Statistical modelling in R

JJ Valletta and TJ McKinley

22 November 2018

Contents

Preface

An introductory workshop to the field of statistical modelling in R. The focus will be on how to fit statistical models in R, rather than on the rigorous underlying mathematics. The target audience is anyone who wants to learn how to fit linear models in R. The progression will be linear models, generalised linear models and linear mixed effects models.

Prerequisites

- Programming basics in R

Learning outcomes

- Understand the key concepts and terminology used in statistical modelling
- Use R to fit linear, generalised linear and mixed effect models in R
- Recognise practical issues with fitting these models
- Checking model fit
- Perform model comparisons

Recommended reading

I highly recommend the following books:

- Statistics: An Introduction using R
- Linear models with R
- Data Analysis Using Regression and Multilevel/Hierarchical Models
- An Introduction to Statistical Learning
- Mixed Effects Models and Extensions in Ecology with R
- Extending the Linear Model with R

Data files

All data files can be downloaded as a ZIP file from here.

Acknowledgements

A big thanks to TJ McKinley for sharing a lot of material with me!

Chapter 1

Introduction

1.1 Motivation

Scientists acquire knolwedge about the state of nature (the real-world) by collecting data. For example:

	A	B	C	D	E	F	G	H	I	J	K	L	M	
1	class	BrdIndx	Area	Round	Bright	Compact	ShpIndx	Mean_G	Mean_R	Mean_NIR	SD_G	SD_R	SD_NIR	LV
2	car	1.27	91	0.97	231.38	1.39	1.47	207.92	241.74	244.48	21.41	20.4	18.69	
3	concrete	2.36	241	1.56	216.15	2.46	2.51	187.85	229.39	231.2	6.57	6.97	7.02	
4	concrete	2.12	266	1.47	232.18	2.07	2.21	206.54	244.22	245.79	6.16	4.93	5.53	
5	concrete	2.42	399	1.28	230.4	2.49	2.73	204.6	243.27	243.32	5.76	5.56	5.46	
6	concrete	2.15	944	1.73	193.18	2.28	4.1	165.98	205.55	208	11.46	8.9	9.77	
7	tree	3.11	169	1.47	172.22	2.49	3.35	240.18	127.65	148.83	8.41	10.34	11.5	
8	car	1.2	44	0.79	208.8	1.14	1.36	180.95	221.61	223.82	35.42	36.45	35.17	
9	car	1	88	0.22	234.51	1.11	1.12	208.5	246.48	248.56	12.3	11.29	8.81	
10	building	1.59	1737	0.67	219.61	1.3	1.64	185.86	233.84	239.13	7.08	7.03	7.28	
11	tree	2.37	153	1.3	120.24	2.85	2.59	184.15	81.5	95.06	10.21	11.99	11.3	
12	building	1.57	3552	0.46	213.22	1.32	1.6	173.03	229.84	236.8	5.5	5.54	5.63	
13	asphalt	4.19	418	2.48	83.35	4.21	4.3	68.07	88.8	93.17	6.51	6.3	6.12	
14	building	1.3	1024	0.58	183.93	1.39	1.64	169.57	209.61	172.6	6.15	6.37	6.8	
15	grass	1.14	289	0.38	173.16	1.21	1.21	213.71	145.56	160.23	10.3	11.55	11.1	
16	shadow	2.03	249	1	39.62	2.2	2.12	35.86	38.92	44.07	7.72	6.36	6.87	
17	building	1.31	1639	0.68	205.31	1.26	1.37	196.24	239.05	180.65	7.15	7.82	8.37	
18	tree	2.68	285	1.48	138.01	2.53	2.7	201.84	98.06	114.13	15.24	15.1	14.31	
19	soil	2.79	293	1.37	237.11	2.4	3.1	227.22	242.45	241.68	7.9	9	9.86	
20	building	1.21	2797	0.78	244.7	1.34	1.23	229.52	252.21	252.37	7.5	4.98	4.42	
21	shadow	1.13	217	0.32	41.25	1.22	1.22	47.51	34.69	41.55	8.79	8.29	7.77	
22	pool	1.28	173	0.85	157.34	1.53	1.41	85.43	161.11	225.49	14.8	12.18	16.76	
23	shadow	3.23	203	2.15	66.75	3.55	3.4	60.3	68.47	71.47	14.52	16.39	14.8	
24	concrete	2.09	178	1.63	215.76	2.4	2.66	188.69	229.03	229.55	5.25	4.95	5.01	
25	tree	2	138	1.31	154.97	2.46	2.04	200.16	124.49	140.25	13.52	13	12.97	
26	grass	1.79	203	1.83	152.09	2.41	2.46	193.34	123.78	139.15	14.03	13.18	10.17	
27	concrete	2	792	1	238.34	1.88	2.24	211.08	251.96	251.97	6.43	3.73	3.78	
28	grass	3.54	429	1.79	176.66	2.73	3.93	205.01	160.12	164.84	7.52	7.29	7.02	
29	building	1.38	1078	0.64	243.74	1.27	1.43	226.39	252.62	252.21	7.17	3.78	4.43	
30	building	1.92	957	0.96	175.33	1.76	2.05	161.9	201.82	162.26	7.79	7.24	6.78	
31	building	2.74	316	1.92	200.53	2.73	3.32	181.09	208.44	212.05	7.5	8.4	8.22	
32	asphalt	2.94	642	1.43	84.16	2.56	3.02	67.13	90.65	94.7	7.06	7.08	7.15	
33	shadow	1.39	220	0.86	49.89	1.43	1.79	61.68	40.14	47.87	9.51	7.03	7.09	
34	building	1.48	3084	0.93	230.71	1.33	2.52	215.62	252.64	223.88	7.04	3.65	7.07	
35	grass	2.44	554	1.72	146.12	3.03	2.49	213.73	102.01	122.61	8.67	6.99	7.24	
36	tree	2.89	288	1.41	124.98	2.4	2.98	187.27	88.61	99.07	15.77	13.45	13.66	
37	building	1.4	1278	0.73	226.77	1.42	2.06	204.75	251.55	224.01	7.03	4.58	7.13	
38	asphalt	2.66	998	1.34	71.58	2.35	2.82	57.98	75.54	81.23	8.51	8.88	8.99	
39	tree	1.66	328	0.9	237.5	1.73	1.74	211.47	250.57	250.48	6.5	4.95	4.99	
40	tree	3.2	373	1.61	148.7	2.78	3.73	213.41	106.93	125.76	13.31	10.07	10.96	
41	concrete	1.41	1046	0.76	239.94	1.37	1.9	215.11	252.47	252.25	6.2	3.72	3.71	

How do we make sense of all of these numbers and text? Visualising the data can give us important insights into the underlying structure of the data (covered in a previous workshop). The next step is to develop a **model** that captures the salient features/structure of the data.

1.2 What is a model?

> A **model** is a human construct/abstraction that tries to approximate the **data generating process** in some useful manner.

Some people go even further:

> Raw data, no matter how extensive, are useless without a model - **Nate Silver**

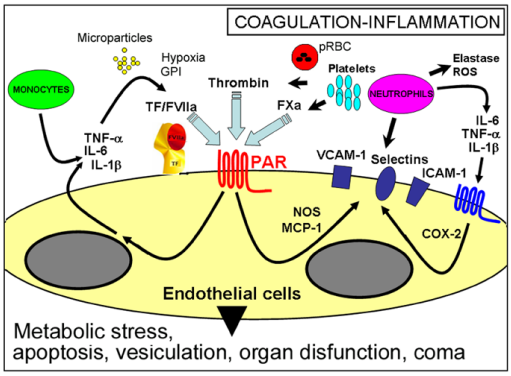
Models are built for various reasons:

- to enhance understanding of a complex phenomenon (e.g how does antibiotic resistance develop?);
- to execute "what if" scenarios (e.g what happens if interest rates go up?);
- to predict/forecast an outcome (e.g how many people will get influenza next winter?);
- to control a system (e.g autonomous vehicles).

Models come in all kinds of "languages":

- a physical model e.g medical training models

- a verbal/pictorial model e.g Francischetti *et al. Microcirculation* (2008)



- a mathematical model

$$\begin{aligned}\frac{\mathrm{d}x}{\mathrm{d}t} &= \sigma(y-x),\\ \frac{\mathrm{d}y}{\mathrm{d}t} &= x(\rho-z)-y\\ \frac{\mathrm{d}z}{\mathrm{d}t} &= xy-\beta z\end{aligned}$$

Assumptions are an inherent part of every model; we cannot build models if we are not prepared to make assumptions. The feasibility of these assumptions is context-dependent but also on how the model will be used.

Remember that models do **not** represent the real-world, they are merely an "abridged" version of the real-world with many (many) caveats!!

> All models are wrong but some are useful - **George E.P. Box**
>
> For every complex question there is a simple and wrong solution - **Albert Einstein**

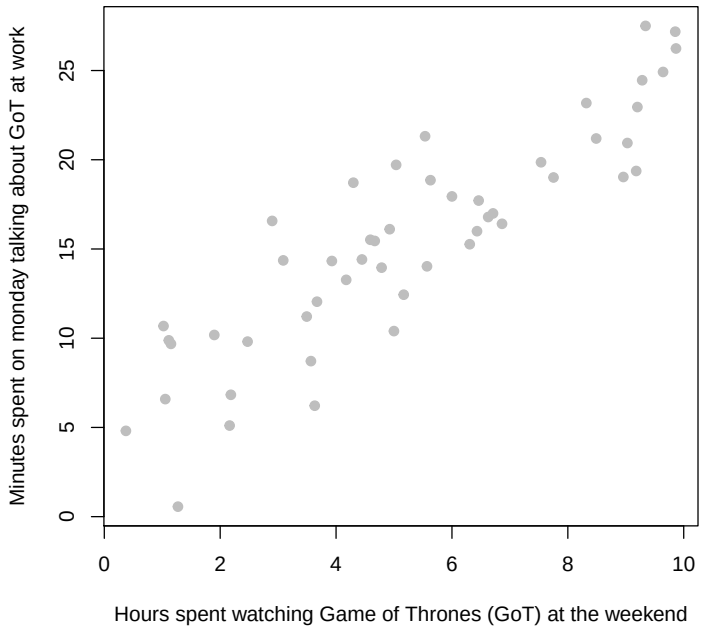
1.3 What is a statistical (stochastic) model?

> A **statistical model** is a mathematical model that makes a set of statistical assumptions in the form of probability distributions (i.e we assume that some variables follow a pre-defined probability distribution).

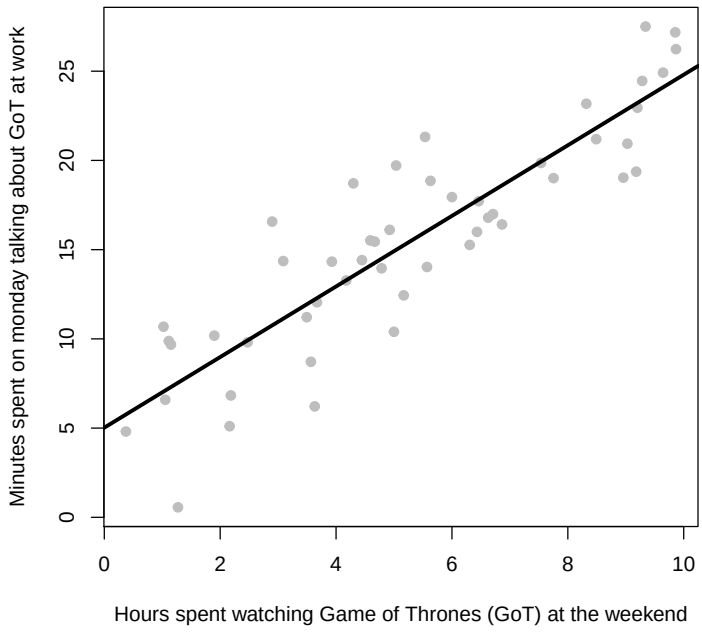
In this workshop we will solely focus on **linear** models. In their simplest form these models can be used to fit "straight lines" (or hyperplanes; "straight lines" in higher dimensions) through data points. However, this modelling framework is much more general, and can model relationships between continuous and categorical variables, and can also be used to fit curved relationships. So it provides a much richer class of models than it first appears (though we will only focus on simple cases in this workshop.)

1.4 Illustrative example

Remember when you used to run simple lab experiments at school and plot data onto graph paper? It used to look something like this (with different axes labels of course):

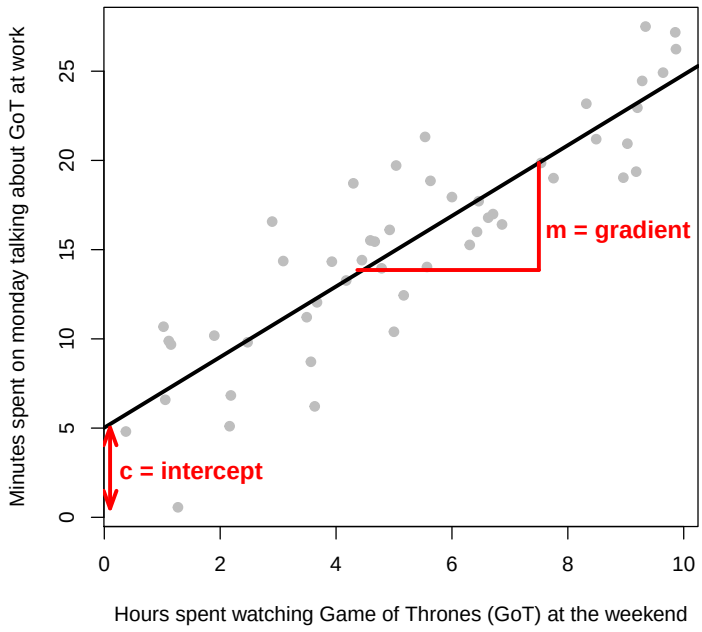


Once you plotted out all of the data points, you were asked to grab a ruler (a rigid, straight object) and find the line of best fit. To do this, you used to try and get a roughly equal number of data points on each side of the line.

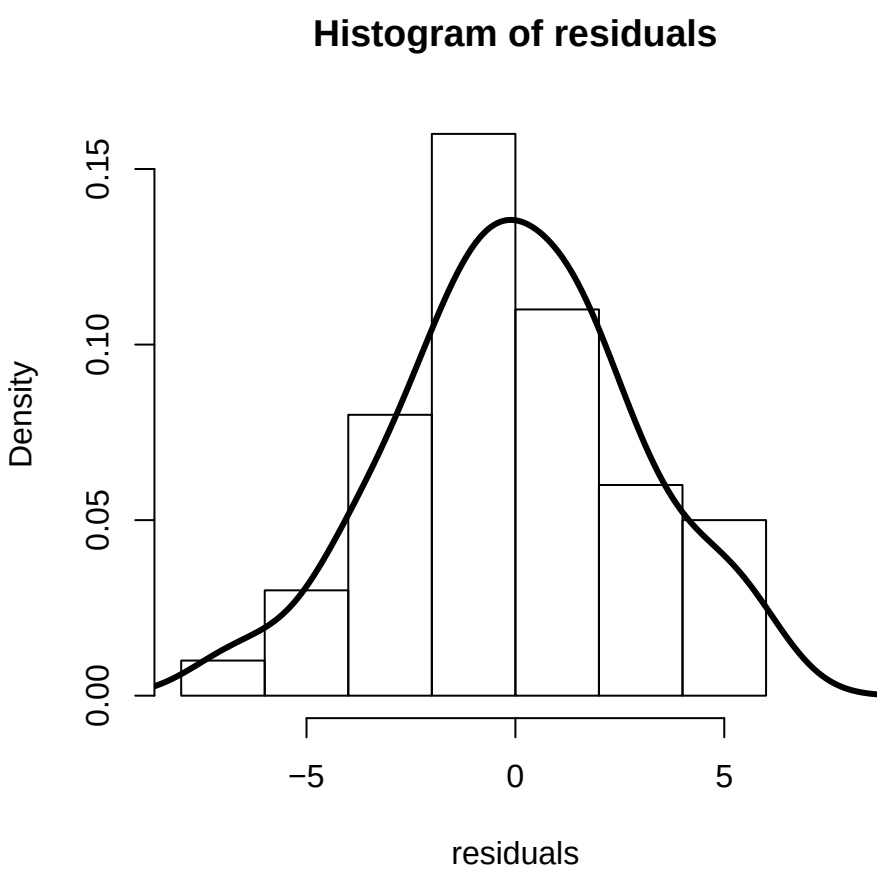


Congratulations! Unknowingly, you were fitting a statistical model to your measured data! Your best fit in that case was equivalent to performing **ordinary least-squares** which minimises the distance between the model (straight line) and your data points.

The fitted linear model is of the form $y = mx + c$. Where m is the gradient and c is the intercept (x is the dependent variable and y the response variable i.e your measured data). At school you were inferring the parameters m and c by hand. In this workshop you will learn how to do it in R.



What are the statistical assumptions in this case? As we will see in the next chapter, in linear regression/modelling we assume that the **residuals** i.e the differences between the model (black line) and "reality" (observed data points), follow a normal/Gaussian distribution (a bell curve).



Model checking involves confirming that our statistical assumptions are sensible. We will delve into this in much detail later on, but just remember that when you hear people saying "check that your data is normal", in the context of linear regression, what this means is that the **residuals** are normal (which also means that the **response** variable is normal).

1.5 Summary

- Models are useful abstractions of the data that can help us understand the underlying system.
- Linear regression is simply a flexible way to put a line-of-best-fit (or hyperplane) through the observed data points.
- Model checking is the process of verifying the statistical assumptions made.

Chapter 2

Linear models

Slides can be downloaded from:

- LMs in R

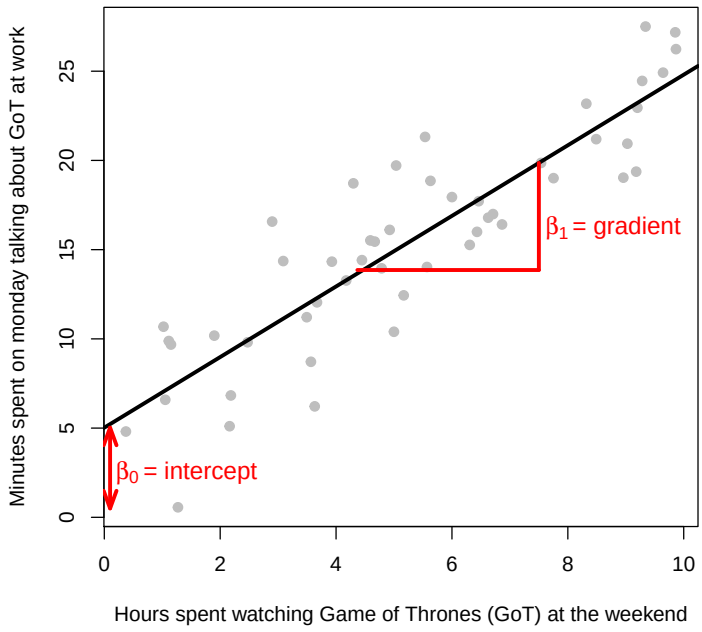
2.1 Simple linear regression

In many scientific applications we are interested in exploring the relationship between a single **response** variable and multiple **explanatory** variables (predictors). We can do this by fitting a linear model. Linear models *per se* do not infer *causality*, i.e defining a variable as response or explanatory is somewhat arbitrary and depends on what the researcher is interested in. *Causality* can however be inferred through careful experimental design in a well-controlled setting.

Consider again the case of one response and one explanatory variable. From the previous chapter we know that this boils down to the equation of a line ($y = mx + c$). Let's rename the parameters c and m to β_0 and β_1, in line with statistical naming convention[1]. It is just a change in name:

$$y = c + mx$$
$$y = \beta_0 + \beta_1 x$$

[1] this notation is such that we can have any arbitrary large number of explantory variables i.e β_1, β_2, $\beta_3 \ldots$ etc., without running out of letters in the alphabet!



Now suppose we measure n independent and identically distributed (i.i.d) normally distributed outcomes $y_1, \ldots, y_n$ (e.g height of people) and we want to model their relationship with some explanatory variable x (e.g weight of people). The linear regression model is defined as follows:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$
$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

i is an index that goes from 1 to n (the total number of observations). The equation is the same as before ($y = \beta_0 + \beta_1 x$), but now we have added an error term ϵ. This term is needed because the straight line cannot go through *all* the data points (unless you have a very questionable dataset)! It represents the discrepancy between the model (the fitted straight line) and the observed data (grey points).

ϵ is also known as the **noise** term. The reason is that in general our response variable has some uncertainty associated with it (e.g counting the number of birds in an area, quantifying gene expression using microarrays or RNA-seq, etc.). The statistical assumption in linear modelling is that these errors are normally distributed with mean zero and some standard deviation σ (mathematically this is written as $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$). This means that the **response** variable is also assumed to be normally distributed. For the maths aficionados amongst you:

$$y_i \sim \mathcal{N}(\beta_0 + \beta_1 x_i, \sigma^2)$$

Note that we do not make explicit assumptions about the **explanatory** variables (the x's) i.e they don't need to be normal.

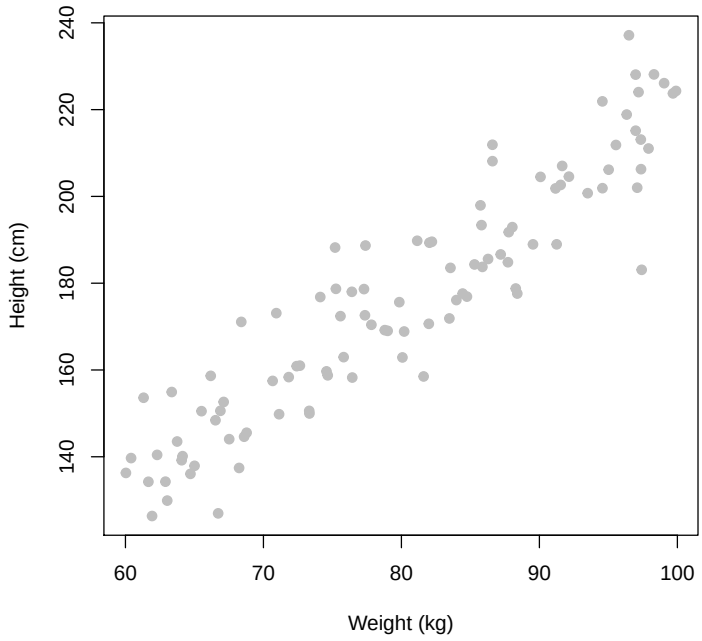
The workflow in linear regression is as follows:

1. Infer the model's parameters β_0 and β_1.
2. Check the model fit.
3. Interpret the **practical** significance of the estimated parameters.

2.2 Doing it in R

Luckily for us, we do not need to worry about the mathematical intricacies of estimating the model's parameters, R will do it for us. Let's generate some fake data just to get things started.

```
set.seed(1453)
N <- 100 ## no. of observations
weight <- runif(n=N, min=60, max=100) ## hypothetical weights in kg
height <- 2.2*weight + rnorm(n=N, mean=0, sd=10) ## hypothetical heights in cm
plot(weight, height, pch=19, xlab='Weight (kg)', ylab='Height (cm)', col='grey')
```



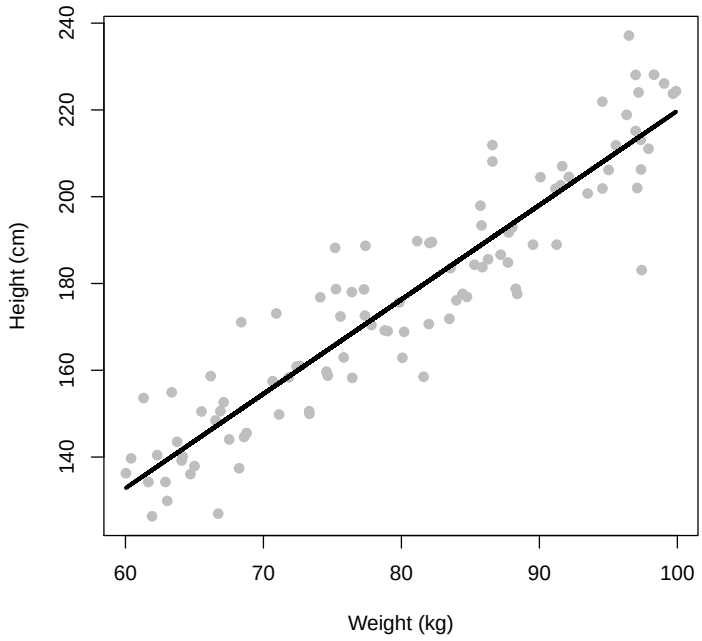
To fit a linear model we use the `lm()` function (**always** read the documentation of a function before using it!). This function requires a **formula** object which has the form of `response ~ explanatory`. So in our case this will be `height ~ weight`. R will then fit the following model [2]:

$$\text{height}_i = \beta_0 + \beta_1 \text{weight}_i + \epsilon_i$$

Let's call the R function, plot the model fit and print the output.

```
## Linear model fit
fit <- lm(height ~ weight)

## Plot model fit
plot(weight, height, pch=19, xlab='Weight (kg)', ylab='Height (cm)', col='grey')
lines(weight, predict(fit), col='black', lwd=3)
```

[2] weight is our explanatory variable (x_i) and height the response variable (y_i)




```
## Print result
print(fit)
```

```
##
## Call:
## lm(formula = height ~ weight)
##
## Coefficients:
## (Intercept)       weight
##       2.352        2.174
```

This outputs two numbers, the `(Intercept)=` 2.352 cm and `weight=` 2.174 cm/kg. These are the β_0 and β_1 parameters.

The β_0 parameter (intercept) is not very useful in this case, it basically tells us what's the expected height of someone that weighs 0 kg (2.352 cm here). This is of course nonsense, but I'm highlighting it as a reminder that the assumptions that underlie these models can only be tested within the range of the observed data.

> **Aside**: using the model to predict outside of the range of the observed data is called **extrapolating**. Oftentimes, statistical models are developed *in order* to be used to extrapolate (e.g. climate modelling). This is however, dangerous, as we can only assess the assumptions of the model over the range of the observed data. When extrapolating we have to assume that these relationships hold beyond the range of the data, which may or may not be reasonable (hence why weather forecast over short-time periods are OK, but climate forecasts are much more uncertain). Hence, we should always view the model as an approximation of the data generating process. In this particualr case, the interpretation of the parameters is not sensible when $x = 0$ (weight = 0 kg), but makes sense in the range that we are interested in exploring. If the case where $x = 0$ is important, then we would have to change the model to ensure that the predictions made sense at those values of x.

The β_1 parameter (gradient) tells us about the relationship between the outcome and explanatory variable. In this case, for every 1 kg increase in weight on *average* the height increases by 2.174 cm.

Warning: the notation used by R can come across as confusing. Although the returned value is labelled as `weight`, in fact it corresponds to the β_1 parameter that relates to the strength of linear relationship between `weight` and `height` (i.e it is a gradient).

2.2.1 Using data frames

The `lm()` function also accepts **data frames** as input arguments.

```
## Create data frame
df <- data.frame(height=height, weight=weight)
head(df)
```

```
##     height   weight
## 1 211.9054 86.59694
## 2 178.6420 77.28891
## 3 206.1585 95.01870
## 4 157.4803 70.66759
## 5 175.6296 79.84837
## 6 176.1113 83.98290
```

```
## Fit linear model
fit <- lm(height ~ weight, data=df)
```

2.2.2 Extended summary

The object returned by `lm()` contains further information that we can display using the `summary()` function.

```
summary(fit)
```

```
##
## Call:
## lm(formula = height ~ weight, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.089  -6.926  -0.689   6.057  24.967
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.35229    7.11668   0.331    0.742
## weight       2.17446    0.08782  24.762   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.31 on 98 degrees of freedom
## Multiple R-squared:  0.8622, Adjusted R-squared:  0.8608
## F-statistic: 613.1 on 1 and 98 DF,  p-value: < 2.2e-16
```

That's a lot of information, let's unpick it line by line:

- **Call**

  This just states the arguments that were passed to the `lm()` function. Remember it's `response ~ explanatory`.

- **Residuals**

  Some basic stats about the residuals (i.e the differences between the model fit and the observed data points). It is easier to plot a histogram of the residuals (shown in the next section), but these basic stats can already give us an indication of whether we have a symmetric distribution with zero mean (i.e we want the median to be close to zero, the third quartile (3Q) to be roughly equal to -1Q (first quartile) and the max to be approximately -min).

- **Coefficients**

  - **`Estimate`**

  The `(Intercept)`= 2.352 cm and `weight`= 2.174 cm/kg are the β_0 and β_1 parameters as discussed earlier.

  - **`Std. Error`**

  The standard error for that parameter. It tells us how confident we are in estimating that particular parameter. If the standard error is comparable or greater than the actual parameter estimate itself then that point estimate should not be trusted. We can also show the confidence intervals for the model parameters to highlight their uncertainty using the `confint()` function:

```
confint(fit, level=0.97) ## pick the 97% confidence intervals
```

```
##                   1.5 %    98.5 %
## (Intercept) -13.319698 18.024268
## weight        1.981077  2.367844
```

  - **`t value`** and **`Pr(>|t|)`**

  This is the result of a hypothesis testing against the null hypothesis that the coefficient is zero.

- **Residual standard error**

  The square root of residual sum of squares/degress of freedom

```
## Residual standard error
sqrt(sum(residuals(fit)^2) / fit$df.residual)
```

```
## [1] 10.31254
```

- **Multiple R-squared**

  Total variation = **Regression** (explained) variation + **Residual** (unexplained) variation

  The R^2 statistic (also known as the **coefficient of determination**) is the proportion of the total variation that is explained by the regression. In regression with a single explanatory variable, this is the same as the Pearson correlation coefficient squared:

```
cor(height, weight)^2
```

```
## [1] 0.8621921
```

- **F-statistic**

$$\text{F-statistic} = \frac{\text{Regression (explained) variation}}{\text{Residual (unexplained) variation}}$$

  The F-statistic can be used to assess whether the amount of variation explained by the regression (M_1) is *statistically significantly* different compared to the **null** model (M_0 - in this case the **null** model is the same as just taking the mean of the data). Large values of the F-statistic correspond to cases where the model fit is better for the more complex model compared to the null model. This test can be

used to generate a p-value to assess whether the model fit is statistically significantly better given a pre-defined level of significance.

$$M_0 : \quad y_i = \beta_0 + \epsilon_i$$
$$M_1 : \quad y_i = \beta_0 + \beta_1 W_i + \epsilon_i$$

2.3 Model checking

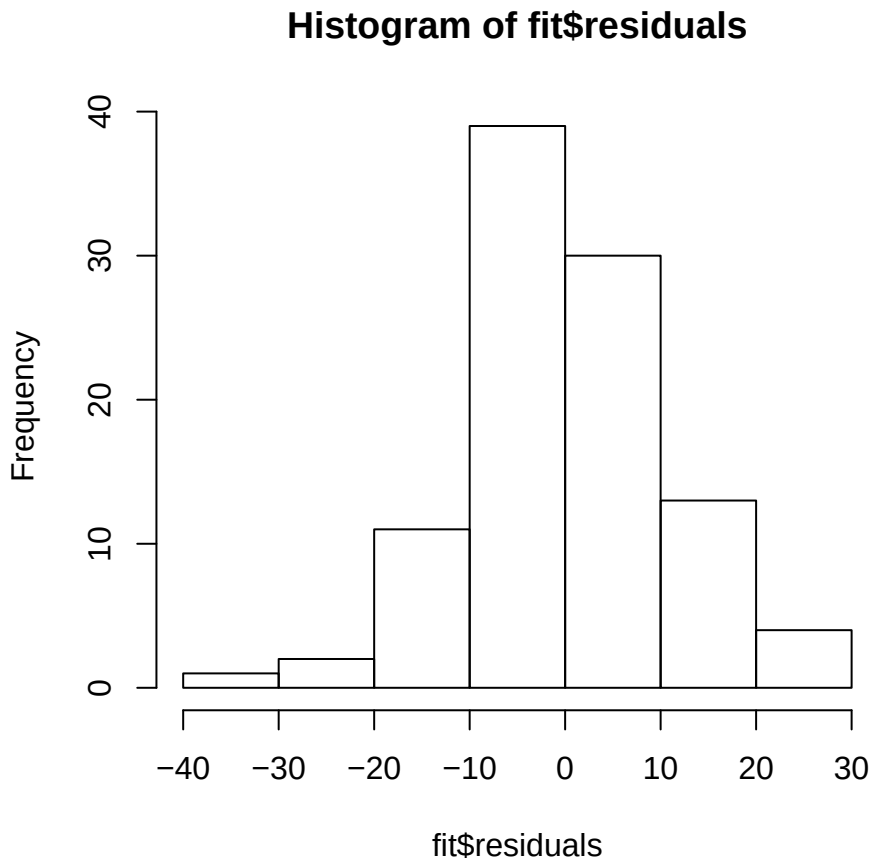
The extended summary leads us nicely to the concept of model checking. In theory, we can fit an infinite number of different models to the same data by placing different assumptions/constraints. Recall:

> All models are wrong but some are useful - **George E.P. Box**

In order to make **robust** inference, we must **check** the model fit

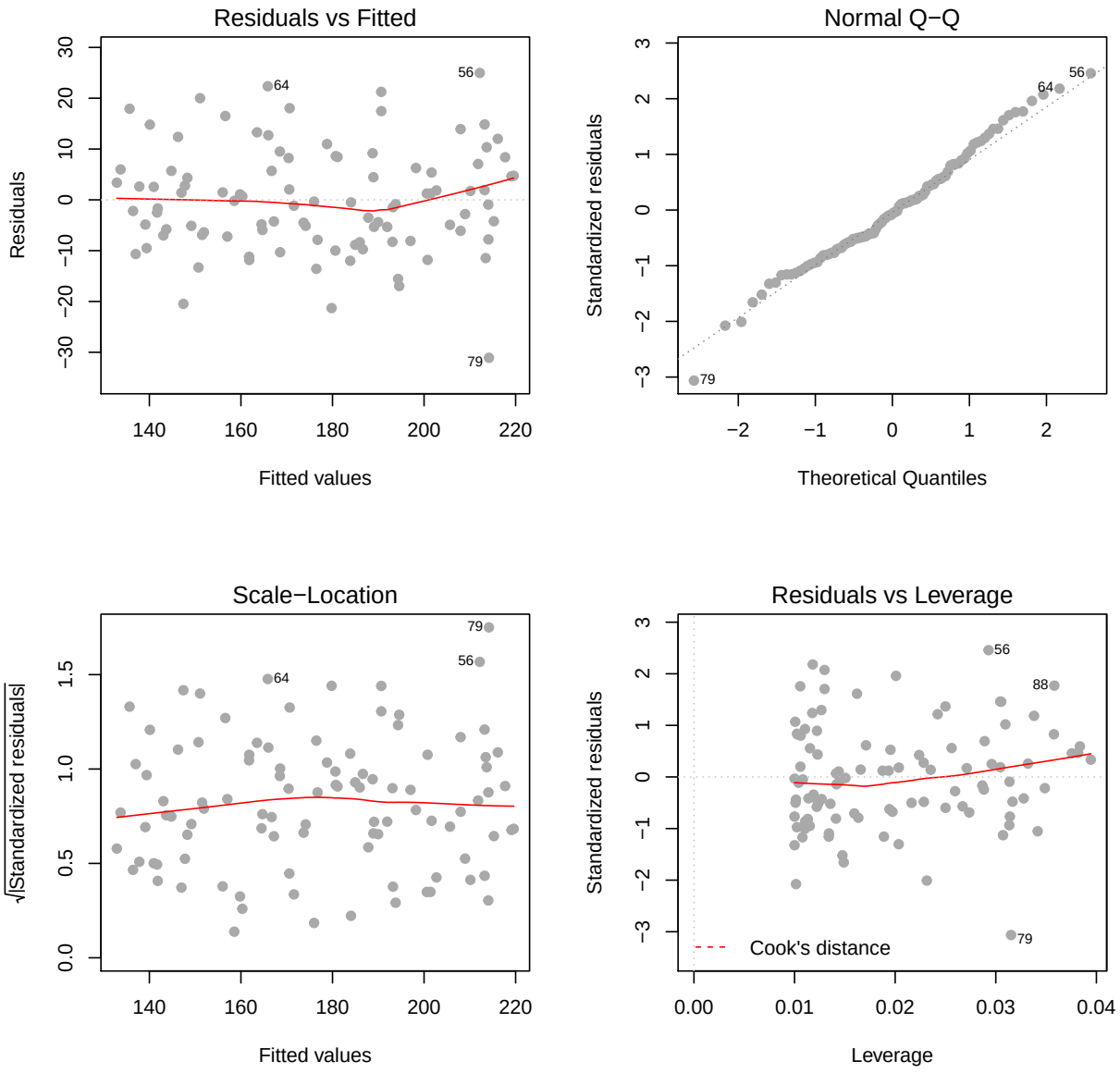
Model checking boils down to confirming whether or not the assumptions that we have placed on the model are reasonable. Our main assumption is that the residuals are normally distributed centred around zero. Let's plot this:

```
hist(fit$residuals)
```



