Variable Selection and Model Validation

Dr. Kiah Wah Ong

Introduction

Previously, when we perform our regression, we assume that:

- all our predictors are identified in advanced.
- we have a very good idea of the basic form of the model.

In reality, there could be many predictors to consider and we need to figure out what to include in the model.

Also, there could be more than one candidate model that pass the usual diagnostics test and can be applied to our data.

What should we do?

Variable Selection

Finding an appropriate subset from a pool of candidate predictors for our model is called the **variable selection problem.**

In controlled setting, such as clinical trials, variable (subset) selection is usually not necessary.

However, in observational studies, we may collect a large number of potential predictive variables without the knowledge to tell which variables are important and should be included in our regression studies.

We want to determine the "best" predictors to include. Next we present a variable search procedure for this problem.

Automated Model Search: Full Search

Given predictor variables, we run a a full search. For example:

Suppose we have 3 candidate predictors x_1, x_2 and x_3, then we have $2^3 = 8$ possible regression model we can consider, namely:

- the one (1) model with no predictors.
- the three (3) models with only one predictor each.
- the three (3) models with two predictor each.
- the one (1) model with all three predictors.

We then use some statistical measures to pick out the "best" option.

This method can quickly run out of control as there are 2^k possible regression models containing k predictors.

Automated Model Search: Stepwise Regression

There are two types of stepwise regression:

- Forward stepwise regression: Add terms one at a time.
- Backward stepwise regression: Start with the full model and then remove terms.

Automated Model Search: Forward Stepwise Regression

First, we need to set a significant level for deciding when to enter a predictor into the stepwise model and when to remove it.

We let α_E be the significance level to enter a variable, while α_R for removing a predictor.

Hence to start the process, we first:

- Specify α_E. This is typical greater than 0.05 level so that it's not too difficult to enter predictors into the model. Many software has the default set as $\alpha_E = 0.15$
- Specify α_R. Again, this is usually set as $\alpha_R = 0.15$ by default in many statistical package.

Automated Model Search: Forward Stepwise Regression

Step 1: Once we have specified α_E, we then:

(1) Fit each of the one-predictor models, i.e., regress y on x_1, regress y on x_2, ... , and y on x_k.

We will get the coefficient for that (one) variable together with the t-test p-value.

(2) Of those predictors whose t-test p-value is less than $\alpha_E = 0.15$, the first predictor that we included will the the predictor that has the smallest t-test p-value.

This will be one of the most significant single term and we will add that to our model.

(3) If no predictor has a t-test p-value less than $\alpha_E = 0.15$, we terminate the process.

Automated Model Search: Forward Stepwise Regression

Step 2:

(1) Suppose x_1 has the smallest p-value below $\alpha_E = 0.15$ and hence being considered as the "best" single predictor arising from the first step.

(2) Next we regress y on x_1 and x_2, regress y on $x1$ and x_3 ... until y regress on x_1 and x_k.

(3) Of those predictors whose t-test p-value is less than $\alpha_E = 0.15$, the second predictor that we selected will be the one with the smallest t-test p-value.

(4) If no predictor has a t-test p-value less than $\alpha_E = 0.15$, we terminate the process.

Automated Model Search: Forward Stepwise Regression

Terminating steps:

We repeat the processes in Step 1 and 2 until adding additional predictor does not yield a t-test p-value below $\alpha_E = 0.15$.

Remark: We have to be careful dealing with the t-test p-value from Steps 2 onward. When we added a new variable say x_2 in Step 2, the t-test p-value of x_1 will be affected.

If the t-test p-value for $\beta_1 = 0$ has become not significant (p-value greater than α_R), we need to remove x_1 from the stepwise model.

