Non-linear models & linear mixed-effects models

Aims of this practical

Carry out a range of non-linear and linear mixed-effects models to test relationships between sets of variables. These tests will include:

- Polynomials
- Self starting non-linear models
- Generalised additive models (GAMs)
- Linear mixed-effects models

Useful functions you will use…

lm – for polynomials

nls – non-linear least squares models

SSasymp - non-linear self-starting asymptotic

SSmicmen - non-linear self-starting Michaelis-Menten model

SSlogis - non-linear self-starting logistic model

gam – for generalised additive models

lmer – linear mixed-effects models

AIC - Akaike information criterion

xyplot – plotting in lattice package

Good starting point for further reading:

Crawley, M.J. 2007. The R Book. Wiley.

Non-linear models

Set up your script like previous weeks with the code you can copy from previous weeks (date, set working directory, .libPaths etc). The first example we will look at is a dataset containing plant trait variability along an environmental stress gradient. The dataset contains the response variable stem flexibility (*flex.index*) measured from the saltmarsh plant *Atriplex portulacoides*. Coastal vegetation plays an important role in protecting coastlines from wave action and it is assumed that the stronger and less flexible a plant is the more wave energy will be dissipated. Stem flexibility was measured from each *Atriplex* individual along a transect running down the saltmarsh elevation gradient.

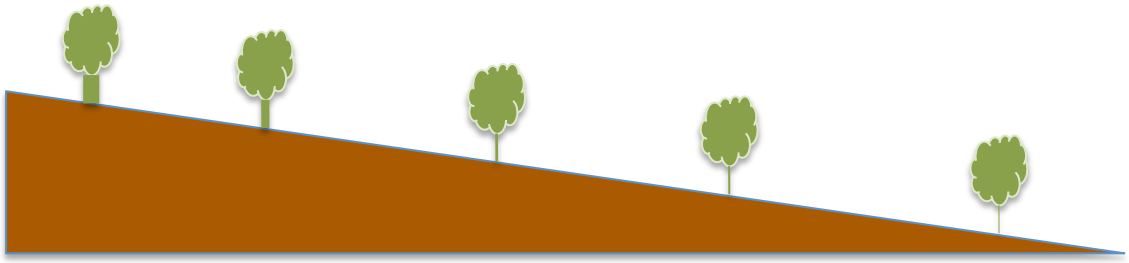

Fig 1. Schematic showing the anticipated stem trait response to increased elevation.

Now set up your R-script and load the dataset '*flex_data.csv*' from Moodle. Plot a scatter graph, remember to put the response variable on the y-axis.

The data appear to be non-linear, plot a linear regression line through these data and see if you think this looks like an adequate fit? We will first use polynomial regression (not a true non-linear model) to fit this data, see the code below.

```
### Polynomial regression techniques
fit1 <- lm(flex.index ~ elevation, data=flex)  # linear model
fit2 <- lm(flex.index ~ elevation + I(elevation^2), data=flex) # 2nd degree polynomial
fit3 <- lm(flex.index ~ elevation + I(elevation^2) + I(elevation^3), data=flex) # 3rd degree polynomial
```

Three terms are entered into the model as predictors: elevation, elevation squared, and elevation cubed. The squared and cubed terms MUST be entered using "I", which in a model formula means "as is." This is because the ^ symbol inside a formula does not have its usual mathematical meaning. If you want it to be interpreted mathematically, which we do here, it must be protected by "I", as in the term "I(elevation^2)". This gives the same result as if we had created a new variable, elevation.squared= elevation ^2, and entered elevation.squared into the regression formula.

We have used three different models and wish to examine the fit of each, the first is a simple linear model, the next is a second degree polynomial and the final one is a third

degree polynomial. Using the summary function check each model output and see if all the model terms are significant? Also, note the R-squared from each model output, notice the increase in each successive model. If the I(elevation^3) term in the third degree polynomial was not significant then that would be a good indicator that the second degree polynomial model was a better fit for this data. However, things are all significant here so we can either compare R-squared values or test the model improvement with the F-statistic by using **anova(fit2, fit3)** which gives the following R output:

```
Analysis of Variance Table

Model 1: flex.index ~ elevation + I(elevation^2)
Model 2: flex.index ~ elevation + I(elevation^2) + I(elevation^3)
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     47 45.946
2     46 35.412  1    10.534 13.684 0.0005759 ***
```

Looking at the P-value tells us that fit3 is a significant improvement on the fit2 model.