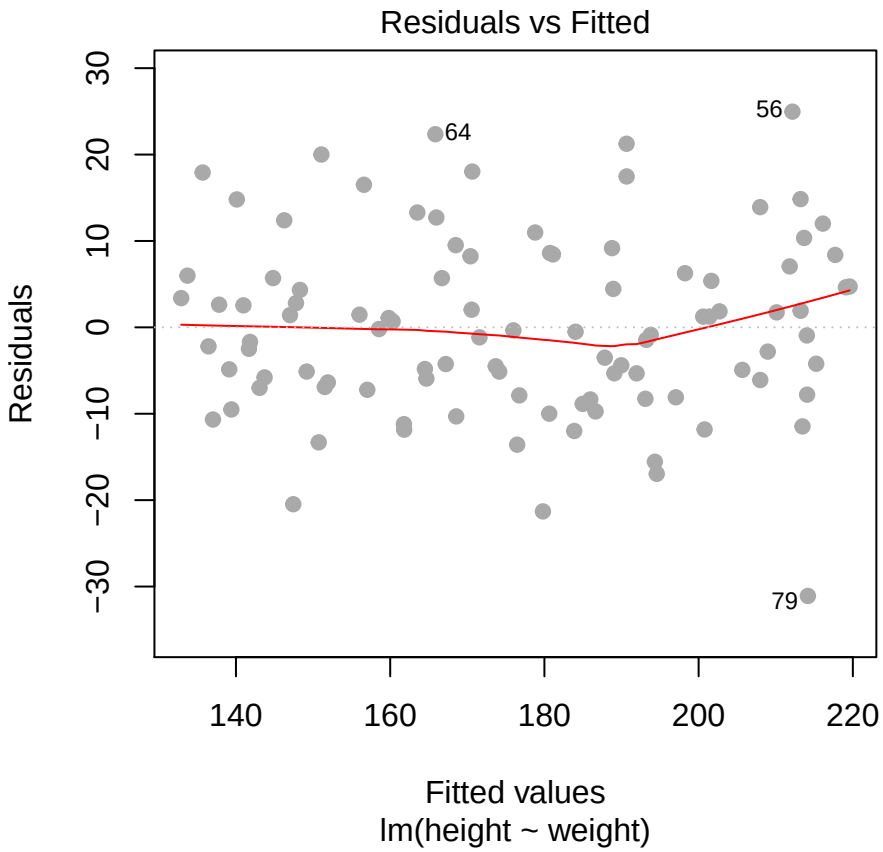
The residuals are fairly normally distributed which is a good sign. R also provides us with the following diagnostic plots:

```
par(mfrow=c(2, 2))
plot(fit, pch=19, col='darkgrey')
```



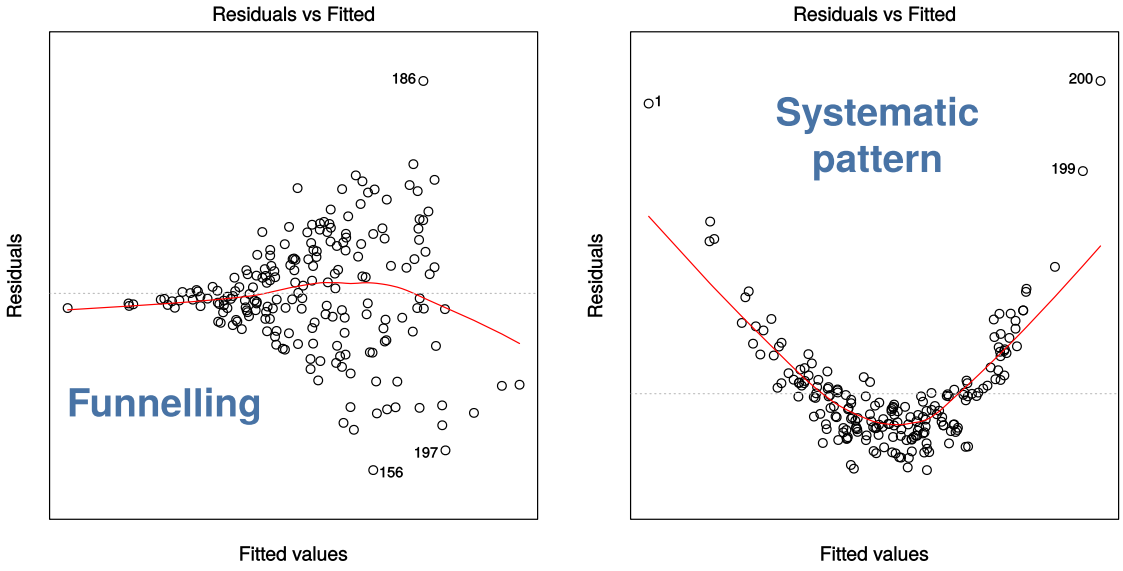
2.3.1 Residuals vs fitted values

```
plot(fit, pch=19, col='darkgrey', which=1)
```



Here we are checking that the variance is constant along the fitted line[3], and that there are no systematic patterns in the residuals.

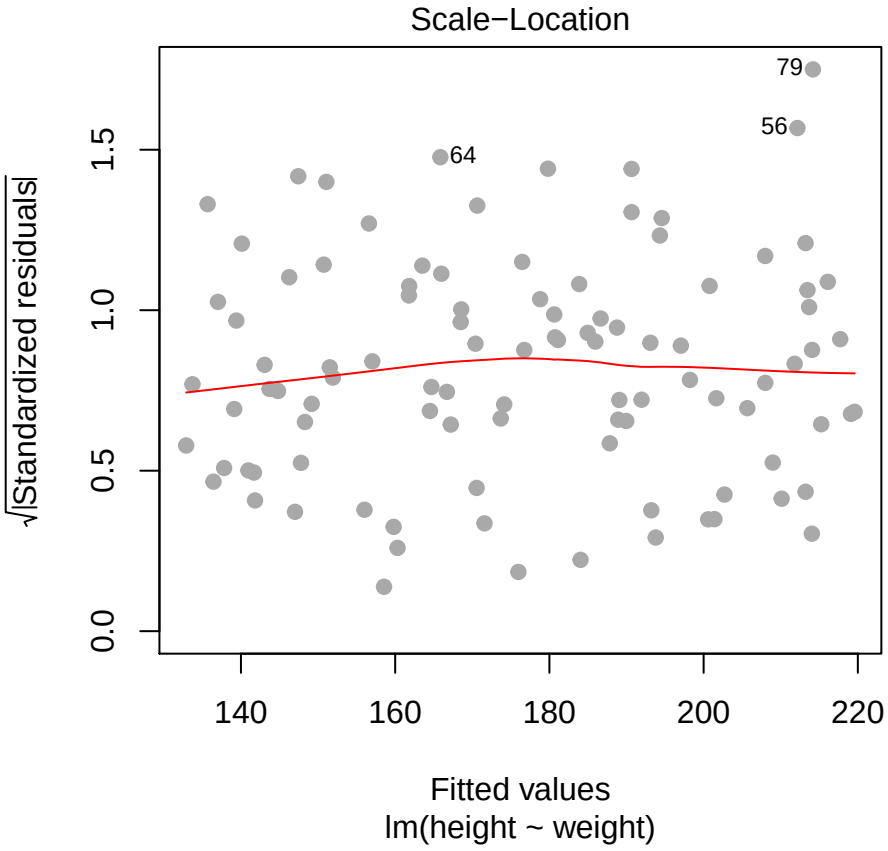
A couple of example where this assumption is violated.



[3]non-constant variance is known as **heteroscedacity**

2.3.2 Residuals vs fitted values (scale-location)

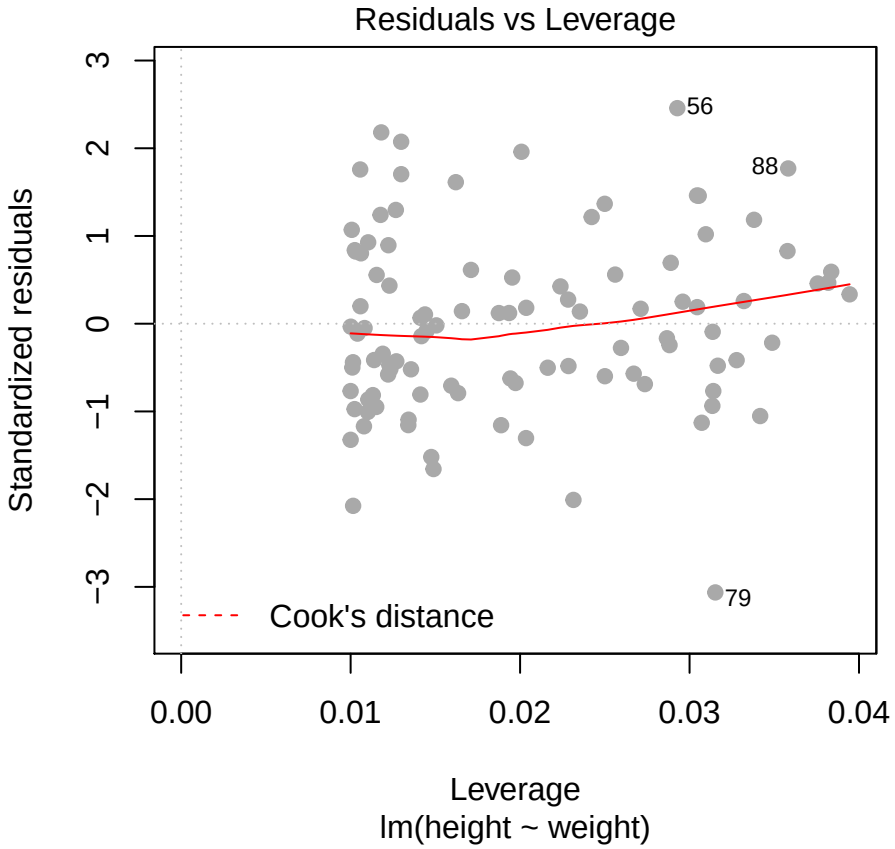
```
plot(fit, pch=19, col='darkgrey', which=3)
```



This is similar to the first plot but on a different scale.

2.3.3 Residuals vs. leverage

```
plot(fit, pch=19, col='darkgrey', which=5)
```

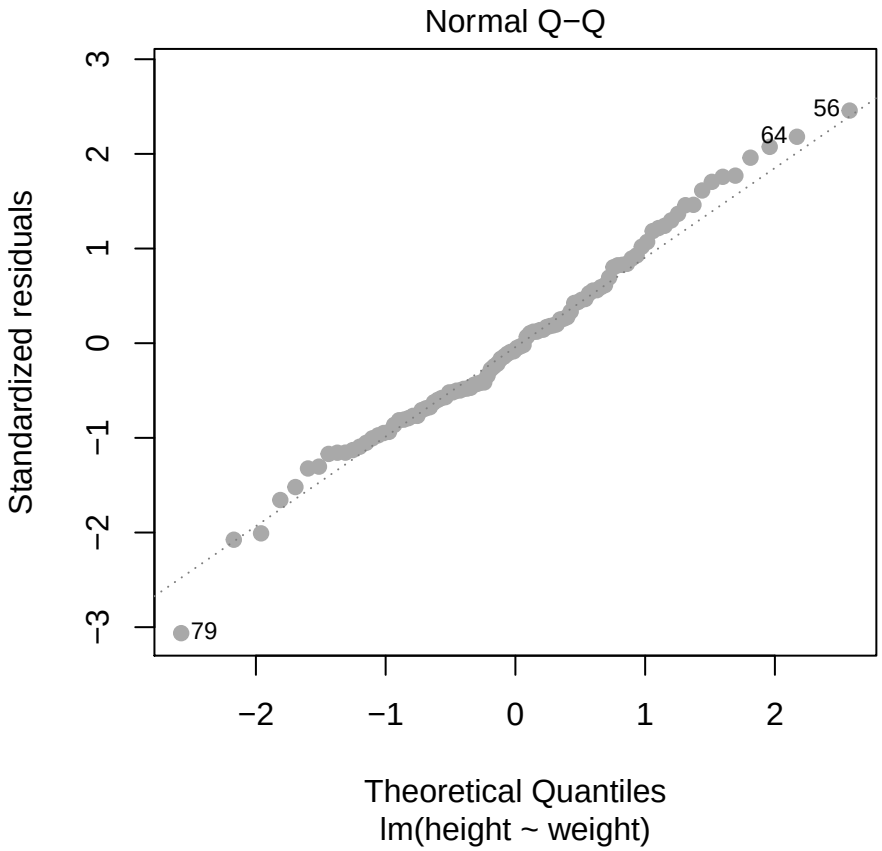


- **Leverage**: a measure of how isolated individual points are in relation to other points
- **Cook's Distance**: a measure of how influential a point is to the regression

These measures help us identify potential outliers.

2.3.4 QQ plots

```
plot(fit, pch=19, col='darkgrey', which=2)
```



Here we are checking that the assumption of normally distributed errors is reasonable. Points should follow the dashed line.

2.4 Practical 1

We will use the fruitfly dataset (Partridge and Farquhar (1981)) introduced in the "Advanced Visualisation and Data Wrangling in R", which is summarised again here (do not worry about the details of the study for now, this is only included for the sake of completeness):

A cost of increased reproduction in terms of reduced longevity has been shown for female fruitflies, but not for males. We have data from an experiment that used a factorial design to assess whether increased sexual activity affected the lifespan of male fruitflies.

The flies used were an outbred stock. Sexual activity was manipulated by supplying individual males with one or eight receptive virgin females per day. The longevity of these males was compared with that of two control types. The first control consisted of two sets of individual males kept with one or eight newly inseminated females. Newly inseminated females will not usually remate for at least two days, and thus served as a control for any effect of competition with the male for food or space. The second control was a set of individual

males kept with no females. There were 25 males in each of the five groups, which were treated identically in number of anaesthetisations (using CO_2) and provision of fresh food medium.

Download the data file from here and save it to your working directory.

Since we are working with `data.frame()` objects, where appropriate, we have provided examples using both base R or `tidyverse` (for those of you who attended the Advanced Visualisation and Data Wrangling workshop).

```
ff <- readRDS("fruitfly.rds")
```

```
head(ff)
```

```
##   partners        type longevity thorax sleep
## 1        8 Inseminated        35   0.64    22
## 2        8 Inseminated        37   0.68     9
## 3        8 Inseminated        49   0.68    49
## 4        8 Inseminated        46   0.72     1
## 5        8 Inseminated        63   0.72    23
## 6        8 Inseminated        39   0.76    83
```

For the purpose of this practical we will assume that the no female case is the control case, whilst the inseminated and virgin female cases are the treatment cases.

- **partners**: number of companions (0, 1 or 8)
- **type**: type of companion (inseminated female; virgin female; control (when partners = 0))
- **longevity**: lifespan, in days
- **thorax**: length of thorax, in mm
- **sleep**: percentage of each day spent sleeping

Task 1

Produce a scatterplot of `longevity` against `thorax`. What does the relationship look like?

Show Solution on P101

Task 2

1. Fit a linear model with lifespan as response variable and thorax length as explanatory variable.
2. Display a summary of the fit, together with the 97% confidence interval for the estimated parameters.
3. Show the diagnostic plots for the model.

Show Solution on P101

2.5 Prediction

One of the key benefits of fitting a statistical model is that we can use it to produce predictions of the response variable for new values of the explanatory variable(s). We can do this using the `predict()` function in R. For example, in the fruitflies example above, we may want to produce estimates of average `longevity` for given values of `thorax`. To do this we must create a new `data.frame` object containing all the values of **explanatory** variables that we want to use for our prediction.

> **Note**: we only require the ***explanatory*** variables in this new data frame, the ***response*** variable will be generated from the model.

For example, if we wanted to produce an estimate of `longevity` for an average individual with a `thorax` length of 0.8mm, we can run:

```
## produce predicted longevity for thorax = 0.8mm
newdata <- data.frame(thorax = 0.8)
predict(fit, newdata)
```

```
##        1
## 54.41478
```

We can also use this to produce **confidence** or **prediction** intervals as follows:

```
## produce predicted longevity for thorax = 0.8mm
newdata <- data.frame(thorax = 0.8)
predict(fit, newdata, interval = "confidence", level = 0.97)
```

```
##        fit      lwr      upr
## 1 54.41478 51.64686 57.18269
```

> **Note**: If you predict *without* an `interval` argument, the `predict()` function returns a `vector`, otherwise it returns a `matrix`.

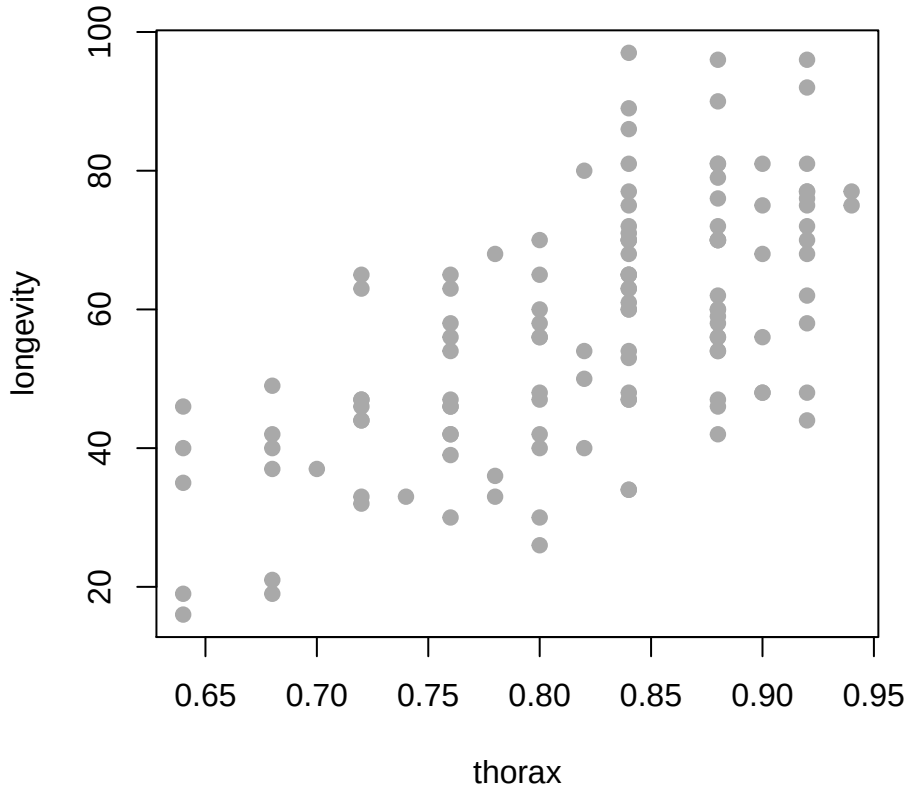
Aside: there are two types of intervals that you might want to produce: **confidence** or **prediction** intervals.

- **Confidence** intervals: these correspond to the uncertainty surrounding our estimate of an **average** individual (i.e. it represents the uncertainty in the **mean**: $y = \beta_0 + \beta_1 x$
- **Prediction** intervals: these correspond to the uncertainty surrounding an **individual** observation: $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$.

If the model fits well, then we would expect $100(1-\alpha)\%$ of the individual measurements to lie within the **prediction** interval, where α is the significance level (so $\alpha = 0.03$ corresponds to a 97% confidence interval).

The `predict()` function can be used as a very general way to produce plots of the fitted values over the observed data. For example, let's consider that we wish to add our fitted regression line to our observed data plot.

```
plot(longevity ~ thorax, data = ff, pch = 19, col='darkgrey')
```



There are various ways to add the correct regression line to this plot, but we will show you a general way that can be used for lots of different types of models. Firstly, we generate a range of x-coordinates that we wish to predict to e.g.

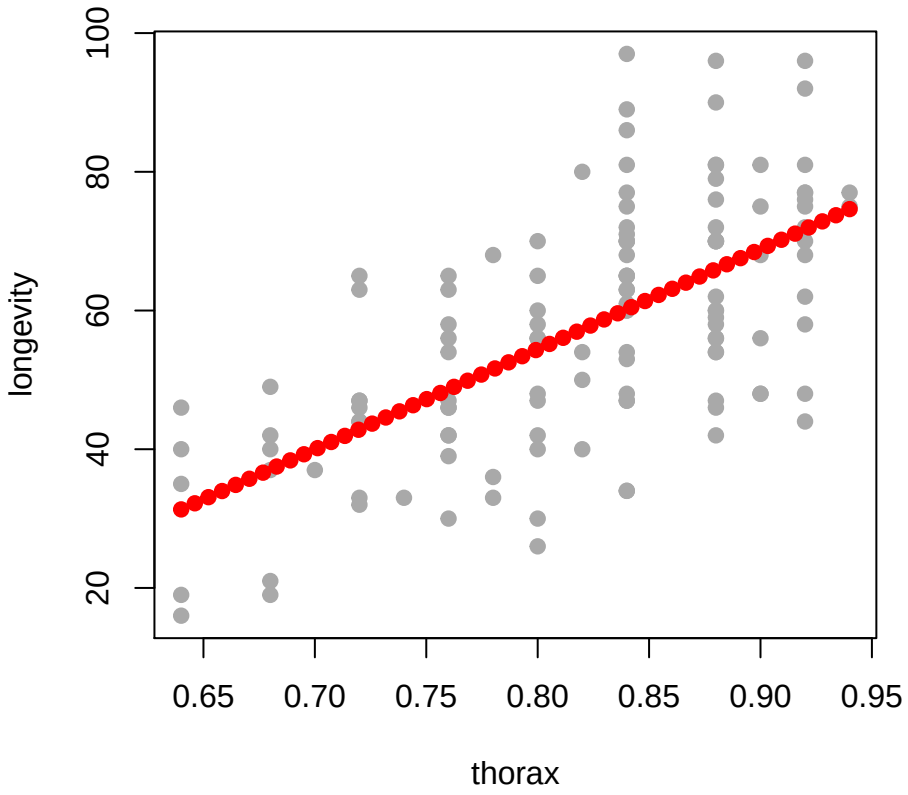
```
## create new data frame to predict to
newdata <- data.frame(thorax=seq(min(ff$thorax), max(ff$thorax), length.out=50))
```

We then have to use the model to predict the mean `longevity` at each of these `thorax` values:

```
## predict longevity form the model
newdata <- cbind(newdata, longevity=predict(fit, newdata))
```

Just as an illustration we will overlay these predictions onto the original scatterplot as points:

```
## add predicted points to original plot
plot(longevity ~ thorax, data = ff, pch = 19, col='darkgrey')
points(longevity ~ thorax, data = newdata, pch = 19, col = "red")
```

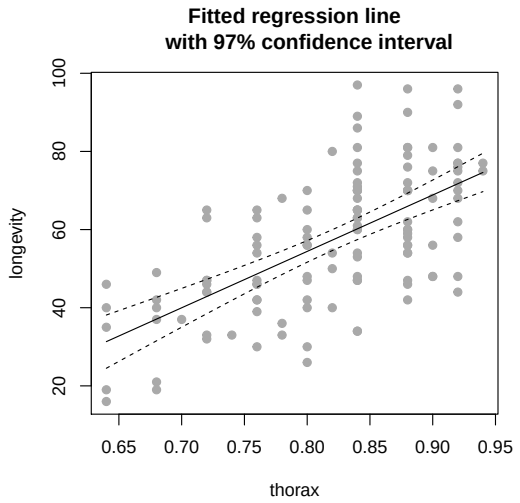


So we have used the `predict()` function to produce predicted coordinates that we can overlay on the original scatterplot. In practice we would usually plot these predictions as a **line** rather than points. This approach is particularly useful if the predictions that we wish to plot are not straight lines (such as when we generate **confidence** or **prediction** intervals). To complete this example we will now plot the fitted line and associated confidence interval as follows:

```
## create new data frame to predict to
newdata <- data.frame(thorax=seq(min(ff$thorax), max(ff$thorax), length.out=50))

## produce predictions and intervals
newdata <- cbind(newdata, predict(fit, newdata, interval="confidence", level=0.97))
newdata$longevity <- newdata$fit
newdata$fit <- NULL
```

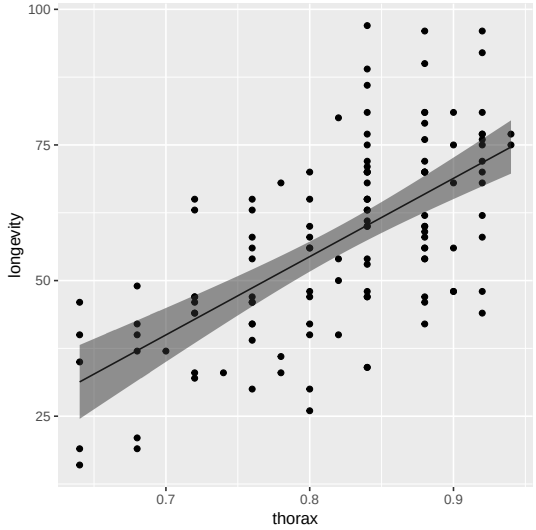
Base R

```
## plot fitted line against the raw data
plot(longevity ~ thorax, data = ff,
     pch = 19, col='darkgrey',
     main = "Fitted regression line
     with 97% confidence interval")
lines(longevity ~ thorax, data = newdata)
lines(lwr ~ thorax, data = newdata, lty = 2
lines(upr ~ thorax, data = newdata, lty = 2
```



tidyverse

```
## plot fitted line against the raw data
ggplot(mapping =
         aes(x = thorax, y = longevity)) +
    geom_point(data = ff) +
    geom_line(data = newdata) +
    geom_ribbon(
        aes(ymin = lwr, ymax = upr),
        data = newdata, alpha = 0.5)
```



Task 3

Produce the same plot but with a 97% **prediction** interval

Show Solution on P103

2.6 Multiple linear regression

So far we have looked at examples with one response and one explanatory variable. In most applications we will have several variables that affect our outcome. Extending our simple linear regression model to accommodate multiple explanatory variables is straightforward, we just add them to the model.

Using our illustrative example of height vs weight, let's add another explanatory variable, for example, the mean height of the individual's parents (`heightParents`). Our linear model is defined as follows:

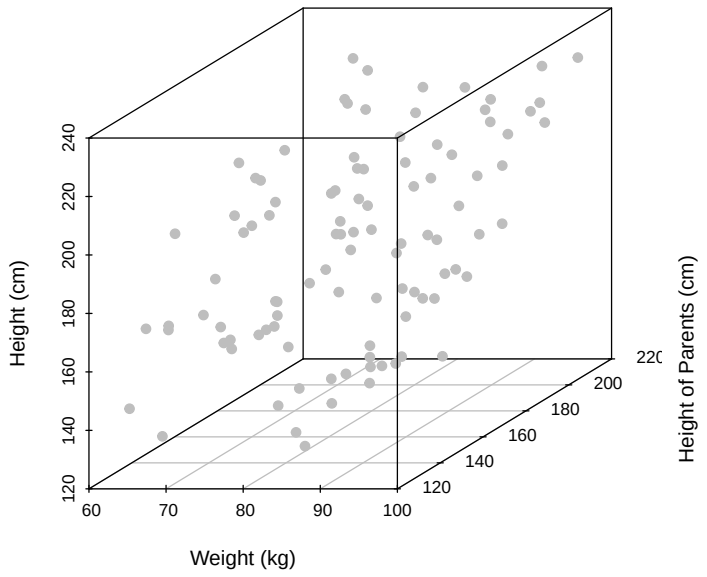
$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \epsilon_i$$
$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Or in English:

$$\text{height}_i = \beta_0 + \beta_1 \text{weight}_i + + \beta_2 \text{heightParents}_i + \epsilon_i$$
$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Let us make up some data again and plot it.

```
set.seed(451)

## no. of observations
N <- 100

## hypothetical weights in kg
weight <- runif(n=N, min=60, max=100)

## hypothetical mean heights of parents in cm
heightParents <- runif(n=N, min=130, max=210)

## hypothetical heights in cm
height <- 0.1*weight + 1.05*heightParents + rnorm(n=N, mean=0, sd=10)

## store as df
df <- data.frame(weight=weight, heightParents=heightParents, height=height)

## Plot
library(scatterplot3d) ## library needed for 3D plotting
scatterplot3d(weight, heightParents, height, pch=19, xlab='Weight (kg)',
              ylab='Height of Parents (cm)', zlab='Height (cm)', color='grey')
```



Note how by adding another explanatory variable we are *spreading* the observations across an additional dimension. That is, we go from a 2D plot to a 3D plot (if we add a third explanatory variable we would need a 4D plot). It is important to keep this in mind; the more explanatory variables we add the more we spread our data thinly across multiple dimensions. In practice, this means that our observations will be sparsely scattered across a high dimensional space, making it hard to fit a robust linear model. In the limit, when we have more explanatory variables than observations, we would not be able to fit a linear model at all (unless we employ a different statistical framework and impose further assumptions/constraints).

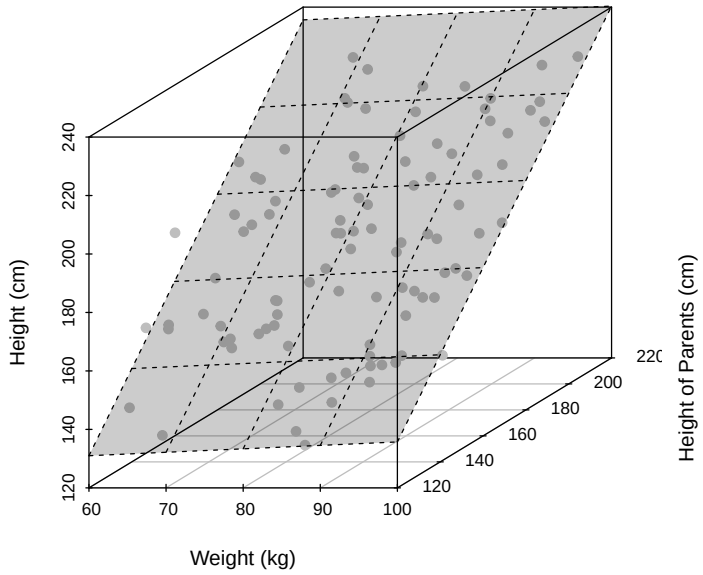
The objective of linear modelling is still the same, finding the "best" straight line, or in this case a **hyperplane** (a line in higher dimensions). You can think of a **hyperplane** as a (rigid) sheet of paper, where the objective is to place it such that it passes as close as possible to the observed data.

To fit a multiple linear regression model we use the `lm()` function again and pass the appropriate **formula** object which has the form of `response ~ explanatory_1 + explanatory_2`. In in our case this will be `height ~ weight + heightParents`.

```
fit <- lm(height ~ weight + heightParents, df)
```

Let us plot the resultant model first (i.e the hyperplane).

```
hFig <- scatterplot3d(weight, heightParents, height, pch=19, xlab='Weight (kg)',
                      ylab='Height of Parents (cm)', zlab='Height (cm)', color='grey')
hFig$plane3d(fit, draw_polygon=TRUE)
```



Let us look at the fit's summary

```
summary(fit)
```

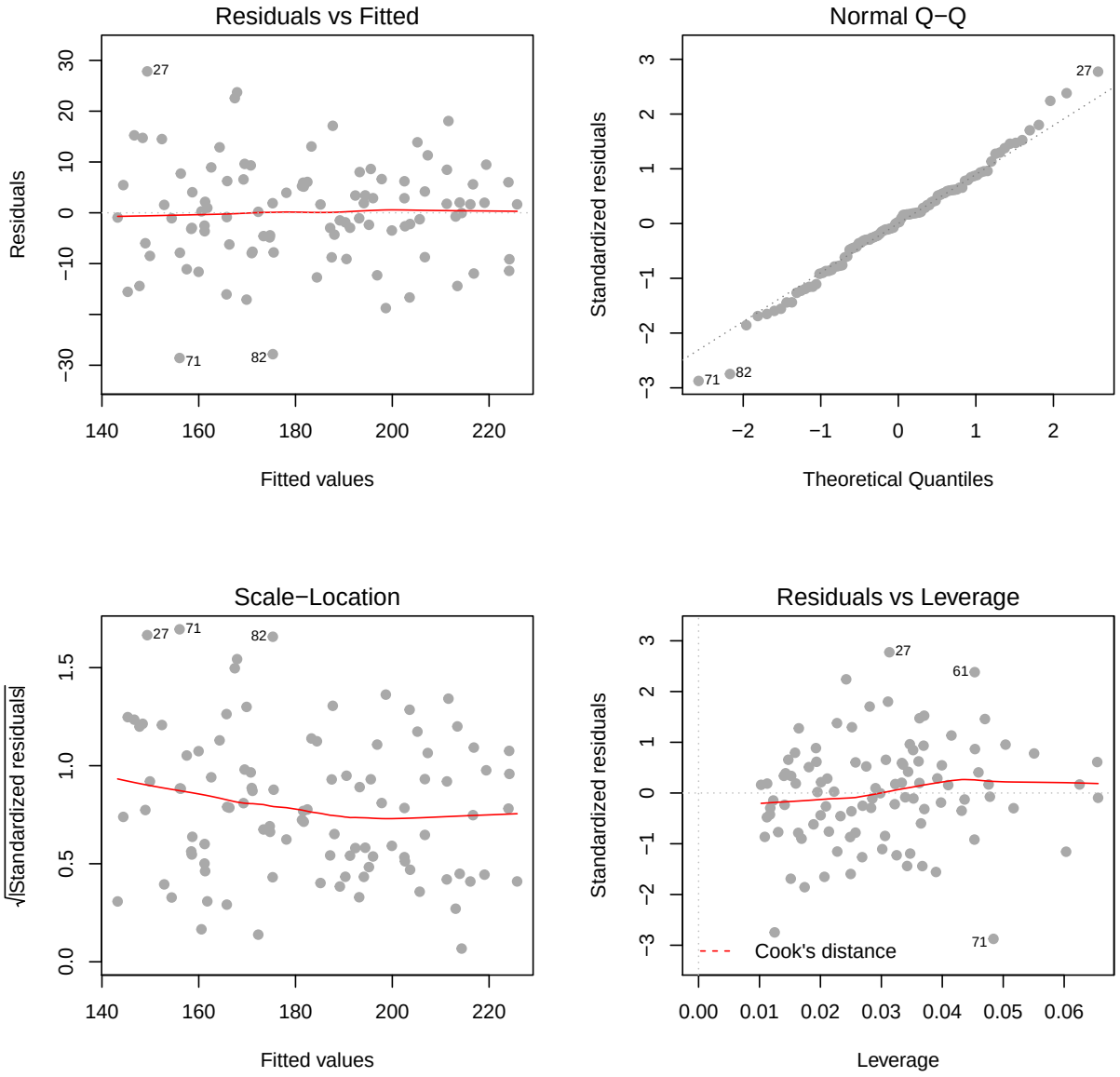
```
##
## Call:
## lm(formula = height ~ weight + heightParents, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.578  -6.052   0.235   6.027  27.829
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.65589   10.70833  -0.155    0.877
## weight         0.11790    0.09198   1.282    0.203
## heightParents  1.04644    0.04707  22.233   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.19 on 97 degrees of freedom
## Multiple R-squared:  0.8372, Adjusted R-squared:  0.8338
## F-statistic: 249.4 on 2 and 97 DF,  p-value: < 2.2e-16
```

Same as before, the `(Intercept)`= -1.656 cm, `weight`= 0.1179 cm/kg and `heightParents`= 1.046 cm/cm are the β_0, β_1 and β_2 parameters.

