Model selection in R, using the MuMIn library

Philip Dixon

2022-11-22

This document explains the code in sat.r and demonstrates the use of functions in the MuMIn (MUlti Model INference) library. It was completely rewritten in November 2022 to use MuMIn functions instead of the leaps library.

The example is the SAT score analysis from the book with two changes:

- Alaska is removed, so there are 49 states
- ltakers, a log transformed version of takers, was added to the set of potential variables

Both have been done for you in the sat.xlsx file.

```
library(readxl)
library(MuMIn)
library(dplyr)

sat <- read_excel('c:/philip/stat 587/data/sat.xlsx')
```

When comparing fits of models, all models must be fit to the same data set. This is not an issue when there are no missing values in the data set. If there are missing values, R will omit any observation with a missing value. This invisibly changes the data set used to fit a model. To see this, imagine a data set of 83 observations in which 4 observations are missing the value of variable D. A model that includes D will be fit to 79 observations; a model that doesn't will be fit to 83 observations. OOPS!

The best solution is to reset the default treatment of missing values to fail. This will generate an error if you try to fit a model to data with missing values.

```
options(na.action='na.fail')
```

If your data set has missing values, you can eliminate all rows with any missing values by:

```
sat <- na.omit(sat)
```

This will drop every row with a missing value. If your data set has extra columns that you won't use in your model selection exercise, you don't want to drop rows with missing values in the unused variables. The best solution is to subset variables to only those you need using select in the dplyr library. Piping (%>%) makes the code easier to read.

```
sat <- sat %>%
  select(sat, takers, ltakers, income, years, public, expend, rank)
```

The MuMIn library implements three very useful tasks:

- Computing and ranking models by AIC, AICc, BIC and some other model selection statistics
- All subsets evaluation of models
- Model averaging parameter estimates over multiple models

We will not discuss model averaging, but it is the up-and-coming replacement for selecting a single "best" model. The MuMIn function is model.avg(). An introductory paper using ecological and environmental

impact examples is here: https://digitalcommons.usu.edu/agstats/2022/all/4/. Another is Dormann, C. et al. 2018 Model averaging in ecology: a review of Bayesian, information-theoretic, and tactical approaches for predictive inference. Ecological Monographs 88, 485–504, And, if you want a more complete, accessible treatment with many examples, see David Fletcher's monograph, Model Averaging (Springer, 2018).

comparing a specific set of models

The logic is to fit each model, then evaluate the model selection statistic. The model.sel() function accepts a sequence of fitted models, separated by commas.

```
sat.lm1 <- lm(sat ~ ltakers, data=sat)
sat.lm2 <- lm(sat ~ expend, data=sat)
sat.lm3 <- lm(sat ~ ltakers + expend, data=sat)
sat.lm4 <- lm(sat ~ ltakers + expend + years + public, data=sat)

sat.ms1 <- model.sel(sat.lm1, sat.lm2, sat.lm3, sat.lm4)
```

The resulting object is an extension of a data frame. Printing it out gives you the names of each model, the parameter estimates, model df, log likelihood, model selection statistic, change from the best (delta) and the associated model weight. We won't talk about model weights. The default model selection statistic is AICc, the small sample corrected AIC.

```
sat.ms1
```

```
## Model selection table
##         (Intrc)  ltkrs   expnd  publc years df   logLik  AICc  delta weight
## sat.lm3  1029.0 -66.17  4.6050               4 -223.096 455.1   0.00   0.74
## sat.lm4   853.4 -64.43  4.0240 0.2906  9.85  6 -221.595 457.2   2.09   0.26
## sat.lm1  1112.0 -59.18                       3 -237.416 481.4  26.26   0.00
## sat.lm2   961.7        -0.5923               3 -278.198 562.9 107.83   0.00
## Models ranked by AICc(x)
```

You can change the model selection statistic by specifying rank = followed by the name of a model selection statistic (actually, the name of the function that computes that statistic). We discussed AIC and BIC, but model.sel() provides others.

You can also request values of "extra" model selection statistics by using the extra = argument.

```
# ranking by specified function
sat.ms1b <- model.sel(sat.lm1, sat.lm2, sat.lm3, sat.lm4, rank=AIC )
sat.ms1c <- model.sel(sat.lm1, sat.lm2, sat.lm3, sat.lm4, rank=BIC )

# printing multiple measures in one table, but still ranking by AICc
#   (or specified rank= quantity)
options(width=100)

model.sel(sat.lm1, sat.lm2, sat.lm3, sat.lm4, extra=c('AIC','BIC') )
```

```
## Model selection table
##         (Intrc)  ltkrs   expnd  publc years   AIC   BIC df   logLik  AICc  delta weight
## sat.lm3  1029.0 -66.17  4.6050              454.2 461.8  4 -223.096 455.1   0.00   0.74
## sat.lm4   853.4 -64.43  4.0240 0.2906  9.85 455.2 466.5  6 -221.595 457.2   2.09   0.26
## sat.lm1  1112.0 -59.18                      480.8 486.5  3 -237.416 481.4  26.26   0.00
## sat.lm2   961.7        -0.5923              562.4 568.1  3 -278.198 562.9 107.83   0.00
## Models ranked by AICc(x)
```

All subsets regression

This is done using the dredge() function. The name is appropriate: just like dredging in real life, you never know whether what you get will be useful.

