Introduction to R for the Geosciences: Regression

Gavin Simpson

30th April — 3rd May

Summary

This practical will use several different data sets to introduce you to various techniques in regression analysis - simple "classical" linear regression analysis, "inverse" linear regression or calibration and multiple linear regression analysis.

1 Simple linear regression

This is the simplest form of regression with a single response or dependent variable y and one predictor or independent variables x. The linear regression model is defined by:

$$y_i = \alpha + \beta x_i + \epsilon_i$$

where α is a constant, the intercept, β is the slope and ϵ_i is the error component.

In this part of the practical class we will model the response of stomatal density (number of stomata mm^{-2} leaf) of modern leaves of *Salix herbacae* (dwarf willow) in relation to atmospheric CO_2 concentrations (ppm by volume, ppmv). Data for 29 collections of leaves growing at different CO_2 concentrations are in the text file `co2.txt`.

You will perform a simple linear regression using R. Refer to your notes from Practical 1 on Exploratory Data Analysis and the introductory notes for R provided at the start of the course. Start R as you have been instructed in previous classes and then load the data file `co2.txt` into an R object.

```
R> co2 <- read.table(file = "co2.txt", header = TRUE, sep = ",")
R> colnames(co2)
R> str(co2)
```

The data contain 29 observations of two variables; `CO2` is the CO_2 concentration and `SD` is the stomatal density. The first thing we should do is plot the data:

```
R> plot(co2$CO2, co2$SD, xlab = expression(CO[2]),
+     main = expression(paste(CO[2], " against Stomatal Density")),
+     ylab = "Stomatal density")
```

Most of this should be familiar to you by now. The only new code is the use of `expression()` and to build up our plot annotation. `paste()` simply sticks its arguments together into a single character vector. `expression()` creates a valid R expression out of its arguments which R then interprets rather than literally reproducing the contents as labels. So instead of the label being printed as CO[2], R interprets the brackets as meaning display this as subscript and formats the text accordingly. R has very powerful capabilities for displaying complex mathematical notation in plots. The code used to format this in a plot borrows heavily from LaTeX; see `?plotmath` for more information.

Linear models in R are specified using the `lm()` function and a standard formula notation that is used in a range of other R base functions and in many add-on packages:

$$response \sim predictor_1 + predictor_2 \ldots + predictor_i$$

To create a linear model of stomatal density as a function of CO_2 type the following:

```
R> co2.lm <- lm(SD ~ CO2, data = co2)
R> summary(co2.lm)

Call:
lm(formula = SD ~ CO2, data = co2)

Residuals:
    Min      1Q  Median      3Q     Max
-33.278  -8.566  -1.153  10.901  34.255

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 294.9904    33.2477   8.873 1.73e-09 ***
CO2          -0.6467     0.1153  -5.608 5.99e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 16.14 on 27 degrees of freedom
Multiple R-squared: 0.538,        Adjusted R-squared: 0.5209
F-statistic: 31.44 on 1 and 27 DF,  p-value: 5.985e-06
```

We then display the ANOVA table for the regression model we just specified. We used the `data` argument of `resid()` to tell R where to find the variables `SD` and `co2` instead of attaching the co2 data frame to the search path. This ensures that the variables being used are the ones you actually specify. The regression summary provides a succinct description of the model specified:

1. The call, which describes the model being summarised,
2. A description of the distribution of the residuals that allows one to quickly evaluate if these are normally distributed,
3. Regression coefficients, standard errors and their associated t-values and significance levels. The stars provide a quick visual guide to the level of significance achieved by each coefficient,
4. Finally the residual standard error, the multiple R-squared (R^2), R^2 adjusted for the degrees of freedom, the F statistic for the model and its associated p-value.

To display the fitted line from the regression first re-plot the data (you can use the up-cursor to move back through the commands you have previously typed, so you do not need to retype everything):

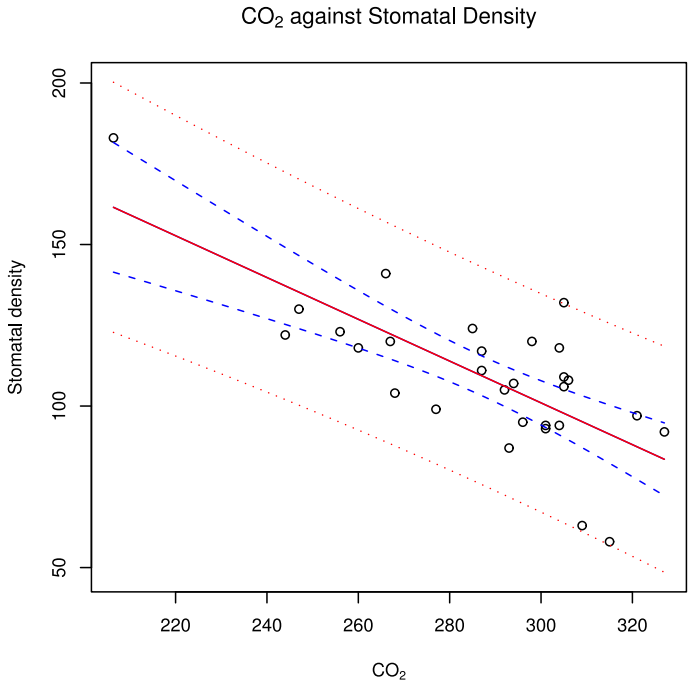
```
R> plot(co2$CO2, co2$SD, xlab = expression(CO[2]),
+     main = expression(paste(CO[2], " against Stomatal Density")),
+     ylab = "Stomatal density")
R> abline(co2.lm, col = "red") #plots the fitted line
```

`abline()` is used for drawing lines on plots. It can be used to plot lines that fulfill many functions (we will use it to plot vertical lines on our diagnostics plots later), but when a linear model (of class `lm()`) is used as an argument, `abline()` extracts the slope and the intercept and plots a line accordingly.

To draw the 95% confidence limits and prediction intervals we need to make use of the function `predict()`. Enter the following code chunk:

```
R> pred.frame <- data.frame(CO2 = seq(min(co2$CO2), max(co2$CO2), length = 50))
R> pp <- predict(co2.lm, int = "p", newdata = pred.frame)
R> pc <- predict(co2.lm, int = "c", newdata = pred.frame)
R> plot(co2$CO2, co2$SD, xlab = expression(CO[2]),
+     main = expression(paste(CO[2], " against Stomatal Density")),
+     ylab = "Stomatal density", ylim = range(co2$SD, pp))
R> matlines(pred.frame$CO2, pc, lty = c(1,2,2), col = "blue")
R> matlines(pred.frame$CO2, pp, lty = c(1,3,3), col = "red")
R> identify(co2$CO2, co2$SD)
```

Figure 1: Stomatal density versus CO_2, fitted line plot (solid) with 95% confidence (dashed) and prediction intervals (dotted)



Q and A

1. Label points on the plot which look like potential outliers or which lie outside the prediction intervals. Remember, after executing the `identify()` funtion you need to click on the plot window to label points, then right click the plot window to finish. You cannot use the R console to enter further commands until you do so—as indicated by the there not being a prompt "$>$" displayed.

This seems much more complicated than it really needs to be! But this illustrates a number of useful R commands. We want to predict stomatal density over the full range of CO_2, not just the discrete CO_2 values in our data. Also, when we plot lines using `plot(x, y, type = "l")` for example, R plots lines between the data points which would result in a cats cradle if the data points are not in order.

First we create a new data frame to hold the CO_2 values we wish to predict stomatal density for. We create a sequence of 50 equally-spaced values ranging from the minimum to the maximum value of CO_2 in our data. We use `predict()` to generate the prediction (`int = "p"`) and confidence (`int = "c"`) values, and instruct R to predict these values for our sequence of CO_2 data. Next comes the familiar plot command where we have added the argument `ylim` to make sure the full range of the prediction interval can be plotted. Finally we use `matlines()` to plot multiple lines from the provided arguments. Effectively we use `matlines()` to plot 3 lines with each call, plotting the confidence interval with dashed blue lines, the prediction interval with dotted red lines and the fitted line (which actually gets plotted twice) in red. The plot should look like Figure 1

1.1 Regression diagnostics

Once we have fitted our linear model, we must examine that model to check for the influence of outliers and whether the assumptions of linear least squares are met by our data. The summary output provided a simple look at the distribution of the residuals from the regression, but we can do much better with some graphical displays.