Now lets plot our model using the **predict** function.

```
plot(flex.index~elevation, data=flex)
points(flex$elevation, predict(fit3), type="l", col="red", lwd=2)
```

This looks much better than a linear fitted line but has an odd tail that may cause problems if we wanted to extrapolate values past the ones we have measured.

We can use the same model validation techniques for polynomials as we have in previous linear modeling exercises. However, for this taster in non-linear modeling we will move on to look at some different modeling methods.

The flex data looks like a typical asymptotic relationship (look this up if you don't know what that means) so the next model we could try is an **asymptotic**. True non-linear modeling requires you to input some starting values for you model; parameters such are the asymptote and the steepest slope values (the a,b,c in the model below). The models work by iteratively changing these values to minimize the sum of squared residuals or maximize the log-likelihood. However, R has a range of self-starting models, which as the names suggests, do not require you to input these values. Lets look at one for the flex data.

```
### Self-starting asymptotic model
SSfit1<-nls(flex.index~SSasymp(elevation,a,b,c), data=flex) # NO parameters to estimate
plot(flex.index~elevation, data=flex)
points(flex$elevation, predict(SSfit1), type="l", col="red", lwd=2)
```

This fitted line looks much better than our previous polynomial attempts. However, we shouldn't always disregard polynomials as these equations are generally much easier to work with than the true non-linear models.

Lets look briefly at the code, 'nls' stands for non-linear least squares and this function will cover most of the non-linear models we use. SSaymp states we want to do a self-starting asymptotic model with our explanatory variable *elevation*. The a,b,c are parameters to start the model, for the self-starting models we are using we don't need to give any values for these. If you carry out a **summary** of this model you get the following output:

```
Formula: flex.index ~ SSasymp(elevation, a, b, c)

Parameters:
  Estimate Std. Error t value Pr(>|t|)
a  10.6405     0.1818  58.530   <2e-16 ***
b   1.7856     0.7137   2.502   0.0159 *
c   0.2703     0.1329   2.034   0.0476 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8146 on 47 degrees of freedom

Number of iterations to convergence: 0
Achieved convergence tolerance: 2.121e-06
```

The key parts here are the a,b and c parameter estimates, these are the bits you can extract and use in the model equation. Try looking up the help file **?SSaymp** and see if you can work out how you might use these?

Model validation plots need to be created manually for the 'nls' function but you can use the code below to generate these.

```
## residuals vs fitted
plot(resid(SSfit1)  ~ fitted(SSfit1), main="Residuals vs fitted values", xlab="Fitted values",
ylab="Residuals")
abline(h=0, lty=2) # add horizontal line

## histogram of residuals
hist(resid(SSfit1), main="Histogram of residuals", xlab="Residuals")

## normal q-q plot
qqnorm(resid(SSfit1), main="Normal Q-Q plot") # qqplot step 1
qqline(resid(SSfit1)) # qqplot step 2
```

The residuals vs fitted plot doesn't look great, we may need to include covariates, collect more data or indeed change the model to improve this.

We will leave this data set now and explore two other self-starting models, Michaelis-Menten and the logistic model. Michaelis-Menten describes a reaction rate increasing quickly at first but asymptotes once the reaction rate is no longer limited (similar to the SSaymp model), whereas the logistic model gives the typical sigmoidal or 'S' shape curve.

Lets try these both out with some new data on pH at different salt concentrations. The data can be entered manually with the code below.

```
concentration <- c(1,2,3,5,10,15,20,25,30,35);
rate <- c(2.8,4.2,3.5,6.3,15.7,21.3,23.7,25.1,25.8,25.9)
plot(concentration, rate, las=1, pch=16)

# Self-starting Michaelis–Menten model
SSfit2<-nls(rate~SSmicmen(concentration,a,b)) # NO parameters to estimate
summary(SSfit2)
plot(rate~concentration, xlim=c(0,40), ylim=c(0,35), las=1, pch=16)
points(concentration, predict(SSfit2), type="l", col="red", lwd=2)

# Self-starting logistic model
SSfit3<-nls(rate~SSlogis(concentration,a,b,c)) # NO parameters to estimate
summary(SSfit3)
plot(rate~concentration, xlim=c(0,40), ylim=c(0,35), las=1, pch=16)
points(concentration, predict(SSfit3), type="l", col="red", lwd=2)
```