The approach is to specify the largest model, i.e., the model containing all the variables being considered. Then dredge() will evaluate all subsets and create a model selection object with one row for each of them. The result is sorted by the model selection criterion. The best model is in the first row. The default model selection criterion is AICc, but you can change that by specifying rank = , e.g., dredge(sat.lm, rank = AIC) or dredge(sat.lm, rank = BIC).

```
# start by specifying the largest model
sat.lm <- lm(sat ~ takers + ltakers + income + years + public + expend + rank,
   data=sat)

sat.ms2 <- dredge(sat.lm)
```

Useful things you can do with the model selection result:

- how many models did dredge fit?

```
nrow(sat.ms2)
```

```
## [1] 128
```

- Print out the best model (in row 1, or has delta = 0)

```
sat.ms2[1,]
```

```
## Global model call: lm(formula = sat ~ takers + ltakers + income + years + public +
##     expend + rank, data = sat)
## ---
## Model selection table
##    (Intrc) expnd ltkrs rank years df   logLik  AICc delta weight
## 86   399.1 3.996 -38.1  4.4 13.15  6 -219.062 452.1     0      1
## Models ranked by AICc(x)
```

```
subset( sat.ms2, delta==0)
```

```
## Global model call: lm(formula = sat ~ takers + ltakers + income + years + public +
##     expend + rank, data = sat)
## ---
## Model selection table
##    (Intrc) expnd ltkrs rank years df   logLik  AICc delta weight
## 86   399.1 3.996 -38.1  4.4 13.15  6 -219.062 452.1     0      1
## Models ranked by AICc(x)
```

- Print out the five (or any other number) best models, i.e., rows 1 to 5 '# five best models

```
sat.ms2[1:5,]
```

```
## Global model call: lm(formula = sat ~ takers + ltakers + income + years + public +
##     expend + rank, data = sat)
## ---
## Model selection table
##     (Intrc) expnd  incom  ltkrs   publc  rank   takrs years df   logLik  AICc delta weight
## 86    399.1 3.996        -38.10         4.400         13.15  6 -219.062 452.1  0.00  0.444
## 88    291.2 3.872 0.1135 -31.16         5.060         13.49  7 -218.491 453.7  1.59  0.201
## 94    361.3 4.143        -33.73 -0.2785 5.158         12.18  7 -218.854 454.4  2.32  0.140
## 118   402.4 3.994        -36.98         4.319 -0.0809 13.28  7 -219.048 454.8  2.70  0.115
```

```
## 6    1029.0 4.605        -66.17                              4 -223.096 455.1  2.98  0.100
## Models ranked by AICc(x)
```

- Print out all models within 2 of the best, i.e., delta <= 2

```
subset(sat.ms2, delta <= 2)
```

```
## Global model call: lm(formula = sat ~ takers + ltakers + income + years + public +
##     expend + rank, data = sat)
## ---
## Model selection table
##    (Intrc) expnd  incom  ltkrs rank years df   logLik  AICc delta weight
## 86   399.1 3.996        -38.10 4.40 13.15  6 -219.062 452.1  0.00  0.689
## 88   291.2 3.872 0.1135 -31.16 5.06 13.49  7 -218.491 453.7  1.59  0.311
## Models ranked by AICc(x)
```

- Get additional information about the best model by using get.models() to extracting it (or multiple models) from the model selection dataframe. The result of get.models() is a list with one entry for each requested model (even if you only request one model). To pry that model out of the list, use [[1]] after the list name. Once extracted, you have the result of fitting that model, e.g. by lm(), and can use any function that works with lm() results.

```
sat.best <- get.models(sat.ms2, delta==0)[[1]]
summary(sat.best)
```

```
##
## Call:
## lm(formula = sat ~ expend + ltakers + rank + years + 1, data = sat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.304  -9.917   0.596  11.880  59.203
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 399.1145   232.3716   1.718  0.09291 .
## expend        3.9957     0.7642   5.228 4.52e-06 ***
## ltakers     -38.1005    11.9152  -3.198  0.00257 **
## rank          4.4003     1.8989   2.317  0.02520 *
## years        13.1473     5.4778   2.400  0.02069 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22.32 on 44 degrees of freedom
## Multiple R-squared:  0.9107, Adjusted R-squared:  0.9025
## F-statistic: 112.1 on 4 and 44 DF,  p-value: < 2.2e-16
```

Optional: dredge() provides allsorts of ways to limit the list of possible models. E.g.:

- do not consider models with both ltakers and takers, or
- only allow models with between 2 and 5 variables (not counting the intercept)

```
sat.ms3 <- dredge(sat.lm, subset = !(ltakers && takers))
sat.ms4 <- dredge(sat.lm, m.lim = c(2,5))
```

calculating the PRESS statistic

The PRESS statistic is based on the leave-one-out residuals. These are residuals for observation i based on a model fit without that observation. There is a matrix algebra trick to efficiently calculate these residuals. PRESS is the Error SS for predicting new observations. It is easier to interpret the root-mean-square-error-of-prediction (rMSEP). This is the sd of the errors predicting new observations. rMSEP = sqrt(PRESS / n).

Here's my function to compute PRESS and rMSEP statistics from a regression model.

```
press <- function(m) {
  eh <- resid(m)/(1-lm.influence(m)$hat)
  press <- sum(eh^2)
  n <- length(eh)
  c(PRESS = press, rMSEP=sqrt(press/n), n=n )
  }
```

You provide a regression model, e.g. fit by lm() or extracted from a model selection object. The result is a vector of 3 numbers: the PRESS statistic, the rMSEP, and the number of observations.

```
press(sat.lm)
```

```
##       PRESS       rMSEP           n
## 33171.89719    26.01879    49.00000
```

```
# for comparison: error sd for the "all variables" model
summary(sat.lm)$sigma
```

```
## [1] 22.77621
```

And, to demonstrate the value of reducing the number of variables in a regression model, here is the rMSEP for the model with the smallest AICc:

```
press(sat.best)
```

```
##       PRESS       rMSEP           n
## 27437.71998    23.66334    49.00000
```