R : Past and Future History

A Draft of a Paper for Interface '98

Ross Ihaka
Statistics Department
The University of Auckland
Auckland, New Zealand

Abstract

R began as an experiment in trying to use the methods of Lisp implementors to build a small testbed which could be used to trial some ideas on how a statistical environment might be built. Early on, the decision was made to use an S-like syntax. Once that decision was made, the move toward being more and more like S has been irresistible.

R has now outgrown its origins and its development is now a collaborative effort undertaken using the Internet to exchange ideas and distribute the results. The focus is now on how the initial experiment can be turned into a viable piece of free software.

This paper reviews the past and present status of R and takes a brief look at where future development might lead.

1 Genesis

A long time ago I discovered a wonderful book by Hal Abelson and Gerald Sussman called *The Structure and Interpretation of Computer Programs.* The book aims to introduce engineering students to computing using the *Scheme* programming language. It presents a wonderful view of programming; investigating a wide variety of interesting and practical examples, and even showing how a language like Scheme can be implemented.

At about the same I obtained access to one of the first releases of Rick Becker and John Chambers' New S language. I remember noticing both similarities and differences between S and Scheme. In particular, I remember that one day I wanted to show Alan Zaslavsky how you could use lexical scope to obtain *own variables.* I didn't have a copy of Scheme handy, so I tried to show him using S. My demonstration failed because of the differences in the scoping rules of S and Scheme. It left me thinking that there were useful additions which could be made to S.

Rather later, Robert Gentleman and I became colleagues at The University of Auckland. We both had an interest in statistical computing and saw a common need for a better software environment in our Macintosh teaching laboratory. We saw no suitable commercial environment and we began to experiment to see what might be involved in developing one ourselves.

It seemed most natural to start our investigation by working with a small Scheme-like interpreter. Because it was clear that we would probably need to make substantial internal changes to the interpreter we decided to write our own, rather than adopt one the many free Scheme interpreters available at the time. This is not quite as daunting a task as it might seem. The process is well mapped out in books such as that of Abelson-Sussman and that of Kamin. Having access to the source code of a number Scheme interpreters also helped with some of the concrete implementation details.

Our initial interpreter consisted of about 1000 lines of C code and provided a good deal of the language functionality found in the present version of R. To make the interpreter useful, we had to add data structures to support statistical work and to choose a user interface. We wanted a command driven interface and, since we were both very familiar with S, it seemed natural to use an S-like syntax.

This decision, more than anything else, has driven the direction that R development has taken. As noted above, there are quite strong similarities between Scheme and S, and the adoption of the S syntax for our interpreter produced something which "felt" remarkably close to S. Having taking this first step we found ourselves adopting more and more features from S.

Despite the similarity between R and S, there remain number of key differences. The two fundamental differ-

```
> total <- 10
> make.counter <-
+   function(total = 0)
+     function() {
+       total <<- total + 1
+      total
+     }
> counter <- make.counter()
> counter()
[1] 1
> counter()
[1] 2
> counter()
[1] 3
```

Figure 1: *A simple function demonstrating how the scoping rules in R differ from those of S.*

ences result from R's Scheme heritage.

- **Memory Management**: In R, we allocate a fixed amount of memory at startup and manage it with an on-the-fly garbage collector. This means that there is very little heap growth and as a result there are fewer paging problems than are seen in S.

- **Scoping**: In S, variables in functions are either local or global. In R we allow functions to access to the variables which were in effect when the function was defined; an idea which dates back to Algol 60 and found in Scheme and other *lexically scoped* languages. Consider the function definition in figure 1. The function `make.counter` returns a value which is itself a function. This "inner function" increments the value of the variable `total`, and then returns the value of that variable. In S, the variable being manipulated is global. In R, it is the one which is in effect when the function is defined; i.e. it is the argument to `make.counter`. The effect is to create a variable which only the inner function can see and manipulate.

  Generally, the scoping rules used in R have met with approval because they promote a very clean programming style. We have retained them despite the fact that they complicate the implementation of the interpreter.

The two differences noted above are of a very basic nature. In addition, we have experimented with a number of other features in R. A good deal of the experimentation has been with the graphics system (which is quite similar to that of S). Here is a brief summary of some of these experiments.

