Lab 12 - Non-linear models

ENVX1002 Handbook

Semester 1, 2025

> 💡 **Learning outcomes**
>
> - Calculate "by hand" the initial estimates of the parameters of a non-linear model
> - Interpret tables of regression coefficients for polynomials to perform hypothesis testing
> - Fit polynomials and non-linear models to data using least-squares fitting using the SOLVER add-in in Excel
> - Fit polynomials and non-linear models to data in R, and interpret the outputs

Before we begin

Create your Quarto document and save it as `Lab-12.qmd` or similar.

The following data files are required:

1) east_creek.csv

Over the past few weeks you have explored linear models and how to interpret model summary output. Again we have stepped up the complexity, now venturing into the world of non-linear models.

This practical focuses on fitting non-linear models to data with an emphasis on 3 important classes of functions that all budding biologists and environmental scientists should know

- polynomials,
- exponential models, and
- logistic models.

A question before we begin:

What are some advantages and disadvantages of non-linear models as compared to polynomials?

Solution

Polynomials	Non-linear models(exponential/logistic)
Parameters hard to interpret	Parameters interpretable in terms of biological and environmental processes
Can be fitted with analytical or numerical methods	Fitting parameters can be difficult as use numerical methods which rely on good initial estimates of parameters
Hypothesis testing and estimating R^2 straightforward	Hypothesis testing and estimating R^2 is a bit more difficult – not absence of R^2 in R output
Difficult to model horizontal asymptotes	Can accommodate horizontal asymptotes
Only polynomials of even order restricted to positive values	Constrained to positive values

---

Polynomials

Quadratic

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

where the parameters are the y-intercept (b_0), the linear component (b_1) and the quadratic component (b_2).

If b_2 is negative then the shape of the function is convex upwards, i.e. y increases with x until reaches a peak and then y decreases.

It is easy to understand so it has been commonly used for modelling the response of yield to inputs such as fertiliser, seeding rates. This is despite much criticism for being unrealistic.

Limitations:

 (i) rate of increase to peak is same as rate of decrease past peak

(ii) does not level off as x becomes small or very large, y just keeps increasing or decreasing.

Cubic

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

Compared to the quadratic model which has 1 turning point, a cubic model has 2 turning points.

Exercise 1: Interpreting polynomials

A study was performed to examine the soil properties that control the within-field variation in crop yield. The focus of this question is on soil pH which (among other things) controls the availability of nutrients to plants.

This exercise does not require you to read in any data, but rather focus on interpreting the model output and comparing the models.

The figure below shows the raw observations of yield plotted against pH with three models fitted; a linear model, quadratic polynomial and a cubic polynomial.

a) which line corresponds to which model?
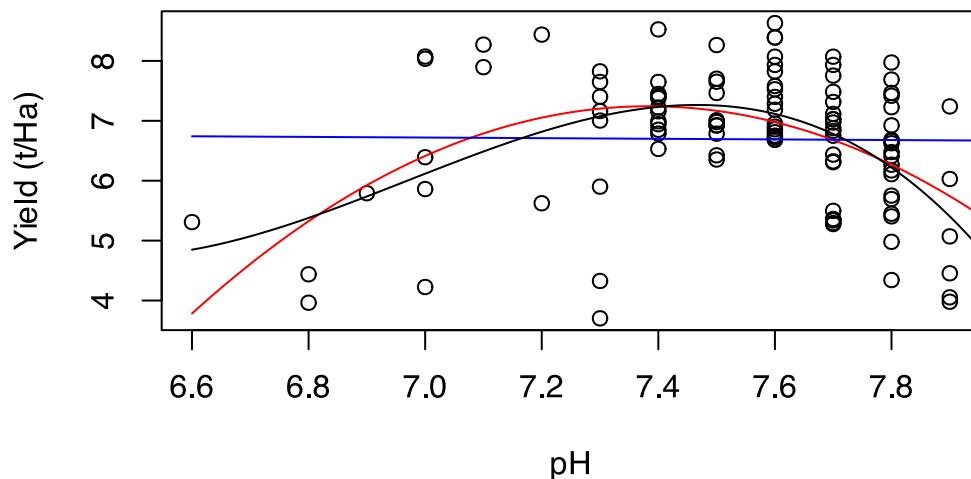
Solution

a)

- The blue line is the linear model – it is straight.