Firstly, we should plot the residuals of the fitted model to look for patterns in the data or to assess whether the assumption of normally-distributed errors is maintained. The LOWESS smooth line shows the patterns in the data.

```
R> plot(fitted(co2.lm), resid(co2.lm), main = "Residuals vs Fitted",
+       ylab = "Residuals", xlab = "Fitted values")
R> lines(list(x = range(fitted(co2.lm)) * c(-1.05, 1.05), y = c(0, 0)),
+       lty = "dashed", col = "grey")
R> lines(lowess(fitted(co2.lm), resid(co2.lm)), col = "red")
```

Another way of visualising whether the residuals of the models follow a normal distribution is to plot a quantile-quantile or Q-Q plot. The `identify()` function allow us to interactively label points in plots. Try the following:

```
R> co2.resid <- resid(co2.lm)
R> co2.qq <- qqnorm(co2.resid)
R> qqline(co2.resid)
R> identify(co2.qq)
```

After executing the `identify()` command above, click with the left mouse button on some of the points in the Q-Q plot. Label those points that deviate somewhat from the 1:1 line in the plot. To finish labelling points press the right mouse button.

The `resid()` function is called an *extractor function* in R. The `lm()` model object contains a list of the residuals. We could access it directly by accessing the various components of the `co2.lm` object. It is better practice, however, to use extractor functions to get at the parts of larger, more complex objects. Another example of an extractor function is `coef()`, which can be used to extract the regression coefficients from an `lm()` object.

To generate good predictive model that generalises well, it is important to determine whether the model is being unduly influenced by outlier values. We can look at a number of indicators that can allow us detect outliers that have a high degree of influence; hat values, DFBETAS and Cook's distance.

A useful guide for identifying outliers is to look for observations that have a hat value that is 2 or 3 times the average hat value. Enter the following commands and identify which leaves lie beyond these thresholds.

```
R> plot(hatvalues(co2.lm), type = "h", main = "Leverage: hat values",
+       xlab = "Observation", ylab = "Leverage")
R> abline(h = c(2,3) * 2/29, lty = 2)
R> identify(hatvalues(co2.lm))
```

We use the `hatvalues()` function to calculate the hat values which we then plot as *histogram like* bars using the argument `type = "h"`. `abline()` is used to plot the horizontal lines at the given heights, `h`.

We can also display the DFBETAS for the model. Here, you should identify those observations that lie at a distance from the main cluster of points.

```
R> plot(dfbetas(co2.lm), type = "h", main = "Leverage: Dfbetas",
+       xlab = "Observation", ylab = "Dfbetas")
R> abline(h = 0, col = "grey")
R> identify(dfbetas(co2.lm))
```

In a similar theme we can plot the Cook's distances for each observation. Again, identify those leaves that have a Cook's distance greater than the threshold value.

```
R> plot(cooks.distance(co2.lm), type = "h", ylim = c(0, 1),
+       main = "Cook's distance", xlab = "Observation",
+       ylab = "Cook's distance")
R> abline(h = 4/27, lty = 2) # 4/(n-k-1) n = 29, k = 1 parameter in model
R> identify(cooks.distance(co2.lm))
```

We use $4/27$ from the formulae $4/(n-k-1)$, where $n = 29$, $k = 1$ parameter model.

Q and A

1. By now you should have enough information to answer the following question. Which of the leaves is a potential outlier?

Having done all the hard work of generating regression diagnostic plots in R, it is worth noting that R comes with a plot method for linear models that produces some of the diagnostic plots we produced by hand. The following code produces a 2 x 2 grid of plots on the open device (window), draws the diagnostic plots and then resets the plotting region to 1 x 1.

```
R> par(mfrow = c(2,2))
R> plot(co2.lm, ask = FALSE)
R> par(mfrow = c(1,1))
```

To find out more about the lower left plot, look at the help for `plot.lm()` by typing `?plot.lm` at the prompt.

In the `co2.txt` data file one of the leaves (leaf 17, CO_2 of 206 ppmv) is a full-glacial age sample matched against the Byrd ice-core CO_2 measurements for 15 800–18 900 years BP. Generate another linear regression model of stomatal density against CO_2 concentration, this time omitting leaf 17 from the analysis. Look to see if the intercepts and slopes differ from those obtained above, and take note of the standard errors of the regression coefficients.

```
R> co2.lm2 <- lm(SD ~ CO2, data = co2[-17,])
R> summary(co2.lm)
```

Refer back to the instructions on how to plot the data with a fitted line if you wish to visually inspect the differences in the two models.

2 Inverse linear regression

Inverse regression is the opposite of classical regression in that the model we are trying to fit is now;

$$x_i = \alpha + \beta y_i + \epsilon_i$$

where x_i is CO_2 concentration and y_i is stomatal density in our case. Clearly we are not suggesting that CO_2 is influenced by stomatal density! Inverse regression is the most powerful way of calibrating or reconstructing CO_2 from stomatal density as it will nearly always give the lowest root mean square error of prediction (RMSEP).

To fit an inverse regression model of CO_2 on stomatal density in R, we simply take the model notation used in the example above and *flip* the parameters around so that the model we are now attempting to fit is

$$\mathrm{CO}_{2i} = \alpha + \beta \mathrm{SD}_i + \epsilon_i$$

Enter the following code:

```
R> co2inv.lm <- lm(CO2 ~ SD, data = co2)
R> summary(co2inv.lm)

Call:
lm(formula = CO2 ~ SD, data = co2)

Residuals:
    Min      1Q  Median      3Q     Max
-32.560 -16.645   1.288  15.129  36.759

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) 378.0580    16.5698  22.816  < 2e-16 ***
SD           -0.8320     0.1484  -5.608 5.99e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.3 on 27 degrees of freedom
Multiple R-squared: 0.538,        Adjusted R-squared: 0.5209
F-statistic: 31.44 on 1 and 27 DF,  p-value: 5.985e-06
```

```
R> plot(co2$SD, co2$CO2,
+      main = expression(paste("Inverse regression model of ", CO[2], " ~ SD")),
+      ylab = expression(CO[2]), xlab = "Stomatal density")
R> abline(co2inv.lm, col = "red")
```

which fits the inverse regression model, prints the model summary and subsequently plots the model data with the fitted regression line. It would be useful to have an idea of the predictive power of our model. Our function `jackLm()` performs a leave-one-out cross-validation procedure to assess the true prediction error of a linear model.

```
R> source("jackLm.R")
R> co2inv.jack <- jackLm(x = co2$SD, y = co2$CO2)
R> summary(co2inv.jack)

Jackknife estimated prediction error

Apparent errors

               MSE       RMSE        Rsqr            Bias
co2$CO2   311.9113   17.66101   0.5380262   -2.939706e-15

Prediction errors

              MSEP      RMSEP        Rsqr         Bias
co2$CO2   372.7042   19.30555   0.4479855   -0.5912099
```

The first line loads our `jackLm()` function on to the search path. The second line runs the function by specifying the x and y variables. The final line produces a summary of the output from the `jackLm()` function, which shows the apparent statistics (those based on all the data used to predict itself) and the prediction estimates based on the leave-one-out cross-validation[1] process.

Q and A

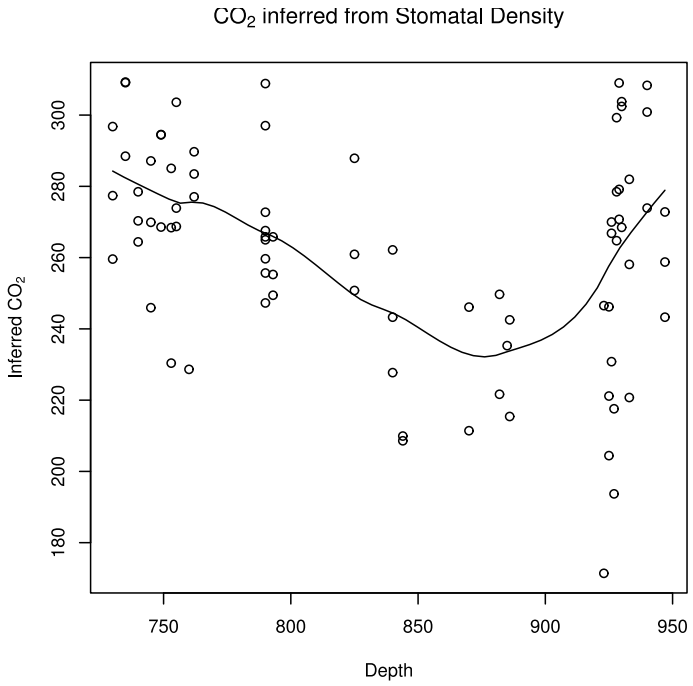
1. Note the differences between the apparent and the prediction errors. What do these values indicate?

