ENVX2001 LAB - PREDICTIVE MODELLING

Table of contents

> **Tip**
>
> Please work on this exercise by creating your own R Markdown file.

Exercise 1: Backward elimination in R

Data: *Dippers* spreadsheet

Dippers are thrush-sized birds living mainly in the upper reaches of rivers, which feed on benthic invertebrates by probing the river beds with their beaks. The dataset in this exercise contains data from a biological survey which examined the nature of the variables thought to influence the breeding of British dippers.

Twenty-two sites were included in the survey. Some of the variables have been transformed.

The variables measured were:

- `Altitude` site altitude
- `Hardness` water hardness
- `RiverSlope` river-bed slope
- `LogCadd` the numbers of caddis fly larvae, transformed
- `LogStone` the numbers of stonefly larvae, transformed
- `LogMay` the numbers of mayfly larvae, transformed
- `LogOther` the numbers of all other invertebrates collected, transformed
- `Br_Dens` the number of breeding pairs of dippers per 10 km of river

In the analyses, the four invertebrate variables were transformed using a $log(x+1)$ transformation.

```r
library(readxl)
Dippers <- read_xlsx("mlr.xlsx" , sheet = "Dippers")
glimpse(Dippers)
```

```
Rows: 22
Columns: 8
$ Altitude   <dbl> 259, 198, 251, 184, 145, 145, 198, 160, 251, 159, 160, 145,~
$ Hardness   <dbl> 12.20, 22.00, 26.30, 22.50, 29.50, 39.90, 42.80, 59.60, 69.~
$ RiverSlope <dbl> 10.90, 14.70, 6.90, 4.60, 1.91, 5.00, 6.20, 14.30, 4.60, 3.~
$ Br_Dens    <dbl> 3.60, 4.30, 3.80, 3.40, 3.80, 4.50, 4.30, 5.00, 4.50, 3.40,~
$ LogCadd    <dbl> 2.303, 2.890, 3.784, 4.419, 3.219, 3.932, 3.664, 4.431, 3.7~
$ LogStone   <dbl> 5.242, 4.344, 5.231, 5.242, 3.829, 4.898, 4.357, 6.337, 5.4~
$ LogMay     <dbl> 0.000, 3.401, 5.826, 5.749, 5.509, 5.749, 5.371, 0.000, 4.8~
$ LogOther   <dbl> 1.386, 1.609, 1.386, 1.386, 1.099, 3.045, 1.386, 2.944, 2.5~
```

You may explore the data on your own (hint: look at last week's exercise on histogram and scatterplot matrices).

When ready, perform a backward elimination starting from the full model:

```r
FullMod <- lm(Br_Dens ~ ., data=Dippers)
RedMod <- step(FullMod, direction = "backward")
summary(FullMod)
summary(RedMod)
AIC(FullMod, RedMod)
```

⚠ Question 1

Which model is chosen? Why?

Exercise 2: Model quality - multiple vs adjusted r^2

Data: *California_streamflow* spreadsheet

Import the "California_streamflow" sheet into R.

```
stream <- read_xlsx("mlr.xlsx", "California_streamflow")
```

in this exercise we will use the same data as last week. To jog your memory, the dataset contains 43 years of annual precipitation measurements (in mm) taken at (originally) 6 sites in the Owens Valley in California. Through model selection via partial F-test we have the final model as below:

```
fit <- lm(L10BSAAM ~ L10OPRC + L10OBPC, data = stream)
```

We will now add a totally useless variable to the dataset. This variable is a random number generated from a normal distribution with mean 3 and standard deviation 2. We use the `set.seed()` function to make sure that everybody gets the same random values.

```
set.seed(100) # to make sure everybody gets the same results


# this generates the random number into the dataset
stream$random_no <- rnorm(n = nrow(stream), mean = 3, sd = 2)
```

We will see the impact of including a totally useless variable, such as this random variable, has on measures of model quality, r^2 and adjusted r^2 values.

Task: create two regression models:

1. `L10BSAAM ~ L10OPRC + L10OPBC`
2. `L10BSAAM ~ L10OPRC + L10OPBC + random_no`

> ⚠ Question 2
>
> Compare each in terms of their multiple r^2 and adjusted r^2 values. Which performance measure (multiple r^2 or adj r^2) would you use to identify which predictors to use in your model?

