GUIDANCE FOR TEACHING R PROGRAMMING TO NON-STATISTICIANS

LANGAN, Dean and WADE, Angie
Great Ormond Street Institute of Child Health,
University College London, London
d.langan@ucl.ac.uk

The Centre for Applied Statistics Courses (CASC) at University College London (UCL) provide short courses on statistics and statistical software packages. Popular day-courses include a well-established 'Introduction to R' course and the newly developed 'Further Topics in R'. In the latter, attendees are taught intermediate-level topics such as loops and conditional statements. Attendees range from postgraduate students, academic researchers and data analysts in the private sector without a strong background in statistics or programming. First, we highlight some issues with providing our training course to this demographic, derived from our experience and from anonymous online feedback. Second, we discuss some of our solutions to these issues that have shaped our course over time. For example, one issue is catering to a wide audience from differing fields, different levels of computer literacy and approaches to learning. To address this, we prepare for a high level of flexibility on the day and include intermittent practical exercises to get real time feedback on the abilities of attendees. Finally, we reviewed the experiences of other teachers on similar courses documented online and compared these experiences with our own. We offer guidance to other teachers running or developing courses for intermediate-level R programming.

INTRODUCTION

In a data driven society, it is becoming increasingly important to have both statistical and computer programming literacy. However, students may not have anticipated this when choosing their area of study, nor professionals when choosing a direction in their career pathway. This can lead to a barrier and the student or professional consequently finding themselves in a position where they need to attend a statistics or computer programming course. For example, university researchers may have a firm grounding in their chosen field, but need to acquire knowledge of statistics and the use of appropriate software to perform analyses, in the absence of having a qualified statistician as part of their project team. In particular, researchers may benefit from having computer programming skills in languages such as R (R Core Team 2016).

The UCL Centre for Applied Statistics Courses (CASC) currently run 16 different short courses for statistics and statistical software (http://www.ucl.ac.uk/ich/short-courses-events/about-stats-courses). The courses are primarily aimed at those without a strong background in statistics or programming but who require a basic knowledge for their research. We market to those that often have limited time for learning in their busy work schedule. Attendees are usually diverse, including university students, internal researchers and external working professionals. Generally, our courses use real data examples and little mathematical formula, an approach also effective for adult learners in the workplace (Westbrooke & Rohan 2012).

We frequently offer a one-day *Introduction to R* course for those with no prior experience in R and only a basic knowledge of statistics. In addition, a newly developed one-day course named *Further Topics in R* is aimed at those who have already been on our introductory course or learnt the basics of R elsewhere. Both courses are held in a classroom with a projector screen and access to 20 desktop computers, but attendees are welcome to bring their own laptops. Given that our attendees often have limited prior experience of programming or statistical analyses, these R courses are taught in such a way that little prior knowledge of either is required, allowing the participants to concentrate on learning the software. Mascaró et al. (2014) also advocate this approach, stating that computational tools and software "generally not only do not help but rather hamper the learning of the statistics concepts as well as of the use of the tools".

In this paper, we focus on our *Further Topics in R* course, give an account of its development and discuss the lessons learnt along the way. We compare our experience with experiences that other teachers of similar courses have documented online. We provide guidance for teaching R to adult learners without a strong background in statistics or computer programming. Most of the guidance is also applicable to teaching other statistical programming languages.

In H. MacGillivray, M. Martin and B. Phillips (Eds.). Proceedings of the Ninth Australian Conference on Teaching Statistics, December 2016, Canberra, Australia.

COURSE STRUCTURE

Topics covered in this course are considered accessible only to the intermediate level programmer. We cover how to:

- Organise and merge multiple datasets.
- Write conditional commands (where code is only executed subject to a certain statement being true).
- Write loops (a way to repeat a sequence of commands under certain conditions).
- Create new functions (a function contains hidden code that serves some purpose, e.g. the function called *mean*, as you might guess, calculates the mean).