- **Colour Model**: R uses a device independent 24-bit model for colour graphics. Colours can be specified in a number of ways.

  1. By specifying the levels of red, green and blue primaries which make up the Colour. For example, the string `"#FFFF00"` indicates full intensity for red and green with no blue; producing yellow.
  2. By giving a colour name. R uses the colour naming system of the X Window System to provide about 650 standard colour names, ranging from the plain `"red"`, `"green"` and `"blue"` to the more exotic `"light goldenrod"`, and `"medium orchid 4"`.
  3. As a index into a user settable colour table. This provides compatibility with the S graphics system.

- **Line Texture Description**: Line textures can also be specified in a flexible fashion. The specification can be:

  1. A texture name (e.g. `"dotted"`).
  2. A string containing the lengths for the pen up/down segments which compose a line. For example, the specification `"52"` indicates 5 points (or pixels) with "pen down" followed by 2 with "pen up", with the pattern replicated for the length of the line.
  3. An index into a fixed set of line types, again providing compatibility with S.

- **Mathematical Annotation**: Paul Murrell and I have been working on a simple way of producing

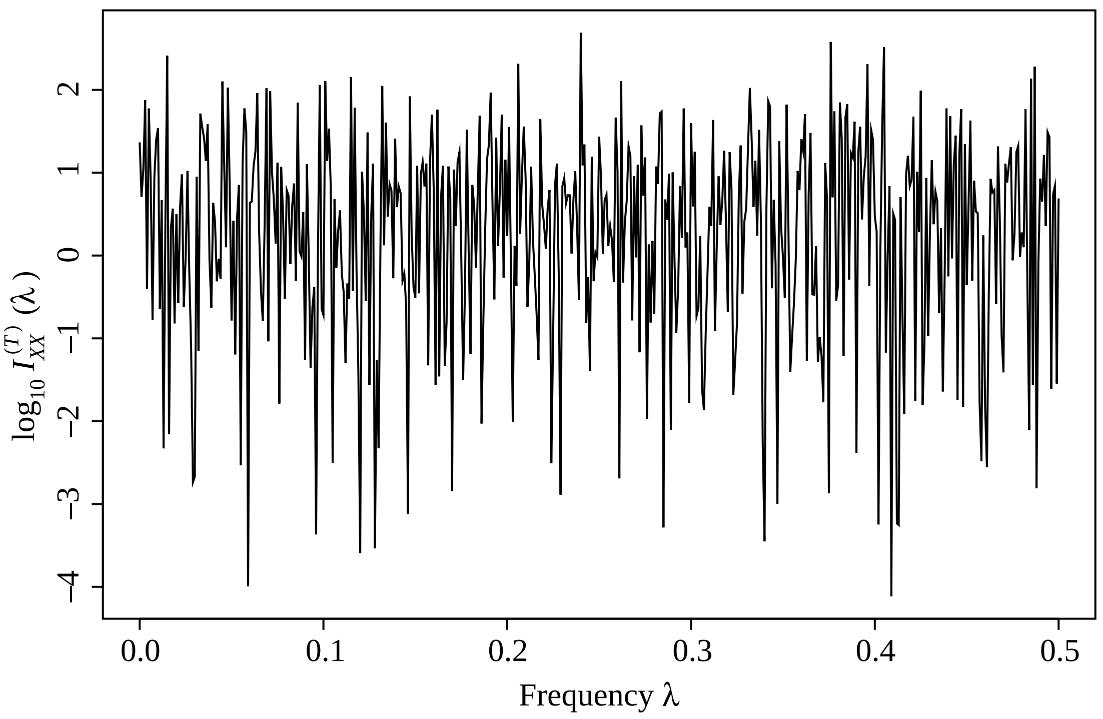

Figure 2: A plot of the periodogram of a white-noise time series, showing the use of mathematical annotation.

mathematical annotation in plots. Mathematical annotation is produced by specifying an unevaluated R expression instead of a character string. For example,

```
expression(x^2+1)
```

can be used to produce the mathematical expression

$$x^2 + 1$$

as annotation in a plot.

