Lecture 1  An example to indicate the role of statistics

You do not have to know how the analyses are done (that comes later) but you should follow the discussion to appreciate how the data are decomposed stepwise into meaningful information.

Experiment data

Any scientific principle must be supported by evidence.¹

The best evidence, especially in applied science and technology, comes from repeatable, controlled experiments.

To be practical, these principles must allow verifiable, reliable predictions.

We shall use the term experiment to stand for a wide range of activities.

- chemical experiment, a field trial or a clinical trial
- astronomical observations. The observer engages in the process by selecting the object(s) and determining what measurements are to be recorded through the telescope.

The observations which can be recorded as numerical responses are termed data.²

Data for statistical analysis are also recorded in surveys and observational studies.

Because we cannot control everything in an experiment, the data will be subject to random influences. For instance,

(i) an electrical measuring instrument may have its “ideal” readings modified by the random (but small) electrical feedback that is associated with the circuit,

(ii) the reagents may not be exactly “pure”,

(iii) the environmental conditions such as temperature or refractive index may change during the experiment, or

(iv) there might be so many factors that cannot be controlled and these induce a measurement error.

There is a component of “measurement error” in all experiment data.

Example of systematic and random effects

Example 1.1

The data in Table 1.1 are the observed concentrations of CO₂ taken monthly from 1959 to 1997 at Mauna Loa (Hawaii).

Table 1.1: Monthly CO₂ concentrations at Mauna Loa

<table>
<thead>
<tr>
<th>Year</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1959</td>
<td>315.4</td>
<td>316.3</td>
<td>316.5</td>
<td>317.6</td>
<td>318.1</td>
<td>318.0</td>
<td>316.4</td>
<td>314.6</td>
<td>313.7</td>
<td>313.2</td>
<td>314.7</td>
<td>315.4</td>
</tr>
<tr>
<td>1960</td>
<td>316.3</td>
<td>316.8</td>
<td>317.4</td>
<td>318.9</td>
<td>319.9</td>
<td>319.4</td>
<td>318.0</td>
<td>315.7</td>
<td>314.0</td>
<td>313.7</td>
<td>314.8</td>
<td>316.0</td>
</tr>
<tr>
<td>1961</td>
<td>316.7</td>
<td>317.5</td>
<td>318.4</td>
<td>319.3</td>
<td>320.4</td>
<td>319.6</td>
<td>318.4</td>
<td>316.6</td>
<td>314.8</td>
<td>315.2</td>
<td>315.9</td>
<td>316.9</td>
</tr>
<tr>
<td>1962</td>
<td>317.8</td>
<td>318.4</td>
<td>319.5</td>
<td>320.4</td>
<td>320.9</td>
<td>320.4</td>
<td>319.4</td>
<td>317.2</td>
<td>316.1</td>
<td>315.3</td>
<td>316.5</td>
<td>317.5</td>
</tr>
<tr>
<td>1995</td>
<td>360.0</td>
<td>361.0</td>
<td>361.7</td>
<td>363.5</td>
<td>363.8</td>
<td>369.0</td>
<td>363.9</td>
<td>369.5</td>
<td>358.1</td>
<td>357.8</td>
<td>359.6</td>
<td>360.7</td>
</tr>
<tr>
<td>1996</td>
<td>362.1</td>
<td>363.3</td>
<td>364.1</td>
<td>364.8</td>
<td>365.4</td>
<td>365.0</td>
<td>363.7</td>
<td>361.5</td>
<td>359.5</td>
<td>359.6</td>
<td>360.8</td>
<td>362.4</td>
</tr>
<tr>
<td>1997</td>
<td>363.2</td>
<td>364.1</td>
<td>364.6</td>
<td>366.4</td>
<td>366.8</td>
<td>365.7</td>
<td>364.5</td>
<td>362.6</td>
<td>360.2</td>
<td>360.8</td>
<td>362.5</td>
<td>364.3</td>
</tr>
</tbody>
</table>

¹Galileo, 1564-1642
²A single response is termed a datum and the appropriate grammar is “... data are ...”.
A model is a mathematical description of the process. To set up a model, the problem is simplified and only those aspects which can be represented mathematically are included. After the problem is solved mathematically, tentative solutions are translated back to the real situation as possible real solutions. The inadequacy of the simple model may be revealed and some parts of the process need to be changed.