Note: If the quadratic or higher order terms are very small for some values of the predictor the line could also appear to be straight.

- The red line is the quadratic, it has one turning point (~pH = 7.4) where the slope of the line = 0.

- The black line is the cubic, it has two turning points, one is around pH ~ 7.5, the other is not on the range of the plot but you can see that the curve is changing direction and bending up at a pH of ~ 6.8, as it transitions from concave down to concave up as values get smaller. This is useful as it allows for a gentler increase in the slope of the fitted line at smaller values of pH.

b) based on the output from the 3 models below, which model fits the data best? Note: no hypothesis testing yet, just how well the model fits the data (R^2).

1. Linear model:

```
Call:
lm(formula = yield ~ ph, data = soil)

Residuals:
    Min      1Q  Median      3Q     Max
-3.0027 -0.5540  0.2189  0.7643  1.9424

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.08842    3.05279   2.322   0.0221 *
ph          -0.05249    0.40438  -0.130   0.8970
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.137 on 108 degrees of freedom
Multiple R-squared:  0.000156,  Adjusted R-squared:  -0.009102
F-statistic: 0.01685 on 1 and 108 DF,  p-value: 0.897
```

2. Quadratic model:

```
Call:
lm(formula = yield ~ ph + I(ph^2), data = soil)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5010 -0.4380 -0.0099  0.5871  1.7064

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -299.9031    54.2430  -5.529 2.30e-07 ***
ph            83.1948    14.6954   5.661 1.27e-07 ***
I(ph^2)       -5.6336     0.9942  -5.667 1.24e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.002 on 107 degrees of freedom
Multiple R-squared:  0.2309,    Adjusted R-squared:  0.2166
F-statistic: 16.07 on 2 and 107 DF,  p-value: 7.921e-07
```

3. Cubic model:

```
Call:
lm(formula = yield ~ ph + I(ph^2) + I(ph^3), data = soil)

Residuals:
    Min      1Q  Median      3Q     Max
-3.3734 -0.4540  0.0526  0.5171  1.9561

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1719.020    1082.451   1.588   0.1152
ph          -745.066     443.767  -1.679   0.0961 .
I(ph^2)      107.461      60.570   1.774   0.0789 .
I(ph^3)       -5.140       2.752  -1.867   0.0646 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9908 on 106 degrees of freedom
Multiple R-squared:  0.2554,     Adjusted R-squared:  0.2344
F-statistic: 12.12 on 3 and 106 DF,  p-value: 6.971e-07
```

Solution

---

b) The R^2 assesses how well the model fits the data. The best model based on this metric is the cubic polynomial (R^2 = 0.255), then the quadratic polynomial (R^2 = 0.231) and finally the linear model (R^2 = 0.00016).

It should be noted that this does not consider the model complexity (i.e. aiming for the most parsimonious model - the one with least predictors). For example, is the extra complexity of the cubic polynomial worth an increase in R^2 of 0.024? Formal hypothesis testing can address this question.

---

c) Use the R output to perform hypothesis testing to find the best model. Write out the hypotheses you are testing.

Solution

---

c) The general approach is to start with the most complex model and work backwards to the simplest model.

First we examine the cubic polynomial and more specifically the cubic term (the most complex term) where the hypotheses we are testing are:

H_0: $\beta_3 = 0$,

H_1: $\beta_3 \neq 0$

The line of R output that addresses this is the 4th line of the coefficients table (i.e. highest order):

```
            Estimate  Std. Error   t value  Pr(>|t|)
(Intercept) 1719.019609 1082.450919  1.588081 0.11524654
ph          -745.065927  443.766859 -1.678958 0.09610580
I(ph^2)      107.461212   60.569624  1.774177 0.07890512
I(ph^3)       -5.140059    2.752473 -1.867433 0.06460381
```

The P-value is > 0.05 so we fail to reject the null hypothesis. This means we should remove the cubic term.