Now we can use our inverse regression model to reconstruct past CO_2 concentrations from measurements of stomatal density of fossil *Salix herbacea* leaves from Kråkenes, a late-glacial site in western Norway studied by Hilary Birks. For this we will use the `predict()` function we have seen before so the commands used should be familiar by now. The fossil data are in the file `co2fossil.txt`.

```
R> co2.fossil <- read.table("co2fossil.txt", header = TRUE, sep = ",")
R> pred.co2 <- predict(co2inv.lm, newdata = data.frame(SD = co2.fossil$SD))
R> plot.ord <- order(co2.fossil$Depth)
R> scatter.smooth(co2.fossil$Depth[plot.ord], pred.co2[plot.ord], span = 0.4,
+      ylab = expression(paste("Inferred ", CO[2])), xlab = "Depth",
+      main = expression(paste(CO[2], " inferred from Stomatal Density")))
```

[1] In leave-one-out cross-validation, or jackknifing as it is also called, each sample in turn is left out of the model building process and the resulting model is then used to predict for the omitted sample and the difference between the observed and the predicted value is noted. This process is repeated until all samples have been omitted in turn from the process. The root mean square error of prediction (RMSEP) is the standard error of these differences.

Figure 2: Reconstructed CO_2 inferred from stomatal density



The `pred.ord` object contains the predicted CO_2 values, and `plot.ord` is used to re-order the data before plotting. The bandwidth or span for the smoother is 0.4. The resulting plot should look like Figure 2.

3 Multiple linear regression

So far we have seen how to fit simple linear regression models with a single predictor variable and a single response variable. In this section of the practical we will take a look at multiple regression models in R, where we have a single response variable with two or more predictor variables we wish to model.

The data we will use is breeding density of dippers at 22 10km stretches of different rivers in Wales in relation to 7 predictor variables ($x_1, x_2, \ldots x_7$). The variables are altitude (m), water hardness (mg $CaCO_3 l^{-1}$), gradient (m km^{-1}), caddis fly larvae density, stonefly larvae density, mayfly larvae density, and other invertebrates. The data are in the file dipper.txt and the variables are labelled with the abbreviations *density* (density of dippers, the response variable), *alt*, *whard*, *gradient*, *caddis stonef*, *mayfly*, and *otherin* respectively.

Use R to explore these data and to develop the simplest, most parsimonious statistical model to predict the breeding density of dippers from the predictor variables using a multiple linear regression model of the form:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m$$

where $x_1 \ldots x_m$ are our predictor variables and β_0, $\beta_1 \ldots \beta_m$ are parameters that we wish to estimate for combinations of y and x_m.

Read in the data from `dipper.txt`, display it and plot a scatter-plot matrix of the variables using the following commands.

```
R> dipper <- read.table("dipper.txt", header = TRUE, sep = ",")
R> dipper
R> plot(dipper, gap = 0)
```

It is useful to have a numerical summary of the information shown in a scatter-plot matrix, and it is easy to generate a correlation matrix using the R function `cor()`. `cor()` does not generate p-values or significance values for the correlations however, and the associated function `cor.test()` only works on two vectors, x and y at a time. Our function[2] corProb() will generate the lower triangle of a correlation matrix with associated p-values shown in the upper triangle. Load the corProb function and generate the correlation matrix.

```
R> source("corProb.R")
R> corProb(dipper)

Correlations are shown below the diagonal
P-values of the F-statistics are shown above the diagonal

          alt     caddis  density gradient mayfly  otherin stonef  whard
alt        1.0000  0.1446  0.0606  0.0206   0.7977  0.3036  0.0516  0.7178
caddis     0.3215  1.0000  0.0003  0.0046   0.0199  0.5966  0.0011  0.1313
density    0.4064  0.7036  1.0000  0.0002   0.2197  0.3499  0.0000  0.1099
gradient   0.4900  0.5803  0.7102  1.0000   0.2620  0.6031  0.0024  0.2777
mayfly    -0.0580  0.4924  0.2726  0.2499   1.0000  0.0264  0.4649  0.0213
otherin   -0.2298  0.1194 -0.2093 -0.1173   0.4725  1.0000  0.1876  0.0242
stonef     0.4200  0.6503  0.8416  0.6130   0.1643 -0.2918  1.0000  0.4172
whard     -0.0817  0.3319  0.3503  0.2421   0.4876  0.4786  0.1821  1.0000
```

Multiple linear regression models are fitted in R in the same way as the simple linear regression model you looked at earlier. To fit the full multiple regression model using all seven predictors use the following commands.

```
R> dipper.lm <- lm(density ~ alt + caddis + gradient + mayfly + otherin +
+      stonef + whard, data = dipper)
```

A short cut is:

```
R> dipper.lm <- lm(density ~ ., data = dipper)
```

Instead of entering all the terms on the right-hand side of the manual you can use the "." notation as shorthand; the two models are the same. Look at the model summary:

```
R> summary(dipper.lm)

Call:
lm(formula = density ~ ., data = dipper)

Residuals:
     Min       1Q   Median       3Q      Max
-0.95722 -0.51827 -0.06338  0.50798  1.17849

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.6442064  0.7411133   3.568  0.00309 **
alt          0.0003696  0.0036352   0.102  0.92045
caddis       0.0031921  0.0028478   1.121  0.28119
gradient     0.0505885  0.0383216   1.320  0.20798
mayfly       0.0001083  0.0009079   0.119  0.90674
otherin     -0.0011373  0.0009894  -1.149  0.26964
stonef       0.0032338  0.0012539   2.579  0.02185 *
whard        0.0077542  0.0050468   1.536  0.14672
```

[2]The `corProb()` function is based on code published on the R-Help mailing list by Bill Venables which has been modified to provide a print method.

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7671 on 14 degrees of freedom
Multiple R-squared: 0.8226,        Adjusted R-squared: 0.7339
F-statistic: 9.272 on 7 and 14 DF,  p-value: 0.0002459
```

Note the values for R^2, adjusted-R^2, F and P for this, the full model.

Q and A

1. What do these results suggest to you about the relationship between the predictor variables and the breeding density of dippers?

We can also produce an ANOVA table using R's `anova()` function, which shows the marginal effect of adding each variable in turn to the model. Each variable is added to the model in the order that you specified in the right-hand side of the model equation or in the order they appear in the data frame or matrix from which the predictors were derived.

```
R> anova(dipper.lm)

Analysis of Variance Table

Response: density
          Df  Sum Sq Mean Sq F value    Pr(>F)
alt        1  7.6659  7.6659 13.0291  0.002844 **
caddis     1 16.9972 16.9972 28.8887 9.796e-05 ***
gradient   1  4.8091  4.8091  8.1735  0.012625 *
mayfly     1  0.1427  0.1427  0.2425  0.630032
otherin    1  1.9976  1.9976  3.3952  0.086663 .
stonef     1  5.1869  5.1869  8.8157  0.010154 *
whard      1  1.3890  1.3890  2.3607  0.146715
Residuals 14  8.2372  0.5884
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In addition to using the correlation and scatter-plot matrices to guide variable selection, variance inflation factors (VIF) are often used to identify superfluous variables that might be candidates for exclusion in the modelling building procedure. R does not have a function for calculating VIFs in default state. Instead we need to load the `car` package by John Fox.

```
R> library(car)
R> vif(dipper.lm)

     alt   caddis gradient   mayfly  otherin   stonef    whard
1.482808 2.653830 2.056748 1.849922 2.002146 2.668009 1.677314
```

3.1 Stepwise regression

To try to develop a simpler model we will use stepwise regression procedures to select only the most significant variables for the model. There are a number of methods that you can use in this automated selection procedure; backward elimination, forward selection and optimal, or best, subsets regression for example. In this section we will use all three methods to try to find a simple, yet adequate model to predict the breeding density of dippers.

The first method we will use is backwards elimination. The `stepAIC()` function in the MASS package[3] can be used to perform stepwise selection. Stepwise regression can also br performed using the `step()` function (which is a simpler version of `stepAIC()`. To perform a backwards elimination procedure on the dipper density data enter the following code.