A general description of a statistical model is:

\[
\text{observed data = systematic effects + random effects}
\]

where systematic effects are smooth curves or averages and random effects are represented by the density curve or pattern of randomness.

<table>
<thead>
<tr>
<th>Term</th>
<th>Mathematical representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>systematic effects</td>
<td>curves, averages, groups</td>
</tr>
<tr>
<td>random effects</td>
<td>density functions</td>
</tr>
</tbody>
</table>

Alternate terms to systematic+random are (i) signal+noise, (ii) explained+unexplained. Our understanding and predictability of the CO\(_2\) data was done using a statistical model.

**Prediction of CO\(_2\) conc. in 2010**

To predict the concentration of CO\(_2\) in 2010, we supply the systematic components into the model. The random or unexplained component leads to an 95\% confidence interval of estimation.

<table>
<thead>
<tr>
<th>Season</th>
<th>Component Estimate</th>
<th>Prediction Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>trend</td>
<td>season</td>
</tr>
<tr>
<td>Summer</td>
<td>382.7</td>
<td>-2.5</td>
</tr>
<tr>
<td>Winter</td>
<td>382.7</td>
<td>+2.5</td>
</tr>
</tbody>
</table>

**Sample and Population**

In most practical situations there is a limit to how much data can be collected and the aim is to collect sufficient data so that we get enough reliable information. The set of data chosen to represent the process is termed a sample. The entire set of units from which the sample is taken is called a population and if the entire population is measured, that set of data is called a census.

A well chosen sample has most of the information about the population but not all. If a different sample were taken using different individuals from the population, the data would be similar but not identical.

Thus the sample contains an error where the understanding of error is in the mathematical sense of a slight discrepancy between the population and its sample.
Some measurements also contain random components due to the inherent random nature of the variables being measured, eg. in weather forecasting.

![Graph showing the relationship between sample size and error]

**Summary**

Statistics is the (mathematical) science of randomness and uncertainty. It serves other sciences by:

(i) providing the means to summarise information (i.e. data) and to make reliable predictions,
(ii) improving understanding of scientific principles by identifying the significance of effects from different components, and
(iii) quantifying the inherent variability of a population.
(iv) Accuracy of estimation is an interval and statistical analysis seeks to estimate that interval efficiently.
Lecture 2  R programming

Abstract This lecture covers the basic principles of programming in R, inputing data, processing, plotting and writing output.

It is assumed that you have set up R and installed the Rcmdr package. The way to do this is described in the notes on setting up R.

R is described by Dalgaard [??],

\textit{R is a statistical program, made available through the Internet under a General Public License (GPL).}

\textit{R provides an environment in which you can perform statistical analysis and produce graphics.}

The citation for R is [??].

R software consists of a set of core functions which are organised into packages, \texttt{base, stats, methods, graphics, utils}. These are loaded and available when you start the R program. There is a vast set of functions for more specialized jobs and these are in libraries. Since specialised jobs are not done routinely, they are not automatically loaded so that the computer space is not cluttered. When required, a library is loaded. These functions are used to write programs which

1. read data,
2. fit models,
3. calculate predictions,
4. tabulate and plot results

\textit{R is a statistical programming package which reads, writes, calculates and plots by processing scripts.}

For early versions of R, the user was required to type the scripts. Many new users found this difficult because the job has to be done carefully and minor errors such as not including brackets etc. caused the program to stop.

The Rcmdr package was introduced in 2006 as an alternative way of R programming.

It is a GUI (Graphical User Interface) which provides menus to enter the information that R needs to do calculations and plots etc. The Rcmdr package then

- generates the R script and
- runs the generated code

This can alleviate many of the frustrations of beginners who have yet to practice scientific computing but it does not altogether replace scripts. In most cases, we shall use Rcmdr to do the main work but in some cases we shall augment this with our own short scripts.

One does not learn R by first learning all the components. Like most computer programs it is learnt by example and practice. In this lecture, the basic concepts are stated so that when you encounter a program in the notes, you have sufficient understanding to parse it to see the logical flow and how results are obtained. Example programs are intended as templates which can be adapted for new jobs.

