STAT0023 Workshop 2: regression and ANOVA in R

This workshop looks at linear regression modelling in R, along with the analysis of variance which, as explained in the lecture, can be regarded as a special case of linear modelling. From time to time it may be helpful to refer to the slides or handouts from the lecture, because it is assumed that students are familiar with the relevant theory as summarised there. The slides / handouts can be downloaded from the STAT0023 Moodle page, under the 'Course overview and useful materials' tab.

1 Setting up

1.1 Material needed

For this workshop, in addition to the lecture slides / handouts you will need:

- These instructions, which you have presumably downloaded already from the 'Week 2' tab on Moodle.
- The R scripts `Workshop2_UStemps.r` and `Workshop2_iris.r`, also available from the 'Week 2' tab. You should save these to the appropriate place on your `N:` drive, as you created it during Workshop 1. Refer to Section 1 of the notes for Workshop 1 if necessary — and remember to make sure that your web browser doesn't change the filenames when downloading them! (Section 1.3 of the notes for Workshop 1).
- The data file `UStemps.rda` — again available from the 'Week 2' tab. This is an 'R data' file: it is in a special format that can be used to store R objects. R data files are an alternative to ASCII files for transferring data between computers, but they can only be read (and written) using R whereas ASCII files are completely portable.

1.2 Starting RStudio

Having downloaded the files, start up RStudio and **use the Session menu to set the working directory to the place where you saved them** (see Section 1.5 of the notes for Workshop 1 if you've forgotten how to do this — but if you *have* forgotten then probably you should be paying a bit more attention to what you're doing during the workshops ...).

2 The US temperature data

In the lecture, we looked at an analysis of January minimum temperatures in different cities thoughout the USA. Your first exercise in this workshop is to look at these data in more detail. Having set the working directory, open the script `Workshop2_UStemps.r` by clicking on it in the 'Files' tab of the lower right-hand pane of the RStudio window. It will display in the top left-hand pane.

As in Workshop 1, you should now work through this script slowly and carefully, one line at a time, at each stage making sure you understand exactly what the commands are doing. Make a note of the commands that are used, and what they do. If you don't understand anything, ask.[1]

If you work effectively, this exercise should take you about an hour. At the end of it, you should understand the following aspects of linear modelling in R:

- How to use the `lm()` command to fit a linear model, including the use of the "`I()`" construction to include quadratic terms and the use of the colon "`:`" to specify interactions.
- How to use the `summary` method for objects of class `lm`, and how to interpret the results — in particular, for purposes of testing hypotheses about individual coefficients, finding an estimate of the error standard deviation and discovering the proportion of variance explained by the model.
- How to use the `update()` command to modify an existing model.
- How to use the `anova()` command to compare nested models using an F test, and to interpret the results.
- How to use the `plot()` command to check some of the modelling assumptions, and how to interpret the resulting plots. You may wonder what the red lines are in the "residuals vs fitted" and "scale-location" plots — see the Appendix for more on this.
- How to identify and deal with influential observations. Note that there is no single 'right answer' here: an appropriate strategy will usually be context-dependent. This is discussed extensively in the script comments.
- How to use the `predict()` command to predict the response variable at new covariate values, and also to compute the standard errors of the predictions.

When you have finished, if you want a record of the output then you can use `sink()` and `source` to send a 'clean' copy of the script output to a file. Refer to the material from Workshop 1 to remind yourself how to do this.

In addition to these aspects of linear modelling, the script introduces several other new techniques relating to data processing and to graphics — including the careful construction of a colour scale for plotting that sends the 'right message' to readers (the colour scale construction is quite advanced!). You should also understand the following, therefore:

- How to use the `pretty()` command to split the range of a vector into a 'tidy' set of intervals, and how to use the `cut()` command to group a continuous variable — so that each group can be plotted in a different colour, for example.

[1] For those of you who are interested in `ggplot()` graphics, there is an optional section at the end of the script, showing you how to use `ggplot()` to produce the exploratory map of the data. If you're not interested in `ggplot()` however, you can skip this part.

- How to use the `par()` command for finer control of your graphics — for example, to get more than one plot on a page, change the sizes of the plot margins and control where the axis labels go.

- How to use the `image()` and `contour()` commands to display functions of two variables. In this case, the variables are Latitude and Longitude and the functions being displayed are the fitted regression functions. You might think that this kind of display is only useful for geographical maps as here, but in fact it is useful much more widely for displaying the results of *any* regression model with two continuous covariates: this is sometimes called 'response surface analysis', because the regression function is a surface. Obviously, if the regression function is linear then such a map is not particularly informative; but if it is nonlinear (as with the quadratic model fitted to the US temperatures) then it can be helpful to visualise it in this way.

