Lecture on Model Selection
Stat 431, Summer 2012

Hyunseung Kang

July 30/31, 2012

1 Introduction

Consider the multiple regression model with p measurements/predictors/independent variables for n individuals

$$Y_i = \beta_0 + \beta_1 X_{i,1} + ... + \beta_p X_{i,p} + \epsilon_i$$

Throughout last week, we have assumed that the number of Xs in the model were given. In practice, we often have a large number of Xs to choose from. Furthermore, with more measurements , the inference for individual coefficients was difficult to interpret, thanks to collinearity. But, more X measurements gave smaller SSE; more independent measurements meant better predictive power.

In this lecture, we'll talk about various ways of choosing which X to fit in the mode and to obtain the "best" set of Xs. This entire topic is known as *model selection* in the statistical inference literature and is an active area of research (i.e my dissertation topic is on inference for penalized regression).

2 Model Selection Procedures

In model selection, we seek a relationship between Y and $X_{,1}, ..., X_{,p}$ that is parsimonious, but still retains good predictive power, as represented by the SSE. Each subset of $X_{,1}, ..., X_{,p}$ that will be used for the fit is one particular model, denoted as $\mathcal{M}$, for studying the relationship between Y and $X_{,1}, ..., X_{,p}$ and we want to choose a subset of$X_{,1}, ..., X_{,p}$ that achieves this goal. The selection procedures described below attempt to achieve this goal through a variety of methods.

2.1 All-Subsets Selection

Suppose we have p $X_{,j}$'s and we want to choose the best subset of $X_{,1}, ..., X_{,p}$. The most natural way to solve this problem is to *try every subset of* $X_{,1}, ..., X_{,p}$. There will be a total of 2^p subsets because each $X_{,j}$ can either be in the model or not in the model and there are p $X_{,j}$'s. We can measure how good each subset is using the information criterion outlined in the next section and choose the subset with the lowest value.

The R command "regsubsets()" in the "leaps" package implements this procedure. An example for the Boston housing data is shown below.

```
### Installing the "leaps" package and loading it into R ###
install.packages("leaps")
library(leaps)

### Reading the Boston Housing data ###
data = read.csv("http://stat.wharton.upenn.edu/~khyuns/stat431/BostonHousing.txt")
attach(data)

### Run All-Subsets ###
```

```
model.allsubsets = regsubsets(log(MEDV) ~ INDUSTRY + NROOMS + AGE + TAX + CRIM,data=data)

summary(model.allsubsets) # Provides a summary of which X is in the model
# Output #
# Each row corresponds to the best subset of Xs for p number of independent variables
# Selection Algorithm: exhaustive
#          INDUSTRY NROOMS AGE TAX CRIM
# 1  ( 1 ) " "      "*"    " " " " " "
# 2  ( 1 ) " "      "*"    " " " " "*"
# 3  ( 1 ) " "      "*"    " " "*" "*"
# 4  ( 1 ) " "      "*"    "*" "*" "*"
# 5  ( 1 ) "*"      "*"    "*" "*" "*"

summary(model.allsubsets)$cp #Provides Mallow's Cp for each p.
# Output #
# Each value corresponds to the Cp value for each p.
# Choose the value with the lowest Cp, in our case, p =4.
# [1] 283.905095  77.475928  23.309122   4.494921   6.000000

summary(model.allsubsets)$bic #Provides BIC for each p
# Output #
# Each value corresponds to the BIC value for p.
# Choose the value with the lowest BIC, in our case p= 4
# [1] -245.5618 -395.2689 -440.8181 -455.2089 -449.4830

# More details about summary(model.allsubsets) can be found by typing
?summary.regsubsets
```

The greatest feature about this procedure is that we can rank all the possible models from each subset, via the information criterion. However, the greatest caveat is that this procedure is only practical for small p as the number of all possible subsets exponentiate for every increase in p. In almost all modern data sets, p is somewhat large and this procedure is impractical.

2.2 Forward Selection

Forward selection is a *greedy* version of all-subsets where it attempts to find the best ranked model in all-subsets by selectively choosing the subsets; in comparison, all-subsets exhaustively chose every single subset of $X_{,1}, ..., X_{,p}$. The procedure, in a nutshell, is to start with the most basic model, $Y = \beta_0 + \epsilon_i$ and add one predictor at a time until there is no statistically significant difference between adding one more predictor.