LEARNING OUTCOMES

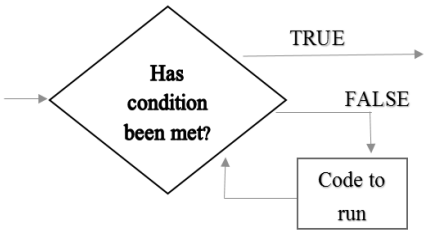
Attendees at the end of the course should have a basic understanding of how R code is structured within these topics above, and have had the opportunity to practice writing code for themselves. However, the primary aim of the course to address any fear that attendees might have of intermediate-level programming and make them more comfortable with further independent learning and the topics outlined above form a framework for this process. Given that a single day is not nearly enough time to cover everything that R has to offer, or even become completely comfortable with all of the topics we cover, it is important that we convey this more general aim. The course also aims to motivate the learner to continue using R for their own project and further develop their skills independently, even if they do not make direct use of the course material. This is achieved by emphasising the breadth of potential that R has. The course reflects the way the first author learnt R programming over the last seven years, which involved formal training only in the beginning but a desire instilled to continue learning through experience.

METHOD OF TEACHING

Each section starts with explaining how the material might be useful for a typical statistical analysis. For example, in the section on loops, we explain that writing a loop in your code can help to produce the same graph for all variables in a dataset. It is important to emphasise 'why this is useful' throughout the course so that attendees feel extrinsically motivated (Coleman 2007); this is known to have a bigger effect on learning in adults than in children (Ormrod 2006). Without relevant examples, there is a danger that the topics covered can seem to be of little practical use.

Next, we introduce the general structure of code and include a diagram to show how this is processed in R. For example, to create a loop in R using the *while* command, the general code structure is:

```
while (conditional statement) {
      ...code to repeat...
}
```



We have found it is more effective to show the general code structure briefly so that any examples that follow can be related to the general context. However, emphasis should be on learning through examples of increasing complexity. We make the first example as simple as possible, even if it does not have any practical application. For instance, we show how a loop can output the numbers 1 to 5:

```
i <- 1 # define a starting value for i
while (i<=5) {
      print (i)
      i <- i + 1 #add 1 to i for every loop
}
```

A loop is not strictly necessary to produce the numbers 1 to 5, but it leads to further examples where loops are required and these are based on real data that attendees find relevant. This approach means that many attendees can more readily transfer the knowledge gained in this course to another context. Example data can be highly tailored when all or most attendees are from the same university department or workplace (Westbrooke & Ellis 2014), but this is not possible for us given our attendees are more heterogeneous. Empirical research has shown students do not naturally generalise what they have learnt to new scenarios (Lovett & Greenhouse 2000, Reed et al. 1985). Using relevant data for R is also advocated by Eglen (2009). For the final example on loops, we go through step-by-step how a loop can be used to read in many datasets from the same folder, explaining that this saves time and prevents errors.

Each section ends with a short exercise to give attendees the opportunity to practice and allows time at the end to go through the solutions as a group. As Lovett and Greenhouse (2000) explain, "students learn best what they practice and perform on their own" and "learning is more efficient when students receive real-time feedback on errors". We give attendees one-on-one help during the exercises and this allows us to identify any common issues and errors in their programming code. Common errors are then discussed as a group, with reasons why the errors occur and how to correct them. This encourages active and collaborative learning, which is known to increase motivation and retention of knowledge (Marriott et al. 2009). An active environment is particularly important for this course, given that the programming language taught has a steep learning curve that can only be tackled by making mistakes and learning from them.

GUIDANCE FOR TEACHING R PROGRAMMING TO NON-STATISTICIANS

In this section, we provide guidance for teaching R programming to non-statisticians, split into nine key points. These are derived from our experience of developing and running the *Further Topics in R* course and from anonymous feedback received from attendees after the course. Feedback is recorded through an online feedback platform (http://opinio.ucl.ac.uk).