Note that the estimates of the quadratic and linear terms are conditional on the cubic term also being in the model so will change if we fit a quadratic polynomial (see the R output).

Next we consider the quadratic polynomial model:

H_0: $\beta_2 = 0$,

H_1: $\beta_2 \neq 0$

The line of R output that addresses this is the 3rd line of the coefficients table (i.e. highest order):

```
            Estimate Std. Error   t value     Pr(>|t|)
(Intercept) -299.903051 54.2429517 -5.528885 2.296782e-07
ph           83.194759 14.6953503  5.661298 1.272952e-07
I(ph^2)      -5.633609  0.9941893 -5.666536 1.243405e-07
```

The P-value is < 0.05 so we reject the null hypothesis and would keep the quadratic term in the model. The best model to use is the quadratic polynomial.

Note that the linear term is also significant but even if it were not, we would keep it in the model as in most situations it is hard to justify a situation where a biological or environmental process could be modelled by a quadratic term alone.

In summary for polynomials we only test the highest order term in each polynomial in deciding whether to keep use the polynomial at all.

Exercise 2: Fitting polynomials in R

This exercise will use real data from a yield-fertiliser trial in Bedfordshire, United Kingdom.

First thing we can do is fit a linear model to the fertiliser data:

```
# create fertiliser and yield objects
fert <- c(0, 100, 170, 225)
yield <- c(3.32, 5.23, 5.41, 5.02)

# Fits a linear model and saves it to an object called lin.mod
lin.mod <- lm(yield ~ fert)
```

```
# Summarises key features of model
summary(lin.mod)
```

```
Call:
lm(formula = yield ~ fert)

Residuals:
      1       2       3       4
-0.4469  0.6727  0.2995 -0.5252

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.766929   0.634765   5.934   0.0272 *
fert        0.007904   0.004243   1.863   0.2035
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7134 on 2 degrees of freedom
Multiple R-squared:  0.6344,    Adjusted R-squared:  0.4515
F-statistic:  3.47 on 1 and 2 DF,  p-value: 0.2035
```

a) What is the model fit like in this model?

Solution

---

a) Our model fit is ok, with R^2 of 0.6344.

Fit and plot a quadratic polynomial in R. In R a quadratic polynomial can be fitted using the following lines of code:

```
# create a new variable which is the square of the fertilizer rates
fert2 <- fert^2

# fit the quadratic model incorporing fert2
quad.mod <- lm(yield ~ fert + fert2)

summary(quad.mod)
```

```
Call:
lm(formula = yield ~ fert + fert2)

Residuals:
        1         2         3         4
```

```
-0.005611  0.024528 -0.032791  0.013874

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.326e+00  4.324e-02   76.92  0.00828 **
fert         2.786e-02  9.014e-04   30.91  0.02059 *
fert2       -9.064e-05  3.921e-06  -23.12  0.02752 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0436 on 1 degrees of freedom
Multiple R-squared:  0.9993,    Adjusted R-squared:  0.998
F-statistic: 731.7 on 2 and 1 DF,  p-value: 0.02613
```

b) What is the fit like for our quadratic model? is it better than our linear model?

Solution

b) Our model fit even better, with R^2 of 0.9993.

In Excel it is easy to fit a line, by creating a scatterplot, then **add Trendline...** and selecting **Polynomial** (2nd order).

To fit our polynomial line in R, we need to obtain model predictions first.

To plot model predictions you first need to predict at fine intervals of the predictor to make a continuous plot that is not jagged or stepped. To create a new prediction dataset you can use the following code:

```
# creates a sequence of numbers from 0 to 225 going up in increments of 1
new.fert <- seq(0, 225, 1)
```

We can use our model to predict at the values in the new prediction dataset, in this case `new.fert`.

```
new.pred <- predict(quad.mod, list(fert = new.fert, fert2 = new.fert^2))
```

The general form of the `predict` function is `predict(model object, list object)`.

