Tutorial 2: Descriptive Statistics and Exploratory Data Analysis

Rob Nicholls – nicholls@mrc-lmb.cam.ac.uk

MRC LMB Statistics Course 2014

A very basic understanding of the R software environment is assumed.
See Tutorial 1 for an introduction to R.

Contents

1 Univariate Statistics and Histograms

The first part of this tutorial will consider univariate statistical analysis using R. This involves simple quantification and visualisation of the distribution of a univariate dataset.

To help illustrate the discussed concepts, we will use an artificial dataset. R contains built-in functions that will generate a dataset of random values typical of a particular distribution (e.g. see `runif`, `rnorm` and `rpois`; for a full list type `?Distributions`). Begin by generating a vector of 1000 random values from the standard Normal distribution using the command:

```
x = rnorm(1000)
```

The distribution of a univariate dataset may be easily visualised using a histogram. To plot a histogram of `x` type:

```
hist(x)
```

Histograms are bar charts, with bars arranged in equal-spaced intervals across the data range. The bar height is equal to the number (frequency) of values that occur in the corresponding interval in the dataset.

Note that it is not necessary to assign the result of `rnorm(1000)` to a variable `x` before creating the histogram – repeatedly executing the command:

```
hist(rnorm(1000))
```

would result in histograms being created using new vectors of artificially-generated random Normal data. Try this now, and see what happens. Note that the scales/limits of the axes automatically adjust according to the contents of the dataset. The x- and y-axis limits may be fixed using the `xlim` and `ylim` arguments, respectively. Also, the text of the main and axes titles can be set using the `main`, `xlab` and `ylab` arguments.

The `freq` argument is useful for switching between displaying frequency (number of observations) and density (proportion) on the y-axis. The command:

```
hist(x,freq=FALSE)
```

would result in density being displayed on the y-axis. This scales the height of the bars so that the sum of all heights totals one. This is useful, as it results in a view that is independent of the dataset length – the scale of the y-axis would be the same whether the vector `x` contained 10 or 10,000 observations.

Another important argument is `breaks`, which determines the size of the interval on the x-axis (i.e. the bar width). Try the two commands:

```
hist(x,breaks=10)
hist(x,breaks=100)
```

Adjusting `breaks` can be useful when attempting to achieve an optimal representation. Note that, if frequency is plotted on the y-axis, changing the number of breaks

can dramatically change the y-axis scale.

Not only does the `hist` function display a histogram, it also returns useful information regarding the histogram construction. Assign the return value of the `hist` function to the variable `y` using the command:

```
y = hist(x)
```

and look at the contents of the `y` variable. You will see that `y` now contains potentially-useful data, such as the mid-points of the histogram bars, and the counts (frequencies) and densities corresponding to each of the bars. These data can be accessed using the '$' symbol. For example, this information could be used to plot points corresponding to the relative heights of the histogram bars in a new plot, e.g. using the command:

```
plot(y$density~y$mids)
```

A smooth curve approximating the histogram can be achieved using the `density` function. In this case, a smooth curve can be added to the plot using the command:

```
lines(density(x))
```

Task 1:

1. Load the 'boot' library by typing: `library(boot)`
2. The dataset 'paulsen' within the 'boot' library contains data relating to neurotransmission in Guinea Pig brains. The current (pico-amperes) flowing into individual brain cells is stored in the variable 'y' (i.e. the vector containing these data can be accessed by typing: `paulsen$y`). Display the distribution of these data using a histogram, using the argument `breaks=30`, displaying density on the y-axis (rather than frequency).
3. Use the `lines` and `density` functions to overplot a smoothed curve onto the histogram that represents the empirical probability density function, coloured red.

1.1 Measures of Location

It is often of interest to identify a statistic representing some notion of the *average value* in a given dataset or distribution. Such an 'average value', which may also be referred to as the *central* or *typical* value, is called a measure of *central tendency* or *location*.

There are various measures of central tendency. Two of the most common are:

- *mean* (R function: `mean`) – the sum of all values divided by the number of observations. Also called the *arithmetic mean*, and is closely related to the *expected value* of a distribution.
- *median* (R function: `median`) – the middle value, such that half of the data are above, and half are below the median.

Recall the histogram created earlier in the tutorial:

```
x = rnorm(1000)
hist(x,freq=FALSE)
```

Vertical lines corresponding to the mean and median can added to the histogram using the commands:

```
abline(v=mean(x),col="red")
abline(v=median(x),col="blue")
```

You will see that, in this case, the mean and median are very close to each other. The `lwd` argument can be used to change the line thickness.

