Statistical software: where, how, and why?

Thomas Lumley,
Associate Professor of Biostatistics,
University of Washington.

R Core Development Team
Outline

• The ‘standard software’ problem

• History and sociology of R

• The survey package

• The next lingua franca?
Language constrains how we think and what we can think about.

the Sapir-Whorf thesis

When C++ is your hammer, everything looks like a thumb.

from the internet
'using standard software'

Standard software is often treated as a fixed constraint of nature by statisticians.

“readily computed in standard software”
“is not available in standard software”
“we show how to compute [estimator] using standard software”
‘tricking standard software’

Sometimes the advice even involves lying to your software:

• using Poisson regression software to get relative risks for binary data

• setting data to missing to get a survey domain estimator (*srmsnet* last week).

Why?
Where does standard software come from?
Charlton Heston brings SAS down from Mt Sinai
Robert Fay writes VPLX at Census Bureau
R emerges from the primordial soup of the internet
More seriously
Why treat software as fixed?

• The modern IT model:
  – all upgrades of software introduce bugs and require retraining.
  – most upgrades don’t add anything very useful

• Conclusion:
  – don’t do any upgrades that can be avoided.

[Transition costs increased still further by real and perceived regulatory problems in some industries.]
Alternative view

• Molecular biology/genomics model
  – high-impact publications or major commercial discoveries need bleeding-edge technology
  – Can’t afford to wait years for new methods to be implemented in software.

• Transition costs still exist, but are small compared to obsolescence costs.
Consequences

• If software changes on a multi-year basis, tricks for forcing it to do what you want are important.

• If software is upgraded twice annually, it’s easier to lobby for changes to make it work right.

• If you can rewrite the software, you can make it work without trickery.
Neophilia

• Attitude to new methods seems more predictive than better-known rationales for using or avoiding R
  - large data problems
  - regulated industries
  - GUIs
  - cost
Why now?

Software needs to be written, published, and found by users.

– Cheap computers and free compilers reduce startup costs of writing software

– Internet dramatically reduces publishing costs

– Internet dramatically reduces search costs

Widely used free software is now possible

– (as in early days of usenet, UUCP).
History of R

• Early 1990s: R. Ihaka & R. Gentleman start writing R in Auckland

• Mid 1990s: R escapes to the world. Martin Maechler persuades R&R to use GPL

(I started using R in early 1996)

• August 16, 1997: R-core created, adding people that R&R had never met.

• March 1999: first physical meeting of R-core

• Feb 29, 2000: R 1.0.0 released.
GPL

• GNU General Public License
• Prevents creation of proprietary derivatives
• More restrictive than (some of) us would like
• Made it possible for our employers to not get greedy and kill off R (cf SPIDA).
R-core management

The theologically approved models for open-source

• Benevolent dictatorship: Linux, Python, Perl
• Formal voting system: Apache
• Delegated authority: Firefox

R-core doesn’t have any such formal structure

- ‘Rough consensus and working code’: IETF
- The lack of academic credit for R
  probably helped
Contributions

• Dogma says that open-source code gets contributions from lots of people.

• Nearly all of R was written by R-core or is previously published code – eg *Applied Statistics* algorithms, LAPACK.

• Users *have* written R packages – less infrastructure needed – we don’t have to trust their competence and taste.
Packages

• The package system provides portability and enforces weak QA
  - documentation exists
  - code matches documentation
  - examples run without producing errors
  - checks for some common coding infelicities

• Logically independent of useful code
  - in practice, has a huge impact [cf Statlib]
A language for statistics

• A language for statisticians to communicate with computers, designed to entice users into writing reusable functions.

• A language for statisticians to communicate statistical methodology and data practice

The combination leads to 1500 packages.
Survey analysis in R

This is the homepage for the *survey* package, which provides facilities in R for analyzing data from complex surveys. The current version is 3.6-13. A much earlier version (2.2) was published in *Journal of Statistical Software*.

A port of the package (version 3.6-8) to S-PLUS 8.0 is available from CSAN (thanks to Patrick Aboyun at Insightful).

Features:

- Means, totals, ratios, quantiles, contingency tables, regression models, for the whole sample and for domains.
- Variances by Taylor linearization or by replicate weights (BRR, jackknife, bootstrap, or user-supplied).
- Multistage sampling with or without replacement.
- Post-stratification, generalized raking/calibration, GREG estimation.
- Two-phase designs. Estimated weights for augmented IPW estimators.
- Graphics

The NEWS file gives a history of features and bug fixes.

Comparison shopping:
Alan Zaslavsky keeps a comprehensive list of survey analysis software for the ASA Section on Survey Research Methods.

Using the survey package:

- Specifying a survey design
- Creating replicate weights
- Simple summary statistics
- Using supplied replicate weights

http://faculty.washington.edu/tlumley/survey/
Survey package

• 2002: I visit Auckland, start writing survey package
• January 2003: First version released
• April 2004: Published in J. Stat. Software
• Spring 2005: Multistage sampling, calibration
• Winter 2006: Two-phase designs
• August 2009: Book (I hope).
Why me?

[ie: Lumley? What does he know about surveys?]

Semiparametric model-based methods are converging on design-based inference
- `sandwich’ variance estimators
- model-robustness
- concept of parameters as functionals on distributions
- IPW in causal inference, missing data.
- two-phase sampling in cohort studies

Emphases are different: that’s what users are for.
Design philosophy

Mostly comes from limited resources:
- Write in high-level language
- Code reuse to expose bugs
- Keep data in memory
- Don’t optimize until someone complains (Moore’s Law)
- Emphasize features that look like biostatistics

Package is about 6000 lines of code
- cf 250,000 for VPLX,
- about 300,000 for all of R; 25,000,000 for SAS (!)
Interesting features

• Secondary analysis/modelling of large surveys
  – graphics, smoothing
  – regression models

• Simulations
  – R programming language
  – objects with accessor methods

• Calibration (raking, GREG) estimators
  – including calibration for fitting regression models.
Large data

- Current design keeps all data in memory
- On a laptop, can handle NHANES-scale analyses
  - load just relevant variables
- Inexpensive 64-bit Linux systems can handle millions of records
- Experimental revision (surveyNG)
  - keep data in database on server
  - R sends SQL queries to do large computations
  - Only data summaries return to R.
Future computing?

- Design of R is based on S, from Bell Labs
  - conscious design for five years in the future
  - Bell Labs has stopped statistical computing
  - R has too many users for experimentation
- Very few people interested in novel design of statistical computing environments
  - little commercial demand (features, yes; syntax, no).
  - little academic credit
  - little government funding.
Future computing?

- We will need something new
  - Distributed computing
  - Real-time data feeds
  - Interactive graphics
  - Very large data

- Has to be affordable and widespread, for teaching and research communication
  - Vertical-market software is a different problem

Where will we find it?