Furthermore, we searched online to find where others have documented their own personal experiences from teaching R programming. We reference them here and highlight any similarities and differences from our own experience and guidance. We found four articles (Bååth 2010, Eglen 2009, Mascaró et al. 2014, Westbrooke & Ellis 2014), one blog post (Peruvankal & Muenchen 2014) and one conference presentation (Levy 2016).

1. Recognise variation in learning styles of attendees

We have observed two distinct learning styles from attendees during the taught demonstrations. Some prefer an active approach, which involves writing and executing similar code to that demonstrated on the projector screen. Others prefer a passive approach; they simply observe the teacher and wait for exercises at the end of each section for hands-on practice. Both approaches have their merits and so attendees should opt for the approach which they feel suits them best. However, we received following online feedback "I found the pace was to fast during the taught sections. I struggled to keep up as the teacher typed too fast". The attendees appears to have struggled because they chose the active approach to learning, when they would have been more suited to the passive approach. A similar issue has been observed by Peruvankal & Muenchen (2014) from experience running an introductory course in R. Learning is less efficient when attendees multitask (Ellis et al. 2010), but there is a common misconception that it increases productivity.

To address this issue, we make clear at the start of the course and throughout that, (1) attendees are not required to copy the demonstrations, (2) the code is given to them in the printed course material and (3) there will be time for practice during the exercises. Furthermore, as suggested by Levy (2016), the code on the projector screen can be streamed to the computers of all attendees allowing them to simply copy and paste if they prefer an active approach.

2. Address any timing issues

After our course, attendees are asked in the online feedback questionnaire "Was one day enough time to cover all the material?" We received 25 responses to this question from the first two times the course ran in April and June 2016; 7 (28%) thought that there was more than enough time to properly cover the material, 7 (28%) thought the timing was right and 11 (44%) though there was not enough time. We feel this range of responses to this question reflects the range in abilities of attendees

and simply adding or reducing material on the course will not solve this issue. To cater for a wider audience, we split each exercise into several parts that increase in difficulty. Attendees are reminded that completing the whole exercise is not essential as the solutions will be discussed as a group shortly afterwards. We benefit from having at least one other teacher available to offer one-to-one support in a class of size 20. It is important to be flexible on the day, given that the number of attendees that struggle with some of the material cannot be fully anticipated in advance. We have found that having a plan for all scenarios is useful so that topics and exercises can be skipped if there is insufficient time.

3. Address bottlenecks that stop the course progressing

Related to the issue of timing, some who thought the course was too rushed may have thought so because much time was spent loading datasets and setting up the R software ready for use. It is necessary for all attendees to have read in the necessary datasets before continuing with subsequent sections of the course. To minimise the time spent on this task, we give USB drives to the attendees at the start of the course to ensure that everyone is working from the same folder (including those choosing to bring their own laptops). The USB drives contain the datasets and a file that contains all datasets combined (an *Rdata* file). This also ensures that we can promptly help attendees get back on track if there are any computer issues that may result in losing some of their work.

4. Use R output for more effective teaching

From our experience, teaching is always more effective when the emphasis is on the output that R produces, graphical output or otherwise. Attendees are more likely to understand how the code works if they are shown the output first, before it is explained line-by-line. Furthermore, we have found that attendees learn more effectively if the output is visual and interesting. For example, when teaching how to program a loop, we add two additional functions into our examples:

i. *Sys.sleep*: This is a function that sends R to sleep for given amount of time and therefore slows down the speed that the loop is processed. Attendees can then see clearly how the code works.
ii. *txtProgressBar*: Given the loop is slowed down by the function above, it is also useful to output a progress bar so that attendees can visualise how long it may take for the code to fully execute.

Emphasis on graphical output also teaches attendees to explore and visualise their data, which is an important skill in statistical learning (Westbrooke & Ellis 2014).