The list object tells R what object contains the data we will use to predict. For example in our case the model was built on fert and fert2 so we have to tell the predict function what object contains the new values for each of these, in our case new.fert.

Now we plot the raw observation as points and add an overlay of the model fit as lines:

```
plot(fert, yield, xlab = "Fertilizer", ylab = "Yield")
lines(new.fert, new.pred) # Adds lines to original plot
```
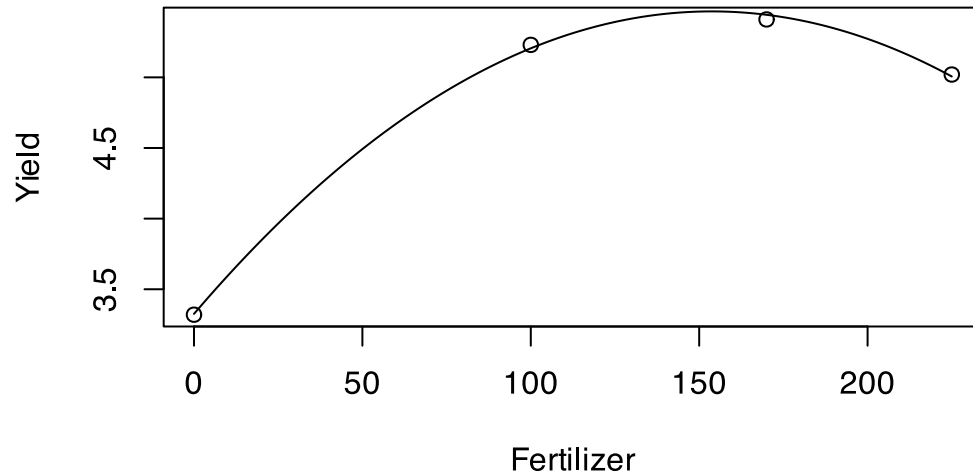
Exponential function

$y = y_0 e^{kx}$

where the parameters are y_0 which is the multiplier which expresses the starting or final value, and k which is negative for exponential decay and positive for the exponential growth.

The half life (for decay) or doubling time (for growth) can be calculated as

$\frac{log_e 2}{k}$

Limitations:

(i) harder to fit than polynomials

(ii) exponential growth has no horizontal asymptote; keeps going up.

Exercise 3: Initial estimates for exponential growth function

In this exercise we will find initial estimates of the parameters of an exponential growth model by visual assessment of plots of the data and/or rough calculations. The initial estimates of the parameters are needed as starting points for the iterative fitting methods we will use in the practicals, e.g. SOLVER in Excel and the `nls()` function in R.
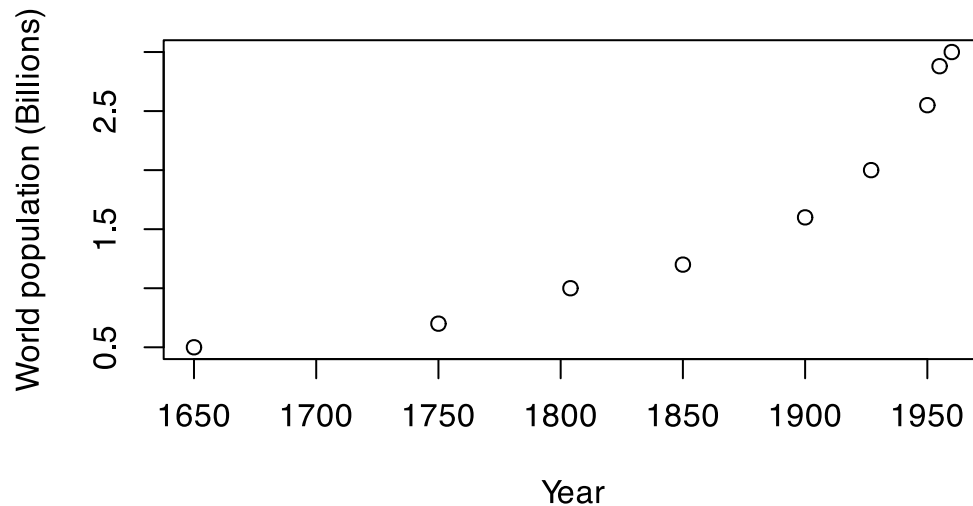
The plot and table below presents the population of the world from 1650-1965.

