MATH 3050 – Predictive Analytics

UNC CHARLOTTE

Topic 4: Statistical Modeling

- ❑ Framing Modeling Question
- ❑ Deciding Response Variable
- ❑ Selecting Explanatory Variables
- ❑ Selecting Appropriate Model
- ❑ Validating the Model

UNC CHARLOTTE 1

1

General Guidelines for Model Selection ★★★★★

The explanatory variables

(a) All explanatory variables continuous	**Regression**
(b) All explanatory variables categorical	**Analysis of Variance** (**ANOVA**)
(c) Explanatory variables both continuous and categorical	**Analysis of Covariance** (**ANCOVA**)

The response variable

(a) Continuous	**Normal regression, ANOVA or ANCOVA**
(b) Proportion	**Logistic regression**
(c) Count	**Log-linear models**
(d) Binary	**Binary logistic analysis**
(e) Time at death	**Survival analysis**

UNC CHARLOTTE 2

2

Best Model Attributes ★★★★★

1. Identify the minimally adequate model to describe the data
2. Produces the least unexplained variation – the minimal residual deviance
3. Parameter estimates that are statistically significant
4. Recognition that any given model may not be suited to all problems
5. Is developed on a representative, stable, accurate, reliable, and unbiased data set

Data Analysis Considerations ★★★★★

1. Do all of the values of each variable appear in the same column?
2. Are all the zeros really 0, or should they be NA?
3. Does every row contain the same number of entries?
4. Validate the data to eliminate mistakes
5. Plot every one of the variables on its own to check for gross errors
6. Look at the relationships between variables

Data Analysis Considerations ★★★★★

7. Think about model choice
   a) Which explanatory variables should be included?
   b) What transformation of the response is most appropriate?
   c) Which interactions should be included?
   d) Which non-linear terms should be included?
   e) Is there pseudoreplication, and if so, how should it be dealt with?
   f) Should the explanatory variables be transformed?
8. Fit a maximal model and simplify it by stepwise deletion
9. Check the minimal adequate model for constancy of variance and normality of errors using plot (model)
10. Emphasize the *effect sizes* and standard errors (summary.lm), and play down the analysis of deviance table (summary.aov)
11. Document carefully what you have done and explain all the steps you took.

Maximum likelihood

What, exactly, do we mean when we say that the parameter values should afford the 'best fit of the model to the data'?

The convention we adopt is that our techniques should lead to **unbiased, variance-minimizing estimators**.

Maximum likelihood ★★★★★

We define ‘best’ in terms of **maximum likelihood**. This notion may be unfamiliar, so it is worth investing some time to get a feel for it. This is how it works:

1. Given the data,
2. and given our choice of model,
3. what values of the parameters of that model
4. make the observed data most likely?

We judge the model on the basis how likely the data would be ***if the model were correct***.

The Principle of Parsimony (Occam’s Razor) ★★★★★

The principle of parsimony is attributed to the early fourteenth-century English nominalist philosopher, William of Occam, who insisted that, given a set of equally good explanations for a given phenomenon, *the correct explanation is the simplest explanation*.

It is called Occam’s razor because he ‘shaved’ his explanations down to the bare minimum: his point was that in explaining something, assumptions must not be needlessly multiplied.

The Principle of Parsimony (Occam's Razor) ★★★★★

For statistical modeling, the principle of parsimony means that:

- Models should have as few parameters as possible
- Linear models should be preferred to non-linear models
- *Experiments relying on few assumptions should be preferred to those relying on many*
- Models should be pared down until they are *minimal adequate*
- Simple explanations should be preferred to complex explanations

A variable should be retained in the model only *if it causes a significant increase in deviance when it is removed from the current model*. Seek simplicity, then distrust it.

Famous Quotes on Simplicity

Einstein: "A model should be as simple as possible. But no simpler."

Oscar Wilde: "Truth is rarely pure, and never simple."

Types of Statistical Models ★★★★★

The objective of modeling is to determine a minimally adequate model from the large set of potential models that might be used to describe the given set of data.

Model	Interpretation
Saturated model	One parameter for every data point Fit: perfect Degrees of freedom: none Explanatory power of the model: none
Maximal model	Contains all (p) factors, interactions and covariates that might be of any interest. Many of the model's terms are likely to be insignificant Degrees of freedom: $n - p - 1$ Explanatory power of the model: it depends
Minimal adequate model	A simplified model with $1 \le p' \le p$ parameters Fit: less than the maximal model, but not significantly so Degrees of freedom: $n - p' - 1$ Explanatory power of the model: $r^2 = SSR/SSY$
Null model	Just one parameter, the overall mean $\bar{y}$ Fit: none; $SSE = SSY$ Degrees of freedom: $n - 1$ Explanatory power of the model: none

SSR – Sum of Squares Regression
SSY – Sum of Squares Total

Types of Models:
- Null Model
- Minimally Adequate Model
- Current Model
- Maximal Model
- Saturated model

Stepwise Progression

The stepwise progression from the saturated model (or the maximal model, whichever is appropriate) through a series of simplifications to the minimal adequate model is made on the basis of **deletion tests**.

These are *F* tests or chi-squared tests that assess the significance of the increase in deviance that results when a given term is removed from the current model.

Main Point: If the addition of a variable to a model does not add to a statistically significant decrease in deviance, then the variable should not be added to the model. A significant decrease in deviance means the variable improves the fit of the model.

Interpretation of Parsimony ★★★★★

Parsimony says that, other things being equal, we prefer:

1. A model with $n - 1$ parameters to a model with n parameters
2. A model with $k - 1$ explanatory variables to a model with k explanatory variables
3. A linear model to a model which is curved
4. A model without a hump to a model with a hump
5. A model without interactions to a model containing interactions between factors
6. A model with easy to measure variables to one with difficult to measure variables
7. A model that is based on a sound mechanistic understanding of the process over purely empirical functions
8. Statistically insignificant variables may remain if they are important to the process being modeled.
9. A model that does not contain redundant parameters to one that does

Fitting the Minimally Adequate Model ★★★★★

We achieve this by fitting a maximal model and then simplifying it by following one or more of these steps:

1. Remove non-significant interaction terms
2. Remove non-significant quadratic or other non-linear terms
3. Remove non-significant explanatory variables
4. Group together factor levels that do not differ from one another
5. In ANCOVA, set non-significant slopes of continuous explanatory variables to zero

Remember: There is just no perfect model!

Scale of Measurement ★★★★★

There may be no optimal scale of measurement for a model. Suppose, for example, we had a process that had Poisson errors with multiplicative effects amongst the explanatory variables. Then, we must choose between three different scales, each of which optimizes one of three different properties:

1. The scale of $\sqrt{y}$ would give constancy of variance;
2. The scale of $y^{2/3}$ would give approximately normal errors;
3. The scale of ln(y) would give additivity

Any measurement scale is always going to be a compromise, and we should choose the scale that gives the best overall performance of the model.

Steps involved in model simplification

Complex models have the following characteristics:

- Large numbers of explanatory variables
- Many interactions and
- Many non-linear terms

Steps involved in model simplification ★★★★★

Step 1: Fit the maximal model.	Fit all the factors, interactions and covariates of interest. Note the residual deviance. If you are using Poisson or binomial errors, check for *overdispersion and rescale* if necessary.
Step 2: Begin model simplification	Inspect the parameter estimates using the R summary() function. Remove the least significant terms first, using update, starting with the highest-order interactions.
Step 3: Delete a variable	If deviance increases insignificantly leave variable out of model. If deviance increases significantly put variable back in the model. Inspect the parameter values again.
Step 4: Repeat Step 3	Repeat step three for all variables in the model, applying the significance rule for keeping or discarding the variable. If none of the variables are significant than the null model is the minimal adequate model.

Caveats ★★★★★

- Interpret deviances and standard errors produced with fixed parameters that have been estimated from the data carefully.
- The search for 'nice numbers' should not be pursued uncritically. This means interpret model results in the same scale as presented by the numbers.

Order of Variable Deletion ★★★★★

There are two main considerations. Whether the data is

1. Orthogonal
2. Non-Orthogonal

Orthogonal variables are uncorrelated, and the order of deletion does not matter.
Non-Orthogonal variables are correlated and the order of deletion matters. Interaction terms are also impacted.

Best Practices ★★★★★

The best practice is as follows:

- Look for orthogonality in your data.
- Eliminate any correlations amongst your explanatory variables.
- Present a minimally adequate model.
- Document the non-significant terms that were omitted, and the deviance changes that resulted from their deletion.

With this information, readers can judge for themselves the relative magnitude of the non-significant factors, and the importance of correlations between the explanatory variables.

Model formulae in R ★★★★★

The structure of the model is specified in the model formula like this:

response variable ~ explanatory variable(s)

The symbol ~ reads 'is modeled as a function of'.

A simple linear regression of *y* on *x* would be written as

`y~x`

and a one-way ANOVA where gender is a two-level factor would be written as

`y~gender`

Examples

The right-hand side of the model formula shows:

- The number of explanatory variables and their identities – their attributes (e.g. continuous or categorical) are usually defined prior to the model fit;
- The interactions between the explanatory variables (if any);
- Non-linear terms in the explanatory variables.

