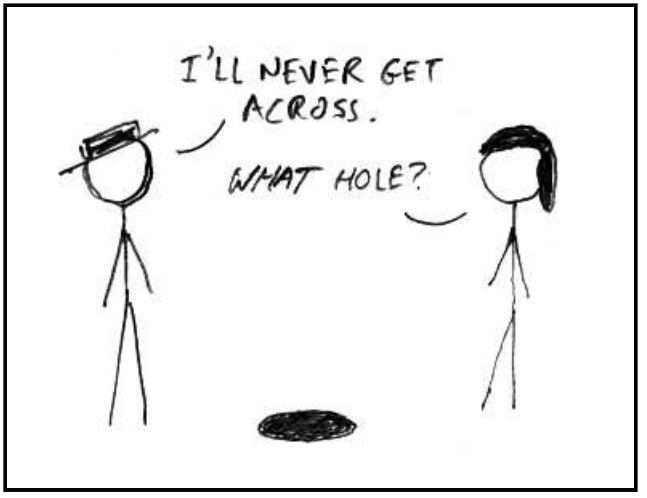
Qualifying R: the Open Source / Industry Disconnect

Kevin A. Buhr
Statistical Data Analysis Center
University of Wisconsin, Madison

May 22, 2012



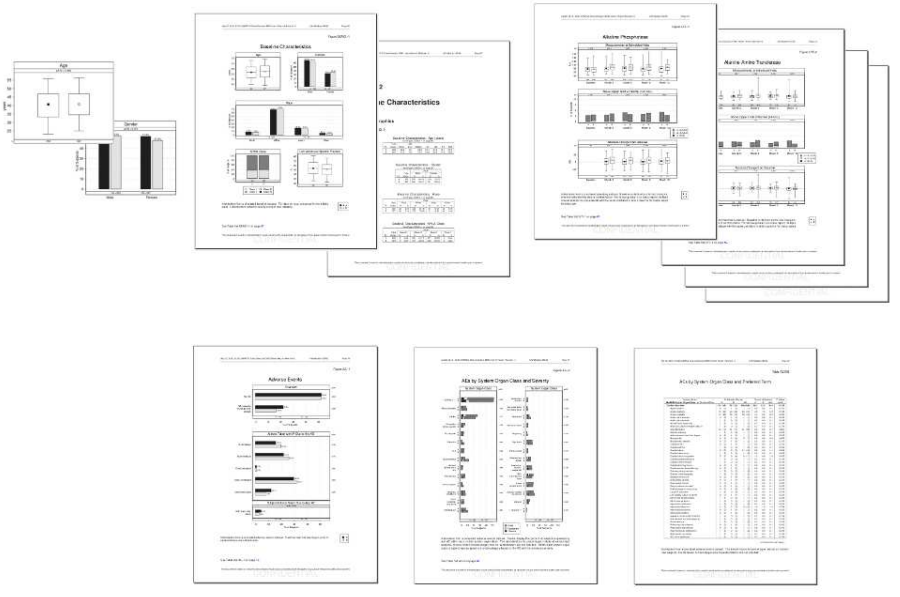
…with apologies to Randall Munroe (xkcd.com)

University of Wisconsin
Statistical Data Analysis Center

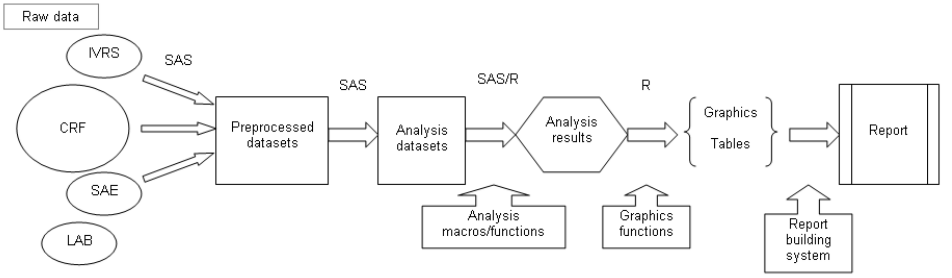
- UW SDAC is an independent statistical center supporting DMCs in monitoring industry-sponsored Phase 3 trials.
- 25 years experience, around 50 phase 3 trials/programs
- Our primary business is the preparation of interim reports on accumulating safety and efficacy data for DMC review.

SDAC Approach to IDMC Reporting

Our reports place emphasis on graphical presentations, using simple graphical elements in a cohesive page-oriented layout with layered presentation integrated into a proper report.

