Regression and Variable Selection

The objective of this case is to get you started with regression model building, variable selection, and model evaluation in R, and help you with Case 2.

Code in this file is not the only correct way to do things, however it is important for you to understand what each statement does. You will have to modify the code accordingly for your homework.

Boston Housing Data

Load Data

```
library(MASS)
data(Boston); #this data is in MASS package
colnames(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"
"age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"
"lstat"   "medv"
```

The original data are 506 observations on 14 variables, medv being the response variable y :

Variable	Description
crim	per capita crime rate by town
zn	proportion of residential land zoned for lots over 25,000 sq.ft
indus	proportion of non-retail business acres per town
chas	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
nox	nitric oxides concentration (parts per 10 million)
rm	average number of rooms per dwelling
age	proportion of owner-occupied units built prior to 1940
dis	weighted distances to five Boston employment centres
rad	index of accessibility to radial highways
tax	full-value property-tax rate per USD 10,000
ptratio	pupil-teacher ratio by town
black	$1000(B - 0.63)^2$ where B is the proportion of blacks by town
lstat	percentage of lower status of the population
medv	median value of owner-occupied homes in USD 1000's

Sampling (Split Dataset Randomly)

Next we sample 90% of the original data and use it as the training set. The remaining 10% is used as test set. The regression model will be built on the training set and future performance of your model will be evaluated with the test set.

```
subset <- sample(nrow(Boston), nrow(Boston) * 0.9)
Boston_train = Boston[subset, ]
Boston_test = Boston[-subset, ]
```

(Optional) Standardization

If we want our results to be invariant to the units and the parameter estimates β_i to be comparible, we can standardize the variables. Essentially we are replacing the original values with their z-score.

1st Way: create new variables manually.

```
Boston$sd.crim <- (Boston$crim - mean(Boston$crim))/sd
(Boston$crim)
```

This does the same thing.

```
Boston$sd.crim <- scale(Boston$crim)
```

2nd way: If you have a lot of variables to standardize then the above is not very pleasing to do. You can use a loop like this. It standardizes every varables other than the last one which is y .

```
for (i in 1:(ncol(Boston_train) - 1)) {
    Boston_train[, i] = scale(Boston_train[, i])
}
```

The technique is not as important in linear regression because it will only affect the interpretation but not the model estimation and inference.

Model Building

You task is to build a best model with training data. You can refer to the regression and variable selection code on the slides for more detailed description of linear regression.

The following model includes all x varables in the model

```
model_1 <- lm(medv ~ crim + zn + chas + nox + rm + dis + rad
+ tax + ptratio +
    black + lstat, data = Boston_train)
```

To include all variables in the model, you can write the statement this simpler way.

```
model_1 <- lm(medv ~ ., data = Boston_train)
summary(model_1)
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.831  -2.808  -0.601   2.115  26.072
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  22.6699     0.2292   98.89  < 2e-16 ***
## crim         -0.9604     0.3028   -3.17  0.00162 **
## zn            1.1326     0.3411    3.32  0.00097 ***
## indus         0.0491     0.4674    0.10  0.91645
## chas          0.7339     0.2388    3.07  0.00225 **
## nox          -2.1542     0.4784   -4.50  8.6e-06 ***
## rm            2.6978     0.3173    8.50  2.9e-16 ***
## age           0.0397     0.4048    0.10  0.92191
## dis          -3.2109     0.4593   -6.99  1.0e-11 ***
## rad           2.5975     0.6220    4.18  3.6e-05 ***
## tax          -1.9474     0.6979   -2.79  0.00549 **
## ptratio      -2.0402     0.3045   -6.70  6.4e-11 ***
## black         0.8555     0.2680    3.19  0.00151 **
## lstat        -3.7356     0.3850   -9.70  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
' 1
##
## Residual standard error: 4.89 on 441 degrees of freedom
## Multiple R-squared: 0.733,   Adjusted R-squared: 0.726
## F-statistic: 93.3 on 13 and 441 DF,  p-value: <2e-16
```

But, is this the model you want to use?

(Optional) Interaction terms in model

If you suspect the effect of one predictor x1 on the response y depends on the value of another predictor x2, you can add interaction terms in model. To specify interaction in the model, you put : between two variables with interaction effect. For example

```
lm(medv ~ crim + zn + crim:zn, data = Boston_train)
# The following way automatically add the main effects of crim
and zn
lm(medv ~ crim * zn, data = Boston_train)
```

For now we will not investigate the interactions of variables.