[3]The MASS package is provided in the VR bundle available from CRAN. The MASS package is an accompaniment to the book *Modern Applied Statistics with S* by Bill Venables and Brian Ripley

```
R> library(MASS)                  #load the MASS package
R> dipper.step <- stepAIC(dipper.lm)

Start:  AIC=-5.61
density ~ alt + caddis + gradient + mayfly + otherin + stonef +
    whard

           Df Sum of Sq     RSS     AIC
- alt       1    0.0061  8.2433 -7.5962
- mayfly    1    0.0084  8.2455 -7.5901
- caddis    1    0.7392  8.9764 -5.7218
- otherin   1    0.7773  9.0145 -5.6285
<none>                   8.2372 -5.6125
- gradient  1    1.0253  9.2625 -5.0315
- whard     1    1.3890  9.6261 -4.1844
- stonef    1    3.9132 12.1504  0.9391

Step:  AIC=-7.6
density ~ caddis + gradient + mayfly + otherin + stonef + whard

           Df Sum of Sq     RSS     AIC
- mayfly    1    0.0065  8.2498 -9.5788
- caddis    1    0.7584  9.0017 -7.6598
- otherin   1    0.7765  9.0198 -7.6158
<none>                   8.2433 -7.5962
- gradient  1    1.2313  9.4745 -6.5336
- whard     1    1.3959  9.6392 -6.1545
- stonef    1    3.9955 12.2388 -0.9015

Step:  AIC=-9.58
density ~ caddis + gradient + otherin + stonef + whard

           Df Sum of Sq     RSS     AIC
<none>                   8.2498 -9.5788
- otherin   1    0.8033  9.0531 -9.5345
- caddis    1    0.9150  9.1648 -9.2648
- gradient  1    1.2419  9.4917 -8.4938
- whard     1    1.5106  9.7604 -7.8797
- stonef    1    3.9919 12.2416 -2.8964
```

The default settings for `stepAIC()` include the printing of summary data as the algorithm proceeds and terms are dropped from the model. You can investigate this output to determine which terms were sequentially dropped from the model. At each step the terms are listed in increasing order of *effect* on the model if that term were deleted from the model. The column labelled `Sum of Sq` denotes the change in variance, whilst `RSS` is the change in residual sum of squares, and `AIC` the change in AIC following deletion of that term. The `summary()` function can be used to obtain a summary of the final regression model determined by backwards elimination of terms.

```
R> summary(dipper.step)

Call:
lm(formula = density ~ caddis + gradient + otherin + stonef +
    whard, data = dipper)

Residuals:
     Min       1Q   Median       3Q      Max
-1.01441 -0.52773 -0.04106  0.50172  1.17363
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.7225020  0.3555338   7.658 9.74e-07 ***
caddis       0.0033216  0.0024934   1.332   0.2015
gradient     0.0521194  0.0335831   1.552   0.1402
otherin     -0.0011064  0.0008864  -1.248   0.2299
stonef       0.0032390  0.0011641   2.782   0.0133 *
whard        0.0077861  0.0045489   1.712   0.1063
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7181 on 16 degrees of freedom
Multiple R-squared: 0.8223,        Adjusted R-squared: 0.7668
F-statistic: 14.81 on 5 and 16 DF,  p-value: 1.625e-05
```

As you can see, the resulting model has five terms instead of the original seven.

Q and A

1. Which terms have been deleted from the model?

The `anova` component of the returned `step` object `dipper.step`, which can be accessed as folows;

```
R> dipper.step$anova
```

```
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
density ~ alt + caddis + gradient + mayfly + otherin + stonef +
    whard

Final Model:
density ~ caddis + gradient + otherin + stonef + whard


      Step Df    Deviance Resid. Df Resid. Dev       AIC
1                                 14   8.237170 -5.612484
2    - alt  1 0.006083612        15   8.243254 -7.596241
3 - mayfly  1 0.006529229        16   8.249783 -9.578823
```

Q and A

1. What is the change in deviance, residual deviance and AIC between the full model and that selected by backward elimination?

AIC is a liberal measure of model performance. Bayes Information Criterion (BIC) is a more strict measure of model performance, which penalizes the use of an extra degree of freedom in a greater way than AIC does. `stepAIC()` is a very flexible function and can use BIC instead of AIC in model selection.

The `k` argument of `stepAIC` is used to to determine the penalty for the degrees of freedom used by models. For BIC, $k = log(n)$, where n is the number of observations.

```
R> dipper.step.bic <- stepAIC(dipper.lm, direction = "backward", k = log(22))
R> summary(dipper.step.bic)
R> dipper.step.bic$anova
```

Note that the anova table produced using the final command above labels the final column "AIC", even though we have used BIC in the elimination procedure.

Q and A

1. How many terms were eliminated from the full model using the BIC criterion?
2. What is the difference in residual deviance between the full model and that selected by backward elimination using BIC?
3. What is the difference in residual deviance between the models selected by AIC and BIC as the penality on the degrees of freedom?

Instead of starting at the full model and sequentially eliminating terms an alternative method is to start with a null model and sequentially add terms until the decrease in residual deviance at the cost of an extra degree of freedom[4] is not significant. To perform forward selection with `stepAIC` specify the argument `direction = "forward"`, and do not specify a value for k so that AIC is used for model comparison.

```
R> dipper.step2 <- stepAIC(lm(density ~ 1, data = dipper),
+       scope = ~ alt + caddis + gradient + mayfly
+       + otherin + stonef + whard, direction = "forward")
R> summary(dipper.step2)
R> dipper.step2$anova
```

Q and A

1. How many terms are added to the final model returned by `stepAIC()`? What is the improvement in model performance over the null model? How does this model compare with the AIC-derived model using backward elimination?

As before, we can add a greater penalty on the number of degrees of freedom used by the models assessed by `stepAIC()` by using BIC.

```
R> dipper.step.bic2 <- stepAIC(lm(density ~ 1, data = dipper),
+       scope = ~ alt + caddis + gradient + mayfly
+       + otherin + stonef + whard, k = log(22),
+       direction = "forward")
R> summary(dipper.step.bic2)
R> dipper.step.bic2$anova
```

Q and A

1. How many terms are added to the final model returned by `stepAIC()` using BIC? What is the improvement in model performance over the null model? How does this model compare with the BIC-derived model using backward elimination?
2. Is there any consensus between the models selected in the terms included in the final model? Which of the four models is the most parsimonious?

3.2 Best subsets regression

An alternative to forward selection and backward elimination is *best subsets regression*. Regression models are, in turn, selected using 1 to n parameters, where n is the maximum number of parameters available for model fitting. The best model with a single term is selected from the available explanatory variables. Then the best model using two terms is selected. This continues until a full model using all available terms is returned. At each step models other than the best model can be evaluated. Best subsets regression is available in the `leaps()` function of the `leaps` package. The `car` package contains some utility functions that help interpret the output from the `leaps()` function.

[4] Or more than one degree of freedom if the variable in question is a factor.

```
R> library(leaps)                 #load the leaps package
R> dipper.subsets <- regsubsets(density ~ ., nbest = 5, data = dipper)
R> summary(dipper.subsets, matrix.logical = TRUE)

Subset selection object
Call: regsubsets.formula(density ~ ., nbest = 5, data = dipper)
7 Variables  (and intercept)
         Forced in Forced out
alt          FALSE      FALSE
caddis       FALSE      FALSE
gradient     FALSE      FALSE
mayfly       FALSE      FALSE
otherin      FALSE      FALSE
stonef       FALSE      FALSE
whard        FALSE      FALSE
5 subsets of each size up to 7
Selection Algorithm: exhaustive
           alt caddis gradient mayfly otherin stonef whard
1  ( 1 ) FALSE  FALSE    FALSE  FALSE   FALSE   TRUE FALSE
1  ( 2 ) FALSE  FALSE     TRUE  FALSE   FALSE  FALSE FALSE
1  ( 3 ) FALSE   TRUE    FALSE  FALSE   FALSE  FALSE FALSE
1  ( 4 )  TRUE  FALSE    FALSE  FALSE   FALSE  FALSE FALSE
1  ( 5 ) FALSE  FALSE    FALSE  FALSE   FALSE  FALSE  TRUE
2  ( 1 ) FALSE  FALSE     TRUE  FALSE   FALSE   TRUE FALSE
2  ( 2 ) FALSE   TRUE    FALSE  FALSE   FALSE   TRUE FALSE
2  ( 3 ) FALSE  FALSE    FALSE  FALSE   FALSE   TRUE  TRUE
2  ( 4 ) FALSE  FALSE    FALSE   TRUE   FALSE   TRUE FALSE
2  ( 5 )  TRUE  FALSE    FALSE  FALSE   FALSE   TRUE FALSE
3  ( 1 ) FALSE  FALSE     TRUE  FALSE   FALSE   TRUE  TRUE
3  ( 2 ) FALSE   TRUE     TRUE  FALSE   FALSE   TRUE FALSE
3  ( 3 ) FALSE  FALSE     TRUE   TRUE   FALSE   TRUE FALSE
3  ( 4 ) FALSE   TRUE    FALSE  FALSE   FALSE   TRUE  TRUE
3  ( 5 )  TRUE  FALSE     TRUE  FALSE   FALSE   TRUE FALSE
4  ( 1 ) FALSE   TRUE     TRUE  FALSE   FALSE   TRUE  TRUE
4  ( 2 ) FALSE  FALSE     TRUE  FALSE    TRUE   TRUE  TRUE
4  ( 3 ) FALSE   TRUE    FALSE  FALSE    TRUE   TRUE  TRUE
4  ( 4 ) FALSE  FALSE     TRUE   TRUE   FALSE   TRUE  TRUE
4  ( 5 )  TRUE  FALSE     TRUE  FALSE   FALSE   TRUE  TRUE
5  ( 1 ) FALSE   TRUE     TRUE  FALSE    TRUE   TRUE  TRUE
5  ( 2 ) FALSE  FALSE     TRUE   TRUE    TRUE   TRUE  TRUE
5  ( 3 ) FALSE   TRUE     TRUE   TRUE   FALSE   TRUE  TRUE
5  ( 4 )  TRUE   TRUE     TRUE  FALSE   FALSE   TRUE  TRUE
5  ( 5 )  TRUE  FALSE     TRUE  FALSE    TRUE   TRUE  TRUE
6  ( 1 ) FALSE   TRUE     TRUE   TRUE    TRUE   TRUE  TRUE
6  ( 2 )  TRUE   TRUE     TRUE  FALSE    TRUE   TRUE  TRUE
6  ( 3 )  TRUE  FALSE     TRUE   TRUE    TRUE   TRUE  TRUE
6  ( 4 )  TRUE   TRUE     TRUE   TRUE   FALSE   TRUE  TRUE
6  ( 5 )  TRUE   TRUE    FALSE   TRUE    TRUE   TRUE  TRUE
7  ( 1 )  TRUE   TRUE     TRUE   TRUE    TRUE   TRUE  TRUE
```