Whilst the mean is mathematically and computationally convenient, and powerful for well-behaved symmetric data, it is highly sensitive to outliers and often inappropriate for describing the 'typical' value from an asymmetric distribution. In contrast, the median is a *robust* statistic, being insensitive to outliers and applicable to asymmetric data.

To demonstrate this, type the command:

```
y = c(x,1000)
```

which creates a variable `y` that contains the same values as `x`, except for the inclusion of an outlier (1000). Now type the commands:

```
mean(x); mean(y)
median(x); median(y)
```

You will see that the mean is greatly affected by the inclusion of the outlier, whilst the median is (almost) unaffected. This can be visualised using the sequence of commands:

```
hist(x)
abline(v=mean(x),col="red")
abline(v=median(x),col="blue")
abline(v=mean(y),col="orange")
abline(v=median(y),col="green")
```

To illustrate the difference between the mean and median when the data are asymmetric (i.e. the distribution is *skewed*), create a new variable:

```
z = x^2
```

Now plot the histogram of `z`, set the number of breaks to 50, and add vertical lines corresponding to the mean and median of `z`, similarly to as above. You will see that there is a large difference between the mean and median. Whilst the mean is the 'average value', it is clearly not a typical value from this distribution, since the vast majority of the data lie below the mean – this can be confirmed using the following command:

```
length(z[z<mean(z)])
```

Task 2:

1. The inbuilt dataset 'rivers' contains data relating to the lengths of 141 rivers in North America. Display the distribution of these data using a histogram, using the argument `breaks=30`, displaying density on the y-axis (rather than frequency).

2. Use the `lines` and `density` functions to overplot a smoothed curve onto the histogram that represents the empirical probability density function, coloured red.

3. Draw vertical lines corresponding to the mean and median of the distribution, coloured orange and blue, respectively.

4. What is the relative difference between the mean and median of the distribution. How much proportionally larger is the mean than the median?

1.2 Measures of Position

In addition to measures of central tendency, it is often useful to calculate statistics regarding position within a distribution. Such measures include:

- *minimum* and *maximum* (R functions: `min` and `max`) – minimum and maximum values in an empirical distribution.

- *quantiles* (R function: `quantile`) – the n^{th} quantile is the value such that $n\%$ of the data/distribution lie below that value. The 25^{th}, 50^{th} and 75^{th} quantiles are also called the 4-*quantiles*, the 1^{st}, 2^{nd} and 3^{rd} quartiles, or the *lower quartile*, *median* and *upper quartile*. The 1^{st}, 2^{nd}, ..., 99^{th} quantiles are also called the 100-*quantiles*, or *percentiles*.

Recall the histogram created earlier in the tutorial:

```
x = rnorm(1000)
hist(x,freq=FALSE)
```

By default, the R function `quantile` returns the minimum, lower quartile, median, upper quartile and maximum values (see the `prob` argument for other quantiles). These measures of position can be drawn onto the histogram using the command:

```
abline(v=quantile(x),col="red")
```

Note that all five values returned by the `quantile` function have been drawn as red vertical lines.

Now recreate the variable `z`:

```
z=x^2
```

For comparison with `x`, create a similar view for `z` (i.e. plot a histogram for `z` and add vertical lines using the `abline` and `quantiles` functions). Note that the distribution of `z` is asymmetric.

A simple intuitive view of such information can be achieved using a *box plot* (also called a *box-and-whisker plot*). Create a box plot of `x` using the command:

```
boxplot(x)
```

The median is displayed as a thick black line; the first and third quartiles correspond to the edges of the box; the 'whisker' lines extend out by approximately 1.5 times the difference between the first and third quartiles (or equal to the min/max values); and any values outside the whiskers are identified using circles (these are often called outliers, under the assumption that the data are Normal).

Create a box plot corresponding to `z`, and compare it with the box plot of `x`.

Another useful visual representation is the *empirical cumulative distribution function*, which is the proportion of observations lower than the n^{th} quantile (y-axis) against the n^{th} quantile (x-axis). This stepwise function may be plotted using:

```
plot(ecdf(x))
```

Similarly to above, vertical lines corresponding to quantiles of the distribution can be added to the plot:

```
abline(v=quantile(x),col="red")
```

Note that the theoretical values of the cumulative distribution function of the standard Normal distribution can be achieved using the `pnorm` function. Consequently, the commands:

```
a = seq(min(x),max(x),length.out=100)
lines(pnorm(a)~a,col="blue")
```

result in the theoretical curve being added to the plot, coloured blue. This helps to visualise the agreement between the randomly generated dataset `x` and the theoretical standard Normal distribution.