A computer program has 3 broad components,

1. read the data,
2. process,
3. return the results.
We shall go through these steps using Rcmdr.

All computing, whether statistical or other, should be organised into directories. The name of the directory is arbitrary so for these examples the directory could be named Lect2. If this directory was a sub-directory of say stat100, then the path to this is C:\My Documents\stat100\Lect2 if using the Windows operating system.

For a different component of the unit, say assignment 1, you should make another directory Asgn1 (or whatever) as another sub-directory of stat100.

This is important for efficient running of R so that files are easily located and not muddled.

**NOTE.** You must have all files associated with the same job in the same directory.

### 2.1 Loading the Rcmdr package

1. Start R by double clicking on the blue icon on the desktop with the Left Mouse Button (LMB).
   
   This opens up a window termed the R Console. At the top there is a toolbar with menus, File, Edit, Misc, Packages, Windows, Help

2. Change to the working directory by selecting Change dir .. from the options under File.
   
   Browse to find this directory, e.g.
   C:\My Documents\STAT100\L2

   Do not work in the default directory which is where the R files are saved in C:\Program Files.

3. Use the Packages menu in the toolbar of the R Console to choose Load package and select Rcmdr.

   This will open the Rcmdr window. Its features are

   1. a toolbar
   2. a script window
   3. an output window
   4. a messages window

   The toolbar has menus which generate GUI’s whereby we can enter data, request calculations, construct graphs, etc.

   The script window is where the generated script is listed.

   Also we can enter commands in this window and then click on **Submit** whereby the command is despatched to the R program running in the adjacent R Console.

   Outputs generated on the fly are printed out in the **Output window** although mostly we would want our results written to a file.

   The Messages window alerts us to any mistakes such as the data are not suitable for what we are trying to do. when all is well, not much happens in the Messages window.

**Example 2.1**

The purpose of this example is to introduce you to programming in R. We are not too much concerned about the statistical issues yet but many of you will recognise the normal curve. We generate random numbers whose frequencies follow the normal curve.

The components that are demonstrated are

- we input some values
- the R program does some calculations
• we output a graph

From the Rcmdr toolbar, use LMB to choose the options

Distributions → Continuous distributions → Normal distribution → Sample from normal distribution

as depicted in Figure 2.1

Figure 2.1: A screen shot of simulating random numbers from a normal distribution

Click OK and the GUI appears on the screen

In this example, the mean is entered as 2, the standard deviation is 3 and the generated random variables are saved in 1 column of 100 rows. Note that the default arrangement is 1 row and 100 columns but the other way is more useful. The numbers will be saved in an object which is named Normal.rv.

With a LMB click on OK, the entries are fed to a program which generates the script and runs it. You can see the script in the script window and as well it appears in the R console.
That script is rather verbose but we do not have to worry about it. We just remember what we set out to do which was generate 100 numbers that are sampled from a normal distribution with mean 2 and standard deviation 3.

An alternate way would be to type the following into the script window and LMB on Submit.

```
Normal.rv <- rnorm(mean=2,sd=3,n=100)
```

Some people prefer this way as it is more direct but it does require careful typing and remembering a few things. either way is good and in practice we use a combination of GUIs and writing our own scripts.

Although the numbers have been generated, we do not see them unless we request the program to display them. They are saved in the object called `Normal.rv` and this name appears in Data set. To inspect, LMB click on View data set.

Just looking at these numbers is not informative so we shall plot the histogram and the density. Henceforth, Rcmdr menus will be described in the fashion of 2.1 and we shall only depict with screen shots occasionally.

Plot the histogram of these numbers with Graphs → Histogram.

Check the button for Density and OK.

The histogram is plotted in the Graphics window which is part of the R console and you view it by a LMB click on that window.
Now we shall superpose a density curve over the histogram.

It requires these steps,

- Use the mouse to highlight `Hist(Normal.rv$obs` and copy this to the clipboard with `Ctrl` C.

- Paste it into the script window, below, with `Ctrl` V.

- Edit this (with typing & erase with ← Backspace) to `lines(density(Normal.rv$obs))`
  You must type with care. Observe the brackets are matched.