2.1 Failures of assumptions

In this particular example, the diagnostic plots and analysis of influential observations indicated that there wasn't too much wrong with the quadratic temperature model, except perhaps for some systematic variation in temperatures that hasn't been captured. We haven't said much about what to do if there *were* some failures of assumptions however. So: here are some ways in which you might spot the failure of assumptions, and their consequences:

- **Systematic structure in mean residuals**: you might spot this in the 'residuals versus fitted values' plot, or (as in the US temperatures example) by plotting the residuals against the covariates. Such structure indicates that the modelled representation of the regression function is inadequate. Whether this matters is context-specific: you may decide, for example, that the residuals are all so small that your model is already predicting well enough.

- **Non-constant variance**: you might spot this in the 'residuals versus fitted values' plot which may exhibit a 'funnel' shape e.g. if the residual variance seems higher at one end than the other, or in the 'scale-location' plot where you might see that the absolute residuals tend to be larger at one end of than the other. In this case, the least-squares estimates are not fully efficient (i.e. you're not making the best use of your data), and the reported standard errors will be incorrect — as will the results of any hypothesis tests and confidence interval calculations.

- **Non-normal residuals**: you might spot this in the normal Q-Q plot, if the residuals don't fall roughly on a straight line. Actually, with one exception this is not critical in large samples: an analogue of the Central Limit Theorem ensures that there is a very wide range of situations under which the least-squares estimates have an approximate multivariate normal distribution in large samples, regardless of the original distribution of the errors. The exception is where you want to calculate prediction intervals for future observations: these will only be accurate if the future observations do indeed have an approximate normal distribution.

- **Lack of independence**: common situations in which it might be a problem are when data are collected at successive time points, or at a collection of spatial locations.[2] Although none of the 'standard' plots is designed to check for this, lack of independence can lead to apparent structure in some of the residual plots. For example, if observations are collected sequentially in time and successive residuals are highly correlated, this can give the appearance of curvature in the "residuals versus fitted values" plot: the curvature is not due to a nonlinear relationship between response and covariates, but simply due to the fact that neighbouring residuals are similar to each other because they are correlated.

If it seems that one or more assumptions does not hold, here are some options for fixing the problem:

- Transform the response variable and / or covariate (but only if the resulting model makes scientific sense). Sometimes, a relationship can be made more linear by taking logs or square roots of one or more quantities; transforming the response variable can also help to make the residual variance more constant, and to make the assumption of normality more reasonable. Don't take logs (or square roots) of quantities that could be negative, though!

- Add additional terms to the model. For the temperature data for example, you might consider extending the quadratic model to include third-degree terms in latitude and longitude; or additional covariates such as altitude if these were available.

- If the diagnostics suggest that the residual variance is related to the fitted values (e.g. the 'residuals versus fitted values' plot has a funnel shape) and there is good reason to suspect that the responses follow non-normal distributions (e.g. because they are counts, so that Poisson distributions might be more appropriate) then a more general class of models may be appropriate — such as generalized linear models (GLMs), which we will examine in Workshop 6.

2.2 Further work

If you have time, you might consider extending the quadratic model in the script to include cubic and maybe even quartic terms: you can compare models and see whether there is any significant improvement as you add more terms. You might also like to plot the fitted surfaces from one or other of these models, following the examples in the script for the quadratic model. Note that a cubic function of variables x_1 and x_2 contains terms in x_1, x_2, x_1^2, x_2^2, x_1x_2, x_1^3, x_2^3, $x_1x_2^2$ and $x_1^2x_2$. A quartic model contains all terms of order up to 4. If you want to implement these models, you will find the `update()` command very helpful! Its use is illustrated in the script that you have just worked through.

[2] Note that although the US temperatures are collected at spatial locations, they are averages over a 30-year period and therefore it seems reasonable to assume that any systematic variation in these data represents genuine structure that should be modelled in the regression function.

3 Regression with factor covariates: the iris data again

In the lecture, we saw that the 'classical' one-way analysis of variance can be regarded as a linear regression model with 'dummy' covariates used to represent which group each observation belongs to. Your next exercise in this workshop is to revisit the iris data from Workshop 1, to understand how to use factor covariates in regression models. The commands for this are in file `Workshop2_iris.r`. Open this and work through it in the usual way. This should take you about 45 minutes. When you have finished, you should understand the following:

- What is the relationship between a 'classical' analysis of variance and a linear regression with dummy covariates.
- How to interpret the summary of a linear regression model with factor covariates, depending on the constraints that have been imposed on the coefficients.
- How to interpret interactions between factors and continuous covariates in regression models.
- How to use R to calculate confidence intervals for regression functions and to plot them.
- How to use the `step()` command to do stepwise regression, if you absolutely have to (and that you should never use this unless you absolutely *do* have to!)