Model	Model formula	Comments
Null	`y~1`	1 is the intercept in regression models, but here it is the overall mean *y*
Regression	`y~x`	*x* is a continuous explanatory variable
Regression through origin	`y~x-1`	Do not fit an intercept
One-way ANOVA	`y~sex`	sex is a two-level categorical variable
One-way ANOVA	`y~sex-1`	as above, but do not fit an intercept (gives two means rather than a mean and a difference)
Two-way ANOVA	`y~sex + genotype`	genotype is a four-level categorical variable
Factorial ANOVA	`y~N * P * K`	*N*, *P* and *K* are two-level factors to be fitted along with all their interactions
Three-way ANOVA	`y~N*P*K - N:P:K`	As above, but do not fit the three-way interaction
Analysis of covariance	`y~x + sex`	A common slope for *y* against *x* but with two intercepts, one for each sex
Analysis of covariance	`y~x * sex`	Two slopes and two intercepts
Nested ANOVA	`y~a/b/c`	Factor *c* nested within factor *b* within factor *a*
Split-plot ANOVA	`y~a*b*c+Error(a/b/c)`	A factorial experiment but with three plot sizes and three different error variances, one for each plot size
Multiple regression	`y~x + z`	Two continuous explanatory variables, flat surface fit
Multiple regression	`y~x * z`	Fit an interaction term as well (`x + z + x:z`)
Multiple regression	`y~x + I(x^2) + z + I(z^2)`	Fit a quadratic term for both *x* and *z*
Multiple regression	`y <- poly(x,2) + z`	Fit a quadratic polynomial for *x* and linear *z*
Multiple regression	`y~(x + z + w)^2`	Fit three variables plus all their interactions up to two-way
Non-parametric model	`y~s(x) + s(z)`	*y* is a function of smoothed *x* and *z* in a generalized additive model
Transformed response and explanatory variables	`log(y)~I(1/x) + sqrt(z)`	All three variables are transformed in the model

Note: In a model formula, the function I (upper case 'I') stands for 'as is' and is used for generating sequences, `I(1:10)`, or calculating quadratic terms, `I(x^2)`.

Note on Formulas ★★★★★

It is very important to note that symbols are used differently in model formulae than in arithmetic expressions. In particular:

+ indicates inclusion of an explanatory variable in the model (not addition);

- indicates deletion of an explanatory variable from the model (not subtraction);

* indicates inclusion of explanatory variables and interactions (not multiplication);

/ indicates nesting of explanatory variables in the model (not division);

| indicates conditioning (not 'or'), so that `y~x | z` is read as '*y* as a function of *x* given *z*'.

Special Symbols

A colon denotes an interaction, so that `A:B` means the two-way interaction between *A* and *B*, and `N:P:K:Mg` means the four-way interaction between *N*, *P*, *K* and *Mg*.

Some terms can be written in an expanded form. Thus:

`A*B*C` is the same as `A + B + C + A:B + A:C + B:C + A:B:C`

`A/B/C` is the same as `A + B%in%A + C%in%B%in%A`

`(A+B+C)^3` is the same as `A*B*C`

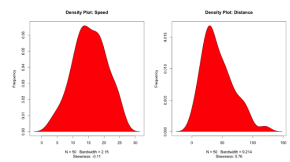
`(A+B+C)^2` is the same as `A*B*C - A:B:C`

Factorial Designs

HW

Using the “cars” data set, write a script to perform the following:

1. Print out the first six observations
2. Create a scatter plot where x = cars$speed and y = cars$dist
3. Add a density line to the scatter plot
4. Divide the graph into two columns and create two separate boxplots. One for speed and one for distance.
5. Create density plots for speed and density and label each plot with skewness measures. Add the polygon density plots. Your graphs should look like the ones to the right.
6. Calculate the correlation between speed and distance
7. Build the linear model where distance is a function of speed.
8. Calculate the ANOVA table for this model
9. What are your conclusions about the relationship between distance and speed.



HW

Using the “data-marketing-budget-12mo.csv” data set, write a script to perform the following:

1. Read the dataset in your program into a variable called “dataset”
2. Print the first 6 records
3. Create boxplots on spend and sales
4. What does the relationship tell you?
5. Create a scatterplot spend and sales
6. Fit the model sales as a function of spend
7. What does the relationship tell you?
8. Create the model fit plots
9. What does the residual analysis tell you?
10. Print the model summary statistics
11. What does the regression output tell you?
12. What can you conclude about the relationship between sales and spend?

Interactions Between Explanatory Variables

Two Types:

A. Categorical
B. Continuous

Interactions Between Explanatory Variables: Categorical

Interactions between two two-level categorical variables of the form A*B means that two main effects and one interaction mean are evaluated.

Experimental Design

		Factor A	
		Level 1	Level 2
Factor B	Level 1	A1B1	A2B1
	Level2	A1B2	A2B2

Number of parameters estimated: 1

Experimental Design

		Factor A		
		Level 1	Level 2	Level 3
Factor B	Level 1	A1B1	A2B1	A3B1
	Level2	A1B2	A2B2	A3B2
	Level 3	A1B3	A2B3	A3B3
	Level 4	A1B4	A2B4	A3B4

Number of parameters estimated: 6

Formula: Number of parameters estimated = (Number of Row Levels – 1) (Number of Column Levels -1) ★★★★★

HW

Use the code below to generate the data needed for this exercise

```
>url = 'http://stats191.stanford.edu/data/salary.table'
>salary.table <- read.table(url, header=T)
>salary.table$E <- factor(salary.table$E)
>salary.table$M <- factor(salary.table$M)
```

In this example, we have data on salaries of employees in IT based on their years of experience, their education level and whether or not they are management.

Outcome: S, salaries for IT staff in a corporation.

Predictors:

X, experience (years)
E, education (1=Bachelor's, 2=Master's, 3=Ph.D)
M, management (1=management, 0=not management)

HW

Write a script to create the following models:

1. Salary (S) = Experience (X) + Education (E) + Management (M)
2. Create the summary residual plots
3. What do they tell you about normality
4. Print the summary statistics?
5. Which variable(s) have the smallest p-values?
6. Which variable(s) have the smallest standard errors?
7. Create the ANOVA table.
8. What is it tell you about the model?
9. Add the interaction term X:E to the model.
10. Does the interaction term improve the fit of the model? Demonstrate with the ANOVA table by comparing it to the ANOVA table without interaction terms.

The intercept as parameter ★★★★★

The simple command causes the null model to be fitted.

```
y ~ 1
```

- This works out to be the grand mean (the overall average) of all the data.
- The total deviance (SSE) equals the total sum of squares, *SSY*, in models with normal errors and the identity link.
- In some cases, this may be the minimal adequate model.

The intercept as parameter with continuous data ★★★★★

To remove the intercept (parameter 1) from a regression model (i.e. to force the regression line through the origin) you fit '–1' like this:

```
y ~ x - 1
```

Most insurance models do not to this because the intercept plays an important role in pricing.

The intercept as parameter with categorical data ★★★★★

Removing the intercept from an ANOVA model where all the variables are categorical has a different effect:

```
y ~ sex - 1
```

This gives the mean for males and the mean for females in the summary table, rather than the mean for females and the difference in mean for males.

Model Formula for Regression ★★★★★

The important point to grasp is that model formulae look like equations but there are important differences.

Our simplest useful equation looks like this:

$$y = a + bx.$$

It is a two-parameter model with one parameter for the intercept, a, and another for the slope, b, of the graph of the continuous response variable y against a continuous explanatory variable x.

Model Formula for Regression ★★★★★

The model formula for the same relationship looks like this:

```
y ~ x
```

The equal sign is replaced by a tilde, and all the parameters are left out.

This is a Simple Linear Regression model.

Model Formula for Regression ★★★★★

This is the Multiple Linear Regression model.

A multiple regression model with two explanatory variables *x* and *z*, the equation would be

```
y = a + bx + cz,
```

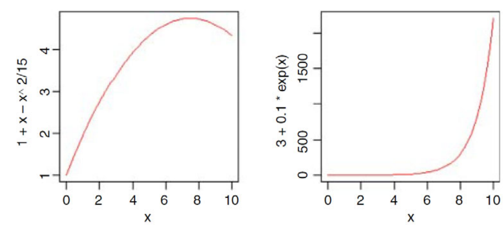
but the R model formula is

```
y~x + z
```

Common Misconception ★★★★★

A common misconception is that linear models involve a straight-line relationship between the response variable and the explanatory variables. This is *not* the case, as you can see from these two linear models:

```
windows(7,4)
par(mfrow=c(1,2))
x <- seq(0,10,0.1)
plot(x,1+x-x^2/15,type="l",col="red")
plot(x,3+0.1*exp(x),type="l",col="red")
```



37

Definition of a Linear Model ★★★★★

The definition of a linear model is an equation that contains mathematical variables, parameters and random variables and that is linear in the parameters!

Examples:

$$y = a + bx$$
$$y = a + bx - cx^2$$
$$y = a + be^x$$

These are all LINEAR models. The highest degree of a, b, c is 1.

$$y = \exp(a + bx)$$
$$\ln(y) = a + bx$$

This is a NON-LINEAR model, but it can be transformed into a LINEAR model by taking natural log of both sides. The log function is the link between these two equations and is therefore called a Log-Link Function.

38

Box–Cox Transformations ★★★★★

A Box-Cox transformation is a way to transform non-normal ***dependent variables*** into a normal shape.

The idea is to find the power transformation, λ (lambda), that maximizes the likelihood when a specified set of explanatory variables is fitted to

$$y(\lambda) = \begin{cases} \frac{y^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0 \\ \log y, & \text{if } \lambda = 0 \end{cases}$$

This test only works for positive data.

Box–Cox Transformations ★★★★★

A Box-Cox transformation for non-negative y-values

$$y(\boldsymbol{\lambda}) = \begin{cases} \frac{(y+\lambda_2)^{\lambda_1} - 1}{\lambda_1}, & \text{if } \lambda_1 \neq 0 \\ \log(y + \lambda_2), & \text{if } \lambda_1 = 0 \end{cases}$$

Remember:

The aim of the Box-Cox transformations is to ensure the usual assumptions for linear models hold.

Testing all possible values by hand is unnecessarily labor intensive; most software packages will include an option for a Box-Cox transformation. The command in R is >boxcox().