- The β_0 parameter (intercept) is the expected height of someone that weighs 0 kg *and* whose parents have a mean height of 0 cm. Again, this parameter is not useful in this case.
- The β_1 parameter tells us about the relationship between `height` and `weight`. For every 1 kg increase in weight on *average* the height increases by 0.1179 cm.
- The β_2 parameter tells us about the relationship between `height` and `heightParents`. For every 1 cm increase in mean height of parents on *average* the person's height increases by 1.046 cm.

As before we look at the model diagnostic plots to check the model fit.

```
par(mfrow=c(2, 2))
plot(fit, pch=19, col='darkgrey')
```



We can extend this framework to any arbitrary large number of explanatory variables. The model is:

$$y_i = \beta_0 + \beta_1 x_{1i} + \ldots + \beta_p x_{pi} + \epsilon_i$$
$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Where i is an index that goes from 1 to n (the total number of observations) and p is the total number of explanatory variables. Hopefully it is now clearer why statisticians use Greek letters $\beta_0, \ldots, \beta_p$, else the equation would be too long to write in English

outcome = intercept + ((gradient outcome vs explanatory variable 1) x (explanatory variable 1)) + ((gradient

outcome vs explanatory variable 2) x (explanatory variable 2)) + . . .)

2.7 Practical 2

We will use the fruitfly dataset (Partridge and Farquhar (1981)) as we did in the previous practical

Task 4

1. Fit a linear model with lifespan as response variable and thorax length and sleep as explanatory variables
2. Display a summary of the fit, together with the 97% confidence intervals for the estimated parameters
3. What's the practical significance of the estimated parameters?
4. Show the diagnostic plots for the model, what can you say about the model fit?
5. What is the total variation explained by `thorax` and `sleep`?

Show Solution on P103

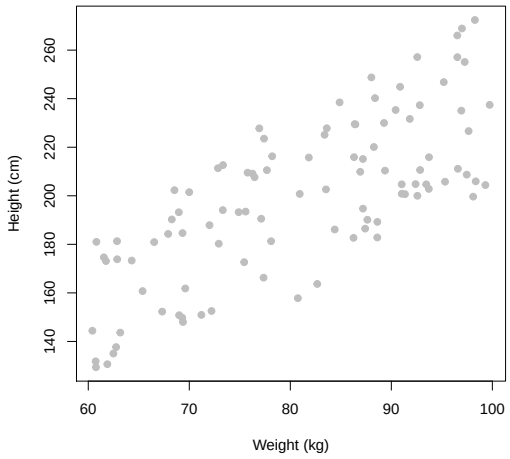
2.8 Categorical explanatory variables

So far we have only considered **continuous** explanatory variables. How do we include **categorical** explanatory variables? Let's go back to our `height` vs `weight` toy example and consider `sex` as an additional categorical variable. Let us simulate some data.

```
set.seed(101)

## no. of observations
N <- 50

## hypothetical weights in kg
weightMale <- runif(n=N, min=60, max=100)

 ## hypothetical weights in kg
weightFemale <- runif(n=N, min=60, max=100)

## hypothetical heights in cm
heightMale <- 2.2*weightMale + rnorm(n=N, mean=0, sd=10) + 40

## hypothetical heights in cm
heightFemale <- 2.2*weightFemale + rnorm(n=N, mean=0, sd=10) + 2
height <- c(heightMale, heightFemale)
weight <- c(weightMale, weightFemale)
sex <- c(rep('Male', N), rep('Female', N))

## Store in data frame
df <- data.frame(weight=weight, sex=sex, height=height)
```

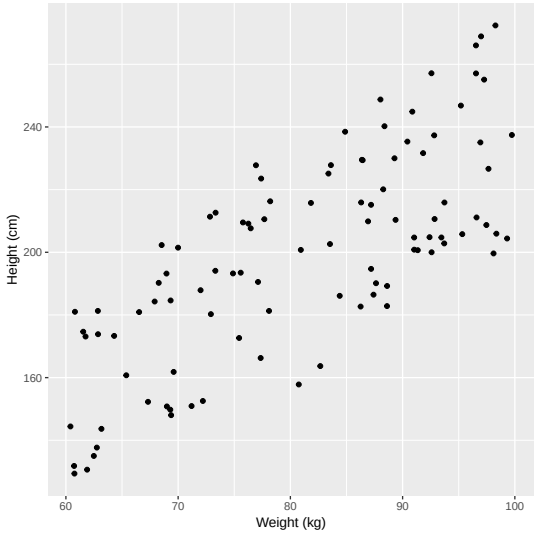
Base R

```
## Plot
plot(height ~ weight, data=df,
     pch=19, xlab='Weight (kg)',
     ylab='Height (cm)', col='grey')
```



tidyverse

```
ggplot(df) +
    geom_point(
        aes(x = weight, y = height)) +
    xlab("Weight (kg)") +
    ylab("Height (cm)")
```

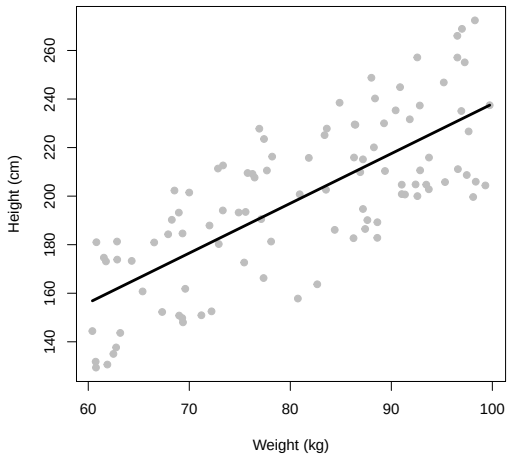


Data looks similar as before, albeit with a larger variation. We can fit a model as we have done before and assess the relationship between `height` and `weight`.

```
## Linear model fit
fit <- lm(height ~ weight, data=df)
```

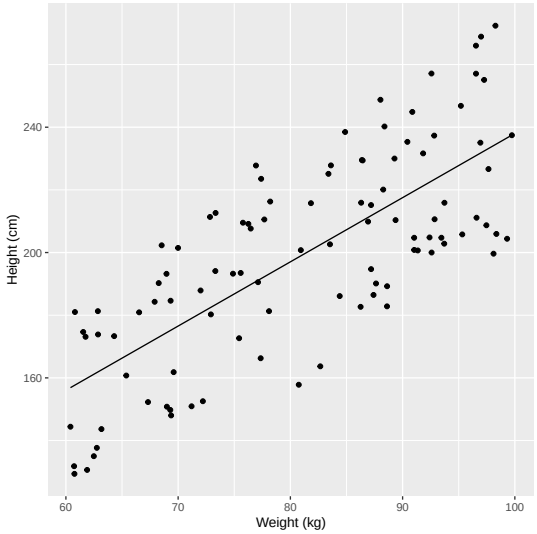
Base R

```
## predictions
newdata <- data.frame(
    weight = seq(min(df$weight),
        max(df$weight), length.out = 50))
newdata$height <- predict(fit, newdata)

## Plot model fit
plot(height ~ weight, data = df,
     pch=19, xlab='Weight (kg)',
     ylab='Height (cm)', col='grey')
lines(height ~ weight, data = newdata,
     col='black', lwd=3)
```



tidyverse

```
## plot model fit
newdata <- data.frame(
    weight = seq(min(df$weight),
        max(df$weight), length.out = 50))
newdata <- mutate(newdata,
    height = predict(fit, newdata))
ggplot(mapping =
        aes(x = weight, y = height)) +
    geom_point(data = df) +
    geom_line(data = newdata) +
    xlab("Weight (kg)") +
    ylab("Height (cm)")
```



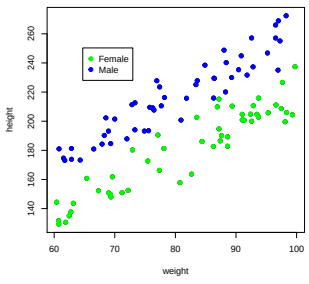
```
## Print result
summary(fit)
```

```
##
## Call:
## lm(formula = height ~ weight, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.753 -20.526   2.579  18.889  37.925
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  33.0877    15.2188   2.174   0.0321 *
## weight        2.0493     0.1859  11.024   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22.08 on 98 degrees of freedom
## Multiple R-squared:  0.5536, Adjusted R-squared:  0.549
## F-statistic: 121.5 on 1 and 98 DF,  p-value: < 2.2e-16
```

If we colour the data by sex it becomes obvious that this extra variation is due to sex.

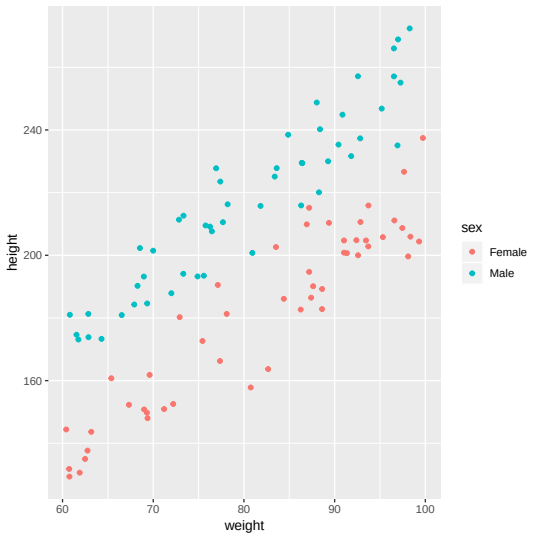
Base R

```
plot(height ~ weight, data = df,
     pch = 19)
points(height ~ weight,
       pch = 19,
       data = df[df$sex == "Female", ],
       col = "green")
points(height ~ weight,
       pch = 19,
       data = df[df$sex == "Male", ],
       col = "blue")
legend(par("usr")[1] * 1.1,
       par("usr")[4] * 0.9,
       legend = c("Female", "Male"),
       pch = c(19, 19),
       col = c("green", "blue"))
```



tidyverse

```
ggplot(df,
       aes(x = weight, y = height,
           colour = sex)) +
    geom_point()
```



`sex` is an important covariate and should therefore be added to the model. That is, we need a model that has different regression lines for each sex. The *syntax* for the R formula remains exactly the same `response ~ explanatory_1 + ... + explanatory_p`, but behind the scenes categorical variables are treated differently to continuous variables.

To cope with categorical variables (known as **factors** in R's lingo), we introduce a **dummy** variable.

$$S_i = \begin{cases} 1 & \text{if } i \text{ is male,} \\ 0 & \text{otherwise} \end{cases}$$

Here, female is known as the **baseline/reference level**

The regression is:

$$y_i = \beta_0 + \beta_1 S_i + \beta_2 x_i + \epsilon_i$$

Or in English:

$$\text{height}_i = \beta_0 + \beta_1 \text{sex}_i + \beta_2 \text{weight}_i + \epsilon_i$$

Where `sex` is the dummy variable `S_i` that can take the value of 0 (female) or 1 (male)

The mean regression lines for male and female now look like this:

- **Female** (`sex=0`)

$$\begin{aligned} \text{height}_i &= \beta_0 + (\beta_1 \times 0) + \beta_2 \text{weight}_i \\ \text{height}_i &= \beta_0 + \beta_2 \text{weight}_i \end{aligned}$$

- **Male** (`sex=1`)

$$\begin{aligned} \text{height}_i &= \beta_0 + (\beta_1 \times 1) + \beta_2 \text{weight}_i \\ \text{height}_i &= (\beta_0 + \beta_1) + \beta_2 \text{weight}_i \end{aligned}$$

By introducing a dummy variable, we have now simplified the problem into **two** regression lines, where the **intercept** is:

- **Female**: β_0
- **Male**: $\beta_0 + \beta_1$

Whilst the **gradient** is β_2 in both cases.

```
## Linear model fit
fit <- lm(height ~ sex + weight, data=df)
```

```
## Print result
summary(fit)
```

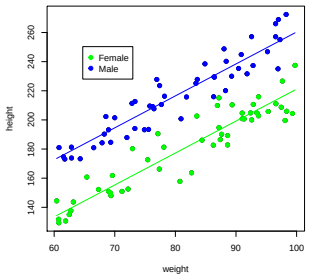
```
##
## Call:
## lm(formula = height ~ sex + weight, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.106  -6.853  -1.162   6.691  22.111
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.8269     7.0411   0.259    0.796
## sexMale      39.2220     1.9976  19.634   <2e-16 ***
## weight        2.1931     0.0841  26.078   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.95 on 97 degrees of freedom
## Multiple R-squared:  0.9103, Adjusted R-squared:  0.9084
## F-statistic: 491.9 on 2 and 97 DF,  p-value: < 2.2e-16
```

The `sexMale` coefficient is β_1 in the equations above and it represents the *difference* in intercept between males and females (remember that the gradient is the same for both sexes). Thus for the *same* weight, men on average will be `sexMale`=39.2 cm taller than females.

To plot the fitted lines, we can use a similar trick using `predict()` as before. This time we need to produce predictions using all relevant combinations of explanatory variables: `weight` and `sex` in this case. A useful function to do this is `expand.grid()`:

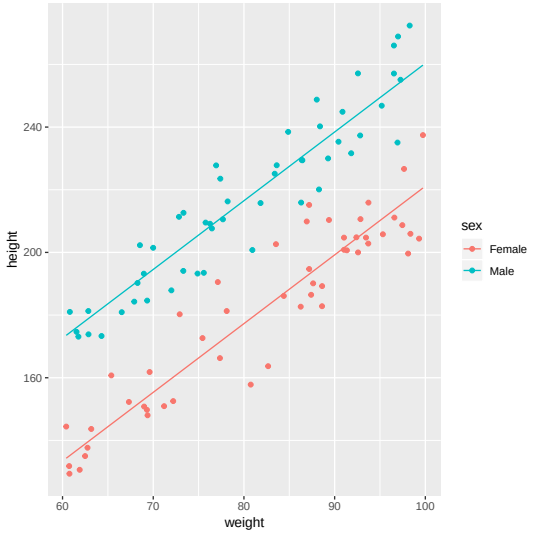
Base R

```
## create dataset to predict to
newdata <- expand.grid(
    weight = seq(min(df$weight),
        max(df$weight), length.out = 50),
    sex = c("Female", "Male"))

## generate predictions
newdata <- cbind(newdata,
    height = predict(fit, newdata))

## plot fitted lines against the data
plot(height ~ weight, data = df,
    pch = 19)
points(height ~ weight,
    pch = 19,
    data = df[df$sex == "Female", ],
    col = "green")
points(height ~ weight,
    pch = 19,
    data = df[df$sex == "Male", ],
    col = "blue")
lines(height ~ weight,
    data = newdata[
        newdata$sex == "Female", ],
    col = "green")
lines(height ~ weight,
    data = newdata[
        newdata$sex == "Male", ],
    col = "blue")
legend(par("usr")[1] * 1.1,
    par("usr")[4] * 0.9,
    legend = c("Female", "Male"),
    pch = c(19, 19),
    col = c("green", "blue"))
```



tidyverse

```
## create dataset to predict to
newdata <- expand.grid(
    weight = seq(min(df$weight),
        max(df$weight), length.out = 50),
    sex = c("Female", "Male"))

## generate predictions
newdata <- mutate(newdata,
    height = predict(fit, newdata))

## plot fitted lines against the data
ggplot(mapping =
        aes(x = weight, y = height,
            colour = sex)) +
    geom_point(data = df) +
    geom_line(data = newdata)
```



Task 5

Add 97% confidence intervals to the fitted lines plots above.

Show Solution on P106

Task 6

Compare the total variation explained by the model with just weight to the one which considers weight *and* sex. What gives rise to this difference?

Show Solution on P107

Note the baseline/reference level in R is automatically taken to be the *first* factor level.

```
levels(df$sex)
```

```
## [1] "Female" "Male"
```

We can change this by re-ordering the levels if we wanted to.

```
df$sex <- factor(df$sex, levels = c('Male', 'Female'))
```

```
## Linear model fit
fit <- lm(height ~ sex + weight, data=df)

## Print result
summary(fit)
```

```
##
## Call:
## lm(formula = height ~ sex + weight, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.106  -6.853  -1.162   6.691  22.111
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.0489     6.8707   5.975 3.81e-08 ***
## sexFemale   -39.2220     1.9976 -19.634  < 2e-16 ***
## weight        2.1931     0.0841  26.078  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.95 on 97 degrees of freedom
## Multiple R-squared:  0.9103, Adjusted R-squared:  0.9084
## F-statistic: 491.9 on 2 and 97 DF,  p-value: < 2.2e-16
```

As expected this change simply changes the sign of the β_1 parameter, because the dummy variable now takes the value of 0 for male and 1 for female (i.e we switched things around).

2.9 Practical 3

We will use the fruitfly dataset (Partridge and Farquhar (1981)) as we did in the previous practical

Task 7

1. Fit a linear model with lifespan as response variable and thorax length and type of companion as explanatory variables
2. Display a summary of the fit, together with the 97% confidence intervals for the estimated parameters
3. Why do we get two parameters for type of companion and what do they mean in practice?

Show Solution on P107

Task 8

4. Plot the mean regression line for each type of companion together with the observed data.

Show Solution on P109

Task 9

5. Show the diagnostic plots for the model, what can you say about the model fit?
6. Compare the total variation explained by this model with the one that only used `thorax` as a covariate. What can you say about the importance of the type of companion?

Show Solution on P109

2.10 Practical issues

The most common issues when trying to fit simple linear regression models is that our response variable is not normal which violates our modelling assumption. There are two things we can do in this case:

1. **Variable transformation**
   - Can sometimes fix linearity
   - Can sometimes fix non-normality and heteroscedasticity (i.e non-constant variance)
2. ***Generalised* Linear Models (GLMs)** to change the **error** structure (i.e the assumption that residuals need to be normal)

2.11 Summary

Linear regression is a powerful tool:

- It splits the data into **signal** (trend / mean), and **noise** (residual error).
- It can cope with **multiple variables**.
- It can incorporate different **types** of variable
- It can be used to produce **point** and **interval** estimates for the parameters.
- It can be used to assess the importance of variables.

But always **check** that the model fit is sensible and that the results make sense within the context of the problem at hand. Investigate thoroughly if they do not!!

Chapter 3

Generalised linear models

Slides can be downloaded from:

- GLMs in R

3.1 Motivation

In the previous workshop we have seen that linear models are a powerful modelling tool. However, we have to satisfy the following assumptions:

1. A **linear** mean function is relevant.
2. Variances are equal across all predicted values of the response (**homoscedatic**)
3. Errors are **normally** distributed.
4. Samples collected at **random**.
5. Errors are **independent**.

If assumptions 1–3 are violated we can *transform* our response variable to try and fix this (power transforms, Tukey's ladder-of-powers, Box-Cox transformation, Tukey and Mosteller's bulging rule). However, in a lot of other cases this is either not possible (e.g binary output) or we want to explicitly model the underlying distribution (e.g count data). Instead, we can use *Generalised* Linear Models (GLMs) that let us change the *error structure* of our data. By error structure we mean the assumption placed on the *residuals*. In the previous simple linear regression case we assumed them to be normal (i.e $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$)

3.2 Generalised Linear Models (GLMs)

Generalised Linear Models (GLMs) have:

1. A linear mean (of your making).
2. A **link function** (like an 'internal' transformation).
3. An **error structure**.

3.2.1 Link functions

A **link** function *links* your ***mean*** function to the scale of the **observed data** e.g.

- **Response** variable Y and explanatory variable(s) X.
- The **regression** parameters are denoted using β_p as before.

- **Linear** function: $\beta_0 + \beta_1 X$.
- $E(Y) = g^{-1}(\beta_0 + \beta_1 X)$.

Here $E(Y)$ denotes the **expected value** (i.e. mean of Y).

The function $g(\cdot)$ is known as the **link function**, and $g^{-1}(\cdot)$ denotes the **inverse** of $g(\cdot)$.

The simple linear regression model we have used so far is a special cases of a GLM:

```
lm(height ~ weight)
```

is equivalent to

```
glm(height ~ weight, family=gaussian(link=identity))
```

Compared to `lm()`, the `glm()` function takes an additional argument called `family`, which specifies the error structure **and** link function.

The default **link function** for the normal (Gaussian) distribution is the **identity**, i.e. for mean μ we have:

$$\mu = \beta_0 + \beta_1 X$$

Defaults are usually good choices (shown in bold below):

Family	Link
`gaussian`	**`identity`**
`binomial`	**`logit`**, `probit` or `cloglog`
`poisson`	**`log`**, `identity` or `sqrt`
`Gamma`	**`inverse`**, `identity` or `log`
`inverse.gaussian`	**`1/mu^2`**
`quasi`	user-defined
`quasibinomial`	**`logit`**
`quasipoisson`	**`log`**

Task 10

Using the fruitfly data introduced in Practical 1, fit a linear model with lifespan as response variable and thorax length and type of companion as explanatory variables using both the `lm` and `glm` functions and compare their summaries.

Show Solution on P110

3.2.2 Workflow

- Exploratory data analysis
- Choose suitable error term
- Choose suitable mean function (and link function)
- Fit model
  - Residual checks and model fit diagnostics
  - Revise model (transformations etc.)
- Model simplification if required
- Check final model

3.3 Poisson regression (for count data)

Count data are ubiquitous in the life sciences (e.g number of parasites per microlitre of blood, number of species in a particular area). These type of data are **discrete** and **non-negative**. In such cases assuming our response variable to be normally distributed is not typically sensible. The Poisson distribution lets us model count data explicitly.

Recall the simple linear regression case (i.e a GLM with a Gaussian error structure and identity link). For the sake of clarity let's consider a single explanatory variable and omit the index i which runs from 1 to n (the total number of observations/data points):

$$\begin{aligned} Y &= \beta_0 + \beta_1 X + \epsilon \\ \epsilon &\sim \mathcal{N}(0, \sigma^2) \end{aligned}$$

Which can be re-written as:

$$\begin{aligned} Y &\sim \mathcal{N}(\mu, \sigma^2) \\ \mu &= \beta_0 + \beta_1 X \end{aligned}$$

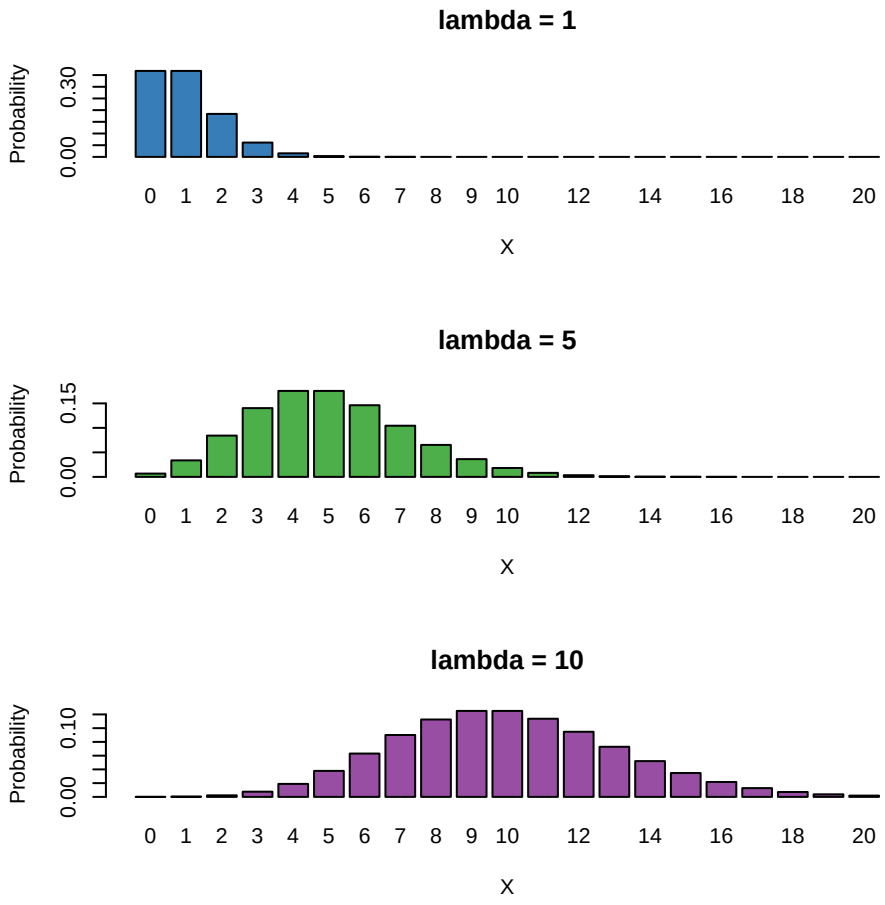
The mean function is **unconstrained**, i.e the value of $\beta_0 + \beta_1 X$ can range from $-\infty$ to $+\infty$. If we want to model count data we therefore want to **constrain** this mean to be positive. Mathematically we can do this by taking the **logarithm** of the mean (it is by no coincidence that log transforms are very popular to normalise response variables). We then assume our count data to be Poisson distributed (a discrete, non-negative distribution), to obtain our Poisson regression model (to be consistent with the statistics literature we will rename μ to λ):

$$\begin{aligned} Y &\sim \mathcal{P}\wr\rangle\int(\lambda) \\ \log \lambda &= \beta_0 + \beta_1 X \end{aligned}$$

Note: we are still fitting straight lines (hyperplanes in higher dimensions) through our data! The only difference is that it is linear for the log transformed observations.

Where's the variance parameter in this case (i.e anagolous to σ^2)? The **Poisson** distribution has the following characteristics:

- **Discrete** variable, defined on the range $0, 1, \ldots, \infty$.
- A single ***rate*** parameter λ, where $\lambda > 0$.
- **Mean** $= \lambda$
- **Variance** $= \lambda$



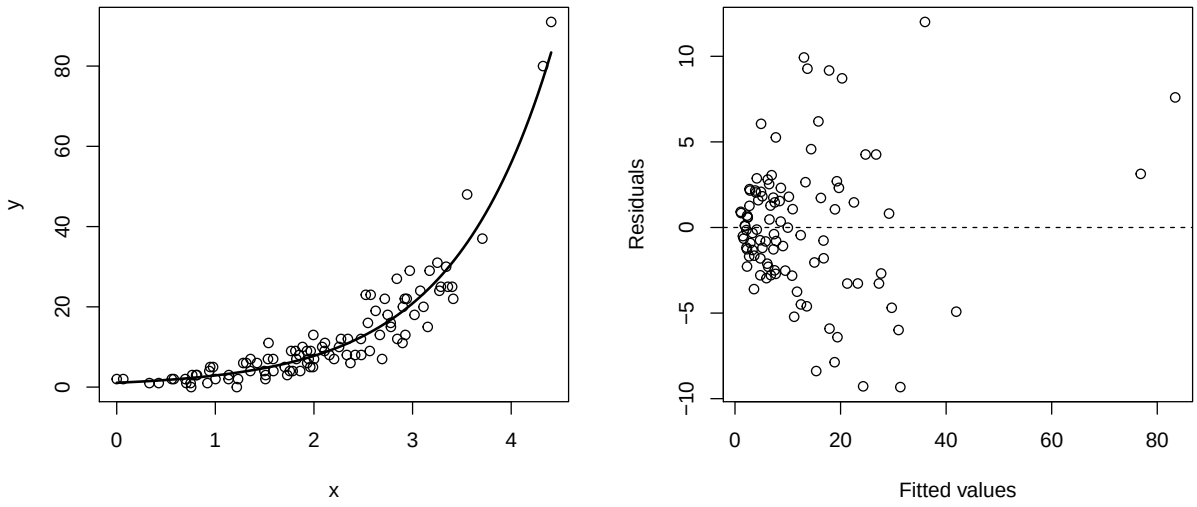
So for the Poisson regression case we assume that the mean and variance are the same. Hence, as the mean *increases*, the variance *increases* also (**heteroscedascity**). This may or may not be a sensible assumption so watch out![1]

Recall the link function and the rules of logarithms (if $\log \lambda = k$, then $\lambda = e^k$):

$$\log \lambda = \beta_0 + \beta_1 X$$
$$\lambda = e^{\beta_0 + \beta_1 X}$$

Thus we are effectively modelling the observed counts (on the original scale) using an exponential mean function.

[1] the negative binomial model and others can cope with cases where this assumption is not satisfied



3.4 Example: Cuckoos

In a study by Kilner *et al.* (1999), the authors studied the begging rate of nestlings in relation to total mass of the brood of **reed warbler chicks** and **cuckoo chicks**. The question of interest is:

> How does nestling mass affect begging rates between the different species?

Download the data file from here and save it to your working directory.[2]

[2]this dataset was obtained by digitising the plot that appear in the original paper, hence you will not be able to fully reproduce the results of the original paper, it is only used here for illustrative purposes

```
cuckoo <- read.csv("cuckoo.csv", header=T)
```

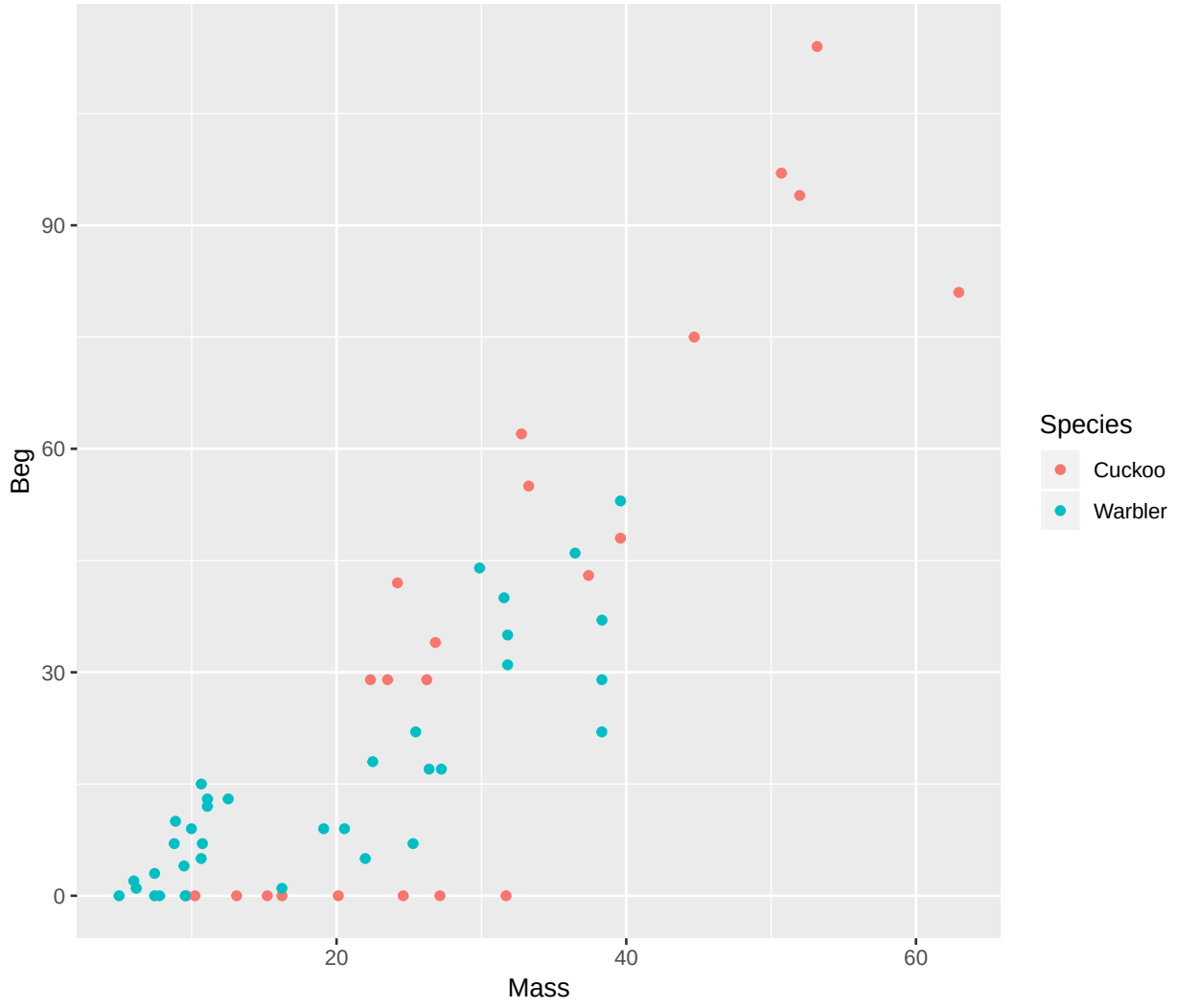
```
head(cuckoo)
```

```
##        Mass Beg Species
## 1  9.637522   0  Cuckoo
## 2 10.229151   0  Cuckoo
## 3 13.103706   0  Cuckoo
## 4 15.217391   0  Cuckoo
## 5 16.231884   0  Cuckoo
## 6 20.120773   0  Cuckoo
```

The data columns are:

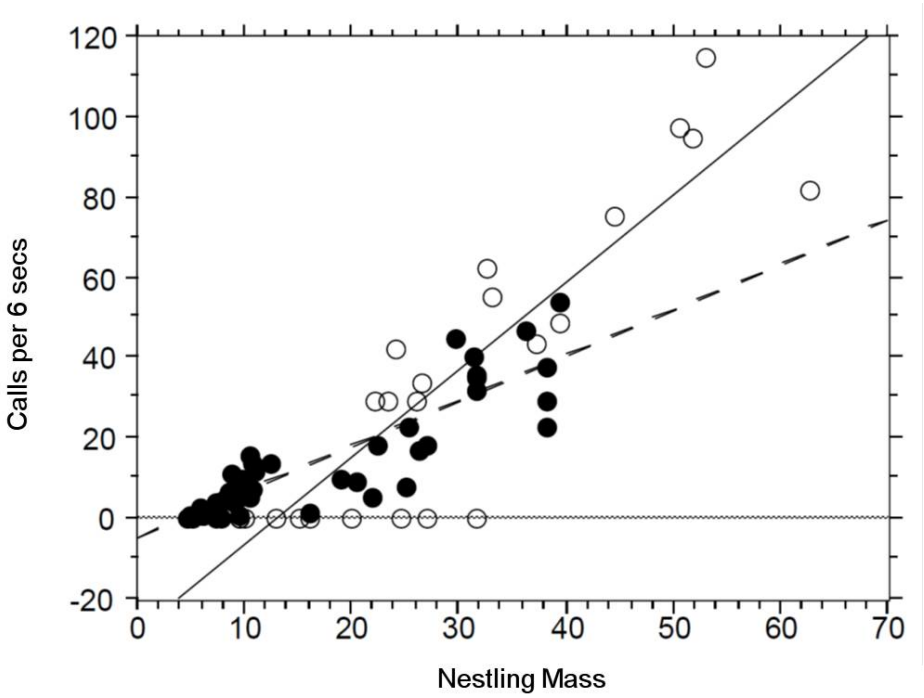
- **Mass**: nestling mass of chick in grams
- **Beg**: begging calls per 6 secs
- **Species**: Warbler or Cuckoo

```
ggplot(cuckoo, aes(x=Mass, y=Beg, colour=Species)) + geom_point()
```



There seem to be a relationship between mass and begging calls and it could be different between species. It

is tempting to fit a linear model to this data. In fact, this is what the authors of the original paper did; **reed warbler chicks** (solid circles, dashed fitted line) and **cuckoo chick** (open circles, solid fitted line):

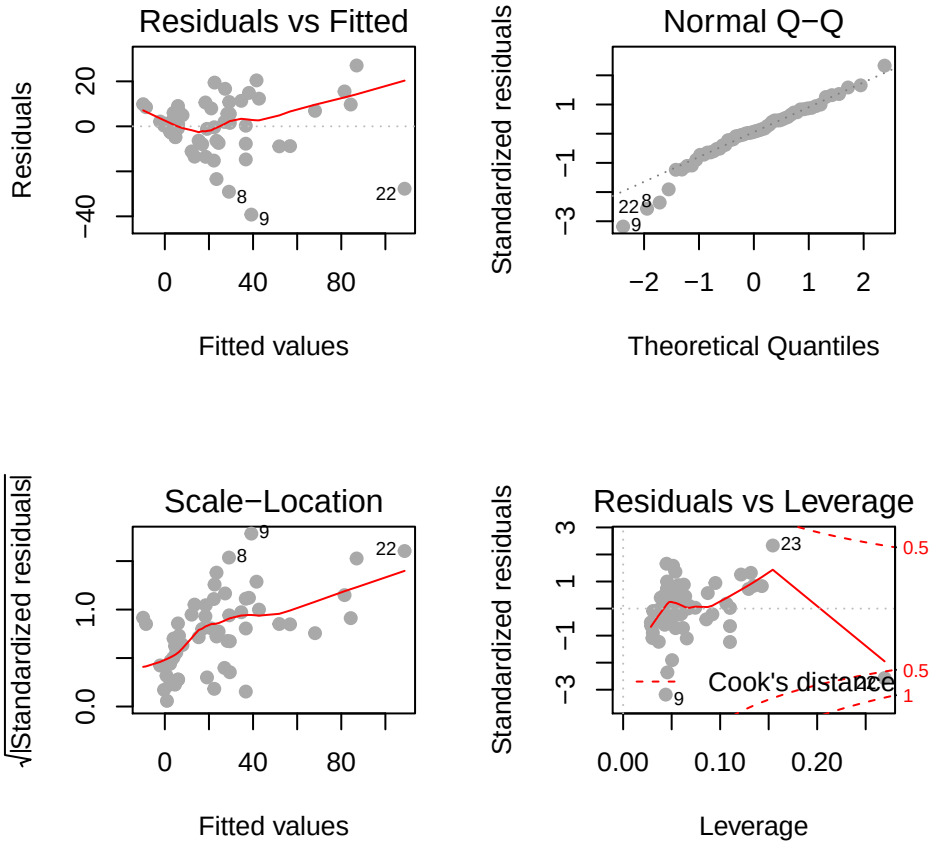


This model is inadequate. It is predicting **negative** begging calls *within* the range of the observed data, which clearly does not make any sense.