Evaluating Model Fitness

In-sample model evaluation (train error)

MSE of the regression, which is the square of 'Residual standard error' in the above summary. It is the sum of squared residuals(SSE) divided by degrees of freedom (n-p-1). In some textbooks the sum of squred residuals(SSE) is called residual sum of squares(RSS). MSE of the regression should be the unbiased estimator for variance of ϵ , the error term in the regression model.

```
model_summary <- summary(model_1)
(model_summary$sigma)^2
```

```
## [1] 23.91
```

R^2 of the model

```
model_summary$r.squared
```

```
## [1] 0.7334
```

Adjusted-R^2 of the model, if you add a variable (or several in a group), SSE will decrease, R^2 will increase, but Adjusted-R^2 could go either way.

```
model_summary$adj.r.squared
```

```
## [1] 0.7256
```

AIC and BIC of the model, these are information criteria. Smaller values indicate better fit.

```
AIC(model_1)
```

```
## [1] 2751
```

```
BIC(model_1)
```

```
## [1] 2813
```

BIC, AIC, and Adjusted R^2 have complexity penalty in the definition, thus when comparing across different models they are better indicators on how well the model will perform on future data.

Out-of-sample prediction (test error)

To evaluate how the model performs on future data, we use predict() to get the predicted values from the test set.

```
# pi is a vector that contains predicted values for test set.
pi <- predict(object = model_1, newdata = Boston_test)
```

Just as any other function, you can write the above statement the following way as long as the arguments are in the right order.

```
pi <- predict(model_1, Boston_test)
```

The most common measure is the Mean Squared Error (MSE): average of the squared differences between the predicted and actual values

```
mean((pi - Boston_test$medv)^2)
```

```
## [1] 39.23
```

A less popular measure is the Mean Absolute Error (MAE). You can probably guess that here instead of taking the average of squared error, MAE is the average of absolute value of error.

```
mean(abs(pi - Boston_test$medv))
```

```
## [1] 4.032
```

Note that if you ignore the second argument of predict(), it gives you the in-sample prediction on the training set:

```
predict(model_1)
```

Which is the same as

```
model_1$fitted.values
```

Variable Selection

Compare Model Fit Manually

```
model_1 <- lm(medv ~ ., data = Boston_train)
model_2 <- lm(medv ~ crim + zn, data = Boston_train)
summary(model_1)
summary(model_2)
AIC(model_1)
AIC(model_2)
```

Best Subset Regression

The 'leaps' package provides procedures for best subset regression.

```
install.packages("leaps")
```

```
library(leaps)
```

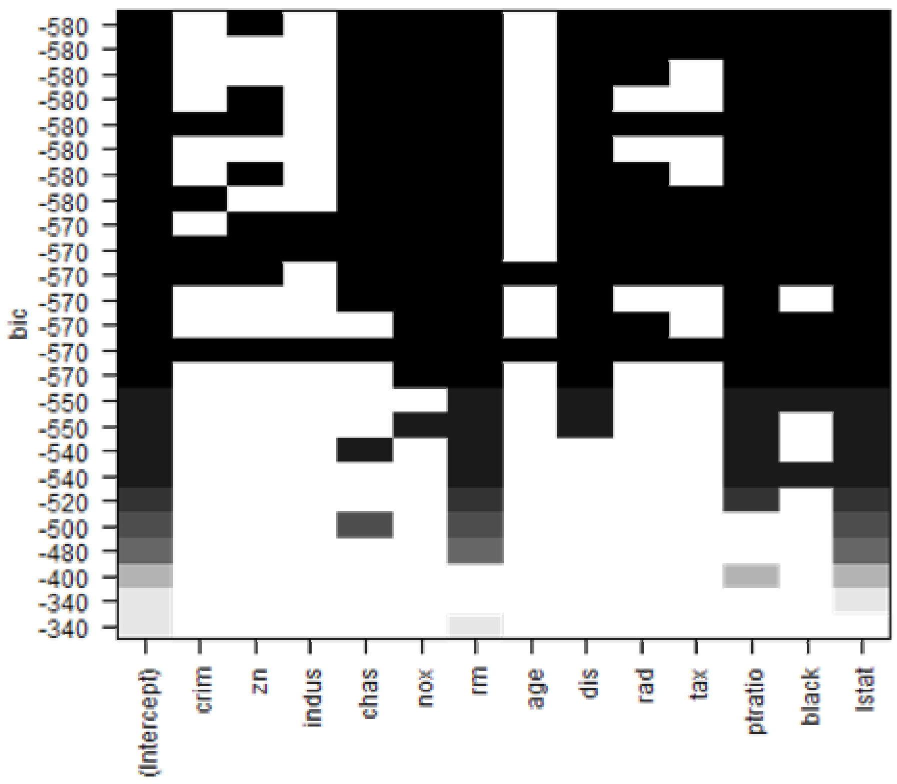
Which subset of variables should you include in order to minimize BIC?

```
# regsubsets only takes data frame as input
subset_result <- regsubsets(medv ~ ., data = Boston_train,
nbest = 2, nvmax = 14)
summary(subset_result)
```

```
## Subset selection object
## Call: regsubsets.formula(medv ~ ., data = Boston_train,
nbest = 2,
##     nvmax = 14)
## 13 Variables  (and intercept)
##          Forced in Forced out
## crim         FALSE      FALSE
## zn           FALSE      FALSE
## indus        FALSE      FALSE
## chas         FALSE      FALSE
## nox          FALSE      FALSE
## rm           FALSE      FALSE
## age          FALSE      FALSE
## dis          FALSE      FALSE
## rad          FALSE      FALSE
## tax          FALSE      FALSE
## ptratio      FALSE      FALSE
## black        FALSE      FALSE
## lstat        FALSE      FALSE
## 2 subsets of each size up to 13
## Selection Algorithm: exhaustive
##           crim zn  indus chas nox rm  age dis rad tax
ptratio black lstat
## 1  ( 1 )  " "  " " " "   " "  " " " " " " " " " " " " " "
" "   "*"
## 1  ( 2 )  " "  " " " "   " "  " " "*" " " " " " " " " " "
" "   " "
## 2  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " " " "
" "   "*"
## 2  ( 2 )  " "  " " " "   " "  " " " " " " " " " " " " "*"
" "   "*"
## 3  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " " "*"
" "   "*"
## 3  ( 2 )  " "  " " " "   "*"  " " "*" " " " " " " " " " "
" "   "*"
## 4  ( 1 )  " "  " " " "   "*"  " " "*" " " " " " " " " "*"
" "   "*"
## 4  ( 2 )  " "  " " " "   " "  " " "*" " " " " " " " " "*"
"*"   "*"
## 5  ( 1 )  " "  " " " "   " "  " " "*" " " "*" " " " " "*"
"*"   "*"
## 5  ( 2 )  " "  " " " "   " "  "*" "*" " " "*" " " " " "*"
" "   "*"
## 6  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" " " " " "*"
" "   "*"
## 6  ( 2 )  " "  " " " "   " "  "*" "*" " " "*" " " " " "*"
"*"   "*"
## 7  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" " " " " "*"
"*"   "*"
## 7  ( 2 )  " "  " " " "   " "  "*" "*" " " "*" "*" " " "*"
"*"   "*"
## 8  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" "*" " " "*"
"*"   "*"
## 8  ( 2 )  " "  "*" " "   "*"  "*" "*" " " "*" " " " " "*"
"*"   "*"
## 9  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" "*" "*" "*"
```