It looks like this:-

![R Commander Interface](image)

- Highlight that new line and Submit

The graph now looks like this:-
In this example, we used the menu of Rcmdr to generate 100 normal (mean=2, sd=3) random numbers and plot the histogram. We added our own script to superpose the density curve.

2.2 Reading data

There are 3 principle ways of entering data.

- The data are filed in tabular form with headings for each column to indicate variable names.
- The data are in a file but not in tabular form. Use `scan()` to read these data.
• The data set is small and can be supplied in the R program using `c()`.

We shall illustrate the first case with an example to calculate the body mass index (bmi) from weights and heights.

If you have installed the RODBC package, Rcmdr can read data from an Excel file otherwise to would record the data in a .txt file using NOTEPAD, say.

Suppose we enter the height and weight data using Excel,

Data files are organised by:
• the columns contain the variables, height & weight in this case
• each row contains the measurements from each unit
• the first row contains headings which will be used to describe the data

Import the data using the Rcmdr menus
Data → Import data → From Excel, .

The GUI will require a name for the data which by default is Dataset. It is a good idea to supply informative names to the data, e.g. BMIdata

Use the file browser to select the file, bmi.xls. You will see in the script window how the program reads the data from the Excel file.

Alternative way of reading data

Since the amount of data is small, we could have supplied the data directly to the variables in a small script. Type these commands into a NOTEPAD document, and save as a .txt file. DO NOT USE WORD for this - it will not work.

Save the file as a .txt file with the name bmi.R

```r
weight <- c(60,72,57,90,95,72)
height <- c(1.75,1.80,1.65,1.90,1.74,1.91)
bmi <- weight/height^2
BMIdata <- data.frame(weight, height, bmi)
print(mean(BMIdata$bmi))
```
Now you can get this script into the Rcmdr script window by,

File → Open script file

Now highlight all the lines with the mouse and LMB on Submit.

It achieves the same end as the previous section on importing data from file.

Sometimes it is more convenient to do things this way rather than making a data file but once again, careful typing is required.

## Calculate new variables

We can now calculate the bmi from the weights and heights. The formula is

\[
\text{bmi} = \text{weight}/(\text{height}^2)
\]

Use Data → Manage variables in active data set → Compute new variable

- Type bmi in the slot for new variable name.
- LMB click in the slot for Expression to compute.
- Now highlight weight in the list of current variables and double click. It appears as a term in the expression to be computed.
- Type “/” after weight as the division operator in the formula.
- highlight height and double click.
- Add “\(^2\)” to square the height in the formula.
- OK

## Calculate summary statistics

Use Statistics → Summaries → Numerical summaries

Highlight bmi in the list of variables and OK.

The mean and quartiles are listed on the output window.

### 2.3 Factors

A vector of categories may be coded as alphabetic labels, e.g.

- A, B, C, ...
- Low, Medium, High
- GlenI, Armidale, Tamworth

It is often convenient to code the categories as 1, 2, 3, 4, etc. where it is understood that the number refers to a category and is not a numeric value.

There is no confusion if the category is defined by alphabetic characters but if numbers are used, R will interpret them as numbers and not categories by default. To have the numbers regarded as categories, use the function `factor()`.
Example 2.2

This principle is illustrated using the data set called sludge.txt which has results of a bioassay of plants with 3 Rates of sludge and zinc extracted from the plants.

<table>
<thead>
<tr>
<th>Block</th>
<th>City</th>
<th>Rate</th>
<th>zinc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>0.5</td>
<td>29.5</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>1</td>
<td>28.3</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>1.5</td>
<td>29.1</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>0.5</td>
<td>33.2</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>1</td>
<td>50.8</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>1.5</td>
<td>76.9</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>0.5</td>
<td>22.5</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>1</td>
<td>26.3</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>1.5</td>
<td>22.0</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>0.5</td>
<td>26.8</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>1</td>
<td>42.5</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>1.5</td>
<td>47.9</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>0.5</td>
<td>34.3</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>1</td>
<td>42.4</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>1.5</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>0.5</td>
<td>22.6</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>1</td>
<td>24.6</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>1.5</td>
<td>23.1</td>
</tr>
</tbody>
</table>

Missing values are recorded as NA.