4 Time to write a script!

By now, you have seen four fairly comprehensive scripts during the course and have worked through them: hopefully they made sense to you. Of course, reading someone else's script isn't the same as writing your own. In the final exercise for this week's workshop therefore, your task is to write a script of your own that will compute the least-squares estimates and their standard errors using the appropriate mathematical formulae, for the first model that you fitted to the US temperature data.

Start by opening a new script file: to do this, click the File menu in RStudio and select New File and then R Script. This creates a blank pane for you to work in. Then go through the steps below. You may want to save your work (i.e. the script) as you go. Make sure you save it in the same place as the rest of this week's work, with a `.r` suffix to the file name.

1. Write a short header section to the script, to remind yourself in the future of what it does. You might also want to add comments to your script as you go through, again to help you if you want to read it again in the future.
2. Load the US temperature data from file `USTemps.rda`.
3. Create the design matrix for the linear model, and store this in a matrix called `X`. You may find the following helpful:

- The design matrix needs to have three columns: the first will be a column of ones, the second will contain the latitudes of the cities and the third will contain the longitudes.
- To create a vector containing the right number of ones, you can use something like `ones <- rep(1,nrow(ustemp))`.
- If you want to 'stack' vectors side-by-side to form a matrix, you can use the `cbind()` command ('bind columns together').

4. Copy the minimum temperatures into a response vector `Y`, using `Y <- ustemp$min.temp`.

5. Calculate the least-squares estimates of the regression coefficients as $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$, and store the result in a vector called `beta.hat`. Print this result, and check that it agrees with the result that you obtained from `lm()` in Section 2 above. You will need to remember the following:

   - The `t()` command is used to transpose a matrix, so $\mathbf{X}'$ is computed as `t(X)`.
   - Matrix multiplication is achieved using the `%*%` symbol, so (for example) $\mathbf{X}'\mathbf{X}$ can be computed as `t(X) %*% X`.
   - To invert a matrix, use the `solve()` command.

   You can find more on all of this in the 'Introduction to R' document, that you were asked to read before the start of the course and that is available from the 'Course overview and useful materials' tab of the Moodle page.

6. Calculate the vector of fitted values $\hat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ (you will need to use matrix multiplication again), and hence calculate the vector of residuals $\mathbf{e} = \mathbf{Y} - \hat{\mathbf{Y}}$.

7. Calculate the estimated error standard deviation as $\hat{\sigma} = \sqrt{\frac{1}{n-k}\sum_{i=1}^{n} e_i^2}$ where e_i is the ith residual, n is the total number of observations and k is the number of coefficients estimated. You may find it convenient to use `n <- nrow(X)` and `k <- ncol(X)`. And use the `sum()` command to sum the squared residuals!

8. Calculate the estimated covariance matrix of the least-squares estimates as $\hat{\sigma}^2\,(\mathbf{X}'\mathbf{X})^{-1}$ and store this in a matrix called `Covmat`.

9. Calculate the standard errors of the least-squares estimates, as the square roots of the diagonal elements of `Covmat` (the `diag()` command gives you the diagonal elements of a matrix). Print the result, and check that it agrees with the result that you obtained in Section 2 above.

When you have done this, congratulate yourself: you have just done your first serious piece of statistical computing for this course!

5 Moodle quizzes

There are more Moodle quizzes for this week. If you have time left in the workshop, try them a few times. If not, you will need to finish the workshop exercises *and* try out the Moodle quizzes in your own time. Remember: you're expected to spend time on this course outside the classroom, just the same as for any other course.

Appendix: smooth curves in diagnostic plots

When you `plot()` a linear model object in R, two of the plots have red curves on them by default. These are:

- The "residuals versus fitted" plot, which shows the residuals $\{e_i\}$ against the fitted values $\{\hat{\mu}_i\}$: any apparent structure here suggests a lack of fit of the model — for example, there may be some covariates missing, or perhaps the relationship between covariates and response isn't really linear.
- The "scale-location" plot, which plots $\sqrt{|e_i|}$ against $\hat{\mu}_i$ and is designed to examine the assumption of constant residual variance. The rationale is that in practice, if the variance isn't constant then often it varies systematically (e.g. increasing or decreasing with the expectation of the response variable). In this case, you may be able to see it in the "residuals versus fitted" plot; but it may be clearer in the scale-location plot where higher variances will tend to produce larger absolute residuals.