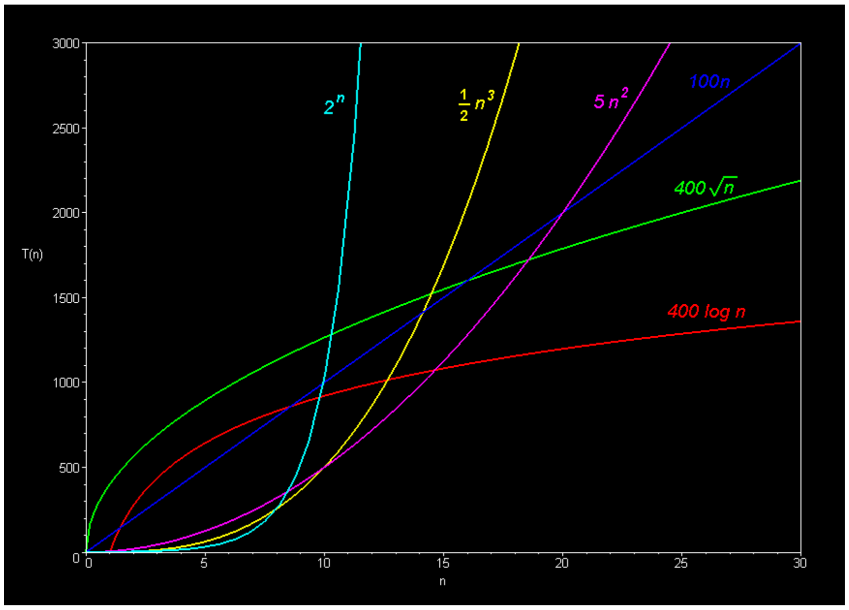
```
"*"    "*"
## 9  ( 2 )  " "  "*" " "   "*"  "*" "*" " " "*" "*" " " "*"
"*"    "*"
## 10  ( 1 ) " "  "*" " "   "*"  "*" "*" " " "*" "*" "*" "*"
"*"    "*"
## 10  ( 2 ) "*"  " " " "   "*"  "*" "*" " " "*" "*" "*" "*"
"*"    "*"
## 11  ( 1 ) "*"  "*" " "   "*"  "*" "*" " " "*" "*" "*" "*"
"*"    "*"
## 11  ( 2 ) " "  "*" "*"   "*"  "*" "*" " " "*" "*" "*" "*"
"*"    "*"
## 12  ( 1 ) "*"  "*" "*"   "*"  "*" "*" " " "*" "*" "*" "*"
"*"    "*"
## 12  ( 2 ) "*"  "*" " "   "*"  "*" "*" "*" "*" "*" "*" "*"
"*"    "*"
## 13  ( 1 ) "*"  "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"
"*"    "*"
```

```
plot(subset_result, scale = "bic")
```



Each row represents a model. Black indicates that a variable is included in the model, while white indicates that it is not. The scale = “” can be “Cp”, “adjr2”, “r2” or “bic”.