Box–Cox Transformations ★★★★★

Below are some common values for lambda

- *lambda* = -3.0 is a cube reciprocal transform.
- *lambda* = -2.0 is a square reciprocal transform.
- *lambda* = -1.0 is a reciprocal transform.
- *lambda* = -0.5 is a reciprocal square root transform.
- *lambda* = 0.0 is a log transform.
- *lambda* = 0.5 is a square root transform.
- *lambda* = 1.0 is no transform.
- *lambda* = 2.0 is a square transform.
- *lambda* = 3.0 is a cube transform.

Box–Cox Transformations ★★★★★

Example: In this example, we want to find the optimal transformation of the response variable, which is timber volume:

```
data <- read.delim("c:\\temp\\timber.txt")
attach(data)
names(data)
[1] "volume" "girth" "height"

library(MASS)
```

The MASS library is required to use the boxcox() function.

The `boxcox` function is very easy to use. Just specify the model formula, and the default options take care of everything else.

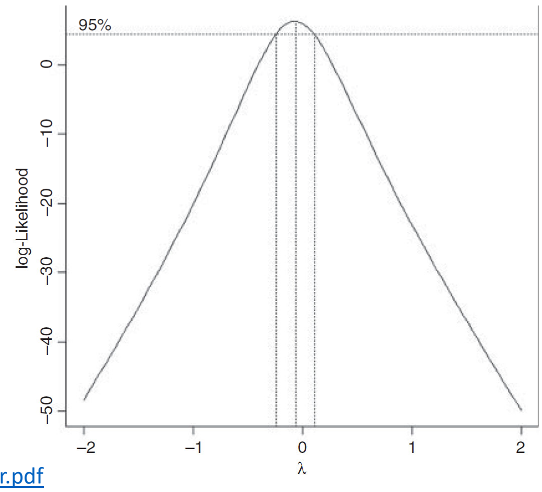
```
>boxcox(volume~log(girth)+log(height))
```

Box–Cox Transformations ★★★★★

It is clear that the optimal value of lambda is close to zero (i.e. the log transformation).

$$f(y_{ij}|z_i) = \frac{y_{ij}^{\lambda-1}}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y_{ij}^{(\lambda)} - x_i^T\beta - z_i)^2\right]$$

$$L(\lambda, \beta, \sigma^2, g) = \prod_{i=1}^{r} \int \left[\prod_{j=1}^{n_i} f(y_{ij}|z_i)\right] g(z_i)dz_i \approx \prod_{i=1}^{r}\sum_{k=1}^{K} \pi_k m_{ik}$$



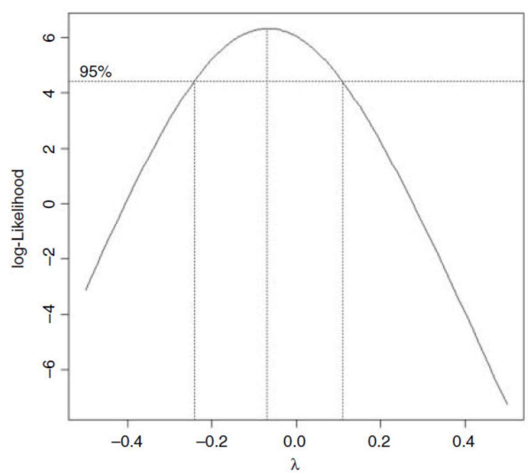
http://www.maths.dur.ac.uk/~dma0je/Posters/iwsm64_poster.pdf

Box–Cox Transformations ★★★★★

We can zoom in to get a more accurate estimate by specifying our own, non-default, range of lambda values. It looks as if it would be sensible to plot from –0.5 to +0.5:

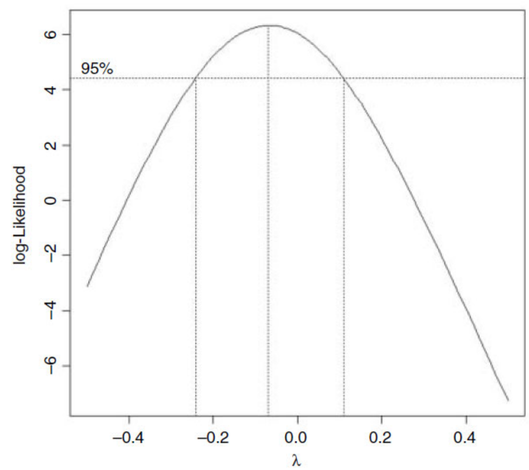
```
>boxcox(volume~log(girth)+log(height),
    lambda=seq(-0.5,0.5,0.01))
```

The likelihood is maximized at $\lambda \approx -0.08$, but the log-likelihood for $\lambda=0$ is very close to the maximum. This also gives a much more straightforward interpretation, so we would go with that, and model log(volume)as a function of log(girth) and log(height).



Box–Cox Transformations ★★★★★

Note: Even though $\lambda \approx -0.08$ is the best answer, the modeler went with $\lambda = 0$ because it results in the log(y) and that is easier to explain without a lot of loss in accuracy.



Box–Cox Transformations ★★★★★

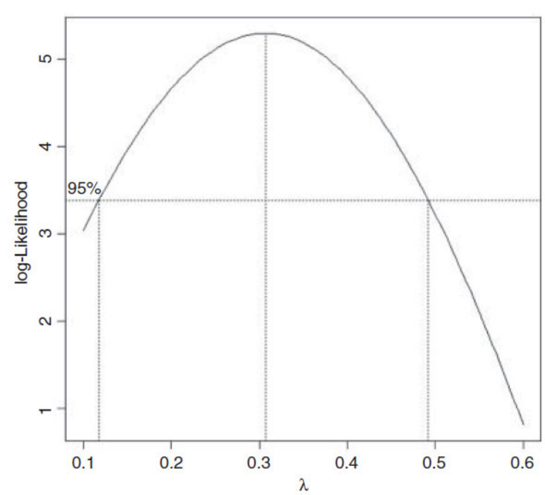
What would have been the optimal transformation of volume in that case if we had not log-transformed the explanatory variables?

To find out, we rerun the `boxcox` function, simply changing the model formula like this:

```
>boxcox(volume~girth+height)
```

We can zoom in from 0.1 to 0.6 like this:

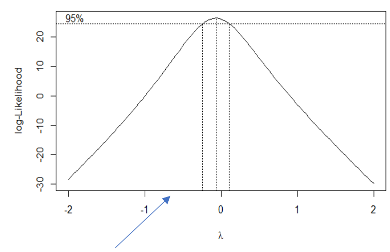
```
>boxcox(volume~girth+height,lambda=seq(0.1,
0.6,0.01))
```



Extraction Values from Box–Cox Transformations

```
data <- read.delim("c:\\temp\\timber.txt")
attach(data)
library(MASS)
bc<-boxcox(volume~log(girth)+log(height))

#Find lambda that maximizes log-Likelihood function
MaxLambda <-bc$x[which.max(bc$y)]
MaxLambda
```



λ = -0.06060606

[1] -0.06060606

bc$x – X axis values
Bc$y – Y axis values

HW: Box – Cox Transformations

You need the following library commands for this exercise:

>library(MASS)
>install.packages("faraway") ← The savings data is in the faraway package
>library(faraway)

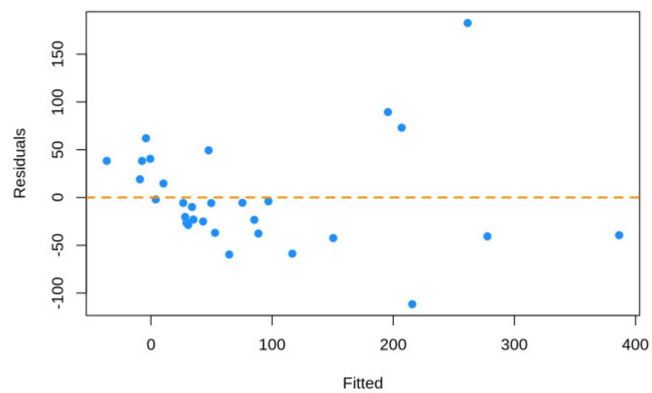
Using the savings dataset, create the following model in R

>savings_model = lm(**sr ~ .**, data = savings)

1. Use the boxcox function to find the best transformation of the data. Find the exact answer in the results using lambda sequenced from 0.5 to 1.5 by 0.1.
2. Calculate the Shapiro-Wilks Test on variable – sr in the savings dataset.

HW: Box – Cox Transformations

Does this residual plot suggest the data is from a normal distribution? Why?



Model Criticism ★★★★★

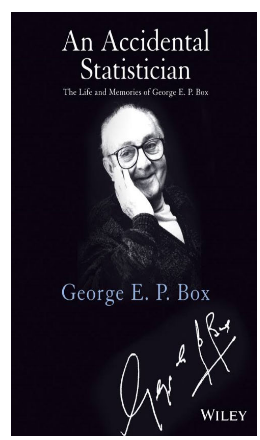
There is a temptation to become personally attached to a particular model. Statisticians call this 'falling in love with your model'.

It is as well to remember the following truths about models:

1. All models are wrong.
2. Some models are better than others.
3. The correct model can never be known with certainty.
4. The simpler the model, the better it is.

"Essentially, all models are wrong, but some are useful."

– *George E.P. Box*



Model Inadequacies ★★★★★

The model might:

1. Predict some of the *y* values poorly;
2. Show non-constant variance;
3. Show non-normal errors;
4. Be strongly influenced by a small number of influential data points;
5. Show some sort of systematic pattern in the residuals;
6. Exhibit overdispersion.

Techniques to Improve Model Fit ★★★★★

Techniques to try:

1. Transform the response variable.
2. Transform one or more of the explanatory variables.
3. Try fitting different explanatory variables if you have any.
4. Use a different error structure.
5. Use non-parametric smoothers instead of parametric functions.
6. Use different weights for different *y* values.