1. Start with the most parsimonious model, $Y = \beta_0 + \epsilon_i$.

2. For the current model,

   (a) For each $X_{,j}$ that's left, *add* it[1] to the model and perform an F test comparing the current model (i.e. the reduced model) with the model that includes $X_{,j}$ (i.e. the full model).

   (b) Choose the $X_{,j}$ with the *lowest* p-value (or largest F value)

   (c) If this p-value is *lower* than a pre-specified significance level (e.g. $\alpha = 0.05$), include it into the model, declare this as your current model, and repeat the procedure. Otherwise, declare the current model as your final model.

R command "add1()" implements this procedure. An example using Boston Housing Data is shown below.

[1]For $X_{,j}$ that have been reformatted from a categorical variable, make sure you add *all* the $X_{,j}$'s associated with that categorical variable. DO NOT add only one reformatted $X_{,j}$ associated with the categorical variable

```
### Assume that alpha = 0.05 ###
### First, build the model with only the intercept ###
model.current = lm(log(MEDV) ~ 1)

### Run add1() ###
# Always put all the Xs under consideration in the model selection procedure
add1(model.current,~INDUSTRY + NROOMS + AGE + TAX + CRIM,test="F")
# Output #
# Choose the independent variable with the lowest p-value or the highest F value.
# Here, NROOMS is chosen
Single term additions

Model:
log(MEDV) ~ 1
         Df Sum of Sq    RSS      AIC F value     Pr(F)
<none>                84.376  -904.37
INDUSTRY  1    24.746 59.630 -1078.02  209.16 < 2.2e-16 ***
NROOMS    1    33.704 50.672 -1160.39  335.23 < 2.2e-16 ***
AGE       1    17.347 67.029 -1018.83  130.43 < 2.2e-16 ***
TAX       1    26.599 57.777 -1093.99  232.03 < 2.2e-16 ***
CRIM      1    23.518 60.858 -1067.70  194.76 < 2.2e-16 ***

### Run add1() again ###
# Since the p-value for NROOMS is less than 0.05, we reset our current model
# and repeat the procedure
model.current = lm(log(MEDV) ~ NROOMS)
add1( model.current,~INDUSTRY + NROOMS + AGE + TAX + CRIM,test="F")
# Output #
# Choose the independent variable with the lowest p-value or the highest F value
# Here, CRIM is chosen.
Single term additions

Model:
log(MEDV) ~ NROOMS
         Df Sum of Sq    RSS     AIC F value     Pr(F)
<none>                50.672 -1160.4
INDUSTRY  1    8.6153 42.057 -1252.7 103.038 < 2.2e-16 ***
AGE       1    8.1436 42.529 -1247.0  96.318 < 2.2e-16 ***
TAX       1   13.1026 37.570 -1309.8 175.424 < 2.2e-16 ***
CRIM      1   13.4387 37.233 -1314.3 181.548 < 2.2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

### Run add1() again ###
# Repeat this step until the lowest p-value is greater than 0.05.
# At this point, return the model.current as the best model

### NOTE ###
# Simply running
steps(log(MEDV)~INDUSTRY + NROOMS + AGE + TAX + CRIM,direction="forward")
# will NOT get you the same results since steps() automatically uses AIC, not F tests!
```

A nice feature about this procedure is that it is computationally attractive for large p. In addition, the procedure can be easily adapted to handle cases where $n << p$, often known as the *high dimensional* cases in statistical inference. However, like backward and stepwise selection, it suffers from not being able to search through all possible models based on the given X's and thus, ranking these models become difficult. In addition, at each step when new predictors are added, the procedure does not take into account the parsimony of the model (i.e. the number of independent variables in the model); the F test, for all practical purposes, looks at the difference in $SSE_{reduced} - SSE_{full}$ (granted, this can be easily solved by using an information criterion (see the next paragraph)). Finally, this procedure only allows *nested models* to be compared. For example, this procedure cannot compare whether

$$Y_i = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \epsilon_i$$

is better/worse than

$$Y_i = \beta_0 + \beta_3 X_{i,3} + \beta_4 X_{i,4} + \epsilon_i$$

The procedure can only compare $Y_i = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \epsilon_i$ to $Y_i = \beta_0 + \beta_1 X_{i,1} + \epsilon_i$, $Y_i = \beta_0 + \beta_2 X_{i,2} + \epsilon_i$, or $Y_i = \beta_0 + \epsilon_i$, nested version of $Y_i = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \epsilon_i$.