Exercise 3: Making predictions

Data: *California_streamflow* spreadsheet

Let's use the same model again. We will import the data once more just in case it has been modified in the previous exercise.

```r
# read in the data
require(readxl)
stream <- read_xlsx("mlr.xlsx", "California_streamflow")
# best model
ML_Mod2 <- lm(L10BSAAM ~ L10OPRC + L10OBPC, data = stream)
summary(ML_Mod2)
```

```
Call:
lm(formula = L10BSAAM ~ L10OPRC + L10OBPC, data = stream)

Residuals:
     Min       1Q   Median       3Q      Max
-0.09832 -0.02350  0.01076  0.03291  0.08568

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.35762    0.10547  31.835  < 2e-16 ***
L10OPRC      0.44437    0.08925   4.979 1.26e-05 ***
L10OBPC      0.21051    0.06861   3.068  0.00385 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.04937 on 40 degrees of freedom
Multiple R-squared:  0.8749,    Adjusted R-squared:  0.8686
F-statistic: 139.8 on 2 and 40 DF,  p-value: < 2.2e-16
```

If we wish to predict y for a specific pair of values of $x1$ and $x2$, we can simply substitute these into the fitted model:

$$\hat{y} = 3.35762 + 0.44437 \times L10OPRC + 0.21051 \times L10OBPC$$

For example, if $L10OPRC = 2$ and $L10OBPC = 3$, then $L10BSAAM = \hat{y} = 4.87789$.

It is also convention to give a standard error (SE) for any prediction. The formula for the SE of a prediction from a 2 predictor linear regression model is complex (see page 215 of Mead et al, 2003). However, in R it is simple to also return a corresponding SE value using `predict()` and specifying `se.fit=T`. In this case the output will then include an element called "se.fit".

The tricky bit with `predict()` in R is that you need to specify `newdata` (see the help file), which has to be exactly the same structure as the original data. So to repeat the above example in R:

```
# create a new data frame with the variables to predict at
# Note that it does not matter what you put in for L10BSAAM
new.df <- data.frame(L10BSAAM = 0, L10OPRC = 2, L10OBPC = 3)

# now use predict() and specify se.fit=T
predLake <- predict(ML_Mod2, newdata = new.df, se.fit = TRUE)

# the output now has two elements:
# the fit
predLake$fit
```

```
      1
4.877918
```

```
# the se of the fit
predLake$se.fit
```

```
[1] 0.07583833
```

This prediction is well within the range original data set.

> **Question 3a**
>
> Why would the prediction be witin the range of of the original dataset? You can use `range()` to figure this out, or just look at the original data.

More interesting is making prediction not part of the original data, but as we discussed in the lectures this means there is a different confidence interval.

R allows you to define the interval using `interval = "prediction"`. The output will then include both the fitted and the prediction confidence interval and the default is to calculate the 95% confidence interval.

If you want to calculate the actual se.fit from the output, you need to subtract the actual prediction and divide by the $t_{0.05}$ for `df = 40` (which is n - p - 1).

Create a new data frame with the variables to predict. Note that it again does not matter what you put in for the response (dependent) L10BSAAM, we have put in five 0 values.

```
new.df <- data.frame(L10OPRC = seq(3.0, 4.0, length = 5),
                     L10OBPC = seq(3.0, 4.0, length = 5))
```

> ⚠ Question 3b
>
> Now use `predict()` and specify `interval="prediction"`.

Inspect the output; the `lwr` and `upr` columns are the upper and lower prediction intervals. Note that the variation and prediction intervals are fairly small.

Exercise 4: Validation and model prediction quality

Data: *California Streamflow* spreadsheet

In this exercise we will test the quality of the developed model, but doing it formally using a comparison on a validation data set.

We have to once again `set.seed()` to make sure your results are the same across the class.

The first step is to sample 20% of the data as a validation data set from the overall data set. We are doing this by using the function `sample()` to pick a random number of rows. We can use `dim()` to check the dimensions of the datasets.

```
#Only use so we are all get same random numbers -
# otherwise R uses computer time to get random number
set.seed(10)

#Sample 20% of the rows, find row numbers
index <- sample(1:nrow(stream), size = 0.20*nrow(stream))
#Split data
valid <- stream[index,]
dim(valid)
```

```
[1] 8 4
```

```
calib <- stream[-index,]
dim(calib)
```

```
[1] 35  4
```

