STAT 413/513: Methods of Data Analysis III

Lab 6: May 5, 2026

Model Selection in Binary Logistic Regression

As in the previous lab, we'll look at the Owl Nesting Locations data from Ch 20 Exercise 15. We load the data in R:

```
library(Sleuth3)
attach(ex2015)
str(ex2015)
```

```
## 'data.frame':    60 obs. of  8 variables:
##  $ Site    : Factor w/ 2 levels "Nest","Random": 2 2 2 2 2 2 2 2 2 2 ...
##  $ PctRing1: num  26 100 32 43 74 63 60 46 80 76 ...
##  $ PctRing2: num  33.3 92.7 22.2 79.9 61.8 85 72.2 68 62.9 63.8 ...
##  $ PctRing3: num  25.6 90.1 38.3 61.4 48.3 85.8 58.1 68.8 42 60.7 ...
##  $ PctRing4: num  19.1 72.8 39.9 81.2 67.4 84.9 54.1 67.9 59.6 59.1 ...
##  $ PctRing5: num  31.4 51.9 22.1 47.7 74.9 78 55.6 71.9 68.4 44.2 ...
##  $ PctRing6: num  24.8 50.6 20.2 69.6 66 78 53.5 69.5 59.7 53.4 ...
##  $ PctRing7: num  17.9 41.5 38.2 54.8 55.8 53.6 67.2 66 60 29.5 ...
```

Our response is whether the site is a nesting location or not and our predictors are the percentages of mature forest at the various rings. We create our binary response variable:

```
nest <- ifelse(Site == "Nest", 1, 0) #create binary response variable for nesting
```

Our goal is to perform variable selection using the drop-in-deviance test and AIC/BIC. Key functions to know:

- glm()
- anova(..., test = "chisq")
- AIC(), BIC()
- add1()
- update()
- drop1()
- step()

Drop-in-Deviance test with anova()

We can compare two nested models using the drop-in-deviance test via the anova() function.

```
model.1 <- glm(nest ~ PctRing1, data = ex2015[,-1],  family = binomial)
model.2 <- glm(nest ~ PctRing1 + PctRing2, data = ex2015[,-1], family = binomial)
anova(model.1, model.2, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: nest ~ PctRing1
## Model 2: nest ~ PctRing1 + PctRing2
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
```

```
## 1        58     71.097
## 2        57     71.067  1 0.029569   0.8635
```

Question: Based on the results, which model would you pick?

Answer: The p-value is quite large, so we fail to reject the null and conclude that the smaller model (Model 1) is sufficient. In other words, PctRing2 is not a significant predictor of whether a site is a nesting location or not.

Comparing models with AIC() and BIC()

We compute AIC and BIC for a model with only PctRing1 as a predictor (Model 1) and a model with both PctRing1 and PctRing2 as predictors (Model 2) using the AIC() and BIC() functions.

```
AIC(model.1)
```

```
## [1] 75.09694
```

```
AIC(model.2)
```

```
## [1] 77.06737
```

```
BIC(model.1)
```

```
## [1] 79.28563
```

```
BIC(model.2)
```

```
## [1] 83.3504
```

Question: Which of these two models would you pick based on AIC? Based on BIC?

Answer: The model with AIC/BIC lower is better. Model 1 has the lower AIC and the lower BIC, so we would select it over Model 2.