5. Explain code step-by step

Fortunately, the R programming language allows you to run small sub-sections of code and produce output. This helps to explain how code works. For example, the *if* statement below returns the value 4 because the conditional statement inside the brackets is true.

```
> x <- 1
> if (x==1) 4
[1] 4
```

To decompose how this line of code works, we can extract the conditional statement (`x==1`) and show explicitly this condition is TRUE:

```
> x==1
[1] TRUE
```

This is a teaching style that we found effective from experience. We have since received positive feedback specifically relating to this approach; “A good example is constructing and evaluating code snippets *outside* of the loop and then passing them in”. Deconstructing code in this way is also useful for debugging errors.

6. Avoid copying and pasting pre-written code

We have always programmed live while teaching, as this demonstrates how R code is written in practice. Levy (2016) explains that this style of teaching is more effective because it forces the teacher to slow down to a more suitable pace and ensures they are flexible and responsive to any questions the attendees may have. The difficulty in coding live is that the teacher has an added responsibility to stick to the general principles of programming and set a good example. Guidelines

are available (Bengtsson 2009), but have not been formally agreed. R users generally advocate adding comments where the code is not self-explanatory and using consistent indentation (Mächler 2014).

7. Errors are an integral part of the learning process

A consequence of live coding during demonstrations is that accidental errors are likely to occur - counterintuitively, this can be a positive thing. Producing errors are an effective way of learning, but attendees often have a fear them and observing errors during taught demonstrations can help tackle this fear. Mistakes produced during demonstrations can also be intentional. For example, we try to calculate the mean of a string variable to produce the warning message "argument is not numeric or logical: returning NA". Errors promote a more active learning environment as the class can be encouraged to participate in debugging. Similarly, Mascaró (2014) stated "students realised that mistakes were not a problem in R, because feedback of the environment tells them if there is a mistake or they get some illogical answer; this promotes reflection on what went wrong".

8. Teach attendees to teach themselves

The course is designed to encourage independent learning and this can be partly achieved by incorporating errors and teaching debugging skills (as mentioned above). It is also important for attendees to be aware of other material and know how to access the help files that are available for all functions and packages. However, Eglen (2009) highlights a common problem with these help files, stating "students often report that it is hard to discover these functions, as they do not know what to search for". Eglen (2009) advocates the use of *Rseek* (http://www.rseek.org), an internet search tool, for finding relevant functions. We generally find online R forums helpful.

Additionally, to help attendees teach themselves, we demonstrate how code can be developed when consulting a help file is required. This demonstration has the following steps: (1) We explain what we would like to achieve, (2) how to find the help file that we need, (3) how to make sense of the help file and (4) how to incorporate the relevant information in our code.

9. Relate R to other software packages

Learning involves integrating new knowledge with existing knowledge (Lovett & Greenhouse 2000). Therefore, attendees can learn more effectively by relating R to other software packages, explaining their similarities and highlighting their differences (Peruvankal & Muenchen 2014). Many attendees will have experience with at least one other statistical software package. For example, we explain that the R function *cat* works in a similar way to the *concatenate* function in Microsoft Excel. Attendees should also be made aware of the quirks in the R programming language. For example, <− is used to assign a name to an object, where other programming languages would use = (Bååth 2010).

CONCLUSION

In this paper, we described an intermediate-level course in R programming aimed at non-statisticians. The course is one of the more popular courses in the Centre for Applied Statistics Courses, suggesting that many research students and professionals require knowledge of statistical computer programming for their analysis, even those without a strong background in statistics. Its popularity also suggests that there is a big market for face-to-face R courses, despite there being a wide range of free material available online to support independent learning. We aim to address the initial fear of R programming in our course, which is more difficult to achieve through online courses without a teachers support. We have provided guidance for teaching or developing similar courses, derived from our own experience and combined with the experiences of other teachers of similar courses. We included guidance that is most applicable to intermediate R programming; guidance for teaching programming in general (Robins et al. 2003) and statistics (Tishkovskaya & Lancaster 2012) can be sought elsewhere.