What is the problem with best subset regression? If there are n independent variables, the number of possible nonempty subsets is $2^n - 1$. If you try a best subset regression with more than 50 variables, you might need to wait for your entire life to get the result.



Forward/Backward/Stepwise Regression Using AIC

To perform the Forward/Backward/Stepwise Regression in R, we need to define the starting points:

```
nullmodel = lm(medv ~ 1, data = Boston_train)
fullmodel = lm(medv ~ ., data = Boston_train)
```

nullmodel is the model with no varaible in it, while fullmodel is the model with every variable in it.

Backward Elimination

```
model.step <- step(fullmodel, direction = "backward")
```

```
## Start:  AIC=1398
## medv ~ crim + zn + indus + chas + nox + rm + age + dis + 
rad + 
##     tax + ptratio + black + lstat
## 
##           Df Sum of Sq   RSS  AIC
## - age      1         0  9244 1396
## - indus    1         7  9251 1397
## <none>                  9244 1398
## - crim     1        63  9307 1399
## - zn       1       168  9412 1404
## - tax      1       190  9434 1405
## - chas     1       305  9550 1411
## - nox      1       356  9600 1413
## - rad      1       366  9611 1414
## - black    1       377  9621 1414
## - dis      1       945 10189 1441
## - ptratio  1      1001 10245 1443
## - lstat    1      1744 10989 1475
## - rm       1      2334 11578 1499
## 
## Step:  AIC=1396
## medv ~ crim + zn + indus + chas + nox + rm + dis + rad + 
tax + 
##     ptratio + black + lstat
## 
##           Df Sum of Sq   RSS  AIC
## - indus    1         7  9251 1395
## <none>                  9244 1396
## - crim     1        63  9307 1397
## - zn       1       171  9415 1403
## - tax      1       190  9434 1403
## - chas     1       307  9551 1409
## - rad      1       370  9614 1412
## - black    1       380  9624 1413
## - nox      1       383  9628 1413
## - ptratio  1      1003 10247 1441
## - dis      1      1029 10273 1442
## - lstat    1      2051 11295 1485
## - rm       1      2474 11719 1502
## 
## Step:  AIC=1395
## medv ~ crim + zn + chas + nox + rm + dis + rad + tax + 
ptratio + 
##     black + lstat
## 
##           Df Sum of Sq   RSS  AIC
## <none>                  9251 1395
## - crim     1        65  9316 1396
## - zn       1       165  9416 1401
## - tax      1       202  9453 1402
## - chas     1       321  9572 1408
## - black    1       375  9626 1411
## - rad      1       375  9627 1411
## - nox      1       385  9637 1411
## - ptratio  1       999 10251 1439
## - dis      1      1102 10354 1444
## - lstat    1      2045 11296 1483
## - rm       1      2469 11721 1500
```