Let us display the model diagnostics plots for this linear model.

```
## Fit model
## We add an interaction term here, we will talk about this later on
fit <- lm(Beg ~ Mass*Species, data=cuckoo)
```

```
par(mfrow=c(2, 2))
plot(fit, pch=19, col='darkgrey')
```



The residuals plot depicts a "funnelling" effect, highlighting that the model assumptions are violated. We should therefore try a different model structure.

The response variable in this case is a classic **count data**: **discrete** and bounded below by zero (i.e we cannot have negative counts). We will therefore try a **Poisson model** using a **log** link function for the mean:

$$\log \lambda = \beta_0 + \beta_1 M_i + \beta_2 S_i + \beta_3 M_i S_i$$

where M_i is nestling mass and S_i a **dummy** variable (refer to Categorical explanatory variables):

$$S_i = \begin{cases} 1 & \text{if } i \text{ is warbler,} \\ 0 & \text{otherwise.} \end{cases}$$

The term $M_i S_i$ is an **interaction** term. Think of this as an additional explanatory variable in our model. Effectively it lets us have **different** slopes for different species (without an interaction term we assume that both species have the same slope for the relationship between begging rate and mass, and only the intercept differ).

The mean regression lines for the two species look like this:

- **Cuckoo** ($S_i = 0$)

$$\log \lambda = \beta_0 + \beta_1 M_i + (\beta_2 \times 0) + (\beta_3 \times M_i \times 0)$$
$$\log \lambda = \beta_0 + \beta_1 M_i$$

- **Intercept** $= \beta_0$, **Gradient** $= \beta_1$
- **Warbler** $(S_i = 1)$

$$\log \lambda = \beta_0 + \beta_1 M_i + (\beta_2 \times 1) + (\beta_3 \times M_i \times 1)$$
$$\log \lambda = \beta_0 + \beta_1 M_i + \beta_2 + \beta_3 M_i$$
$$\log \lambda = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) M_i$$

- **Intercept** $= \beta_0 + \beta_2$, **Gradient** $= \beta_1 + \beta_3$

To specify an interaction term in R we use the `*` operator.

- Model with **no** interaction term: $\log \lambda = \beta_0 + \beta_1 M_i + \beta_2 S_i$

```
glm(Beg ~ Mass + Species, data=cuckoo, family=poisson(link=log))
```

- Model **with** interaction term: $\log \lambda = \beta_0 + \beta_1 M_i + \beta_2 S_i + \beta_3 M_i S_i$

```
glm(Beg ~ Mass*Species, data=cuckoo, family=poisson(link=log))
```

Fit the model with the interaction term in R:

```
fit <- glm(Beg ~ Mass*Species, data=cuckoo, family=poisson(link=log))
summary(fit)
```

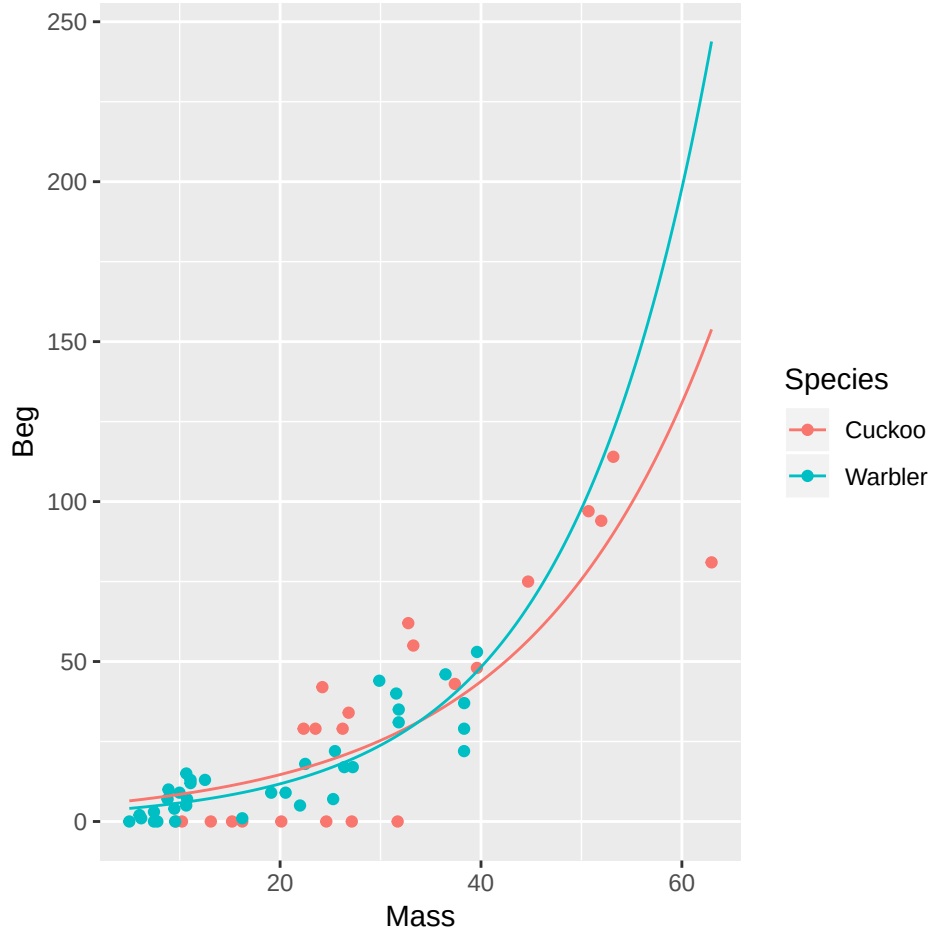
```
##
## Call:
## glm(formula = Beg ~ Mass * Species, family = poisson(link = log),
##     data = cuckoo)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.4570  -3.0504  -0.0006   1.9389   5.2139
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          1.589861   0.104531  15.209  < 2e-16 ***
## Mass                 0.054736   0.002298  23.820  < 2e-16 ***
## SpeciesWarbler      -0.535546   0.161304  -3.320 0.000900 ***
## Mass:SpeciesWarbler  0.015822   0.004662   3.394 0.000689 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1730.04  on 57  degrees of freedom
## Residual deviance:  562.08  on 54  degrees of freedom
## AIC: 784.81
##
## Number of Fisher Scoring iterations: 5
```

For the sake of clarity here is the mapping of the β_p regression coefficients used above and those returned by R.

- `(Intercept)` $= \beta_0$ (intercept for the **reference/baseline** species, **cuckoo** in this case)
- `Mass` $= \beta_1$ (slope for the baseline species)
- `SpeciesWarbler` $= \beta_2$ (the increase/decrease in intercept relative to the baseline species)
- `Mass:SpeciesWarbler` $= \beta_3$ (the increase/decrease in slope relative to the baseline species)

Plot the mean regression line for each species:

```
newdata <- expand.grid(Mass=seq(min(cuckoo$Mass), max(cuckoo$Mass), length.out=200),
                       Species=levels(cuckoo$Species))
newdata <- cbind(newdata, Beg=predict(fit, newdata, type='response'))


ggplot(mapping=aes(x=Mass, y=Beg, colour=Species)) + geom_point(data=cuckoo) +
    geom_line(data=newdata)
```



We get an exponential curve in the scale of the original data, which is the **same** as a straight line in the log-scaled version of the data.

3.5 Practical 4 - Species richness

A long-term agricultural experiment had 90 grassland plots, each 25m x 25m, differing in biomass, soil pH and species richness (the count of species in the whole plot). It is well known that species richness declines with increasing biomass, but the question addressed here is whether the slope of that relationship differs with soil pH (i.e there's an interaction effect). The plots were classified according to a 3-level factor as high, medium or low pH with 30 plots in each level.

The response variable is the **count** of species (`Species`), so a GLM with Poisson errors is a sensible choice. The continuous explanatory variable is long-term average biomass measured in June (`Biomass`), and the

categorical explanatory variable is soil pH (`pH`).

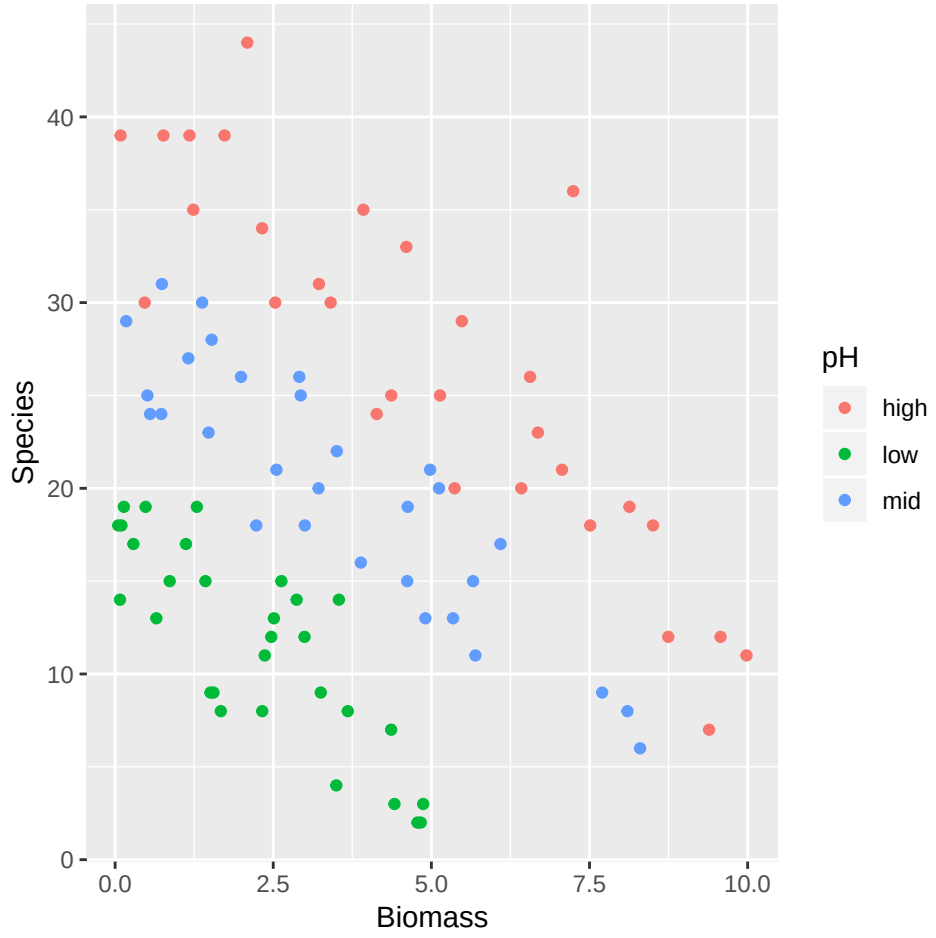
Download the data from here

```
df <- read.csv("species.csv", header=T)
```

```
head(df)
```

```
##     pH   Biomass Species
## 1 high 0.4692972      30
## 2 high 1.7308704      39
## 3 high 2.0897785      44
## 4 high 3.9257871      35
## 5 high 4.3667927      25
## 6 high 5.4819747      29
```

```
ggplot(df, aes(x=Biomass, y=Species, colour=pH)) + geom_point()
```



Task 11

1. Fit a simple linear regression model (i.e assuming residuals to be normally distributed) with `Species` as response variable and `Biomass` and `pH` as explanatory variables. Assume a **different** slope for each `pH` level. Display a summary of the fit.
2. Plot the mean regression lines for all three `pH` levels (low, medium, high)
3. As `Biomass` tends to increase what is the expected number of species found in the grassland for the different pH levels? Is this biologically plausible?
4. Repeat 1–3 this time fitting a **Poisson** regression model.

Show Solution on P111

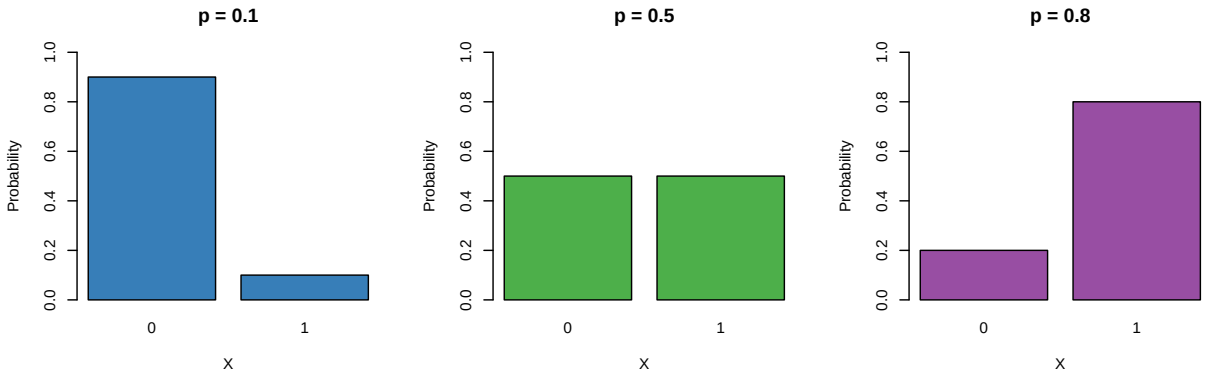
3.6 Logistic regression (for binary data)

So far we have only considered continuous and discrete data as response variables. What if our response is a categorical variable (e.g passing or failing an exam, voting yes or no in a referendum, whether an egg has successfully fledged or been predated)?

We can model the **probability** p of being in a particular class as a function of other explanatory variables. Here we will focus on variables with two levels (e.g dead/alive). [3]. These type of **binary** data are assumed to follow a **Bernoulli** distribution which has the following characteristics:

$$Y \sim \mathcal{B}]\nabla\backslash(p)$$

- **Binary** variable, taking the values 0 or 1 (yes/no, pass/fail).
- A **probability** parameter p, where $0 < p < 1$.
- **Mean** $= p$
- **Variance** $= p(1-p)$



Let us now place the Gaussian (simple linear regression), Poisson and logistic models next to each other:

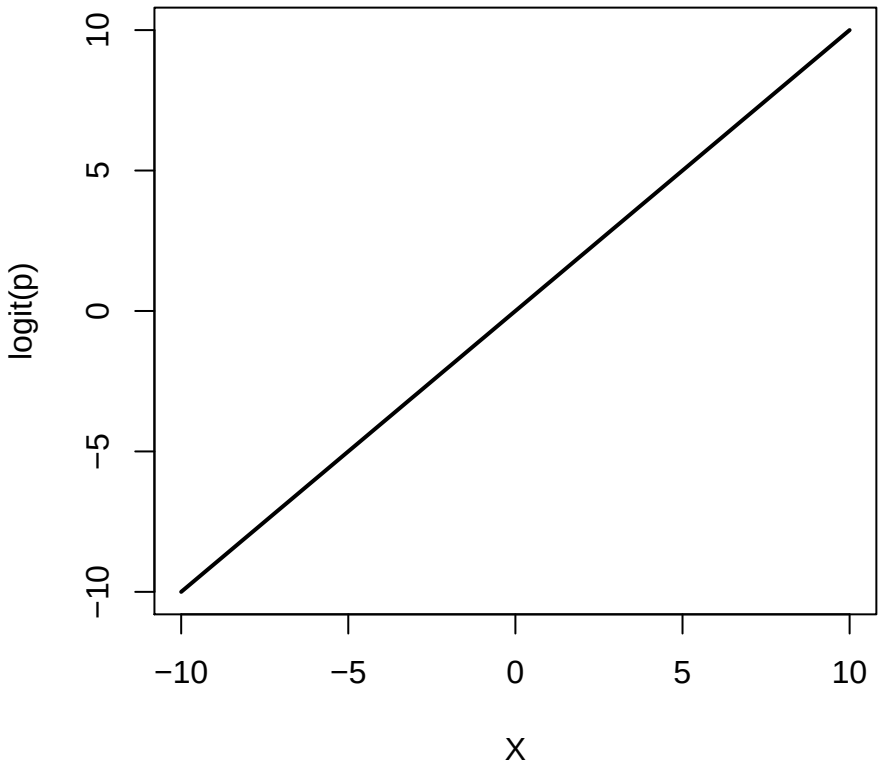
$$\begin{array}{lll} Y \sim \mathcal{N}(\mu, \sigma^2) & Y \sim \mathcal{P}l\rangle\int(\lambda) & Y \sim \mathcal{B}]\nabla\backslash(p) \\ \mu = \beta_0 + \beta_1 X & \log\lambda = \beta_0 + \beta_1 X & ?? = \beta_0 + \beta_1 X \end{array}$$

[3] the multinomial logistic regression model generalises logistic regression to multiclass problems

Now we need to fill in the `??` with the appropriate term. Similar to the Poisson regression case, we cannot simply model the probabiliy as $p = \beta_0 + \beta_1 X$, because p **cannot** be negative. $\log p = \beta_0 + \beta_1 X$ won't work either, because p cannot be greater than 1. Instead we model the **log odds** $\log\left(\frac{p}{1-p}\right)$ as a linear function. So our logistic regression model looks like this:

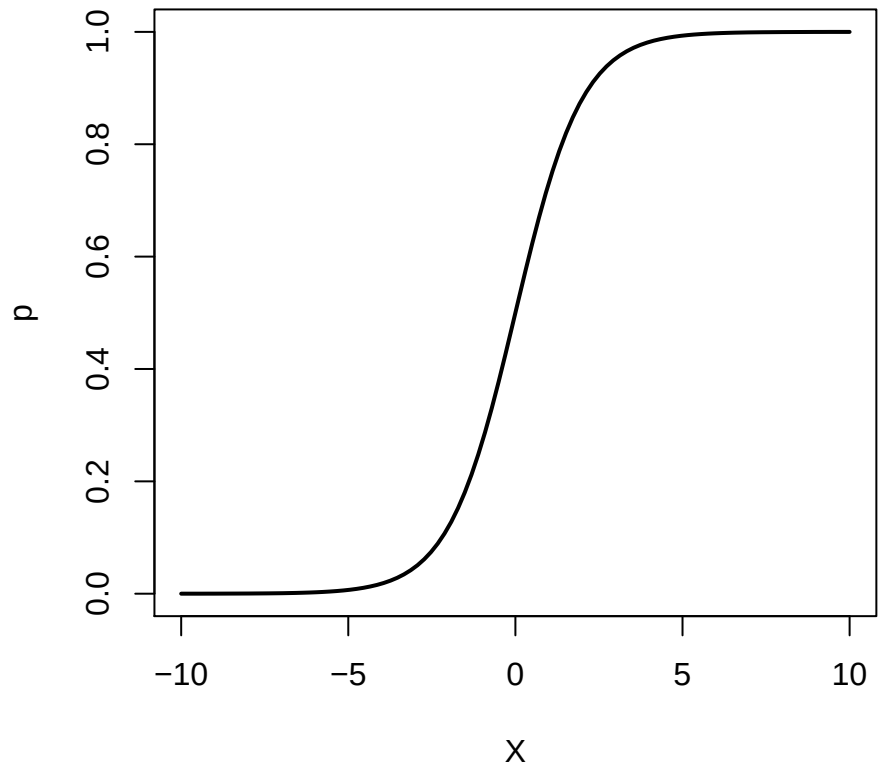
$$\begin{aligned} Y &\sim \mathcal{B}]\nabla\backslash(p) \\ \log\left(\frac{p}{1-p}\right) &= \beta_0 + \beta_1 X \end{aligned}$$

Again, note that we are still "only" fitting straight lines through our data, but this time in the log odds space. As a shorthand notation we write $\log\left(\frac{p}{1-p}\right) = \text{logit}(p) = \beta_0 + \beta_1 X$.



We can also re-arrange the above equation so that we get an expression for p

$$p = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$



Note how p can only vary between 0 and 1.

To implement the logistic regression model in R, we choose `family=binomial(link=logit)` (the Bernoulli distribution is a special case of the Binomial distribution when the number of trials is 1).

```
glm(response ~ explanatory, family=binomial(link=logit))
```

3.7 Example: 1992 US election survey

Voters were asked if they preferred George Bush (Republican) or Bill Clinton (Democrat) (voters who preferred other candidates were excluded). The respondent's income was characterised on a 5-point scale (1 - poor to 5 - rich). The question of interest in this case is:

> Do voters with higher incomes prefer conservative[4] candidates?

Download the data file from here and save it to your working directory.

```
USA <- read.csv("US1992.csv", header=T)
```

```
head(USA)
```

```
##           Vote Income
## 1  George Bush      4
```

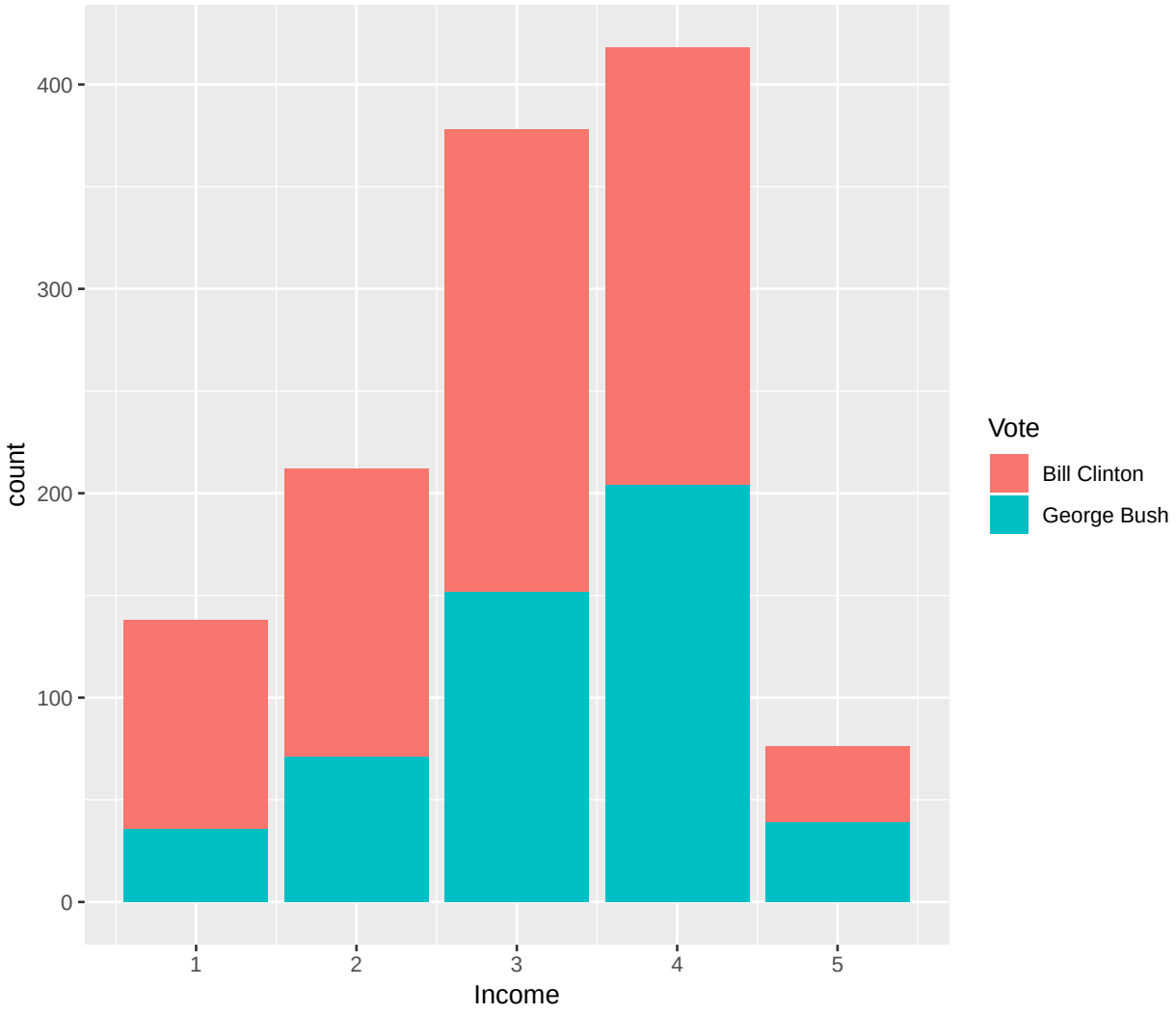
[4] i.e. Republican

```
## 2  George Bush      2
## 3 Bill Clinton      1
## 4  George Bush      2
## 5 Bill Clinton      3
## 6 Bill Clinton      4
```

The data columns are:

- **Vote**: Whether voter preferred Bill Clinton (Democrat) or George Bush (Republican)
- **Income**: 1 - poor to 5 - rich (based on quantiles of earnings in the US)

```
ggplot(USA, aes(x=Income)) + geom_bar(aes(fill=Vote))
```



It does look like people with low income are more likely to prefer Bill Clinton over George Bush. Let us fit a logistic regression model to dig deeper into this. Note that by default R will use the order of the levels to define which one is class "0" (fail) and which one is class "1" (success).

```
levels(USA$Vote) # class 0, class 1
```

```
## [1] "Bill Clinton" "George Bush"
```

So in this case p will represent the probability of preferring George Bush. The probability of preferring Bill

Clinton is simply $1-p$ because we are only considering two options.

```
fit <- glm(Vote ~ Income, data=USA, family=binomial(link=logit))
summary(fit)
```

```
##
## Call:
## glm(formula = Vote ~ Income, family = binomial(link = logit),
##     data = USA)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2699  -1.0162  -0.8998   1.2152   1.6199
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.3017     0.1828  -7.122 1.06e-12 ***
## Income        0.3033     0.0551   5.505 3.69e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1655.0  on 1221  degrees of freedom
## Residual deviance: 1623.5  on 1220  degrees of freedom
## AIC: 1627.5
##
## Number of Fisher Scoring iterations: 4
```

Recall that we are fitting the model:

$$\begin{aligned} Y &\sim \mathcal{B}]\nabla\backslash(p) \\ \log\left(\frac{p}{1-p}\right) &= \beta_0 + \beta_1 X \end{aligned}$$

- `(Intercept)` $= \beta_0 =$ -1.3
- `Income` $= \beta_1 = 0.303$

It is common to interpret variables according to some **central tendency** e.g at the central income category (i.e X=3):

$$\begin{aligned} P(\text{Republican vote at } X=3) &= \text{logit}^{-1}\left(-1.3 + 0.3 \times 3\right) \\ &= \frac{e^{-1.3+0.3\times 3}}{1+e^{-1.3+0.3\times 3}} \\ &= 0.4. \end{aligned}$$

We can also check for $X=4$ and $X=5$ (rich)

$$\begin{aligned} P(\text{Republican vote at } X=4) &= \text{logit}^{-1}\left(-1.3 + 0.3 \times 4\right) \\ &= \frac{e^{-1.3+0.3\times 4}}{1+e^{-1.3+0.3\times 4}} \\ &= 0.48. \end{aligned}$$

$$\begin{aligned} P(\text{Republican vote at } X = 5) &= \text{logit}^{-1}(-1.3 + 0.3 \times 5) \\ &= \frac{e^{-1.3+0.3\times 5}}{1 + e^{-1.3+0.3\times 5}} \\ &= 0.55. \end{aligned}$$

So there is a tendency for voters with higher incomes to prefer Republicans over Democrats.

An **increase** of 1 unit on the **income scale** results in a positive difference of 0.3 on the **logit** scale in support of Bush.

A convenient way to express this on the **probability scale** is to consider what effect a 1 unit change has close to the **central value**, e.g. between $X = 3$ and $X = 2$

$$\begin{aligned} &\text{logit}^{-1}(-1.3 + 0.3 \times 3) \\ &\quad - \text{logit}^{-1}(-1.3 + 0.3 \times 2) = 0.07. \end{aligned}$$

Hence an increase in income of 1 unit around this central value results in a 7% increase in the probability of supporting Bush.

3.7.1 The 'divide-by-four' rule

A useful *rule-of-thumb* can be given by the **'divide-by-four'** rule.

That is, the **maximum difference** in $P(Y = 1)$ (P(Republican vote) in our example) corresponding to a **unit** difference in X is given by $\beta/4$.

In this example, the **maximum difference** in P(Republican vote) corresponding to a **unit** difference in income is given by $0.3/4 = 0.076$ (or a 7.6% change).

3.7.2 Odds ratios

An common interpretation of logistic regression coefficients is in terms of **odds ratios**.

Odds:

$$\frac{P(\text{event happens})}{P(\text{event does not happen})}$$

Probability of a voter in income category 3 voting Republican is

$$\begin{aligned} P(\text{Republican vote for } X = 3) &= p_{X=3} \\ &= \text{logit}^{-1}(-1.3 + 0.3 \times 3) \\ &= 0.4. \end{aligned}$$

Odds:

$$\frac{p_{X=3}}{1 - p_{X=3}}$$

Odds of a voter in income category 3 voting Republican is 0.4 / 0.6 = 0.67.

Odds ratio:

$$\frac{\text{odds in one group}}{\text{odds in another group}}$$

e.g. odds ratio for voters in income category 3 voting Republican compared to voters in income category 2 is:

$$\frac{\text{odds of voting Republican when } X=3}{\text{odds of voting Republican when } X=2} = \frac{\frac{p_{X=3}}{1-p_{X=3}}}{\frac{p_{X=2}}{1-p_{X=2}}}.$$

Take **logs**:

$$\begin{aligned}\log\left(\frac{\frac{p_{X=3}}{1-p_{X=3}}}{\frac{p_{X=2}}{1-p_{X=2}}}\right) &= \log\left(\frac{p_{X=3}}{1-p_{X=3}}\right) - \log\left(\frac{p_{X=2}}{1-p_{X=2}}\right)\\ &= \beta_0 + (\beta_1 \cdot 3) - \beta_0 - (\beta_1 \cdot 2)\\ &= \beta_1(3-2)\\ &= \beta_1\end{aligned}$$

So β_1 is the **log-odds ratio** for voting Republican per unit increase in income, and e^{β_1} is the **odds ratio**. This measure does **not** rely on the **level** of income

In this example the **odds ratio** is $e^{0.3} = 1.35$.

Hence the odds of voting Republican increase by a factor of 1.35 per unit increase in income.

Odds ratios are a tricky thing to understand, and many people (including me) find them **unintuitive**.

