The Past, Present, and Future of the R Project

Kurt Hornik

Prehistory of R: S

v1	1976–1980	Honeywell GCOS, FORTRAN-based	
v2	1980–1988	Unix: macros, interface language	
	1981–1986	QPE (Quantitative programming environment)	
	1984–	general outside licensing, books	
v3	1988–1993	C-based, S functions & objects	Blue
	1991	Classes & methods, statistical models	White
	1992	Performance improvements	
v4	???	Programming with data	Green

Next book by John Chambers: “Software for Data Analysis: Programming with R”

1998 Association for Computing Machinery (ACM) Software System Award to John M. Chambers, the principal designer of S, for

> the S system, which has forever altered the way people analyze, visualize, and manipulate data . . .
>
> S is an elegant, widely accepted, and enduring software system, with conceptual integrity, thanks to the insight, taste, and effort of John Chambers.

R history: Early days

In 1992, Ross Ihaka and Robert Gentleman started the R project

> in trying to use the methods of LISP implementations to build a small test-bed which could be used to trial some ideas on how a statistical environment might be built.

S-like syntax motivated by both familiarity and similarity to Lisp:

```
parse(text = "sin(x + 2)") ⇔ (sin ("+" x 2))
```

R history: R Core

First binary copies of R on Statlib in 1993

First release of sources under GPL in 1995

Many people started sending bug reports and helpful suggestions

Formation of an “R Core Team” who collectively develop R, first official release by R Core: R 0.60 on 1997-12-05

Currently 19 members, including John Chambers

First R Core meeting at DSC 1999 in Vienna

R Core $\Rightarrow$ R Foundation

R milestones

R 1.0.0 released on 2000-02-29 (once every 400 years): R as a reference implementation of S3

R 2.0.0 released on 2004-10-04: R as a reference implementation of S4 (connections, methods) and substantially beyond (name spaces, grid/lattice, Sweave, . . .)

Workings towards R3: i18n/l10n ("'R learns to speak your language'"), graphics device drivers, 64 bits, object system and name spaces, performance enhancements, . . .

But where is R really going?

Wrong question.

Applying Theorem 7 of Wittgenstein’s Tractatus Logico-Philosophicus: no answer.

R base and extensions

R Core provides a “base system” only.

Even key statistical functionality available via contributed extensions (“packages”), some recognized as *recommended* and to be made available in every binary distribution of R

R distribution: R interpreter (plus tools) plus

- Base packages: base, stats, graphics, methods, . . .
- Recommended packages: MASS, boot, nlme, survival, . . .

Two-or-more-tier development model

R packages

R extensions come as "packages" containing

- meta-information, currently serialized as DESCRIPTION file in Debian Control File format (tag-value pairs)
- code and documentation for R
- foreign code to be compiled/dynloaded (C, C++, FORTRAN, . . .) or interpreted (Shell, Perl, Tcl, . . .)
- package-specific tests, data sets, demos, . . .

The key things to note are:

- Packages have meta-data, including license, version (so that we know when they're out of date), and internal and external dependencies
- Package can contain "anything"
- R knows a lot about taking advantage of what they contain (but there is always room for improvement)

Meta-packages, frameworks, compendia, . . .

The package system is one of the cornerstones of R's success.

So where is R going?

Multi-tier development model: “R” as in “base R” or the “R Multiverse”?

Base R developed by R Core: collection of individuals who work on stuff they need and/or like to work on.

Some trends and open issues: large data sets, object systems, event loops, byte compiler, . . .

Mostly unrewarding: “statisticians” do not get academic credit for working on such tasks.

The R Foundation could hire programmers, but has not (yet?)

There is no real project management.

But what do people really need to be added to or changed in base R?

The future of R

Really need to consider contributed packages

Packages come in repositories, with CRAN, Bioconductor and Omegahat the “standard” ones

In what follows, we focus on CRAN (the CRAN package repository)

Possibly substantial bias from focussing on standard repositories (ignores R stuff distributed in different ways)

What is the number of CRAN packages?