We wish to model the data with an exponential growth function of the form;

$y = y_0 e^{kx}$

where

- y is the population in the year x,

- y_0 is the population in 1650 and

- k is the rate constant.

Year	1650	1750	1804	1850	1900	1927	1950	1955	1960
Pop (billions)	0.5	0.7	1	1.2	1.6	2	2.55	2.88	3

a) Provide an initial estimate of y_0.

Solution

---

a) Use the first measurement, in 1650 the world population was 0.5 billion.

The parameter k can be estimated from a linear model fitted to $\log_e$ population against year.

Rather than formally fitting a linear model you could estimate the slope approximately by using the smallest and largest value to estimate the slope and therefore k.

b) Use this approach to estimate k.

Hint:

$$slope = k = \frac{\log_e y_{max} - \log_e y_{min}}{x_{max} - x_{min}}$$

Solution

10

b) The smallest value is 0.5 in 1650 and the largest value is 3 billion in 1960.

We log_e the population values and calculate the slope of the line between the 2 observations. The slope is an estimate of k:

$$slope = k = \frac{log_e3 - log_e0.5}{1960 - 1650} = \frac{1.792}{310} = 0.00577987$$

c) For an exponential growth model the doubling time of a population can be estimated by log_e2 / k.

Examine the graph and/or table to estimate the doubling time and use this to estimate k. You will have to make k the subject in the equation for estimating the doubling time.

Solution

c) The smallest value is 0.5 in 1650 and the year it has doubled (1 billion) is in 1804.

Using the difference between the two years, we can solve for k :

1804-1650 = 154

$154 = \frac{log_e2}{k}$; multiply both sides by k

$154k = log_e2$; divide both sides by 154

$k = \frac{log_e2}{154} = 0.004501$

Alternatively, you can also use the slope calculation to obtain k for the doubling time:

$$slope = k = \frac{log_e1 - log_e0.5}{1804 - 1650} = \frac{0.693}{154} = 0.004501$$

d) How similar were the estimates of k?

Solution

d) Both estimates are of a similar magnitude. It should be noted that the estimate of the slope (b1) is an approximation. If we formally estimate the slope (`lm(log(pop)~year`) the value is 0.005835.

Exercise 4 : Exponential growth models

This data is from Jenkins & Adams (2010) who studied soil respiration rates against temperature for different vegetation communities in the Snowy Mountains. They fitted an exponential growth model to the data.

The purpose of this exercise is to illustrate the dangers of using Excel's in-built functions for statistics more complex than calculating means and fitting simple models.

Plot the data in Excel and using the **Add Trendline...** option. Make sure tick the option for displaying the equation in the graph.

a) The researchers performed the experiment up to a temperature of 40 degrees C, would you expect exponential growth in the respiration rate to continue if high temperatures were considered? Is there a better model?

Solution

a) Eventually it would get too hot and the microbes which cause soil respiration would start to "die" or the community structure would change.

Some sort of horizontal asymptote could be reached, which could be modelled by a logistic model, or a decline is needed after a peak; there are non-linear models that accommodate such data that follows such a pattern.