Model Checking ★★★★★

After fitting a model to data we need to investigate how well the model describes the data. In particular, we should look to see if there are any systematic trends in the goodness of fit.

We can work with the raw residuals:

residuals = actual *y* values − fitted *y* values

Model Checking ★★★★★

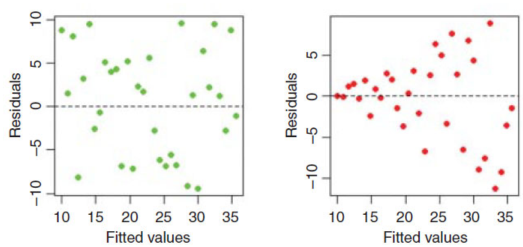
We should routinely plot the residuals against:

1. the fitted values (to look for heteroscedasticity);
2. the explanatory variables (to look for evidence of curvature);
3. the sequence of data collection (to look for temporal correlation);
4. standard normal deviates (to look for non-normality of errors).

Heteroscedasticity ★★★★★

A plot of standardized residuals against fitted values should look like the sky at night (points scattered at random over the whole plotting region), with no trend in the size or degree of scatter of the residuals.

A common problem is that the variance increases with the mean, so that we obtain an expanding, fan-shaped pattern of residuals (right-hand panel):



The plot on the left is what we want to see: no trend in the residuals with the fitted values.

The plot on the right is a problem. There is a clear pattern of increasing residuals as the fitted values get larger. This is a picture of what heteroscedasticity looks like.

Non-Normality of Errors ★★★★★

Errors may be non-normal for several reasons.

- They may be skewed, with long tails to the left or right.
- They may be kurtotic, with a flatter or more pointy top to their distribution.

Linear modeling theory is based on the assumption of normal errors. If the errors are *not* normally distributed, then we may not know how this affects our interpretation of the data and our model inferences will likely be incorrect.

Interpreting Normal Error Plots ★★★★★

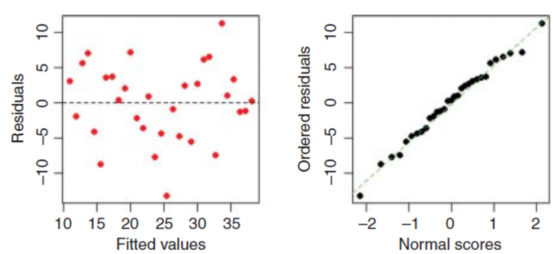
It takes considerable experience to interpret normal error plots. The function `mcheck` can help investigate the error plots. This function was developed by John Nelder, British Statistician.

```
mcheck <- function (obj, ...){      A modeling object like an ANOVA table is passed as an object.
rs <- obj$resid
fv <- obj$fitted                    These variables extract residual and fitted values from the modeling object.
windows(7,4)
par(mfrow=c(1,2))
plot(fv, rs, xlab="Fitted Values", ylab="Residuals", pch=16,col="red")
abline(h=0, lty=2)
qqnorm(rs, xlab="Normal scores", ylab="Ordered residuals", main="", pch=16)
qqline(rs, lty=2,col="green")
par(mfrow=c(1,1))
invisible(NULL)                     These two statements reset the graphics environment.
}
```


Interpreting Error Plots

Example – Normal Errors for the Model: $y = 10 + x + \varepsilon$ where the errors, ε, have zero mean.

```
x <- 0:30
e <- rnorm(31,mean=0,sd=5)
yn <- 10+x+e
mn <- lm(yn~x)
mcheck(mn)
```

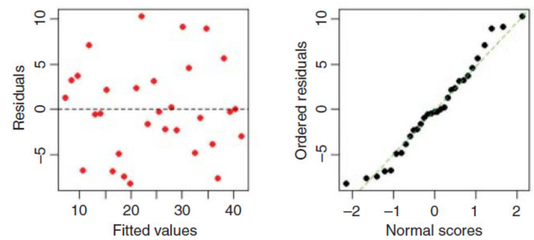


There is no suggestion of non-constant variance (left plot) and the normal plot (right) is reasonably straight.

Interpreting Error Plots

Example – Uniform Errors for the Model: $y = 10 + x + \varepsilon$ where the errors, ε, have zero mean.

```
x <- 0:30
eu <- 20*(runif(31)-0.5)
yu <- 10+x+eu
mu <- lm(yu~x)
mcheck(mu)
```

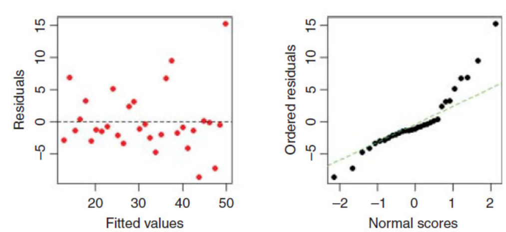


Uniform errors show up as an S-shaped pattern in the quantile–quantile plot on the right. The fit in the center is fine, but the largest and smallest residuals are too small (they are constrained in this example to be ± 10).

Interpreting Error Plots

Example – Negative Binomial Errors for the Model: $y = 10 + x + \varepsilon$ where the errors, ε, have zero mean.

```
x <- 0:30
enb <- rnbinom(31,2,.3)
ynb <- 10+x+enb
mnb <- lm(ynb~x)
mcheck(mnb)
```

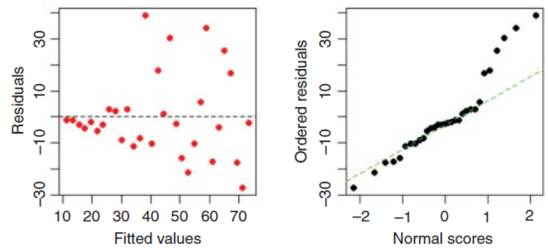


The large negative residuals are all above the line, but the most obvious feature of the plot is the single, very large positive residual (in the top right-hand corner). In general, negative binomial errors will produce a J-shape on the quantile–quantile plot. The biggest positive residuals are much too large to have come from a normal distribution. These values may turn out to be highly influential.

Interpreting Error Plots

Example – Negative Binomial Errors for the Model: $y = 10 + x + \varepsilon$ where the errors, ε, have zero mean.

```
x <- 0:30
eg <- rgamma(31,1,1/x)
yg <- 10+x+eg
mg <- lm(yg~x)
mcheck(mg)
```



The left-hand plot shows the residuals increasing steeply with the fitted values and illustrates an asymmetry between the size of the positive and negative residuals. The right-hand plot shows the highly non-normal distribution of errors.

Homework

1. A two analysis of variance model have 5 levels on one factor and 4 on the other factor. How many parameters will be estimated:
   A. 5
   B. 4
   C. 9
   D. 12
   E. 20

2. (T/F) A linear model is NOT linear in the parameters but in the random variables.

3. Use the mcheck function to investigate the errors for the following distributions
   a) Beta (a=2, b=3)
   b) Weibull(α = 2, λ = 4)
   c) Logistic (μ=3, s = 2)

Homework

4. Find the optimal Box Cox power transformation parameter using the cars data set and the model: dist ~ speed.
   a. What value of lambda maximized the log-likelihood function?
   b. What is the maximum likelihood value?

Influence ★★★★★

One of the most common reasons for a lack of fit is the existence of outliers in the data.

A point may *appear* to be an outlier because of misspecification of the model, and not because there is anything wrong with the data.

It is key to understand that an analysis of residuals is a very poor way of looking for influence. A point is highly influential. It forces the regression line close to it, and hence the influential point may have a very small residual.

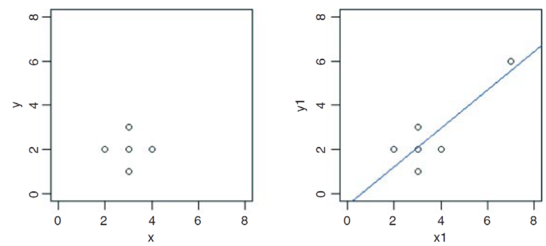
Influence ★★★★★

Example:

```
x <- c(2,3,3,3,4)
y <- c(2,3,2,1,2)
windows(7,4)
par(mfrow=c(1,2))
plot(x,y,xlim=c(0,8),ylim=c(0,8))

x1 <- c(x,7)
y1 <- c(y,6)
plot(x1,y1,xlim=c(0,8),ylim=c(0,8))
abline(lm(y1~x1),col="blue")
```

No significant regression of *y* on *x*.



Significant regression of *y* on *x*.

The outlier is responsible for the significant relationship and is said to be highly ***influential***.

Influence ★★★★★

Describing magnitude of outlier's influence.

2nd Smallest in Absolute Value.

```
reg <- lm(y1~x1)
summary(reg)
Call:
lm(formula = y1 ~ x1)
Residuals:
      1       2        3        4        5       6
0.78261 0.91304 -0.08696 -1.08696 -0.95652 0.43478
Coefficients:
            Estimate       Std. Error    t value       Pr(>|t|)
(Intercept) -0.5217        0.9876        -0.528        0.6253
x1            0.8696       0.2469        3.522         0.0244 *
```

Topic 4: Statistical Modeling

Influence ★★★★★

Describing magnitude of outlier's influence.

```
influence.measures(reg)

Influence measures of
         lm(formula = y1 ~ x1) :

   dfb.1  dfb.x1   dffit cov.r  cook.d   hat inf
1  0.687 -0.5287  0.7326 1.529 0.26791 0.348
2  0.382 -0.2036  0.5290 1.155 0.13485 0.196
3 -0.031  0.0165 -0.0429 2.199 0.00122 0.196
4 -0.496  0.2645 -0.6871 0.815 0.19111 0.196
5 -0.105 -0.1052 -0.5156 1.066 0.12472 0.174
6 -3.023  4.1703  4.6251 4.679 7.62791 0.891   *
```

Point 6 has the largest Cook's d and is therefore the greatest influencer on the model.

