Nonlinear Regression and Weights

Statistics 516, Homework 2

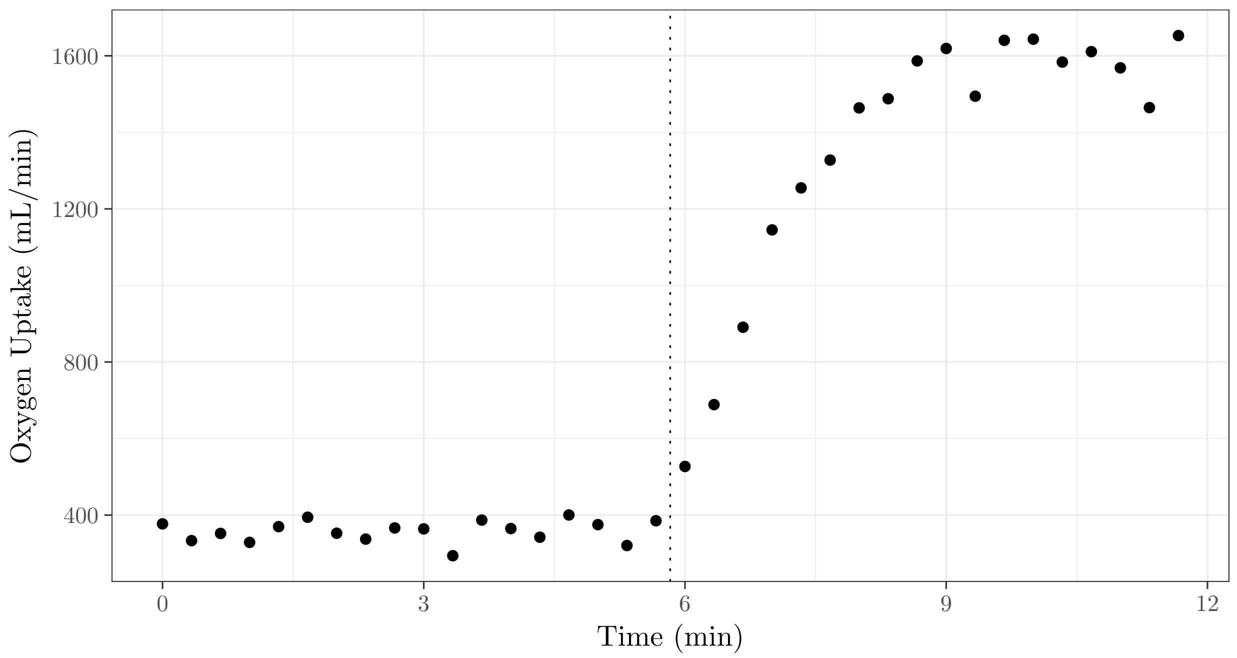
Instructions

1. This assignment is due by the beginning of class on Wednesday, March 8th. A hard copy is required unless you have made arrangements with me in advance submit an electronic copy. Late assignments will **not** be accepted.
2. Your solutions must be **typed** and **very** neatly organized. I will not try to infer your solutions if they are not clearly presented. Equations need not be typeset perfectly but they should be clear. You may substitute letters for symbols (e.g., b1 for β_1), and you may write-out equations (neatly) by hand if necessary.
3. You must include with your solutions the relevant R output **and** R scripts that created them. Be sure that you provide sufficient code that I can replicate your results. Include both the code and the output within the text of your solutions (not in an appendix) using cut-and-paste. But edit your output so as to provide only that which is relevant to answering the questions. Use a monospace font (e.g., Courier or Monoco) for R code and output for clarity. Do not use a monospace font for text that is not R code or output.
4. When using R output to answer a specific question, be sure to demonstrate that you know where in the output the answer can be found. This can be done in the text of your answer (e.g., "As can be seen from the output using the `anova` function, the value of the F test statistic is 2.5.") or you could highlight the relevant part of the output (e.g., bold or italics) and refer the reader to that. It should be clear to me that you know where in the output the answer to a question can be found.
5. Plots from R Studio can be exported in various formats or directly to the clipboard using the "export" menu in the top-left part of the plot panel.
6. It is permitted for you to discuss the homework with other students in the course. However you must still write your own R scripts, produce your own output, and write up your own solutions.
7. You are welcome to ask me questions. I will be happy to clarify what I am asking in any of the questions. I will also be particularly open to helping with any R problems. If you email me with a R question, it will usually be helpful for you to include enough of your R script so that I can replicate your analysis/error. However you should avoid leaving asking questions to the last minute such as the night before or the day that the homework is due. Those are busy times for me and I will not feel obligated to respond if you have clearly left too much to the last minute.

