The R Project: A Brief History and Thoughts About the Future

Ross Ihaka

The University of Auckland

The R Language and Environment

- R is a computer language and run-time environment which can be used to carry out statistical (or other quantitative) computations.
- The base part of R comes with a wide range of standard statistical and graphical analyses built in.
  - parametric statistical modelling
  - multivariate analysis
  - nonparametric statistics
  - smoothing and nonparametric fitting
  - time series analysis
- There are a large number of user-developed extension *packages* which provide an even richer set of capabilities.

Licensing

- R is free software released under the Free Software Foundation's General Public License.
- This means that R is free of any restrictions on how it can be disseminated.
- Versions of R can be obtained without charge and can be redistributed to others.
- The license is intended to prevent the creation of encumbered derived works (i.e. commercial versions).

Uptake

- Because of its license, it is very hard to determine what the installed user base of R might be.
- The R development group has considered how to estimate the number of R users, but considers the problem to be intractable.
- A 2009 article in the New York Times presented the two estimates: one million users (Intel Capital) and two million users (Revolution Analytics).

The R Language

- R is an expression-based language.
  - Users type language *expressions* at the R prompt.
  - These expressions are *evaluated* by the R *interpreter*..
  - The computed values of the expressions are printed.
- R is extensible.
  - Users can implement new functionality in the form of *functions*.
  - Developers can implement new *packages* of functionality that extends the base system.

An Example

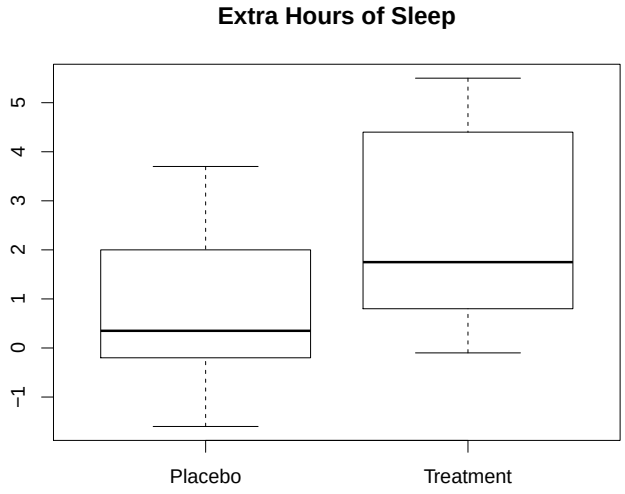
The "`sleep`" data in R contains data that W. S. Gosset ("Student") used for one of the first t-tests.

The data set is stored as a *data frame*. We can find out what variables it contains as follows:

```
> objects(sleep)
[1] "ID"    "extra" "group"
```

We can examine the data with a box-and-whisker plot.

```
> boxplot(extra ~ group, data = sleep,
          main = "Extra Hours of Sleep",
          names = c("Placebo", "Treatment"))
```


Extra Hours of Sleep
−1
0
1
2
3
4
5
Placebo
Treatment

Example Continued . . .

Each subject has a pair of placebo/treatment observations, so it is appropriate to carry out a paired t-test.

We are interested in whether the treatment provides more additional sleep than the placebo.

```
> with(sleep,
       local({
            placebo = extra[group == 1]
            treatment = extra[group == 2]
            t.test(treatment, placebo,
                   alternative = "greater",
                   paired = TRUE)
       }))
```

Example Continued . . .

The t-test produces the following output.

```
        Paired t-test

data:  treatment and placebo
t = 4.0621, df = 9, p-value = 0.001416
alternative hypothesis: true difference in means
         greater than 0
95 percent confidence interval:
 0.8669947       Inf
sample estimates:
mean of the differences
                  1.58
```

Early History - 1990

- Ross Ihaka joins the Department of Statistics at the University of Auckland.
- Robert Gentleman spends 1990 in Auckland on sabbatical from the University of Waterloo.
- During a chance encounter in the corridor, the following exchange takes place:

  Gentleman: "Let's write some software."
  Ihaka: "Sure, that sounds like fun."

- The initial goal is to build a testbed for trying out ideas and to publish a paper or two.