Topic 4: Statistical Modeling

Influence ★★★★★

Describing magnitude of outlier's influence.

```
lm.influence(reg)
$hat
        1         2         3         4         5         6
0.3478261 0.1956522 0.1956522 0.1956522 0.1739130 0.8913043
$coefficients
   (Intercept)           x1
1  0.67826087 -0.130434783
2  0.37015276 -0.049353702
3 -0.03525264  0.004700353
4 -0.44065805  0.058754407
5 -0.10068650 -0.025171625
6 -2.52173913  0.869565217
$sigma
        1         2         3         4         5         6
0.9660918 0.9491580 1.1150082 0.8699177 0.9365858 0.8164966
$wt.res
          1           2           3           4           5           6
 0.78260870  0.91304348 -0.08695652 -1.08695652 -0.95652174  0.43478261
```

This command will produce all the influence measures.

Influence Metric. The larger the value the larger the influence.

Provides impact on model when i^{th} data element removed.

Provides impact on standard error when i^{th} data element removed.

A vector of weighted residuals (or deviance residuals in a generalized linear model) or raw residuals if weights are not set.

Influence ★★★★★

Note: The residual error or the error not accounted for by the model drops by $0.8164966^2 = 0.666666$ when the outlier is dropped from the analysis.

```
summary.aov(lm(y1[-6]~x1[-6]))

            Df Sum Sq Mean Sq F value Pr(>F)
x1[-6]       1      0  0.0000       0      1
Residuals    3      2  0.6667
```

This value indicates the model is not fit very well by a linear model. This should make sense since the data forms a circle.

HW: Influence

For the models you built on the slides below, calculate the influence metric to assess the data elements having most influence on the model.

1. Slide 25
2. Slide 26
3. Slide 30
4. Slide 48

Summary of statistical models in R ★★★★★

`lm` fits a linear model with normal errors and constant variance; generally this is used for regression analysis using continuous explanatory variables.

`aov` fits analysis of variance with normal errors, constant variance and the identity link; generally used for categorical explanatory variables or ANCOVA with a mix of categorical and continuous explanatory variables.

`glm` fits generalized linear models to data using categorical or continuous explanatory variables, by specifying one of a family of **error structures** (e.g. Poisson for count data or binomial for proportion data) and a particular **link function**.

`gam` fits generalized additive models to data with one of a family of error structures (e.g. Poisson for count data or binomial for proportion data) in which the continuous explanatory variables can (optionally) be fitted as arbitrary smoothed functions using non-parametric smoothers rather than specific parametric functions.

Summary of statistical models in R ★★★★★

`lme` and `lmer`
fit linear mixed-effects models with specified mixtures of fixed effects and random effects and allow for the specification of correlation structure among the explanatory variables and autocorrelation of the response variable (e.g. time series effects with repeated measures).

`lmer` allows for non-normal errors and non-constant variance with the same error families as a GLM.

`nls` fits a non-linear regression model via least squares, estimating the parameters of a specified non-linear function.

`nlme` fits a specified non-linear function in a mixed-effects model where the parameters of the non-linear function are assumed to be random effects; it allows for the specification of correlation structure among the explanatory variables and autocorrelation of the response variable (e.g. time series effects with repeated measures).

Summary of statistical models in R ★★★★★

`loess` fits a local regression model with one or more continuous explanatory variables using non-parametric techniques to produce a smoothed model surface.

`tree` and `rpart`
fit a regression tree model using binary recursive partitioning whereby the data are successively split along coordinate axes of the explanatory variables so that at any node the split is chosen that maximally distinguishes the response variable in the left and right branches. With a categorical response variable, the tree is called a classification tree, and the model used for classification assumes that the response variable follows a *multinomial distribution.*

Generic Modeling Functions ★★★★★

`summary` produces parameter estimates and standard errors from `lm`, and ANOVA tables from `aov`; this will often determine your choice between `lm` and `aov`. For either `lm` or `aov` you can choose `summary.aov` or `summary.lm` to get the alternative form of output (an ANOVA table or a table of parameter estimates and standard errors; see p. 517).

`plot` produces diagnostic plots for model checking, including residuals against fitted values, normality checks, influence tests, etc.

`anova` is a wonderfully useful function for comparing different models and producing ANOVA tables.

`update` is used to modify the last model fit; it saves both typing effort and computing time.

`coef` gives the coefficients (estimated parameters) from the model.

Generic Modeling Functions ★★★★★

`fitted` gives the fitted values, predicted by the model for the values of the explanatory variables included.

`resid` gives the residuals (the differences between measured and predicted values of *y*).

`predict` uses information from the fitted model to produce smooth functions for plotting a line through the scatterplot of your data. Make sure you provide a list or a dataframe containing all of the necessary information on each of the explanatory variables in your model to enable the prediction to be made.

Topic 4: Statistical Modeling

Model Fitting Function Arguments ★★★★★

Useful options:

1. `subset`
2. `weights`
3. `data`
4. `offset`
5. `na.action`

Model Fitting Function Arguments ★★★★★

Ipomopsis dataset

	Root	Fruit	Grazing
1	6.225	59.77	Ungrazed
2	6.487	60.98	Ungrazed
3	4.919	14.73	Ungrazed
4	5.130	19.28	Ungrazed
5	5.417	34.25	Ungrazed
6	5.359	35.53	Ungrazed
7	7.614	87.73	Ungrazed
8	6.352	63.21	Ungrazed
9	4.975	24.25	Ungrazed
10	6.930	64.34	Ungrazed

40 observations
3 variables

An example involving analysis of covariance with a mix of both continuous and categorical explanatory variables:

```
data <- read.table("c:\\temp\\ipomopsis.txt",header=T)
attach(data)
names(data)

[1] "Root" "Fruit" "Grazing"
```

The response is seed production (`Fruit`) with a continuous explanatory variable (`Root`, Root diameter) and a two-level factor (`Grazing`, with levels `Grazed` and `Ungrazed`).

Model Fitting Function Arguments: Subsets ★★★★★

Fitting a model to a subset of data.

```
model <- lm(Fruit[Grazing=="Grazed"]~Root[Grazing=="Grazed"])
```

Dependent Variable

Independent Variable

Model Operator

[ ] Notice how subscripting is used only certain values of "Grazing"!

Model Fitting Function Arguments: Subsets ★★★★★

Fitting a model to a subset of data.

```
model <- lm(Fruit[Grazing=="Grazed"]~Root[Grazing=="Grazed"])
```

The following works equally as well and is more compact.

```
model <- lm(Fruit~Root, subset=(Grazing=="Grazed"))
```

Notice the subset () function!

Model Fitting Function Arguments: Weights ★★★★★

The default is for all the values of the response to have equal weights (all equal to 1).

```
weights = rep(1, n.observations)
```

Where data points are to be weighted unequally, the classical approach is to weight each value by the inverse of the variance of the distribution from which that point is drawn. This downplays the influence of highly variable data.

Model Fitting Function Arguments: Weights ★★★★★

Instead of using initial root size as a covariate (as above) you could use Root as a weight in fitting a model with Grazing as the sole categorical explanatory variable

```
model <- lm(Fruit~Grazing,weights=Root)
summary(model)
```

```
Call:
lm(formula = Fruit~Grazing, weights = Root)
Coefficients:
                  Estimate   Std. Error   t value   Pr(>|t|)
(Intercept)       70.725     4.849        14.59     <2e-16     ***
GrazingUngrazed   -16.953    7.469        -2.27     0.029      *

Residual standard error: 62.51 on 38 degrees of freedom
Multiple R-Squared: 0.1194,  Adjusted R-squared: 0.0962
F-statistic: 5.151 on 1 and 38 DF,  p-value: 0.02899
```


Model Fitting Function Arguments: Weights ★★★★★

When weights (w) are specified the model is fitted using weighted least squares, in which the quantity to be minimized is $w \times d^2$ (rather than d^2), where d is the difference between the response variable and the fitted values predicted by the model.

The use of weights alters the parameter estimates and their standard errors:

```
model <- lm(Fruit~Grazing)
summary(model)
```

```
Coefficients:
                  Estimate    Std. Error   t value    Pr(>|t|)
     (Intercept)   67.941          5.236    12.976    1.54e-15  ***
GrazingUngrazed   -17.060          7.404    -2.304      0.0268  *

Residual standard error: 23.41 on 38 degrees of freedom
Multiple R-Squared: 0.1226,  Adjusted R-squared: 0.09949
F-statistic: 5.309 on 1 and 38 DF,  p-value: 0.02678
```

Model Fitting Function Arguments: Weights ★★★★★

Conclusion:

1. Fitting root size as a statistical weight is scientifically wrong in this case: why should values from larger plants be given greater influence?
2. Also, this analysis gives entirely the wrong interpretation of the data (ungrazed plants come out as being *less* fecund than the grazed plants).
3. Analysis of covariance reverses this interpretation, showing that for a given root size, the grazed plants produced 36.013 *fewer* fruits than the ungrazed plants; the problem was that the big plants were almost all in the grazed treatment.

Model Fitting Function Arguments: Missing Values ★★★★★

What to do about missing values in the dataframe is an important issue. If there are missing values, you have two choices:

1. Leave out any row of the dataframe in which one or more variables are missing, then `na.action = na.omit`
2. Fail the fitting process, so `na.action = na.fail (Will stop process is there are NAs in the data.)`

Model Fitting Function Arguments: Missing Values ★★★★★

```
>Root[37] <- NA

>model <- lm(Fruit~Grazing*Root)
Error in eval(predvars, data, env) : object 'Fruit' not found


>model <- lm(Fruit~Grazing*Root, na.action=na.fail)

Error in na.fail.default(list(Fruit = c(59.77, 60.98, 14.73, 19.28, 34.25, :
       missing values in object
```

If you are carrying out regression with time series data that include missing values, then you should use `na.action = NULL` so that residuals and fitted values are time series as well (if the missing values were omitted, then the resulting vector would not be a time series of the correct length).