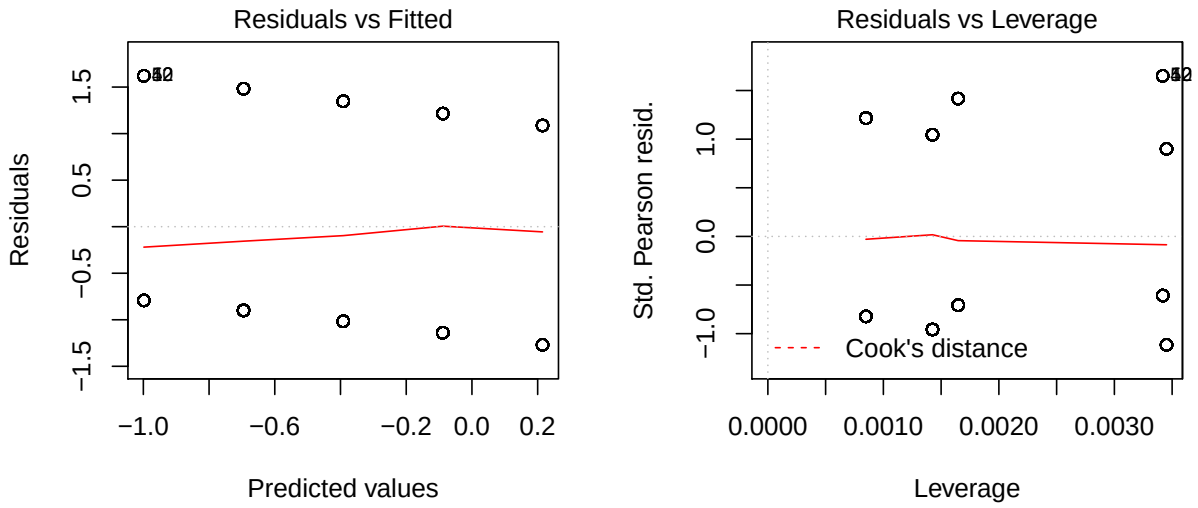
Are useful in **case-control** studies, where the prevalence of an outcome is unknown.

3.7.3 Model diagnostics

Once we have fitted a regression model, we can use it to predict the **mean** probability of success for given individual (**fitted values**).

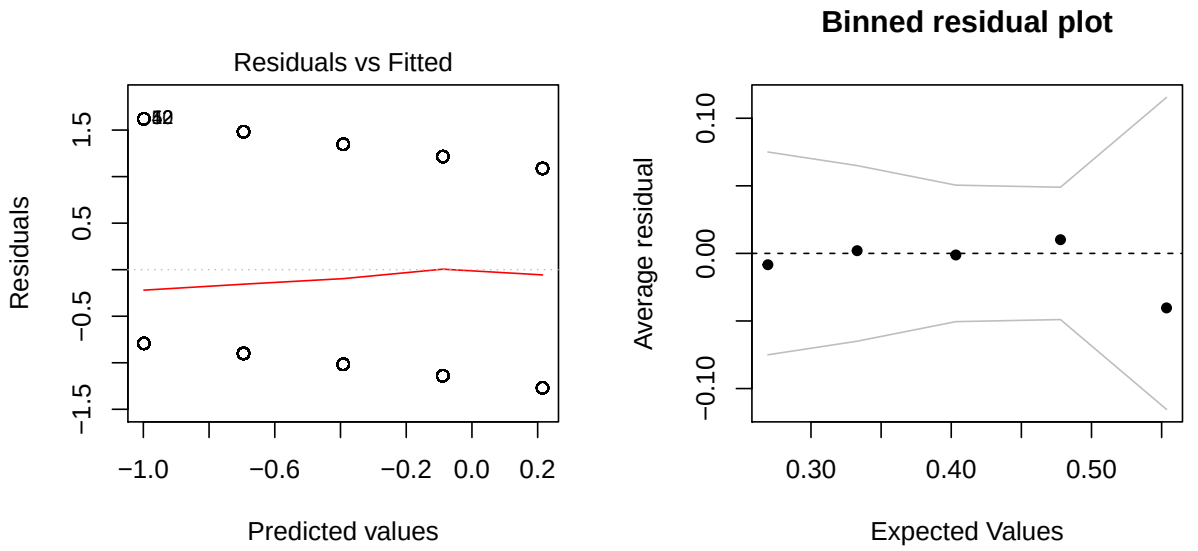
We can then generate **residual plots** as before:

```
plot(fit, which = c(1, 5))
```



Alternative **residual plots** are possible to generate using the `binnedplot()` function in the `arm` package in R:

```
USA$Vote <- ifelse(USA$Vote %in% 'Bill Clinton', 0, 1) # change to 0/1 for plot purposes
library(arm)
plot(fit, which = 1)
binnedplot(predict(fit, type = "response"),
           USA$Vote - predict(fit, type = "response"))
```



3.8 Practical 5 - Wine

The analysis determined the quantities of 13 constituents found in each of two types of wine. [5]

Download the data file from here and save it to your working directory.

```
wine <- read.csv("wine.csv", header=T)
```

```
head(wine)
```

```
##   WineType Alcohol MalicAcid  Ash AlcalinityAsh Magnesium TotalPhenols
## 1        A   14.23      1.71 2.43          15.6       127         2.80
## 2        A   13.20      1.78 2.14          11.2       100         2.65
## 3        A   13.16      2.36 2.67          18.6       101         2.80
## 4        A   14.37      1.95 2.50          16.8       113         3.85
## 5        A   13.24      2.59 2.87          21.0       118         2.80
## 6        A   14.20      1.76 2.45          15.2       112         3.27
##   Flavanoids NonflavanoidPhenols Proanthocyanins ColourIntensity  Hue
## 1       3.06                0.28            2.29            5.64 1.04
## 2       2.76                0.26            1.28            4.38 1.05
## 3       3.24                0.30            2.81            5.68 1.03
## 4       3.49                0.24            2.18            7.80 0.86
## 5       2.69                0.39            1.82            4.32 1.04
## 6       3.39                0.34            1.97            6.75 1.05
##   OD280_OD315 Proline
## 1        3.92    1065
## 2        3.40    1050
## 3        3.17    1185
## 4        3.45    1480
## 5        2.93     735
## 6        2.85    1450
```

The data columns are self-explanatory, but for the purpose of this practical we will just focus on `Alcohol` content and `TotalPhenols` (a class of chemical compound). The question of interest is:

> Can we differentiate between the two types of wine using `Alcohol` and `TotalPhenols`?

Task 12

1. Plot a scatterplot of `Alcohol` vs `TotalPhenols` and colour data points by `WineType`

Show Solution on P115

Task 13

2. Fit a logistic regression model with `WineType` as response variable and `Alcohol` and `TotalPhenols` as explanatory variables. Display a summary of the fit.

Show Solution on P115

[5] the full dataset contains three types of wine and is available here

Task 14

3. What is the probability that a wine with alcohol content of 12.5% and total phenols of 2.5 units, is of `WineType` B? Hint: Use the `invlogit` function already available in the `arm` package (`install.packages('arm')`)

Show Solution on P116

Task 15

4. Plot the mean regression lines on the **probability scale** for *varying* values of `Alcohol` but fixed values of `TotalPhenols` of 1.5, 2.5 and 3.5 (i.e a plot with `Alcohol` on the x-axis and predicted probability of `WineType` B on the y-axis for the three cases of `TotalPhenols`).

Show Solution on P117

3.9 Summary

GLMs are powerful and flexible.

They can be used to fit a wide variety of data types.

Model checking becomes trickier.

Extensions include:

- **mixed models**;
- **survival models**;
- **generalised additive models** (GAMs).

Chapter 4

Mixed effects models

My thanks in particular to Dave Hodgson, who's fantastic Master's course offered up the prototype for this workshop.

Slides can be downloaded from:

- (G)LMMs in R

Firstly, install if necessary (using `install.packages()`; uncomment the code below if required) and load the `lme4` package:

```
## load libraries
library(tidyverse)
library(lme4)
```

> **Note**: In this session I will use `tidyverse` as the principal way to generate plots etc. because it's ***far*** easier for many of the examples in this workshop. The base R code is provided for those of you that are not familiar with `tidyverse`.

This practical will focus on how to analyse data when the experimental design (or the surveyed explanatory variables) obliges us to study non-independent experimental units. You will find yourself distinguishing between random effects and fixed effects. You will find yourself frustrated at the lack of consensus regarding how to simplify fixed factors in mixed models. It takes a long time to understand how to deal with blocks (and other random effects): don't expect understanding to come overnight, and do rely on books and websites to help you. But be warned, at this level of statistical prowess, much of the literature is written in Greek symbols and matrix algebra.

Health Warning: some of these mixed-effects modelling packages are quickly evolving, and future updates might play some havoc with the coding, and outputs from, the scripts written here. You will need to keep your eye on the forums in the future if this happens. As always, let me or your demonstrators know if you encounter difficulties.

If this workshop has whetted your appetite, then a range of nice examples can be found at Julian Faraway's site: http://www.maths.bath.ac.uk/~jjf23/mixchange/index.html. These examples are discussed in his book: Extending the Linear Model with R.

4.1 A note on variable selection and model simplification

By design we have been a bit wary in focusing on the eponymous null hypothesis significance testing (NHST) ideas (and associated idolisation of p-values). We have done this because we want to emphasise that a statistical model is more than just a p-value, and also that a p-value on its own (without some measure of

effect size) is meaningless. We appreciate that this might fly in the face of the way that you might have seen statistical models presented in courses or papers, and this section is our attempt to explain to you some of the controversies surrounding these approaches.

There are many philosophical complexities with the way that p-values are used in many papers, and there have been many papers published discussing their misuse and misinterpretations. A good one is Halsey et al. (2015).

This is not to say that null hypothesis significance testing (hereafter NHST) is wrong, it's just that it's easy to misuse, even for experienced data modellers. Here are some common challenges / misconceptions:

1. A p-value is the *"probability of observing a test statistic equal to or more extreme than the observed test statistic if the null hypothesis is true"*; it is **not** the *"probability that the null hypothesis is false"*.
2. Leads to a binary interpretation: *"is there an effect?"*", rather than, *"what is the size of the effect"*?
3. They are heavily dependent on **sample size**:
    - a *larger* study with the same effect size and variance will be *more* statistically significant. Example: study is exploring differences between sample means from two groups, $\bar{x}$ and $\bar{y}$.
        - $\bar{x} = 10$, $\bar{y} = 12$, $s = 5$ (pooled SD), $n = 5$ (per group): $\mathbf{p = 0.4}$.
        - $\bar{x} = 10$, $\bar{y} = 12$, $s = 5$ (pooled SD), $n = 30$ (per group): $\mathbf{p = 0.0332}$.
4. The p-value itself is a **random variable**, and can have a very wide variance
    - Geoff Cumming's 'dance of the p-values'.
    - Type I, II, S and M errors.
5. Often, the null hypothesis of a **zero effect** is not a **biologically** valid one.
    - The difference between "significant" and "not significant" is not itself statistically significant...
6. Direction and magnitude of effects are important.
    - Consider a **large** study which finds a **small** effect size of x ($p < 0.001$), with the **minimum biologically significant** effect size being ***larger*** than x.
    - Therefore, the ***highly (statistically) significant*** p-value provides strong evidence of a **lack-of-effect**. It corresponds to a **negligible real-world** effect estimated with high precision!

Many of these problems go away when a study is appropriately **powered**, with careful experimental design. In this case the sample size is large enough to have high power for the comparison of interest, and the study is often set up in such a way as to make the comparison of interest meaningful. Note that these kinds of study were what NHST were derived to model.

That being said, there are times when NHST can be useful, particularly if you have a complex system (perhaps with large numbers of explanatory variables), and you wish to produce a more parsimonious model (perhaps because it is easier to interpret).

4.1.1 Common NHST approaches to model simplification

Traditionally, if the response variable is Gaussian (normal), then you may have come across two frequently used approaches:

- F-tests: based on comparing the residual mean squared error with the regression mean squared error, or
- Likelihood ratio tests (LRT): based on comparing the model deviance between two models.

Both of these cases are **exact** tests for linear regression with Gaussian errors (but for mixed models these become approximate). Let's have a look at a simple example using the fruitflies data from earlier practicals. These data are available in the "fruitfly.rds" file.

```
ff <- readRDS("fruitfly.rds")
```

```
ff_lm <- lm(longevity ~ type, data = ff)
summary(ff_lm)
```

```
##
## Call:
## lm(formula = longevity ~ type, data = ff)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -31.74 -13.74   0.26  11.44  33.26
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       63.560      3.158  20.130  < 2e-16 ***
## typeInseminated    0.520      3.867   0.134    0.893
## typeVirgin       -15.820      3.867  -4.091 7.75e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.79 on 122 degrees of freedom
## Multiple R-squared:  0.2051, Adjusted R-squared:  0.1921
## F-statistic: 15.74 on 2 and 122 DF,  p-value: 8.305e-07
```

```
anova(ff_lm, test = "F")
```

```
## Analysis of Variance Table
##
## Response: longevity
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## type        2  7845.3  3922.7  15.738 8.305e-07 ***
## Residuals 122 30407.5   249.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here the `anova()` function performs an F-test for the `longevity ~ type` model vs. the null model (`longevity ~ 1`). The idea is that *if the fit from the competing models is the same*, then the ratio of mean squared errors will follow an F-distribution on 2 and 122 degrees-of-freedom (d.f.) in this case.

Here we obtain a test statistic of 15.738, which can be compared against an F-distribution on 2 and 122 d.f., which gives a p-value of 8.3×10^{-7} (i.e. a statistically significant change even at the 1% and 0.1% levels). Thus we would conclude that dropping `type` produces a statistically significantly inferior model fit compared to when it is left in the model.

We can do the same thing but using a likelihood ratio test as:

```
anova(ff_lm, test = "Chisq")
```

```
## Analysis of Variance Table
##
## Response: longevity
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## type        2  7845.3  3922.7  15.738 8.305e-07 ***
## Residuals 122 30407.5   249.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here the test statistic produces a p-value of 5.9×10^{-7}; again highly statistically significant even at the 0.1%

level, but slightly different to the F-test (because they are different tests). Here we are comparing $-2\times$ the difference in log-likelihoods between the competing models, which under the null hypothesis that the fits are similar, should follow a chi-squared distribution on 2 d.f. here (the d.f. is the difference in the number of parameters between the two nested models). In general, in linear regression with Gaussian errors the F-test is slightly more powerful than the LRT, but you can use either.

For Generalised Linear Models (GLMs) with non-Gaussian error structure the F-test is no longer valid, and the LRT is **approximate** (the latter is in fact **asymptotically chi-squared**, which means that the approximation gets better for larger sample sizes, but can be misleading in small samples).

If using F-tests or LRTs to compare models, then the models must be **nested** (i.e. you can get one model to equal the other by fixing some of the parameters—usually setting coefficients to be zero, as in the fruitflies example). An alternative is to use something like Akaike's Information Criterion (AIC), which does not assess statistical significance and does not require the models to be nested (it is in essence a measure of predictive accuracy).

For **mixed models** things get trickier again, and there is much debate about optimal ways to perform model simplification and inference—see GLMM FAQ for more discussion. A nice description of the types of approaches we can use in different cases can be found at:

https://rdrr.io/cran/lme4/man/pvalues.html

A useful quote from the GLMM FAQ is:

> "For special cases that correspond to classical experimental designs (i.e. balanced designs that are nested, split-plot, randomized block, etc.) ... we can show that the null distributions of particular ratios of sums of squares follow an F distribution with known numerator and denominator degrees of freedom (and hence the sampling distributions of particular contrasts are t-distributed with known df). In more complicated situations (unbalanced, GLMMs, crossed random effects, models with temporal or spatial correlation, etc.) it is not in general clear that the null distribution of the computed ratio of sums of squares is really an F distribution, for any choice of denominator degrees of freedom."

The developers of `lme4` recommend (from **worst** to **best**)[1]:

- Wald Z-tests;
- For **balanced**, **nested LMMs** where degrees of freedom can be computed according to classical rules: Wald t-tests;
- Likelihood ratio test, either by setting up the model so that the parameter can be isolated/dropped (via `anova()` or `drop1()`, or via computing likelihood profiles;
- Markov chain Monte Carlo (MCMC) or **parametric bootstrap confidence intervals**.

If I do perform model simplification or variable selection, even then I would present the final model results in terms of effect sizes and confidence intervals where possible (or via predictive plots), since although CIs suffer with some of the same problems as p-values, at least they focus on the magnitude of the effects. If I have large enough sample sizes and not too many variables, then it may well be fine just to fit one model and perform inference from that.

In this workshop we will introduce some common scenarios in which mixed models can be applied, and give examples of model simplification and inference in these cases. We will tend to use LRTs and profile confidence intervals simply because they are easier to compute. A section at the end will explore the use of parametric bootstrapping for those of you who are interested.

[1] see GLMM FAQ

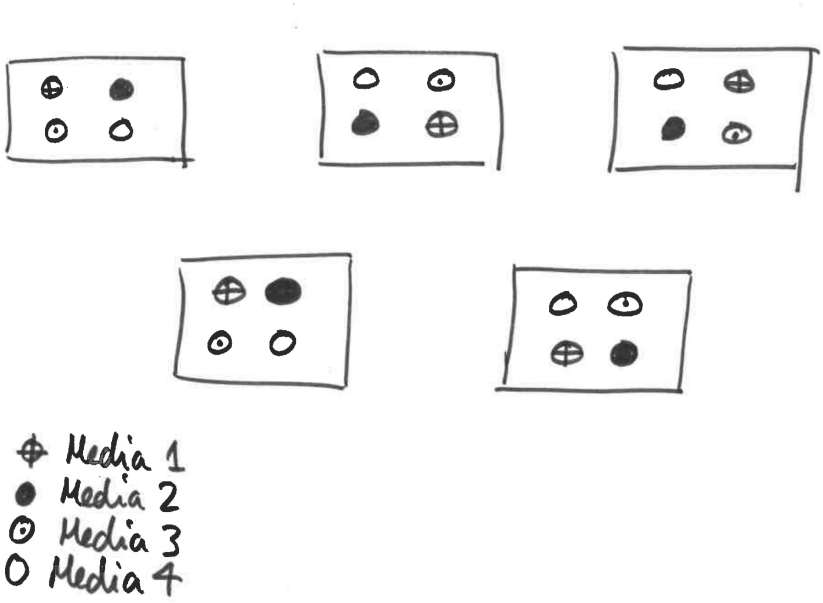


Figure 4.1: Schematic for bacterial growth example

4.2 Non-independent data: Randomised Complete Block Design

Often you will find it impossible to allocate treatments to experimental units at random because they are structured in time or in space. This means that units are not independent of each other, hence their residuals will not be independent, and we have violated an important assumption of generalised linear modelling (and, indeed, of analysis of variance). However the recognition of natural structuring in our population of experimental units can be **very** useful. If the experimental design can be stratified so that all units that share a common 'trait' (e.g. share a corner of a field, or share a single incubator) can represent one or more full replicate of the proposed experiment, then we should use this natural structuring to absorb some of our enemy: noise (also called residual deviance).

An extreme case of using error-absorbers is the Randomised Complete Block Experimental Design. So-called because it includes blocks, each block contains a single replicate of the experiment, and experimental units **within** blocks are allocated to treatment combinations **at random**. The concept is best explained using an example.

A microbiologist wishes to know which of four growth media is best for rearing large populations of anthrax, quickly. However, this poorly funded scientist does not own a large enough incubator in which to grow lots of replicate populations. Instead he requests space in five different incubators owned by other, better-funded researchers. Each incubator just has space for four bottles of medium. Our scientist allocates each growth medium to one bottle per incubator **at random**, inoculates with anthrax then monitors population growth rate. A schematic is given in Figure 4.1.

These data are available in the "bacCabinets.rds" file. Read in this dataset and familiarise yourself with its structure.

```
bac <- readRDS("bacCabinets.rds")
```

Let's do the analysis **wrongly** to begin with:

```
bac_lm <- lm(growth ~ media, data = bac)
drop1(bac_lm, test = "F")
```

```
## Single term deletions
##
## Model:
## growth ~ media
##        Df Sum of Sq    RSS    AIC F value  Pr(>F)
## <none>              152.38 48.613
## media   3    88.577 240.96 51.778  3.1002 0.05638 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question 16

Why is this wrong?

Show Answer on P117

Let's refit the model with a `cabinet` effect, and then test for the statistical significance of `media`:

```
bac_lm <- lm(growth ~ media + cabinet, data = bac)
drop1(bac_lm, test = "F")
```

```
## Single term deletions
##
## Model:
## growth ~ media + cabinet
##         Df Sum of Sq    RSS    AIC F value    Pr(>F)
## <none>                15.23 10.551
## media    3    88.577 103.81 42.936  23.264 2.734e-05 ***
## cabinet  4   137.150 152.38 48.613  27.016 6.380e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This is better. We need a `cabinet` effect here to deal with the fact that measurements within cabinets are not independent. Notice that the sum-of-squares for `media` is the same in this model compared to the one without `cabinet` in it. This is because we have a **balanced design** (i.e. the same number of measurements in each combination of explanatory variables). However, the F-statistic is different, and this is because this depends on the **residual sum-of-squares**, which is the second model is different due to the inclusion of `cabinet`, which soaks up some of the residual variance that can be attributed to differences between cabinets.

> **Note**: here the `cabinet` effect **must** be included in the model. It does not make sense for us to drop the `cabinet` effect and test for whether it should be included or not. It needs to be included **by design**! Hence we can ignore the `cabinet` line output by `drop1()`.

Task 17

Try with an **interaction** effect between `media` and `cabinet`. What happens and why?

Show Solution on P117

4.2.1 Mixed model approach

Let's re-do this analysis using a **mixed model**. If you haven't already, you will need to load the `lme4` package:

```
library(lme4)
```

Fixed and random effects:

- `media` is a **fixed** effect—we chose the media to be tested, each media has a specific identity, we want to estimate the differences in bacterial growth between different media;
- `cabinet` is a **random** effect—we don't care about the identity of each cabinet, each cabinet is sampled from a population of possible cabinets, we just want to predict and absorb the variance in bacterial growth rate explained by cabinet.

Now on with the analysis:

```
bac_lmer <- lmer(growth ~ media + (1 | cabinet), data = bac)
summary(bac_lmer)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: growth ~ media + (1 | cabinet)
##    Data: bac
##
## REML criterion at convergence: 68.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.2306 -0.5407  0.1088  0.4320  1.7696
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  cabinet  (Intercept) 8.255    2.873
##  Residual             1.269    1.127
## Number of obs: 20, groups:  cabinet, 5
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   5.5800     1.3801   4.043
## media2        1.7800     0.7125   2.498
## media3       -1.9600     0.7125  -2.751
## media4       -3.8400     0.7125  -5.389
##
## Correlation of Fixed Effects:
##        (Intr) media2 media3
## media2 -0.258
## media3 -0.258  0.500
## media4 -0.258  0.500  0.500
```

Syntax for `lmer`: The `lmer` command has two important sections:

- The first part corresponds to the 'fixed effects' part of the model (in this case, `growth ~ media`), which is identical to the model you would enter into `glm` if there were no problems of non-independence of data.
- The second part is where you enter the 'random effects' part of the model, where you tell R how the data is nested and where the non-independence lies. There is a whole world of possibilities here, from spatially autocorrelated variance/covariance matrices to independently

varying random slopes and intercepts. We keep it simple in this course. For this example, we assume no interaction between media and cabinet, hence we just want to predict and absorb the additive variance due to different cabinets. Hence our random effect term is `(1 | cabinet)`. This means: 'the intercept of our linear model will vary according to cabinet'.

Mathematically this corresponds to:

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \gamma_{C_i} + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma^2)$, the x_{i*} are dummy **binary** variables corresponding to each level of `media`. For example:

$$x_{i1} = \begin{cases} 1 & \text{if measurement } i \text{ has media} = 2 \\ 0 & \text{otherwise,} \end{cases}$$

and similarly for x_{i2} and x_{i3}. Here γ_{C_i} is the random effect term for each `cabinet` ($C_i = 1, \ldots, 5$). The reason why γ_{C_i} are denoted as **random** effects, is that we assume they come from some distribution such that $\gamma_j \sim N(0, \sigma^2_\gamma)$ (for $j = 1, \ldots, 5$). I promise this will be the last equation in these notes!

4.2.1.1 REML vs. ML

To fit these models we use an approach called **maximum likelihood** (**ML**). ML is a very general technique with desirable *asymptotic* properties. However, ML estimators can be **biased**, particularly for small sample sizes. In mixed models an alternative approach, known as **restricted maximum likelihood** (**REML**), can be used, which produces better estimates of the model coefficients, particularly for small sample sizes. However, the REML likelihood function is different to the ML likelihood function, and only makes sense for mixed models (since it's designed to estimate the variance components).

We may have to switch between these two approaches depending on what we want to do. In general, if you want to compare nested models that only differ in the random effects, then you can use either REML or ML. However, if you want to compare models that differ in their fixed effects, then for **balanced, nested** designs we can usually derive suitable tests based on REML fits. For **unbalanced** or **non-nested** designs you will need to use ML. Hence, if we are performing variable selection or model simplification, then we *may* have to switch to ML when comparing models, but then refit the model using REML in order to produce estimates of the coefficients.

> **Note:** for various reasons the authors of `lme4` do not provide p-values as standard from a call to `summary()` or `anova()`. We can perform a **likelihood ratio test** for the impact of dropping `media` from the fitted model. The simplest way to do this is to use the `drop1()` function:
>
> Since this is a **balanced design**, then we can use either REML or ML to generate appropriate likelihood ratio tests for assessing the impact of dropping **fixed** effects. In an **unbalanced** design, then we would need to refit the model using ML. The easiest way to do this is *in situ*, using the `update()` function and setting the argument `REML = F`. This does not change the original `bac_lmer` model object.
>
> Trying both approaches here should give the same result.

```
drop1(bac_lmer, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## growth ~ media + (1 | cabinet)
##        Df     AIC    LRT  Pr(Chi)
## <none>     85.544
## media   3 108.333 28.789 2.48e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
drop1(update(bac_lmer, REML = F), test = "Chisq")
```

```
## Single term deletions
##
## Model:
## growth ~ media + (1 | cabinet)
##        Df     AIC    LRT  Pr(Chi)
## <none>     85.544
## media   3 108.333 28.789 2.48e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We have used a **likelihood ratio test**[2] to assess whether the removal of `media` corresponds to a statistically significantly inferior model fit, which in this case it does at both the 5% and 1% levels. Hence we should leave `media` in the model. Once we have the final model, we can look at a summary of the model fit:

```
summary(bac_lmer)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: growth ~ media + (1 | cabinet)
##    Data: bac
##
## REML criterion at convergence: 68.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.2306 -0.5407  0.1088  0.4320  1.7696
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  cabinet  (Intercept) 8.255    2.873
##  Residual             1.269    1.127
## Number of obs: 20, groups:  cabinet, 5
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   5.5800     1.3801   4.043
## media2        1.7800     0.7125   2.498
## media3       -1.9600     0.7125  -2.751
## media4       -3.8400     0.7125  -5.389
##
## Correlation of Fixed Effects:
##        (Intr) media2 media3
## media2 -0.258
## media3 -0.258  0.500
## media4 -0.258  0.500  0.500
```

Note that this summary has been produced from the model fit using REML, even though the last LRT was conducted using the model fitted using ML—this is because we updated the model *in situ* within `drop1()`.

The 'intercept' of the `lmer` model is the mean growth rate in `media1` for an **average** cabinet. `lmer` output also gives you information criteria about the model, tells you the standard deviation of the random effects, correlations between levels of fixed effects, and so on.

[2] you could also use an F-test using the `Anova()` function in the `car` package in this particular case

Note: we use a likelihood ratio test (LRT) here, rather than an F-test. The F-test would be OK in this particular case, with Gaussian error structure and fully balanced, nested design. However, LRTs are more general, and thus for consistency I will tend to use these (or AIC) throughout where required.

We can derive confidence intervals using `confint()`, and also produce predictions from the model using `predict()`.

Task 18

Produce a 97% confidence interval for each regression coefficient of the model. What do each of these coefficients mean?

Show Solution on P118

Task 19

The `abrasion` data in the `faraway` package contains information on an experiment where four materials were fed into a wear testing machine and the amount of wear recorded. Four samples could be processed at the same time and the position of these samples may be important, but we really care about assessing wear rates between different materials. Multiple runs were performed. You can load the package and data as follows:

```
library(faraway)
data(abrasion)
```

Take a look at the data and the help file for abrasion (i.e. `?abrasion`). Fit an appropriate linear mixed model to these data and assess whether there is any statistical evidence for differences in wear rates between the different materials.

Show Solution on P119

4.3 Non-independent data: pseudoreplication, nested variance and derived variable analysis

We take an example from Sokal & Rohlf (1981). The experiment involved a simple one-way anova with 3 treatments given to 6 rats. The analysis was complicated by the fact that three preparations were taken from the liver of each rat, and two readings of glycogen content were taken from each preparation. This generated 6 pseudoreplicates per rat to give a total of 36 readings in all. A schematic is given in Figure 4.2.

Clearly, it would be a mistake to analyse these data as if they were a straightforward one-way anova, because that would give us 33 degrees of freedom for error. In fact, since there are only two rats in each treatment, we have only one degree of freedom per treatment, giving a total of 3 d.f. for error.

The variance is likely to be different at each level of this nested analysis because:

1. the readings differ because of variation in the glycogen detection method within each liver sample (measurement error);
2. the pieces of liver may differ because of heterogeneity in the distribution of glycogen within the liver of a single rat;
3. the rats will differ from one another in their glycogen levels because of sex, age, size, genotype, etc.;
4. rats allocated different experimental treatments may differ as a result of the fixed effects of treatment.

If all we want to test is whether the experimental treatments have affected the glycogen levels, then we are not interested in liver bits within rat's livers, or in preparations within liver bits. We could combine all the

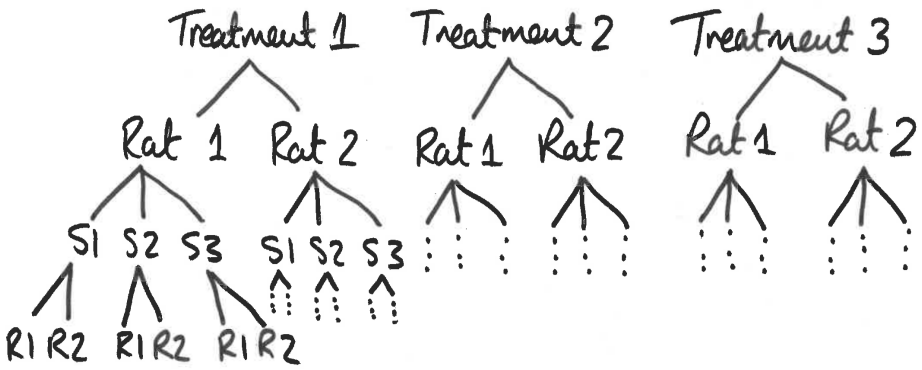


Figure 4.2: Schematic for glycogen in rats example

pseudoreplicates together, and analyse the 6 averages. This would have the virtue of showing what a tiny experiment this really was. This latter approach also ignores the nested sources of uncertainties. Instead we will use a linear mixed model.

The new concept here is that there are multiple random effects that are **nested**. These data are available in the "rats.rds" file. First, read in the data:

```
rats <- readRDS("rats.rds")
summary(rats)
```

```
##     Glycogen     Treatment Rat    Liver
##  Min.   :125.0   1:12      1:18   1:12
##  1st Qu.:135.8   2:12      2:18   2:12
##  Median :141.0   3:12             3:12
##  Mean   :142.2
##  3rd Qu.:150.0
##  Max.   :162.0
```

4.3.1 The Wrong Analysis

Here is what not to do (try it anyway)!

```
rats_lm <- lm(Glycogen ~ Treatment * Rat * Liver, data = rats)
```

The model has been specified as if it were a full factorial with no nesting and no pseudoreplication. Note that the structure of the data allows this mistake to be made. It is a very common problem with data frames that include pseudoreplication.

An ANOVA table gives:

```
anova(rats_lm, test = "F")
```

```
## Analysis of Variance Table
##
## Response: Glycogen
##                      Df  Sum Sq Mean Sq F value    Pr(>F)
## Treatment             2 1557.56  778.78 36.7927 4.375e-07 ***
## Rat                   1  413.44  413.44 19.5328 0.0003308 ***
## Liver                 2  113.56   56.78  2.6824 0.0955848 .
## Treatment:Rat         2  384.22  192.11  9.0761 0.0018803 **
## Treatment:Liver       4  328.11   82.03  3.8753 0.0192714 *
## Rat:Liver             2   50.89   25.44  1.2021 0.3235761
## Treatment:Rat:Liver   4  101.44   25.36  1.1982 0.3455924
## Residuals            18  381.00   21.17
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

> **Note**: I can only use the `anova()` function in this way because the data are **balanced** (or at least the model formulation above with no nesting and no pseudoreplication ends up treating the data as if they were balanced). In general you have to be very careful to use `anova()` in this way.

This says that there was a highly statistically significant difference between the treatment means, at least some of the rats were statistically significantly different from one another, and there was a statistically significant interaction effect between treatments and rats. **This is wrong!** The analysis is flawed because it is based on the assumption that there is only one error variance and that its value is 21.2. This value is actually the measurement error; that is to say the variation between one reading and another from the same piece of liver. For testing whether the treatment has had any effect, it is the rats that are the replicates, and there were only 6 of them in the whole experiment. Note also that the way the rats have been coded makes it look like there are only two rat "groups". Again this is wrong (see mixed models section below).

One way that we could analyse these data is to use a **derived variable** analysis.

4.3.2 Derived variables avoid pseudoreplication

The idea is to get rid of the pseudoreplication by averaging over the liver bits and preparations for each rat. A useful way of averaging, in R, is to use `aggregate` (or `tidyverse`):

tidyverse

```
ratsNew <- rats %>%
    group_by(Rat, Treatment) %>%
    summarise(Glycogen = mean(Glycogen))
ratsNew
```

```
## # A tibble: 6 x 3
## # Groups:   Rat [?]
##   Rat   Treatment Glycogen
##   <fct> <fct>        <dbl>
## 1 1     1             132.
## 2 1     2             150.
## 3 1     3             134.
## 4 2     1             148.
## 5 2     2             152.
## 6 2     3             136
```

Base R

```
ratsNew <- aggregate(Glycogen ~ Rat +
    Treatment, data = rats, mean)
ratsNew
```

```
##   Rat Treatment Glycogen
## 1   1         1 132.5000
## 2   2         1 148.5000
## 3   1         2 149.6667
## 4   2         2 152.3333
## 5   1         3 134.3333
## 6   2         3 136.0000
```

Here, `ratsNew` is a new dataset (hopefully shorter than its ancestral dataset) but has the same column headings as `rats`. (Hence be careful to specify the **correct** data set using the `data` argument where necessary.)

```
ratsNew_lm <- lm(Glycogen ~ Treatment, data = ratsNew)
drop1(ratsNew_lm, test = "F")
```

```
## Single term deletions
##
## Model:
## Glycogen ~ Treatment
##           Df Sum of Sq    RSS    AIC F value Pr(>F)
## <none>                 132.94 24.589
## Treatment  2    259.59 392.54 27.085   2.929 0.1971
```

This is a statistically valid analysis, but it ignores the uncertainties around the pseudo-replicates, and the interpretation of the response variable is actually the mean of a bunch of measurements, not the measurements themselves. In balanced designs this is OK, but it may not be possible to generate derived variables for some studies (how do you average a categorical response for example)?

4.3.3 Mixed model approach

4.3.3.1 Nested and crossed random effects

The bacterial loads example we saw earlier is an example of a **crossed** random effect i.e. the `cabinet` level `1` corresponds to the **same** cabinet for each of the `media` levels. In the `rats` data set this is no longer so, and we have to be careful to get the model formula correct. **This section is subtle but important!**

Notice that in the `rats` data the `Rat` column is coded as a `1` or a `2`. This means that we could use the formula:

```
rats_lmer <- lmer(Glycogen ~ Treatment + (1 | Rat), data = rats)
```

without throwing an error.

