How Should Programming R be Taught in an Introductory Course in Statistics?

Rasmus Bååth
Lund University Cognitive Science
rasmus.baath@lucs.lu.se

Introduction

A sound knowledge of statistics is necessary in many academic fields. Even though it is possible to calculate statistics using pen and paper most will probably resort to using statistics software. It is possible to use standard spreadsheet programs (e.g. Excel) for basic statistical calculations but for more advanced statistics one will have to use dedicated software such as Stata\(^1\) or SPSS\(^2\). One piece of software that has gained popularity in the last couple of years is R\(^3\) (Muenchen, 2010). R differs in two respects from most other statistics software: 1. It is open-source and then also free of charge, this being one of the main reasons for its popularity. 2. Interaction with R is done by writing programs in the R programming language, not through a point-and-click interface as in e.g. SPSS. This last point poses a problem when teaching an introductory course in statistics using R as both introductory statistics and introductory programming has to be taught. The focus in such a course should of course be the statistics but without some programming knowledge it is not possible to use R in a meaningful way.

The reason for the topic of this paper is a course given at Lund University in 2009 with the same structure as outlined above, that is, an introductory statistics course using R. Many students did not have a background in programming and in the course evaluation there was critique of the choice of R as the statistics software. This year the course will be given again using R but three optional, two hour lessons covering R programming will be offered. It is of course impossible to cover programming in three lectures so this paper is an attempt to determine what parts of R programming should be taught in this course and what teaching method to use.

First an attempt to characterize the R language (henceforth referred to as R) will be made. R differs in many respects to popular languages such as Java and C. What possibilities and constraints do these differences imply when teaching? Then follows an outline of what parts of R programming should be covered. The minimum requirement is to cover what is presented in the course book (Crawley, 2005), but inspiration will also be drawn from Eglen (2009) who reports lessons learned while teaching R to biology students. After that the teaching method will be outlined. The aim will be to involve the students and to find a method that encourages active learning (McKeachie & Svinicki, 2006). At last a proposed layout of the three lessons will be presented.

Properties of R

R is an non-traditional language in many respects and it should impact how it is taught. R shares many of the properties of the Python programming language which have been pointed out as characteristic for a language suitable for teaching programming (Grandell et al. 2006). Below is given a list of properties that could aid teaching R and features that one should be aware of as they could make

---

3. http://www.r-project.org/
teaching R more difficult (many of these are taken from Grandell et al.).

**Properties of R that Could Aid Teaching**

- **Small and clean syntax**
  Compared to languages such as Java or C++, R has a concise and clean syntax. There is for example no need end lines with “;” and much of the formal bookkeeping clutter of for example Java is not present (see fig. 1). This enables students to start trying to program quickly without the need to spend a lot of time learning syntax details.

<table>
<thead>
<tr>
<th>Java</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>public class MyProgram {</td>
<td>xs = 1:100</td>
</tr>
<tr>
<td>public static void main(String[] args) {</td>
<td></td>
</tr>
<tr>
<td>xs = new int[100];</td>
<td></td>
</tr>
<tr>
<td>for(int i = 0; i &lt; 100; i++) {</td>
<td></td>
</tr>
<tr>
<td>xs[i] = i + 1;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1. Comparison of creation of an array containing the numbers 1 to 100 in Java and R.

- **Dynamic typing**
  R is dynamically typed, this further reduces the notation. It also makes it possible to postpone introducing students to the type system of R, which of course is important to learn eventually.

- **Expressive semantics**
  R has many powerful types built in, especially useful for statistical analysis: Because of this lists, arrays and data frames (an table structure) can be introduced at an early stage.

- **Interpreted**
  As R uses an interpreter (as opposed to a compiler). This means that lines of code can be run one at a time and this gives immediate feedback. This short feedback cycle makes it possible to learn by trial and error to a much larger degree.

- **Interactive prompt**
  R is designed to be interacted with through an interactive prompt. Again this allows student to start using R without sufficient knowledge and to *explore* the results of their code

- **Open-source**
  R is open source which means that it is freely available and that is has been ported to all modern operating systems. This enables students to install R on any computer and work from home.