Model Fitting Function Arguments: Offsets ★★★★★

You would not use offsets with a linear model (you could simply subtract the offset from the value of the response variable, and work with the transformed values).

But with generalized linear models you may want to specify part of the variation in the response using an offset.

Dataframes containing the same variable names ★★★★★

If you have several different dataframes containing the same variable names (say, *x* and *y*) then the simplest way to ensure that the correct variables are used in the modelling is to name the dataframe in the function call:

```
model <- lm(y~x,data=correct.frame)
```

The alternative is much more cumbersome to type:

```
model <- lm(correct.frame$y~correct.frame$x)
```


Akaike's information criterion ★★★★★

Akaike's information criterion (AIC) is known in the statistics trade as a **penalized log-likelihood**. If you have a model for which a log-likelihood value can be obtained, then

$$\text{AIC} = -2 \times \log\text{-likelihood} + 2(p + 1)$$

where *p* is the number of parameters in the model, and 1 is added for the estimated variance.

Akaike's information criterion ★★★★★

To demystify AIC let us calculate it by hand. These data show the relationship between growth and dietary tannin for caterpillars in a feeding experiment:

```
data <- read.table("c:\\temp\\regression.txt",header=T)
attach(data)
names(data)

[1] "growth" "tannin"
```


Akaike's information criterion ★★★★★

The regression model for these data is worked out, one term at a time, by hand in Chapter 10 (The R Book).

```
model <- lm(growth~tannin)
```

To calculate the log-likelihood we need three quantities (p. 282): the sample size, `n`; the error variance `s2` $= \sigma_2$; and the sum of the squares of the residuals, `sse` $=(y-\mu)^2$:

```
n <- length(growth)
sse <- sum((growth-fitted(model))^2)
s2 <- sse/(n-2)
s <- sqrt(s2)
```

Akaike's information criterion ★★★★★

Recall: The formula for the log-likelihood assuming a normal distribution is

$$l(\mu, \sigma) = -\frac{n}{2}\log(2\pi) - n\log(\sigma) - \sum (y_i - \mu)^2 / 2\sigma^2$$

Now we can compute the log-likelihood:

```
-(n/2)*log(2*pi)-n*log(s)-sse/(2*s2)

[1] -16.51087
```


Akaike's information criterion ★★★★★

There is an R function `logLik` to calculate the log likelihood from any appropriate model object directly:

```
logLik(model)
'log Lik.' -16.37995 (df=3)
```

The three degrees of freedom (`df`) refer to the slope, the intercept and the variance. The difference between the two estimates is just rounding error.

Now we can compute AIC:

```
-2 * -16.37995 + 6

[1] 38.7599
```

Akaike's information criterion ★★★★★

Not surprisingly, there is an R function called `AIC` to compute the information criterion directly from the model object:

```
AIC(model)

[1] 38.7599
```


AIC as a measure of the fit of a model ★★★★★

- The more parameters there are in the model, the better the fit.
- You could obtain a perfect fit if you had a separate parameter for every data point, but this model would have absolutely no explanatory power.
- There is always going to be a trade-off between the goodness of fit and the number of parameters required by parsimony.
- AIC is useful because it explicitly penalizes any superfluous parameters in the model, by adding $2(p + 1)$ to the deviance.

AIC as a measure of the fit of a model ★★★★★

When comparing two models, the smaller the AIC, the better the fit. This is the basis of automated model simplification using step.

You can use the function AIC to compare two models, in exactly the same way as you can use anova

```
>model.1 <- lm(Fruit~Grazing*Root)
>model.2 <- lm(Fruit~Grazing+Root)
>AIC(model.1, model.2)
```

```
        df    AIC
model.1 5 263.6269
model.2 4 261.7835
```

Penalties

```
model.1:   2x(4+1)= 10
model.2:   2x(3+1)= 8
```

`Model.2` has the *lower* AIC than `model.1`. Therefore, **`Model.2 is preferred to Model.1`**

AIC as a measure of the fit of a model ★★★★★

If you want to compare many models, you can combine the models into a list,

```
models <- list (model1, model2, model3, model4, model5, model6)
```

then extract the AIC of each of them using `lapply` like this:

```
aic <- unlist(lapply(models, AIC))
```

where `aic` will be a vector of numbers in which you can search for the minimum.

HW: AIC

For the models you built on the slides below, calculate the AIC metric:

1. Slide 25
2. Slide 26
3. Slide 30
4. Slide 48

Topic 4: Statistical Modeling

Leverage ★★★★★

Points increase in influence to the extent that they lie on their own, a long way from the mean value of *x* (to either the left or right). To account for this, measures of leverage for a given data point *y* are proportional to

$$(x - \bar{x})^2$$

Leverage ★★★★★

Here are the *x* data from our earlier example:

```
x <- c(2,3,3,3,4,7)
```

The commonest measure of leverage is
$$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\Sigma(x_i - \bar{x})^2}$$
where the denominator is *SSX*.

A good rule of thumb is that a point is highly influential if its

$$h_i > \frac{2p}{n}$$

where *p* is the number of parameters in the model.

Leverage ★★★★★

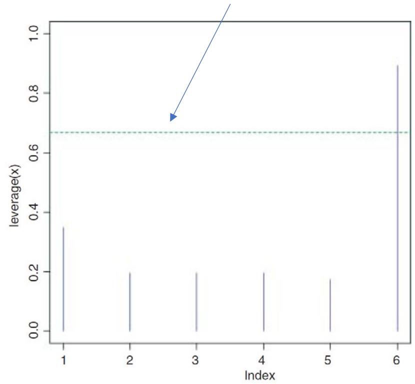
We could easily calculate the leverage value of each point in our vector. It is more efficient, perhaps, to write a general function that could carry out the calculation of the *h* values for any vector of *x* values,

```
leverage <- function(x){1/length(x)+(x-
mean(x))^2/sum((x-mean(x))^2)}
```

and then use this function with our vector of *x* values to produce a leverage plot:

```
plot(leverage(x),type="h",ylim=c(0,1),col="blue")
abline(h=4/6,lty=2,col="green")
```

The horizontal green dashed line shows Index = $2p/n$ = 4/6.



As you can see, only the sixth=point shows more leverage than is reasonable.

Misspecified Model ★★★★★

The model may have the wrong terms in it, or the terms may be included in the model in the wrong way.

When both the error distribution and functional form of the relationship are unknown, there is no single specific rationale for choosing any given transformation in preference to another. The aim is pragmatic, namely, to find a transformation that gives:

- Constant error variance;
- Approximately normal errors;
- Additivity;
- A linear relationship between the response variables and the explanatory variables;
- Straightforward scientific interpretation.

Misspecified Model ★★★★★

The choice is bound to be a compromise and, as such, is best resolved by quantitative comparison of the deviance produced under different model forms.

Testing for non-linearity in the relationship between *y* and *x* we might add a term in x_2 to the model; a significant parameter in the x^2 term indicates curvilinearity in the relationship between *y* and *x*.

A further element of misspecification can occur because of **structural non-linearity**. Such as

$$y = a + \frac{b}{x} \quad \text{OR} \quad y = a + \frac{b}{c + x}$$

Model checking in R

The data we examine in this section are on the decay of a biodegradable plastic in soil: the response, *y*, is the mass of plastic remaining and the explanatory variable, *x*, is duration of burial:

```
>Decay <- read.table("c:\\temp\\Decay.txt",header=T)
>attach(Decay)

>names(Decay)
[1] "time" "amount"

>model <- lm(amount~time)
>par(mfrow=c(2,2))
>plot(model)
```


Model checking in R

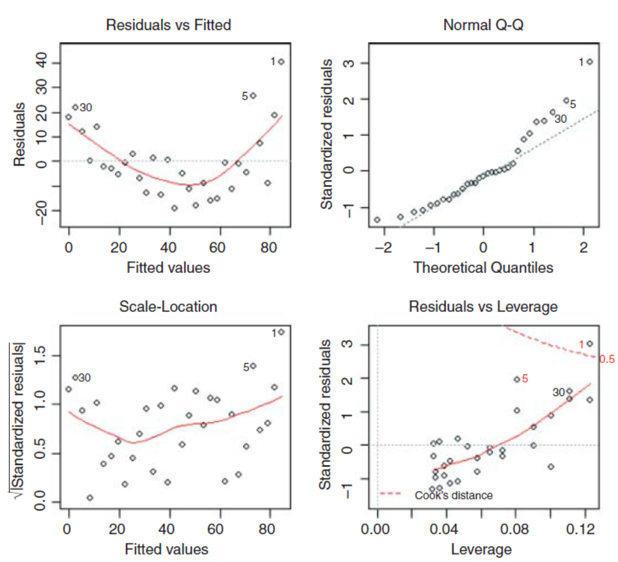
```
>plot(model)
```

This one command produces a series of graphs.

The upper two graphs are the most important.

First, you get a plot of the residuals against the fitted values (top left) which shows very pronounced curvature; most of the residuals for intermediate fitted values are negative, and the positive residuals are concentrated at the smallest and largest fitted values. Remember, this plot should look like the sky at night, with no pattern of any sort.

The relationship between *y* and *x* is non-linear rather than linear.