The annotation system is relatively simple, and not designed to have the full capabilities of a system such as TEX. Even so, it can produce quite nice results. Figure 2 shows a simple example of a time series periodogram plot produced in R. The plot was produced with a single R command which used `expression` to describe the labels.

- **Flexible Plot Layouts**: A part of his PhD research, Paul Murrell has been looking at a scheme for specifying plot layouts. The scheme provides a simple way of specifying how the surface of the graphs device should be divided up into a number of rectangular plotting regions. The regions can be constrained in a variety of ways. Paul's original work was in Lisp, but he has implemented a useful subset to R.

These graphical experiments were carried out at Auckland, but others have also bound R to be an environment which can be used as a base for experimentation.

- **Compilation**: Luke Tierney has performed some experiments to see what kind of performance gains could be obtained by using byte-code compilation of R. His experiments indicated that a speed-up by a factor of 20 might be possible for some interpreted code. As yet, the internal data structures in R are probably not stable enough to make it worthwhile to follow up on this work.

- **WWW Interface**: Jeff Banfield has developed RWeb, which is a WWW based interface to R.

- **Tcl/Tk Interface**: Very recently Balasubramanian Narasimhan has begun looking into how Tcl/Tk might be used to add a fully graphical user interface to R.

2 A Free Software Project

2.1 A Brief History

The initial work on R by Robert Gentleman and I produced what looked like a potentially useful piece of software and we began preparing it for use in our teaching laboratory. We were heartened enough by our progress to place some binary copies of R at Statlib and make a small announcement on the *s–news* mailing list in August of 1993.

A number of people picked up our binaries and offered feedback. The most persistent of these was Martin Mächler of ETH Zurich, who encouraged us to release the R source code as “free software”.

We had some initial doubts about doing this, but Martin's arguments were persuasive, and we agreed to make the source code available by ftp under the terms of the Free Software Foundation's GNU general license. This happened in June of 1995.

At this point, the development of R was a relatively closed process. Robert and I (soon joined by Martin) would get bug reports by e-mail and from time-to-time release updated versions of R. We quickly noticed that there was no real forum for users to discuss R with each other and so we began maintaining a small mailing list.

As interest in R grew (mostly by word of mouth) it became clear that manually maintaining the mailing list was not an effective option. Worse than that, at Auckland we were paying for e-mail, and the cost was beginning to become noticeable. Eventually Martin volunteered the use of facilities at ETH Zurich to establish automated mailing lists to carry discussions about R and R development. In March of 1996 the *r–testers* mailing list was started. Roughly a year later this was replaced with three newsgroups: *r–announce*, *r–help* and *r–devel*.

As R developed and people began porting applications to it, it became clear that we needed a better distribution mechanism. After some discussion it was decided a formal archive mechanism was desirable. Kurt Hornik of TU Wien took on the task of establishing the archive. In addition to the master site in Austria there are a number of mirror sites, including StatLib.

With the introduction of the mailing lists, development on R accelerated. This was partly because we obtained many more reports and suggestions and partly because we also began to receive patches and code contributions. The contributions ranged from fixes for typos through to changes which provided substantial increases in functionality and performance.

The level of contribution was such that Robert, Martin and I couldn't always make changes at a rate which was satisfactory to those asking for changes. As a result, in mid-1997 we established a larger “core group” who can make changes to the source code CVS archive. This group currently consists of:

> Doug Bates, Peter Dalgaard,
> Robert Gentleman, Kurt Hornik,
> Ross Ihaka, Friedrich Leisch,
> Thomas Lumley, Martin Mächler,
> Paul Murrell, Heiner Schwarte,
> and Luke Tierney.

Since all work on R is strictly of a voluntary nature, the organisation is very loose, with members contributing when and as they can.

2.2 Contributors

When Robert and I started work on R, we were hopeful that we might be able to produce something we could use to teach our introductory data analysis courses. Had we continued to work strictly on own it is likely that this is precisely what we would have achieved.

The decision to make R free software has enabled us to set rather higher goals, because it has given us access to a large pool of very talented individuals who have been willing to invest significant effort in the project. Indeed, one of the very best things about having worked on R has been the chance to work with such a great group of people.