Oxygen Kinetics and the 6-Minute Walk Test

The data frame `O2K` from the **nlstools** package contains data from a 6-minute walk test of a patient with a pulmonary disease. The results from such a test have diagnostic value for patients with pulmonary diseases. The data are plotted below.

```r
p <- ggplot(O2K, aes(x = t, y = VO2)) + geom_point()
p <- p + xlab("Time (min)") + ylab("Oxygen Uptake (mL/min)")
p <- p + geom_vline(xintercept = 5.83, linetype = 3) + theme_bw()
plot(p)
```



In this test the oxygen uptake of the patient was observed at several time points. At first the patient is observed during a rest with relatively constant oxygen uptake, and then the patient starts walking and oxygen uptake increases toward a maximum.[1] The dotted line shows that time point at which the patient started walking, which was at 5.83 minutes from the beginning of observation. A nonlinear regression model that is useful for modeling the change in oxygen uptake over time is

$$E(Y_i) = \begin{cases} \theta_1, & \text{if } t_i \le \delta, \\ \theta_1 + (\theta_2 - \theta_1)(1 - e^{-(t_i - \delta)/\theta_3}), & \text{if } t_i > \delta, \end{cases}$$

where Y_i is oxygen uptake (in mL/min) of the i-th observation, t_i is time (in minutes) of the i-th observation, and δ is the time (in minutes) at which the patient starts walking. The three unknown parameters are θ_1 (the expected oxygen uptake at rest — i.e., when $t_i \le \delta$), θ_2 (the maximum expected oxygen consumption — i.e., the upper asymptote as $t \to \infty$), and θ_3 which is related to how quickly expected oxygen consumption approaches the θ_2. It can be shown that $\theta_3 = t^*/\log(2)$ where t^* is the minutes of exercise (walking, not resting)

[1]The data from the rest phase after walking are not given, but they are not necessary for the modeling.

required for the expected oxygen consumption to be at halfway between the expected energy consumption at rest (θ_1) and the maximum expected energy consumption (θ_2). Note that for these data δ is *not* an unknown parameter. It is known that the patient started walking at $\delta = 5.83$ minutes into the test.

1. Estimate the *nonlinear* regression model described above using the data in the `VO2` data frame and report your results using `summary` and `confint`. There are several ways that you could specify this model to accommodate the case-wise nature of the function. One approach is to use one or two indicator variables for whether the time point is greater than δ of less than or equal to δ. Another approach is to use the `ifelse` function.

2. Create a plot that shows the raw data *and* the estimated curve from he previous problem that shows the relationship between time and expected oxygen uptake. You can start with the code given above.

3. It might also be useful to make inferences concerning the *change* in expected oxygen consumption from rest to the maximum during walking. In terms of the model parameters this is simply $\theta_2 - \theta_1$. While a point estimate of this quantity can be obtained as $\hat{\theta}_2 - \hat{\theta}_1$, but more involved calculations are necessary to compute the standard error. But note that $\theta_2 - \theta_1$ is a *linear combination*, and inferences concerning a linear combination of parameters can be obtained using the `lincon` function from the **trtools** package.[2] A linear combination of parameters from a nonlinear regression model can be written as

$$\ell = a_1\theta_1 + a_2\theta_2 + \cdots + a_q\theta_q,$$

where $\theta_1, \theta_2, \ldots, \theta_q$ are the q model parameters, and $a_1, a_2, \ldots, a_q$ are user-specified coefficients that define the desired linear combination of the model parameters. Use `lincon` to obtain a point estimate, standard error, and confidence interval for $\theta_2 - \theta_1$.

[2]Many (but not all) quantities of interest from a regression model can be written as a linear combination of the model parameters. Later we will consider how to make inferences concerning quantities that *cannot* be written as linear combinations of model parameters.

Iteratively Weighted Least Squares

As we will discuss in lecture, the parameters of a Poisson regression model are usually estimated using a technique known as *maximum likelihood.* However for generalized linear models (which includes Poisson regression), there is a very close connection between maximum likelihood estimation and *weighted least squares* estimation. Consider for simplicity but without loss of generality a log-linear regression model with a single explanatory variable such that

$$E(Y_i) = e^{\beta_0} e^{\beta_1 x_i}.$$

The parameters of this model (i.e., β_0 and β_1) could be estimated using `nls`, but inferences would assume *homoscedasticity.* If Y_i is a count then it might have a Poisson distribution in which case the *variance* of Y_i is also $E(Y_i)$. To account for heteroscedasticity using weighted least squares, the weight w_i should be inversely proportional to the variance of Y_i, which suggests using weights of $w_i = 1/E(Y_i)$. We do not know $E(Y_i)$ but we could estimate it with $\hat{y}_i$. However we cannot obtain $\hat{y}_i$ without the weights, and we cannot obtain the weights without $\hat{y}_i$. A solution to this problem is to use an algorithm known as *iteratively weighted least squares.*[3] This algorithm is useful whenever the variance (and thus the weights) are assumed to be a function of $E(Y_i)$. The algorithm can be described as follows.