Rather than rerunning the calibration we are going to reuse the two models from last week with different data and compare the results. We will use the model with 2 variables and the model with 3 variables.

```
# use model 2 and model 3 from topic 7/8 practical (last week)
# and test which one is the best model, but use calib data
ML_Mod2 <- lm(L10BSAAM ~ L10OPRC + L10OBPC, data = calib)
ML_Mod3 <- lm(L10BSAAM ~ L10OPRC + L10OBPC + L10APSAB, data = calib)

# compare the models
summary(ML_Mod2)
```

```
Call:
lm(formula = L10BSAAM ~ L10OPRC + L10OBPC, data = calib)

Residuals:
     Min       1Q   Median       3Q      Max
-0.10393 -0.02396  0.01011  0.03323  0.08423

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.35322    0.11734  28.577  < 2e-16 ***
L10OPRC      0.42263    0.09872   4.281 0.000158 ***
L10OBPC      0.23667    0.07411   3.193 0.003150 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05016 on 32 degrees of freedom
Multiple R-squared:  0.8852,    Adjusted R-squared:  0.878
F-statistic: 123.3 on 2 and 32 DF,  p-value: 9.154e-16
```

```
summary(ML_Mod3)
```

```
Call:
lm(formula = L10BSAAM ~ L10OPRC + L10OBPC + L10APSAB, data = calib)

Residuals:
     Min       1Q   Median       3Q      Max
-0.10398 -0.02560  0.00741  0.02820  0.09170

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.25540    0.13977  23.291  < 2e-16 ***
L10OPRC      0.42590    0.09786   4.352 0.000136 ***
L10OBPC      0.22977    0.07364   3.120 0.003892 **
L10APSAB     0.05279    0.04187   1.261 0.216703
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0497 on 31 degrees of freedom
Multiple R-squared:  0.8908,    Adjusted R-squared:  0.8802
F-statistic: 84.26 on 3 and 31 DF,  p-value: 5.361e-15
```

Based on the results, ML_Mod3 is better based on a slightly better adj r^2. Because the difference is so slight, should do a more thorough investigation which model is better.

You might want to check the residual plots of the predictions. Do you observe anything suspicious?

```
par(mfrow=c(2,2))
plot(ML_Mod2)
```

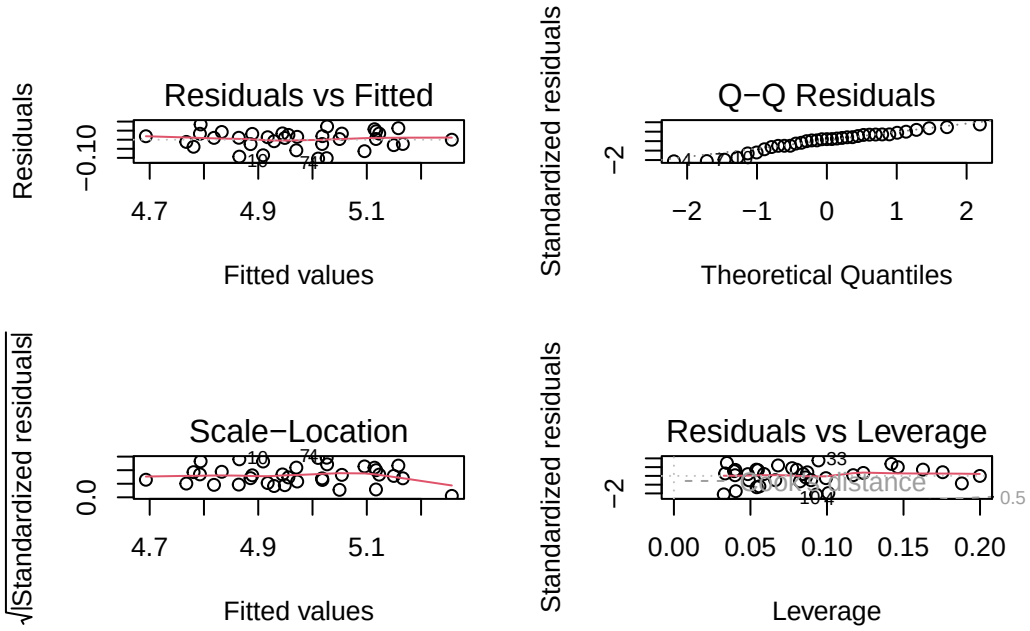

Figure 1: Residual plots for Model with 2 parameters

```
par(mfrow=c(1,1))
```

```
par(mfrow=c(2,2))
plot(ML_Mod3)
```