Forward Selection

```
model.step <- step(nullmodel, scope = list(lower = nullmodel,
upper = fullmodel),
    direction = "forward")
```

```
## Start:  AIC=2023
## medv ~ 1
##
##           Df Sum of Sq   RSS  AIC
## + lstat    1     21021 17644 1668
## + rm       1     20805 17860 1674
## + ptratio  1     10535 28130 1881
## + indus    1      9560 29105 1896
## + tax      1      9277 29387 1900
## + nox      1      6850 31815 1937
## + crim     1      6297 32368 1944
## + rad      1      6080 32585 1947
## + age      1      5367 33298 1957
## + zn       1      4968 33697 1963
## + black    1      4730 33935 1966
## + dis      1      2412 36253 1996
## + chas     1      1465 37199 2008
## <none>                 38665 2023
##
## Step:  AIC=1668
## medv ~ lstat
##
##           Df Sum of Sq   RSS  AIC
## + rm       1      4705 12938 1529
## + ptratio  1      2390 15253 1604
## + chas     1       891 16752 1647
## + dis      1       772 16872 1650
## + age      1       471 17172 1658
## + tax      1       239 17404 1664
## + black    1       205 17439 1665
## + zn       1       123 17520 1667
## + indus    1       105 17538 1668
## <none>                 17644 1668
## + crim     1        63 17581 1669
## + nox      1        16 17627 1670
## + rad      1        12 17631 1670
##
## Step:  AIC=1529
## medv ~ lstat + rm
##
##           Df Sum of Sq   RSS  AIC
## + ptratio  1      1327 11611 1482
## + chas     1       642 12296 1508
## + black    1       594 12344 1510
## + tax      1       321 12617 1520
## + dis      1       319 12619 1520
## + crim     1       116 12822 1527
## + rad      1       115 12823 1527
## <none>                 12938 1529
## + indus    1        51 12888 1529
## + age      1        39 12899 1530
## + zn       1        32 12906 1530
## + nox      1         4 12934 1531
##
## Step:  AIC=1482
## medv ~ lstat + rm + ptratio
```

```
##
##         Df Sum of Sq   RSS  AIC
## + chas   1       515 11096 1463
## + black  1       485 11126 1464
## + dis    1       458 11153 1466
## + age    1        76 11535 1481
## <none>               11611 1482
## + tax    1        32 11579 1483
## + crim   1        28 11583 1483
## + zn     1        22 11589 1483
## + nox    1        11 11600 1483
## + rad    1         9 11602 1484
## + indus  1         1 11610 1484
##
## Step:  AIC=1463
## medv ~ lstat + rm + ptratio + chas
##
##         Df Sum of Sq   RSS  AIC
## + black  1       449 10647 1446
## + dis    1       344 10753 1451
## <none>               11096 1463
## + nox    1        39 11058 1464
## + tax    1        35 11062 1464
## + age    1        28 11068 1464
## + crim   1        23 11074 1464
## + zn     1         6 11090 1465
## + indus  1         4 11093 1465
## + rad    1         3 11093 1465
##
## Step:  AIC=1446
## medv ~ lstat + rm + ptratio + chas + black
##
##         Df Sum of Sq   RSS  AIC
## + dis    1       458 10189 1429
## + rad    1        98 10549 1444
## <none>               10647 1446
## + age    1        38 10609 1447
## + zn     1        10 10637 1448
## + indus  1         7 10640 1448
## + crim   1         5 10643 1448
## + tax    1         1 10646 1448
## + nox    1         1 10646 1448
##
## Step:  AIC=1429
## medv ~ lstat + rm + ptratio + chas + black + dis
##
##         Df Sum of Sq   RSS  AIC
## + nox    1       424  9765 1411
## + zn     1       145 10044 1424
## + indus  1       131 10059 1425
## + age    1        85 10105 1427
## <none>               10189 1429
## + tax    1        35 10154 1429
## + rad    1        21 10169 1430
## + crim   1         1 10188 1430
##
## Step:  AIC=1411
## medv ~ lstat + rm + ptratio + chas + black + dis + nox
##
##          Df Sum of Sq  RSS  AIC
## + rad    1     168.8 9596 1405
## + zn     1     135.4 9630 1407
## <none>               9765 1411
## + age    1      11.6 9753 1413
## + indus  1      10.1 9755 1413
## + tax    1       9.0 9756 1413
## + crim   1       0.1 9765 1413
##
## Step:  AIC=1405
## medv ~ lstat + rm + ptratio + chas + black + dis + nox +
rad
##
##          Df Sum of Sq  RSS  AIC
## + tax    1     134.6 9462 1401
## + zn     1      88.6 9508 1403
## <none>               9596 1405
## + crim   1      41.4 9555 1405
## + indus  1      16.8 9580 1406
## + age    1       3.8 9593 1407
##
## Step:  AIC=1401
## medv ~ lstat + rm + ptratio + chas + black + dis + nox +
rad +
##     tax
##
##          Df Sum of Sq  RSS  AIC
## + zn     1     145.6 9316 1396
## + crim   1      45.6 9416 1401
## <none>               9462 1401
## + age    1       2.4 9459 1403
## + indus  1       2.1 9460 1403
##
## Step:  AIC=1396
## medv ~ lstat + rm + ptratio + chas + black + dis + nox +
rad +
##     tax + zn
##
##          Df Sum of Sq  RSS  AIC
## + crim   1      64.8 9251 1395
## <none>               9316 1396
## + indus  1       9.2 9307 1397
## + age    1       0.0 9316 1398
##
## Step:  AIC=1395
## medv ~ lstat + rm + ptratio + chas + black + dis + nox +
rad +
##     tax + zn + crim
##
##          Df Sum of Sq  RSS  AIC
## <none>               9251 1395
## + indus  1      7.28 9244 1396
## + age    1      0.00 9251 1397
```

Stepwise Selection (Output Omitted)

```
model.step <- step(nullmodel, scope = list(lower = nullmodel,
upper = fullmodel),
    direction = "both")
```

One caution when comparing fit statistics using AIC, the definition varies by program/procedure.

```
extractAIC(model_1)
```

```
## [1]   14 1398
```

```
AIC(model_1)
```

```
## [1] 2691
```

- For pros and cons of variable/model selection using the common fit statistics: (adjusted) R^2 , MSE, AIC, BIC, etc. refer to Ch9 in “Applied Linear Regression Models” by Kutner et al.
- For other variable selection methods refer to section 3.4 - 3.8 of “Elements of Statistical Learning” (Free Online).

Cross Validation

Cross validation is an alternative approach to training/testing split. For k-fold cross validation, the dataset is divided into k parts. Each part serves as the test set in each iteration and the rest serve as training set. The out-of-sample performance measures from the k iterations are averaged.

