ST2304 Exercises Week 10: Selecting Models

Bob O'Hara

14 March 2018

Model Testing in R

We can use the Bird Brains data to look at different ways of comparing models. First we need the data:

```
BirdBrains <- read.csv("../Data/BirdBrains.csv")
BirdBrains$Mode.of.development <- factor(BirdBrains$Mode.of.development)
BirdBrains$M.of.Dev <- c("precocial", "semi-precocial", "semi-altricial", "altricial")[BirdBrains$Mode.
BirdBrains$M.of.Dev. <- factor(BirdBrains$M.of.Dev, levels =
                                 c("altricial", "semi-altricial",
                                   "semi-precocial", "precocial"))

Covars <- c("Maximum.lifespan", "Incubation.length",
            "Clutch.size", "Mean.latitude","logBodyMass", "M.of.Dev.")
# Scale the continuous covariates (messily)
ToScale <- Covars[sapply(Covars, function(wh, df)
  is.numeric(df[,wh]), df=BirdBrains)]
BirdBrains[,ToScale] <- scale(BirdBrains[,ToScale])
```

ANOVA

First, we will use ANOVA to look at the effect of maximum lifespan. We will include body mass, as we know this has a big effect. So we fitrst fit a model with log body mass, and then add maximum lifespan:

```
mod.bm <- lm(logBrainMass~logBodyMass, data=BirdBrains)
mod.lspan <- lm(logBrainMass~logBodyMass + Maximum.lifespan,
                data=BirdBrains)
```

We can use `anova()` to compare models in different ways. If we are comparing two models, then we give both models as arguments, with the smallest first:

```
anova(mod.bm, mod.lspan)
```

```
Analysis of Variance Table

Model 1: logBrainMass ~ logBodyMass
Model 2: logBrainMass ~ logBodyMass + Maximum.lifespan
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1    382 38.413
2    381 36.581  1    1.8322 19.083 1.616e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`anova()` tests whether the second model is better than the first (more generally, you can give it more than 2 models, and it goes through in order and compares each model to the previous one). We see that adding maximum lifepsan to the model improves the model (or to be strict, we are very unlikely to get the likelihood we estimate in the model with maximum lifespan if the data were generated by a process where maximum lifespan did not have an effect: the probability of getting at least this difference in the likelihoods if lifespan did not have an effect is about 0.0016%.

Note that the models have to be nested, i.e. the second one has to be formed by adding a term (or terms) to the first.

A simpler way to use `anova()` is to let R work out the models, and test them in the order that they are given:

```
anova(mod.lspan)
```

```
Analysis of Variance Table

Response: logBrainMass
                 Df Sum Sq Mean Sq  F value    Pr(>F)
logBodyMass       1 428.70  428.70 4465.000 < 2.2e-16 ***
Maximum.lifespan  1   1.83    1.83   19.083 1.616e-05 ***
Residuals       381  36.58    0.10
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So, because the model is written `logBrainMass ~ logBodyMass + Maximum.lifespan`, R fits the model with the intercept, then adds `logBodyMass` and then adds `Maximum.lifespan`. Contrast that to the model where the terms are added in the opposite order:

```
mod.lspanRev <- lm(logBrainMass~Maximum.lifespan + logBodyMass,
                data=BirdBrains)
anova(mod.lspanRev)
```

```
Analysis of Variance Table

Response: logBrainMass
                 Df  Sum Sq Mean Sq F value    Pr(>F)
Maximum.lifespan  1 249.724 249.724  2600.9 < 2.2e-16 ***
logBodyMass       1 180.808 180.808  1883.2 < 2.2e-16 ***
Residuals       381  36.581   0.096
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can also add interactions:

```
mod.ints <- lm(logBrainMass~logBodyMass*Maximum.lifespan,
                data=BirdBrains)
anova(mod.ints)
```

```
Analysis of Variance Table

Response: logBrainMass
                            Df Sum Sq Mean Sq   F value    Pr(>F)
logBodyMass                  1 428.70  428.70 4503.7143 < 2.2e-16 ***
Maximum.lifespan             1   1.83    1.83   19.2481 1.489e-05 ***
logBodyMass:Maximum.lifespan 1   0.41    0.41    4.3035   0.03871 *
Residuals                  380  36.17    0.10
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Comparing all subsets

