# R skills review

as at 1 September 2018

## *Why is this needed?*

For our specialist and international workshops, we make extensive use of R statistical software, and we expect participants to be familiar with R. We will be using JAGS, and the JAGS language is similar to R. If two or three people come along with no knowledge of R, we can't take a couple of hours of everyone's time to get them started.

Even people who have used R a lot for their own work may not have used some of the specific things we need for the workshop, such as lists, matrices, random number generators, or loops. So this review acts as a check list of the techniques you will need during the workshop.

All of these exercises can be done with the basic R functions; you do not need to use any contributed packages.

**All the skills in the review will be needed for the workshop. If there is an exercise that you can't do, please take the time to learn how to do it before submitting your reply. You should have code which works for *all* the exercises, even if it isn't the most elegant code.**

## *What should I do?*

Please write a script with code for each of the exercises below; we use scripts during the workshop, so ability to use scripts is itself an essential skill.

Please insert comments in the script to indicate the exercise numbers. You do not need to include the R output in the script (except for the last exercise); we will run your code in R and see what it produces.

The script must be your own work. We can't assess *your* ability to write R code if it is copied from someone else's work, and it defeats the objective of ensuring that all participants are competent in R coding.

## *The Exercises:*

### *1. Creating and annotating a script*

Open R and start a new script. At the top of the script insert comments with

(a) your name; and
(b) the workshop you wish to attend.

### *2. Simple arithmetic*

An easy one to begin with! Write code to add together 3 and 5 and divide the total by 7. Assign the result to an object called "`s1`" and display the value of `s1`.

### *3. Creating vectors*

Create three vectors:

`v1` should contain the three values 1.2, 3.4 and 5.6;
`v2` has the names Abid, Beth and Chan;
`v3` has the integers from 3 to 53.

Display all three. We'll use these vectors for later exercises.

### *4. Indexing with numbers*

Write code to display

(a) the 2nd element in `v1`;
(b) all except the 2nd element in `v2`;
(c) the 7th, 8th, 9th and 10th elements in `v3`.

### *5. Indexing with names*

'`islands`' is a built-in data set in R; display it in the Console and check the details with `?islands`; you will see that the elements have names.

(a) Use the *names* to display the area for just Borneo island.

(b) The areas are arranged in alphabetical order; create a new object called '`islands2`' with the values in order of size (smallest to largest).

(c) display the area of just Borneo using `islands2`.

### *6. Indexing with logical vectors*

We have data for counts of magpie robins at 25 sites and the dates of the surveys; copy-paste the following data into your script and run it in R to create two vectors:

```
 count <- c(1, 1, 0, 0, 1, 2, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 2, 0, 1, 0, 0, 0, 1, 0)
 date <- c(62, 92, 70, 0, 83, 66, 78, 82, 61, 0, 75, 86, 0, 0, 94, 72, 88, 68, 0, 99, 76, 73, 90, 69,
60)
```

There are two problems with these data that we need to fix:

(a) The dates are Julian dates, with 1 = 1st Jan, but it has zeros. We investigate and discover that zero indicates that the site has not been surveyed. Write code to change the zeros in the '`date`' vector to '`NA`'.

(b) If the site has not been surveyed, the '`count`' is recorded as zero (though sometimes the actual count of magpie robins was zero). Write code to change the entries in '`count`' to '`NA`' for the sites not surveyed.

### *7. Creating a data frame*

A rain gauge at the top of Mount Hujan is checked every month.

- The rainfall readings for Jan, Feb, Mar and Apr were 1.2, 2.3, 3.4 and 4.5 respectively.
- In Jan, snow was lying around the rain gauge, but not in the other months.
- The gauge was read by Abid in Jan and April, by Beth in Feb and by Chan in March.

Create a data frame called "`df1`" with these data. We'll use this data frame for later exercises.

### *8. Object classes*

Write code to display the classes of `v1`, `v2`, `v3` (from Exercise 3) and `df1` (Ex.7), and each of the columns in `df1`.

### *9. Reading in data from a file*

During the workshop, we'll ask you to download data files and read them into R. To ensure this works smoothly for everyone, we ask you to do this now. Go to http://bcss.org.my/forms/R_skills.htm for the link to download the "R_skills_sox.csv" file.

(a) Write code to read in the "R_skills_sox.csv" file and assign the data to an object called `'sox'`.