Note that the above could have been achieved using a single command by utilising R's "`->`" assignment operator:

```
lines(pnorm(seq(min(x),max(x),length.out=100)->a)~a,col="blue")
```

A similar plot can be achieved corresponding to `z`, using the commands:

```
plot(ecdf(z))
abline(v=quantile(z),col="red")
lines(pchisq(seq(min(z),max(z),length.out=100)->a,1)~a,col="blue")
```

which utilises knowledge that `z` follows a χ^2_1 distribution (the reason for this will become apparent later in the Statistics course).

Another useful function is `summary`, which provides a summary of an R object. For example, the command:

```
summary(x)
```

returns the mean of `x` in addition to the values returned by the `quantile` function. The `summary` function is incredibly versatile, providing contextual information that depends on the class of the input object. For example, the command:

```
summary(hist(x))
```

returns information regarding the return values of the `hist` function, along with information regarding the data returned by the particular function call.

Task 3:

1. Recall the 'rivers' dataset. Display the distribution of these data using a histogram, using the argument `breaks=30`, displaying density on the y-axis.
2. Draw a vertical line corresponding to the median of the distribution, coloured red.
3. Identify the 5^{th} and 95^{th} percentiles of the distribution, and draw vertical lines corresponding to these values, coloured blue.

1.3 Measures of Dispersion

Measures of dispersion are also important statistics, which describe the overall 'spread' of the data/distribution. Such measures include:

- *variance* (R function: `var`) – average of the squared differences between the observations and the sample mean.
- *standard deviation* (R function: `sd`) – square root of the sample variance.
- *range* (R function: `range`) – difference between the maximum and minimum.
- *interquartile range* (R function: `IQR`) – difference between the 1^{st} and 3^{rd} quartiles.

Similarly to how the median is a robust statistic describing the central tendency, the interquartile range is a robust measure of dispersion. However, the variance and standard deviation are more often used for reasons of mathematical and computational convenience.

For Normally-distributed data, observations have approximately 95% chance of lying within two standard deviations of the sample mean. Rephrased, since 95% of observations have values between the 2.5% and 97.5% quantiles, this means that the values of the 2.5% and 97.5% quantiles would be approximately two standard deviations from the sample mean, assuming the data are approximately Normally distributed.

Task 4:

1. Display the histogram of the artificial standard Normal data: `x = rnorm(1000)` using the argument `breaks=30`.

2. Calculate the two values that are two standard deviations from the sample mean.

3. Calculate the quantiles that these two values correspond to (i.e. the proportion of the distribution lower than these values). How close are these values to the 2.5% and 97.5% quantiles?

4. Repeat the calculations a few times using new datasets (i.e. recalculate `x = rnorm(1000)` each run). How much do the results vary over a few runs?

2 Comparing the Distributions of Datasets

It is often useful to compare datasets or distributions, such as when:

- comparing the empirical distribution of observed data with a theoretical distribution;
- comparing the overall empirical distributions of two datasets (unpaired);
- comparing two datasets that are paired, i.e. each observation in one dataset directly corresponds to an observation in the other. This is referred to as bivariate data analysis, and will be considered in Section 3.

2.1 Comparing Empirical and Theoretical Distributions

In Section 1.2 we saw how empirical and theoretical distributions can be visually compared by over-plotting their cumulative distribution functions (using `ecdf` and `pnorm`, for Normal data).

Another useful visual tool for comparing distributions is the Q-Q plot (Quantile-Quantile plot), which plots the quantiles from the empirical distribution against their theoretical counterparts. If the empirical distribution is in agreement with the theoretical distribution then all data points should lie on or close to the diagonal line.

Continuing to use the variable `x` containing a vector of artificially-generated random Normal variates, we can compare the empirical distribution of `x` against the standard Normal distribution using the command:

```
qqnorm(x)
```

which creates a Normal Q-Q plot. To aid visualisation of (dis)agreement, the line passing through the 1st and 3rd quartiles can be added to the Normal Q-Q plot using the command:

```
qqline(x)
```

It should be evident that there is a strong agreement between the empirical and

theoretical distributions in this case, particularly in the centre of the distribution, with more noise being present near the edges.