These plots are fine if you have relatively small datasets. However, it isn't always easy to see what's happening. For example:

- If you have a very large dataset, each of these plots will just appear as a mass of points and it may be impossible to see anything useful.

- If the distribution of fitted values is very non-uniform (e.g. if you have hundreds of points at one end of the plot but only a few points at the other end), then the range of residuals is likely to be greater in areas where there are more points even if the modelling assumptions are satisfied.

To aid interpretation therefore, R adds red curves to the plots by default. These are *non-parametric regression curves*: they show how the expected value of e_i (or $\sqrt{|e_i|}$) is estimated to vary with $\hat{\mu}_i$. If the model assumptions are satisfied, $\mathbb{E}(e_i) = 0$ throughout and $\mathbb{E}\left(\sqrt{|e_i|}\right)$ is a constant — so the red curves in both plots should be roughly horizontal lines. Bear in mind, though, that the curves are computed from the data and hence subject to sampling variation. This can be substantial if your sample size is small-ish, so don't overinterpret the curves! To suppress them, include `add.smooth=FALSE` in your `plot()` command e.g. `plot(Temp.Model2, which=1:4, add.smooth=FALSE)`.

You haven't yet encountered nonparametric regression: we'll cover it in Workshop 6. The idea is to allow the data themselves to determine the form of the regression relationship without making any assumptions that the relationship follows a predetermined mathematical form such as a straight line. There are several techniques for producing nonparametric regression curves: the one used in the `lm` diagnostic plots is a 'locally weighted scatterplot smoother' (lowess), which can be implemented in R directly using the `lowess()` command.

How does lowess work?

For the 'residuals versus fitted' plot, the basic idea is: let $m(\mu)$ be the expected value of the residual when the fitted value is μ. Then the relationship between the residuals and fitted

values can be written as $e = m(\mu) + z$ say, where $m(\cdot)$ is a smooth function (if the linear model assumptions are satisfied then $m(\mu) \equiv 0$ and $e = z$, but the more general formulation here is designed to allow specifically for the fact that $m(\mu)$ might not be zero everywhere).

In the first instance, suppose we want to estimate $m(\mu_0)$ for a single specific value μ_0. If we're prepared to assume that $m(\cdot)$ is continuous and differentiable, then a Taylor expansion tells us that for μ close to μ_0, $m(\mu) \simeq m(\mu_0)+b_1(\mu-\mu_0)$ say, where $b_1 = m'(\mu_0)$. In other words: in a small enough neighbourhood of μ_0, the relationship between μ and $m(\mu)$ is approximately linear. So, to estimate the value of $m(\mu_0)$, we could select all of the observations for which $|\hat{\mu}_i - \mu_0|$ is 'small', carry out a linear regression of the residuals $\{e_i\}$ upon the quantities $\{\hat{\mu}_i - \mu_0\}$ using just these observations, and then take the intercept in this regression as an estimate of $m(\mu_0)$. This procedure could then be repeated for different values of μ_0, to build up a picture of the curve $m(\mu_0)$ as μ_0 varies.

In practice, the lowess procedure incorporates two refinements of the procedure described above. The first is that instead of selecting all observations for which $|\hat{u}_i - \mu_0|$ is 'small', all observations are used but each observation is assigned a 'weight', w_i say: the weight is high if $\hat{\mu}_i$ is close to μ_0, and (close to) zero if $|\hat{\mu}_i - \mu_0|$ is very large. A weighted least-squares linear regression of the $\{e_i\}$ upon the $\{\hat{\mu}_i - \mu_0\}$ would then minimise

$$\sum_{i=1}^{n} w_i \{e_i - [b_0 + b_1 (\hat{\mu}_i - \mu_0)]\}^2 \ .$$

with respect to b_0 and b_1. However, the *second* refinement in the lowess procedure is that the fitting is done using a robust regression technique (you learned about these in the workshop) instead of least squares in order to reduce the influence of outliers etc.

Exactly the same procedure is used to produce the smooth curve on the scale-location plot.

There are several choices to be made when implementing the methods described above. The most important is probably the choice of the weights $\{w_i\}$. Details of this are beyond the scope of this course: the `lowess()` command handles all of this automatically.

The lowess procedure is not only useful for residual plots, of course: the idea can also be used to enhance scatterplots when carrying out exploratory analysis, for example to judge whether a relationship may be approximately linear if this isn't clear from the scatterplot alone. As usual though: beware overinterpretation in small samples! The techniques to be covered in Workshop 6, which allow for the construction of uncertainty bands around the estimated regression functions, may be preferable in this context.