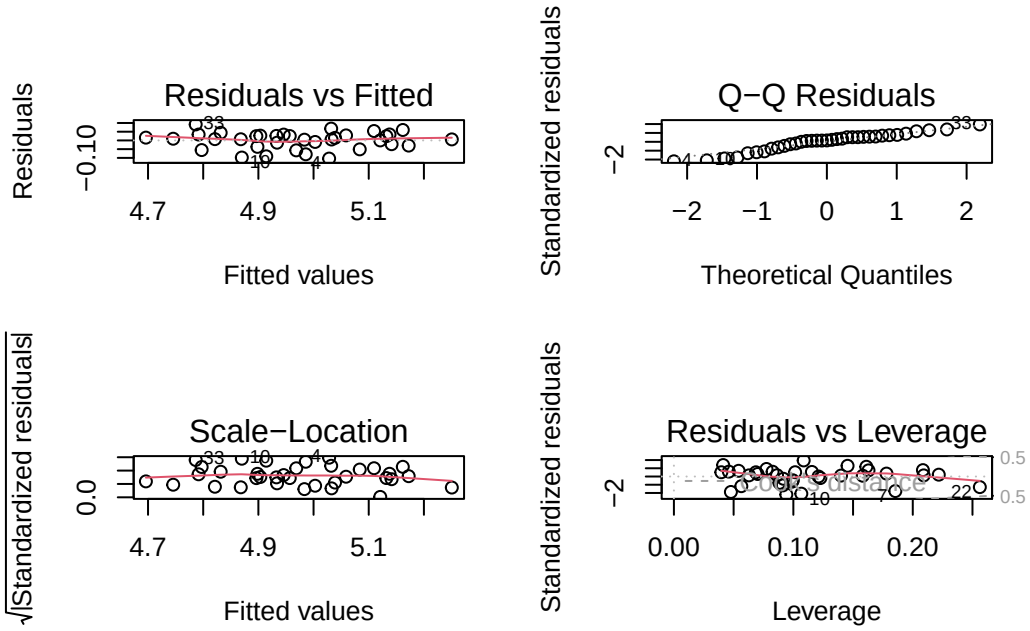

Figure 2: Residual plots for Model with 3 parameters

```
par(mfrow=c(1,1))
```

Question 4

Accuracy: check RMSE and bias of the calibrated models:
Use the equation for RMSE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

and for bias (Mean Error):

$$ME = \frac{1}{n}\sum_{i=1}^{n} y_i - \hat{y}_i$$

Check both the models and for both calibration and validation. To derive the validation data, use (for example for Model 2):

```
predict(ML_Mod2, newdata = valid)
```

```
       1        2        3        4        5        6        7        8 
4.803687 5.129393 4.918709 4.859044 4.924527 4.911822 4.937253 5.064019 
```

We can subsequently make a plot of the calibration and validation data sets for both observed data and the predicted data and compare to the 1:1 line.

```
# plot predicted versus observed
par(mfrow = c(2,1), mar=c(4,4,3,2))

# model 2
plot(calib$L10BSAAM, predict(ML_Mod2),
     # colour = red, type = "16" and size is 20% larger
     pch = 16, col = "blue", cex = 1.2,
     # add titles for axes and main
     xlab = "Observed dataset", ylab = "Predicted",
     main="model 2 predictors")
# insert a 1:1 line, dashed line, width = 2
abline(0, 1, lty = 2, lwd = 2)

# add the validation data
points(valid$L10BSAAM, predict(ML_Mod2, newdata = valid),
       # colour = blue, type = "16" and size is 20% larger
       col = "red", pch = 16, cex = 1.2)
# add a legend to the first plot
legend("topleft", c("calibration", "validation", "1 : 1 line"),
       pch = c(16, 16, NA), lty = c(NA, NA, 2), col = c("blue", "red", 1),
       # 20% smaller
       cex=0.8)

# model 3
plot(calib$L10BSAAM, predict(ML_Mod3),
     # colour = red, type = "16" and size is 20% larger
     pch = 16, col = "blue", cex = 1.2,
     # add titles for axes and main
```

```
     xlab = "Observed dataset", ylab = "Predicted",
     main="model 3 predictors")
# insert a 1:1 line, dashed line, width = 2
abline(0, 1, lty = 2, lwd = 2)
# add the validation data
points(valid$L10BSAAM, predict(ML_Mod3, newdata = valid),
       # colour = blue, type = "16" and size is 20% larger
       col = "red", pch = 16, cex = 1.2)
```