Finally, instead of using the F-test to evaluate the significance of adding a predictor, we can use the information criterion. Specifically, we would compute the information criterion for the current model and for each model with one of the terms added. We would only add the $X_{,j}$ if the information criterion associated with it is smaller than the current model. Otherwise, we would declare the current model as our final model.

2.3 Backward Selection

Backward selection is another greedy procedure that attempts to find the best ranked model from all-subsets selection. It does this by selectively removing $X_{,j}$'s from the full model, $Y_i = \beta_0 + \beta_1 X_{i,p} + ... + \beta_p X_{i,p}$ until there is a statistically significant difference between dropping an $X_{,j}$ and retaining that $X_{,j}$.

1. Start with the least parsimonious model, $Y_i = \beta_0 + \beta_1 X_{i,p} + ... + \beta_p X_{i,p} + \epsilon_i$.
2. For the current model,
   (a) For each $X_{,j}$ that's in the current model, *remove* it[2] from the model and perform an F test comparing the current model (i.e. the full model) with the model that does not include the $X_{,j}$ (i.e. the reduced model).
   (b) Choose the $X_{,j}$ with the *largest* p-value (or smallest F value)
   (c) If this p-value is *larger* than a pre-specified significance level (e.g. $\alpha = 0.05$), remove $X_{,j}$, declare this as your current model, and repeat the procedure. Otherwise (i.e when removing a predict creates a significant difference where p-value is less than 0.05), declare the current model (with that $X_{,j}$ in it) as your final model.

R command "drop1()" implements this procedure. An example using Boston Housing Data is shown below.

```
### Assume that alpha = 0.05 ###
### First, build the model with only the intercept ###
model.current = lm(log(MEDV) ~ INDUSTRY + NROOMS + AGE + TAX + CRIM)

### Run drop1() ###
drop1(model.current,test="F")
# Output #
# Choose the independent variable with the highest p-value or the lowest F value.
# Here, INDUSTRY is chosen to be taken out.
Single term deletions

Model:
```

[2]For $X_{,j}$ that have been reformatted from a categorical variable, make sure you remove *all* the $X_{,j}$'s associated with that categorical variable. DO NOT remove only one reformatted $X_{,j}$ associated with the categorical variable

```
log(MEDV) ~ INDUSTRY + NROOMS + AGE + TAX + CRIM
         Df Sum of Sq    RSS     AIC  F value     Pr(F)
<none>                32.238 -1381.2
INDUSTRY  1    0.0319 32.270 -1382.7   0.4949 0.4820677
NROOMS    1   16.1974 48.436 -1177.2 251.2152 < 2.2e-16 ***
AGE       1    0.8919 33.130 -1369.4  13.8330 0.0002224 ***
TAX       1    1.0323 33.270 -1367.3  16.0112 7.250e-05 ***
CRIM      1    3.6262 35.864 -1329.3  56.2408 2.946e-13 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

### Run drop1() again ###
# Since the p-value for INDUSTRY  is greater than 0.05, we reset our current model
# without INDUSTRY and repeat the procedure
model.current = lm(log(MEDV) ~  NROOMS + AGE + TAX + CRIM)
drop1( model.current,test="F")
# Output #
# Choose the independent variable with the highest p-value or the lowest F value
# Here, AGE is chosen. However, since the p-value for AGE is less than 0.05
# we declare the current model log(MEDV) ~ NROOMS + AGE + TAX + CRIM to be
# our final model.
Single term deletions

Model:
log(MEDV) ~ NROOMS + AGE + TAX + CRIM
       Df Sum of Sq    RSS     AIC F value     Pr(F)
<none>              32.270 -1382.7
NROOMS  1   17.7340 50.004 -1163.1 275.324 < 2.2e-16 ***
AGE     1    1.3420 33.612 -1364.1  20.835 6.305e-06 ***
TAX     1    1.7801 34.050 -1357.5  27.637 2.171e-07 ***
CRIM    1    3.5944 35.864 -1331.3  55.804 3.587e-13 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

### NOTE ###
# Simply running
step(log(MEDV)~INDUSTRY + NROOMS + AGE + TAX + CRIM,direction="backward")
# will NOT get you the same results since steps() automatically uses AIC, not F tests!
```