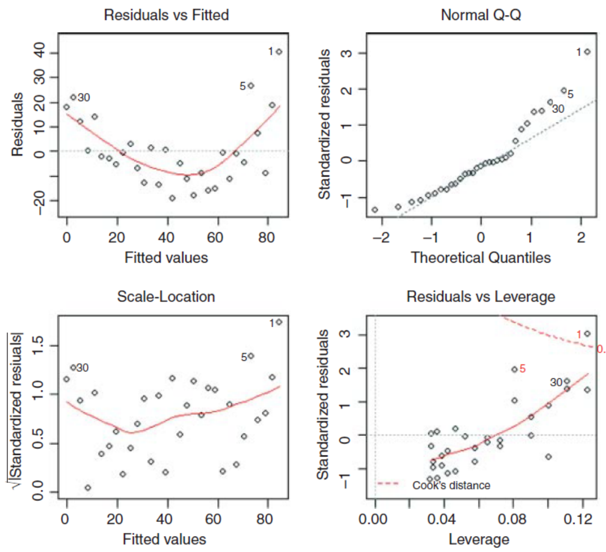


Model checking in R

Second (top right), you get a quantile–quantile plot which indicates pronounced non-normality in the residuals (the line should be straight, not banana-shaped as here).

Third, the third graph is like a positive-valued version of the first graph; it is good for detecting non-constancy of variance (heteroscedasticity), which shows up as a triangular scatter (like a wedge of cheese) with an increasing red line through it.

Fourth, the fourth graph shows a pronounced pattern in the standardized residuals as a function of the leverage. The graph also shows Cook's distance, highlighting the identity of particularly influential data points.



Cook's Distance

Cook's distance is an attempt to combine leverage and residuals in a single measure. The absolute values of the deletion residuals $|r_i^*|$ are weighted as follows:

$$C_i = |r_i^*| \left(\frac{n-p}{p} \cdot \frac{h_i}{1-h_i} \right)^{1/2}$$

Data points 1, 5 and 30 are singled out as being influential, with point 1 especially so. When we were happier with other aspects of the model, we would repeat the modelling, leaving out each of these points in turn.

Extracting information from model objects

We often want to extract material from fitted models (e.g. slopes, residuals or *p* values) and there are three different ways of doing this:

1. by name, e.g. `coef(model);`
2. with list subscripts, e.g. `summary(model)[[3]];`
3. using `$` to name the component, e.g. `model$resid.`

Extracting information from model objects

The model object we use to demonstrate these techniques is the simple linear regression run on the regression.txt data.

```
>data <- read.table("c:\\temp\\regression.txt",header=T)
>attach(data)
>names(data)

[1] "growth" "tannin"

>model <- lm(growth~tannin)

>summary(model)
```

Extracting Information From Model Objects by Name

```
Call:
lm(formula = growth ~ tannin)

Residuals:
    Min      1Q  Median      3Q     Max
-2.4556 -0.8889 -0.2389  0.9778  2.8944

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  11.7556     1.0408  11.295 9.54e-06 ***
tannin       -1.2167     0.2186  -5.565 0.000846 ***

Residual standard error: 1.693 on 7 degrees of freedom

Multiple R-squared: 0.8157,     Adjusted R-squared: 0.7893
F-statistic: 30.97 on 1 and 7 DF,  p-value: 0.0008461
```

You can extract:

- the coefficients of the model,
- the fitted values, the residuals,
- the effect sizes
- the variance–covariance matrix by name

Extracting Information From Model Objects by Name

```
>coef(model)     #Gives the intercept and other beta estimates
>fitted(model)   #Gives the predicted y values produced by the model
>resid(model)    #Gives the residuals (y – fitted values)
>vcov(model)     #Gives the variance-covariance matrix
```

Extracting Information From Model Objects by List Subscripts

```
summary.aov(model)

             [1]   [2]      [3]      [4]      [5]
             Df Sum Sq Mean Sq F value    Pr(>F)
tannin        1  88.82   88.82   30.97 0.000846 ***
Residuals     7  20.07    2.87
```

```
summary.aov(model)[[1]][1]
summary.aov(model)[[1]][2]
summary.aov(model)[[1]][3]
summary.aov(model)[[1]][4]
summary.aov(model)[[1]][5]
```

[[1]] means this is the first object in the list
[ ] refers to the column of the object. The statements return the columns indicated in [ ].

Extracting Information From Model Objects by List Subscripts

It can be quite involved to extract the numerical values that you might want to use in subsequent work.

For instance, to get the *F* ratio (30.974) out of the fourth element of the list, we need to `unlist` the object, then use `as.numeric`, and then add a further subscript:

```
as.numeric(unlist(summary.aov(model)[[1]][4]))[1]
```

This statement says unlist the elements in the 4th column of the 1st indexed object, then give me then 1st element. Notice the column headings are not treated as elements.

Extracting Information From Model Objects by List Subscripts

```
summary(model)

Call:
lm(formula = growth ~ tannin)

Residuals:
    Min      1Q  Median      3Q     Max
-2.4556 -0.8889 -0.2389  0.9778  2.8944

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  11.7556     1.0408  11.295 9.54e-06 ***
tannin       -1.2167     0.2186  -5.565 0.000846 ***

Residual standard error: 1.693 on 7 degrees of freedom
Multiple R-squared: 0.8157,     Adjusted R-squared: 0.7893
F-statistic: 30.97 on 1 and 7 DF,  p-value: 0.0008461
```


Extracting Information From Model Objects by List Subscripts

The first element of the list is the model formula (or `Call`) showing the response variable (`growth`) and the explanatory variable(s) (`tannin`):

```
>summary(model)[[1]]

lm(formula = growth ~ tannin)
```

Extracting Information From Model Objects by List Subscripts

The second describes the attributes of the object called summary(model):

```
>summary(model)[[2]]
```

```
growth ~ tannin
attr(,"variables")
list(growth, tannin)

attr(,"factors")
tannin
growth 0
tannin 1

attr(,"term.labels")
[1] "tannin"

attr(,"order")
[1] 1
attr(,"intercept")
[1] 1
attr(,"response")
[1] 1

attr(,".Environment")
<environment: R_GlobalEnv>
attr(,"predvars")
list(growth, tannin)
attr(,"dataClasses")
growth tannin
"numeric" "numeric"
```


Extracting Information From Model Objects by List Subscripts

The third gives the residuals for the nine data points:

```
summary(model)[[3]]
```

The fourth gives the parameter table, including standard errors of the parameters, *t* values and *p* values. This is the really important information:

```
summary(model)[[4]]

             Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 11.755556  1.0407991 11.294740 9.537315e-06
tannin      -1.216667  0.2186115 -5.565427 8.460738e-04
```

Extracting certain values

```
summary(model)[[4]] [1]
[1] 11.75556
summary(model)[[4]] [2]
[1] -1.216667
summary(model)[[4]] [3]
[1] 1.040799
summary(model)[[4]] [4]
[1] 0.2186115
summary(model)[[4]] [8]
[1] 0.0008460738
```

Extracting Information From Model Objects by List Subscripts

The fifth is concerned with whether the corresponding components of the fit (the model frame, the model matrix, the response or the QR decomposition) should be returned. The default is `FALSE`:

```
summary(model)[[5]]
```

```
(Intercept) tannin
FALSE FALSE
```

The sixth is the residual standard error: the square root of the error variance from the `summary.aov` table (s^2=2.867; see above):

```
summary(model)[[6]]
```

```
[1] 1.693358
```


Extracting Information From Model Objects by List Subscripts

The seventh shows the number of rows in the `summary.lm` table (showing two parameters to have been estimated from the data with this model, and the residual degrees of freedom (d.f. = 7):

```
summary(model)[[7]]
```

```
[1] 2 7 2
```

The eighth is r^2=*SSR/SST*, the fraction of the total variation in the response variable that is explained by the model (see p. 456 for details):

```
summary(model)[[8]]
```

```
[1] 0.8156633
```

Extracting Information From Model Objects by List Subscripts

The ninth is the adjusted R_2, explained on p. 461 but seldom used in practice:

```
summary(model)[[9]]
[1] 0.7893294
```

The tenth gives *F* ratio information: the three values given here are the *F* ratio (30.973 98), the number of degrees of freedom in the model (i.e. in the numerator, numdf) and the residual degrees of freedom (i.e. in the denominator, dendf):

```
summary(model)[[10]]
    value    numdf  dendf
30.97398 1.00000 7.00000
```


Extracting Information From Model Objects by List Subscripts

The eleventh component is the correlation matrix of the parameter estimates:

```
summary(model)[[11]]

               (Intercept)      tannin

   (Intercept)  0.37777778 -0.06666667
   tannin      -0.06666667  0.01666667
```

Extracting Information From Model Objects by $

```
>model$coef

>model$df
```


Using lists with models

You might want to extract the coefficients from a series of related statistical models, and you want to avoid the use of a loop.

Here are the data with `y` as a function of `x`:

```
x <- 0:100
y <- 17+0.2*x+3*rnorm(101)
```

Now create three linear models of increasing complexity:

```
model0 <- lm(y~1)
model1 <- lm(y~x)
model2 <- lm(y~x+I(x^2))
```

Make a list containing the three model objects:

```
models <- list(model0,model1,model2)
```

Using lists with models

To obtain the coefficients from the three models, it is simple to use `lapply` on the list to apply the function `coef` to each element of the list:

```
lapply(models,coef)
[[1]]
(Intercept)
26.90530

[[2]]
(Intercept) x
15.8267899 0.2215701

[[3]]
(Intercept) x I(x^2)
1.593695e+01 2.148935e-01 6.676673e-05
```


Using lists with models

To get a *vector* (rather than a list) as output, and to select only the three intercepts, we use subscripts `[c(1,2,4)]` with `unlist` and `as.vector` like this:

```
as.vector(unlist(lapply(models,coef)))[c(1,2,4)]
[1] 26.90530 15.82679 15.93695
```

Using lists with models

Here we extract the AIC of each model:

```
>lapply(models,AIC)

[[1]]
[1] 672.7502
[[2]]
[1] 510.787
[[3]]
[1] 512.5231
```

The winner. It has the lowest AIC.

summary.lm vs. summary.aov

It is important to understand the difference between `summary.lm` and `summary.aov` for the same model.

```
>comp <- read.table("c:\\temp\\competition.txt",header=T)
>attach(comp)
>names(comp)
>levels(clipping)
>model <- lm(biomass~clipping)
>summary.aov(model)
>summary.lm(model)
```

summary.lm vs. summary.aov

Take Note of Differences!