- The column zinc is a numeric vector and we can do numerical operations on it, e.g.

```
> mean(sludge$zinc)
[1] 34.98611
```
- The variate City is a categorical variable or a factor and we use those variables to indicate different groups. For instance

```
> sludge$City
Levels: A B C
```
indicates that there are 3 levels. Because each cell is coded as a character string, R automatically assumes that it is a factor and hence it does not have to be declared.

- The column Block is coded as 1 or 2 and unless we specify otherwise, R assumes that it is a numeric variable (the default). However, the numerals 1,2 in this case are convenient labels for a category so we must define this as a factor.

Suppose the Block codes represent samples taken in January and March. We can still use 1 and 2 to define the categories since that makes data recording simpler but attach to these the labels Jan and Mar.

- Depending on how we wish to interpret the response zinc, Rate may be either a categorical variable, i.e. a factor with levels 0.5, 1.0 and 1.5, or it may be treated as a numeric variable. If we wanted to tabulate means, it would work best as a factor but if we required a plot of the response to Rate, it would be a numeric variable.

Entering the data and recoding using Rcmdr

- The data are read using Rcmdr, Data → Import data → From text file

(Or you could be reading from an Excel file.)

- Name the data set sludge, OK
• View data set
  the close that box.

• Data → Manage variables in active data set → Convert numeric variables to factors

• Double click on Block. The levels of the factor can be Jan, Mar for example.
  In the script window, you can see how the R program does this,

\[
\text{sludge$Block} \leftarrow \text{factor(sludge$Block, labels=c('Jan', 'mar'))}
\]

Plot the data

• Plot zinc against City
  – Graphs → Boxplot
  – Select zinc as the variable
  – [Plot by groups] and select City. [OK]
  – Examine the graph in the graphic window of the R console.

• Plot zinc against rate
  – Graphs → Scatterplot
  – Rate is the x-variable and zinc is the y-variable.
  – [Plot by groups] → [City] [OK]
  – Complete the plot with [OK] and go to the graphics window.

### 2.4 Exercises

**Exercise 2.1**

Repeat Example 2.1 to generate normal random numbers and plot the histogram and density curve.

**Exercise 2.2**

The data in Table 2.1 are the numbers of White Clover stolons counted in each of 2 quadrats which are 100cm\(^2\) in area. Use R to estimate the numbers of stolons m\(^{-2}\) for each variety.

1. Open a file to record the R script.
2. Enter the data, \(q1 \leftarrow c(42, \ldots )\)
3. Add the counts from the 2 quadrats and scale up to units of m\(^2\), \((q1+q2)*50\)
4. Save file and source the program.

<table>
<thead>
<tr>
<th>Variety</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>quadrat 1</td>
<td>42</td>
<td>77</td>
<td>63</td>
<td>40</td>
<td>37</td>
<td>73</td>
</tr>
<tr>
<td>quadrat 2</td>
<td>54</td>
<td>91</td>
<td>85</td>
<td>62</td>
<td>55</td>
<td>67</td>
</tr>
</tbody>
</table>
Exercise 2.3

The data are widths of etch lines measurements on silicon wafers that have been produced under different viscosity/temperature conditions for different baking times, referred to as bakeT. These data (wafer.txt) may be downloaded from the Datasets web page.

using Rcmdr

• Read the data

• Convert VT to a factor. The levels can be labelled as 1, 2 or 3.

• Make a scatterplot of etch against bakeT with grouping factor VT

<table>
<thead>
<tr>
<th>wafer</th>
<th>VT</th>
<th>bakeT</th>
<th>etch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>20</td>
<td>2.84</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>30</td>
<td>3.14</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>40</td>
<td>3.65</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>20</td>
<td>2.20</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>30</td>
<td>3.07</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>40</td>
<td>3.60</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>20</td>
<td>4.50</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>30</td>
<td>4.62</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>40</td>
<td>4.85</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>20</td>
<td>3.31</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>30</td>
<td>3.60</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>40</td>
<td>4.17</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>20</td>
<td>3.00</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>30</td>
<td>3.60</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>40</td>
<td>3.70</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>20</td>
<td>4.28</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>30</td>
<td>4.40</td>
</tr>
<tr>
<td>18</td>
<td>3</td>
<td>40</td>
<td>4.84</td>
</tr>
</tbody>
</table>
Lecture 3   Reading and Plotting Data