The Initial Language

```
> (set x (seq 10))
(1 2 3 4 5 6 7 8 9 10)

> (sum x)
55

> (set factorial (lambda (x)
  (if (< x 1)
      1
      (* x (factorial (- x 1))))))
<closure>

> (factorial 5)
120
```

Early History - 1992

- Robert Gentleman joins the Department of Statistics at Auckland.
- We set a goal of developing enough of a language to teach introductory statistics courses at Auckland.
  - It is decided to adopt the syntax of the S language developed at Bell Laboratories.
  - As a joke, the name “R” is coined for the language (standing for Robert and Ross).

Early History - 1994

- An initial version of the language is complete.
- Colleagues overseas encourage us to release the language as "free software."
- A little thought convinces us that there are limited prospects for the software as a commercial product.
- We adopt the Free Software Foundation GPL as our license and begin to make releases via the Internet.
- We start a small email list so that we and our users can discuss R.

The original R developers plotting world domination.

Early History - 1996

- By 1996 we were becoming victims of our own success.
- We were being supplied with a continuous stream of bug reports and suggestions for improvement.
- Maintaining the mailing list was becoming problematic.
- It was beginning to be clear that the project was getting close to the limit of what two of us could handle.

Early History - 1997

- The mailing list turned out to be very successful and our user base increased enormously (to nearly 100!).
- The list was so successful that was split into the present `r-help` and `r-devel` lists.
- Kurt Hornik and Fritz Leisch established the CRAN archive at the Technical University of Vienna as a repository for user contributions.
- We became so deluged with patches and requests for enhancements that we decided to open up the development process by giving a selected “core” of developers direct access to the CVS archive.

R Becomes A GNU Project

From: Richard Stallman <rms@gnu.ai.mit.edu>
To: ihaka@stat.auckland.ac.nz
cc: rms@gnu.ai.mit.edu
Subject: Re: Seen on your wishlist
Date: Tue, 16 Sep 1997 21:56:06 -0400

> So [explicitly], yes we would like R to be considered as a GNU program.

I hereby dub R GNU software!

A Free Software Project

- Since we opened up the project, it has gone ahead in leaps and bounds.
- On February 29, 2000, the software was deemed fully featured enough and stable enough for the 1.0 release to take place.
- There are now nearly 20 core developers maintaining and extending the language interpreter and its basic functionality.
- The group includes a number of well-known researchers in Statistical Computing.
- The software now has a regular six-monthly release cycle and will shortly see the release of version 2.14.

The intense software development effort leading up to R version 1.

R Core Developers

Peter Dalgaard	University of Copenhagen
John Chambers	Bell Labs and Stanford University
Robert Gentleman	Genentech
Kurt Hornik	University of Vienna
Stefano Iacus	University of Milan
Ross Ihaka	University of Auckland
Friedrich Leisch	University of Munich
Thomas Lumley	University of Auckland
Martin Mächler	ETH Zurich
Duncan Murdoch	University of Western Ontario
Paul Murrell	University of Auckland
Martyn Plummer	International Agency for Research on Cancer
Brian Ripley	Oxford University
Duncan Temple Lang	University of California
Luke Tierney	University of Iowa
Simon Urbanek	AT&T

Current Status

- The *R Project* is an international collaboration of researchers in statistical computing.
- The formal structure for the project is provided by the *R Foundation*, a non-profit foundation based in Vienna.
- The Foundation collects donations and uses them to maintain infrastructure and sponsor some development work.
- The software is stable and is undergoing only relatively minor enhancements.
- The software continues to be released under a “free software” license.

Current Status

- There are more than a hundred books which have been published (or are in preparation) dealing with R and its applications.
- Springer has a dedicated book series dedicated to R (UseR).
- The “R Newsletter” has matured and been relaunched as the “R Journal.”
- There are nearly 3000 extension *packages* which have been contributed to CRAN. (The number is growing exponentially).

R's Limitations

R is a useful piece of software, but it does have limitations.

- It discourages direct solution of problems and favours the use of some rather strange idioms.
- It can make huge demands on system resources, particularly memory.
- It is very slow, particularly for element-by-element computations.
- It uses a single thread of execution model which is very restrictive.