Question 20

Why is this wrong?

Show Answer on P120

In this case we must tell `lmer()` that `Liver` is **nested** within `Rat`, and `Rat` within `Treatment`. We can do this in various ways, but the easiest thing here is to generate a unique coding for each of the 6 rats.

tidyverse

```
rats <- rats %>%
    unite(Rat, Treatment, Rat,
          sep = "_", remove = F) %>%
    mutate(Rat = factor(
        as.numeric(factor(Rat))))
```

Base R

```
rats$Rat <- paste(rats$Treatment,
    "_", rats$Rat)
rats$Rat <- factor(
    as.numeric(factor(rats$Rat)))
```

We can use **nested** random effects to account for the hierarchy of measurements:

```
rats_lmer <- lmer(Glycogen ~ Treatment + (1 | Rat / Liver), data = rats)
drop1(update(rats_lmer, REML = F), test = "Chisq")
```

```
## Single term deletions
##
## Model:
## Glycogen ~ Treatment + (1 | Rat/Liver)
##           Df    AIC    LRT Pr(Chi)
## <none>       245.27
## Treatment  2 247.77 6.4962 0.03885 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note: the syntax `Rats / Liver` means that the `Liver` levels are **nested** within the `Rat` levels. In this case we do not have to recode `Liver` to be unique, since the nesting ensures that `lmer()` knows that `Liver = 1` inside `Rat = 1` is different from `Liver = 1` inside `Rat = 2`.

Task 21

The data for the alcohol example in the slides can be found in the "drunk.rds" file. Read this data set into R and:

1. Conduct a derived variable analysis, to test for the impact of `freshener` use on the model fit using a likelihood ratio test.
2. Produce 97% confidence intervals for the `freshener` effect.
3. Fit a mixed model using `lmer()`, taking care to get the nesting of the random effects correct.
4. Use a likelihood ratio test to assess the impact of `freshener` on the model fit. Does it matter whether you use REML or ML here, and why?
5. Produce an estimate of the `freshener` effect with 97% profile confidence intervals.
6. How do these estimates compare?

Show Solution on P121

4.4 Non-independent data: Split-plot analyses

Split-plot experiments are like nested designs in that they involve plots of different sizes and hence have multiple error terms (one error term for each plot size). They are also like nested designs in that they involve pseudoreplication: measurements made on the smaller plots are pseudoreplicates as far as the treatments applied to larger plots are concerned. This is spatial pseudoreplication, and arises because the smaller plots nested within the larger plots are not spatially independent of one another. The only real difference between nested analysis and split plot analysis is that other than blocks, all of the factors in a split-plot experiment are typically fixed effects, whereas in most nested analyses most (or all) of the factors are random effects.

This experiment involves the yield of cereals in a factorial experiment with 3 treatments, each applied to plots of different sizes within 4 blocks. The largest plots (half of each block) were irrigated or not because of the practical difficulties of watering large numbers of small plots. Next, the irrigated plots were split into 3 smaller split-plots and seeds were sown at different densities. Again, because the seeds were machine sown, larger plots were preferred. Finally, each sowing density plot was split into 3 small split-split plots and fertilisers applied by hand (N alone, P alone and N + P together). A schematic is given in Figure 4.3.

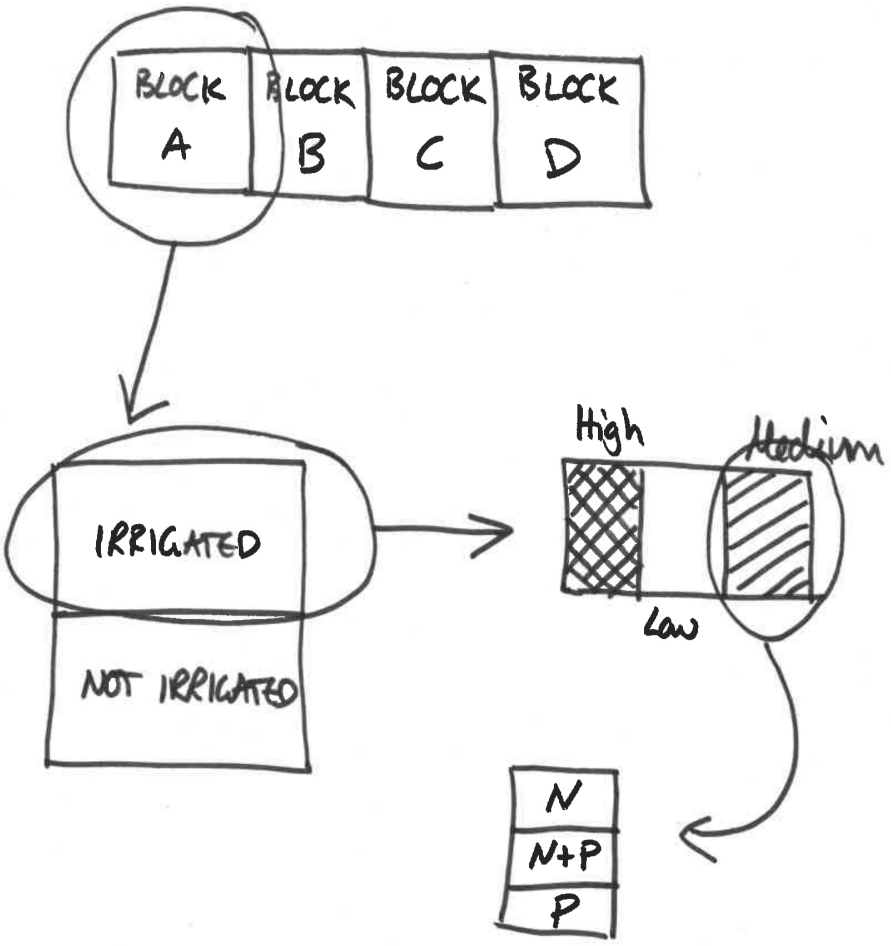


Figure 4.3: Schematic for split plot example

The data look like:

		Control			Irrigated		
	Density / Fertiliser	N	NP	P	N	NP	P
	high	81	93	92	78	122	98
Block A	medium	92	92	89	121	119	110
	low	90	107	95	80	100	87
	high	74	74	81	136	132	133
Block B	medium	98	106	98	99	123	94
	low	83	95	80	102	105	109
	high	82	94	78	119	136	122
Block C	medium	112	91	104	90	113	118
	low	85	88	88	60	114	104
	high	85	83	89	116	133	136
Block D	medium	79	87	86	109	126	131
	low	86	89	78	73	114	114

These data are available in the "splityield.rds" file. First, let's read in the data and take a look at it.

```
splityield <- readRDS("splityield.rds")
head(splityield)
summary(splityield)
```

```
##   yield block irrigation density fertilizer
## 1    90     A    control     low          N
## 2    95     A    control     low          P
## 3   107     A    control     low         NP
## 4    92     A    control  medium          N
## 5    89     A    control  medium          P
## 6    92     A    control  medium         NP
```

```
##      yield        block      irrigation   density   fertilizer
##  Min.   : 60.00   A:18   control  :36   high  :24   N :24
##  1st Qu.: 86.00   B:18   irrigated:36   medium:24   NP:24
##  Median : 95.00   C:18                  low   :24   P :24
##  Mean   : 99.72   D:18
##  3rd Qu.:114.00
##  Max.   :136.00
```

4.4.1 Analysing split-plot designs using `lmer`

Now, how do we set up a mixed effects model to analyse these data? `block` is the only random effect but our data are nested. Our fixed effects are `irrigation`, `density` and `fertilizer`. Here is the model, including prediction of the variance due to block's random effect on the intercept of the model.

Note: Ordinarily we would write the nested random effect terms using the syntax:

```
split_lmer <- lmer(yield ~ irrigation * density * fertilizer +
                     (1 | block / irrigation / density / fertilizer), data = splityield)
```

```
## Error: number of levels of each grouping factor must be < number of observations
```

However, if you run this, you will notice that the function returns an error. This is because there are no replicates within the final nesting (i.e. the table above on has a single replicate in each cell). To overcome this we need to remove the final nesting. Hence we can run:

```
split_lmer <- lmer(yield ~ irrigation * density * fertilizer +
                       (1 | block / irrigation / density), data = splityield)
```

> **Aside:** Another way of writing `(1 | block / irrigation / density / fertilizer)` is:
>
> `(1 | block) + (1 | block:irrigation) + (1 | block:irrigation:density) + (1 | block:irrigation:density:fertilizer)`.
>
> Pretty grim eh? However, in this case we note that the final nested random effect only has one replicate per group, so we could simply remove this term. Leaving:
>
> `(1 | block) + (1 | block:irrigation) + (1 | block:irrigation:density)`.
>
> This is equivalent to `(1 | block / irrigation / density)`

We can now perform model simplification (remembering to refit the model using ML in each case):

```
drop1(update(split_lmer, REML = F), test = "Chisq")
```

```
## Single term deletions
##
## Model:
## yield ~ irrigation * density * fertilizer + (1 | block/irrigation/density)
##                               Df    AIC    LRT Pr(Chi)
## <none>                           573.51
## irrigation:density:fertilizer  4 569.00 3.4938  0.4788
```

We can see that the three-way interaction term is not statistically significant, so we can drop it from the model and then test dropping the two-way interaction effects.

```
split_lmer <- update(split_lmer, ~ . - irrigation:density:fertilizer)
drop1(update(split_lmer, REML = F), test = "Chisq")
```

```
## Single term deletions
##
## Model:
## yield ~ irrigation + density + fertilizer + (1 | block/irrigation/density) +
##     irrigation:density + irrigation:fertilizer + density:fertilizer
##                       Df    AIC     LRT  Pr(Chi)
## <none>                   569.00
## irrigation:density     2 576.71 11.7088 0.002867 **
## irrigation:fertilizer  2 577.05 12.0424 0.002427 **
## density:fertilizer     4 565.19  4.1888 0.381061
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We are thus safe to remove the `density:fertilizer` interaction term:

```
split_lmer <- update(split_lmer, ~ . - density:fertilizer)
```

```
drop1(update(split_lmer, REML = F), test = "Chisq")
```

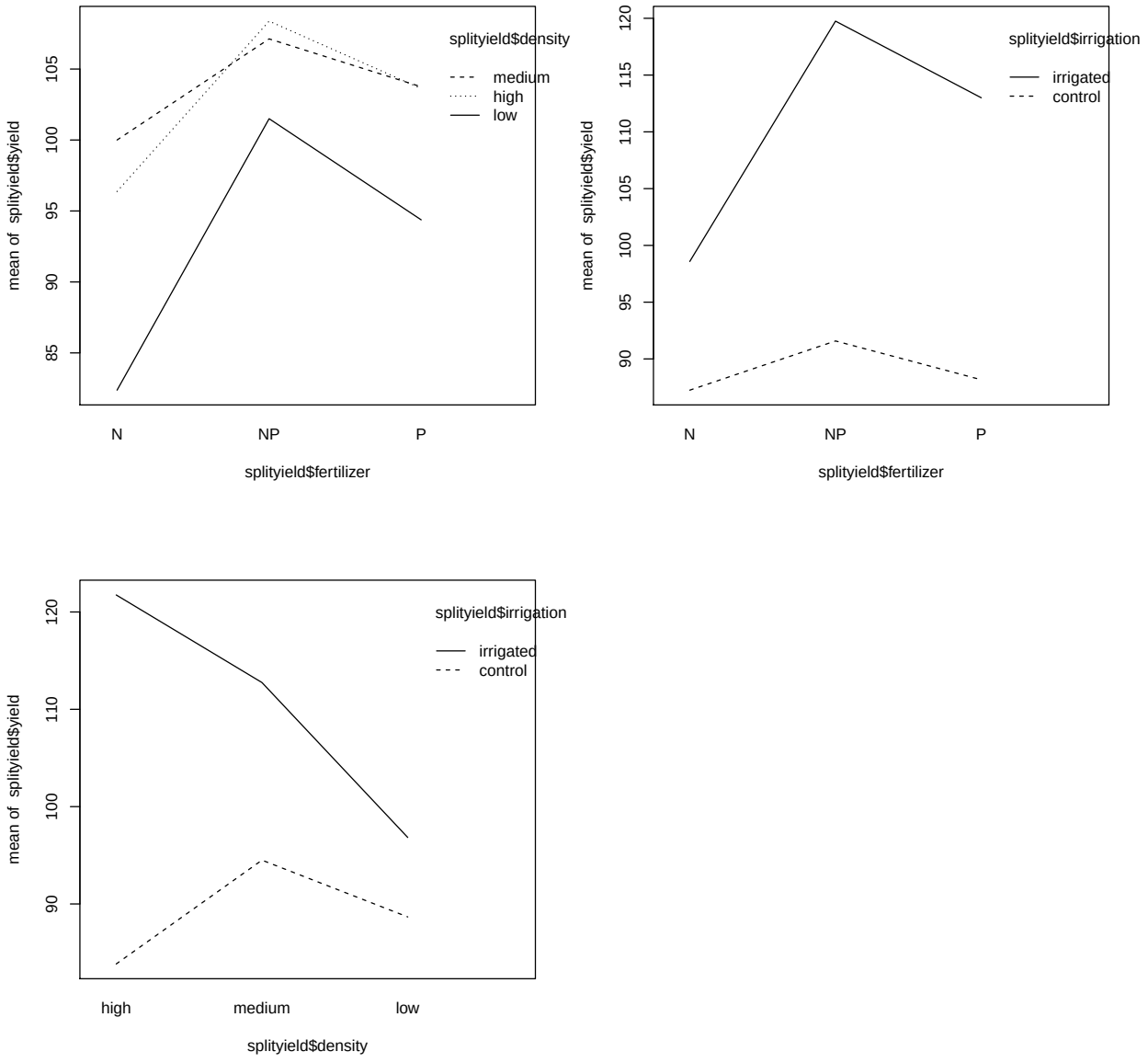
```
## Single term deletions
##
## Model:
## yield ~ irrigation + density + fertilizer + (1 | block/irrigation/density) +
##     irrigation:density + irrigation:fertilizer
##                        Df    AIC    LRT  Pr(Chi)
## <none>                    565.19
## irrigation:density      2 572.90 11.709 0.002867 **
## irrigation:fertilizer   2 572.34 11.144 0.003803 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We thus have a simplified final model. If you really want to know all the coefficients, type `summary(split.lmer)`, but here we will try to understand the interaction terms graphically.

A useful way to understand these is to use the `interaction.plot()` function. The variables are listed in a non-obvious order: first the factor to go on the x-axis, then the factor to go as different lines on the plot, then the response variable.

There are 3 plots to look at so we make a 2×2 plotting area:

```
par(mfrow = c(2, 2))
interaction.plot(splityield$fertilizer, splityield$density, splityield$yield)
interaction.plot(splityield$fertilizer, splityield$irrigation, splityield$yield)
interaction.plot(splityield$density, splityield$irrigation, splityield$yield)
par(mfrow = c(1, 1))
```



The really pronounced interaction is that between irrigation and density, with a reversal of the high to low density difference on the irrigated and control plots. Overall, the 3-way interaction was not statistically significant, nor was the 2-way interaction between density and fertilizer. The 2-way interaction between irrigation and fertilizer, however, was highly statistically significant.

Another way to visualise the model outputs would be to generate a plot of the predicted yield (plus confidence

intervals), for each combination of explanatory factors. I have created a `data.frame` with these predictions using bootstrapping. The code to replicate this can be found here, but is a bit time consuming to run, so I have included the predictions as a file called "split_pred.rds".

First, read in the predictions:

```
split_pred <- readRDS("split_pred.rds")
```

Now to plot these predictions (note, once again these ease of `ggplot2` compared to base R graphics):

tidyverse

In `ggplot2` we can use the `geom_pointrange()` function, a `colour` aesthetic and a `facet_wrap()`.

Base R

To plot the CIs in base R, we can proceed as follows:

1. Set the figure layout to have two figures side-by-side.
2. Calculate limits for the y-axis based on the range of the CIs to be plotted.
3. Calculate manual limits for the x-axis to enable us to jitter the error bars around each fertiliser level.
4. For each level of `irrigation`:
   - Extract the subset of data corresponding to the level of `irrigation`.
   - Produce an empty plot (setting the y-limits and x-limits). Remove the x-axis (using the `xaxt = "n"` argument). This latter step is so that we can add the `fertilizer` labels manually in the next step.
   - For each level of `density`:
     - Extract the subset of data corresponding to the level of `density`.
     - Sort this so that the levels of `fertilizer` are the same at each stage of the `i` loop[a].
     - Add points for the estimates and add confidence intervals using the `arrows()` function[b]. Notice that I have used a `for` loop to allow me to plot the three CIs for each level of `fertilizer`. All that this loop does is to run the `arrows()` command 3 times, substituting `j` each time for the corresponding row of the sorted data set (which will be of length three here). You could do this explicitly in three lines of code if you prefer. Notice that I've jittered the points slightly so that the three levels of `density` can be plotted side-by-side.
   - Add a new x-axis and set appropriate labels for the points.
   - Add a legend.

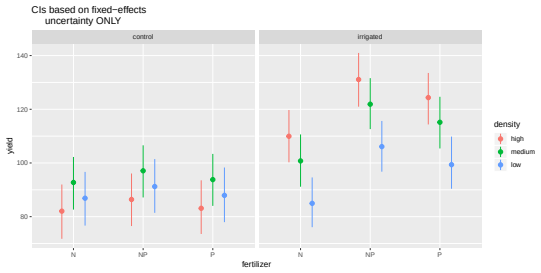
Phew! (Now check out the `tidyverse` solution and tell me it's not worth the effort to learn `ggplot2`!)

[a] at this point, `temp1` will have three rows, one for each level of `fertilizer`

[b] I'll let you check the help file for `arrows()` to figure out the arguments and what they are doing

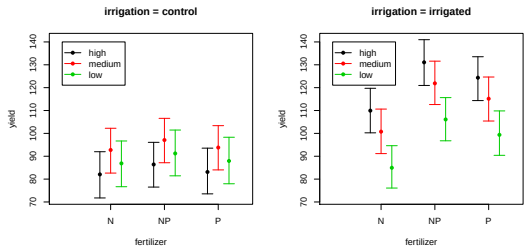
tidyverse

```
## plot confidence intervals
ggplot(newdata,
    aes(x = fertilizer, colour = density)) +
geom_pointrange(
    aes(y = yield, ymin = lci, ymax = uci),
    position = position_dodge(width = 0.5)) +
facet_wrap(~ irrigation) +
labs(title = "CIs based on fixed-effects
     uncertainty ONLY")
```



Base R

```
## set plot layout
par(mfrow = c(1, 2))

## set plot limits
ylims <- min(newdata[, -c(1:3)])
ylims <- c(ylims, max(newdata[, -c(1:3)]))
xlims <- c(0, 3.5)

for(m in levels(newdata$irrigation)) {
    ## extract control predictions
    temp <- newdata[newdata$irrigation == m, ]

    ## produce empty plot
    plot(NULL, ylim = ylims, xlim = xlims,
         xaxt = "n", main = paste("irrigation =", m),
         xlab = "fertilizer", ylab = "yield")

    ## plot yield against fertiliser stratified by density
    for(i in 1:length(levels(newdata$density))) {
        temp1 <- temp[
            temp$density == levels(newdata$density)[i], ]
        temp1 <- temp1[order(temp1$fertilizer), ]
        for(j in 1:nrow(temp1)){
            points(j - 0.2 * (2 - i), temp1$yield[j],
                col = i, pch = 16)
            arrows(j - 0.2 * (2 - i), temp1$lci[j],
                j - 0.2 * (2 - i), temp1$uci[j],
                length = 0.1, col = i,
                angle = 90, code = 3)
        }
    }

    ## add custom x-axis
    axis(side = 1, at = 1:3,
         labels = levels(newdata$fertilizer))

    ## add legend
    legend(0.1,
        diff(par("usr")[3:4]) * 0.95 + par("usr")[3],
        legend = levels(newdata$density),
        col = 1:3, lty = 1, pch = 16)
}
```




```
## reset plot layout
par(mfrow = c(1, 1))
```

So we can see that in the control group the `medium` density sowing seems to produce slightly higher yields on average than the `low` or `high` density settings, but we note that these differences are not statistically significantly different at the 5% level. However, irrigation tends to not only produce higher yields on average (except for the `N` fertiliser), but also to favour `high` density sowing efforts, which are now much preferred over `low` density sowing efforts (although not too dissimilar from `medium` density efforts). The model tends to favour the `NP` fertiliser in all cases, though there's not much between `NP` and `P`.

Note: This approach does now take into account the random effects variances or covariances. It is therefore likely to underestimate the variance. To account for these additional uncertainties is difficult. If you want a more consistent approach, go Bayesian.

4.5 Non-independent data: Absorbing the influence of random effects

Eight groups of students walked through Hell's Gate National Park and estimated the distance to groups of individuals of several species of mammalian herbivores. Here we work with data on distances to three key species: Thomson's gazelle, warthog and zebra. We are interested in whether distances to these animals differ among species, whether distances depend on animal group size, and also whether the relationship between distance and group size differs among species. These data are available in the "hg.rds" file.

A naïve analysis would ignore the identity of the student group, and therefore treat all the distance observations as independent. It would probably fit a `lm` of distance against species, number and the interaction between species and number. My own exploration of the data suggests that raw distances are skewed, and that a log-transformation improves homoscedasticity and normality of residuals. So before analysing we do this transformation.

```
hg <- readRDS("hg.rds")
```

```
hg$ldist <- log(hg$Distance)
hg_lm <- lm(ldist ~ Species * Number, data = hg)
drop1(hg_lm, test = "F")
```

```
## Single term deletions
##
## Model:
## ldist ~ Species * Number
##                Df Sum of Sq    RSS     AIC F value   Pr(>F)
## <none>                      212.51 -346.77
## Species:Number  2    5.3917 217.91 -339.19  5.7846 0.003305 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This suggests a statistically significant interaction between `Species` and `Number` ($F_{2,456} = 5.78$, $p = 0.003$).

```
summary(hg_lm)
```

```
##
## Call:
## lm(formula = ldist ~ Species * Number, data = hg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.32646 -0.39255  0.04353  0.44779  1.85666
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             5.4357462  0.0689700  78.813  < 2e-16 ***
## Specieswarthog         -0.4178168  0.1066872  -3.916 0.000104 ***
## Specieszebra           -0.1247300  0.1203792  -1.036 0.300685
## Number                  0.0009105  0.0118508   0.077 0.938790
## Specieswarthog:Number   0.0156732  0.0254035   0.617 0.537563
## Specieszebra:Number    -0.0742589  0.0238692  -3.111 0.001981 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6827 on 456 degrees of freedom
## Multiple R-squared:  0.09402,    Adjusted R-squared:  0.08409
## F-statistic: 9.465 on 5 and 456 DF,  p-value: 1.338e-08
```

The interaction appears to be driven by a decrease in distance with increasing group size, but only in zebras.

We can draw the relevant figure (notice that this is where `tidyverse` comes into its own, with much more concise code resulting in a better plot).

tidyverse

Generate 'fake' data in order to plot fitted lines:

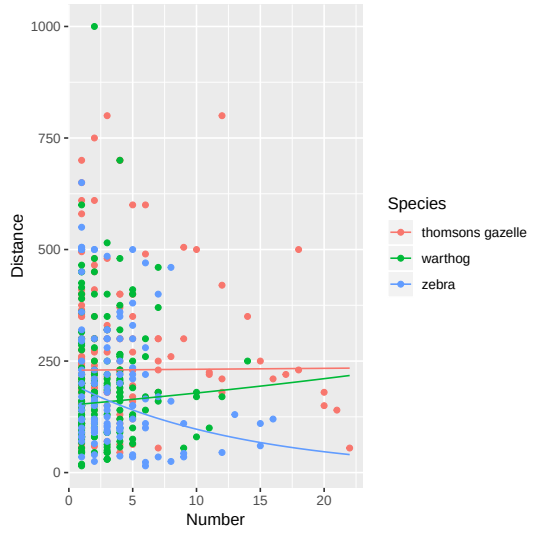
```
newdata <- expand.grid(
        Number = seq(min(hg$Number),
            max(hg$Number), length = 100),
        Species = levels(hg$Species)
    )
```

Now get the predicted values for the fake data, and back-transform log-distance to real distances, using the `exp` command:

```
newdata <- newdata %>%
    mutate(Distance =
        exp(predict(hg_lm, newdata)))
```

Now plot the fitted lines against the observed data points:

```
ggplot(hg, aes(x = Number, y = Distance,
    col = Species)) +
geom_point() +
geom_line(data = newdata)
```



Base R

Generate 'fake' data in order to plot fitted lines:

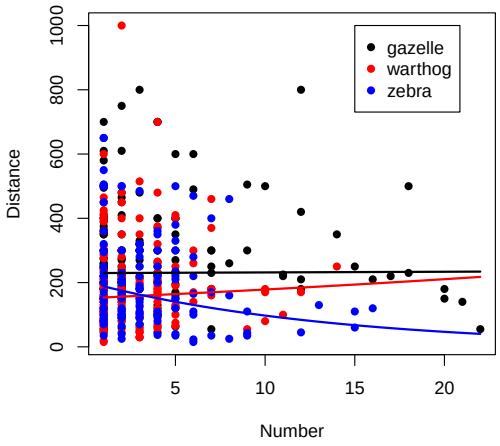
```
newdata <- expand.grid(
    Number = seq(min(hg$Number), max(hg$Number),
        length = 100),
    Species = levels(hg$Species)
)
```

Now get the predicted values for the fake data, and back-transform log-distance to real distances, using the `exp` command:

```
newdata$Distance <- exp(predict(hg_lm, newdata))
```

Now plot the fitted lines against the observed data points:

```
plot(Distance ~ Number, data = hg, type = "n")
points(Distance ~ Number, data = hg,
       subset = (Species == "thomsons gazelle"),
       pch = 16)
points(Distance ~ Number, data = hg,
       subset = (Species == "warthog"),
       pch = 16,
       col = "red")
points(Distance ~ Number, data = hg,
       subset = (Species == "zebra"),
       pch = 16,
       col = "blue")
lines(Distance ~ Number, data = newdata,
      subset = (newdata$Species == "thomsons gazelle"),
      lwd = 2)
lines(Distance ~ Number, data = newdata,
      subset = (newdata$Species == "warthog"),
      lwd = 2,
      col = "red")
lines(Distance ~ Number, data = newdata,
      subset = (newdata$Species == "zebra"),
      lwd = 2,
      col = "blue")
legend(15, 1000,
       pch = rep(16, 3),
       col = c("black", "red", "blue"),
       legend = c("gazelle", "warthog", "zebra"))
```



This naïve analysis is fundamentally flawed because the distance estimations are not independent of each other. Several estimates come from each of 8 student groups, and it's easy to imagine (or remember) that observers vary dramatically in the bias of their distance estimates (and indeed in their estimates of group size, possibly even species identity). It's entirely possible that a group of observers who estimate "short" distances also encountered strangely large groups of zebras: such a correlation between `Group.Name` and `Distance` could drive the observed relationship for zebras.

One solution would be to fit `Group.Name` as a categorical explanatory variable to our model. This approach has two main problems:

1. It wastes valuable degrees of freedom (8 student groups, therefore 7 d.f. as a main effect and 7 d.f. more for each interaction), giving the residual variance fewer d.f. and therefore weakening the power of our analysis.
2. It could easily result in a complicated minimal adequate model involving something that we don't really care about and which does not appear in our hypothesis: `Group.Name`.

Instead we would like to be able to absorb the variance in distance estimates among student groups. Such a model would estimate the relationship between `Distance`, `Species` and `Number` for an **average** group of observers. It could also tell us just how much variance exists among groups of observers.

This is a classic example of where a **mixed-effects model** is useful. We care about the influence of `Species` and `Number` on `Distance`. These are **fixed effects**. We know that `Group.Name` will have an influence on `Distance`, but the categories of `Group.Name` are uninformative to the lay reader and cannot be replicated in future field trips. Hence we treat `Group.Name` as a **random effect**. An ideal mixed-effects model will test the influence of `Species` and `Number` on `Distance`, meanwhile absorbing the influence of `Group.Name`.

```
hg_lmer <- lmer(ldist ~ Species * Number + (1 | Group.Name), data = hg)
```

Note that there is no nesting going on here; `Species` and `Number` have the same interpretation regardless of the `Group.Name`. Let's take a look at the summary of the fitted model.

```
summary(hg_lmer)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: ldist ~ Species * Number + (1 | Group.Name)
##    Data: hg
##
## REML criterion at convergence: 814.2
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.4891 -0.6386 -0.0128  0.5896  2.6806
##
## Random effects:
##  Groups     Name        Variance Std.Dev.
##  Group.Name (Intercept) 0.2085   0.4566
##  Residual               0.3046   0.5519
## Number of obs: 462, groups:  Group.Name, 8
##
## Fixed effects:
##                           Estimate Std. Error t value
## (Intercept)             5.4411458  0.1721299  31.611
## Specieswarthog         -0.3947870  0.0869880  -4.538
## Specieszebra           -0.2975063  0.0997439  -2.983
## Number                 -0.0001202  0.0099875  -0.012
## Specieswarthog:Number -0.0049066  0.0208474  -0.235
## Specieszebra:Number   -0.0289725  0.0199376  -1.453
```

```
##
## Correlation of Fixed Effects:
##             (Intr) Spcswr Spcszb Number Spcsw:N
## Speciswrthg -0.216
## Specieszebr -0.173  0.362
## Number      -0.235  0.415  0.318
## Spcswrthg:N  0.101 -0.710 -0.148 -0.460
## Spcszbr:Nmb  0.084 -0.196 -0.747 -0.416  0.200
```

This summary table is packed with information. First, we can see that the standard deviation in `Distance` among student groups is 0.46, which is almost as big as the residual standard deviation of 0.55. This suggests variation among observers is important and could dramatically change our results. Second, the estimates of slopes and differences between species are going in the 'same direction' as the estimates from the naïve `lm()` analysis, but seem to be smaller.

With **unbalanced** experimental designs like this, the fixed effects part of the model cannot be simplified in the usual way. This is because the removal of a fixed effect will **fundamentally** change the model fit: it's not just a case of dumping the deviance explained by a fixed effect into the residual deviance, because the residual deviance is different for each nested level of the experimental design. Furthermore, if the experiment is unbalanced, we should beware of using F-tests to check the statistical significance of terms. Instead, we should use AIC or **likelihood ratio tests**, and if we do this we need to change the fitting mechanism from REML to ML whilst performing LRT. (We do this *in situ* using the `update()` function.)

```
drop1(update(hg_lmer, REML = F), test = "Chisq")
```

```
## Single term deletions
##
## Model:
## ldist ~ Species * Number + (1 | Group.Name)
##                 Df    AIC    LRT Pr(Chi)
## <none>             800.99
## Species:Number  2 799.15 2.1605  0.3395
```

As usual, likelihood ratio tests in this context have a chi-squared distribution with d.f. equal to the difference in d.f. between the models. Here we have 8 - 6 = 2 d.f. So, here we find no evidence for an interaction between `Species` and `Number` ($X^2_2 = 2.16$, $p = 0.34$).

Since we have an **unbalanced** design here, we can continue model simplification by dropping each of the main effect terms from the current model and seeing which has the largest impact on model fit (in this case the model simplification will be different depending on the ***order*** in which variables are added/removed from the model).

```
## set baseline model from previous round
hg_lmer <- update(hg_lmer, ~ . - Species:Number)

## now drop terms compare to baseline model
drop1(update(hg_lmer, REML = F), test = "Chisq")
```

```
## Single term deletions
##
## Model:
## ldist ~ Species + Number + (1 | Group.Name)
##         Df    AIC    LRT   Pr(Chi)
## <none>     799.15
## Species  2 849.64 54.490 1.471e-12 ***
## Number   1 797.68  0.528    0.4674
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here dropping `Number` has little impact on the model fit, whereas dropping `Species` results in a statistically significantly inferior fit. Hence the next stage is to drop `Number`. Therefore our baseline model now has just the main effect for `Species`. The final stage is to drop `Species` and assess the change in fit relative to our current model.

Question 22

Why do we need to do this last step, since we assessed the impact of removing `Species` before?

Show Answer on P122

```
## set baseline model from previous round
hg_lmer <- update(hg_lmer, ~ . - Number)

## now drop terms compare to baseline model
drop1(update(hg_lmer, REML = F), test = "Chisq")
```

```
## Single term deletions
##
## Model:
## ldist ~ Species + (1 | Group.Name)
##         Df    AIC    LRT   Pr(Chi)
## <none>     797.68
## Species  2 847.64 53.965 1.913e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This suggests there is a clear effect of `Species` identity on `log(Distance)` ($X^2_2 = 54.0$, $p < 0.001$).

Now we'd like to see our model summary (notice that R has automatically kept the REML fit; since we only converted the model objects to use ML during the model comparison exercise):

```
summary(hg_lmer)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: ldist ~ Species + (1 | Group.Name)
##    Data: hg
##
## REML criterion at convergence: 797.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.4743 -0.6479 -0.0538  0.5904  2.6838
##
## Random effects:
##  Groups     Name        Variance Std.Dev.
##  Group.Name (Intercept) 0.2161   0.4649
##  Residual               0.3042   0.5516
## Number of obs: 462, groups:  Group.Name, 8
##
## Fixed effects:
##                 Estimate Std. Error t value
## (Intercept)     5.43619    0.17008  31.962
## Specieswarthog -0.41088    0.06060  -6.780
## Specieszebra   -0.40508    0.06627  -6.113
##
## Correlation of Fixed Effects:
##             (Intr) Spcswr
## Speciswrthg -0.179
## Specieszebr -0.164  0.466
```

This final summary tells us that the standard deviation among observer groups is almost as big as the residual standard deviation in distances. It tells us that both warthogs and zebra are closer to the observer than Thomson's gazelles, which are approximately $e^{5.44} = 230$ metres away on average. The number of animals in a group has negligible influence on distance given the uncertainties in the data.

We're now left with a final model that has a single categorical variable. We could provide a boxplot of the raw data to describe it, but in this instance it would be misleading because it would not differentiate between observer-level noise and residual noise. In this instance we will produce confidence intervals for the mean distance from the road for each species and plot them.

The simplest way to generate confidence intervals for each of the three parameters of the model is to refit the model three times, changing the baseline species each time and using the `confint` function to extract the confidence intervals (converting to the correct scale at the end). For consistency we'll use a bootstrap method to generate the intervals (using 500 replicates, which is the default, though you might want to change to a larger number in practice). (Note that you could use a `predict()` solution as in the earlier example, which is often easier for more complex predictions.)