Now fit the same model in R using the nls function. Code to get you started is:

```
temp<-c(5,10,20,25,30,35,40)
respiration<-c(1,2,4,6,8,11,18)

##Initial parameters
exp.mod<-c(y0=1.0,k=0.1)

##Fits exponential model
res.exp<-nls(respiration ~ y0 * exp(k*temp), start=exp.mod,trace=T)
```

```
2025.224    (3.23e+01): par = (1 0.1)
90.64319    (7.34e+00): par = (0.6559559 0.09162898)
2.841421    (9.49e-01): par = (0.7302258 0.08091595)
1.488431    (5.63e-02): par = (0.7944381 0.07731867)
1.483685    (1.52e-03): par = (0.8033809 0.07707711)
1.483681    (4.12e-05): par = (0.8036903 0.07706805)
1.483681    (1.68e-06): par = (0.8037011 0.07706768)
```

```
##Summarise model
summary(res.exp)
```

```
Formula: respiration ~ y0 * exp(k * temp)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
y0 0.803701   0.118632   6.775  0.00107 **
k  0.077068   0.004048  19.037 7.37e-06 ***
```

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

Residual standard error: 0.5447 on 5 degrees of freedom

Number of iterations to convergence: 6
Achieved convergence tolerance: 1.681e-06
```

b) Now you can fit a line to the plot. Does this look similar to your trendline in Excel?
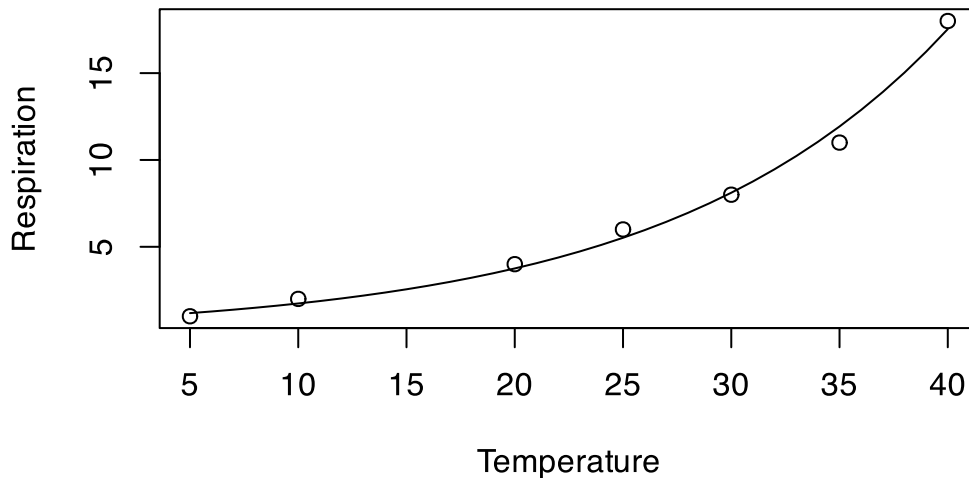
```
#Plots raw data
plot(temp,respiration,xlab='Temperature',ylab='Respiration')

#Creates new dataset for predictions ( 5 to 40 at an interval of 1)
temp.new<-seq(5,40,1)

#Makes predictions onto temp.new
pred.exp<-predict(res.exp,list(temp=temp.new))

#Adds model fit to existing plots
lines (temp.new,pred.exp)
```


Solution

b) Yes, they do look similar.

Compare the parameters values between Excel and R. You can extract the RSS value from an `nls` object by using the code below:

```
deviance(res.exp)
```

```
[1] 1.483681
```

c) Calculate the RSS value for the Excel exponential model. Based on this, which is the better model?

Solution

c) The better model is the one fitted in R which is optimized to fit the actual observations rather than the approximate linearisation approach used by Excel.

When faced with the need to fit an exponential function, one approach that was used before computing power became readily accessible was to log the y values which linearises the relationship with x, enabling the modeller to use a simple linear model.

If we linearise, model would be $log_e(y) = b_0 + b_1 x$, where e^{b_0} is the y_o parameter in an exponential model, and b_1 is the k parameter in the exponential model. This is similar to what was demonstrated in the lecture this week.

In Excel, log the soil respiration data and fit a linear model. You will see that the fitted model gives the same values as the exponential model fitted to the untransformed data.

If you compare the R² values for both you will see they are the same. This means that Excel reports the R² of the linear model fitted to logged respiration as the R² of the exponential model fitted to the raw data. This is naughty of Excel.

For the dataset used here the exponential model fits it so well the Excel approach is only slightly different to the correct approach used in R.

In cases where the model does not fit the data so well the differences would be larger. Logarithm makes smaller values larger and larger values smaller.

WHY IS THIS SUB-OPTIMAL?

- Regression modelling assumes that the residuals are normally distributed so logging normally distributed data will change the distribution to a non-normal one – it is best to analyse the data without transformation.

- Modelling data on the logged scale reduces the impact that larger values have on minimising the RSS but when you plot the fitted model with the original data you may observe large discrepancies for larger values. In other words, using a linear model on the log(data) can result in a higher discrepancy for larger values when plotting the fitted model on the original data. Therefore, the model fitted to the logged data is not necessarily the best on the original data.

- The reporting of the R-squared on the logged data on a model purported to be fitted to untransformed data is just wrong as the R-squared on the log scale will be better as the variation in the data has been reduced but we really want to know how well an exponential model fits the raw data.

Logistic function

There are several versions of the logistic function. We will use the following:

$$y = \frac{Asym}{1 + e^{\frac{xmid-x}{scal}}}$$

where

- $Asym$ is the maximum value of y (upper limit, horizontal asymptote).
- $xmid$ is the value of x when y is halfway between the lower and upper limits (inflection point, $y = 0.5 \times Asym$).
- $scal$ is the rate of change: the rate at which y approaches the upper limit.

Commonly used to model growth that has a sigmoid shape, i.e. where growth is initially slow, then picks up to a maximum, then slows down again as the system reaches a maximum.

Limitations:

Harder to fit than polynomials.

Exercise 5: Logistic models

In this exercise you will model the yield of pasture over time (since sowing).

Note, we will assume the yield at sowing = 0 is 0 t/ha, which allows us to use the equation above (that is pre-defined in `SSlogis()`).

If this were not the case, we would need to use a slightly different equation; $y = y_0 + \frac{Asym}{1+e^{-scal(x-xmid)}}$, where y_0 is the yield at sowing.

a) Fit the model in R using the `nls` function. Estimate the starting parameters manually (using the previous exercise and the equation above as a guide). Then use the `SSlogis()` function to automatically estimate the parameters (see L12, Tut12 or the handbook).

Solution

a) Use the following code to read in the code and run the model:

```
# Data input
week <- c(9,14,21,28,42,57,63,70,79)
yield <- c(8.93,10.80,18.59,22.33,39.35,56.11,61.73,64.62,67.08)

# Estimating parameters
```

```
log.mod <- c(Asym=70.0, xmid=40.0, scal=1.0)
yld.log <- nls(yield ~ Asym/(1 + exp((xmid - week)/scal)), start = log.mod,
trace=T)
```

```
1836.927    (1.43e+00): par = (70 40 1)
1124.107    (4.86e+00): par = (69.9852 25.47246 8.26585)
1115.702    (4.87e+00): par = (69.96521 25.49818 8.285329)
1099.053    (4.91e+00): par = (69.92565 25.54971 8.324271)
1066.386    (4.99e+00): par = (69.84821 25.65308 8.402085)
1003.516    (5.15e+00): par = (69.70002 25.86098 8.557394)
887.1432    (5.47e+00): par = (69.42984 26.28115 8.866419)
688.2579    (6.13e+00): par = (68.98988 27.13636 9.475723)
400.9711    (6.89e+00): par = (68.47614 28.88513 10.64292)
116.6069    (4.13e+00): par = (68.626 32.37177 12.67415)
12.30649    (7.23e-01): par = (71.36289 38.10072 15.12695)
8.059833    (2.07e-02): par = (72.51167 38.91436 14.87333)
8.056533    (1.15e-03): par = (72.45789 38.86434 14.84398)
8.056523    (7.26e-05): par = (72.46251 38.86759 14.8459)
8.056523    (4.57e-06): par = (72.46222 38.86739 14.84577)
```