In contrast, creating a Normal Q-Q plot corresponding to `z` (=`x^2`) should clearly indicate that `z` is not Normally distributed. Given information that `z` should follow a χ^2_1 distribution, we can create a Q-Q plot for `z` using the following commands:

```
qqplot(z,qchisq(c(1:1000)/1001,1))
```

Here, we use the `qchisq(c(1:1000)/1001,1)` command to generate a list of 1000 quantiles from the χ^2_1 distribution, against which to compare our dataset `z`.

Note that this procedure can also be used to plot the quantiles of two theoretical distributions against each other. For example, the standard Normal and χ^2_1 distributions can be compared using the commands:

```
a=c(1:1000)/1001
qqplot(qnorm(a),qchisq(a,1))
```

Or, more elegantly, using one command:

```
qqplot(qnorm(c(1:1000)/1001->a),qchisq(a,1))
```

Task 5:

1. The inbuilt 'faithful' dataset contains data regarding the waiting time and eruption duration of a geyser. Create a histogram of the eruption durations, which are stored in the variable 'eruptions'. What do you notice about this distribution?

2. Create a Normal Q-Q plot corresponding to the eruption durations data, to visually assess the Normality of the data. Add a Normal Q-Q line. Can the data be considered to be Normally distributed?

3. Recreate the Normal Q-Q plot with Q-Q line using the transformed data: `abs(faithful$eruptions-3)`. Can the transformed data be considered to be Normally distributed? Can you think of a good reason why such a transformation might be applied to this sort of dataset prior to analysis?

2.2 Comparing (Unpaired) Empirical Distributions

Recreate the variables `x` and `z`, as used earlier in the tutorial:

```
x=rnorm(1000)
z=x^2
```

The Q-Q plot can also be used to compare two empirical distributions. This can be achieved using the command:

```
qqplot(x,z)
```

In this case, it should be clear that the distributions of these datasets are not similar. However, two datasets artificially generated by taking random numbers from the standard Normal distribution should produce Q-Q plots that indicate similarity of those distributions:

```
x = rnorm(1000)
y = rnorm(1000)
qqplot(x,y)
abline(a=0,b=1)
```

Here, the `abline` command draws a 45° line (i.e. intercept `a=0`, gradient `b=1`) to aid visualisation.

In Section 1.2 we visualised the distribution of univariate data using box plots. R allows multiple box plots to be plotted side-by-side, using the same axis scale, allowing intuitive visual comparison of distributions. For example, type the command:

```
boxplot(list(x=x,y=y,z=z))
```

Here, we create a list with three entries (the three vectors containing the datasets), informing the `boxplot` function that separate boxplots for these three datasets should be plotted next to each other, using the same y-axis scale.

Note that assigning the datasets names within the list (i.e. setting `x=x`, `y=y` and `z=z`) allows these names to be displayed as text on the x-axis – the simpler command: `boxplot(list(x,y,z))` would have worked, but the box plots would have been labelled '1', '2' and '3' rather than 'x', 'y' and 'z'.

It should be clear that the two variables containing standard Normal data (`x` and `y`) have similar distributions, whilst the other dataset (`z`) has a dramatically different distribution.

Note that the above box plot can also be created by converting the data to two vectors, where the second vector indicates the 'factor', e.g.:

```
a = c(x,y,z)
b = c(rep("x",length(x)),rep("y",length(y)),rep("z",length(z)))
boxplot(a~b)
```

Task 6:

1. The inbuilt 'PlantGrowth' dataset contains results from an experiment to compare plant yields obtained under a control and two different treatment conditions. The plant yields are stored in the variable 'weight'. Display a box plot corresponding to these data.

2. Data regarding the treatments are contained in the variable 'group'. Using a single plotting command, display side-by-side box plots of the distribution of weights for each of the three groups. Is there any visual evidence for differences between the three treatment groups?

Task 7:

1. The inbuilt 'chickwts' dataset contains the results of an experiment to measure and compare the effectiveness of various feed supplements on the growth rate of chickens. The chicken weights are stored in the variable 'weight'. Display a box plot corresponding to these data.

2. Data regarding the differing feed supplements are contained in the variable 'feed'. Using a single plotting command, display side-by-side box plots of the distribution of weights for each of the groups. Is there any visual evidence for differences between the groups? Are there any feed supplements that seem to result in particularly light or particularly heavy chickens?

