MTH 4230 R Notes 9

1 Automated Model Selection Procedures

1.1 All-Subsets Model Selection Procedure with Cp, R^2, or R_a^2 in R

- The package `'leaps'` has functions for selecting subsets of predictor variables for inclusion in a model.

  After downloading and installing `'leaps'`:

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

  load it into the current R session by typing:

  ```
  library(leaps)
  ```

- In particular, within the `'leaps'` *package* is a *function* called `leaps()` that becomes available once the `'leaps'` package is loaded into the current session.

  The `leaps()` function performs the ***all-subsets model selection procedure***, which fits models corresponding to **every subset** of the predictor variables that are available for inclusion in a model.

  ```
  leaps()     # Searches all possible models for the best model of each
              # size (i.e. the best for each given number of predictors)
  ```

  By default, it uses the ***Mallows' Cp*** criterion for choosing the best model. Alternatively it will use the $\boldsymbol{R^2}$ or the $\boldsymbol{R^2_{\text{adj}}}$ criterion if told to do so.

- `leaps()` takes arguments `x`, a *matrix* whose columns are the predictor variables being considered for inclusion in the model, `y`, the response vector, and `method`, one of `"Cp"` (the default), `"r2"`, or `"adjr2"`.

- As an example, suppose we have the following *data frame*:

  ```
  my.data
  ```

```
##    response x1  x2 x3
## 1        21 10 2.4 94
## 2        22 11 1.9 75
## 3        24 13 2.6 66
## 4        20 12 1.8 77
## 5        26 13 1.3 75
## 6        26 16 0.9 81
## 7        25 15 1.2 72
## 8        27 20 1.0 55
## 9        31 18 1.3 44
## 10       34 19 1.0 69
## 11       29 19 0.8 73
## 12       33 22 0.7 90
```

Below, we create a *matrix* `x.mat` from the columns `x1`, `x2`, and `x3` of the data frame `my.data` and a vector `y.vec` from the column `response`:

```
x.mat <- cbind(my.data$x1, my.data$x2, my.data$x3)
is.matrix(x.mat)

## [1] TRUE

y.vec <- my.data$response
is.vector(y.vec)

## [1] TRUE
```

Now we're ready to call `leaps()`:

```
leaps(x = x.mat, y = y.vec, method = "Cp", names = c("x1", "x2", "x3"))

## $which
##      x1    x2    x3
## 1  TRUE FALSE FALSE
## 1 FALSE  TRUE FALSE
## 1 FALSE FALSE  TRUE
## 2  TRUE FALSE  TRUE
## 2  TRUE  TRUE FALSE
## 2 FALSE  TRUE  TRUE
## 3  TRUE  TRUE  TRUE
##
## $label
## [1] "(Intercept)" "x1"          "x2"          "x3"
##
## $size
## [1] 2 2 2 3 3 3 4
##
## $Cp
## [1]  0.01755588  9.18151258 24.36928145  2.00041859  2.01492829  9.71064546  4.00000000
```

The function `leaps()` returns a *list* with several components, among them:

- The component `$which` describes which variables are included in a given model.
- The component `$Cp` gives the ***Cp*** value for each of the models.
- The `$size` component gives the number of parameters in a given model.

A **small** value of ***Cp*** indicates a good model, and **Cp** shouldn't be too much larger (if at all) than the value of ***p*** (the number of parameters in the given model). From the output, we see that the model with just `x1` would be a good choice, as would either the model with `x1` **and** `x3` or the model with `x1` **and** `x2`.

- To perform an ***all-subsets model selection procedure*** with $\boldsymbol{R^2}$ or $\boldsymbol{R^2_a}$ as the model selection criterion (instead of ***Cp***), use `method = "r2"` or `method = "adjr2"`, respectively, in the call to `leaps()`.

1.2 Stepwise Model Selection Procedures

- We can perform any of the following automated model selection procedures:

  1. ***Backward stepwise*** procedure.
  2. ***Forward stepwise*** procedure.
  3. ***Backward elimination*** procedure.
  4. ***Forward selection*** procedure.

  using the function:

```
step()      # Select a model using a stepwise procedure, a backward
            # elimination procedure, or a forward selection procedure
```

- Regardless of which procedure is carried out using `step()`, by default the decision as to whether or not to add or remove a predictor from a model is based on whether or not doing so would lower the value of ***AIC***.

  If there are more than one predictors whose addition or removal from a model lowers the ***AIC***, the predictor that lowers it the most is the one that's added or removed.

  The procedure terminates when there are no predictors left whose addition or removal would lower the ***AIC***.

- Alternatively, we can choose to base the procedure on the value of ***BIC*** (instead of ***AIC***) by setting the optional argument `k = log(n)` in the call to `step()`, where `n` is the sample size.

1.2.1 The Backward Stepwise Procedure

- To illustrate, to perform a ***backward stepwise*** procedure (using `my.data` from above), we first fit the regression model with **all three** predictors:

```
my.reg <- lm(response ~ x1 + x2 + x3, data = my.data)
```

This is the model used as the **starting model**.