3.1 Plotting

Try this elementary plot:-

```r
x <- 1:20
plot(x, log(x), type='l', xlab="x", ylab="log(x)", las=1)
```

The function `plot()` takes the first argument, `x`, as the independent variable and the second, `log(x)` as the dependent variable. The argument `type="l"` specifies that the relationship is drawn as a line rather than points and `xlab` and `ylab` put the labels on the graph. Setting `las=1` puts the axis ticks at right angles to the axes.

We are free to vary this to make a plot to our suiting. For instance,

```r
plot(x, log(x), type='p', xlab="variable", ylab="log transform (base e)", las=0)
```

The full details on `plot` can be read by typing `help(plot)` in the R console.

Example 3.1

The data in Table 3.1 are the uptakes of CO$_2$ in wheat leaves when subject to airflows with varying concentrations of CO$_2$. The experimenter controls the Concentration and it is termed the independent variable. The Uptake variable is termed dependent because it is a consequence of the input variable.

Table 3.1 shows 2 forms of the data. The left frame might be how the experimenter records the data in a notebook and is organised for visual inspection. However, that form would be confusing to the computer and so the data is rewritten to that in the right frame of Table 3.1 with the observations in columns and each row being the set of observations for a single case.

<table>
<thead>
<tr>
<th>Conc (x)</th>
<th>Uptakes (cm$^2$dm$^{-2}$hour$^{-1}$)</th>
<th>Concentration</th>
<th>Uptake</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td>0.00</td>
<td>75</td>
<td>0.00</td>
</tr>
<tr>
<td>100</td>
<td>0.65 0.50 0.40</td>
<td>100</td>
<td>0.65</td>
</tr>
<tr>
<td>120</td>
<td>1.00</td>
<td>100</td>
<td>0.50</td>
</tr>
<tr>
<td>130</td>
<td>0.95 1.30</td>
<td>130</td>
<td>0.40</td>
</tr>
<tr>
<td>160</td>
<td>1.80 1.80 2.10</td>
<td>120</td>
<td>1.00</td>
</tr>
<tr>
<td>190</td>
<td>2.80</td>
<td>130</td>
<td>0.95</td>
</tr>
<tr>
<td>200</td>
<td>2.50 2.90 2.45 3.05</td>
<td>130</td>
<td>1.30</td>
</tr>
<tr>
<td>240</td>
<td>4.30</td>
<td>160</td>
<td>1.80</td>
</tr>
<tr>
<td>250</td>
<td>4.50</td>
<td>160</td>
<td>2.10</td>
</tr>
</tbody>
</table>

Table 3.1: Data for uptake of CO$_2$ in wheat leaves.
The independent variable is put on the X-axis of the plot and the dependent variable is on the Y-axis.

Although it is clear that uptake of CO$_2$ increases with concentration, the experimenter is interested in more than that. Possible questions are

- What is the shape of the response - straight line or curved?
- What is the rate of uptake of CO$_2$?

Both variables are continuous and the $x$ variable is controlled by the experimenter.

Some concentrations (eg 100, 130, 160, 200) are replicated and the measurements from these replicates vary, implying that there is an element of randomness. A plot of the data is usually the first step in data analysis.

The data (CO2uptake.txt) can be downloaded from the Datasets link on the stat100 website and saved in an appropriately named sub-directory of stat100, Exercises3 say.

Reading the data

Data \rightarrow Import data \rightarrow from text file

Enter name for data set [CO2uptake]

[OK]

Double LMB click on CO2uptake.txt.

Plotting the data

- Ensure that CO2uptake is the active data set. If you have just read the data, it will be the active data set but if you have done something else in between, make it active by:
  Data \rightarrow Active data set \rightarrow Select active data set

- Graphs \rightarrow Scatterplot

  x-variable = Conc
  y-variable = Uptake

Figure 3.1: Plot of the uptake of CO$_2$ in wheat leaves
The plot makes it somewhat simpler to suggest how these data could be represented as a statistical model. The systematic part is a straight line and the random part will be of smaller magnitude than systematic differences.