Step 1. Estimate the parameters using nonlinear regression without weights.[4]

Step 2. Compute approximate weights by using $\hat{y}_i$ as an estimate of $E(Y_i)$.

Step 3. Estimate the parameters using nonlinear regression with *weighted* least squares, using the weights computed in the previous step.

Step 4. Repeat the previous *two* steps several times, so that the parameter estimates are used to recompute $\hat{y}_i$, and thus the weights, and the recomputed weights are then used to re-estimate the parameters, and thus $\hat{y}_i$.

Use the model and algorithm described above and the elephant mating data from the `case2201` data frame in the **Sleuth3** package (with Y_i as the number of matings and x_i as age) to answer the following questions.

1. Estimate β_0 and β_1 using nonlinear regression but with no weights. To get starting values use the following strategy. If we assume that $Y_i \approx e^{\beta_0} e^{\beta_1 x_i}$ then $\log(Y_i) \approx \beta_0 + \beta_1 x_i$, so estimating a *linear* model with the log of Y_i as the response variable provides crude estimates of β_0 and β_1 that can be used as starting values. Report the output of `summary` for this model.

2. The previous problem is the first step of the iteratively weighted least squares algorithm. Program the rest of the algorithm. Recall that `predict` can be used to compute $\hat{y}_i$, and the

[3]This is actually a variation on the algorithm that most software for generalized linear models (including the `glm` function in R) uses for estimating the parameters of a generalized linear model (including Poisson regression). But it is also a useful technique to use is some other models for dealing with heteroscedasticity via weighted least squares whenever the variance is assumed to be related to $E(Y_i)$.

[4]Whenever no weights are specified when using `lm` or `nls` all weights are set to one (i.e., all $w_i = 1$). Least squares with such weights is sometimes called *unweighted* or *ordinary* least squares.

`weights` argument of `nls` can be used to specify weights. There are a couple of ways you can program the repeating steps. One is to simply cut-and-paste the R code for the second and third steps several times. Alternatively you can use a *loop* if you are comfortable with that level of R programming. Also you can use the same starting values as you obtained in the previous problem each time you estimate β_0 and β_1, but if you are so inclined to do a little more programming a more efficient approach is to use the estimates from the last iteration starting values. Report the results of `summary` for the last iteration of the algorithm. If you do it correctly your *point estimates* should match those obtained using `glm` as shown below.[5]

```r
m <- glm(Matings ~ Age, family = poisson, data = case2201)
summary(m)$coefficients
```

```
               Estimate Std. Error   z value     Pr(>|z|)
(Intercept) -1.58200796 0.54462132 -2.904785 3.675052e-03
Age          0.06869281 0.01374578  4.997375 5.811590e-07
```

Repeat the steps of the iteratively weighted least squares algorithm enough times to that the point estimates match those given above to the fifth decimal place. Report the output of `summary` of your final step.

[5]Standard errors and quantities derived from standard errors (e.g., confidence intervals and test statistics) will not necessarily agree between maximum likelihood and iteratively weighted least squares. The reason for this is that iteratively weighted least squares assumes only that the variance is *proportional* to $E(Y_i)$ whereas maximum likelihood (assuming a Poisson distribution for Y_i) makes a stronger assumption that the variance of Y_i is *equal* to $E(Y_i)$. The approach used here is more closely related to using a *quasi*-likelihood which we will discuss later.

Linear Models by Nonlinear Regression

In principle, any *linear* model can be estimated using software for *nonlinear* regression because a linear model is simply a special case. Software for linear regression often includes convenient shortcuts for creating things like indicator variables and iteractions, but this obscures the underlying mathematical form of the model unless one has a good grasp of how the software specifies the model mathematically. Software for nonlinear regression requires one to specify the actual mathematical form of the model that relates the expected response to the explanatory variables. It is (in my opinion) a useful exercise to learn how to specify models without shortcuts in functions like `lm` by specifying the model "manually" instead.

Use the `nls` function to estimate the parameters of each of the six models for the anorexia data from the previous homework assignment. For each model report the results of `summary` to verify that they match those that I gave in that assignment using the `lm` function. Note that for each model there are many ways that you could specify the model. For example, you could create indicator variables using the `ifelse` function and add them to the data frame, or create indicator variables within the model formula itself.[6]

[6]For examples see the solutions to the in-class example with the Michaelis-Menten model, or the example of estimating the linear model for the insulation data with `nls` from lecture. I would **not** recommend trying to use `ifelse` within the model formula itself. It is useful when there are only two categories of an explanatory variable, but with with three or more categories it can lead to some unwieldy code if you try to use nested `ifelse` statements — i.e., it can be made to work but the code will be messy and harder to debug.