**Chapter 5 – Model selection and evaluation**

The purpose of this chapter is

* Determine the most parsimonious model that best estimates the response variable
* Assess how well the model fits the data

There will be many similarities here to what you would learn in a normal linear regression analysis course.

**Section 5.1 – Variable selection**

In some situations, you will have a specific number of explanatory variables where you want to include all of them in the model. For example, in a designed experiment setting, you may have three explanatory variables with all of them needing to be in the model.

In other situations, due to knowledge of the subject matter, the specific variables and their form may lead to a particular model. For example, explanatory variables that are “non-significant” may still be included in the model.

In contrast, you may have access to a number of explanatory variables and you do not know which to include in a model. This includes observational studies such as with placekicking example. Of course, you would only like to choose those explanatory variables that are “important” rather than those that do not help estimate the response variable. The purpose of this section is to examine methods that help in this regard.

Information criteria

Suppose you would like to determine which of two estimated models are “best” to use for a data set. A LRT can be used, but one model needs to be “nested” within the other (all variables in one model need to be within the other model).

A residual deviance or even just a likelihood function evaluated at the parameter estimates could be used to compare the two models. You may think that the smaller the residual deviance (or larger the likelihood function), the “better” the model. Unfortunately, this is not good to do for similar reasons as why R2 generally should not be used to compare two models. What is this reason!?

Information criteria avoid these problems by using the likelihood function and adding a “penalty” to it. The general form is

IC(k) = -2log(Likelihood function evaluated at parameter estimates) + kr

where k is a numerical penalty and r is the total number of parameters in the model. The most common IC(k) are:

1. Akaike’s Information Criterion



1. Corrected AIC



1. Bayesian Information Criterion (BIC)



Comments:

* See Burnham and Anderson (2004) and Emiliano et al. (2014) for discussion
* 2n/(n-r-1) > 2; for large n relative to r, 2n/(n-r-1) ≈ 2
* log(n) > 2 when n ≥ 8

With all three information criteria, the best estimated model has the lowest IC(k). Why?

Comments:

* AIC tends to favor larger models than AICc and BIC. This can be good or bad. Why?
* What if a large number of models have very similar IC(k) values?
* Which information criteria is best? There are statistical evaluation methods to examine “best” and these are described in my book. Overall, one information criterion is not necessarily best for all statistical evaluation methods.
* The AIC() function can be used to calculate the information criteria, where the k argument is used to specify the appropriate value of k. As we will see shortly, many R functions calculate IC(k) as part of their regular computations (e.g., glm()).
* Different R functions may give numerical values of an IC(k) that are unequal, even if the same model is used for the calculation. The reason is due to computing the log-likelihood function without parts that are constant across any model. The same conclusions about which model is best will still be reached as long as the log-likelihood function portion of the IC(k) is computed in the same way (with or without the constant values) across these models.

Example: The  part of the Poisson log-likelihood function will not change due to different models. This may or may not be included in the calculation of IC(k).

All-subsets regression

Now that we have numerical measures to decide which model is “best”, we need to determine which models to compare. One way is to compare “all possible” models that one could have!

Simply, if you have P total explanatory variables to choose from, one could have 2P different models (this depends – more on this soon). For example, if there are P = 3 explanatory variables x1, x2, and x3 to consider, the possible models include:

1. No explanatory variables
2. x1
3. x2
4. x3
5. x1, x2
6. x1, x3
7. x2, x3
8. x1, x2, x3

Thus, 23 = 8 possible models. From these models, the “best” model is the model with the smallest IC(k). Also, because “subsets” of all possible variables are considered, this method is referred to as all-subsets regression.

We are NOT really considering ALL possible models here. For example, what about interactions or various transformations of variables? Thus, it is somewhat a misnomer to really think this examines all possible models. However, the approach does at least provide a good way to start to determine which models are best.

Of course, 2P can become large quickly as P grows. This can lead to computational problems. We will discuss an approach to get around these problems shortly.