Forward stepwise selection

Forward selection by hand

We can use AIC, BIC and the drop in deviance test to compare the model with only PctRing1 with every model with PctRing1 and *one* more predictor. We use the add1() function to perform all these comparisons. We use the k argument to set the penalty to AIC ($k = 2$) or BIC ($k = log(n) = log(nrow(ex2015))$). Alternatively, we can specify test = "Chisq" or test = "LRT" to perform the drop-in-deviance test. (Note: we have to provide a scope argument to tell add1 which predictors to consider adding. In this case, we set the scope to include main effects for all explanatory variables with the formula 'nest ~ . ')

```
#AIC
add1(model.1, scope = terms(nest ~ ., data = ex2015[,-1]),
     k = 2)
```

```
## Single term additions
##
## Model:
## nest ~ PctRing1
##          Df Deviance    AIC
## <none>        71.097 75.097
## PctRing2  1   71.067 77.067
## PctRing3  1   65.265 71.265
## PctRing4  1   70.112 76.112
## PctRing5  1   66.003 72.003
## PctRing6  1   66.027 72.027
```

```
## PctRing7  1   71.095 77.095
```

```
#BIC
add1(model.1, scope = terms(nest ~ ., data = ex2015[,-1]),
     k = log(nrow( ex2015 )))
```

```
## Single term additions
##
## Model:
## nest ~ PctRing1
##          Df Deviance    AIC
## <none>        71.097 79.286
## PctRing2  1   71.067 83.350
## PctRing3  1   65.265 77.548
## PctRing4  1   70.112 82.395
## PctRing5  1   66.003 78.286
## PctRing6  1   66.027 78.310
## PctRing7  1   71.095 83.378
```

```
#drop in deviance
add1(model.1, scope = terms(nest ~ ., data = ex2015[,-1]),
     test = "Chisq")
```

```
## Single term additions
##
## Model:
## nest ~ PctRing1
##          Df Deviance    AIC    LRT Pr(>Chi)
## <none>        71.097 75.097
## PctRing2  1   71.067 77.067 0.0296  0.86347
## PctRing3  1   65.265 71.265 5.8320  0.01574 *
## PctRing4  1   70.112 76.112 0.9852  0.32092
## PctRing5  1   66.003 72.003 5.0938  0.02401 *
## PctRing6  1   66.027 72.027 5.0701  0.02434 *
## PctRing7  1   71.095 77.095 0.0020  0.96413
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question: Based on this output, which term would you add first based on AIC? Based on BIC? Based on the drop in deviance test?

Answer: You would pick PctRing3 first based on AIC/BIC as it gives us the lowest AIC/BIC of all the possible additions. Likewise, you would select PctRing3 first based on the drop-in-deviance test as it gives the lowest p-value.

We can easily add (or subtract) terms from the model using the update function. For example, we can add PctRing3 to model.1 with the following code:

```
model.1 <- update(model.1, ~ . + PctRing3)
model.1
```

```
##
## Call:  glm(formula = nest ~ PctRing1 + PctRing3, family = binomial,
##     data = ex2015[, -1])
##
## Coefficients:
## (Intercept)     PctRing1     PctRing3
##    -6.49136      0.03090      0.06133
```

```
##
## Degrees of Freedom: 59 Total (i.e. Null);  57 Residual
## Null Deviance:       83.18
## Residual Deviance: 65.26     AIC: 71.26
```

We can then run add1 again to see what term to add next:

```
add1(model.1, scope = terms(nest ~ ., data = ex2015[,-1]), k = 2)
```

```
## Single term additions
##
## Model:
## nest ~ PctRing1 + PctRing3
##          Df Deviance    AIC
## <none>        65.265 71.265
## PctRing2  1   59.909 67.909
## PctRing4  1   65.054 73.054
## PctRing5  1   63.481 71.481
## PctRing6  1   63.145 71.145
## PctRing7  1   65.064 73.064
```

This procedure (adding one term at a time based on some selection criterion) is known as *forward stepwise selection*.

Forward selection with the step() function

We can use the step() function to perform forward selection with AIC or BIC automatically. We'll perform a quick demo with trace = 1 so you can see how it performs multiple stepwise comparisons in a row.

```
model.1 <- glm(nest ~ PctRing1, data = ex2015[,-1],  family = binomial)
step(model.1,
     scope = list(lower = nest ~ 1,  upper = terms(nest ~ ., data = ex2015[,2:5]) ),
     direction = "forward", k = 2, trace = 1)
```

```
## Start:  AIC=75.1
## nest ~ PctRing1
##
##            Df Deviance    AIC
## + PctRing3  1   65.265 71.265
## <none>          71.097 75.097
## + PctRing4  1   70.112 76.112
## + PctRing2  1   71.067 77.067
##
## Step:  AIC=71.26
## nest ~ PctRing1 + PctRing3
##
##            Df Deviance    AIC
## + PctRing2  1   59.909 67.909
## <none>          65.265 71.265
## + PctRing4  1   65.054 73.054
##
## Step:  AIC=67.91
## nest ~ PctRing1 + PctRing3 + PctRing2
##
##            Df Deviance    AIC
## <none>          59.909 67.909
## + PctRing4  1   59.650 69.650
```

```
##
## Call:  glm(formula = nest ~ PctRing1 + PctRing3 + PctRing2, family = binomial,
##     data = ex2015[, -1])
##
## Coefficients:
## (Intercept)     PctRing1     PctRing3     PctRing2
##    -6.84619      0.06305      0.12149     -0.08833
##
## Degrees of Freedom: 59 Total (i.e. Null);  56 Residual
## Null Deviance:       83.18
## Residual Deviance: 59.91     AIC: 67.91
```

Now, we use the step() function to perform forward selection with both AIC and BIC and check the selected models.