A Code Example

Given numeric `x` and `y` vectors, the following R fragment carries out a useful computation. What does it do?

```
> x[apply(outer(x, y, function(x, y) abs(y - x)),
          2, function(u) which(u == min(u))[1])]
```

A Code Example

Given numeric `x` and `y` vectors, the following R fragment carries out a useful computation. What does it do?

```
> x[apply(outer(x, y, function(x, y) abs(y - x)),
          2, function(u) which(u == min(u))[1])]
```

Answer: It computes the nearest x to each y.

```
> x = seq(0, 1, by = .1)
> y = runif(5)
> nx = x[apply(outer(x, y, function(x, y) abs(y - x)),
               2, function(u) which(u == min(u))[1])]
> rbind(y, nx)
       [,1]      [,2]     [,3]      [,4]      [,5]
y  0.9880671 0.5387734 0.184014 0.5022672 0.3539751
nx 1.0000000 0.5000000 0.200000 0.5000000 0.4000000
```

Alternative Coding

The computation can also be written in a more straightforward way.

```
> nx = numeric(length(y))
> for(j in 1:length(y)) {
      dmin = Inf
      imin = 0
      for(i in 1:length(x)) {
           d = abs(x[i] - y[j])
           if (d < dmin) {
                dmin = d
                imin = i
           }
      }
      nx[j] = x[imin]
  }
```

Comparison

The first code fragment is “cute,” but there are several problems with it.

- It is hard to tell what it does.
- It uses a potentially huge amount of machine memory and, hence, time.

The second fragment is transparent and, in theory, makes less demand on machine resources, but it also has a problem.

- It runs very slowly in R (element-by-element computations are slow).

Fixing the Problems

- There are two approaches to fixing the problems with R.
- The first approach is to try to make R run faster.
- There have been (at least) three projects which have sought to improve R's performance.
- At best, these approaches offer a 5–10 fold improvement in speed and little improvement in resource use.

A Second Approach

- Another way to gain performance is to make fundamental changes to the way the language works.
- This means defining a new (incompatible) language which avoids those features of R which cause problems.
- The major problems are that:
  - There is far too much unnecessary copying of data in R.
  - Scalar (i.e. element-by-element) computation is slow.
  - There are bizarre effects which result from the way R handles scoping.
- These issues are inherent in the way R works and eliminating them requires a new language.

Copying

- R semantics mean that functions are not permitted to change their arguments ("call by value").
- Essentially, functions work on copies of their arguments.
- Moving to a call-by-reference model will eliminate over-copying, but code will have to be changed to avoid destroying valuable data.

Scalar Performance

- R's slow element-by-element performance results from the fact that it has no scalars and every operation incurs array access and bound checking overhead.
- One way to speed things up is provide true scalar support.
- This makes the language much more complicated and will require the addition of (optional) type declarations to gain maximum performance.

Scoping Issues

- R variables are created by assignment.
- This *overloading* of declaration and assignment is rather peculiar.
- Variables may or may not be local and whether the variable even exists is not determined until runtime.
- It sort this problem out it will be necessary to introduce mandatory scope declarations.

A New Language

- Research is under way to provide the technology which can be used to build a better language for doing statistics (and other quantitative calculations).
- Present collaborators are Duncan Temple Lang, Brendan McArdle and myself.
- Initial experiments indicate that we can expect a three order-of-magnitude speedup in the kind of computations which are slow in R.
- By using the right algorithms it should be possible to reduce memory demands significantly.

An Example

A small `sum` function written in R.

```
sum =
    function(x) {
        s = 0
        for(i in 1:length(x))
            s = s + x[i]
        s
    }
```

An Example

A small `sum` function with scoping declarations in a hypothetical new language.

```
sum =
    function(x) {
        local i, s = 0
        for(i in 1:length(x))
            s = s + x[i]
        s
    }
```

An Example

A small `sum` function with scoping and type declarations in a hypothetical new language.

```
sum =
    function(double x[]) {
        local integer i, n = length(x)
        local double s = 0
        loop(i = 1; i <= n; i = i + 1)
            s = s + x[i]
        s
    }
```

Summary

- R provides a useful platform carrying out statisti.cal computations.
- Its particular strength is its extensibility.
  - New techniques can be added “on the fly.”
  - New methodology is easily packaged and distributed.
- R is limited in the size of problem it can attack.
- A new, but similar, language is under development which will make it possible to keep the advantages of R while making it possible to handle (much?) larger problems.