Example 3.2

A different type of plot is one where the independent variable is of categories (discrete) rather than a set of numbers (continuous). The boxplot works well to explore the data. The data in table 3.2 are 4 concentrations of arsenic in the ground water at 4 locations and the file on the website is named arsenic.txt.

Table 3.2: Arsenic concentrations in groundwater

<table>
<thead>
<tr>
<th>location</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.14</td>
<td>2.06</td>
<td>1.92</td>
</tr>
<tr>
<td>2</td>
<td>2.05</td>
<td>1.97</td>
<td>1.85</td>
</tr>
<tr>
<td>3</td>
<td>2.21</td>
<td>2.04</td>
<td>1.92</td>
</tr>
<tr>
<td>4</td>
<td>2.08</td>
<td>2.01</td>
<td>1.89</td>
</tr>
<tr>
<td>5</td>
<td>2.12</td>
<td>2.03</td>
<td>1.97</td>
</tr>
</tbody>
</table>

The plot is very similar to the previous except that we use the boxplot() function.

- Data → Import data → From text file
  Enter name for data set arsenic
  Double LMB click on arsenic.txt
- Make location a factor.
  Data → Manage variables in active data set → Convert numeric variables to factor
  Choose location and check the button Use numbers for Factor levels.
- Graphs → Boxplot
  1. Choose arsenic as the variable by highlighting with the mouse.
  2. LMB on Plot by groups
  3. Highlight location as the group variable.
  4. OK

The boxes denote the middle 50% of the data and the line inside the box indicates the middle point or median. The whiskers denote 95% of the range about the median.

3.2 Saving graphs

3.2.1 Windows

Graphs can be copied and printed using the menus on the toolbar. You make the graphics window active by clicking on its border - the border is blue, not gray, when it is active. Then click on Files on the toolbar and make your choice for printing or saving.

You can save the graphics as a .bmp file but that is bulky in terms of bytes. The .wmf format (Windows meta file) is okay for drafts.
Exercise 3.1

These data are wheat yields and rainfall in a particular month of the growing season, taken over 25 years. The data are filed in SAvwheat.txt which may be downloaded from the web page or retyped into a file. Note that the column in the data set that is listed here is Rain1.

<table>
<thead>
<tr>
<th>Rain</th>
<th>Yield</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.61</td>
<td>0.31</td>
</tr>
<tr>
<td>4.96</td>
<td>5.89</td>
</tr>
<tr>
<td>6.88</td>
<td>8.68</td>
</tr>
<tr>
<td>3.26</td>
<td>0.00</td>
</tr>
<tr>
<td>3.37</td>
<td>1.30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rain</th>
<th>Yield</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.60</td>
<td>0.00</td>
</tr>
<tr>
<td>0.67</td>
<td>0.03</td>
</tr>
<tr>
<td>3.30</td>
<td>2.49</td>
</tr>
<tr>
<td>1.52</td>
<td>0.00</td>
</tr>
<tr>
<td>3.24</td>
<td>0.03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rain</th>
<th>Yield</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.94</td>
<td>5.47</td>
</tr>
<tr>
<td>8.49</td>
<td>16.03</td>
</tr>
<tr>
<td>2.52</td>
<td>0.98</td>
</tr>
<tr>
<td>3.25</td>
<td>2.98</td>
</tr>
<tr>
<td>1.78</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rain</th>
<th>Yield</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.26</td>
<td>16.73</td>
</tr>
<tr>
<td>3.60</td>
<td>6.57</td>
</tr>
<tr>
<td>5.93</td>
<td>11.89</td>
</tr>
<tr>
<td>3.42</td>
<td>2.72</td>
</tr>
</tbody>
</table>

Using the uptake of CO₂ data set in Example 3.1 as an example, plot Yield as a response to Rain.
Exercise 3.2

R has datasets accompanying the program to demonstrate certain features. We shall use one here.

The dataset is called `InsectSprays` and the data frame is loaded by

Data → Data in packages → read data from attached package

select datasets followed by selecting `InsectSprays` from the list on the right.

Produce a boxplot of `count` (the response) versus `spray` which is the grouping variable.