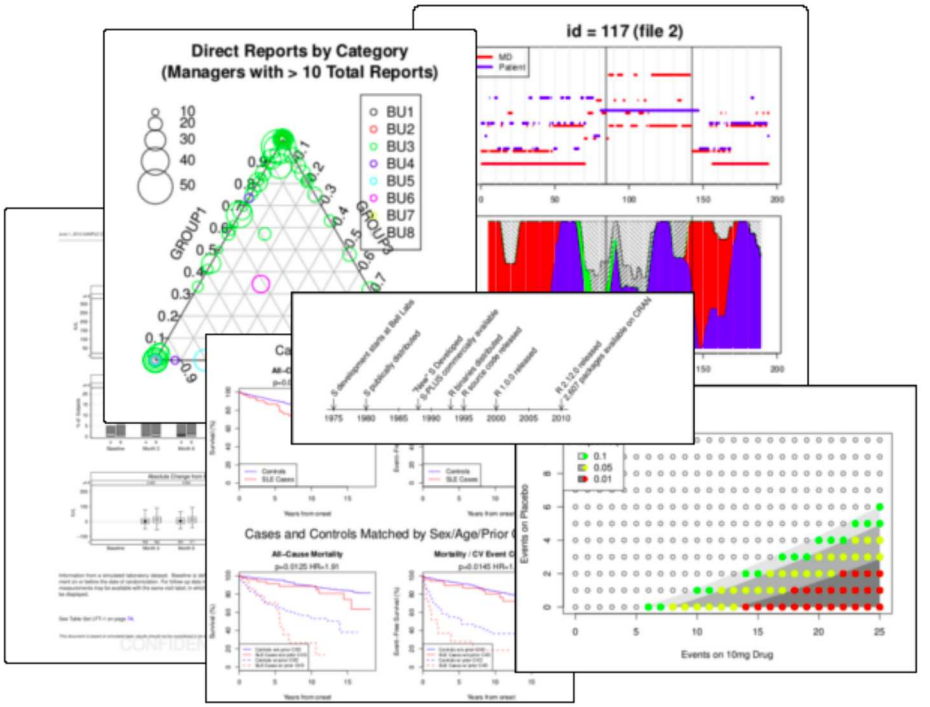


SDAC's Pipeline



- Most of our analysts:
  - use SAS for data exploration,* manipulation, and nuts-and-bolts analysis
  - use R for generating graphics and tables, and more complex analysis
- We've claimed we use SAS and R in ways that emphasize their relative strengths
- I've become somewhat skeptical

OMG, Graphics!



Business Case for R

- R's statistical and graphical capabilities
- productivity gains
- *quality and correctness*

Qualification of Programmable Systems

- qualification of an off-the-shelf (OTS) programmable statistical / data analysis environment is different than qualification of other kinds of OTS software
- it's not necessarily more difficult
- but successful qualification has different implications

Defects and Failures in Statistical Analysis

For a programmable, OTS statistical system:

- defects may exist:
  - in the OTS system itself (system defects)
  - in the programs produced by analysts (program defects)
- choice of system:
  - affects rate of failures due to system defects
  - but may also affect rate of failures due to *program* defects*
- the largest organization risk comes from defects in programs*

Defects and Failures in Statistical Analysis

Therefore:

- Choosing a qualified system over an unqualified system does not necessarily reduce risk of failure.
- Resources directed at qualifying a system to address system defects may better be spent addressing program defects.

SDAC Approach to Ensuring Quality

- SDAC isn't involved in regulatory submissions
- Because of our evaluation of relative risks due to system and program defects:
  - We don't qualify our OTS software
  - We rely on independent validation of results: another analyst attempts to reproduce analyses from raw datasets using independent code (and often independent facilities of a system or even independent systems)
- *cf.* attitude of others in R community

Defects and Failures in Statistical Analysis

Therefore,

- choosing a qualified system over an unqualified system does not necessarily reduce risk of failure
- resources directed at qualifying a system to address system defects may better be spent addressing program defects
- *but* this depends on the costs of qualification:
  - program defects => big risk, hard to address
  - system defects => small risk, easy to address

The OSS/R Community / Industry Disconnect

R's Software Life Cycle

Development and release of R code is potentially hostile to qualification:

- R's development process is open: who? qualifications?
- R has a fast release cycle (2.x.0 releases twice a year)
- no formal release or testing policy
- anyone can build and distribute a version of R and call it R 2.15.0
- volunteers create "official" binary builds:
  - built with dependencies on shared libraries (e.g., BLAS)
  - unclear if build process includes successfully running compile-time test suite
  - compile-time test results not available
  - built without installing run-time test suite

R's Package Ecosystem

R's package ecosystem (one of its greatest strengths) is potentially hostile to qualification:

- R has many packages available (3825 on CRAN) of varying quality
- How does end-user know that library(survival) is "approved" but library(Hmisc) isn't?
- install.packages()

R's Test Suite

R has a large and publicly available test suite included with the source code (for R itself, base and recommended packages, and other packages).