Three functions for implementing all-subsets regression are:

* dredge() of the MuMIn package
  + Up to 30 explanatory variables
  + Can be slower than other functions
* bestglm() function of the bestglm package
  + Up to 15 explanatory variables
  + Does not compute AICc
  + Poor syntax for arguments and components
* bic.glm() of the BMA package
  + Up to 50 explanatory variables
  + Only computes BIC

We will focus on dredge() in these notes.

Example: Placekicking (AllSubsetsPlacekick.R, Placekick.csv)

How many models are there to examine?

> placekick <- read.csv(file = "C:\\data\\Placekick.csv")

> head(placekick)

week distance change elap30 PAT type field wind good

1 1 21 1 24.7167 0 1 1 0 1

2 1 21 0 15.8500 0 1 1 0 1

3 1 20 0 0.4500 1 1 1 0 1

4 1 28 0 13.5500 0 1 1 0 1

5 1 20 0 21.8667 1 0 0 0 1

6 1 25 0 17.6833 0 0 0 0 1

Output from dredge():

> mod.fit <- glm(formula = good ~ ., family = binomial(link = "logit"), data = placekick, na.action = na.fail)

> library(MuMIn)

> allsub.aic <- dredge(global.model = mod.fit, rank = "AIC")

Fixed term is "(Intercept)"

> allsub.aic[allsub.aic$delta < 1,]

Global model call: glm(formula = good ~ ., family = binomial(link = "logit"), data = placekick,

na.action = na.fail)

---

Model selection table

(Intrc) chang dstnc PAT week wind df

148 4.752 -0.3351 -0.08724 1.230 -0.5234 5

212 4.990 -0.3478 -0.08723 1.228 -0.02468 -0.4765 6

84 4.950 -0.3538 -0.08653 1.243 -0.02773 5

20 4.676 -0.3402 -0.08646 1.245 4

147 4.599 -0.08676 1.319 -0.5326 4

logLik AIC delta weight

148 -378.343 766.7 0.00 0.256

212 -377.537 767.1 0.39 0.211

84 -378.629 767.3 0.57 0.192

20 -379.666 767.3 0.65 0.185

147 -379.831 767.7 0.98 0.157

Models ranked by AIC(x)

Comments:

* A model needs to be estimated first that contains all of the explanatory variables. We estimate this model using glm(). The period in formula = good ~ . represents all of the variables in the data frame, excluding the one used for the response variable.
* The na.action = na.fail argument in glm() is needed. This indicates that the model fitting should fail if there are missing data values. One needs to include this for the subsequent dredge() function to work.
* The dredge() function estimates all possible models and computes the AIC for each model.
* The “best” model as given by the AIC is model #148 that includes change, distance, PAT, and wind. The delta value in allsub.aic is the change in the AIC from the best model.
* The AIC values are relatively close, indicating some uncertainty about which model is best.

When there are a larger number of explanatory variables and/or interactions of interest, it is best to use a different approach to examine the models. The genetic algorithm does not look at “all possible” models but it does a good job of exploring potential models. It works as follows:

1. Random sets of explanatory variables are put together into models. This is called the “first generation” of models.
2. From the best performing of these models, new models are put together in a random manner to form a “second generation” of models.
3. This process continues for multiple generations with random additions and deletions (*mutations*) of explanatory variables to determine if better models can be formed.
4. Eventually the algorithm converges to a “best” model.

Due to the stochastic nature of the algorithm (notice the use of the word “random” above), different final models could result. Therefore, it is best to run the algorithm more than once to determine if even better models can be found.

The glmulti() function of the glmulti package performs all-subsets regression and the genetic algorithm. Unfortunately, there are drawbacks to its use:

* The package uses a separate Java program for its computation, so it requires a Java installation on the computer to use it. This Java installation needs to be a 32- or 64-bit that matches the 32- or 64-bit version of R that you are using.
* The set.seed() function does not work to set a seed so this prevents one from reproducing the exact same results. The likely reason is because the random number aspect is set within the Java program it uses.

