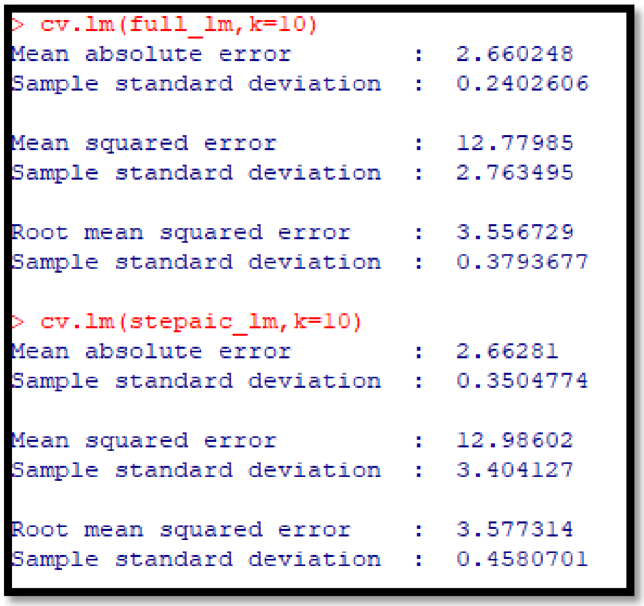
logm3_lm=lm(log(mpg)~cylindersc1+weight+acceleration+yeary1+origin1,data=df, **x=T, y=T**)

cv.lm(full_lm,k=10)

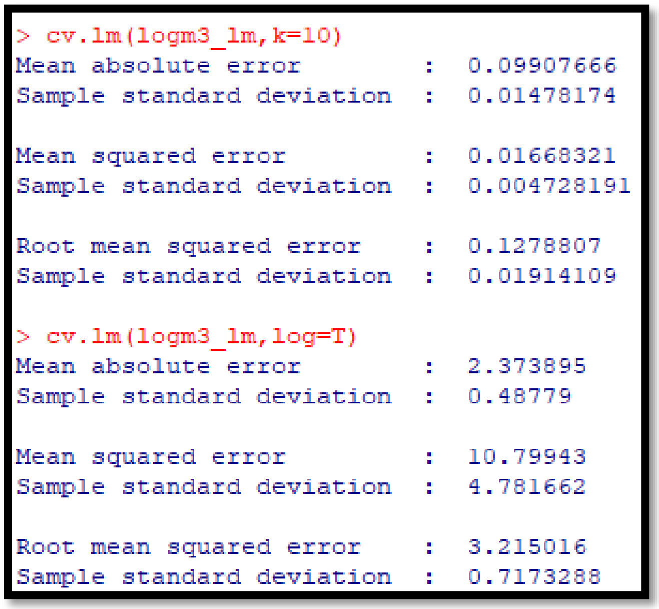
cv.lm(stepaic_lm,k=10)

cv.lm(logm3_lm,k=10)

cv.lm(logm3_lm,**log=T**) ➔ set log=T, so that they will produce prediction errors on y, not log(y)



```
> cv.lm(full_lm,k=10)
Mean absolute error          :  2.660248
Sample standard deviation    :  0.2402606

Mean squared error           :  12.77985
Sample standard deviation    :  2.763495

Root mean squared error      :  3.556729
Sample standard deviation    :  0.3793677

> cv.lm(stepaic_lm,k=10)
Mean absolute error          :  2.66281
Sample standard deviation    :  0.3504774

Mean squared error           :  12.98602
Sample standard deviation    :  3.404127

Root mean squared error      :  3.577314
Sample standard deviation    :  0.4580701
```



```
> cv.lm(logm3_lm,k=10)
Mean absolute error          :  0.09907666
Sample standard deviation    :  0.01478174

Mean squared error           :  0.01668321
Sample standard deviation    :  0.004728191

Root mean squared error      :  0.1278807
Sample standard deviation    :  0.01914109

> cv.lm(logm3_lm,log=T)
Mean absolute error          :  2.373895
Sample standard deviation    :  0.48779

Mean squared error           :  10.79943
Sample standard deviation    :  4.781662

Root mean squared error      :  3.215016
Sample standard deviation    :  0.7173288
```

Method – 2: is a general method that can be applied to any transformation on y

The idea is simple. We use the caret package, and use trainControl to perform the N-fold cross validation.

We need to save the predictions, and apply de-transformation to convert it back to the predicted values for the original y variable. And then calculate prediction errors, such as MAE, MSE, RMSE, etc

```
train.control <- trainControl(method = "cv", number = 10,savePredictions = TRUE)
model2 <- train(log(mpg)~cylindersc1+weight+acceleration+yeary1+origin1, data = df,
method = "glm",trControl = train.control)
model2
model2#results
calcrmse(model2$pred)
```

Here, model$pred has pred, obs, Resample (Fold01, Fold02,..) as columns.

```
calcrmse<-function(y){
#y=model2$pred
folds=unique(y$Resample)
for(k in 1:length(folds)){
 #n=length(which(y$Resample==folds[k]))
 dfx=y[y$Resample==folds[k],]
 sumsq=0
 for(i in 1:nrow(dfx)){
  yobs=exp(dfx$obs[i])
  ypred=exp(dfx$pred[i])
  sumsq=sumsq+(yobs-ypred)%*%(yobs-ypred)
 }
 rms[k]=sqrt(sumsq/nrow(dfx))
}
return(mean(rms))
}
```