Ok, you can check out the model validation for each but lets try something new first. To check the goodness of fit between models we can use Akaike Information Criterion or AIC. This is a commonly implemented method you will come across in papers. Unlike R-squared we have been using for linear models, AIC does not relate to the amount of variance explained by your model. AIC is not useful on it's own; it is only useful for comparing between models of the same data. The lower the AIC the better the goodness of fit. Some model outputs display the AIC but R also has a handy function **AIC()**, try this with SSfit2 and SSfit3 to see which one has the lowest AIC and hence best fit.

Generalised additive models (GAM)

Sometimes we can see that the relationship between y and x is non-linear but we don't have any theory or any mechanistic model to suggest a particular functional form (mathematical equation) to describe the relationship. In such circumstances, generalized additive models (GAM) are particularly useful because they fit non-parametric smoothers to the data without requiring us to specify any particular mathematical model to describe the non-linearity (see Chapter 18, Crawley 2007 for further information)

Use the code below to generate some fictitious data, species richness of plants over time on a heavily disturbed dune system.

```
# simulate some data
time<-seq(0,100,1)
SR<-runif(1,5,15)*exp(-runif(1,0.01,0.05)*time)+rnorm(101,0,0.5)
```

Go ahead and plot the data, try putting a linear regression line on the plot and see if you think it's a good fit?

To run the **gam** function you need to load the **mgcv** package (might also need to download the mgcv package first). Running a gam is again a very easy bit of code, see below

```
### Generalized Additive Model (GAM)
library(mgcv)
fit7<-gam(SR~s(time))
summary(fit7)
plot(fit7)
```

The plot function for GAM gives us a plot of the modeled line rather than model validation, unlike previous modeling functions. Model validation for GAMs needs to be done manually as we did above. The **summary** function looks similar to a *lm* output but has split the results; the upper part of the table contains the parametric Intercept term whereas the lower section gives the explanatory variable as a smoothing term. The model output also gives us a R-squared and Deviance values that can be used to compare models.

```
### Plotting modelled line (GAM)
newX<-data.frame("time"=seq(0,100,1))   # created x values
newY<-predict(fit7, newdata=newX) # predict values of y using the created x values
plot(SR~time)
points(newY~newX$time, type="l",lwd=2)
```

Keeping with the theme of species diversity and disturbance, load the '*disturbance.csv*' dataset from Moodle. This dataset is looking at mammal diversity across a disturbance gradient. The variables are:

diversity – diversity of mammals based on camera trap records
dist.index – disturbance index (scores below zero have very low disturbance) based on environmental measures such as vegetation cover, soil compaction etc
foot.index – an index of human influence on the site, visitor numbers, rubbish counts etc

Plot the data and describe the patterns you see. To run the gam model you can use the basic code from the last example (call the model fit8). Try modeling the relationship between diversity and disturbance index and plot the shape of the relationship (i.e. the modeled line). Also, have a look at the model output using **summary**. You can plot the modeled line along with the original data using the code below and the **predict** function.

```
newX.dist<-data.frame("dist.index"=seq(-4,1,length=43)) # created x values
newY.dist<-predict(fit8, newdata=newX.dist) # predict values of y using the created x values
par(mfrow=c(1,1))
plot(diversity~dist.index, data=disturbance)
points(newY.dist~newX.dist$dist.index, type="l",lwd=2)
```

It's also very simple to add extra explanatory variables to our gam. The script below shows you how to do this using the *foot.index* as an extra covariate.

```
fit9<-gam(diversity~s(dist.index)+s(foot.index), data=disturbance)
summary(fit9)
```

The summary now includes coefficients for both the explanatory variables; both are significant in our model. Has this improved our model compared to the single term model? Use the AIC function to compare the goodness of fit between the two models?