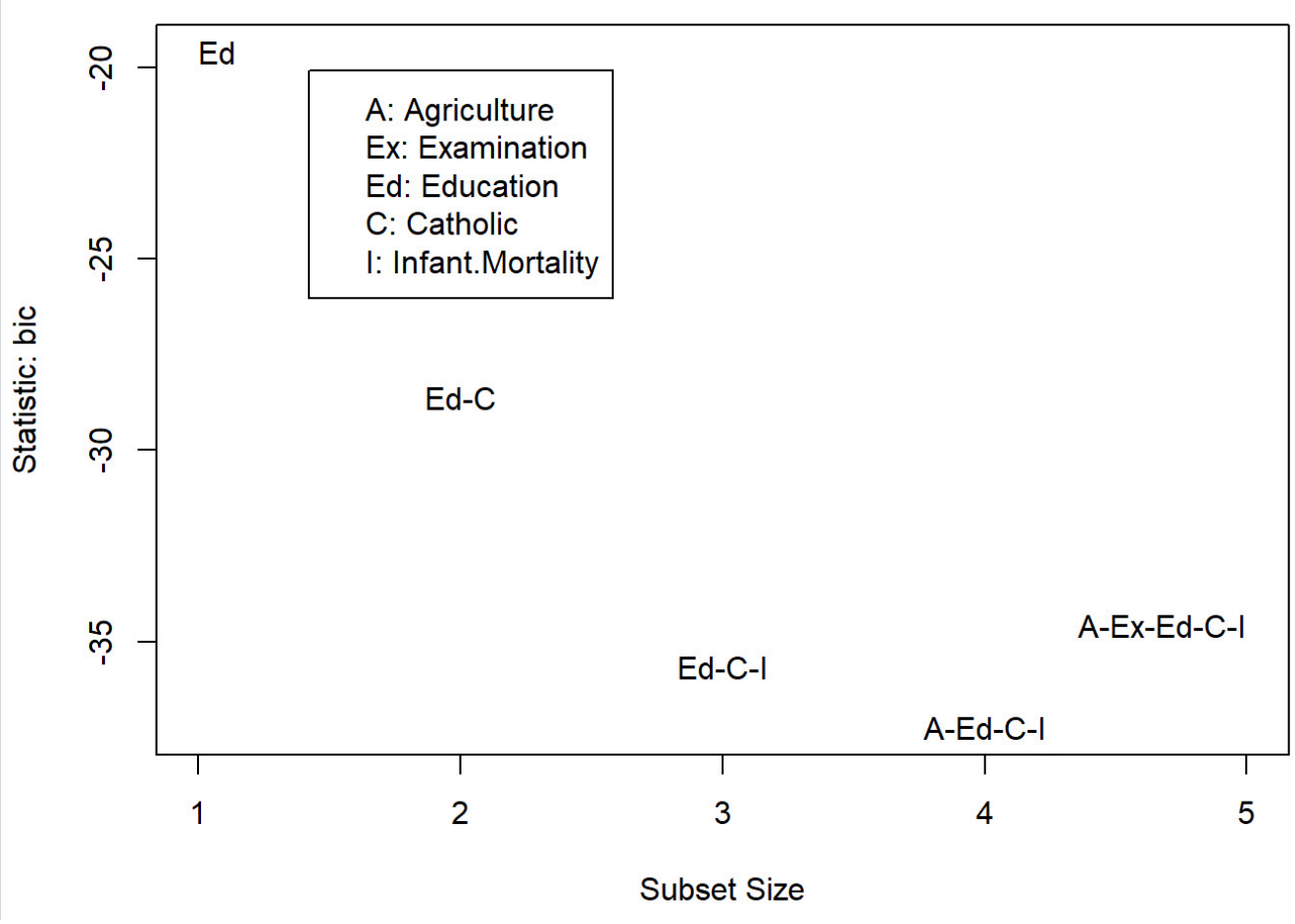
Remark: Backward stepwise regression is done in a similar manner.

Automated Model Search in R

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

library(datasets); data(swiss)
head(swiss)

library(leaps)
library(car)

#nbest = number of best subsets of each size to keep in the result (default=1)
#nvmax = maximum size of subsets to examine

vselect1<-regsubsets(Fertility~.,data=swiss, nbest=1, method="forward", nvmax=5)

#plot statistic by subset size (rsq, cp, adjr2, bic, rss)
subsets(vselect1, statistics="bic")
```

Remark: Bayesian Information Criterion is based on information theory, it places a greater penalty on adding regressors as the sample size increases.

Automated Model Search in R



Example: The 4 predictors variable for this forward search is

$$\text{Fertility} \sim \text{Agriculture} + \text{Education} + \text{Catholic} + \text{InfantMortality}$$

Automated Model Search in R

The backward stepwise regression and the full search is shown below.

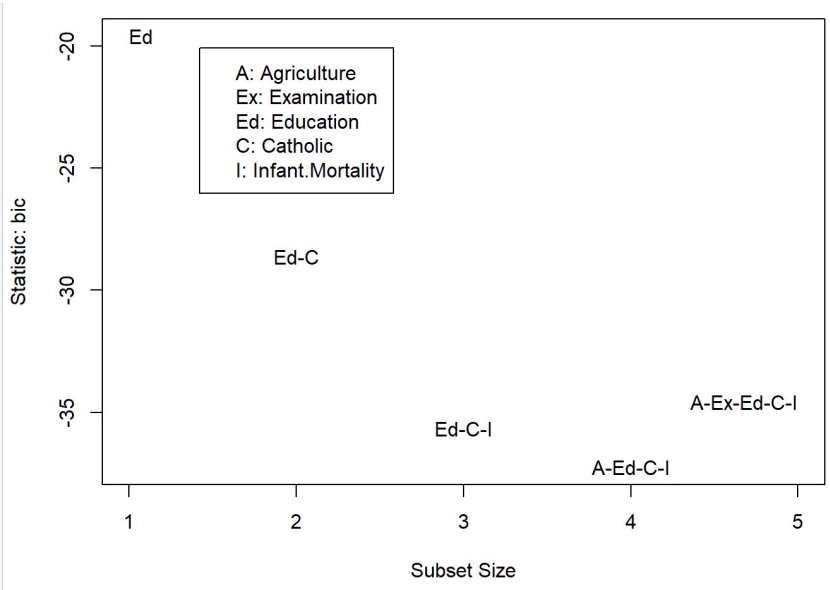
```
vselect2<-regsubsets(Fertility~.,data=swiss, nbest=1, method="backward", nvmax=5)
subsets(vselect2, statistics="bic")