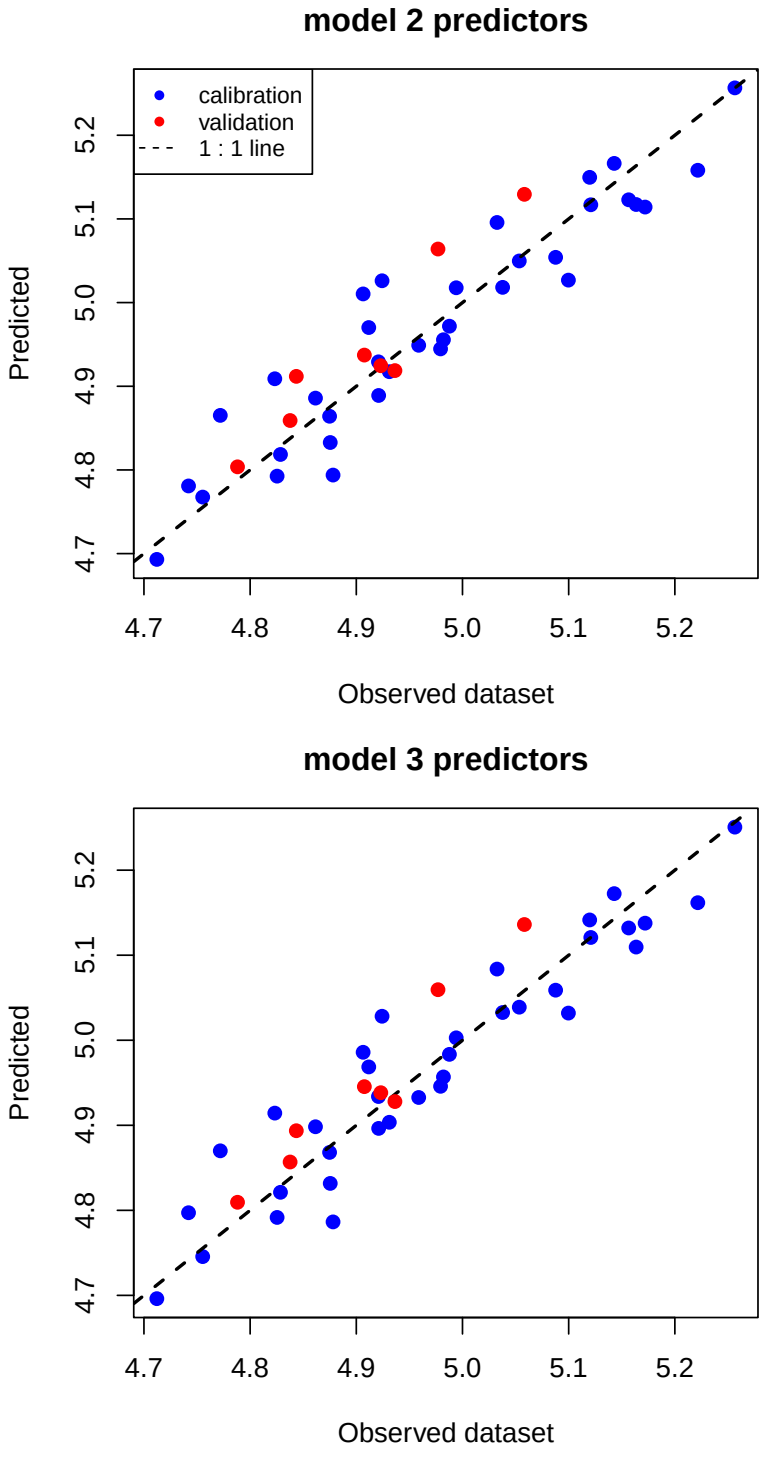

Figure 3: Plots of predicted versus observed for model with 2 predictors (top) and model with 3 predictors (bottom)

This gives the opportunity for a visual inspection, but we can also calculate the correlation (to echo the model derivation) for the calibration and validation data sets. You can also calculate the multiple r^2 by squaring the correlation coefficient. In this case the multiple r^2 is ok to use as we are using it to assess prediction quality rather than the initial model fit.

> ⚠ Question 4b
>
> Calculate correlation using and r^2 using `cor()`

> ⚠ Question 4c
>
> Now have a go at calculating Lin's concordance correlation coefficient, using the lecture slides. Don't forget to call `library(epiR)` and maybe `install.packages("epiR")` if the package is not installed.

> ⚠ Question 4d
>
> Draw conclusions about which model is the best model to predict L10BSAAM from the other variables, use the results to support your argument.

That's it for today! Great work fitting multiple linear regression and trying your hand at some predictive modelling! Next week we jump into stepwise selection and predictive modelling!