ANOVA Table

```
          Df Sum Sq Mean Sq F value   Pr(>F)
clipping   4  85356   21339   4.302 0.00875 **
Residuals 25 124020    4961
```

Regression Output

```
Call:
lm(formula = biomass ~ clipping)

Residuals:
     Min       1Q   Median       3Q      Max
-103.333  -49.667    3.417   43.375  177.667

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   465.17      28.75  16.177  9.4e-15 ***
clippingn25    88.17      40.66   2.168  0.03987 *
clippingn50   104.17      40.66   2.562  0.01683 *
clippingr10   145.50      40.66   3.578  0.00145 **
clippingr5    145.33      40.66   3.574  0.00147 **

Residual standard error: 70.43 on 25 degrees of freedom
Multiple R-squared: 0.4077,     Adjusted R-squared: 0.3129
F-statistic: 4.302 on 4 and 25 DF,  p-value: 0.008752
```

We will revisit how to calculate these numbers.

HW: ANOVA v. lm

For the models you built on the slides below, extract elements [[1]] through [[11]]:

1. Slide 25
2. Slide 26
3. Slide 30
4. Slide 48

Model simplification by stepwise deletion

A stepwise *a posteriori* procedure is to aggregate non-significant factor levels in a model.
Let's look at an example:

```
comp <- read.table("c:\\temp\\competition.txt",header=T)
attach(comp)
names(comp)}
```

The biomass of control plants is compared to the biomass of plants grown in conditions where competition was reduced in one of four different ways. There are two treatments in which the roots of neighboring plants were cut (to 5 cm or 10 cm depth) and two treatments in which the shoots of neighboring plants were clipped (25% or 50% of the neighbors were cut back to ground level).

Model simplification by stepwise deletion

```
> table(clipping, biomass)
```

```
          biomass
clipping  415 417 438 449 450 457 499 508 511 517 522 551 555 563 573 580 583 595 613 615 631 632 633 639 648 656 677 679 731
  control   1   1   1   1   0   0   0   0   0   1   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  n25       0   0   0   0   1   1   1   0   0   0   0   1   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1
  n50       0   0   0   0   0   0   0   1   0   1   0   0   0   0   0   1   1   1   0   0   0   0   1   0   0   0   0   0   0
  r10       0   0   0   0   0   0   0   0   0   0   1   0   0   1   0   0   0   0   1   0   1   0   0   0   0   1   0   1   0
  r5        0   0   0   0   0   0   0   0   1   0   0   0   0   0   1   0   0   0   0   1   0   0   0   1   1   0   1   0   0
```

Model simplification by stepwise deletion

```
model3 <- aov(biomass~clipping)
summary.lm(model3)

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    465.17      28.75  16.177  9.4e-15 ***
clippingn25     88.17      40.66   2.168  0.03987 *
clippingn50    104.17      40.66   2.562  0.01683 *
clippingr10    145.50      40.66   3.578  0.00145 **
clippingr5     145.33      40.66   3.574  0.00147 **
```


Model simplification by stepwise deletion

```
clip2 <- clipping
```

Now inspect the level numbers of the various factor level names:

```
levels(clip2)
[1] "control" "n25" "n50" "r10" "r5"

levels(clip2)[4:5] <- "root"
levels(clip2)
[1] "control" "n25" "n50" "root"
```

This is a simplification if the model.

Model simplification by stepwise deletion

```
model4 <- aov(biomass~clip2)
anova(model3,model4)
```

We can compare the models to see if the more complicated model is better than the simpler one.

```
Analysis of Variance Table

Model 1: biomass ~ clipping
Model 2: biomass ~ clip2
  Res.Df    RSS Df Sum of Sq  F Pr(>F)
1     25 124020
2     26 124020 -1 -0.083333  0 0.9968
```

We accept the null hypothesis that the simpler model is better than the more complicated model.

Model simplification by stepwise deletion

```
>summary.lm(model4)
```

Can we simplify further?

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   465.17      28.20  16.498 2.72e-15 ***
clip2n25       88.17      39.87   2.211 0.036029 *
clip2n50      104.17      39.87   2.612 0.014744 *
clip2root     145.42      34.53   4.211 0.000269 ***
```

★ We should combine these two levels, since they are not significantly different from each other.

Model simplification by stepwise deletion

```
clip3 <- clip2

levels(clip3)[2:3] <- "shoot"
levels(clip3)
[1] "control" "shoot" "root"
```

Then we fit a new model with `clip3` in place of `clip2`:

```
model5 <- aov(biomass~clip3)
anova(model4,model5)
```


Model simplification by stepwise deletion

```
model5 <- aov(biomass~clip3)
anova(model4,model5)

Analysis of Variance Table

Model 1: biomass ~ clip2
Model 2: biomass ~ clip3
  Res.Df    RSS Df Sum of Sq     F Pr(>F)
1     26 124020
2     27 124788 -1      -768 0.161 0.6915
```

We accept the null hypothesis that the simpler model is better than the more complicated model.

Model simplification by stepwise deletion

```
clip4 <- clip3

levels(clip4)[2:3] <- "pruned"
levels(clip4)
[1] "control" "pruned"
```

Now fit a new model with `clip4` in place of `clip3`:

```
model6 <- aov(biomass~clip4)
anova(model5,model6)
```


Model simplification by stepwise deletion

```
model6 <- aov(biomass~clip4)
anova(model5,model6)

    Analysis of Variance Table

    Model 1: biomass ~ clip3
    Model 2: biomass ~ clip4
      Res.Df    RSS Df Sum of Sq      F  Pr(>F)
    1     27 124788
    2     28 139342 -1    -14553 3.1489 0.08726 .
```

We accept the null hypothesis that the simpler model is better than the more complicated model.

This simplification was close to significant, but we are ruthless ($p > 0.05$), so we accept the simplification.

Model simplification by stepwise deletion

Now we have the minimal adequate model:

```
summary.lm(model6)

Call:
aov(formula = biomass ~ clip4)

Residuals:
     Min       1Q   Median       3Q      Max
-135.958  -49.667   -4.458   50.635  145.042

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    465.2       28.8  16.152 1.01e-15 ***
clip4pruned    120.8       32.2   3.751 0.000815 ***

Residual standard error: 70.54 on 28 degrees of freedom
Multiple R-squared: 0.3345,     Adjusted R-squared: 0.3107
F-statistic: 14.07 on 1 and 28 DF,  p-value: 0.0008149
```

It has just two parameters: the mean for the controls (465.2) and the difference between the control mean and the four treatment means (465.2 + 120.8 = 586.0):

Model simplification by stepwise deletion

It has just two parameters: the mean for the controls (465.2) and the difference between the control mean and the four treatment means (465.2 + 120.8 = 586.0):

```
tapply(biomass,clip4,mean)
control   pruned
465.1667 585.9583
```

Model simplification by stepwise deletion

We know that these two means are significantly different because of the *p* value of 0.000 815, but just to show how it is done, we can make a final `model7` that has no explanatory variable at all (it fits only the overall mean). This is achieved by writing *y* ~ 1 in the model formula:

```
model7 <- aov(biomass~1)
anova(model6,model7)
```

```
Analysis of Variance Table

Model 1: biomass ~ clip4
Model 2: biomass ~ 1
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     28 139342
2     29 209377 -1    -70035 14.073 0.0008149 ***
```

We reject the null hypothesis that the simpler model is better than the more complicated model.

Note that the *p* value is exactly the same as in model6.

Summary of statistical modelling

The steps in the statistical analysis of data are always the same, and should always be done in the following order:

(1) data inspection (plots and tabular summaries, identifying errors and outliers);
(2) model specification (picking an appropriate model from many possibilities);
(3) ensure that there is no pseudoreplication, or specify appropriate random effects;
(4) fit a maximal model with an appropriate error structure;
(5) model simplification (by deletion from a complex initial model);
(6) model criticism (using diagnostic plots, influence tests, etc.);
(7) repeat steps 2 to 6 as often as necessary.

Homework: Stepwise Deletion

Use the following data for this exercise:

>url2 = 'http://pages.stat.wisc.edu/~ane/st572/data/toxic.txt'
>Toxicity.Data <- read.table(url2, header=T)

A study was conducted to assess the toxic effect of a pesticide on a given species of insect. dose: dose rate of the pesticide, weight: body weight of an insect, toxicity: rate of toxic action.

Topic 4: Statistical Modeling

Homework: Stepwise Deletion

Use the following data for this exercise:

>url2 = 'http://pages.stat.wisc.edu/~ane/st572/data/toxic.txt'
>Toxicity.Data <- read.table(url2, header=T)

1. Use the lines above to attach the toxicity data to a script.
2. Consider the following models:
   a) fit1 <- $y_i = \beta_0 + e_i$
   b) fit2 <- $y_i = \beta_0 + \beta_1 \text{dose}_i + e_i$
   c) fit3 <- $y_i = \beta_0 + \beta_2 \text{weight}_i + e_i$
   d) fit4 <- $y_i = \beta_0 + \beta_1 \text{dose}_i + \beta_2 \text{weight}_i + e_i$
3. Compare them using the anova function
4. Which is the best mode? Support your answer with the statistical output.