In addition to the core group listed above, I would like to acknowledge the following individuals who have made significant contributions to R.

> Valerio Aimale, Ben Bolker,
> John Chambers, Simon Davies,
> Paul Gilbert, Arne Kovac,
> Philippe Lambert, Alan Lee,
> Jim Lindsey, Patrick Lindsey,
> Mike Meyer, Martyn Plummer,
> Anthony Rossini, Bill Venables,
> Gregory Warnes, and
> `mward@wolf.hip.berkeley.edu`

(I apologise for omissions here. Our record keeping has not been all that it could be).

In addition a host of other individuals have made contributions.

2.3 Present Status

R is still under active development and there is still some work needed before it can be considered ready for widespread use. In particular, some changes will be required to support moderate to large-sized data sets.

More importantly, there is an almost complete lack of introductory documentation, although much of what has been written about S directly applicable to R.

Despite this, it seems that R is beginning to reach the point where it is stable enough for regular use (at least under Unix). I am hopeful that during the next year we can release a complete R version 1.0 package as part of the Free Software Foundation's GNU suite of software.

3 The Future

3.1 R

It is the present aim of the R project to produce a free implementation of something "close to" version 3 of the S language and to provide ongoing support and maintenance for the resulting software. Some members of the R core have proposed that future developments in S version 4 should be also tracked. At this point it is unclear whether this will happen.

One development which would help R a good deal would be the development of an integrated graphical user interface. Some initial work has begun on this and I believe that it is something which will come quite quickly.

My personal future interest in R is mainly as a user. Given the investment I have made in it, I hope that I will be able to get substantial use out of R for statistical work and teaching.

3.2 Related Work

Working on R has shown me that there a number of interesting questions related to building statistical software. My own conclusion has been that it is important to pursue efficiency issues, and in particular, speed.

As noted in section 1, Luke Tierney performed some experiments with R to see what kind of speed increase could be obtained using byte-code compilation; the indications were that a speedup by a factor of 20 might be possible for some computations.

There is other evidence that a factor of 100 (roughly the speed of unoptimised C) might be possible with compilation to native machine code. With this level of performance, there would be no need for any foreign function interface and *all* computations could take place in a single language environment.

I am intrigued by what such an environment might offer. An increase in performance of this magnitude is likely to produce a qualitative change in the use it gets puts to.

The difficulty is that the creation of such a compiled environment requires the hand of an expert in compilation. There is a real problem in finding such an expert who is also aware of the type of problems which statisticians handle.

4 Acknowledgements

It goes without saying that R would not exist without the pioneering work of John Chambers and his AT&T collaborators. John has changed the way that many of us think about statistical computing and the fact that R has evolved to resemble S as closely as it does is is a testimony to the extent that people enjoy doing data analysis with S.

The Free Software Movement (or movements perhaps) has been another major source or ideas and influence on R. I do my development work on a workstation which runs the FreeBSD operating system and is equipped with a rich set of development tools from the Free Software Foundation. Hopefully, R represents some pay back to the free software community for what they have provided to me and the other R developers.

Finally, I'd like to acknowledge my partner-in-crime; Robert Gentleman. During our work on R we have practically lived in one another's back pockets. It speaks volumes that, not only are we still on speaking terms, but we still go out for beer together on Friday evenings.

References

Abelson, H. and G. J. Sussman, with J. Sussman (1985). *Structure and Interpretation of Computer Programs.* Cambridge MA: MIT Press.

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1986). *The new S Language: A programming environment for data analysis and graphics.* Pacific Grove, CA: Wadsworth & Brooks/Cole.

Chambers, J. M. and T. J. Hastie, Eds. (1991). *Statistical Models in S.* Pacific Grove, CA: Wadsworth & Brooks/Cole.

R. Gentleman and R. Ihaka (1997). "The R language", In *Proceedings of the 28th Symposium on the Interface*, L. Billard and N. Fisher Eds. The Interface Foundation of North America.

Ihaka, R. and R. Gentleman (1996). "R: A language for data analysis and graphics," *Journal of Computational and Graphical Statistics*, **5**, 299–314.

Kamin, S. N. (1990). *Programming languages.* Addison Wesley.