Backward selection has the same pros as forward selection, except that it cannot handle $n << p$ gracefully. It also has the same cons as forward selection, mainly that it cannot rank all the possible models that can be derived from the given $X_{,1}, ..., X_{,p}$'s

Finally, instead of using the F-test to evaluate the significance of removing predictor, we can use the information criterion. Specifically, we would compute the information criterion for the current model and for each model with one of the terms removed. We would only drop the $X_{,j}$ if the information associated with it is smaller than the current model. Otherwise, we would declare the current model as our final model. AIC is implemented by default, but you can ask it to run BIC

2.4 Stepwise Selection

Stepwise selection is a combination of forward and backward selection. It attempts to find the best-ranked model from all-subset selection baesd on combining the adding $X_{,j}$ and removing $X_{,j}$ in the following manner.

1. Start with the some model. In R, this usually is the least parsimonious model: $Y_i = \beta_0 + \beta_1 X_{i,1} + .. + \beta_p X_{i,p} + \epsilon_p$

2. For the current model, compute the information criterion.
   (a) Consider all the variables that can be removed. For each of the variables removed, compute the information criterion.
   (b) Consider all the variables that can be added. For each of the variables added, compute the information criterion
   (c) From the models formulated by removing or adding predictors, choose the model with the smallest information criterion.
   (d) If the information criterion associated with this model has a smaller value than current model, replace the current model with this one. Otherwise, declare the current model as the final model.

R command "step" implements this procedure. An example using Boston Housing Data is shown below.

```
### We'll use AIC as our information criterion ###
### First, start with the least parsimonious model ###
model.current = lm(log(MEDV) ~ INDUSTRY + NROOMS + AGE + TAX + CRIM)

### Run step###
model.stepwise = step(model.current,direction="both",test="F")
# Output #
# - next to the the indep. variable indicates removal of that variable
# + indicates adding that variable. By default, AIC is computed and
# used as the criterion.
Start:  AIC=-1381.21
log(MEDV) ~ INDUSTRY + NROOMS + AGE + TAX + CRIM

           Df Sum of Sq    RSS     AIC  F value     Pr(F)
- INDUSTRY  1    0.0319 32.270 -1382.7   0.4949 0.4820677
<none>                  32.238 -1381.2
- AGE       1    0.8919 33.130 -1369.4  13.8330 0.0002224 ***
- TAX       1    1.0323 33.270 -1367.3  16.0112 7.250e-05 ***
- CRIM      1    3.6262 35.864 -1329.3  56.2408 2.946e-13 ***
- NROOMS    1   16.1974 48.436 -1177.2 251.2152 < 2.2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Step:  AIC=-1382.71
log(MEDV) ~ NROOMS + AGE + TAX + CRIM

           Df Sum of Sq    RSS     AIC  F value     Pr(F)
<none>                  32.270 -1382.7
+ INDUSTRY  1    0.0319 32.238 -1381.2   0.4949    0.4821
- AGE       1    1.3420 33.612 -1364.1  20.8352 6.305e-06 ***
- TAX       1    1.7801 34.050 -1357.5  27.6366 2.171e-07 ***
- CRIM      1    3.5944 35.864 -1331.3  55.8045 3.587e-13 ***
- NROOMS    1   17.7340 50.004 -1163.1 275.3241 < 2.2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

### NOTE ###
# The code below will use the BIC instead of the AIC in the stepwise procedure.
n = nrow(data)
step(log(MEDV)~INDUSTRY + NROOMS + AGE + TAX + CRIM,direction="both",k=log(n))
```

Stepwise selection has the same pros as backward and forward selection. It also has the same cons as forward selection, mainly that it cannot rank all the possible models that can be derived from the given $X_{,1}, ..., X_{,p}$'s

Finally, instead of starting with the most parsimonious model, we can technically start with the least parsimonious model. As long as the procedure knows the number of variables available in the model (i.e. by specifying the most parsimonious model), this shouldn't be a problem for the stepwise algorithm

2.5 Cross-Validation

Cross-validation (CV) is another model selection technique where we divide up the data by *training sample* and a *testing sample*. The training sample is used to fit the regression model while the testing sample is used to test how accurate the model is.