Task 8:

1. The inbuilt 'DNase' dataset contains data obtained during development of an ELISA assay for the recombinant protein DNase in rat serum. The dataset contains columns pertaining to optical density at a given concentration, over 11 experimental runs. Display a box plot corresponding to the distribution of the optical density.

2. Display side-by-side box plots of the distribution of the optical density over all experimental runs, for each of the 8 concentrations. Does there appear to be a relationship between optical density and concentration?

3 Bivariate/Multivariate Data Analysis

So far we have only considered the comparison of univariate datasets, i.e. when there is no direct relationship between individual observations in the compared datasets. However, it is often interest to identify whether there are any relationships between paired variables within or between datasets. In the case where there are only two variables, this is called *bivariate data analysis*; where there are more than two variables it is called *multivariate data analysis*.

3.1 Bivariate Data Analysis – Investigating Correlations

The most common way of visualising the relationship between two paired variables is by using a scatterplot, allowing visual identification of any relationships that may exist between the two variables. For example, type the commands:

```
x = rnorm(1000)
y = rnorm(1000)
plot(y~x)
```

The fact that there is no relationship between these two variates is clarified by the lack of any visually observable correlation in the plot. Now type the commands:

```
x = rnorm(1000)
z = x + rnorm(1000)
plot(z~x)
```

Now, logically there should be a positive linear relationship between `z` and `x`, given that `z` is explicitly dependent on `x`. This assertion can be visually confirmed by looking at the plot of `z` against `x`, which clearly depicts a positive linear dependency.

The *Pearson product-moment correlation coefficient*, or simply the *correlation*, denoted by r, is a measure of the linear dependency between two variables. The correlation is bounded by $[-1, 1]$, with positive values indicating a positive correlation, negative values indicating a negative correlation, and values close to zero indicating little/no correlation. Plots that demonstrate a strong positive correlation tend to have many data points in the upper right and lower left quadrants, and few data points in the upper left and lower right quadrants (and the converse is true for plots demonstrating a strong negative correlation).

The correlation between two variables can be achieved using the `cor` function. Looking at the correlation between `x` and `y`, and `x` and `z`, using the commands:

```
cor(x,y)
cor(x,z)
```

we can identify that there is no correlation between `x` and `y`, whilst `x` and `z` are positive correlated. Consequently, we can use the Pearson correlation r to quantitatively distinguish between correlated and uncorrelated variables, and distinguish between degrees of correlation. Importantly, such quantification does not rely on visual interpretation, which can be subjective.

However, it is important to remember that the Pearson correlation coefficient is only well-suited to the identification/quantification of linear relationships. In cases where there is a non-linear relationship between variables, a *non-parameteric* correlation coefficient, such as the *Spearman rank correlation coefficient*, may be more suitable. The Spearman correlation coefficient is defined as the Pearson correlation coefficient of the variables after converting them to *ranks*. This essentially means that the Spearman coefficient rewards a monotonic relationship, independent of linearity, at the expense of power. To illustrate this, consider the following example:

```
x = rnorm(1000)
y = exp(x)
plot(y~x)
cor(x,y)
cor(x,y,method="spearman")
```

Note that the spearman correlation is equal to 1, whilst the default (Pearson) correlation is considerably less than 1 (for a more extreme example consider the case where `y = exp(exp(x))`). Consequently, it is always important to consider the nature of the data when deciding on appropriate statistics from which to draw conclusions.

Relationships between variables will be further discussed later in the Statistics course as part of the Linear Regression topic.

Task 9:

1. Load the 'boot' library by typing: `library(boot)`
2. The dataset 'calcium' within the 'boot' library contains data relating to an experiment for the biochemical analysis of intracellular storage and transport

of calcium across plasma membrane. The dataset records the calcium uptake against the time that the cells were suspended in solution. Create a scatterplot of calcium uptake against time.

3. Calculate the (Pearson) correlation between calcium uptake and time.

4. Calculate the Spearman correlation coefficient between calcium uptake and time. Is there a large difference between the Pearson and Spearman correlation coefficients? Can you explain any difference (or lack of difference) between them?

Task 10:

1. Load the 'boot' library by typing: `library(boot)`

2. The dataset 'survival' within the 'boot' library contains data relating to the survival percentages of batches of rats who were given varying doses of radiation. The dataset records the survival rate (as a percentage) against the radiation dose. Create a scatterplot of survival rate against dose.