The `summary()` function displays the results of the best subsets regression, with the `nbest = 5` argument of `regsubsets()` used to indicate that we want the five best models at each step returned. The `matrix.logical = TRUE` argument of `summary()` forces the printing of a matrix indicating whether variables are included (TRUE) or not (FALSE) in each of the selected models.

The `subsets()` function in the `car` package can be used to provide a graphical summary of the models selected at each stage of a best subsets regression, and provides a means of evaluating which of the subsets is the most parsimonious.

```
R> subsets(dipper.subsets, main = "Optimal Subsets Method")
```

```
         Abbreviation
alt                 a
caddis              c
gradient            g
mayfly              m
otherin             o
stonef              s
whard               w
```

This will draw a plot of the BIC of the models against subset size. Click in an empty area of the plot to place the legend, which indicates which terms each of the letters represents.

To return the single best model at each step, use the following:

```
R> dipper.subsets <- regsubsets(density ~ ., nbest = 1, data = dipper)
R> summary(dipper.subsets, matrix.logical = TRUE)
R> subsets(dipper.subsets, main = "Optimal Subsets Method")
```

```
Subset selection object
Call: regsubsets.formula(density ~ ., nbest = 1, data = dipper)
7 Variables  (and intercept)
         Forced in Forced out
alt          FALSE      FALSE
caddis       FALSE      FALSE
gradient     FALSE      FALSE
mayfly       FALSE      FALSE
otherin      FALSE      FALSE
stonef       FALSE      FALSE
whard        FALSE      FALSE
1 subsets of each size up to 7
Selection Algorithm: exhaustive
           alt caddis gradient mayfly otherin stonef whard
1  ( 1 ) FALSE  FALSE    FALSE  FALSE   FALSE   TRUE FALSE
2  ( 1 ) FALSE  FALSE     TRUE  FALSE   FALSE   TRUE FALSE
3  ( 1 ) FALSE  FALSE     TRUE  FALSE   FALSE   TRUE  TRUE
4  ( 1 ) FALSE   TRUE     TRUE  FALSE   FALSE   TRUE  TRUE
5  ( 1 ) FALSE   TRUE     TRUE  FALSE    TRUE   TRUE  TRUE
6  ( 1 ) FALSE   TRUE     TRUE   TRUE    TRUE   TRUE  TRUE
7  ( 1 )  TRUE   TRUE     TRUE   TRUE    TRUE   TRUE  TRUE

         Abbreviation
alt                 a
caddis              c
gradient            g
mayfly              m
otherin             o
stonef              s
whard               w
```

Remember that you will need to click on the plot to place the legend before you can continue.

Q and A

1. Which model results in the lowest BIC? Are there any other models that perform almost as well as this *best* model?

You should now be able to produce regression summaries, ANOVA tables and plots for the full model and the *best* models chosen by backward elimination, forward selection and best subsets regression.

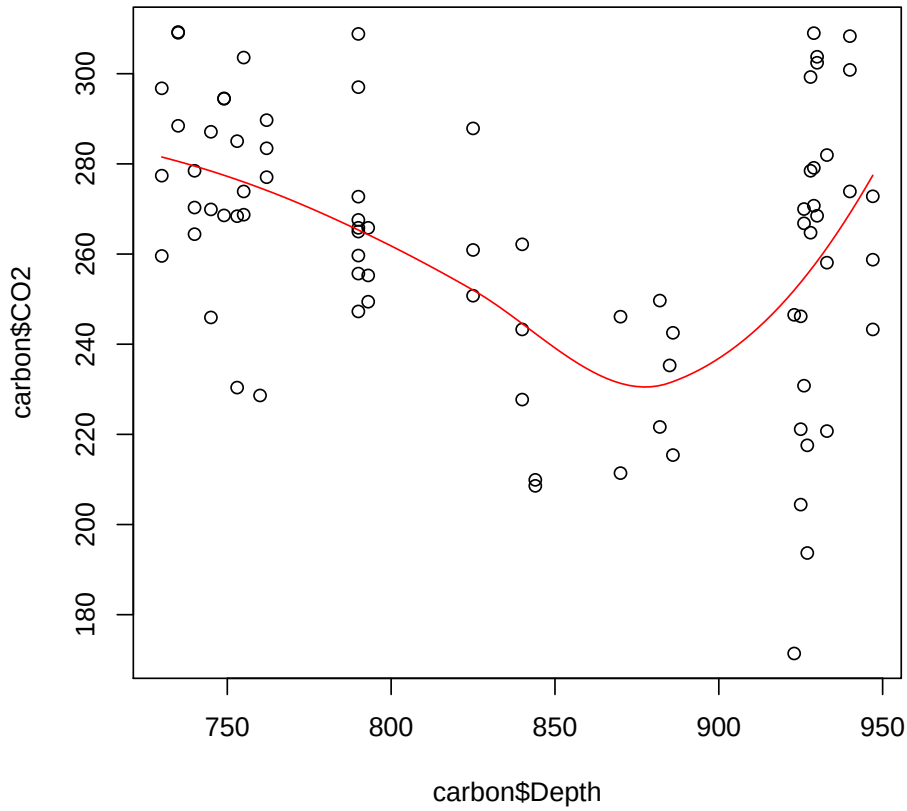


Figure 3: LOWESS smooth line, using default span.

Q and A

1. Compare each of these models.

4 Locally weighted scatterplot smoothers (LOWESS)

In this part of today's practical you will use the reconstructed CO_2 data you generated in the practical yesterday. These data are in the `carbon` object we just created by loading the `co2.csv` data file in to R.

```
R> carbon <- read.csv("co2.csv")
R> co2.low <- loess(CO2 ~ Depth, data = carbon)
R> pred.data <- with(carbon, data.frame(Depth = seq(min(Depth),
+         max(Depth), length = 100)))
R> co2.pred <- predict(co2.low, newdata = pred.data)
R> plot(CO2 ~ Depth, data = carbon, xlim = range(carbon$Depth))
R> lines(pred.data$Depth, co2.pred, col = "red")
```

Most of this should be self explanatory by now; we use the `loess()` function to fit a LOWESS model of CO_2 as a function of depth. `pred.data` is produced to create an evenly distributed set of 100 depth values, which lie between the minimum and maximum depth in our original data. This is then used as new data, with which we predict the CO_2 concentration for each of the 100 depths. Finally we plot the original data and add the fitted LOWESS line in red.