K-fold CV is where we randomly divide data into K parts and where each part takes turns serving as the training sample[3]. The algorithm can be described as follows

1. Randomly divide your observations $1, 2, ..., n$ into K parts. Each part should have roughly n/K observations.
2. For each part
   (a) Define this part to be your testing sample.
   (b) Define all other parts to be your training sample.
   (c) Fit the regression using only the training sample.
   (d) Compute the prediction MSE, denoted as $PMSE$

$$PMSE = \frac{1}{\text{\# of testing samples}} \left(\sum_{i \in \text{testing sample}} (Y_i - \hat{Y}_i)^2 \right)$$

3. Take the average of the K PMSE computed in the loop.

For possible models in consideration, we compute the K-fold CV and obtain the PMSE for each model. We generally choose the model with the smallest PMSE. In practice, K is chosen to be 5, although it can depend on the initial sample size.

Cross-validation is useful if you are mainly interested in selecting a model with good prediction power since it will choose the model that has good *out-of-sample* error. It also works with any type of modeling/fitting procedures, even those that aren't multiple regression. However, without a large dataset, cross validation suffers from highly biased measures of performance since we have to sacrife $1/K$ amount of data for testing the peformance of the model. In your last homework, you will implement this procedure in R.

2.6 Penalized Regression

Penalized regression does model selection by penalizing for large values of $\hat{\beta}_j$ in the optimization problem that helped us obtain the estimates for $\hat{\beta}_j$.By penalizing for large values of $\hat{\beta}_j$, the optimization problem is forced to *shrink* some of the $\hat{\beta}_j$ and set them close to zero. If a $\hat{\beta}_j = 0$ from running a penalized regression procedure, we know that its associated $X_{,j}$ does not play a role in explaining the variation on Y. Hence, non-zero $\hat{\beta}_j$ indicate that their associated $X_{,j}$'s are *important* variables in the regression while zeroed $\hat{\beta}_j$ indicate that their associated $X_{,j}$'s are *not important*.

Mathematically, the penalized regression procedure can be written as an optimization problem:

$$\min_{\beta_0, \beta_1, ..., \beta_p} \underbrace{\sum_{i=1}^{n} (Y_i - (\beta_0 + \beta_1 X_{i,1} + ... + \beta_p X_{i,p}))^2}_{\text{minimize residuals}} + \lambda \underbrace{\text{pen}(\beta_0, \beta_1, ..., \beta_p)}_{\text{penalty for} \beta_j\text{'s}}$$

[3]A special case of K-fold CV is *leave-one-out CV*, where $K = n$.

The first part of the equation, the summand part, is identical to the multiple regression optimization problem. The second part, the pen(), is a non-negative function of β_js that penalizes for large β_js. $\lambda > 0$ controls the degree to which the penalty function should influence the optimization.

Common penalty functions are listed below, along with their properties

1. *L_0 or the BIC Penalty:* The L_0 penalty function is define as

$$pen(\beta_0, ..., \beta_p) = \sum_{j=0}^{p} I(\beta_j = 0)$$

   Unfortunately, optimization with this penalty function is computationally infeasible. Despite this, there is a lot of nice theoretical properties if the optimization can be done in time.

2. *L_1 or Lasso Penalty:* The L_1 penalty function is defined as

$$pen(\beta_0, ..., \beta_p) = \sum_{j=0}^{p} |\beta_j|$$

   This penalty function has a tendency to set $\hat{\beta}_j$s associated with non-important $X_{,j}$s to zero. This penalty has the nice feature that it estimates $\hat{\beta}_j$ as well as selects which $X_{,j}$'s are important at the same time. Finally, this penalty is computationally tractable.

3. *L_2 or Ridge Penalty:* The L_2 penalty function is defined as

$$pen(\beta_0, ..., \beta_p) = \sum_{j=0}^{p} \beta_j^2$$

   This penalty function has a tendency to shrink $\hat{\beta}_j$s close to zero, but not necessarily equal to zero. This penalty has the nice feature that you can take partial derivatives and actually derive a closed-form formula for the $\hat{\beta}_j$ expression and compute its value very quickly. Unfortunately, it is unable to "select" which $X_{,j}$'s are important by setting $\hat{\beta}_j$s that are associated with unimportant $X_{,j}$s to zero.