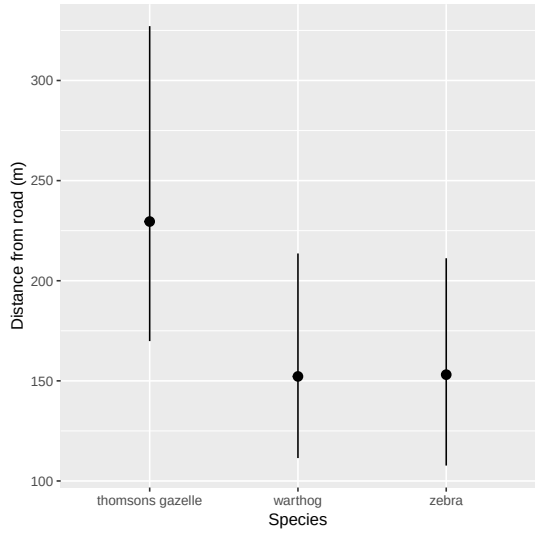
```
hg_cis <- matrix(NA, length(levels(hg$Species)), 3)
hg_spec <- levels(hg$Species)
for(i in 1:length(hg_spec)){
    hg$Species <- relevel(hg$Species, ref = levels(hg$Species)[i])
    hg_lmer <- update(hg_lmer, data = hg)
    hg_cis[i, ] <- c(
        coef(summary(hg_lmer))[ , "Estimate"][1],
        confint(hg_lmer, quiet = T, method = "boot")[3, ]
    )
}
hg_cis <- exp(hg_cis)
hg_cis <- as.data.frame(hg_cis)
colnames(hg_cis) <- c("mean", "lci", "uci")
hg_cis$Species <- hg_spec
hg_cis
```

```
##       mean      lci      uci          Species
## 1 229.5666 169.8597 327.1454 thomsons gazelle
## 2 152.2176 111.4347 213.6262          warthog
## 3 153.1030 107.7038 211.2413            zebra
```

tidyverse

In ggplot2 we can use the `geom_pointrange()` function.

```
ggplot(hg_cis, aes(x = Species)) +
    geom_pointrange(
        aes(y = mean, ymin = lci,
        ymax = uci)) +
    xlab("Species") +
    ylab("Distance from road (m)")
```

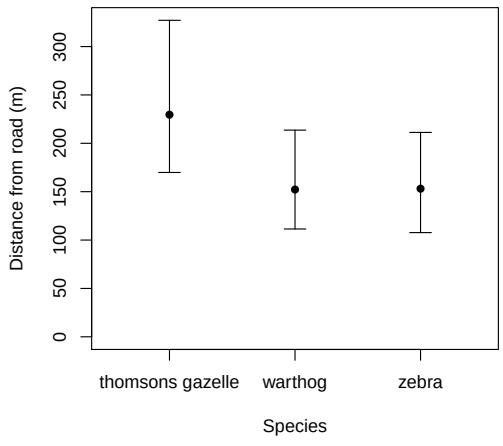


Base R

To plot the CIs in base R, we can proceed as follows:

1. Calculate limits for the y-axis based on the range of the CIs to be plotted.
2. Calculate manual limits for the x-axis to enable the species names to render correctly.
3. Produce a scatterplot of points defined on the ranges set above, but remove the x-axis (using the `xaxt = "n"` argument). This latter step is so that we can add the species labels manually in the next step.
4. Add a new x-axis and set appropriate labels for the points.
5. Add confidence intervals using the `arrows()` function. Notice that Ive used a `for` loop to allow me to plot the three CIs in one line of code as in the earlier example. You could do this explicitly in three lines of code if you prefer.

```
ylims <- c(0, max(hg_cis[, -ncol(hg_cis)]))
xlims <- c(0.5, 3.5)
plot(1:nrow(hg_cis), hg_cis$mean,
     xlab = "Species",
     ylab = "Distance from road (m)",
     xaxt = "n", pch = 16, ylim = ylims,
     xlim = xlims)
axis(1, 1:nrow(hg_cis), hg_cis$Species)
for(i in 1:nrow(hg_cis)) {
    arrows(i, hg_cis$lci[i],
           i, hg_cis$uci[i],
           code = 3,
           angle = 90,
           length = 0.1)
}
```



4.6 Model checking with mixed models

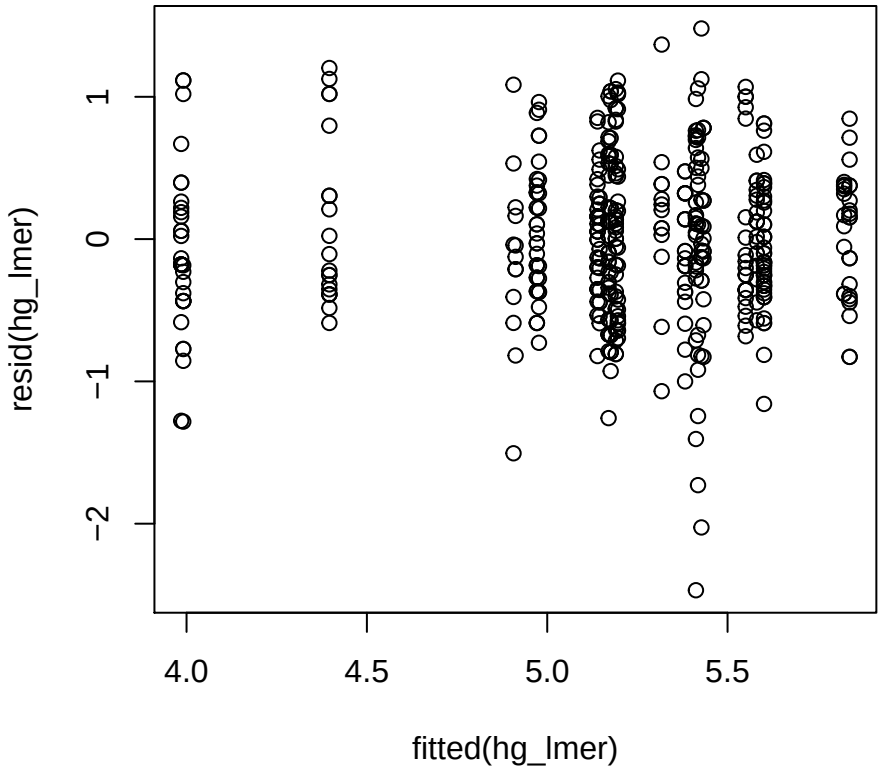
We need to be just as conscious of testing the assumptions of mixed effects models as we are with any other. The assumptions are:

1. Within-group errors are independent and normally distributed with mean zero and variance σ^2.
2. Within-group errors are independent of the random effects.
3. Random effects are normally distributed with mean zero.
4. Random effects have a covariance matrix Ψ that does not depend on the group... this is rather advanced.
5. Random effects are independent for different groups, except as specified by nesting... I'm not really sure what this means.

Several model check plots would help us to confirm/deny these assumptions, but note that QQ-plots may not be relevant because of the model structure. Two commonly-used plots are:

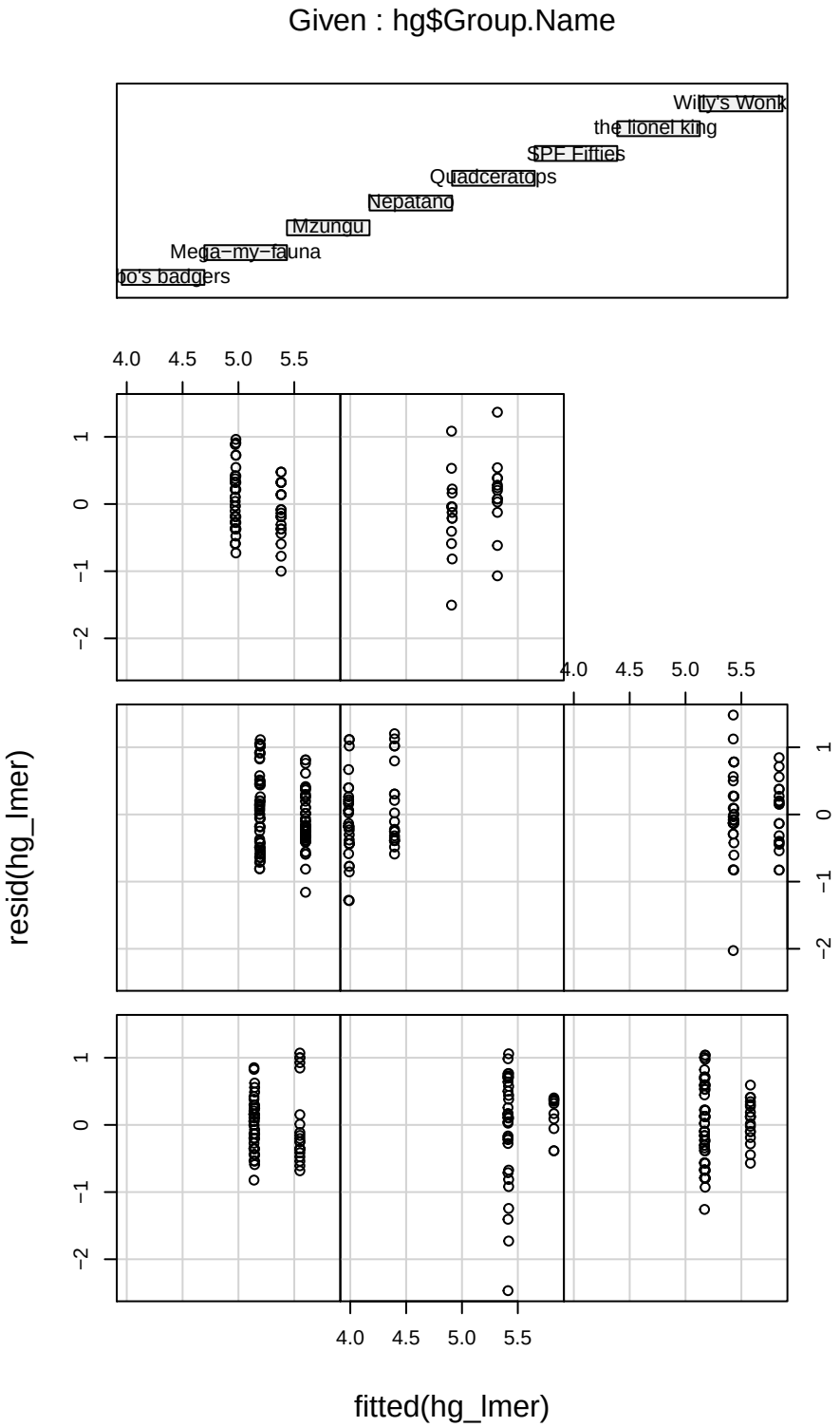
1. A simple plot of residuals against fitted values, irrespective of random effects (note we have to do this "by hand" here):

```
plot(resid(hg_lmer) ~ fitted(hg_lmer))
```



It can often be useful to check the distribution of residuals in each of the groups (e.g. blocks) to check assumptions 1 and 2. We can do this by plotting the residuals against the fitted values, separately for each level of the random effect, using the precious `coplot` function:

```
coplot(resid(hg_lmer) ~ fitted(hg_lmer) | hg$Group.Name)
```



You'll notice that each sub-figure of this plot, which refers to an individual group of observers, doesn't contain much data. Therefore it's hard to judge whether the spread of residuals around fitted values is the same for each observer group. But, with better-replicated experiments (i.e. with many more animals seen per observer

group) we could check that the residuals are homoscedastic both **within** and **among** observers. (Note that the order of the coplots relating to the groups denoted in the top plot is left-to-right, bottom-to-top: so `bilbo's badgers` are bottom-left, `Mega-my-fauna` are bottom-middle and so on...)

4.7 Why aren't GLMs much good at modelling non-independent data?

This is a short section, but very important. When you perform ANOVA or linear regression with a Gaussian (normal) error structure, you can use F-tests which are based on 'model' and 'residual' mean-squares or deviances (remember an F-statistic is the ratio of two variances). When data are non-independent, this is really useful because you can choose which part of the experimental design structure from which to select your residual deviance. Remember that you can also use likelihood ratio testing, in which case the chi-squared test statistic is an **exact** test (although subtlely different to the F-test).

When you perform Generalised Linear Modelling with non-normal error structures, you have to use likelihood ratio testing (i.e. chi-square tests). In the case of a standard GLM, then the LRT test statistic is **asymptotically chi-squared** (i.e. it's approximately chi-squared, where the approximation gets better as the sample size gets bigger). However, these are based only on 'model' deviance and do not take into account residual deviance. Hence a model of non-independent data cannot provide correct chi-square tests because they ignore the nesting in the experimental design.

Moral: It's far better to use **Generalised Linear Mixed Modelling** in these cases.

4.8 Generalised Linear Mixed Modelling (GLMM)

Obviously given your expertise with non-normal data and error structures, by now (if you're still reading) you'll be craving a mixed-effects modelling technique that copes with Poisson, binomial, quasi etc. errors: `glmer()` gives it a good try.

As an example, suppose our hypotheses change. Now we believe that the group sizes vary among species and with distance from the road. Using `Number` as a response variable is unlikely to have normal residuals (I'm confident). If this was a GLM, we would start by trying `family = poisson`, because the data are counts (close to zero). Miraculously, that's all we have to do in `glmer()` as well.

```
hometime <- glmer(Number ~ Species * ldist + (1 | Group.Name),
    data = hg, family = poisson)
summary(hometime)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: Number ~ Species * ldist + (1 | Group.Name)
##    Data: hg
##
##      AIC      BIC   logLik deviance df.resid
##   2241.8   2270.8  -1113.9   2227.8      455
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.9259 -0.9735 -0.4698  0.5877  9.6966
##
## Random effects:
##  Groups     Name        Variance Std.Dev.
##  Group.Name (Intercept) 0.128    0.3577
## Number of obs: 462, groups:  Group.Name, 8
##
## Fixed effects:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   2.50760    0.34501   7.268 3.64e-13 ***
## Specieswarthog               -1.84315    0.43850  -4.203 2.63e-05 ***
## Speciesthomsons gazelle      -1.11766    0.45641  -2.449 0.014332 *
## ldist                        -0.23485    0.06493  -3.617 0.000298 ***
## Specieswarthog:ldist          0.32048    0.08687   3.689 0.000225 ***
## Speciesthomsons gazelle:ldist  0.23311    0.08697   2.680 0.007354 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) Spcswr Spcstg ldist  Spcsw:
## Speciswrthg -0.571
## Spcsthmsnsg -0.502  0.413
## ldist       -0.920  0.607  0.533
## Spcswrthg:l  0.576 -0.988 -0.417 -0.629
## Spcsthgzll:  0.541 -0.435 -0.989 -0.591  0.451
```

Simplification to find the Minimal Adequate Model is... a challenge for you to complete!

Congratulations.

4.9 Parametric bootstrapping

Bootstrapping is a resampling technique that was designed for inference when it is not possible (or hard) to generate theoretical sampling distributions for particular quantities of interest. It was introduced by Bradley Efron in 1979. The name is based on the adage: "to pull oneself up by one's bootstraps".

The basic idea is that we can generate **empirical** sampling distributions for a quantity of interest (e.g. a median for example) by sampling from the underlying system a very large number of times and calculating the quantity in each case. This sampling distribution can be used directly to generate confidence intervals for example. Of course, in practice we cannot sample multiple times from the underlying system (we often only have one replicate—the **observed data**). Instead, bootstrapping aims to approximate this idea by instead **re-sampling with replacement** from the observed data, and then generating empirical bootstrapped sampling distributions. These ideas have been developed greatly over the years, but the technique is attractive

because it is simple to implement (although the estimates are often slightly biased). `lme4` has a function `bootMer()` that can be used to implement specific types of bootstrapping—in this case we want to use **parametric** bootstrapping. We do not dwell on the details here, but if you want more information please come and see me.

To generate the confidence intervals used in the split-plot design earlier, you can use the following code:

```
## new data to predict to
newdata <- expand.grid(
    density = levels(splityield$density),
    irrigation = levels(splityield$irrigation),
    fertilizer = levels(splityield$fertilizer)
)

## prediction function to use in bootstrap routine
predFun <- function(mod) {
  predict(mod, newdata = newdata, re.form = NA)
}

## produce 1000 bootstrapped samples
boot1 <- bootMer(split_lmer, predFun, nsim = 1000, type = "parametric", re.form = NA)

## function to produce percentile based CIs
sumBoot <- function(merBoot) {
    data.frame(
        yield = apply(merBoot$t, 2, function(x){
            mean(x, na.rm = T)
        }),
        lci = apply(merBoot$t, 2, function(x){
            quantile(x, probs = 0.025, na.rm = T)
        }),
        uci = apply(merBoot$t, 2, function(x){
            quantile(x, probs = 0.975, na.rm = T)
        })
    )
}

## bind CIs to newdata
split_pred <- cbind(newdata, sumBoot(boot1))
```

4.10 Bayesian modelling

Many of the challenges associated with **mixed effects** models go away if you move your inference into a **Bayesian** framework. That's not to say that other challenges do not arise in their place, mainly in terms of variable selection, however, in general I would recommend using a Bayesian framework for complex models with hierarchical structures, particularly spatio-temporal modelling.

For GLMMs there is an R package called `MCMCglmm` that will fit a wide range of models in a Bayesian framework. Alternatively there are general purpose Bayesian packages such as WinBUGS, OpenBUGS, JAGS or more recently Stan.

These approaches are beyond the scope of this course, but we are hoping to run a Bayesian Modelling workshop next year, so keep your ears to the ground!

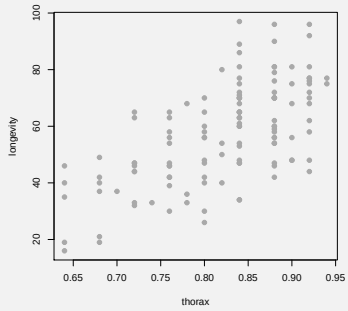
Appendix A

Answers

Solution 1

Base R

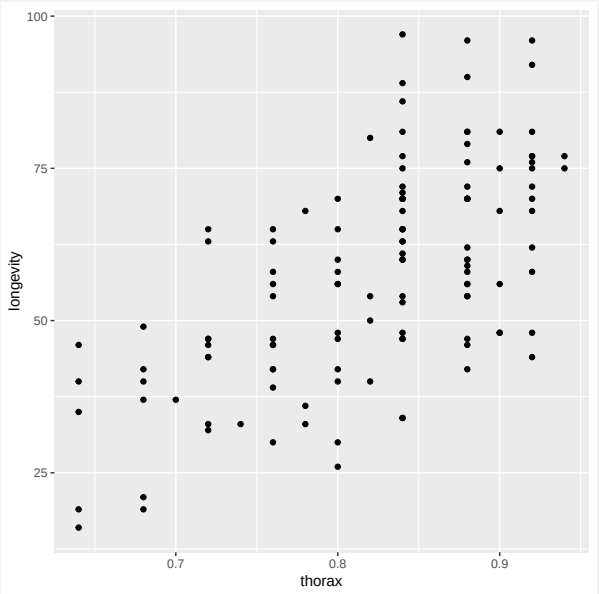
```
plot(longevity ~ thorax, data = ff,
     pch=19, col='darkgrey')
```



The plot suggests that a linear relationship might exist between the two variables. So we can proceed by fitting a linear model in R.

tidyverse

```
ggplot(ff) +
    geom_point(aes(x = thorax, y = longevity))
```



The plot suggests that a linear relationship might exist between the two variables. So we can proceed by fitting a linear model in R.

Return to task on P25

Solution 2

```
fit <- lm(longevity ~ thorax, ff)
summary(fit)
```

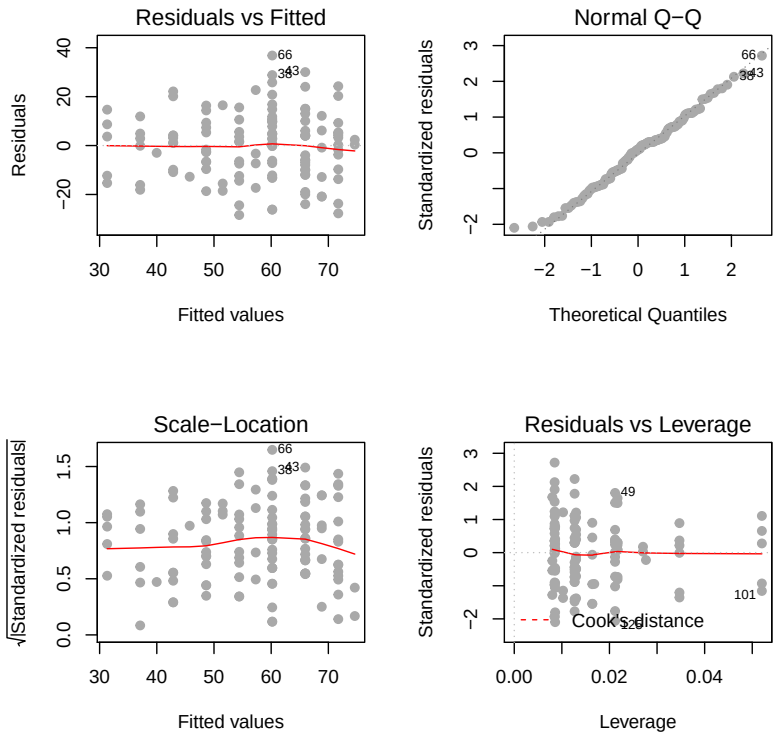
```
##
## Call:
## lm(formula = longevity ~ thorax, data = ff)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.415  -9.961   1.132   9.265  36.812
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -61.05      13.00  -4.695  7.0e-06 ***
## thorax        144.33      15.77   9.152  1.5e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.6 on 123 degrees of freedom
## Multiple R-squared:  0.4051, Adjusted R-squared:  0.4003
## F-statistic: 83.76 on 1 and 123 DF,  p-value: 1.497e-15
```

```
confint(fit, level=0.97)
```

```
##                 1.5 %   98.5 %
## (Intercept) -89.60261 -32.5008
## thorax      109.70817 178.9580
```

```
par(mfrow=c(2, 2))
plot(fit, pch=19, col='darkgrey')
```



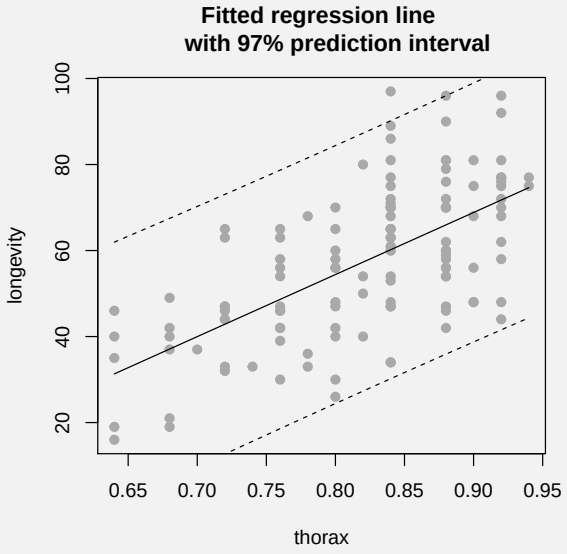

```
par(mfrow=c(1, 1))
```

Return to task on P25

Solution 3

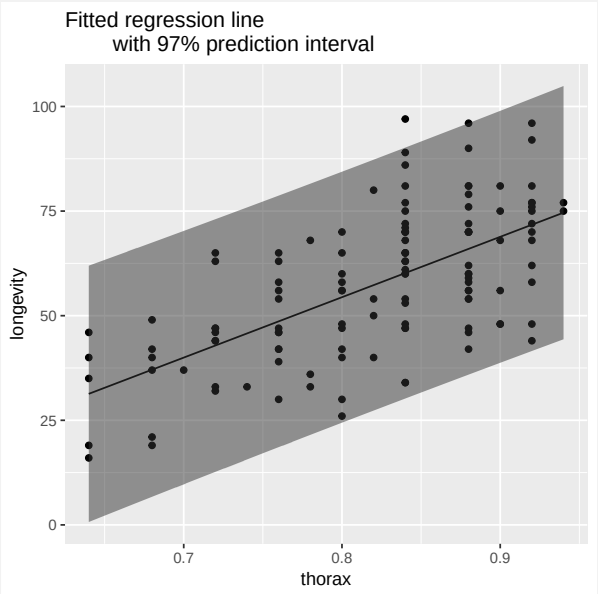
Base R

```
## create new data frame to predict to
newdata <- data.frame(
    thorax = seq(min(ff$thorax),
        max(ff$thorax), length.out = 50)
)

## produce predictions and intervals
newdata <- cbind(newdata,
    predict(fit, newdata,
        interval = "prediction",
        level = 0.97))
newdata$longevity <- newdata$fit
newdata$fit <- NULL

## plot fitted line against the raw data
plot(longevity ~ thorax, data = ff,
     pch = 19, col='darkgrey',
     main = "Fitted regression line
     with 97% prediction interval")
lines(longevity ~ thorax, data = newdata)
lines(lwr ~ thorax, data = newdata,
      lty = 2)
lines(upr ~ thorax, data = newdata,
      lty = 2)
```



tidyverse

```
## create new data frame to predict to
newdata <- data.frame(
    thorax = seq(min(ff$thorax),
        max(ff$thorax), length.out = 50)
)

## produce predictions and intervals
newdata <- cbind(newdata,
    predict(fit, newdata,
        interval = "prediction",
        level = 0.97)) %>%
    rename(longevity = fit)

## plot fitted line against the raw data
ggplot(mapping =
        aes(x = thorax, y = longevity)) +
    geom_point(data = ff) +
    geom_line(data = newdata) +
    geom_ribbon(aes(ymin = lwr, ymax = upr),
        data = newdata, alpha = 0.5) +
    ggtitle("Fitted regression line
        with 97% prediction interval")
```



Return to task on P29

Solution 4

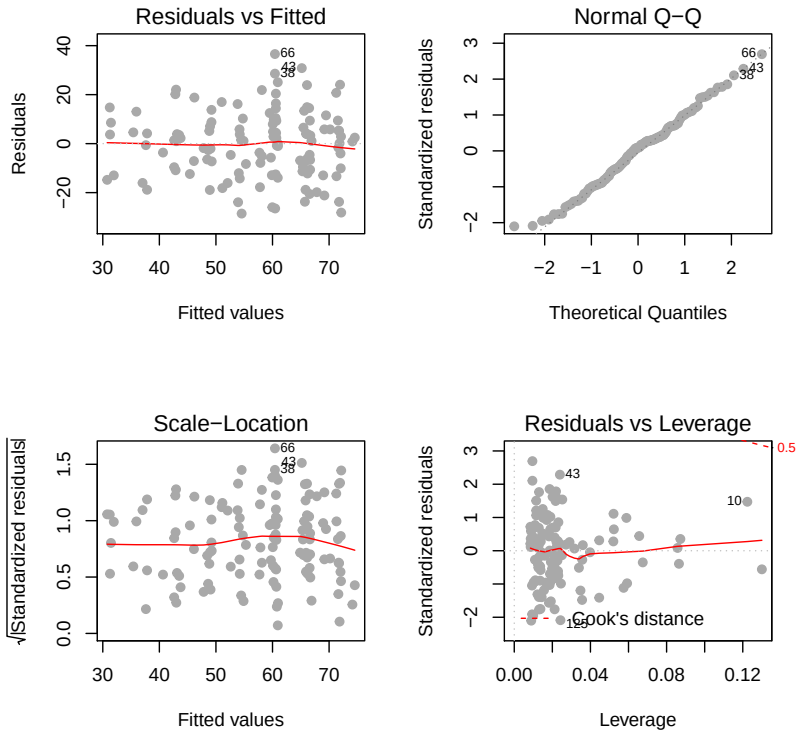
```
fit <- lm(longevity ~ thorax + sleep, ff)
summary(fit)

##
```

```
## Call:
## lm(formula = longevity ~ thorax + sleep, data = ff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.548 -10.062   1.116   8.787  36.572
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -60.53266   13.07649  -4.629 9.24e-06 ***
## thorax      144.89926   15.85031   9.142 1.68e-15 ***
## sleep        -0.04193    0.07731  -0.542    0.589
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.64 on 122 degrees of freedom
## Multiple R-squared:  0.4065, Adjusted R-squared:  0.3968
## F-statistic: 41.79 on 2 and 122 DF,  p-value: 1.502e-14
```

```
confint(fit, level=0.97)
```

```
##                   1.5 %      98.5 %
## (Intercept) -89.2456056 -31.8197076
## thorax      110.0956304 179.7028839
## sleep        -0.2116944   0.1278351
```

```
par(mfrow=c(2, 2))
plot(fit, pch=19, col='darkgrey')
```




```
par(mfrow=c(1, 1))
```

Practical significance of estimated parameters

The `intercept` parameter (-60.5 days) represents the expected longevity for a fruitfly that has a thorax length of zero and that spends no time sleeping. This is of course biologically implausible, but again acts as a reminder that the

linearity assumption is not necessarily true outside the range of the observed data

The `thorax` parameter (145 days/mm) is the expected increase in longevity for every 1mm increase in `thorax`. A 1mm increase in thorax length is huge and so for ease of interpretation we can instead state it as 14.5 days per 0.1mm increase in thorax length. The confidence intervals around this parameter is relatively narrow and always positive, suggesting that having a longer thorax is associated with an increased lifespan.

The `sleep` parameter (-0.0419days/%) is the expected increase in longevity for every 1% increase in `sleep`. This parameter has a standard error which is comparable to the parameter estimate itself, suggesting a huge uncertainty around this estimate. This is reflected in the 97% confidence intervals which spans from a negative to a positive value. It is therefore unlikely that `sleep` is associated with `longevity`.

Model fit

All of the diagnostic plots look sensible, suggesting that our model assumptions are satisfied.

Explained variation

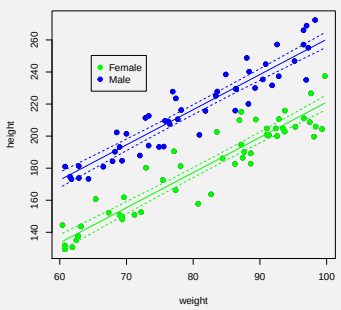
The R^2 statistic (also known as the **coefficient of determination**) is the proportion of the total variation that is explained by the regression. Which in this case is 40.7%. This is essentially the same as the previous model which only contained `thorax`, suggesting that `sleep` is not an important predictor of `longevity`.

Return to task on P34

Solution 5

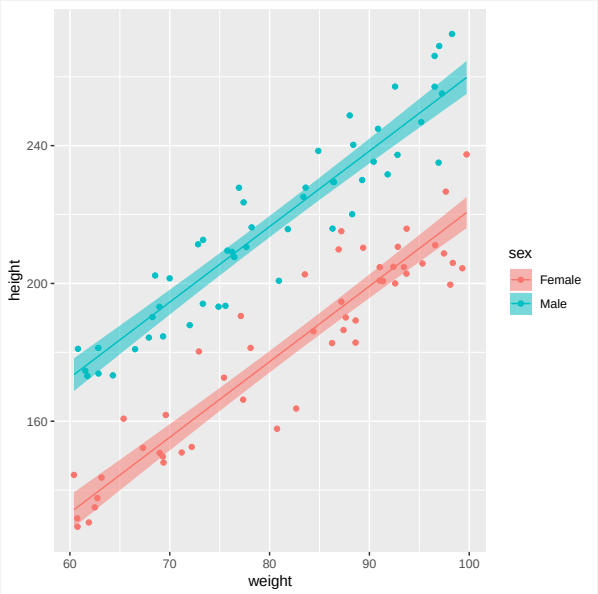
Base R

```
## create dataset to predict to
newdata <- expand.grid(
  weight = seq(min(df$weight),
    max(df$weight), length.out = 50),
  sex = c("Female", "Male"))

## generate predictions
newdata <- cbind(newdata,
  predict(fit, newdata, level = 0.97,
    interval = "confidence"))
newdata$height <- newdata$fit
newdata$fit <- NULL

## plot fitted lines against the data
plot(height ~ weight, data = df, pch = 19)
points(height ~ weight,
  pch = 19, col = "green",
  data = df[df$sex == "Female", ])
points(height ~ weight,
  pch = 19, col = "blue",
  data = df[df$sex == "Male", ])
lines(height ~ weight, col = "green",
  data = newdata[newdata$sex == "Female", ])
lines(height ~ weight, col = "blue",
  data = newdata[newdata$sex == "Male", ])
lines(lwr ~ weight,
  data = newdata[newdata$sex == "Female", ],
  col = "green", lty = 2)
lines(lwr ~ weight,
  data = newdata[newdata$sex == "Male", ],
  col = "blue", lty = 2)
lines(upr ~ weight,
  data = newdata[newdata$sex == "Female", ],
  col = "green", lty = 2)
lines(upr ~ weight,
  data = newdata[newdata$sex == "Male", ],
  col = "blue", lty = 2)
legend(par("usr")[1]*1.1, par("usr")[4]*0.9,
  pch = c(19, 19), col = c("green", "blue"),
  legend = c("Female", "Male"))
```



tidyverse

```
## create dataset to predict to
newdata <- expand.grid(
    weight = seq(min(df$weight),
        max(df$weight), length.out = 50),
    sex = c("Female", "Male"))

## generate predictions
newdata <- cbind(newdata,
    predict(fit, newdata,
        interval = "confidence",
        level = 0.97)) %>%
    rename(height = fit)

## plot fitted lines against the data
ggplot(mapping = aes(x = weight)) +
    geom_point(aes(y = height, colour = sex),
        data = df) +
    geom_line(aes(y = height, colour = sex),
        data = newdata) +
    geom_ribbon(aes(ymin = lwr, ymax = upr,
        fill = sex), data = newdata,
        alpha = 0.5)
```



Return to task on P42

Solution 6

By omitting `sex` from our model, we have no other alternatives but to treat this extra variation as **noise**, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. The unexplained variation (σ^2) is due to observation noise *and* sex. Once we put `sex` into our model, it will "soak" up the variation that is due to sex which improves our model considerably. The unexplained variation (σ^2) is now only due to the observation error (and other factors that we might be omitting).

Return to task on P42

Solution 7

```
fit <- lm(longevity ~ type + thorax, ff)
summary(fit)
```

```
##
## Call:
## lm(formula = longevity ~ type + thorax, data = ff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.330  -6.803  -2.531   7.143  29.415
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -56.521     11.149  -5.069 1.45e-06 ***
## typeInseminated     3.450      2.759   1.250    0.214
## typeVirgin        -13.349      2.756  -4.845 3.80e-06 ***
## thorax            143.638     13.064  10.995  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.21 on 121 degrees of freedom
## Multiple R-squared:  0.6024, Adjusted R-squared:  0.5925
## F-statistic:  61.1 on 3 and 121 DF,  p-value: < 2.2e-16
```

```
confint(fit, level=0.97)
```

```
##                       1.5 %     98.5 %
## (Intercept)     -81.005320 -32.037404
## typeInseminated  -2.609106   9.509537
## typeVirgin      -19.400571  -7.298282
## thorax          114.949413 172.326573
```

The type of companion parameters

In our toy example we introduced one dummy variable because the variable `sex` only had two factors (male or female). Type of companion has three factors (inseminated, virgin and control), we therefore need **two** dummy variables as follows:

$$S_{1i} = \begin{cases} 1 & \text{if } i \text{ is inseminated,} \\ 0 & \text{otherwise} \end{cases}$$

$$S_{2i} = \begin{cases} 1 & \text{if } i \text{ is virgin,} \\ 0 & \text{otherwise} \end{cases}$$

Here, control is the **baseline/reference level** for type of companion.