Using the vis.gam plotting function below we can see how our response variable (species diversity) varies with both of the explanatory variables; diversity is now a surface that reduces from top-right to bottom-left, i.e. more mammal species with low disturbance and low human influence.

```
vis.gam(fit9, type = "response", plot.type = "contour")
```

Linear mixed-effects models

Mixed models are useful for dealing with grouped data. Groups can be nested within locations, as in nested field experiments, or they could be nested within participants, such as individuals being repeatedly measured. **In previous weeks we have treated all categorical variables in linear models as if they were the same, however, this is not the case!** There are important differences that need to be recognized, but recognition is not straightforward and will take a great deal of practice before you can be certain you have specified the correct model. The different variables are called **fixed effects** and **random effects**. Nesting (or hierarchical structure) of random effects is a classic source of pseudoreplication, so it is important that you are able to recognize it and hence not fall into its trap.

Borrowing heavily from Crawly 2007 (Chapter 19) lets try to understand the difference between fixed and random effects using an example. 'In most mammal species the categorical variable sex has two levels: male and female. For any individual that you find, the knowledge that it is, say, female conveys a great deal of information about the individual, and this information draws on experience gleaned from many other individuals that were female. A female will have a whole set of attributes (associated with her being female) no matter what population that individual was drawn from. Take a different categorical variable like genotype. If we have two genotypes in a population we might label them A and B. If we take two more genotypes from a different population we might label them A and B as well. In a case like this, the label A does not convey any information at all about the genotype, other than that it is probably different from genotype B. In the case of sex, the factor level (male or female) is informative: **sex is a fixed effect**. In the case of genotype, the factor level (A or B) is uninformative: **genotype is a random effect**.' Below are some examples of fixed and random effects

Fixed effects	**Random effects**
Female or male	Genotype
Insecticide sprayed or not	Brood
Nutrient addition or not	Block within a field
Upland vs lowland	Family
Light vs shade	Individuals with repeated measures

Key point: because the random effects come from a large population, there is not much point in concentrating on estimating means of our small subset of factor levels, and no point at all in comparing individual pairs of means for different factor levels. Much better to recognise them for what they are, random samples from a much larger population, and to concentrate on their variance.

Ok, that was heavy and a lot of text, lets try an example. The data we are going to use is sheep offspring mass by age of the mother. We want to know if age (explanatory variable) is a significant predictor of the offspring mass (response variable). The offspring weight from individual sheep mothers was recorded over ages 1-5 years. Download the dataset 'soay_mass_data.csv' and call it 'offMassData'. The variables in the dataset are:

Id – Sheep identity number (1-20)
Age – age of the mother (1-5yrs)
logMass – mass of the offspring (logged, kg)

Lets explore the data before we start modeling. Plot logMass against Age and describe the relationship. Now explore the effect of sheep Id, lets use an **xyplot** from the **lattice** package.

```
## Id effect
library(lattice)
offMassData<-transform(offMassData, Id=factor(Id)) # Change Id to a factor
# make separate plot for each individual
xyplot(logMass ~ Age | factor(Id), data=offMassData, type=c("g","p","r","l"), layout=c(5,4))

# OR all on the same plot
xyplot(logMass ~ Age, groups=Id, data=offMassData, type=c("g","b"))
```

Another way to look at the sheep Id effect is by plotting a boxplot to see if offspring mass changes for any of the individual sheep mothers (see below).

```
## Id effect
boxplot(logMass ~ Id, data = offMassData, xlab = "Sheep ID", ylab = "Offspring mass (log kg)")
abline(h = mean(offMassData$logMass, na.rm = TRUE), lty = 2) # horizontal line through mean
```

Ok, do you think sheep Id will be having an effect?

Now lets run the linear mixed-effects model. For this we are going to use the **lme4** package and the **lmer** function to create a random slope model. Having Age|ID means you have a random slope model - you assume that the relationship between mass and age varies among sheep. The fixed effects part of the model [~Age] estimates the mean slope, while the Age random effect [Age|id] estimates the variance in slope among sheep. Remember, what we are interested in is still whether there is a relationship between logMass and Age but we are taking into account the effect of Id.

```
### Linear mixed-effects model
offMassMod1 <- lmer(logMass ~ Age + (1 + Age | Id), data=offMassData)
```

Lets check the model validation plots (again we need to create these manually).

```
# Model validation
## residuals vs fitted
plot(resid(offMassMod1)  ~ fitted(offMassMod1), ylim=c(-1,1)/2,
    main="Residuals vs fitted values", xlab="Fitted values", ylab="Residuals")
abline(h=0, lty=2) # add horizontal line

## histogram of residuals
hist(resid(offMassMod1), xlim=c(-1,1)*0.6,
    main="Histogram of residuals", xlab="Residuals")

## normal q-q plot
qqnorm(resid(offMassMod1), main="Normal Q-Q plot") # qqplot step 1
qqline(resid(offMassMod1)) # qqplot step 2
```