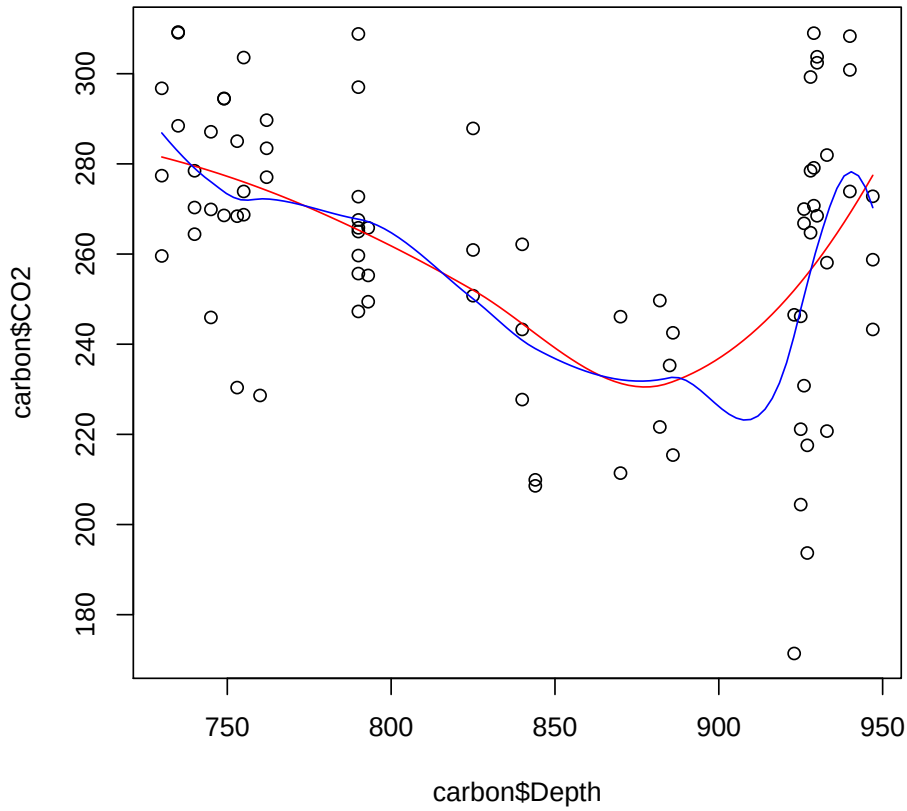


Figure 4: LOWESS smooth lines using different spans: red = 0.75, blue = 0.4.

The LOWESS model we just fit used the default bandwidth or span. In `loess()`, the default span is set at `span = 0.75`, in other words, each window contains 75% of the total data series, over which the local regression is fitted.

Now fit a new LOWESS model, this time using `span = 0.4`:

```
R> co2.low2 <- update(co2.low, span = 0.4)
R> co2.pred2 <- predict(co2.low2, newdata = pred.data)
```

To reserve sufficient plotting space to add both fitted lines to our plot, we calculate the min and the max values (the range) over both fitted lines *and* the original data—this is stored in `y.lim` and is used as the value for the `ylim` argument in the `plot()` call below. Re-plot the data series, add the original LOWESS model fitted line in red (i.e. scroll back up through your commands to repeat them instead of typing the middle two lines below). To view how the change in `span` has affected the fitted line, add the fitted line from the new model to the current plot:

```
R> y.lim <- range(co2.pred, co2.pred2, carbon$CO2)
R> plot(carbon$Depth, carbon$CO2,
+         xlim = range(carbon$Depth), ylim = y.lim)
R> lines(pred.data$Depth, co2.pred, col = "red")
R> lines(pred.data$Depth, co2.pred2, col = "blue")
```

Q and A

In the Kråkenes core, the Allerød-Younger Dryas boundary is at 925cm. and the Younger Dryas-Holocene boundary is at 756.5 cm.

1. How has the atmospheric CO_2 concentration fluctuated during the late-glacial?
2. Investigate the effects of changing the span on the LOWESS smoother.
3. Remember, the span is the smoothness parameter—as span increases the curve becomes smoother and is a function of the fraction of the total number of points used to fit the local regressions.
4. Try spans of 0.2, 0.25, 0.3, 0.35, 0.4, 0.5, 0.6, 0.75 and 0.8 and decide which is the most useful and realistic portrayal of the actual data points. As Cleveland (1993) says of LOWESS, "*the choice of the parameters must be based on a combination of judgement and trial and error. There is no substitute for the latter*".
5. Plot a couple of smoothers at a time, using various line colours (e.g. use `col = colour` in your `lines()` calls substituting `colour` with valid R colours (e.g. `"red"`, `"green"`, `"blue"`, `"orange"`) and line types (e.g. use `lty = type` in your `lines()` calls substituting `type` with valid R line types (e.g. `"soild"`, `"dashed"`, `"dotted"`, `"dotdash"`). You must also create a set of new predictions for the fitted line for each model. See the example below which shows you how to fit 4 separate LOWESS models with differing spans:

```
R> co2.low3 <- update(co2.low, span = 0.2)
R> co2.pred3 <- predict(co2.low3, newdata = pred.data)
R> co2.low4 <- update(co2.low, span = 0.25)
R> co2.pred4 <- predict(co2.low4, newdata = pred.data)
R> y.lim <- range(co2.pred, co2.pred2, co2.pred3, co2.pred4, carbon$CO2)
R> plot(carbon$Depth, carbon$CO2,
+         xlim = range(carbon$Depth), ylim = y.lim)
R> lines(pred.data$Depth, co2.pred, col = "red")
R> lines(pred.data$Depth, co2.pred2, col = "blue")
R> lines(pred.data$Depth, co2.pred3, col = "green")
R> lines(pred.data$Depth, co2.pred4, col = "orange")
```

5 Generalised linear models — GLMs

5.1 Binomial GLM — Species response curves

The first thing to do is to load the package required for today'sthis section of the practical and read in the relevant data sets. These data are in the analogue package and are the classic Imbrie and Kipp foraminifera data and summer sea-surface temperature data.

```
R> library(analogue)
R> data(ImbrieKipp)
R> data(SumSST)
```

We also need to perform some data manipulations. We want to model the relationship between summer sea surface temperature (`SumSST`) and species abundance for several species of foraminifera. The species data is currently as percentages but we require these are proportions, so we divide the data set by 100

```
R> ik <- ImbrieKipp / 100
```

We can use techniques already encountered on the course to fit the equivalent Gaussian logit models using the `glm()` function. Notice that we have to *isolate* the quadratic terms to get R to parse the formula as we want

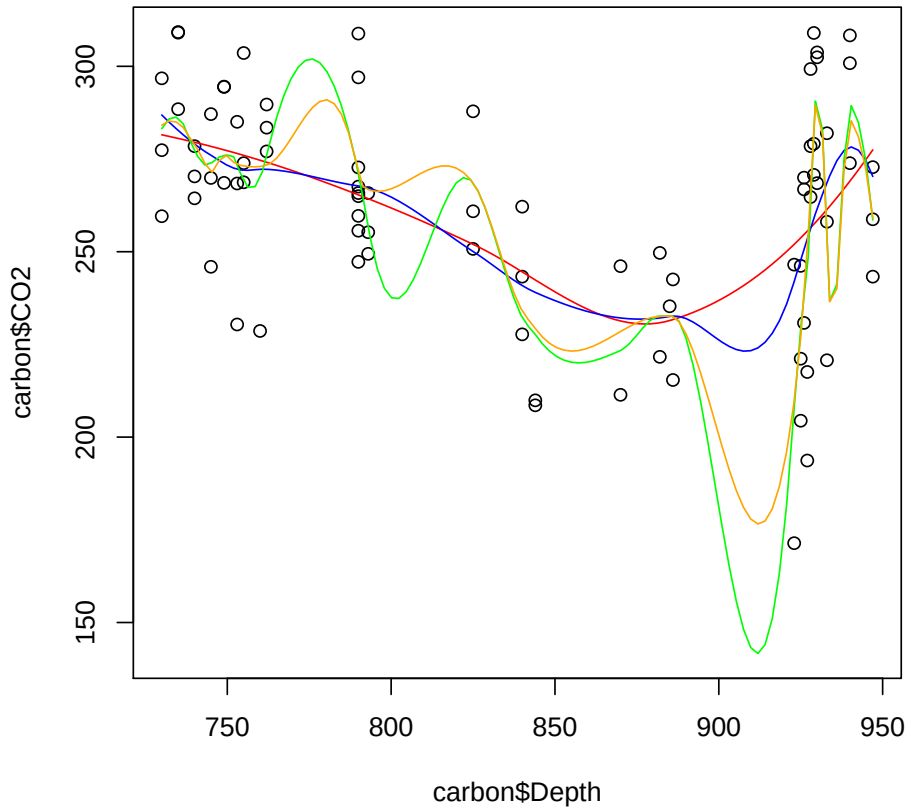


Figure 5: LOWESS smooth lines using a variety of spans.

```
R> mod1 <- glm(G.pacL ~ SumSST + I(SumSST^2), data = ik, family = binomial)
R> mod2 <- glm(G.pacR ~ SumSST + I(SumSST^2), data = ik, family = binomial)
R> mod3 <- glm(G.ruber ~ SumSST + I(SumSST^2), data = ik, family = binomial)
```