Example: Placekicking (AllSubsetsPlacekick.R, Placekick.csv)

Key arguments within glmulti() include:

* method: Examines all possible models (exhaustive search, method = "h") or implement the genetic algorithm (method = "g")
* Allows for the specification of interactions that respect the principle of marginality (level = 2, marginality = TRUE)

Below is how glmulti() is used with the genetic algorithm.

> library(glmulti)

> search.gmarg.aic <- glmulti(y = good ~ ., data = placekick, fitfunction = "glm", level = 2, marginality = TRUE, method = "g", crit = "aic", family = binomial(link = "logit"))

Initialization...

TASK: Genetic algorithm in the candidate set.

Initialization...

Algorithm started...

After 10 generations:

Best model: good~1+week+distance+elap30+PAT+type+field+wind+

distance:week+elap30:week+PAT:week+type:week+

type:elap30+field:week+field:distance+

wind:distance+wind:type+wind:field

Crit= 776.68216003651

Mean crit= 786.54504175157

Change in best IC: -9223.31783996349 / Change in mean IC: -9213.45495824843

<OUTPUT EDITED>

> print(search.gmarg.aic)

glmulti.analysis

Method: g / Fitting: glm / IC used: aic

Level: 2 / Marginality: TRUE

From 100 models:

Best IC: 763.846611167118

Best model:

[1] "good ~ 1 + week + distance + change + PAT + field + wind + wind:distance + "

[2] " wind:field"

[1] "good ~ 1 + week + distance + change + PAT + field + wind + PAT:distance + "

[2] " wind:distance + wind:field"

Evidence weight: 0.0293947456382157

Worst IC: 773.376224111534

46 models within 2 IC units.

74 models to reach 95% of evidence weight.

Convergence after 730 generations.

Time elapsed: 11.1725265145302 minutes.

Comments:

* Including the pairwise interactions leads to 2P+A possible models, where A = P(P – 1)/2 is the number of interactions. Thus, there are 236 different models here!
* It is best to only include those interactions among explanatory variables that make sense in the context of the problem. Unfortunately, there is not a way in glmulti() to exclude specific interactions (the exclude argument for the function does not work properly).
* A model is specified in the y argument to obtain the response and explanatory variables. This does not mean that ONLY this model will be evaluated!
* The level argument is equal to 1 for main effects and 2 for the main effects with interactions.
* The algorithm converged after 730 generations, meaning that it did not find any better models to add to its top 100 for 50 consecutive generations.
* Oddly, there was a tie for the best model. The only difference between the two models was the PAT:distance interaction. We will discuss a unique aspect of the interaction later in this chapter.

Below is a summary of the top 5 models.

> head(weightable(search.gmarg.aic))

model

1 good ~ 1 + week + distance + change + PAT + field + wind + wind:distance + wind:field

2 good ~ 1 + week + distance + change + PAT + field + wind + PAT:distance + wind:distance + wind:field

3 good ~ 1 + distance + change + PAT + field + wind + PAT:distance + wind:distance + wind:field

4 good ~ 1 + distance + change + PAT + field + wind + wind:distance + wind:field

5 good ~ 1 + week + distance + change + PAT + field + wind + PAT:distance + wind:week + wind:distance + wind:field

aic weights

1 763.847 0.029

2 763.847 0.029

3 763.992 0.027

4 763.992 0.027

5 764.014 0.027

The AIC values are relatively close, indicating some uncertainty about which model is best.

The same code was run 4 times due to the random aspects of the genetic algorithm. The same three best models are found through four separate runs.

When some models have IC(k) values similar to each other, this is a sign that uncertainty exists about which model is “best”. An option then is to not choose one best model overall but rather use many models. This is the purpose of model averaging. Chapter 5 contains a section about this alternative approach.