3. Calculate the (Pearson) correlation between survival rate and dose.

4. Calculate the Spearman correlation coefficient between survival rate and dose. Is there a large difference between the Pearson and Spearman correlation coefficients? Can you explain any difference (or lack of difference) between them?

5. Apply the transformation: $f(x) = \log(1/x)$ to the survival rate field. This can be achieved by creating a new variable: `a = log(1/survival$surv)`. Create a scatterplot of the transformed survival rate (`a`) against dose.

6. Recalculate the Pearson and Spearman correlation coefficients, this time between the transformed survival rate (`a`) against dose. What is the difference between the new and old Pearson correlation coefficients? What is the difference between the new and old Spearman correlation coefficients? Can you explain the difference (or lack of difference) in the changes in these statistics?

3.2 Simple Multivariate Data Analysis

Datasets often contain more than two paired variables. From the point of view of visual exploratory data analysis, the difficulty with such cases is that the relationship between more than two variables cannot be easily represented using a simple 2-dimensional plot. Furthermore, variables may be continuous (e.g. floating point responses), discrete responses (e.g. integer or boolean) or controlled factors – the nature of the dataset determines how it should be analysed.

If a dataset (a 'data.frame') contains $n > 2$ variables/columns then the versatile R function `plot` will by default plot an $n \times n$ array of pairwise scatterplots corresponding to all possible variable-pair combinations, which can be useful for preliminary exploratory data analysis.

Consider the inbuilt dataset 'iris', which contains measurements of the sepal and petal length and width of 50 flowers from each of 3 species of iris. The command:

```
plot(iris)
```

creates a 5×5 array of scatterplots of the 5 variables (4 responses and 1 factor). A glance at this plot reveals which variables are most correlated (e.g. Petal.Width and Petal.Length), which display obvious clustering (e.g. Petal.Width and Petal.Length), and which factors are responsible for such clustering (the *setosa* species is clearly distinct from the *versicolor* and *virginica* species).

We might then choose to display multiple box plots allowing visualisation of the distributions of the four response variables at each of the Species levels – multiple side-by-side box plots can be useful for visually investigating the affect of different factors (independent variables whose values are discrete and can be categorised). This can be done using the sequence of commands:

```
quartz(width=7,height=7)
par(mfrow=c(2,2))
boxplot(iris$Petal.Length~iris$Species,main="Petal.Length")
boxplot(iris$Petal.Width~iris$Species,main="Petal.Width")
boxplot(iris$Sepal.Length~iris$Species,main="Sepal.Length")
boxplot(iris$Sepal.Width~iris$Species,main="Sepal.Width")
```

Here, the `quartz(width=7,height=7)` command opens a new graphics device window with the specified size in inches (availability of `quartz` may depend on the system – if your system doesn't support `quartz` then you can use the `x11` function as an alternative). The `par(mfrow=c(2,2))` command specifies for the graphics window to be split into a 2×2 array of subfigures, allowing four separate figures to be simultaneously viewed.

Looking at the four box plots, it is clear that Petal.Length and Petal.Width are able to distinguish between *setosa* and the other two species, having much shorter Petal.Length and Petal.Width values. Perhaps interestingly, the Sepal.Width of *setosa* appears to be noticeably larger than for the other two species, on average.

If provided with a data frame containing n numerical columns, the `cor` function will return an $n \times n$ matrix containing the pairwise correlations corresponding to each variable-pair combination. In this case, the command:

```
cor(iris[,1:4])
```

returns the 4×4 correlation matrix describing the pairwise correlations between the response variables. This agrees with the previous assertion that Petal.Length and Petal.Width are highly correlated, and also indicates the correlation of Petal.Length and Sepal.Length. Furthermore, Sepal.Width is recorded as being negatively correlated with Petal.Length and Petal.Width, and Sepal.Length and Sepal.Width appear relatively uncorrelated (and negatively correlated, if at all).

We will now consider an example using the 'poisons' dataset in the 'boot' library. If you haven't done so already, load the 'boot' library by typing:

```
library(boot)
```

The 'poisons' dataset contains data regarding animal survival times after been poisoned with one of three poisons, and treated with one of four treatments. The

command:

```
boxplot(poisons$time∼poisons$poison)
```

produces a multiple box plot displaying the distributions of survival times corresponding to each of the three poisons. It is visually evident that poison 3 systematically results in the shorter survival times, whilst poison 1 seems to result in longer survival times, on average. In an attempt to further explore poison 1, we can display box plots corresponding to the four treatments given poison 1:

```
boxplot(poisons$time[poisons$poison==1->a]∼poisons$treat[a])
```

We can see that, for poison 1, treatment A results in the shortest survival times, whilst treatment B results in the longest survival times.