**Properties of R that can Make Teaching Difficult**

- **Dynamic typing**
  As mentioned above this property lessens the “administrative overhead” but it also makes it more easy to write sloppy and hard to read code.
• **Unusual syntax**
  R syntax have some quirks that needs to be pointed out for people with some programming experience. E.g. “<-” is used for assignment instead of the usual “=” and “.” is an allowed character in variable names and is not used to access members of a object as in object oriented languages such as Java.

**What Parts of R Should be Covered?**

The main aim of the course is that the students should learn statistics, not programming. Therefore the parts of R programming the students will be taught should be the minimum required to handle the course. What should guide what parts of R programming to be covered is foremost the course book: “Statistics: an introduction using R”, Crawley (2005). The course book has a very short chapter on programming (23 pages) and it can be argued that it does not cover the material in much detail. Never the less the chapter indicates what parts of R programming that is fundamental:

• **The R environment**
  How to setup the basic environment, configure paths and loading and saving of sessions.

• **The interactive prompt**
  The prompt is the principal way of interacting with R and students should know e.g. about auto completion, searching the interactive help and how to stop malfunctioning scripts.

• **Basic arithmetics**
  Learning to use the usual operators such as “+” and “*” but also functions such as $\log(x)$.

• **Variable assignment**
  Introduction of variable names and the assignment operator “<-”. Here it is important to stress that assignment goes from the right to the left, that is, $5 \leftarrow x$ is not legal.

• **Basic data types**
  Creating and using the basic data types: Vectors, strings and data frames.

• **Vector manipulation**
  R have very advanced subscripting facilities and correct use of them is not straight forward. An example is the use of “-” in subscripts or using booleans indices to exclude values.

• **Input and output**
  The basic output and input functions of R should be covered so that the students can import from and save data to the file system.

• **Writing of functions**
  The students should be able to write simple functions and to know the merits of breaking up their program into meaningful parts.

• **Conditional Statements and Loops**
  These two topics are not really necessary to do basic statistics in R but they are the building blocks of programming and should at least be touched upon. Also they occur in the course book without much explanation and the students will be more prepared if they have at least
Eglen (2009) reports on lessons learned from teaching R to biology students. He highlights some common problems encountered when learning R. To avoid these problems some other issues that will be addressed are:

- **Common errors and error messages**
  The students should not only be shown the correct way to program but they should also be shown what happens when something goes wrong and what to do about it.

- **Type system**
  It is not necessary nor required that the students understand the intricacies of the R type system. But much confusion could be saved by pointing out that there exists a type system and to show the relation between the most common types (e.g. a matrix is a special case of a vector).

- **How to find help**
  An important skill to learn, only briefly covered by Crawley (2005), is how to use the many sources of help available to the beginning programmer. R has an extensive built in help system, that unfortunately uses a language that is quite technical. There are also excellent resources on the web such as Quick-R⁴.

This is a large list, but not all subjects needs to be covered in detail. The important that the students have heard about the concept. If they have at least some knowledge about a concept they should be able to use the help facilities of R to look it up further.

It has been argued that correct mental models are important when learning to program (Mayer, 1981). One mental model that will be introduced to the students is the input-output+side effects model of functions shown in figure 1. If variables are the nouns of programming functions are the verbs. When calling a function, say `sort(x)`, it is important to know what and that something goes in, comes out, and that there could be side effects. In the case of `sort(x)` it is easy to confuse the output with a side effect. Calling `sort(x)` on its own does not change anything since the sorting is not a side effect but a property of the output.

---

![Figure 1: A function is a process that always take some data as input and outputs a result. There might also be side effects such as changes to the input variables. Here it is important to be clear about that output and side effects are not the same thing.](http://www.statmethods.net/)
A proposal for what concepts should be introduced during the lessons is:

- **Lesson 1:** The R environment, the interactive prompt, basic arithmetics, variable assignment and basic data types.
- **Lesson 2:** Vector Manipulation, input and output and the input-output + side effects model of functions.
- **Lesson 3:** Writing of functions, the type system, conditional statements and loops.

Common errors, error messages and how to find help will be part of every lesson.

**Method of Teaching**

As pointed out before R share many of the properties of Python and the proposed teaching approach of Grandell et al. (2006) could also be used when teaching R. Grandell et al. are inspired by active learning theory that claims that students learning is facilitated if they become active participants in their own learning process (Bonwell, Eison, & Education, 1991). This leads them to take a “learning by doing”-approach and their approach is closely followed here.