They all look fine so we now just need to look at the output of the model. In a normal *lm*, R compares your t-statistic against your df to get a P-value. But in a mixed-effects model, it is unclear how many df you have. E.g. you have 100 observations of 20 groups, but as observations are not independent, it is difficult to translate that into the number of df you have. However, we can compare our model with a Null Model and then use the F-statistic to test.

```
## Is the fixed effect of age significant?
offMassMod1 <- lmer(logMass ~ 1 + Age + (1 + Age | Id), offMassData, REML=FALSE)
offMassMod2 <- lmer(logMass ~ 1 +          (1 + Age | Id), offMassData, REML=FALSE)
anova(offMassMod1, offMassMod2) # yes
```

The output should look like this:

```
Data: offMassData
Models:
offMassMod2: logMass ~ 1 + (1 + Age | Id)
offMassMod1: logMass ~ 1 + Age + (1 + Age | Id)
            Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
offMassMod2  5 71.728 84.754 -30.864   61.728
offMassMod1  6 56.619 72.250 -22.309   44.619 17.109      1  3.529e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

For your results you would need to quote the Chisq value, DF and P-value, e.g. Age is a significant predictor of offspring mass (Chi=17.109, df=1, P<0.001).

Ok great, you've done your first linear mixed-effects model! Lets try it on another dataset. This time we are looking at the white water lily (*Nymphaea alba*) and the relationship between the size of two parts of the plant, the petiole (stalk that attaches a leaf to a stem) diameter and midline distance (petiole junction to pad tip). Measurements were taken in different lakes, some having beavers present and other where beavers are absent. Beavers consume lilies and hence can impact the plants growth form.

Load the '*lilies.csv*' dataset from Moodle. The variables are:
Site – individual lake id
Mid_line_Diameter - petiole junction to pad tip (mm)
Petiole_Diameter – diameter of the petiole stalk (mm)
Beaver – presence or absence of beavers

Plot the response variable (*Petiole_Diameter*) against the explanatory variable (*Mid_line_Diameter*), does there seem to be a relationship? If so what type of relationship? Now lets explore the Site variables for any random effects, try the boxplot and xyplot (lattice package) as in the earlier example.

```
#Convert Site into a factor
Lilies$fSite <- factor(Lilies$Site)
str(Lilies)
table(Lilies$fSite) # observations per cluster

#Site effect
boxplot(Mid_line_Diameter ~ fSite, data = Lilies, varwidth = TRUE,
        xlab = "Site", ylab = "Mid-line Diameter")
abline(h = mean(Lilies$Mid_line_Diameter, na.rm = TRUE), lty = 2)

# Make separate plot for each lake
xyplot(Mid_line_Diameter ~ Petiole_Diameter | factor(fSite), data=Lilies, type=c("g","p","r","l"),
layout=c(5,4))
```

Do you think Site could be a significant factor? Looking at the boxplot the variability across Sites suggests we should try to include this nested variable. Ok, lets model Mid_line_Diameter as a function of Petiole_Diameter and add Site as a random effect.

```
### Linear mixed-effects model
LM1 <- lmer(Mid_line_Diameter ~ Petiole_Diameter + (1 | fSite), data = Lilies)
```

Here we have specified a random intercepts model where you assume that Mid_line_Diameter varies among sites, but also that the relationship between Mid_line_Diameter and Petiole_Diameter is the same for all Sites, i.e. no difference in slope. If you look at the xyplot of this data that is a pretty fair assumption.

Explore the model validation plots as above. Do these look acceptable for this data?

Now we need to report our results, again we can use a Null Model to test the effect of our explanatory variable. Write a results statement using the output of the code below.

```
## Is the fixed effect of Petiole_Diameter significant?
LM1 <- lmer(Mid_line_Diameter ~ Petiole_Diameter + (1 | fSite), data = Lilies, REML=FALSE)
LM2 <- lmer(Mid_line_Diameter ~      1         + (1 | fSite), data = Lilies, REML=FALSE)
anova(LM1, LM2)
```