We then pass the *lm* object to `step()`, specifying `direction = "both"` to indicate that we want the procedure to be performed **stepwise**, i.e. that we want to allow **both** additions **and** deletions of predictors from the model during the steps:

```
step(my.reg, direction = "both")

## Start:  AIC=25.48
## response ~ x1 + x2 + x3
##
##        Df Sum of Sq     RSS    AIC
## - x2    1     0.003  51.490 23.478
## - x3    1     0.096  51.584 23.500
## <none>               51.488 25.477
## - x1    1    49.626 101.113 31.576
##
## Step:  AIC=23.48
## response ~ x1 + x3
##
##        Df Sum of Sq     RSS    AIC
## - x3    1     0.110  51.601 21.504
## <none>               51.490 23.478
## + x2    1     0.003  51.488 25.477
## - x1    1   156.837 208.328 38.250
##
## Step:  AIC=21.5
## response ~ x1
##
##        Df Sum of Sq     RSS    AIC
## <none>               51.601 21.504
## + x3    1     0.110  51.490 23.478
## + x2    1     0.017  51.584 23.500
## - x1    1   175.399 227.000 37.281
##
## Call:
## lm(formula = response ~ x1, data = my.data)
##
## Coefficients:
## (Intercept)           x1
##       10.52         1.02
```

From the output, we see the following **sequence of steps** was taken, each of which lowers the value of ***AIC***:

- **Start** with the **full model** containing `x1`, `x2`, and `x3` (***AIC* = 25.48**)
- **Step 1**: **remove** `x2` (resulting in ***AIC* = 23.48**).
- **Step 2**: **remove** `x3` (resulting in ***AIC* = 21.50**).
- **Terminate** the procedure with the model that contains just `x1` (***AIC* = 21.50**) because the ***AIC*** can't be made any smaller by removing `x1` or adding either of `x2` or `x3`.

Also shown in the output are the ***extra sums of squares*** (`Sum of Sq`), their associated degrees of freedom (`Df`), and the ***error sums of squares*** (`RSS`).

1.2.2 The Forward Stepwise Procedure

- A ***forward stepwise*** procedure is performed using `step()` in a similar manner except we **start** with a model containing **just an intercept**, and we have to tell `step()` **which variables** we're considering including in the model by passing their names in a model *formula* via the `scope` argument.

- For example, using `my.data` from above, we first fit the regression model with **just an intercept**:

```
my.reg <- lm(response ~ 1, data = my.data)
```

This will be the **starting model** in the ***forward stepwise*** procedure.

Next we pass this *lm* object to `step()` along with the full model (containing the variables we're considering adding to the starting model) and the *data frame* containing the variables:

```
step(my.reg, scope = response ~ x1 + x2 + x3, direction = "both", data = my.data)

## Start:  AIC=37.28
## response ~ 1
##
##        Df Sum of Sq     RSS    AIC
## + x1    1   175.399  51.601 21.504
## + x2    1   116.420 110.580 30.650
## <none>              227.000 37.281
## + x3    1    18.672 208.328 38.250
##
## Step:  AIC=21.5
## response ~ x1
##
##        Df Sum of Sq     RSS    AIC
## <none>               51.601 21.504
## + x3    1     0.110  51.490 23.478
## + x2    1     0.017  51.584 23.500
## - x1    1   175.399 227.000 37.281
##
## Call:
## lm(formula = response ~ x1, data = my.data)
##
## Coefficients:
## (Intercept)           x1
##       10.52         1.02
```

Interpretation of the output is as described in Subsection 1.2.1.

1.2.3 The Backward Elimination Procedure

- We carry out a ***backward elimination*** procedure using `step()` as described in Subsection 1.2.1, except now we specify `direction = "backward"`. In this case, only **removals** of predictors are allowed in the procedure – once a predictor is removed, it's out of the model for good.
- Here's an example:

```
my.reg <- lm(response ~ x1 + x2 + x3, data = my.data)
```

```
step(my.reg, direction = "backward")

## Start:  AIC=25.48
## response ~ x1 + x2 + x3
##
##        Df Sum of Sq     RSS    AIC
## - x2    1     0.003  51.490 23.478
## - x3    1     0.096  51.584 23.500
## <none>               51.488 25.477
## - x1    1    49.626 101.113 31.576
##
## Step:  AIC=23.48
## response ~ x1 + x3
##
##        Df Sum of Sq     RSS    AIC
## - x3    1      0.11  51.601 21.504
## <none>               51.490 23.478
## - x1    1    156.84 208.328 38.250
##
## Step:  AIC=21.5
## response ~ x1
##
##        Df Sum of Sq     RSS    AIC
## <none>               51.601 21.504
## - x1    1     175.4 227.000 37.281
##
```

```
## Call:
## lm(formula = response ~ x1, data = my.data)
##
## Coefficients:
## (Intercept)           x1
##       10.52         1.02
```

Interpretation of the output is as described in Subsection 1.2.1.

1.2.4 The Forward Selection Procedure

- We carry out a ***forward selection*** procedure using `step()` as described in Subsection 1.2.2, except now we specify `direction = "forward"`. In this case, only **additions** of predictors are allowed in the procedure – once a predictor is added, it's stuck in the model for good.
- Here's an example:

```
my.reg <- lm(response ~ 1, data = my.data)
```

Now we pass this model along with the full model to `step()`:

```
step(my.reg, scope = response ~ x1 + x2 + x3, direction = "forward", data = my.data)

## Start:  AIC=37.28
## response ~ 1
##
##        Df Sum of Sq     RSS    AIC
## + x1    1   175.399  51.601 21.504
## + x2    1   116.420 110.580 30.650
## <none>              227.000 37.281
## + x3    1    18.672 208.328 38.250
##
## Step:  AIC=21.5
## response ~ x1
##
##        Df Sum of Sq    RSS    AIC
## <none>              51.601 21.504
## + x3    1  0.110295 51.490 23.478
## + x2    1  0.016911 51.584 23.500
##
## Call:
## lm(formula = response ~ x1, data = my.data)
##
## Coefficients:
## (Intercept)           x1
##       10.52         1.02
```

Interpretation of the output is as described in Subsection 1.2.1.