Expanding our attention to also include poisons 2 and 3, we can display box plots corresponding to the four treatments, for each of the three poisons:

```
colour = c(rep("red",4),rep("blue",4),rep("orange",4))
boxplot(poisons$time∼poisons$treat+poisons$poison,col=colour)
```

Here, we specify the sum: `poisons$treat+poisons$poison` as the factor, which automatically expands to all possible pairwise combinations (=12 levels in this case). We also colour the box plots red, blue and orange, corresponding to poisons 1, 2 and 3, respectively, aiding visual clarity. Visual inspection of the resulting box plots confirms that the behaviour observed for poison 1 holds for poisons 2 and 3 also – given a particular poison, treatment A results in systematically shorter survival times, whilst treatment B results in systematically longer survival times.

We may now be interested to know whether, given a particular treatment, poison 3 always results in systematically lower survival times. To answer this visually, we can reorder and recolour the box plots:

```
colour = c(rep("red",3),rep("blue",3),rep("orange",3),rep("green",3))
boxplot(poisons$time∼poisons$poison+poisons$treat,col=colour)
```

making it easy to conclude that poison 3 does always result in systematically lower survival times, given a particular treatment.

Task 11:

1. Recall the example that focussed on the 'iris' dataset. Using this dataset, calculate the three separate correlation matrices of the four response variables, which correspond to the three levels in the Species factor.

2. Is the previous assertion that Petal.Length is highly positively correlated with Petal.Width justified?

3. Is the previous assertion that Petal.Length is highly positively correlated with Sepal.Length justified?

4. Is the previous assertion that Petal.Length and Petal.Width are negatively correlated with Sepal.Width justified?

5. Is the previous assertion that Sepal.Length and Sepal.Width are uncorrelated (but negatively correlated, if at all) justified?

6. What conclusions can you draw from this exercise?

Task 12:

1. The inbuilt dataset 'CO2' contains data from an experiment on the cold tolerance of the grass species *Echinochloa crus-galli*. The dataset records the carbon dioxide uptake rates (response), ambient carbon dioxide concentration (independent variable), and three factors (Plant, Type and Treatment). Create a multiple box plot showing the distribution of CO2 uptake rates for the two types 'Quebec' and 'Mississippi', side-by-side. Does it appear that there is a difference between the CO2 uptake rates for plants originating from the different regions?

2. Create a multiple box plot showing the distribution of CO2 uptake rates for the twelve plants, side-by-side. Given knowledge that the first six plants originate from Quebec, and the latter six from Mississippi, colour the box plots corresponding to Quebec red, and those corresponding to Mississippi in blue.

3. Create a multiple box plot showing the distribution of CO2 uptake rates for the seven concentrations, side-by-side. How does ambient CO2 concentration appear to affect CO2 uptake?

4. Create a multiple box plot showing the distribution of CO2 uptake rates for the seven concentrations, for each of the two types, side-by-side (14 box plots total). It would seem that CO2 uptake is dependent on concentration – does this information reinforce or change your previous assertions regarding the difference between the CO2 uptake rates for plants originating from the different regions?

5. Create a multiple box plot showing the distribution of CO2 uptake rates for the two treatments 'nonchilled' and 'chilled', side-by-side. Does it appear that there is a difference between the CO2 uptake rates for plants exposed to different treatments?

6. Create a multiple box plot showing the distribution of CO2 uptake rates for the two treatments, for each of the two types, side-by-side (4 box plots total). Colour the box plots so that those corresponding to the 'nonchilled' treatment are coloured red, and those corresponding to the 'chilled' treatment are coloured blue.

7. Create a multiple box plot showing the distribution of CO2 uptake rates for the two treatments, for each of the two types, for each of the seven concentrations, side-by-side (28 box plots total). Colour the box plots so that those corresponding to nonchilled:Quebec, chilled:Quebec, nonchilled:Mississippi and chilled:Mississippi are coloured red, blue, orange and green, respectively. According to visual inspection, which factor seems to be more inhibitive for CO2 uptake: Type, Treatment or Plant?