If we just want to find a good model, we can compare all of the subsets. There are a few ways to do this in R, here we'll us the `bestglm` package. We have to give it a dataframe with the design matrix followed by the response:

```
Covars <- c("Maximum.lifespan", "Mean.latitude",
            "logBodyMass", "M.of.Dev.")

library(bestglm) # might need install.packages("bestglm")
UseData <- cbind(BirdBrains[,Covars], BirdBrains$logBrainMass)
AllSubsets <- bestglm(Xy=UseData, IC="AIC")
```

```
Morgan-Tatar search since factors present with more than 2 levels.
```

```
summary(AllSubsets$BestModel)
```

```
Call:
lm(formula = y ~ ., data = data.frame(Xy[, c(bestset[-1], FALSE),
    drop = FALSE], y = y))

Residuals:
     Min       1Q   Median       3Q      Max
-0.84246 -0.15423  0.00709  0.16466  0.71875

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)               0.70139    0.01881  37.289  < 2e-16 ***
Maximum.lifespan          0.07791    0.01955   3.985 8.10e-05 ***
logBodyMass               1.09634    0.02273  48.230  < 2e-16 ***
M.of.Dev.semi-altricial -0.02137    0.05752  -0.371    0.711
M.of.Dev.semi-precocial -0.31903    0.05495  -5.806 1.36e-08 ***
M.of.Dev.precocial      -0.47993    0.03934 -12.201  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2546 on 378 degrees of freedom
Multiple R-squared:  0.9475,    Adjusted R-squared:  0.9468
F-statistic:  1365 on 5 and 378 DF,  p-value: < 2.2e-16
```

The function returns an object that has the best model (`AllSubsets$BestModel`): here we can see that it has Maximum.lifespan, logBodyMass and M.of.Dev. as terms (and M.of.Dev. is a factor with 4 levels: the parameters are contrasts to the first level).

The function also reports the best models with 0, 1, 2 etc. parameters (and puts a star by the best of the best). Note that a lower AIC is better:

```
xtable::xtable(AllSubsets$Subsets)
```

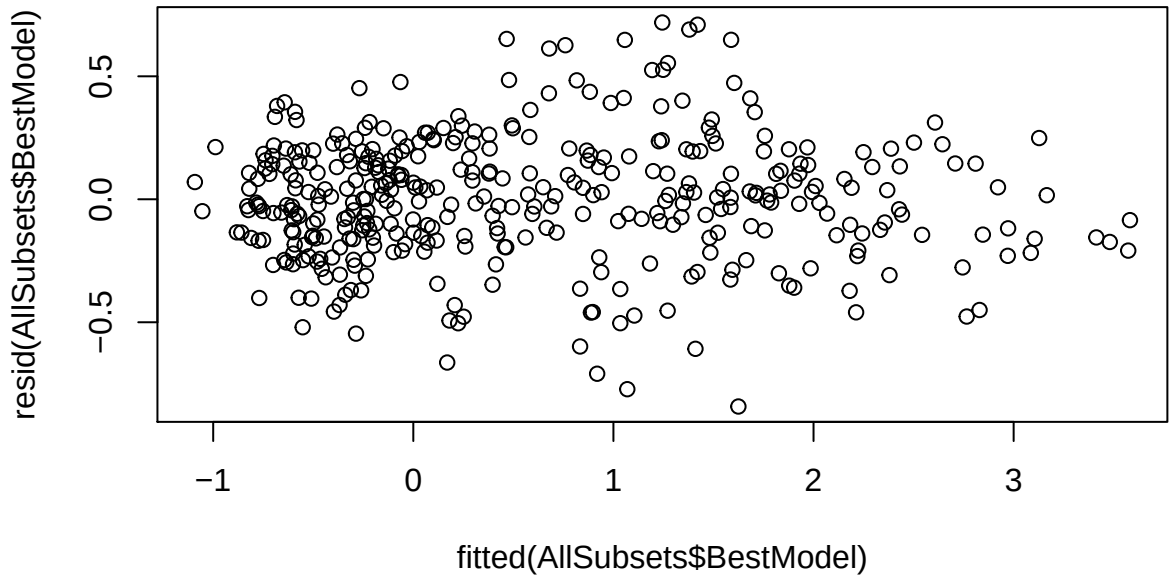
% latex table generated in R 3.4.3 by xtable 1.8-2 package % Thu Mar 15 20:24:53 2018

	Intercept	Maximum.lifespan	Mean.latitude	logBodyMass	M.of.Dev.	logLikelihood	AIC
0	TRUE	FALSE	FALSE	FALSE	FALSE	-37.62	75.24
1	TRUE	FALSE	FALSE	TRUE	FALSE	442.03	-882.06
2	TRUE	FALSE	FALSE	TRUE	TRUE	520.44	-1032.88
3*	TRUE	TRUE	FALSE	TRUE	TRUE	528.34	-1046.68
4	TRUE	TRUE	TRUE	TRUE	TRUE	528.86	-1045.72