CRAN growth in the past

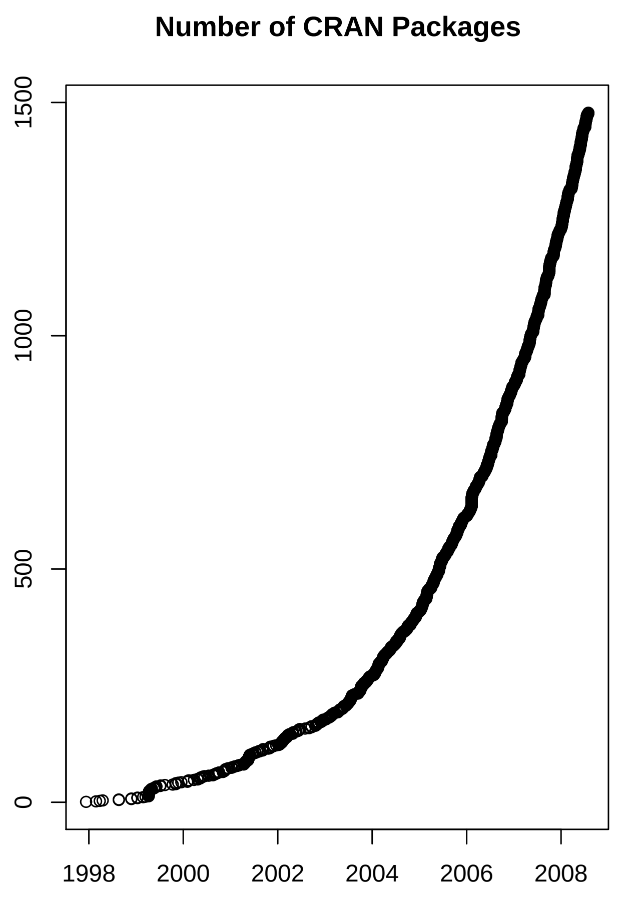
Look at the time series of packages available on CRAN.

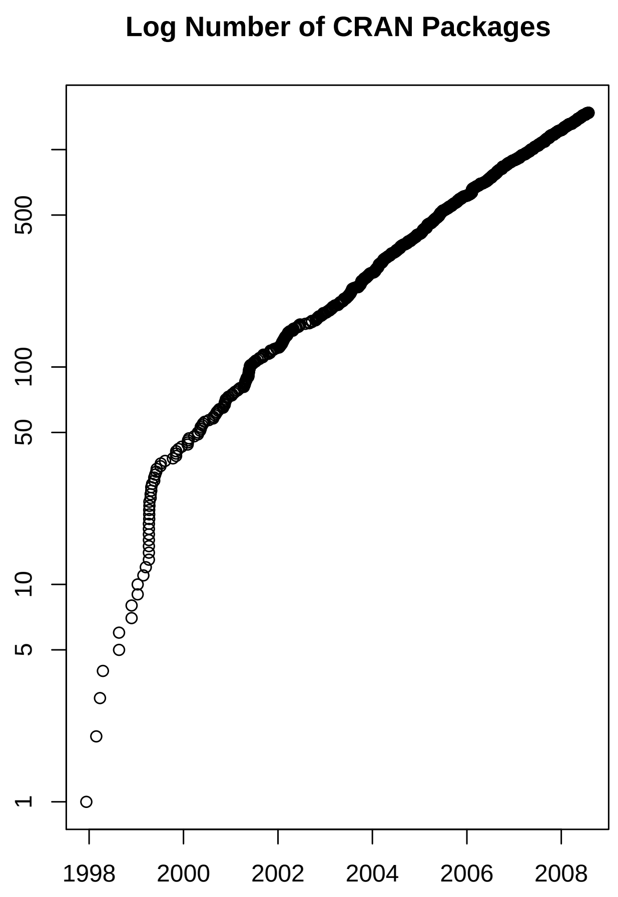
Not entirely straightforward because there is no real transaction logging. Proxies: older talks by myself, numbers of binary packages for older versions of R.

E.g., sometimes mid 2002 (article “Vienna and R: Love, Marriage and the Future”, talk at GRASS 2002): 165–170; mid 2004 (article “R: The next generation” for COMPSTAT 2004): more than 300, mid 2005 (article “R Version 2.1.0” for Computational Statistics): about 500, mid 2007 (talk at Rmetrics 2007): close to 1100.

Analysis based on the mtimes of package/bundle files on the CRAN master. May be imprecise for the early days; should be rather reliable since the advent of R News.

The Past, Present, and Future of the R Project





CRAN growth in the future

Always (almost) exponential?

Truth is: we have no idea what is driving this process.

Driven by developeRs who upload packages to CRAN. Who are they? Why do they do this? What are their expectations in doing this? How do they interact?

How does R “grow” across domains and regions?

Could have many more “infrastructure” packages reflecting collaborative development (does collaboration lead to fragmentation?)

Could also have packages wrapping data or ebooks.

What is the percentage of CRAN packages with a single author?

Poor person's SNA

Analyze how packages are interrelated using their "formal" dependencies (Depends/Imports/Suggests/Enhances) as declared in their DESCRIPTION files.

For a given dependency relation R (e.g., "p depends on q"):

The *forward* dependencies of p are all q for which pRq.

The *reverse* dependencies of p are all q for which qRp (inverse of R)

The *recursive reverse* dependencies of p are all q which eventually depend on p, i.e., for which there are $q_0 = q, \ldots, q_n = p$ with $q_{i-1} R q_i$ for $1 \le i \le n$ (transitive closure of the inverse of R).

"High priority" packages are base and recommended packages.