Each lesson will be divided in two parts, one lecture part and one lab part.

**Lecture part**

In the lecture part the teacher will introduces the relevant concepts. This will done in a problem oriented way. Instead of introducing a concept and then showing examples of how it is used, a problem will be posed and a solution will be developed as the same time as the new concept is introduced. If for loops are to be introduced, the teacher could pose the problem: “How to make a plot for each of these columns?”. The teacher develops a solution not using for loops, showing how this leads to code duplication. The for loops are introduced as the solution to this problem and the program is modified to use for loops. The lecture will be held showing a projection of the teacher's R prompt to the students. In that way the students can see the development of solutions using the same tool as they will later use in the lab part.

When teaching programming there might be a bias towards teaching the right way to do things. A problem faced even by experienced programmers is that when programming much time is spent identifying and fixing problems you have created yourself (i.e. bugs and syntax errors). Therefore it is also important to expose students to common mistakes and show them how to reason about and fix them. During the lecture the teacher should also use every opportunity to show how to use the help system and other sources of documentation. These can be used, e.g., when showing new concepts (instead of having prepared power point slides).

**Lab part**

As R is interpreted, dynamically typed and interacted with through a interactive prompt it is possible for students to start experiment with R almost immediately. After the lecture part the students will be give handouts containing exercises to be solved. These exercises will not be “recipes” to follow, rather they will be open ended and problem based. The exercises should require the use of the concepts learned in the lecture. By being forced to use them the students will reinforce what was learned during the lecture and what was not learned will become obvious. By suggestion from McKeachie & Svinicki (2006) the students will be encouraged to work in pairs as peer-learning often is a very effective way of learning.
Students might have different levels of experience both in programming and using computers in general. To assess whether the lessons are at the right level, or if they are perceived as too advanced or too basic, minute papers, as described by Angelo & Cross (1993), will be collected at the end of each lesson.

References


introductory statistics course will depend on how the integration is carried out. If students perceive R as something extra, perceptions of R after engaging with programming in an introductory statistics course. Importantly, subgroups of students who initially felt the most negative about learning R (e.g. students who listed R as a concern, students without prior experience) and students traditionally. I am looking for a statistical software package which I can use in an introductory course of statistics for a social science study programme. The students have no prior knowledge of statistics and no experience with programming languages either. The goal is to introduce them to basic statistical concepts (as means, variance, sum of squares, p-values, and finally linear regression) and to enable them to conduct basic analyses on their own using example datasets. The course should be about learning concepts by doing statistics rather than memorizing formulas (although I think formulas are im Statistical programming is harder to define. One definition might be that it’s the kind of computer programming statisticians do but statisticians do all sorts of programming. Another would be that it’s the kind of programming one does when one is doing statistics but again, statistics involves a wide variety of computing tasks. Statistical programming is closely related to other forms of numerical programming. It involves optimization and approximation of mathematical functions. In our introduction to programming, we will show how to control the ow of execution of a program. After trying an online programming course, I was so inspired that I enrolled in one of the best computer science programs in Canada. Two weeks later, I realized that I could learn everything I needed through edX, Coursera, and Udacity instead. So I dropped out. Each course must fit four criteria: It must be an introductory course with little to no statistics or probability experience required. It must be on-demand or offered every few months. It must be of decent length: at least ten hours in total for estimated completion. I recommend this course to anyone interested in statistical analysis (as an introduction to machine learning, big data, data science, etc.). On a scale from 1 to 10, I give 50! Please note each course’s description and syllabus are accessible via the links provided above. In this course, you will continue your introduction to R programming and learn how R works with numeric vectors, etc. Read more. The course, Introduction to R Programming Part Two, taught by Joris Meys was EXCELLENT! All of the course materials were extremely helpful in allowing me not only to understand the details of R programming but also to gain a solid perspective on the fundamentals of R coding. The feedback on the assignments was very detailed and the explanations included why certain approaches of coding were preferable to others. Educators who have made important contributions to the field of statistics or online education in statistics. The majority of our instructors have more than five years of teaching experience online at the Institute.