Look at the summaries of the three models or use `anova` to see if the models are significant fits for the three species, e.g.

```
R> summary(mod1)

Call:
glm(formula = G.pacL ~ SumSST + I(SumSST^2), family = binomial,
    data = ik)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.70141  -0.02808  -0.02244   0.05358   1.26854

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  6.538975   5.537073   1.181    0.238
SumSST      -0.704146   0.877313  -0.803    0.422
I(SumSST^2)  0.005742   0.034025   0.169    0.866
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 33.8165  on 60  degrees of freedom
Residual deviance:  3.0819  on 58  degrees of freedom
AIC: 13.645

Number of Fisher Scoring iterations: 10
```

```
R> anova(mod1)
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: G.pacL

Terms added sequentially (first to last)


            Df Deviance Resid. Df Resid. Dev
NULL                           60     33.817
SumSST       1  30.7097        59      3.107
I(SumSST^2)  1   0.0249        58      3.082
```

To visualise the response curve for the three species, we first create some SumSST values to predict the proportions of the species at and then use the `predict` function to get predictions on the scale of the response

```
R> pdat <- data.frame(SumSST = seq(min(SumSST), max(SumSST), length = 100))
R> p1 <- predict(mod1, pdat, type = "response")
R> p2 <- predict(mod2, pdat, type = "response")
R> p3 <- predict(mod3, pdat, type = "response")
```

The final step then is to plot these predictions and the associated SumSST values, plus the observed data

```
R> layout(matrix(1:3, ncol = 3, byrow = TRUE))
R> plot(G.pacL ~ SumSST, data = ik, pch = 21, col = "red", bg = "yellow", cex = 1.5)
R> lines(p1 ~ SumSST, data = pdat, col = "blue", lwd = 3)
R> plot(G.pacR ~ SumSST, data = ik, pch = 21, col = "red", bg = "yellow", cex = 1.5)
R> lines(p2 ~ SumSST, data = pdat, col = "blue", lwd = 3)
R> plot(G.ruber ~ SumSST, data = ik, pch = 21, col = "red", bg = "yellow", cex = 1.5)
R> lines(p3 ~ SumSST, data = pdat, col = "blue", lwd = 3)
R> layout(1)
```

Using the fitted models, it is easy to extract the estimated coefficients and compute the GLM-based optima for the three species. The `coef` function is used to extract the coefficients we require.

```
R> m1.c <- coef(mod1)
R> m1.opt <- - m1.c[2] / (2 * m1.c[3])
R> m2.c <- coef(mod2)
R> m2.opt <- - m2.c[2] / (2 * m2.c[3])
R> m3.c <- coef(mod3)
R> m3.opt <- - m3.c[2] / (2 * m3.c[3])
R> c(m1.opt, m2.opt, m3.opt)

  SumSST   SumSST   SumSST
61.31317 15.26847 29.92634
```

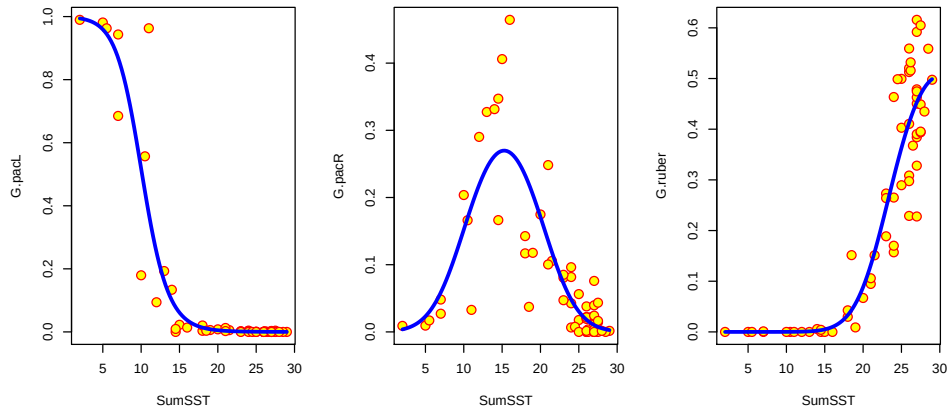


Figure 6: Observed abundance and fitted response curve for *Globigerina pachyderm* left-coiling, *Globigerina pachyderm* left-coiling and *Globigerina ruber* along a gradient of summer sea surface temperature

Q and A

1. Can you suggest a reason for the GLM-derived optima for *G. pachyderm* L?

5.2 Gamma GLM — simple age depth modelling

In tihs section of the practial you'll look at the radiocarbon dates from the peat core studied by Maddy and Brew. Load the data

```
R> peat <- read.csv("maddy_peat.csv", row.names = 1)
R> head(peat)
```

A small amount of data processing is required to add depth and calibrated age mid-points

```
R> peat <- within(peat, {
+     midDepth <- upperDepth - ((upperDepth - lowerDepth) / 2)
+     calMid <- calUpper - ((calUpper - calLower) / 2)})
R> head(peat)
```

Plot the age depth data with the following code

```
R> plot(calMid ~ midDepth, data = peat, pch = 21, bg = "black")
```

A linear model assuming Gaussian errors can be fitted using `lm()`

```
R> m1 <- lm(calMid ~ midDepth, data = peat)
R> summary(m1)

Call:
lm(formula = calMid ~ midDepth, data = peat)

Residuals:
     Min       1Q   Median       3Q      Max
-270.725  -26.404    4.399   52.143  255.496

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 170.2435    58.6819   2.901   0.0158 *
midDepth     12.4133     0.2759  44.995 7.07e-13 ***
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 148.4 on 10 degrees of freedom
Multiple R-squared: 0.9951,        Adjusted R-squared: 0.9946
F-statistic:  2025 on 1 and 10 DF,  p-value: 7.074e-13
```

```
R> anova(m1)

Analysis of Variance Table

Response: calMid
          Df   Sum Sq  Mean Sq F value    Pr(>F)
midDepth   1 44570413 44570413  2024.5 7.074e-13 ***
Residuals 10   220151    22015
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The fitted model and 95% confidence interval can be visualised with the following code chunk

```
R> pdat <- with(peat,
+              data.frame(midDepth = seq(min(midDepth), max(midDepth),
+                         length = 100)))
R> m1.p <- predict(m1, newdata = pdat, se.fit = TRUE)
R> resiDF <- m1$df.residual
R> crit.t <- qt(0.975, df = resiDF)
R> pdat <- within(pdat, fitted <- m1.p$fit)
R> pdat <- within(pdat, {
+     upperCI <- fitted + (crit.t * m1.p$se.fit)
+     lowerCI <- fitted - (crit.t * m1.p$se.fit)})
```

The linearity assumption seems reasonable, but the least squares model is possibly inappropriate given the known increase in uncertainty in the dating with increasing depth. A gamma GLM with identity link function retains the linearity of accumulation rates but allows the variance of the response to increase with increasing depth. A gamma GLM is fitted using

```
R> m2 <- glm(calMid ~ midDepth, data = peat, family = Gamma(link = "identity"))
R> summary(m2)

Call:
glm(formula = calMid ~ midDepth, family = Gamma(link = "identity"),
    data = peat)

Deviance Residuals:
      Min         1Q     Median         3Q        Max
-0.196221  -0.012606  -0.001604   0.050645   0.092314

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 181.0393    26.0842   6.941 3.99e-05 ***
midDepth     12.2807     0.5025  24.441 3.00e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Gamma family taken to be 0.005924447)

    Null deviance: 10.439047  on 11  degrees of freedom
Residual deviance:  0.063394  on 10  degrees of freedom
```

```
AIC: 148.83

Number of Fisher Scoring iterations: 4
```

```
R> anova(m2, test = "F")
```

```
Analysis of Deviance Table

Model: Gamma, link: identity

Response: calMid

Terms added sequentially (first to last)


         Df Deviance Resid. Df Resid. Dev      F    Pr(>F)
NULL                        11    10.4390
midDepth  1   10.376        10     0.0634 1751.3 1.455e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Q and A

1. Is the gamma GLM significant?
2. Locate the regression coefficient. What does this tell you about the accumulation rate of peat estimated from the model?