*Note: You do not need to edit the CSV file before reading it in.*

(b) Check that `sox` has 6 columns. (If it has only 2 columns, you need to fix this before going further!)

### *10. Displaying data frames*

The data frame `df1` is small and can conveniently be displayed in the Console, but that's not good for big data frames. Write code to

(a) display the first 6 rows of `sox` in the Console;

(b) display a summary of `sox` in the Console;

(c) display the whole `sox` data frame *in a separate R window*.

### *11. Adding columns to data frames*

The rainfall figures in `df1` are in inches. Add a column to `df1` with the rainfall in millimetres (1 inch = 25.4 mm) and display the whole data frame in the Console.

### *12. Subsetting data frames*

We discover that Gail in the `sox` data set usually wears spectacles, but she had left them behind when she did the experiment – everything beyond 1m was a blur. We decide to exclude her results from the data set. Write code to create a new data frame (call it "`sox1`") with all the rows in `sox` except those for Gail.

### *13. Working with factors*

Categorical variables are important, and R uses factors to represent these.

(a) Display the class of each of the columns of `sox1`.

(b) Write code to calculate the proportion of successes (result = 1) for each person in `sox1` (hint: check `?tapply`); you may see that "Gail" still appears in the result with a value of `NA`.

(c) Check the levels of the `Name` column in `sox1` and change the levels to exclude "Gail"; rerun (b).

(d) Create a new column in `sox1` with the die score converted to a factor.

(e) We want to see if there's a correlation between the die score and success in getting the sock in the box. Use the new column to calculate the proportion of successes (result = 1) for each of the die scores, 0 to 9, and assign the result to an object called "`dieResult`" – we'll plot it later.

### *14. Working with lists*

Lists allow us to combine disparate objects into a single object:

(a) Create a *named* list called "`list1`" with four components: `v1`, `v2`, and `v3` (from Ex.3) and `df1` (Ex.7).

(b) Display the names of the components of `list1` in the Console.

(c) Use index numbers to extract the 2nd and 3rd components from `list1` and assign to a new object called "`list2`"; display `list2` in the Console.

(d) Use index numbers to extract the first component of `list1` and multiply it by 2.

### *15. Simulating a sample with random numbers*

R is great for simulating data to try out analysis methods.

(a) Simulate a sample of size $n = 10$ from a normal distribution with mean 90 and standard deviation 10.

(b) Calculate the sample mean, the sample standard deviation (SD), and the standard error of your estimate of the population mean (SE) and display them in the Console.

### *16. Using a "for" loop to simulate 1000 samples*

One simulated sample doesn't tell us very much, but we can easily do 1000 samples.

(a) Write code for a "for" loop to simulate 1000 samples like those in the previous question (ie, samples of size $n = 10$ from a normal distribution with mean 90 and standard deviation 10); the result should be a *matrix* with 1000 rows, and columns for mean, SD and SE; call it "`simsResult`".

*Note: It's possible to do this without using a loop (with* `apply`*), but we will need to use loops during the workshop, so please use a loop for this exercise.*

(b) Calculate the means of the columns in `simsResult`.

(c) Because this is a simulation, we know the true mean and SD for the population. Calculate the root mean squared error (RMSE) for the sample means and the sample SDs in `simsResult`.

### *17. Basic plots*

R is good at producing simple plots.

(a) Write code to use the `dieResult` object from Ex. 13 for a scatter plot of proportion of successes against the die score; bonus points for adding a trend line!

(b) Use the `simsResult` object from Ex. 16 to do histograms of the sample means, SDs and SEs.

(c) Use R's built-in data set `chickwts` to do box plots showing the weights of the chicks for each type of feed supplement used.

### *18. Details of your session*

Finally, do `sessionInfo()` in R and copy-paste the output from `sessionInfo()` into your script.

## *Submitting your answers*

Save your script as "R_skills_<your_name>.R".

Check your script by "sourcing" it in R (File > Source R code… (Windows), File > Source File… (Mac)) – it should not produce errors.

Email your script to **workshops@bcss.org.my** with "R skills review" in the subject line. Some email systems won't let you send .R files; you can get around this by changing the file to a .TXT file.

You should get a reply within four days; if not, please email mike@mikemeredith.net with details.

**Thank you for your cooperation in completing the skills review exercises!**