```
# with AIC
# (note: the upper argument in scope needs the data argument to get all candidate terms)
aic1 <- step(model.1,
             scope = list(lower = nest ~ 1,  upper = terms(nest ~ ., data = ex2015[,-1]) ),
             direction = "forward", k = 2, trace = 0)


#with BIC
bic1 <- step(model.1,
             scope = list(lower = nest ~ 1,  upper = terms(nest ~ ., data = ex2015[,-1]) ),
             direction = "forward", k = log(nrow( ex2015 )), trace = 0)
aic1
```

```
##
## Call:  glm(formula = nest ~ PctRing1 + PctRing3 + PctRing2 + PctRing6 +
##     PctRing7, family = binomial, data = ex2015[, -1])
##
## Coefficients:
## (Intercept)     PctRing1     PctRing3     PctRing2     PctRing6     PctRing7
##   -10.00412      0.05694      0.12258     -0.11969      0.12615     -0.04784
##
## Degrees of Freedom: 59 Total (i.e. Null);  54 Residual
## Null Deviance:       83.18
## Residual Deviance: 52.98     AIC: 64.98
```

```
bic1
```

```
##
## Call:  glm(formula = nest ~ PctRing1 + PctRing3 + PctRing2 + PctRing6,
##     family = binomial, data = ex2015[, -1])
##
## Coefficients:
## (Intercept)     PctRing1     PctRing3     PctRing2     PctRing6
##   -10.27479      0.06428      0.12331     -0.11990      0.08113
##
## Degrees of Freedom: 59 Total (i.e. Null);  55 Residual
## Null Deviance:       83.18
## Residual Deviance: 55.04     AIC: 65.04
```

Question: Which method selected the model with fewer terms? Why do you think that might be?

Answer: BIC selects the smaller model. This makes sense because $\log(n) > 2$, so BIC applies a larger penalty

on the number of model parameters.

Backward stepwise selection

A related procedure is *backward selection*, wherein we start with a large model (in this case, a model including main effects for all of the explanatory variables) and see what happens to the AIC/BIC if we remove a single explanatory variable.

Backward selection by hand

We fit the large model in R:

```
model.rich <- glm(nest ~ ., data = ex2015[,-1],  family = binomial)
```

and we use the drop1() function to compare this model with each model with one explanatory variable *removed*

```
#AIC
drop1(model.rich, k = 2)
```

```
## Single term deletions
##
## Model:
## nest ~ PctRing1 + PctRing2 + PctRing3 + PctRing4 + PctRing5 +
##     PctRing6 + PctRing7
##          Df Deviance    AIC
## <none>        52.107 68.107
## PctRing1  1   54.846 68.846
## PctRing2  1   58.723 72.723
## PctRing3  1   59.854 73.854
## PctRing4  1   52.265 66.265
## PctRing5  1   52.903 66.903
## PctRing6  1   55.142 69.142
## PctRing7  1   54.357 68.357
```

```
#BIC
drop1(model.rich, k = log(nrow( ex2015 )))
```

```
## Single term deletions
##
## Model:
## nest ~ PctRing1 + PctRing2 + PctRing3 + PctRing4 + PctRing5 +
##     PctRing6 + PctRing7
##          Df Deviance    AIC
## <none>        52.107 84.862
## PctRing1  1   54.846 83.507
## PctRing2  1   58.723 87.383
## PctRing3  1   59.854 88.515
## PctRing4  1   52.265 80.925
## PctRing5  1   52.903 81.564
## PctRing6  1   55.142 83.802
## PctRing7  1   54.357 83.018
```

```
#drop in deviance
drop1(model.rich, test = "Chisq")
```

```
## Single term deletions
##
```

```
## Model:
## nest ~ PctRing1 + PctRing2 + PctRing3 + PctRing4 + PctRing5 +
##     PctRing6 + PctRing7
##          Df Deviance    AIC    LRT Pr(>Chi)
## <none>        52.107 68.107
## PctRing1  1   54.846 68.846 2.7393 0.097907 .
## PctRing2  1   58.723 72.723 6.6158 0.010108 *
## PctRing3  1   59.854 73.854 7.7473 0.005379 **
## PctRing4  1   52.265 66.265 0.1577 0.691276
## PctRing5  1   52.903 66.903 0.7961 0.372253
## PctRing6  1   55.142 69.142 3.0346 0.081507 .
## PctRing7  1   54.357 68.357 2.2504 0.133581
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question: Based on this output, which term would you add first based on AIC? Based on BIC? Based on the drop in deviance test? Answer: We see that removing PctRing4 leads to the greatest reduction in AIC and BIC. In addition, the drop-in-deviance test comparing model.rich to the model excluding PctRing4 gives the largest p-value and that p-value is much larger than 0.05. This suggests that PctRing4 is not significant and can be safely removed from the model.