The fitted gamma GLM can be visualised using

```
R> m2.p <- predict(m2, newdata = pdat, se.fit = TRUE)
R> pdat2 <- pdat
R> pdat2 <- within(pdat2, fitted <- m2.p$fit)
R> pdat2 <- within(pdat2, {
+     upperCI <- fitted + (crit.t * m2.p$se.fit)
+     lowerCI <- fitted - (crit.t * m2.p$se.fit)})
```

Q and A

1. How does the fitted gamma GLM differ from the least squares model fitted earlier?

6 Generalised additive models — GAMs

6.1 Species response curves

In this section of the practical, you'll look at fitting a GAM to model the relationship between abundance and summer sea-surface temperature for one of the species of foraminifera investigated earlier. First load the mgcv package so it is ready for use

```
R> require(mgcv)
```

The model is fitted in much the same way as before, except that the formula contains a smooth function of the covariate, indicated using the `s()` function.

```
R> g1 <- gam(G.pacR ~ s(SumSST), data = ik, family = binomial)
R> summary(g1)
```

```
Family: binomial
Link function: logit

Formula:
G.pacR ~ s(SumSST)

Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.8442     0.6264   -4.54 5.62e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
            edf Ref.df Chi.sq p-value
s(SumSST) 1.896  2.375  4.237   0.158

R-sq.(adj) =  0.623   Deviance explained = 66.6%
UBRE score = -0.85849  Scale est. = 1         n = 61
```

```
R> plot(g1)
```

Q and A

1. How many degrees of freedom are used by the fitted smooth function?
2. Is the smooth statistically significant? What is this test testing?
3. Compare this to the GLM response curve fitted earlier. How do the two models relate?
4. Using code from the bimoial GLM, see if you can draw the fitted response curve from the GAM on top of the species data.

6.2 Smoothing sediment core data

In this section you'll fit a GAM to a well-studied strontium isotope ratio record. First, load the package containing the data and plot it

```
R> require("SemiPar")
R> data(fossil)
R> xlim <- with(fossil, rev(range(age)))
R> plot(strontium.ratio ~ age, data = fossil, xlim = xlim,
+      ylab = "Strontium Isotope Ratio",
+      xlab = "Age (Myr)")
```

Next, fit the GAM. Here we use REML smoothness selection, which has reecntly been shown to perform well at recovering underlying relationships in simulation tests.

```
R> m <- gam(strontium.ratio ~ s(age), data = fossil, method = "REML")
R> summary(m)

Family: gaussian
Link function: identity

Formula:
strontium.ratio ~ s(age)

Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 7.074e-01  2.551e-06  277241   <2e-16 ***
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
         edf Ref.df  F p-value
s(age) 8.244   8.84 88  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.881   Deviance explained =   89%
REML score = -930.01  Scale est. = 6.9006e-10  n = 106
```

Q and A

1. How many degrees of freedom are used by the fitted smooth function?
2. Is the smooth statistically significant?
3. Can you think of any problems there may be with the statistical approach used here?

Finally, plot the fitted model through the observed data points

```
R> pdat <- with(fossil, seq(min(age), max(age), length = 100))
R> pgam <- predict(m, newdata = data.frame(age = pdat))
R> xlim <- with(fossil, rev(range(age)))
R> ylim <- with(fossil, range(pgam,strontium.ratio))
R> plot(strontium.ratio ~ age, data = fossil,
+      xlim = xlim, ylim = ylim,
+      ylab = "Strontium Isotope Ratio",
+      xlab = "Age (Myr)",
+      main = "GAM fit; TPRS; REML smoothness selection")
R> lines(pgam ~ pdat, col = "darkorange", lwd = 3)
```

7 Shrinkage methods — ridge regression, the lasso, and the elastic net

In this section, you'll take a brief look at shrinkage methods as alternatives to the stepwise selection techniques explored earlier. You'll recreate the analysis of the ozone air quality data from the Los Angeles basin, USA, shown in the lecture handouts. First load the data, which are stored in the earth package. In addition, a file containing a custom plotting function is provided that gives a nicer plot the the default supplied with the glmnet package.

```
R> require(earth)
R> require(vegan)
R> require(glmnet)
R> source("shrinkage_funs.R")
```

load the data and display the first few rows to get a feel for the data. The ai is to model the ozone concentration as a function of the other covariates

```
R> ## load the Ozone data
R> data(ozone1)
R> ## look at the first few data points
R> head(ozone1)

  O3   vh wind humidity temp  ibh dpg ibt vis doy
1  3 5710    4       28   40 2693 -25  87 250  33
```

```
2  5 5700    3       37   45  590 -24 128 100  34
3  5 5760    3       51   54 1450  25 139  60  35
4  6 5720    4       69   35 1568  15 121  60  36
5  4 5790    6       19   45 2631 -33 123 100  37
6  4 5790    3       25   55  554 -28 182 250  38
```

glmnet doesn't fit ridge regression but we can fake it by setting a very low value of the α parameter (remember α controls the mixing of the lasso and ridge penalties in the elastic net). A low value for α indicates a stronger weighting towards the ridge penalty. Unlike many other modelling functions, `glmnet()` has a rudimentary interface where we specify the vector response and matrix of predictor variables as separate argumnts, not via a model formula. The response (ozone concetration) is log transformed to account for the larger variance at higher ozone concentrations. The `cv.glmnet()` performs a k-fold CV to help choose the optimal ridge penalty.

```
R> ## ridge regression
R> mod.r <- glmnet(data.matrix(ozone1[,-1]), log(ozone1[,1]), alpha = 0.00001)
R> cv.r <- cv.glmnet(data.matrix(ozone1[,-1]), log(ozone1[,1]), alpha = 0.00001)
```

First, plot the results of the CV

```
R> cvPlot(cv.r)
```

Q and A

1. Identify the model with the minimum CV error and the simplest model within 1 standard error of the best. Roughly what value of the penalty term is indicated for each? Remember these are on the log scale.

The coefficient paths can be visualised using the `coefPlot()` function

```
R> op <- par(mar = c(5,4,4,4) + 0.1)
R> coefPlot(mod.r, label = TRUE, allDF = FALSE)
R> par(op)
```

The best fitting and 1SE model λ values can be found in the `cv.r` object

```
R> cv.r
```

The regression coefficients at that value of λ can be extracted using the `coef()` method and these can be compared with the coefficients of a linear model fit

```
R> coef(mod.r, s = cv.r$lambda.1se)
```

```
10 x 1 sparse Matrix of class "dgCMatrix"
                          1
(Intercept) -3.410788e+00
vh           8.166141e-04
wind         3.217844e-03
humidity     4.193660e-03
temp         1.162285e-02
ibh         -6.204606e-05
dpg          1.662007e-03
ibt          1.690589e-03
vis         -7.799448e-04
doy         -4.482533e-04
```

```
R> ozone.std <- cbind.data.frame(O3 = ozone1$O3, scale(ozone1[, -1]))
R> coef(lm(log(O3) ~ ., data = ozone.std))
```

```
 (Intercept)           vh         wind     humidity         temp          ibh
 2.212966878 -0.001314598 -0.017787819  0.100997214  0.432689476 -0.153233420
         dpg          ibt          vis          doy
 0.019537543  0.037588810 -0.066622801 -0.106588393
```

Q and A

1. Compare the coefficients for the ridge regression with those for the linear model. How do the estimated coefficients differ between the two methods?

2. Which coefficients have been shrunk

3. Repeat the above exerise using the λ for the model with minimum CV error.

The lasso model path and CV can be fitted using similar code. Here the α parameter is set to 1 to put full weight on the lasso penalty.

```
R> ## lasso
R> mod.l <- glmnet(data.matrix(ozone1[,-1]), log(ozone1[,1]), alpha = 1)
R> cv.l <- cv.glmnet(data.matrix(ozone1[,-1]), log(ozone1[,1]), alpha = 1)
```

The elastic net penalty can be fitted by suppling a value of α between 0 and 1. Here we choose equal weighting of the lasso and ridge penalties by setting $\alpha = 0.5$

```
R> ## elastic net
R> mod.el <- glmnet(data.matrix(ozone1[,-1]), log(ozone1[,1]), alpha = 0.5)
R> cv.el <- cv.glmnet(data.matrix(ozone1[,-1]), log(ozone1[,1]), alpha = 0.5)
```

Q and A

1. For each of the lasso and elastic net models, draw the CV plot to help identify what value of the penalty shoud be chosen. Use the 1SE rule to identify the best model. Remember you can also extract the λ for this model from the `$lambda.1se` component of the fitted model.

2. Repeat the above exerise using the λ for the model with minimum CV error.Once you have identified the values of λ for each technique, extract the coefficients for that value of λ and compare them with the least squares coefficients. How similar are they?

3. Finally, draw the coefficient paths for the lasso and elastic net models. Describe how the three paths differ between the various shrinkage methods applied.