Providing a successful training course in R is ultimately about understanding the mind-set of the attendees and anticipating the difficulties with programming that they may have. A post-graduate student, after completing a module in R stated, "I found that teachers can often over-complicate things in an attempt to project their knowledge-base to students. When returning to my notes as an intermediate R user, I realised how important simplicity is when absorbing information. The code could often be simplified." Teachers in a university setting may demonstrate their knowledge of a given subject whilst concentrating less on the intended learning outcomes for those attending the

course. The guidance in this paper can help promote simplicity and relevance of short statistical programming courses aimed at non-statisticians and provide motivation for their development.

REFERENCES

Bååth, R. (2010). How Should Programming R be Taught in an Introductory Course in Statistics? Introduction to pedagogy at college level, fall semester 2010, Lund University. Retrieved 15/7/16 from www.sumsar.net/papers/rasmus_baath_teaching_r_programming_01122010.pdf

Bengtsson, H. (2009). R Coding Conventions (RCC) – a draft. Retrieved from www.scribd.com/document/266557294/R-Coding-Conventions-RCC-A-Draft [accessed 19th September 2016]

Coleman, Y. (2007). Tips for Teaching Software – the approach [Web blog post]. Retrieved from metacole.com/2007/12/30/tips-for-teaching-software-the-approach/ [accessed 14th July 2016]

Eglen, S.J. (2009). A quick guide to teaching R programming to computational biology students. *PLoS computational biology*; 5(8), e1000482.

Ellis, Y., Daniels, B., & Jauregui, A. (2010). The effect of multitasking on the grade performance of business students. *Research in Higher Education Journal*, 8, 1.

Levy, M.A. (2016). Teaching R to 200 people in a week. Presented at the International R user conference, Stanford, CA. Retrieved from channel9.msdn.com/Events/useR-international-R-User-conference/useR2016/Teaching-R-to-200-people-in-a-week [accessed 14th July 2016]

Lovett, M.C. & Greenhouse, J.B. (2000). Applying cognitive theory to statistics instruction. *The American Statistician*, 54(3), 196-206.

Mächler, M. (2014). Good Practices in R Programming. Presented at the International R user conference, Vienna, Austria. Retrieved from stat.ethz.ch/Teaching/maechler/R/ useR_2014/ [accessed 14th July 2016]

Marriott, J., Davies, N. & Gibson, L. (2009). Teaching, Learning and Assessing Statistical Problem Solving. *Journal of Statistics Education*, 17(1).

Mascaró, M., Sacristán, A.I. & Rufino, M. (2014). Teaching and learning statistics and experimental analysis for environmental science students, through programming activities in R. In *Constructionism and Creativity-Proceedings 3rd Intl. Constructionism Conf* (pp. 407-416).

Ormrod, J.E. (2006). How Motivation Affects Learning and Behavior. Pearson Allyn Bacon Prentice Hall, www.education.com/reference/article/motivation-affects-learning-behavior.

Peruvankal, J.P. & Muenchen, B. (2014). Secrets of Teaching R [Web blog post]. Retrieved from blog.revolutionanalytics.com/2014/03/secrets-of-teaching-r.html [accessed 14th July 2016]

R Core Team (2016). R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing. www.R-project.org

Reed, S.K., Dempster, A., Ettinger, M. (1985). Usefulness of Analogous Solutions for Solving Algebra Word Problems. *Journal of Experimental Psychology*, 11, 106–125.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education*, 13(2), 137-172.

Tishkovskaya, S., & Lancaster, G. A. (2012). Statistical education in the 21st century: a review of challenges, teaching innovations and strategies for reform. *Jnl of Statistics Education*, 20(2), 1-55.

Westbrooke I., Ellis P. (2014). Training to develop modern statistics in the workplace using R and R commander - experiences from the New Zealand government sector. *Proceedings of the Ninth International Conference on Teaching Statistics (ICOTS9, July, 2014).*

Westbrooke I., Rohan M. (2012). Statistical training in the Workplace. In Topics from Australian Conferences on Teaching Statistics 2014 (pp. 311-327). Springer New York.