QUEEN MARY, UNIVERSITY OF LONDON

MTH6102: Bayesian Statistical Methods

Practical 2

2023-2024

These practicals contain R commands to work through which implement and illustrate the methods covered in the lectures. Please try other examples and look in the help files for the commands to see other options for their use. It's better to type the commands yourself rather than copying from this file, so that you are more likely to remember the syntax.

Within R or RStudio, commands may be entered and run one at a time in the console; or in a script, from which one or more lines can be run, and which you can save to run again or modify. It is good practice to have a copy of any analysis done, so you should use a script. In R, a new empty script can be opened using the menu option `File -> New script`. In RStudio, the corresponding option is `File -> New file -> R script`.

To run a command, select the lines you want to run and then click `CTRL+R` in R, or `CTRL+ENTER` in RStudio. Scripts are saved using the extension `.R`. There is also a menu option `File -> Open script` to open an existing script.

Simple graphs

Some commands for creating graphs are listed below:

- `plot` - create a new plot, can be lines or points; from two vectors; and from columns in a dataframe; from a function; ...
- `lines` - add a line to existing plot;
- `points` - add points to existing plot;
- `curve` - plot a function, either a new or existing plot.

The R code that follows illustrates the use of these commands. Try experimenting with them to make sure that you understand how they work.

```
plot(x=iris$Petal.Length, y=iris$Sepal.Length)

r = seq(from=-1, to=7, length=201)
s = sin(r)
t = cos(2*r)

plot(x=r, y=s)

plot(x=r, y=s, type="l")

plot(x=r, y=s, type="l", lwd=2)

plot(x=r, y=s, type="l", col="red")
lines(x=r, y=t, col="blue")

curve(sin, from=0, to=2*pi, col="red")
curve(cos, from=0, to=2*pi, col="blue", add=TRUE)
```

In the above, `type="l"` stands for "line", so it is a letter l, and not the number 1.

There is a (long) list of graph options if you type `help(par)`. It can be difficult to find what you need there. We will cover a few options to change the appearance of graphs later.

There is an entirely different way of producing graphs, using the `ggplot2` package, which many people prefer the appearance of:
`https://ggplot2.tidyverse.org/`

We don't cover this, as the code such as we used above is more straight-forward for simple graphs. There is a free book mentioned in the link above, R for Data Science. The chapters on "data visualisation" and "graphics for communication" use `ggplot2`.

Importing datasets

To import a dataset, we can use the `read.table` command. There is a file on QMPlus called "practical2.txt". Download this file and save it somewhere. Before importing the file into R, it is convenient to change the working directory to the folder where the file was saved. The command to change directory is:

```
setwd("...")
```

where `"..."` is replaced with the name of the folder location. This folder name must use forward slashes /, not back slashes \, so these need to be changed when copying and pasting folder locations (at least on Windows PCs).

Then the command to import the dataset is:

```
data = read.table("practical2.txt", header=TRUE)
```

The `header=TRUE` option says that the first line of the file contains the column name(s), which is the case for this dataset.

This dataset contains ozone measurements over time. The two columns are called "t" and "ozone", and as before we can use the symbol `$` to refer to them.

Try plotting the ozone concentration against time.

Numerical summaries for all variables can be calculated using `summary(data)`, or for a single column commands such as these can be used:

```
summary(data$ozone)
mean(data$ozone)
var(data$ozone)
```

The function `mean`, when applied to a sample $y_1, \ldots, y_n$, gives the sample mean $\bar{y}$, and `var` gives the sample variance

$$\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2.$$

To display or use the number of rows or columns in a dataset, we can use these commands:

```
nrow(data)
ncol(data)
n = nrow(data)
```

For example, some of the formulae that we have derived such as MLEs use the sample size n.

Binomial likelihood

In the lectures, we covered a binomial example, where the observed data was k, with $k \sim Binom(n, q)$.

Plot the likelihood for a given k and n as a function of q, which is the unknown parameter. For example, the illustrations in the lecture slides were for $n = 40, k = 12$.

There is more than one way of doing this. We could generate a vector for q and a separate vector for the likelihood, and plot these. The following command generates a vector of 100 values for q between 0 and 1:

```
q = seq(from=0, to=1, length.out=100)
```

Suppose that variables n and k have been given values in the R code. The binomial cofficient $\binom{n}{k}$ can be calculated using

```
choose(n, k)
```

Generate a vector containing the binomial likelihood using the vector q, then plot this likelihood against q.

Then calculate the log-likelihood and plot that against q.

Calculate the maximum likelihood estimate (MLE) $\hat{q}$, by hand not in R, for this example. Check on the graph that the likelihood does appear to take its maximum at this value.