vselect3<-regsubsets(Fertility~.,data=swiss, nbest=1, method="exhaustive", nvmax=5)
subsets(vselect3, statistics="bic")
```

Automated Model Search in R

In this case, all three methods gave the same variables selection.

However, the three procedures do not necessarily lead to the same final model.

Also, none of them guarantees to find the best subset regression model.



Model Validation

After setting up a regression model, there are some important questions to consider, such as:

- How much can we trust our regression model?
- Suppose we've set up a model from a training data, but how well the model is going to perform given a different set of data?

We need to perform model validation to verify that our model is performing as expected.

Model Validation

One way to run model validation is to collect new data after the model has been trained, i.e. the model has been fitted to an existing data set.

If the existing model gives realistic predictions for the new data, we can then have confidence in the validity of the model.

This validation method is the most effective, but could be costly, time consuming or impossible.

Model Validation

A more cost effective way of performing model validation is to perform data splitting.

That is, we partition the data into two parts:

- Training data: this is use for fitting the model
- Validation data: this is use for evaluating the fitted model.

Remark: Usually the training set is larger than the validation set to ensure sufficient precision.

k-Fold Cross Validation

One commonly used validation method is known as k-fold cross validation and it is carried out in the following manner:

- We randomly split the original data into k sub-samples.
- In the first iteration, the first subset is used as the test subset while all the other subsets ($k - 1$ of them) are used as a training set.
- We then train the model with the training data and evaluate it using the test subset. The evaluation score or error rate is kept and then discard the model.

k-Fold Cross Validation

- In the next iteration, using the second subset as the test set, we make the rest of the subsets ($k-1$ of them) as training sets.
- We retrain the model with the training sets and then test it with the new test set. The evaluation score or error rate is kept and then discard the model.
- Continue iterating the steps k times.

k-Fold Cross Validation in R

Here is an example showing how to fit a multiple linear regression model to the data set $\mathrm{Mv1}$ and perform a k-fold cross validation with $k = 5$.

```
library(caret)

mydata<-read.csv("Mv1.CSV", header=TRUE, sep=",")
x1<-mydata$x1
x2<-mydata$x2
x3<-mydata$x3
x4<-mydata$x4
y<-mydata$y

#Specify the cross-validation method
ctrl<-trainControl(method="cv", number=5)

#fit a regression model and use k-fold CV to evaluate performance
model1<-train(y~x1+x2+x3+x4, data=mydata, method="lm", trControl=ctrl)

#view summary of the k-fold CV
print(model1)
```

k-Fold Cross Validation in R

Here is the R-ouput:

```
> print(model1)
Linear Regression

200 samples
  4 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 160, 160, 160, 160, 160
Resampling results:

  RMSE       Rsquared   MAE
  0.2075788  0.9995821  0.1694695

Tuning parameter 'intercept' was held constant at a value of TRUE
```

Interpretation of the Output

- The root mean square error (RMSE) measures the average difference between the predictions made by the model and the actual observations.

  The lower the RMSE, the more closely a model can predict the actual observations.

- R^2 measure the correlation between the predictions made by the model and the actual observations.

  The higher the R^2, the more closely a model can predict the actual observations.

- The mean absolute error (MAE) is the average absolute difference between the predictions made by the model and the actual observations.

- The lower the MAE, the more closely a model can predict the actual observations.

k-Fold Cross Validation in R

You can use

$$\text{model1\$finalModel}$$

to examine the final model fit.

```
> model1$finalModel

Call:
lm(formula = .outcome ~ ., data = dat)

Coefficients:
(Intercept)           x1           x2           x3           x4
     2.0409       2.4546       0.5447       4.9945       4.0033
```

That is

$$y = 2.0409 + 2.4546x_1 + 0.5447x_2 + 4.9945x_3 + 4.0033x_4$$

k-Fold Cross Validation in R

It is also useful to view the model predictions made from each fold by calling

model1$resample

as indicated below.

```
> model1$resample
      RMSE  Rsquared       MAE Resample
1 0.2020076 0.9995687 0.1584344    Fold1
2 0.2149710 0.9995995 0.1837908    Fold2
3 0.1718566 0.9997660 0.1330509    Fold3
4 0.1995829 0.9995636 0.1677125    Fold4
5 0.2494761 0.9994125 0.2043587    Fold5
```