NRES 798 — Lab 8

Regression

1 The trees data set

Load the `trees` data set: `data(trees)`. Examine the `summary` and `str`. See the variables description in the *Help* (`?trees`).

"Girth" is actually diameter at breast height, change the variable name to "Dbh" (use `names` for that). Change the values to metric (`trees$Dbh <- 2.54 * trees$Dbh` cm, etc. 1 foot = 0.3048 m). Check if the numbers make sense.

Do `pairs(trees)`. What is it?

2 Nonlinear regression

We want a relationship for estimating volume from height and/or dbh measurements. See your previous graph, which variable looks more promising? What kind of curve might work?

2.1 One predictor

Let's try dbh alone first (measuring heights is more hassle). Plot volume over diameter, diameter squared, diameter cubed. Set `xlim` and `ylim` to include the origin ($\mathit{Dbh} = \mathit{Volume} = 0$). By the way, besides `plot(x, y)`, one can also use a formula: `plot(y ~ x, data)`. Just be careful of enclosing expressions with the *identity* function `I()`, because the arithmetic operators in formulas can have special meanings: `Dbh^2` does not work, `I(Dbh^2)` does (try it!).

It looks like we could use $V \approx \beta_1 D^{\beta_2}$, for some β_1 and β_2. Use `nls` to estimate the parameters: `nls(Volume ~ ..., data=trees,` `start=c(b1=..., b2=...))`. In `nls` it is not necessary to use `I()`. For initial estimates you can use a rough estimate of the slope from one of your graphs. Interpret the output. See what happens if the starting values are really bad. In nonlinear least-squares, and in general optimization algorithms, there is a danger of converging to a spurious local optimum; the only defense is to use decent starting points, and/or several different ones.

One can get more info from the result by using `summary` or other functions, so that it can be useful to store the result, e.g., `fit <- nls(...)`. Use `summary`. Study the output. *Std. Error* is a rough estimate of the parameter standard error (standard deviation of $\hat{\beta}_i$), s_{β_i}, based on a linear approximation. The *t value* is $\hat{\beta}_i / s_{\beta_i}$ (check!), a test statistic for the hypothesis $H_0 : \beta_i = 0$. The test statistic has a t-distribution, and the last column are the p-values. What are the results of these hypothesis tests? Would you have expected otherwise? The *residual standard error* is another name for the SE of regression; check that it is the square root of the RSS (obtained before) divided by the degrees of freedom $n - 2$.

A more interesting H_0 might be $\beta_2 = 2$, for example. One could use the test statistic $(\hat{\beta}_i - 2)/s_{\beta_i}$ (note that with the alternative hypothesis $\beta_2 \neq 2$ the test should be two-sided). An easier way is to run the fit again, reparametrizing $\beta_2 \to 2 + \beta_2$ (why?). Do it. Compare the parameter estimates to the previous ones. Is the hypothesis rejected?

Plot *Volume* over *Dbh*, as before, and add the fitted model. You can use `curve` with the parameter `add=TRUE`.

2.2 Two predictors

Let us try now including *Height* as a predictor, $V \approx \beta_1 D^{\beta_2} H^{\beta_3}$. Fit this with `nls`, call the result `fit2`. You can start with the previous estimates and $\beta_3 = 0$, which is the same as the currently best model (yes?). Stepping up from simpler models is a good strategy.

Better? Compare the RSS and RSE. The p-value for β_3 suggest that it is not 0. Think about what exactly that hypothesis might mean. Another formal test for model differences uses ANOVA and an F-ratio: `anova(fit, fit2)`, try it.

Notice that a null hypothesis $\beta_1 = 0$ would not be rejected. What might that mean?

A number of functions extract further information from the regression result.