Note

1. We use the **entire** dataset for cross validation

2. We need to use glm instead of lm to fit the model (if we want to use cv.glm fucntion in boot package)

3. The default measure of performance is the Mean Squared Error (MSE). If we want to use another measure we need to define a cost function.

5-fold Cross Validation

```
library(boot)
model_2 = glm(medv ~ indus + rm, data = Boston)
cv.glm(data = Boston, glmfit = model_2, K = 5)$delta[2]
```

```
## [1] 41.37
```

LOOCV (Leave-one-out Cross Validation)

```
cv.glm(data = Boston, glmfit = model_2, K = nrow(Boston))
$delta[2]
```

```
## [1] 39.94
```

5-fold Cross Validation Using MAE

Here we need to define a MAE cost function. The function takes 2 input vectors, pi = predicted values, r = actual values.

```
model_2 = glm(medv ~ indus + rm, data = Boston)

MAE_cost = function(pi, r) {
    return(mean(abs(pi - r)))
}

cv.glm(data = Boston, glmfit = model_2, cost = MAE_cost, K =
5)$delta[2]
```

```
## [1] 4.288
```

Another package DAAG also does cross validation. It prints out the performance in each fold and gives you a plot at the end. But currently I cannot figure out how to get the cross-validation error programmatically.

```
install.packages("DAAG")
```

```
library(DAAG)
```

```
model_2 = lm(medv ~ indus + rm, data = Boston)
cv.lm(df = Boston, form.lm = model_2, m = 3)
```

Diagnostic Plots

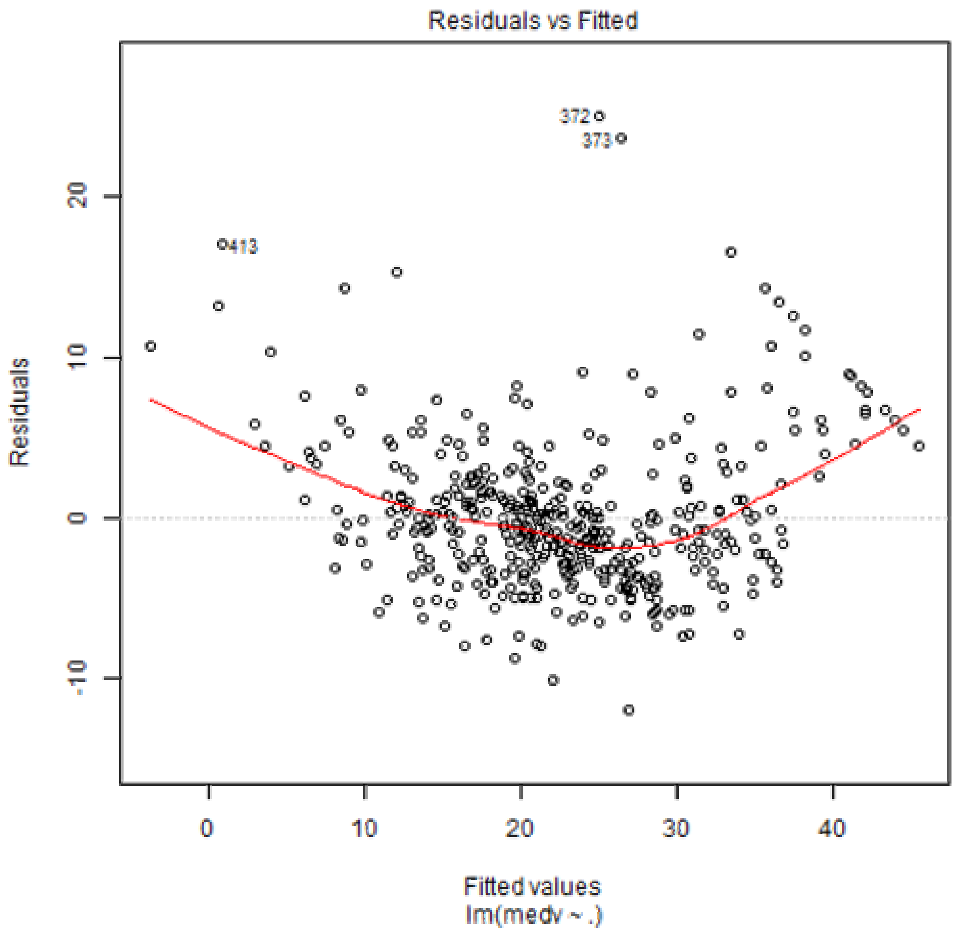
The diagnostic plots are not as important when regression is used in predictive (supervised) data mining as when it is used in economics. However it is still good to know:

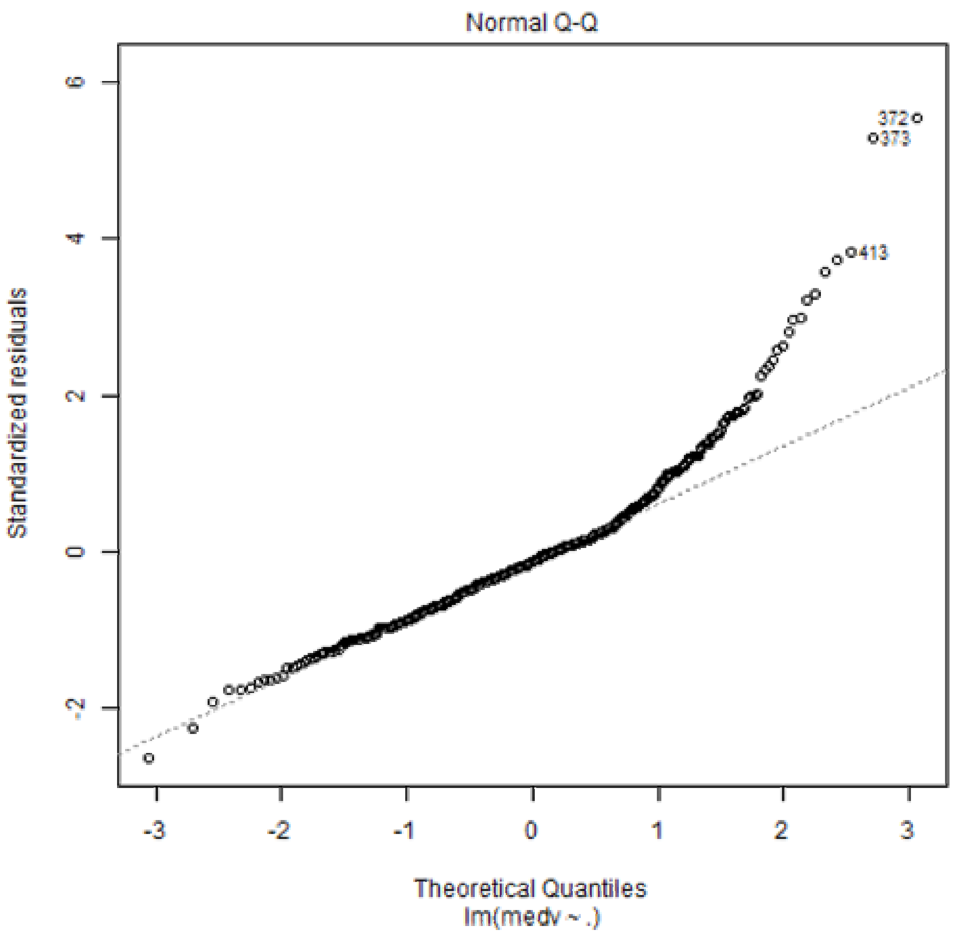
1. What the diagnostic plots should look like when no assumption is violated?
2. If there is something wrong, what assumptions are possibly violated?
3. What implications does it have on the analysis?
4. (How) can I fix it?

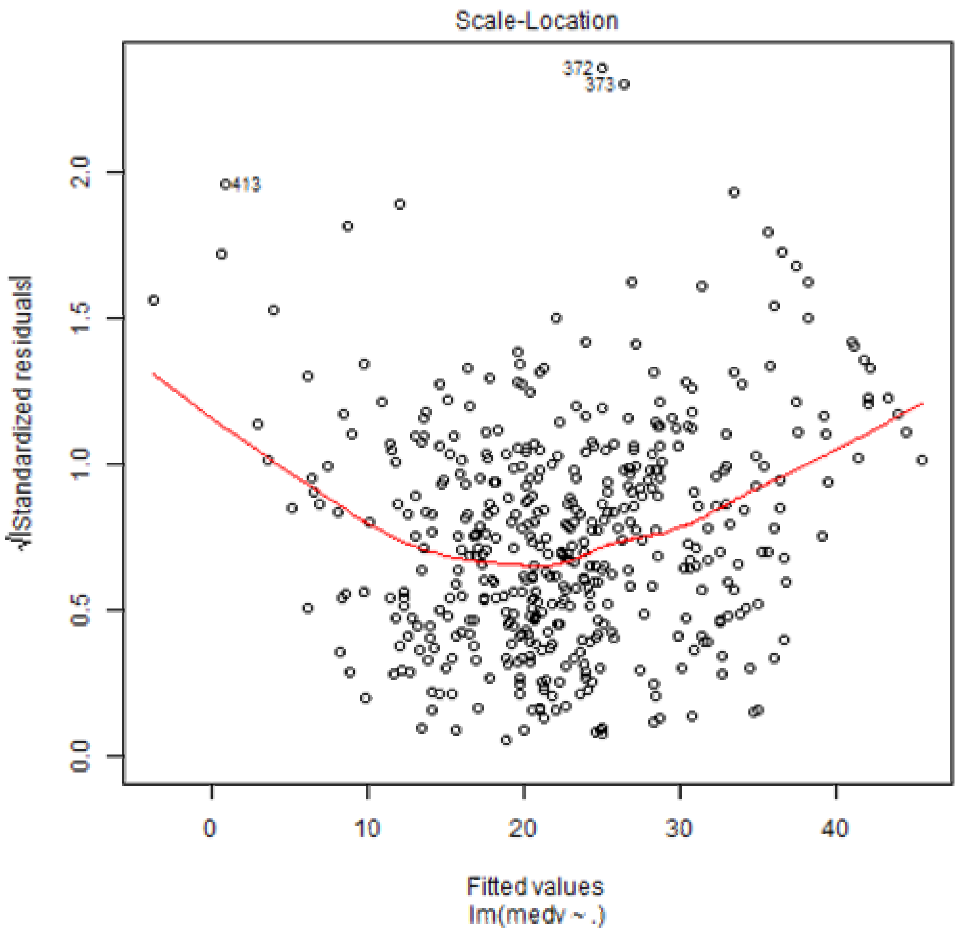
Roughly speaking, the table summarizes what you should look for in the following plots