However,

- comprehensiveness/coverage of tests is not documented
- tests are not clearly identified and organized
- by design, some random number tests fail with small, positive probability
- on a successful run, test suite doesn't give clean output
- on any run, no summary of test results
- many CRAN packages don't include test suites (an arbitrary sample of five packages showed none with a tests subdirectory)

What, Me Qualify?

Is it any wonder there are those in industry who are skeptical R can be qualified?

The disconnect:

- By and large, industry sees these problems as insurmountable
- By and large, the R community fails to see these as problems

Some Apparent Weaknesses may be Strengths

R's Software Life Cycle

R's development process is open:

- complete source tree (with revision control) is publicly available
  svn checkout https://svn.r-project.org/R/trunk/ *path*
- complete change history is publicly available
- bug database is publicly available
  https://bugs.r-project.org

R's Software Life Cycle

R has a fast release cycle:

- new features can be widely tested
- bugs are quickly found and fixed

Anyone can build and distribute a version of R:

- "anyone" includes the end-user or site
- "anyone" includes a third-party vendor

R's Software Life Cycle

This leads to the following observations:

- fine-grained customization and control of release process is possible at site level
- no pressure to use the latest version or even any "official" version
- let other R users be your beta testers
- need critical fix but otherwise happy with current version: no problem!
- need a build of R that meets your qualification requirements: no limit on customization

To take advantage of these benefits, you must be willing to build it yourself or buy it from a third-party.

R's Package Ecosystem

R has many packages available:

- Some are of high quality, many with a test suite that runs automatically
- Consider these alternatives:
  - qualifying and using an existing CRAN package
  - writing and qualifying your own package
  - having users code one-off analyses from scratch
  - having users copy random code off a webpage

A rich package ecosystem:

- comes with some inherent risk
- probably reduces use of even riskier alternatives

R's Test Suite

- The test suite is large and publicly available
- The test suite covers R itself, base R, and recommended packages
- Some other packages come with test suites
- The test suite is extensible
- Infrastructure exists for package testing

Addressing Other Weaknesses

Again, if you are willing to build it yourself or buy it from a third-party, you can take an active role in the release cycle, including:

- implement release and testing policies
- control dependencies on shared libraries
- run and ensure success of a compile-time test suite
- install and use a run-time test suite
- apply the same scrutiny to packages
- implement package (and external code) use policies
- identify the result as a site's qualified R w/ build-time (design), installation, and run-time (operation) qualification

Criticism versus Contribution

Typical OSS community reaction to criticism versus contribution:

- "Your software doesn't do X, Y, and Z."
  "No right-thinking person would want to do X, Y, and Z, and here are one million reasons why!"
- "I've patched your software to do X, Y, and Z."
  "Wow, thanks! I've checked it into the source."
- "I want to pay someone money to do X, Y, and Z."
  "Funny you should mention that..."

R Community / Industry Collaboration

The R community and industry should collaborate on:

- cleaning up and expanding the test suite
  - document (and improve) coverage
  - organize and document tests
  - make tests run cleanly
  - provide simple summary ("certificate") of test suite results

R Community / Industry Collaboration

The R community and industry should collaborate on:

- adding build and run-time infrastructure for qualified builds:
  - make monolithic build (with all critical dependencies) easy
  - enforce success of compile-time testing
  - install compile-time test certificate and run-time tests
  - support installation qualification
  - add "qualification-only" mechanism for restricting packages

The Way Forward

- industry case studies (of all types: take, build, or buy)*
- identification of industry qualification needs
- implementation of general mechanisms to aid qualification
- *contributing success back to community*
- public, documented procedures for qualifying R
- public, written guidance to developers (core and packages) to help them address these needs on an ongoing basis
- *collaboration*, especially with package developers, to help them address these needs on an ongoing basis

Conclusions

- if you care about correctness, using R can be a net win
- the R community underestimates the seriousness of the barriers to using R in industry
- industry overestimates the difficulty in overcoming those barriers
- more dialog between R community and industry representatives can lead to shared solutions, that like R, can be made freely available