```
# Self-starting function
yld.log <- nls(yield ~ SSlogis(week, Asym, xmid, scal))
```

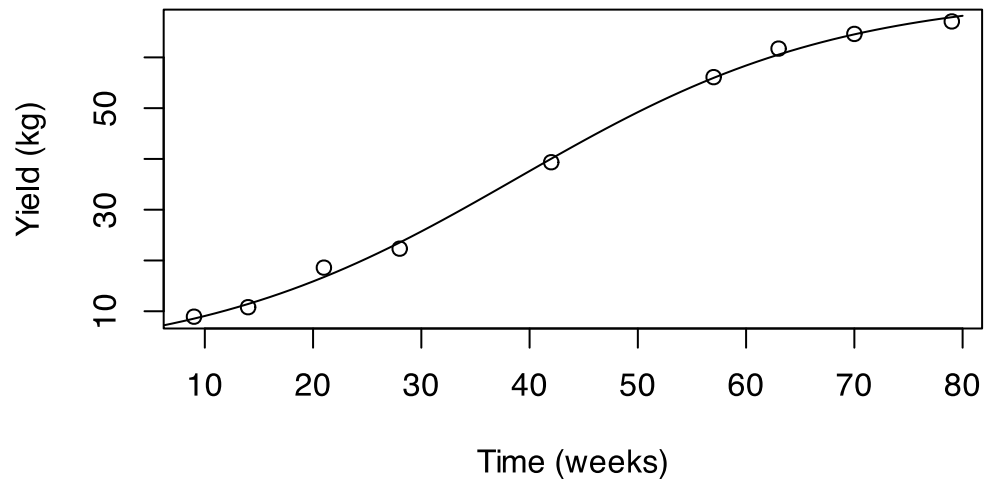b) Plot the fitted model with the observations.

Solution

b) Use the following code:

```
# Creates new dataset for predictions (5 to 40 at an interval of 1)
week.new <- seq(0,80,1)

#Makes predictions onto temp.new
pred.log <- predict(yld.log, list(week = week.new))

#Adds model fit to existing plots
plot(week, yield,
     xlab='Time (weeks)', ylab='Yield (kg)')
lines(week.new, pred.log)
```

```
## In ggplot
# ggplot(data = data.frame(week, yield), aes(x = week, y = yield)) +
#   geom_point() +
#   geom_line(y = predict(yld.log, week.new), color = "red")
```

c) Compare the final model parameters when we provide starting estimates, and when we use
SSlogis(). Are they the same?

Solution

c) From the code below we can see the estimated parameters are 72.46 for $Asym$, 38.87 for $xmid$
and 14.85 for $scal$. This should be similar to what Solver obtains in Excel.

```
#Examine model
summary(yld.log)
```

```
Formula: yield ~ SSlogis(week, Asym, xmid, scal)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
Asym  72.4622     1.7340   41.79 1.26e-08 ***
xmid  38.8674     1.1794   32.95 5.19e-08 ***
scal  14.8458     0.7596   19.54 1.16e-06 ***
```

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

Residual standard error: 1.159 on 6 degrees of freedom

Number of iterations to convergence: 1
Achieved convergence tolerance: 1.11e-06
```

STARTING VALUES

When fitting non-linear functions (i.e. logistic or exponential) using iterative procedures such as `nls` or SOLVER the starting estimates of the parameters need to be approximated. If the values are too far, the model will not run and return an error.

The best way to ensure that you have suitable starting values is to plot the data with the predictions overlaid for your starting parameters. You can then see how close your initial model is to the data.

The reason we go through the process of estimating parameters is because `nls` and SOLVER can fit *any* nonlinear equation. They are more versatile

Realistically, most nonlinear relationships you would fit are covered by a self-starting function; i.e. `SSlogis()`, `SSasymp()`, or `nlraa::SSexpf()`. These estimate the parameters for you and are more efficient to run. We recommend using these when suitable.

That's it for Module 3! Great work exploring non-linear models!

Thank you all (students and demonstrators!) for your hard work and enthusiasm throughout this Module. Good luck with Project 3 and the final exam!