We remove PctRing4 from the model

```
model.rich2 <- update(model.rich, nest ~ . - PctRing4)
model.rich2
```

```
##
## Call:  glm(formula = nest ~ PctRing1 + PctRing2 + PctRing3 + PctRing5 +
##     PctRing6 + PctRing7, family = binomial, data = ex2015[, -1])
##
## Coefficients:
## (Intercept)     PctRing1     PctRing2     PctRing3     PctRing5     PctRing6
##    -9.49471      0.05531     -0.12032      0.11495      0.03082      0.10272
##    PctRing7
##    -0.05293
##
## Degrees of Freedom: 59 Total (i.e. Null);  53 Residual
## Null Deviance:       83.18
## Residual Deviance: 52.26     AIC: 66.26
```

and perform a second backward step (just with AIC this time) with drop1()

```
drop1(model.rich2, k = 2)
```

```
## Single term deletions
##
## Model:
## nest ~ PctRing1 + PctRing2 + PctRing3 + PctRing5 + PctRing6 +
##     PctRing7
##          Df Deviance    AIC
## <none>        52.265 66.265
## PctRing1  1   54.948 66.948
## PctRing2  1   59.913 71.913
## PctRing3  1   59.978 71.978
## PctRing5  1   52.979 64.979
## PctRing6  1   55.162 67.162
## PctRing7  1   54.654 66.654
```

Backward selection with the step() function

As with forward stepwise selection, we can use step() to automatically perform backward selection (by specifying direction = "backward"):

```
step(model.rich,
     scope = list(lower = nest ~ 1,  upper = terms(nest ~ ., data = ex2015[,-1]) ),
     direction = "backward", k = 2, trace = 1)
```

```
## Start:  AIC=68.11
## nest ~ PctRing1 + PctRing2 + PctRing3 + PctRing4 + PctRing5 +
##     PctRing6 + PctRing7
##
##            Df Deviance    AIC
## - PctRing4  1   52.265 66.265
## - PctRing5  1   52.903 66.903
## <none>          52.107 68.107
## - PctRing7  1   54.357 68.357
## - PctRing1  1   54.846 68.846
## - PctRing6  1   55.142 69.142
## - PctRing2  1   58.723 72.723
## - PctRing3  1   59.854 73.854
##
## Step:  AIC=66.26
## nest ~ PctRing1 + PctRing2 + PctRing3 + PctRing5 + PctRing6 +
##     PctRing7
##
##            Df Deviance    AIC
## - PctRing5  1   52.979 64.979
## <none>          52.265 66.265
## - PctRing7  1   54.654 66.654
## - PctRing1  1   54.948 66.948
## - PctRing6  1   55.162 67.162
## - PctRing2  1   59.913 71.913
## - PctRing3  1   59.978 71.978
##
## Step:  AIC=64.98
## nest ~ PctRing1 + PctRing2 + PctRing3 + PctRing6 + PctRing7
##
##            Df Deviance    AIC
## <none>          52.979 64.979
## - PctRing7  1   55.045 65.045
## - PctRing1  1   55.900 65.900
## - PctRing6  1   59.904 69.904
## - PctRing2  1   60.770 70.770
## - PctRing3  1   62.087 72.087
```

```
##
## Call:  glm(formula = nest ~ PctRing1 + PctRing2 + PctRing3 + PctRing6 +
##     PctRing7, family = binomial, data = ex2015[, -1])
##
## Coefficients:
## (Intercept)     PctRing1     PctRing2     PctRing3     PctRing6     PctRing7
##   -10.00412      0.05694     -0.11969      0.12258      0.12615     -0.04784
##
## Degrees of Freedom: 59 Total (i.e. Null);  54 Residual
```

```
## Null Deviance:       83.18
## Residual Deviance: 52.98     AIC: 64.98
```