`coefficients(fit)` or `coef(fit)` returns the parameter estimates. `fitted` gives the estimated y for all the data points, and `residuals` or `resid` gives the residuals, observed minus estimated. The function `predict` can be used to estimate y for new observations. Calculate the `summary` and the standard deviation (`sd`) of `residuals(fit2)`. Is the mean close to 0? (is equals 0 in linear models). Compare the SD to the RSE, why the difference? Hint: denominator.

A model should be checked ("validated") by plotting the residuals. Plot the residuals of `fit2` over the estimated volume, over dbh, and over height. Add the 0 line with `abline(0, 0)` or `abline(h=0)`. We look for lack of pattern, balanced positive and negative values, and homogeneous variance. Notice the poor residuals over height in `fit`. A common mistake is to plot residuals over the *observed* y; patterns would be expected in that case (can you see why?). A q-q plot of the residuals can be used to check for normality (`qqnorm / qqline`).

Some of the graphs suggest an increase of variance with tree size (*heterocedasticity*), contrary to the regression theory assumptions. If that happens, estimates are still approximately unbiased (exactly unbiased in linear models), but not as efficient as they could be (the variance of the parameter estimates is inflated). Things may be improved by transforming y, usually with the logarithm. Another way is to use *weighted regression*, making `weights` in `nls` or `lm` a vector proportional to the residual variance. For instance, the graphs might suggest a residual standard deviation proportional to height, so the weights equal the heights squared. An equivalent result is obtained by dividing the left- and right-hand sides of the model equation by a value proportional to the standard deviation, in this example fitting V/H instead of V (note that the variance of V/H is the variance of V divided by H^2).

3 Linear regression

Pretty much the same as above, except that the models are of the form

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_{p-1} x_{p-1} \ .$$

The case $p = 2$ is called *simple linear regression*, and $p > 2$ is *multiple linear regression.* Instead of `nls`, the *R* function is `lm` (linear model). The same `lm` is used with categorical variables in ANOVA and ANCOVA. For regression, the first argument is a formula of the form `y ~ x1 + x2 + ...`. Here `+`

does not mean sum, but inclusion of variables. Remember to protect any expressions within `I()`. The intercept can be excluded with `-1`.

Using our `trees` data set, fit a model $V = \beta_0 + \beta_1 D^2$. Proceed as with `nls`, but the formula is `Volume` $\sim$ `I(Dbh^2)`. No starting values are used, estimates are obtained from explicit equations, not an iterative algorithm. Notice any differences with the output of `nls`.

In fact, the model could be fitted with `nls`, try it. But `lm` is more efficient (saves a few milliseconds!), and importantly, more reliable (no false convergences), and can produce some additional information.

Fit and compare the following models (the fit statistics for different y's cannot be directly compared):

$$\begin{aligned}
V &= \beta_0 + \beta_1 D^2 H \\
V &= \beta_0 + \beta_1 D^2 + \beta_2 H + \beta_3 D^2 H \\
\log V &= \beta_0 + \beta_1 \log D + \beta_2 \log H \\
V/H &= \beta_0/H + \beta_1 D^2 \\
V/D^2 &= \beta_0 + \beta_1 H
\end{aligned}$$

In the second one, drop non-significant terms, one at a time.

Do graphical examinations of residuals for those models. Plotting the `lm` result, e.g., `plot(lfit2)`, gives several diagnostic graphs, see `?plot.lm`.

Some of the models predict negative volumes for small D and H (how can you tell?). Any implications for the choice of regression data?

Classical ANOVA tables for regression are produced as `anova(lfit2)`.

4 R Commander

If you have time left, or sometime later, try out the *R Commander*. If you have not already done it install the package: `install.packages("Rcmdr")` (or use the *Tools* menu. That can take a while.

Load it: `library(Rcmdr)`. Explore the menus. *R Commander* types the necessary *R* commands for you. It can be useful for infrequent users, and/or may help in learning *R*. Of course, only a small part of the whole system is available that way, but many of the common statistical analyses are there. Note that many calculations are done by special functions; in principle, one can find out what they do by typing the function name and pressing *Enter*.