We can see several other "good" models

Becuse we get the best model, we can do all the usual things with it:

```
plot(fitted(AllSubsets$BestModel), resid(AllSubsets$BestModel))
```



Question 1

Back in Exercise 2 we looked at fitting polynomials to the life expectancy data. Now we can ask which is the best model. First we need the data (if you can't find it, download it from Blackboard BUT DON'T OPEN IT IN ANY OTHER PROGRAMME!):

```
rawdata <- read.csv("../Data/LifeExpectancy.csv") # NOTE: the file path might be different!
NoSA <- rawdata[rawdata$Country!="South Africa",]
```

Now you can find out what order polynomial is the best. You should fit all of the models up to order 10, and:

- compare them using AIC and decide which model is best (remember: a lower value of AIC means a better model, and generally AICs or BICs within 2 of each other are equivalent).
- compare the models using BIC. Do you get a different result?

There are a couple of ways to calculate AIC and BIC. One way is by hand. AIC is $-2logLik + 2p$, wher p is the number of parameters:

```
mod0 <- lm(Life.exectancy~1, data=NoSA)
mod1 <- update(mod0, .~ . + Health.Spending.per.capita)
mod2 <- update(mod1, .~ . + I(Health.Spending.per.capita^2))
AIC.1 <- -2*logLik(mod1) + 2*mod1$rank
```

A nicer way to do this is to use the AIC function:

```
AIC(mod0, mod1, mod2)
```

```
##      df      AIC
## mod0  2 236.9086
## mod1  3 215.9586
## mod2  4 187.3340
```

Similarly, BIC is $-2logLik + log(n)p$, where n is the number of observation, so we can do these (the rank of a model is the number of parameters, nobs() is the number of observations used to fit a model):

```
-2*logLik(mod1) + log(nobs(mod1))*mod1$rank
```

```
## 'log Lik.' 217.4339 (df=3)
```

```
CalcBIC <- function(mod) -2*logLik(mod) + log(nobs(mod))*mod$rank
BIC(mod1)
```

```
## [1] 221.1716
```

The values may differ from each other by a constant, but this is fine as we only want to compare them to each other (because of this we can add or subtract any constants).

Note: there is an idea called the *principle of marginality* which says that if you fit an interaction then you should fit all of the main effects and lower order interactions (e.g. if you fit a model with A:B:C you need A, B, C, A:B, A:C, and B:C). Similarly, if you fit a model with a polynomial of order p then you should also include the polynomial terms for orders 1, 2, ..., $p-1$ (i.e. x, x^2, x^3 etc.). The reason for this is that if you don't do this then you make some arbitrary assumptions, e.g. that the slope of the curve is zero at the origin (if you don't include the linear term). This also means that you can change the origin (e.g. when you centre the covariates) without changing the model.

Question 2

In Exercise 2 we looked at regression going bad. Now you get to see what happens when you test the different models. This is the code for simulation (if you prefer, re-use the code you wrote before):

```
N <- 50
library(MASS)
muX <- c(0,0) # mean of bivariate distribution
Corr <- 0.5 # correlation
sigmaX <- matrix(c(1,Corr,Corr,1), nrow=2) # covariance matrix
x <- mvrnorm(N, muX, Sigma=sigmaX) # 2 columns: x[,1] & x[,2]

# Simulate from a different model
N <- 50; alpha <- 0; sigma <- 1 # same throughout
beta1 <- -20
beta2 <- 5000

mu <- alpha + beta1*x[,1] + beta2*x[,2]
y <- rnorm(N, mu, sigma)
mod <- lm(y ~ x[,1] + x[,2])
```

For all of the following, fit a model with both x_1 and x_2, and use anova() to select the best model. Also change the order of the variables in the model (i.e. also fit `lm(y ~ x[,1] + x[,2])` and see if this changes the anova()).

1. Simulate x_1 and x_2 from a standard normal distribution with no correlation. Then simulate the model (above) with $\beta_1 = 1$ and $\beta_2 = 1$.
2. Simulate x_1 and x_2 from a standard normal distribution with correlation 0.7. Then simulate the model (above) with $\beta_1 = 1$ and $\beta_2 = 0$, i.e. where there is no effect of β_2.
3. Simulate x_1 and x_2 from a standard normal distribution with correlation 0.7. Then simulate the model (above) with $\beta_1 = 1$ and $\beta_2 = 1$.
4. Now simulate x_1 and x_2 from a standard normal distribution with correlation -0.8. Then simulate the model (above) with $\beta_1 = 5$ and $\beta_2 = 5$.