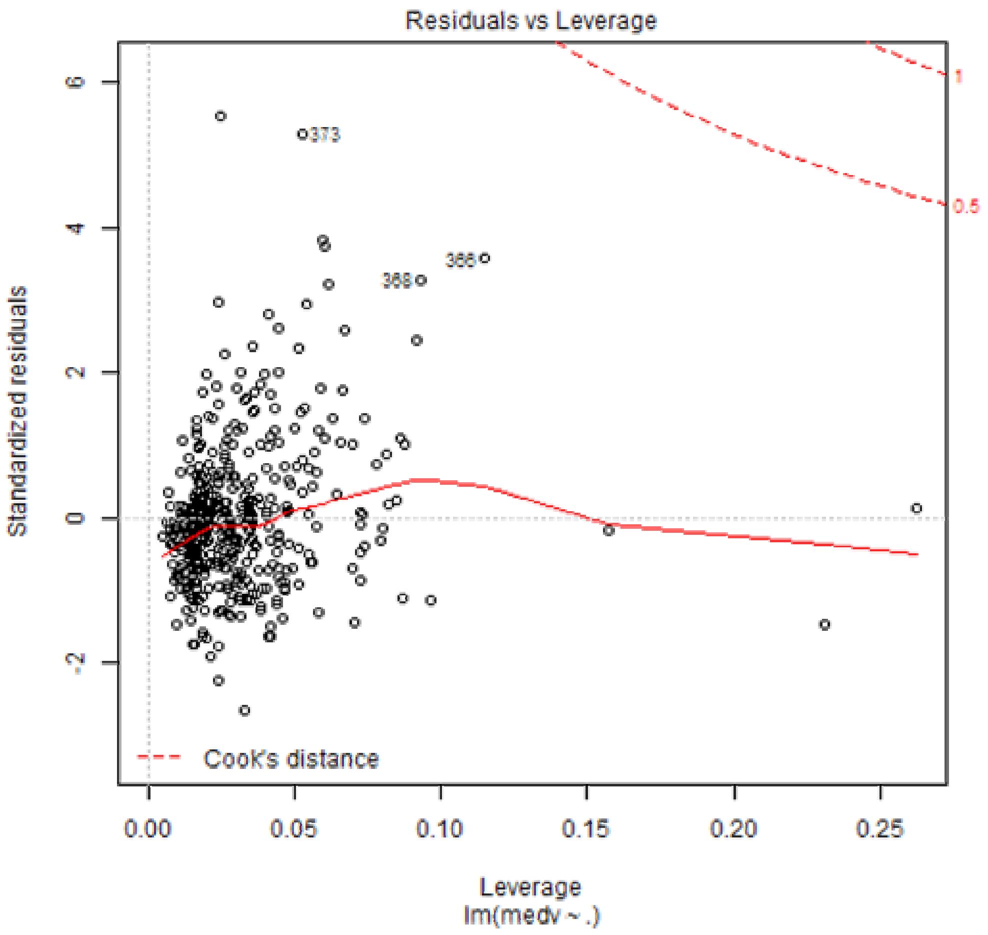
Plot Name	Good
Residual vs. Fitted	No pattern, scattered around 0 line
Normal Q-Q	Dots fall on dashed line
Residual vs. Leverage	No observation with large Cook's distance

```
plot(model_1)
```


Residuals vs Fitted
Residuals
Fitted values
lm(medv ~ .)
372
373
413
20
10
0
-10
0
10
20
30
40


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(medv ~ .)
372
373
413
-2
0
2
4
6
-3
-2
-1
0
1
2
3


Scale-Location
372
373
413
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
0
10
20
30
40
Fitted values
lm(medv ~ .)


Residuals vs Leverage
Standardized residuals
Leverage
lm(medv ~ .)
Cook's distance
373
368
366
1
0.5
-2
0
2
4
6
0.00
0.05
0.10
0.15
0.20
0.25