The great benefit about penalized regression is that it can handle $n << p$ gracefully. In addition, it is purely an optimization problem. Unlike previous procedures where they relied on computing the information criterion or evaluating F test, the model selection is simply optimizing a couple of variables. Tons of modern statistical research is on penalized regression, in particular on the L_1 penalty.

However, the downside about penalized regression is its difficult to obtain exact inference procedures. That is, it is very difficult, or often impossible, to obtain sampling distributions for the estimates obtained from the optimization procedures. This is the topic of my thesis, on developing inference procedures for penalized regression, especially the L_1 regression. So far, I have made absolutely no progress.

2.7 Inference after Model Selection

It is important to remember that model selection procedure is a stochastic, random procedure because *we draw random samples from the population.* If we drew a different $(Y_i, X_{i,1}, ..., X_{i,p})$, we may end up with an entirely different final model when we run our model selection procedures. Hence, when we test for coefficients with the model selected by the procedure, we must account for the randomness associated with not only the sample, but also the procedure that generated this model. This is known as *inference after model selection*, another popular research topic in this statistics department.

3 AIC, BIC, and Mallow's C_p

There are several ways to evaluate whether certain model is better than another model. Here, we define a "better" model to be one that is parsimonious in the number of predictors (i.e. p) and one that has smaller residuals (i.e. small SSE). There are three possible numerical evaluations based on this definition of "best", , stated as definitions below

Definition 3.1. *Akaike Information Criterion* (or AIC) for a model $\mathcal{M}$ is defined as

$$AIC(\mathcal{M}) = n\log(SSE_{\mathcal{M}}/n) + 2(p_{\mathcal{M}} + 1)$$

where p_M is the number of predictors in the model.

Remark 3.1. AIC is the most commonly used evaluator. It is used in the R command "step()".

Definition 3.2. *Bayesian Information Criterion* (or BIC) for a model $\mathcal{M}$ is defined as

$$BIC(\mathcal{M}) = n\log(SSE_{\mathcal{M}}/n) + \log(n)(p_{\mathcal{M}} + 1)$$

where p_M is the number of predictors in the model.

Remark 3.2. BIC has a $log(n)$ penalty for the number of predictors, in contrast to 2 in the AIC. This implies that $AIC(\mathcal{M}) \geq BIC(\mathcal{M})$ for $\exp(2) \approx 7.3 \leq n$. Practically speaking, BIC puts a harsher penalty for having more independent variables than AIC and model selection procedures that use BIC has a tendency to choose models with few independent predictors than those that use AIC.

Definition 3.3. *Mallow's C_p* (or C_p) for a model $\mathcal{M}$ is defined as

$$C_p(\mathcal{M}) = SSE_{\mathcal{M}}/MSE_{full} + 2(p_{\mathcal{M}} + 1) - n$$

where MSE_{full} is the MSE of the full model (i.e. the mode with all the X's) and p_M is the number of predictors in the model.

A couple of remarks about all the definitions

1. For each criterion, a smaller value represents a *better* the model. Here, we define a "better" model to be one that is parsimonious in the number of predictors (i.e. $p_{\mathcal{M}}$), but still have small residuals (i.e. $SSE_{\mathcal{M}}$) with those set of predictors.

2. With each criterion, we can rank which model is better than the other.

3. If the number of predictors is fixed, each criterion is merely a function of SSE.

We can use any one of the information criterion in the model selection procedure as a measure of how "good" various models are.

3.1 Theoretical Justification of the Information Criterions

From the previous section, we konw that a small value for an information criterion indicates a "better" model. For the "best" model, the AIC, BIC, and/or Mallow's C_p should give the smallest value amongst all possible models under consideration in our model selection procedure; otherwise the information criterion is a faulty measure of how "good" a model is and the procedures that use these criterions may choose the wrong final model.

In this section (if I have time), we'll briefly explore how these information criterion gives the smallest value for the "best" model. Specifically, we'll justify, with mathematical theory, that, under large sample size and under certain conditions on β_j's, the best model gives the lowest AIC, BIC, and Mallow's C_p values.