The mean regression line is:

$$y_i = \beta_0 + \beta_1 S_{1i} + \beta_2 S_{2i} + \beta_3 x_i$$

Or in English:

$$\text{longevity}_i = \beta_0 + \beta_1 \text{inseminated}_i + \beta_2 \text{virgin}_i + \beta_3 \text{thorax}_i$$

Where `inseminated` is a dummy variable that takes the value of 1 (inseminated) or 0 (otherwise) and `virgin` is another dummy variable that takes the value of 1 (virgin) or 0 (otherwise). For `control` both dummy variables are zero.

The mean regression lines for the three cases is as follows:

- **Control** (`inseminated=0` and `virgin=0`)

$$\text{longevity}_i = \beta_0 + (\beta_1 \times 0) + (\beta_2 \times 0) + \beta_3 \text{thorax}_i$$
$$\text{longevity}_i = \beta_0 + \beta_3 \text{thorax}_i$$

- **Inseminated** (`inseminated=1` and `virgin=0`)

$$\text{longevity}_i = \beta_0 + (\beta_1 \times 1) + (\beta_2 \times 0) + \beta_3 \text{thorax}_i$$
$$\text{longevity}_i = (\beta_0 + \beta_1) + \beta_3 \text{thorax}_i$$

- **Virgin** (`inseminated=0` and `virgin=1`)

$$\text{longevity}_i = \beta_0 + (\beta_1 \times 0) + (\beta_2 \times 1) + \beta_3 \text{thorax}_i$$
$$\text{longevity}_i = (\beta_0 + \beta_2) + \beta_3 \text{thorax}_i$$

The **intercept** for these three regression lines is therefore:

- **Control**: β_0 = `intercept`=-56.5 days
- **Inseminated**: $\beta_0 + \beta_1$ = `intercept` + `typeInesminated` = -56.5 + 3.45 = -53.1 days
- **Virgin**: $\beta_0 + \beta_2$ = `intercept` + `typeVirgin` = -56.5 + -13.3 = -69.9 days

The intercepts *per se* are not a useful measure as they represent the expected longevity of a fruitfly which has a thorax length of zero, which is clearly absurd. What is important is the *difference* between these intercepts as it represents a vertical shift in the regression line for the three conditions (see plot).

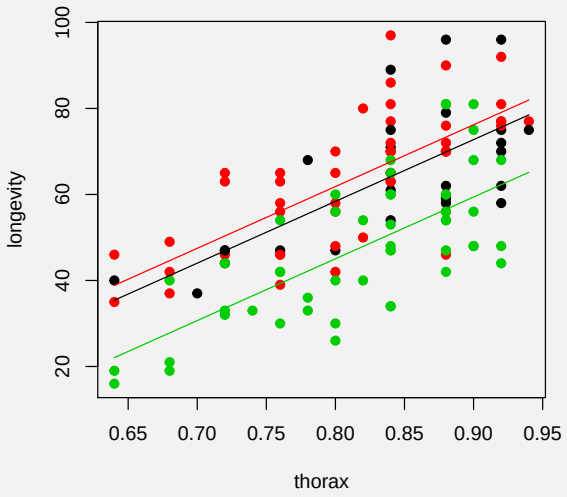
`typeInseminated` = 3.45 days is the expected increase in longevity for inseminated with respect to control for *any* thorax length. Similarly, `typeVirgin` = -13.3 days, is the expected increase in longevity for virgin with respect to control. We can also compute the expected increase in longevity for virgin with respect to inseminated which is `typeVirgin` - `typeInseminated` = -13.3 + 3.45 = -16.8 days.

Return to task on P43

Solution 8

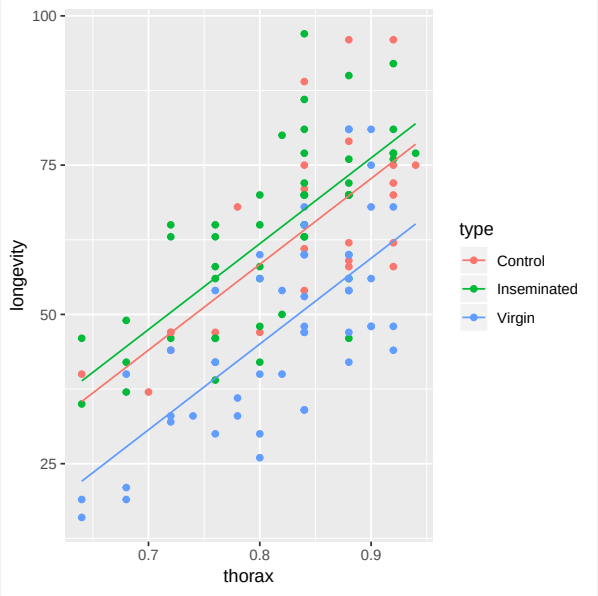
Base R

```
newdata <- expand.grid(
    thorax = seq(min(ff$thorax),
        max(ff$thorax), length.out = 50),
    type = levels(ff$type)
)
newdata <- cbind(newdata,
    longevity = predict(fit, newdata))
plot(longevity ~ thorax, data = ff,
     pch = 19, col = as.numeric(ff$type))
for(i in 1:3){
    lines(longevity ~ thorax,
        data = newdata[newdata$type ==
            levels(ff$type)[i], ],
        col = i)
}
```



tidyverse

```
newdata <- expand.grid(
    thorax = seq(min(ff$thorax),
        max(ff$thorax), length.out = 50),
    type = levels(ff$type)
)
newdata <- cbind(newdata,
    longevity = predict(fit, newdata))
ggplot(mapping = aes(x = thorax, y
        = longevity, colour = type)) +
    geom_point(data = ff) +
    geom_line(data = newdata)
```

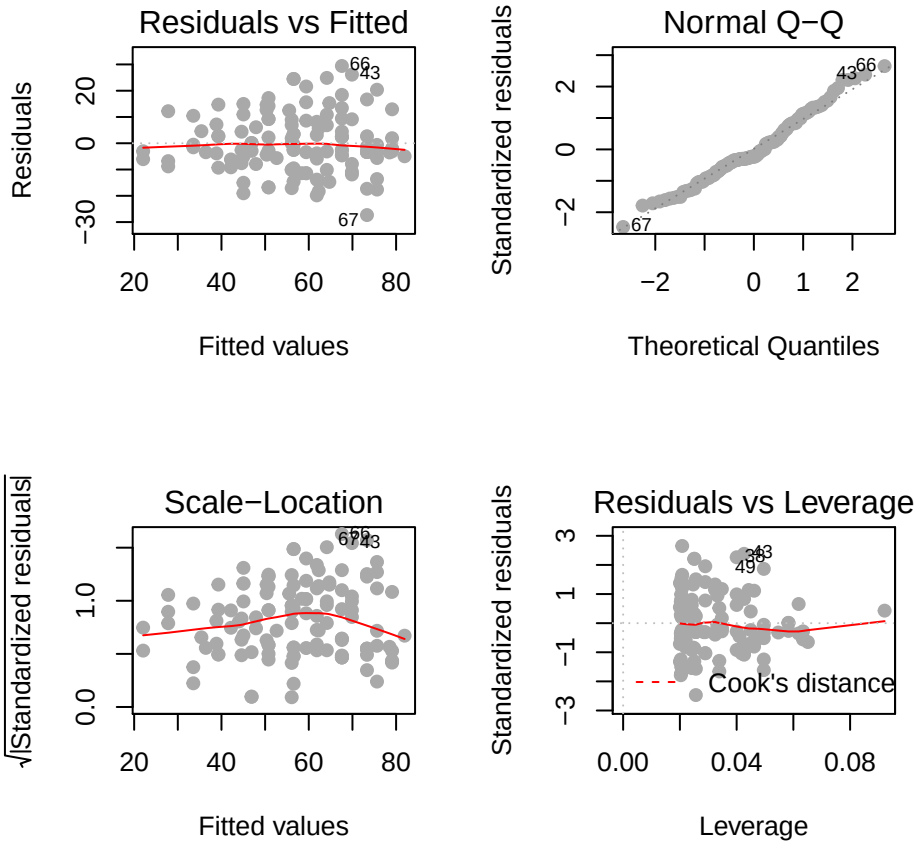


Return to task on P43

Solution 9

Model fit

```
par(mfrow=c(2, 2))
plot(fit, pch=19, col='darkgrey')
```




```
par(mfrow=c(1, 1))
```

All of the diagnostic plots look sensible, suggesting that our model assumptions are satisfied.

Explained variation

The R^2 statistic (also known as the **coefficient of determination**) is the proportion of the total variation that is explained by the regression. Which in this case is 60.2%. This is considerably higher than the 40% achieved with just `thorax`, suggesting that type of companion is indeed an important covariate and thus should be included in the model.

Return to task on P43

Solution 10

```
ff <- readRDS("fruitfly.rds")
```

```
fit_lm <- lm(longevity ~ type + thorax, ff)
summary(fit_lm)
```

```
##
## Call:
## lm(formula = longevity ~ type + thorax, data = ff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.330  -6.803  -2.531   7.143  29.415
##
## Coefficients:
```

```
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -56.521     11.149  -5.069 1.45e-06 ***
## typeInseminated    3.450      2.759   1.250    0.214
## typeVirgin       -13.349      2.756  -4.845 3.80e-06 ***
## thorax           143.638     13.064  10.995  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.21 on 121 degrees of freedom
## Multiple R-squared:  0.6024, Adjusted R-squared:  0.5925
## F-statistic:  61.1 on 3 and 121 DF,  p-value: < 2.2e-16
```

```
fit_glm <- glm(longevity ~ type + thorax, ff, family=gaussian)
summary(fit_glm)
```

```
##
## Call:
## glm(formula = longevity ~ type + thorax, family = gaussian, data = ff)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -27.330   -6.803   -2.531    7.143   29.415
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -56.521     11.149  -5.069 1.45e-06 ***
## typeInseminated    3.450      2.759   1.250    0.214
## typeVirgin       -13.349      2.756  -4.845 3.80e-06 ***
## thorax           143.638     13.064  10.995  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 125.7095)
##
##     Null deviance: 38253  on 124  degrees of freedom
## Residual deviance: 15211  on 121  degrees of freedom
## AIC: 964.92
##
## Number of Fisher Scoring iterations: 2
```

The model fits are exactly the same

Return to task on P46

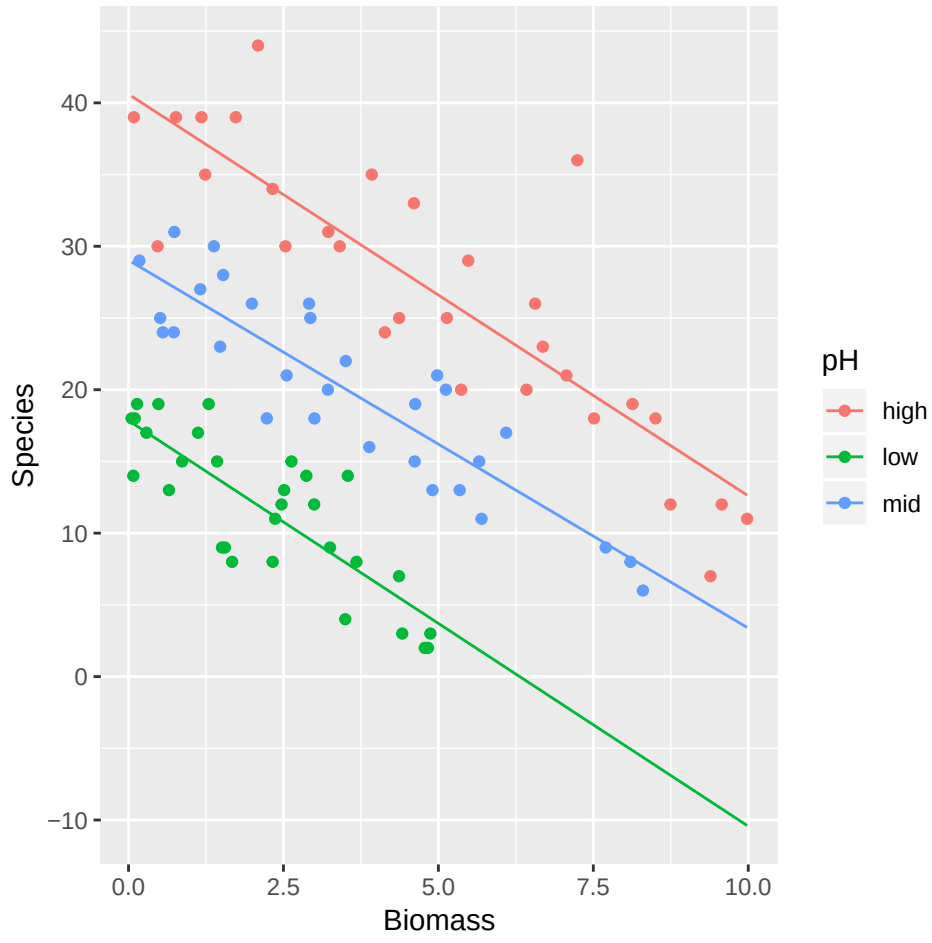
Solution 11

```
## Simple linear regression model

## Model fit
fit <- lm(Species ~ Biomass*pH, data=df)
summary(fit)
```

```
##
## Call:
## lm(formula = Species ~ Biomass * pH, data = df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
```

```
## -9.290 -2.554 -0.124  2.208 15.677
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    40.60407    1.36701  29.703  < 2e-16 ***
## Biomass        -2.80045    0.23856 -11.739  < 2e-16 ***
## pHlow         -22.75667    1.83564 -12.397  < 2e-16 ***
## pHmid         -11.57307    1.86926  -6.191  2.1e-08 ***
## Biomass:pHlow  -0.02733    0.51248  -0.053    0.958
## Biomass:pHmid   0.23535    0.38579   0.610    0.543
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.818 on 84 degrees of freedom
## Multiple R-squared:  0.8531, Adjusted R-squared:  0.8444
## F-statistic: 97.58 on 5 and 84 DF,  p-value: < 2.2e-16
```

```
## Plot mean regression lines
newdata <- expand.grid(Biomass=seq(min(df$Biomass), max(df$Biomass), length.out=200),
                       pH=levels(df$pH))
newdata <- cbind(newdata, Species=predict(fit, newdata, type='response'))


ggplot(mapping=aes(x=Biomass, y=Species, colour=pH)) + geom_point(data=df) +
    geom_line(data=newdata)
```

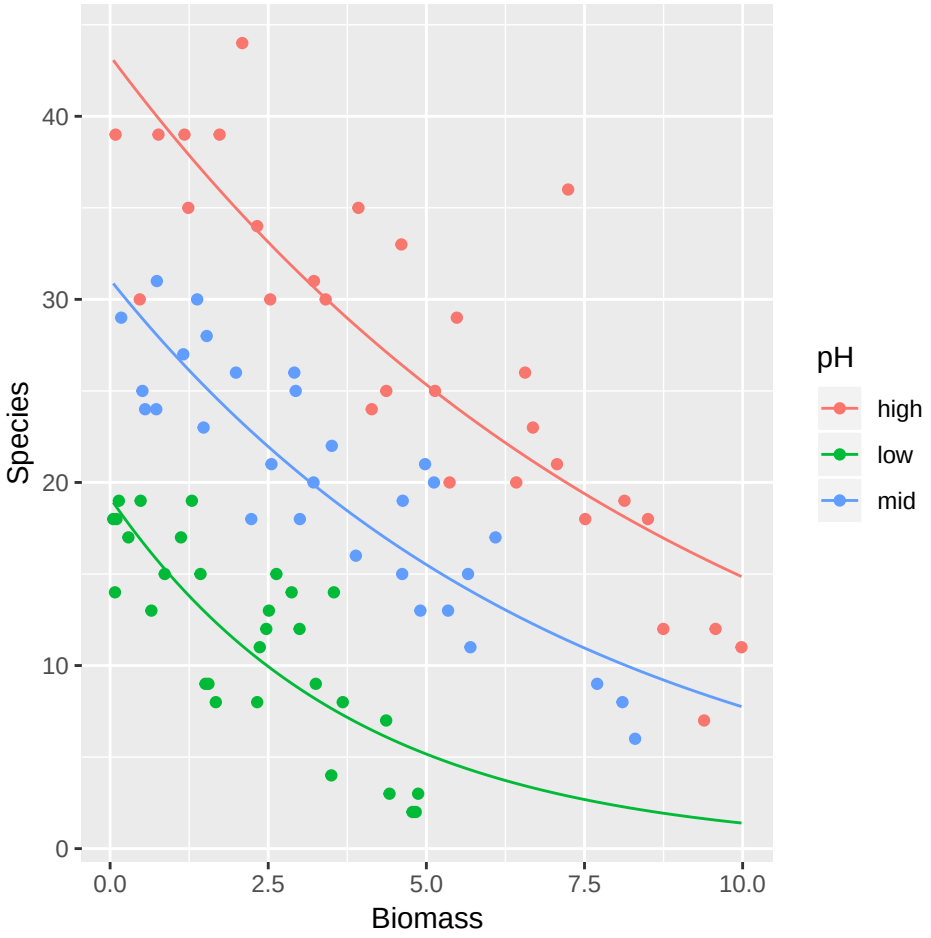


As `Biomass` increases the expected number of species tends towards a negative number for all three `pH` levels, which is not biologically plausible.

```r
## Poisson regression model

## Model fit
fit <- glm(Species ~ Biomass*pH, data=df, family=poisson(link=log))
summary(fit)
```

```
##
## Call:
## glm(formula = Species ~ Biomass * pH, family = poisson(link = log),
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4978  -0.7485  -0.0402   0.5575   3.2297
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    3.76812    0.06153  61.240  < 2e-16 ***
## Biomass       -0.10713    0.01249  -8.577  < 2e-16 ***
## pHlow         -0.81557    0.10284  -7.931 2.18e-15 ***
## pHmid         -0.33146    0.09217  -3.596 0.000323 ***
## Biomass:pHlow -0.15503    0.04003  -3.873 0.000108 ***
## Biomass:pHmid -0.03189    0.02308  -1.382 0.166954
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 452.346  on 89  degrees of freedom
## Residual deviance:  83.201  on 84  degrees of freedom
## AIC: 514.39
##
## Number of Fisher Scoring iterations: 4
```

```r
## Plot mean regression lines
newdata <- expand.grid(Biomass=seq(min(df$Biomass), max(df$Biomass), length.out=200),
                       pH=levels(df$pH))
newdata <- cbind(newdata, Species=predict(fit, newdata, type='response'))


ggplot(mapping=aes(x=Biomass, y=Species, colour=pH)) + geom_point(data=df) +
    geom_line(data=newdata)
```



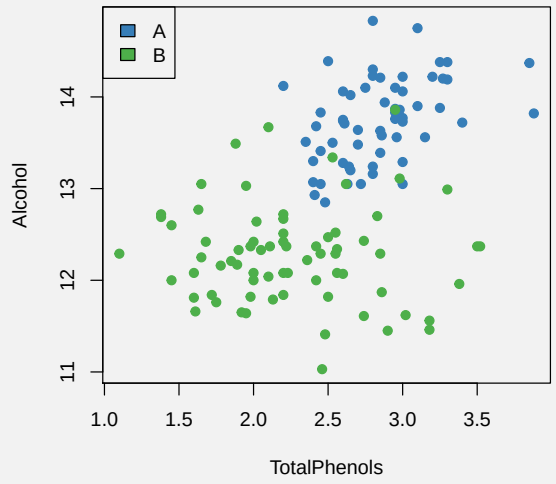
As `Biomass` increases the expected number of species tends towards zero for all three `pH` levels, which is what we would expect.

Return to task on P56

Solution 12

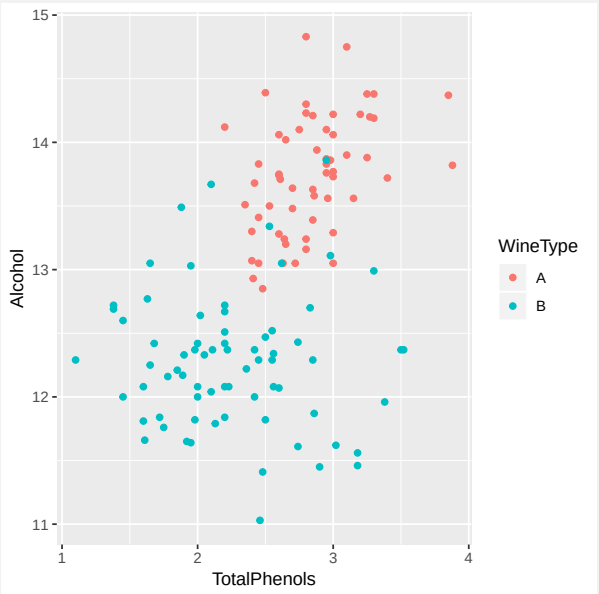
Base R

```
colA <- '#377eb8' # colour for Wine Type A
colB <- '#4daf4a' # colour for Wine Type B
plot(Alcohol ~ TotalPhenols, data=wine[wine$Win
     pch=19, col=colA, xlim=c(min(wine$TotalPhe
     ylim=c(min(wine$Alcohol), max(wine$Alcohol
points(Alcohol ~ TotalPhenols, data=wine[wine$W
       pch=19, col=colB)
legend('topleft', c('A', 'B'), fill=c(colA, col
```



tidyverse

```
ggplot(wine, aes(x=TotalPhenols, y=Alcohol, colour=Win
```



Return to task on P64

Solution 13

```
fit <- glm(WineType ~ Alcohol + TotalPhenols, data=wine, family=binomial(link=logit))
summary(fit)
```

```
##
## Call:
## glm(formula = WineType ~ Alcohol + TotalPhenols, family = binomial(link = logit),
##     data = wine)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.69074  -0.17802   0.03363   0.17346   2.93912
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   64.2693    12.5036   5.140 2.75e-07 ***
## Alcohol       -4.5603     0.9226  -4.943 7.70e-07 ***
## TotalPhenols  -1.8202     0.9239  -1.970   0.0488 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 179.109  on 129  degrees of freedom
## Residual deviance:  48.005  on 127  degrees of freedom
## AIC: 54.005
##
## Number of Fisher Scoring iterations: 7
```

Return to task on P64

Solution 14

```
## extract model parameters
beta0 <- round(coef(fit)[1], 4) # (Intercept)
beta1 <- round(coef(fit)[2], 4) # Alcohol
beta2 <- round(coef(fit)[3], 4) # Total Phenols

## compute log odds and then the probability
logOdds <- beta0 + beta1*12.5 + beta2*2.5
p <- round(invlogit(logOdds), 2)
print(paste0('For Alcohol=12.5% and Total Phenols=2.5:   p(WineType=B) = ', p))
```

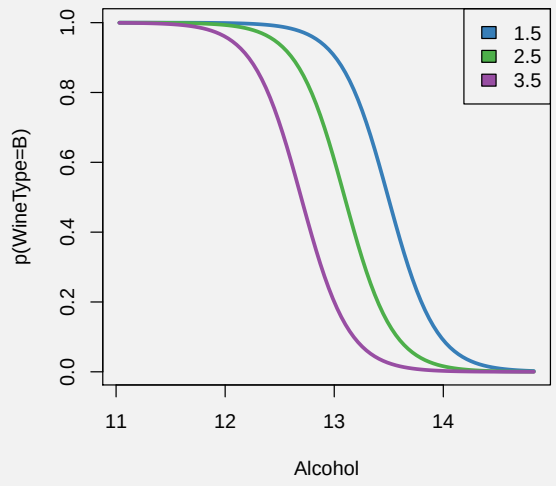
```
## [1] "For Alcohol=12.5% and Total Phenols=2.5:   p(WineType=B) = 0.94"
```

Return to task on P65

Solution 15

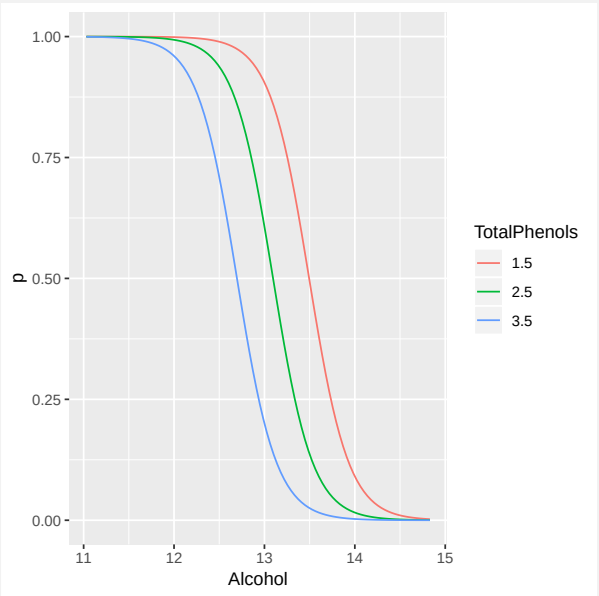
Base R

```
cols <- c("#377eb8", "#4daf4a", "#984ea3") # colours for the three cases
phenol <- c(1.5, 2.5, 3.5) # phenol levels to plot

## compute mean regression line on the probabil
newdata <- expand.grid(Alcohol=seq(min(wine$Alc
                       TotalPhenols=phenol)
newdata <- cbind(newdata, p=predict(fit, newdat

## plot mean regression line
i <- 1
plot(p ~ Alcohol, data=newdata[newdata$TotalPhe
     col=cols[i], type='l', ylab='p(WineType=B)
for (i in 2:3)
{
    lines(p ~ Alcohol, data=newdata[newdata$Tot
          col=cols[i], lwd=3)
}
legend('topright', c('1.5', '2.5', '3.5'), fill
```



tidyverse

```
phenol <- c(1.5, 2.5, 3.5) # phenol levels to plot

## compute mean regression line on the probability sca
newdata <- expand.grid(Alcohol=seq(min(wine$Alcohol),
                       TotalPhenols=phenol)
newdata <- cbind(newdata, p=predict(fit, newdata, type
newdata$TotalPhenols <- as.factor(newdata$TotalPhenols

## plot mean regression line
ggplot(mapping=aes(x=Alcohol, y=p, colour=TotalPhenols
```



Return to task on P65

Answer 16

Because the analysis does not account for the cabinet effects. There are not 20 *independent* measurements here.

Return to task on P72

Solution 17

```
summary(lm(growth ~ media * cabinet, data = bac))

##
```

```
## Call:
## lm(formula = growth ~ media * cabinet, data = bac)
##
## Residuals:
## ALL 20 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.1         NA      NA       NA
## media2                1.3         NA      NA       NA
## media3               -0.6         NA      NA       NA
## media4               -1.6         NA      NA       NA
## cabinet2              3.2         NA      NA       NA
## cabinet3              8.5         NA      NA       NA
## cabinet4              1.3         NA      NA       NA
## cabinet5              4.4         NA      NA       NA
## media2:cabinet2      -1.2         NA      NA       NA
## media3:cabinet2      -1.7         NA      NA       NA
## media4:cabinet2      -2.2         NA      NA       NA
## media2:cabinet3       2.1         NA      NA       NA
## media3:cabinet3      -2.0         NA      NA       NA
## media4:cabinet3      -4.0         NA      NA       NA
## media2:cabinet4       1.1         NA      NA       NA
## media3:cabinet4      -1.3         NA      NA       NA
## media4:cabinet4      -1.1         NA      NA       NA
## media2:cabinet5       0.4         NA      NA       NA
## media3:cabinet5      -1.8         NA      NA       NA
## media4:cabinet5      -3.9         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 19 and 0 DF,  p-value: NA
```

We are faced with all these `NA`s because the model is **saturated**: there is no replication so we cannot calculate the residual deviance/variance. This also means that there are no degrees-of-freedom remaining in the model to calculate the residual sum-of-squares. We have twenty observations and twenty parameters to estimate. This is analogous to fitting a straight line through *two* data points; we get a perfect fit, and thus overfits.

Return to task on P72

Solution 18

```
confint(bac_lmer, level = 0.97)
```

```
## Computing profile confidence intervals ...
```

```
##                   1.5 %     98.5 %
## .sig01        1.4056367  6.3307155
## .sigma        0.7099854  1.5901069
## (Intercept)   2.2329510  8.9270527
## media2        0.2810402  3.2789599
## media3       -3.4589598 -0.4610401
## media4       -5.3389598 -2.3410401
```

The first two terms are the CIs for σ_γ and σ respectively. The third term is the CI for bacterial growth within an **average** cabinet. The next three terms correspond to the difference in bacterial growth for `media` levels 2, 3 and 4, relative to `media` level 1. Each of these effects are statistically significantly different to the baseline `media` at the 3% level. The model suggests that `media3` and `media4` have attenuated growth compared to `media1`, whereas `media2` has exacerbated growth.

This approach uses the profile likelihood to calculate CIs. There is an option to `confint()` that allows you to calculate parameteric bootstrapped CIs if you prefer. The latter are likely to be more accurate, and thus are beneficial if your CI/p-value is hovering around the required level of statistical significance and you wish to improve this estimate. However, bootstrapping is a simulation-based approach and can be computationally expensive.

Return to task on P76

Solution 19

Note that this is a **crossed** design, so we cannot use F-tests. Instead we must use LRTs and we need to refit the model using ML when assessing the impact of dropping `media` from the model.

```
## load data
library(faraway)
```

```
##
## Attaching package: 'faraway'
```

```
## The following objects are masked from 'package:arm':
##
##     fround, logit, pfround
```

```
data(abrasion)

## fit model
abrasion_lmer <- lmer(wear ~ material + (1 | position) + (1 | run), data = abrasion)
drop1(update(abrasion_lmer, REML = F), test = "Chisq")
```

```
## Single term deletions
##
## Model:
## wear ~ material + (1 | position) + (1 | run)
##          Df    AIC    LRT   Pr(Chi)
## <none>      134.32
## material  3 151.69 23.364 3.391e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(abrasion_lmer)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: wear ~ material + (1 | position) + (1 | run)
##    Data: abrasion
##
## REML criterion at convergence: 100.3
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.08973 -0.30231  0.02697  0.42254  1.21052
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  position (Intercept) 107.06   10.347
##  run      (Intercept)  66.90    8.179
##  Residual              61.25    7.826
## Number of obs: 16, groups:  position, 4; run, 4
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  265.750      7.668  34.656
```

```
## materialB    -45.750      5.534  -8.267
## materialC    -24.000      5.534  -4.337
## materialD    -35.250      5.534  -6.370
##
## Correlation of Fixed Effects:
##           (Intr) matrlB matrlC
## materialB -0.361
## materialC -0.361  0.500
## materialD -0.361  0.500  0.500
```

This suggests that there are highly statistically significant differences in wear between *at least some* of the materials, with material `B` wearing the fastest, followed by `D`, `C` and `A`.

Note that we could derive confidence intervals for these effects, or for predictions / plotting etc. if we liked. There are many other options available here.

Return to task on P76

Answer 20

It is wrong because this would add **two** random effect terms, one for rat `1` and one for rat `2`. In fact there are 6 rats altogether. The way that the data have been coded allows for these kinds of mistakes to happen. The same is true for `Liver`, which is coded as a `1`, `2` or `3`. This means that we could write:

```
rats_lmer <- lmer(Glycogen ~ Treatment + (1 | Rat) + (1 | Liver), data = rats)
```

thinking that we are including the correct random effects for `Rat` and `Liver`. In fact, this assumes that the data come from a **crossed** design, in which there are 2 rats and 3 parts of the liver, and that `Liver = 1` corresponds to the same type of measurement in rats `1` and `2` and so on. Sometimes this is appropriate, but not here!

The nature of the way that many data sets are coded makes these kinds of mistakes very easy to make!

Return to task on P79

Solution 21

tidyverse

1.

```
drunk <- readRDS("drunk.rds")
alc <- drunk %>%
    group_by(student, freshener) %>%
    summarise(alcohol = mean(alcohol)) %>%
    ungroup()
alc_lm <- lm(alcohol ~ freshener,
    data = alc)
drop1(alc_lm, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## alcohol ~ freshener
##           Df  Sum of Sq        RSS     AIC Pr(>Chi)
## <none>                  0.00042582 -53.320
## freshener  1 0.00028843 0.00071424 -52.216  0.07813 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.

```
confint(alc_lm, level = 0.97)[2, ]
```

```
##       1.5 %      98.5 %
## -0.01391368  0.04164701
```

3.

```
drunk_lmer <- lmer(alcohol ~ freshener +
    (1 | student / sample), data = drunk)
```

4.

```
drop1(update(drunk_lmer, REML = F),
    test = "Chisq")
```

```
## Single term deletions
##
## Model:
## alcohol ~ freshener + (1 | student/sample)
##           Df     AIC    LRT Pr(Chi)
## <none>       -812.16
## freshener  1 -811.06 3.1033 0.07813 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
drop1(drunk_lmer, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## alcohol ~ freshener + (1 | student/sample)
##           Df     AIC    LRT Pr(Chi)
## <none>       -812.16
## freshener  1 -811.06 3.1033 0.07813 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Doesn't matter whether you use REML or ML here, since data are balanced and nested.

5.

```
confint(drunk_lmer, level = 0.97)[5, ]
```

```
## Computing profile confidence intervals ...
##        1.5 %       98.5 %
## -0.004529647  0.032262984
```

6. The profile likelihood based CI from the mixed model is slightly narrower than the exact CI generated from the derived variable analysis, but both are marginal in terms of their statistical significance (and give the same LRT statistic here).

Base R

1.

```
drunk <- readRDS("drunk.rds")
alc <- aggregate(alcohol ~ student +
    freshener, data = drunk, mean)
alc_lm <- lm(alcohol ~ freshener,
    data = alc)
drop1(alc_lm, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## alcohol ~ freshener
##           Df  Sum of Sq        RSS     AIC Pr(>Chi)
## <none>                  0.00042582 -53.320
## freshener  1 0.00028843 0.00071424 -52.216  0.07813 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.

```
confint(alc_lm, level = 0.97)[2, ]
```

```
##       1.5 %      98.5 %
## -0.01391368  0.04164701
```

3.

```
drunk_lmer <- lmer(alcohol ~ freshener +
    (1 | student / sample), data = drunk)
```

4.

```
drop1(update(drunk_lmer, REML = F),
    test = "Chisq")
```

```
## Single term deletions
##
## Model:
## alcohol ~ freshener + (1 | student/sample)
##           Df     AIC    LRT Pr(Chi)
## <none>       -812.16
## freshener  1 -811.06 3.1033 0.07813 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
drop1(drunk_lmer, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## alcohol ~ freshener + (1 | student/sample)
##           Df     AIC    LRT Pr(Chi)
## <none>       -812.16
## freshener  1 -811.06 3.1033 0.07813 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Doesn't matter whether you use REML or ML here, since data are balanced and nested.

5.

```
confint(drunk_lmer, level = 0.97)[5, ]
```

```
## Computing profile confidence intervals ...
##        1.5 %       98.5 %
## -0.004529647  0.032262984
```

6. The profile likelihood based CI from the mixed model is slightly narrower than the exact CI generated from the derived variable analysis, but both are marginal in terms of their statistical significance (and give the same LRT statistic here).

Return to task on P80

Answer 22

Because the LRT at the previous stage was comparing the baseline model `ldist ~ Species + Number` with the model `ldist ~ Number`. Since we have now removed `Number`, the new LRT will compare the baseline model `ldist ~ Species` with the null model `ldist ~ 1` (i.e. a model with just a single intercept value). In **unbalanced** designs these two comparisons are different tests and thus can give different results.

Return to task on P93