Distribution of forward dependencies

	Min	Q_1	Med	Q_3	Max	Mean
Depends	0	0	1	2	20	1.304
Imports	0	0	0	0	7	0.164
Suggests	0	0	0	0	33	0.629
Enhances	0	0	0	0	9	0.025

When excluding the forward dependencies with high priority:

	Min	Q_1	Med	Q_3	Max	Mean
Depends	0	0	0	1	20	0.684
Imports	0	0	0	0	4	0.039
Suggests	0	0	0	0	30	0.468
Enhances	0	0	0	0	9	0.023

Distribution of reverse dependencies for all CRAN packages:

	Priority	Min	Q_1	Med	Q_3	Max	Mean
Depends	high	0	9.5	21.5	43.25	153	36.310
Depends	low	0	0.0	0.0	0.00	41	0.666
Imports	high	0	0.0	2.5	8.75	43	7.308
Imports	low	0	0.0	0.0	0.00	6	0.038
Suggests	high	0	2.0	6.5	11.50	57	9.423
Suggests	low	0	0.0	0.0	0.00	23	0.446
Enhances	high	0	0.0	0.0	0.00	1	0.115
Enhances	low	0	0.0	0.0	0.00	1	0.023

Distribution of recursive reverse dependencies for all CRAN packages:

	Priority	Min	Q_1	Med	Q_3	Max	Mean
Depends	high	0	13.5	26.5	121.20	509	97.230
Depends	low	0	0.0	0.0	0.00	59	0.948
Imports	high	0	0.0	3.5	15.25	60	11.190
Imports	low	0	0.0	0.0	0.00	6	0.042
Suggests	high	0	4.0	17.0	67.75	186	50.730
Suggests	low	0	0.0	0.0	0.00	69	1.405
Enhances	high	0	0.0	0.0	0.00	1	0.115
Enhances	low	0	0.0	0.0	0.00	3	0.027

The social network underlying R package development

Currently, we observe rather little cooperation

R-Forge is a platform providing state of the art infrastructure facilitating collaboration between developeRs (and useRs)

We know very little about the social fabric of R developeRs and CRAN package authors and maintainers.

Research on this should be fascinating and rewarding

Something useRs could do!

The needle in the haystack

“CRAN has grown so big that I cannot find anything in it.”

“You really think we should look at all packages in alphabetical order?”

“Can’t you provide a good (full-text) search engine? A news ticker?”

“Which packages should I use?”

“Can’t you make recommendations?”

How big is CRAN?

Numbers and sizes of files in CRAN:

	Number	Size
Code	22588	104538340
Docs	37675	76839947
Data	6927	494720847
Vignettes	384	7966941
Sources	5749	97957418
Headers	2059	11275786
Sum	75382	793299279
Total	99384	2033966979
PDFs	1708	328553306

What is a repository?

Think of packages as “products” and potential useRs as “customers”.

CRAN and other package repositories are really warehouse-like storage areas from which products can be delivered.

How do customers get their products?

Modern views driven by understanding that customers are extremely heterogeneous with individual needs and preferences.

There is no “typical” useR.

Let there be a multitude of services . . .

Computer science is not information technology.

Services

Wikis, blogs, folksonomies, . . .

(Sub)Views (e.g., CRAN Task Views) and portals

Why not have portals where useRs can register, specify a profile, and then get recommendations, informations, . . . ? Could even be a commercial offering

Does one need “authorization” for offering such services?

Where CRAN could improve: we actually have very little information about its content in structured form — need “knowledge discovery” (data mining?)

Metadata

Package metadata in DESCRIPTION files in DCF (fields in tag: value format)

A few fields (used by the R package management system) are standardized and QCed.

Other fields (date, author, description, license, . . .), quite unfortunately, are not.

CRAN should go Dublin Core.

Alongside, possibly add annotation and classifications (MSC, ACM, JEL, . . .)

But will CRAN package maintainers play along? (Who are they? Why . . . ?)

The CRAN knowledge space

CRAN should go Semantic Web and beyond.

Several possible ways of “mapping” the knowledge space.

View CRAN as a huge document collection and start “text mining”.

E.g., learn a topic model or an ontology.

Start annotating ... problems: lack of codified domain knowledge (is there a good thesaurus for [academic] statistics?), multiplicity of relevant domains, ...

Possible synergies with CIS and ongoing Bibliographic Knowledge Network initiative

UseRs can contribute in a variety of ways ...

Food for thought

Are packages the right level of granularity?

Should they be smaller so that they can “more easily” be combined?

Duplication and code patterns . . .

Who gets credit in case of re-packaging or meta-packaging?

How should software be cited?

Is credit important? What is “credit”? (Something scientometric?)

Conclusion

Growth of R, expressed in particular in the number of packages in the standard repositories, has been extremely impressive, but comes at a price.

Need a variety of research initiatives and applications of state of the aRt technologies to cope.

UseRs can contribute in a variety of ways . . .

Why not use your competencies in computing *with* R for computing *on* R?

Contact

Kurt Hornik

Department of Statistics and Mathematics

Wirtschaftsuniversität Wien

Augasse 2–6, A-1090 Wien

Austria

email: `Kurt.Hornik@wu-wien.ac.at`

URL: `http://statmath.wu-wien.ac.at/~hornik`