cran2deb: A fully automated CRAN to Debian package generation system

UseR! 2009 Presentation

Charles Blundell[1]   Dirk Eddelbuettel[2]

[1] Gatsby Computational Neuroscience Unit
University College London, UK

[2] Debian and R Projects
Chicago, IL, USA

Université Rennes II, Agrocampus Ouest
Laboratoire de Mathématiques Appliquées
8-10 July 2009

Overview

1. Why: Background and Motivation
2. How: Key aspects of the approach and implementation
3. Status: Where are we now?
4. Open Issues

About R – and its repositories

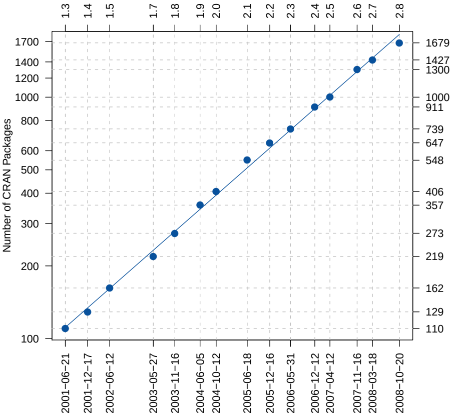
An open statistical language / environment – with lots of excellent code contributions

A few key facts that are non-controversial at a *useR!* conference:

- R is now a standard for statistical applications and research
- *"Success has many fathers"*: several key drivers can be identified as to why R has done so well
- We would like to stress *repositories* and available packages here: CRAN, as well as BioConductor and Omegahat.
- CRAN has been one of the drivers: an open yet rigorously QA'ed repository which has experienced tremendous growth

CRAN Packages

Exponential Growth



Source: Fox (2008, 2009), our calculations

- CRAN archive network growing by 40% p.a., now at around 1750 packages
- John Fox provided this chart in an invited lecture at the last *useR!* meetings.

Debian and Ubuntu

Open Linux distributions

A few key points:

- Debian is *the* community-driven Linux distribution where numerous volunteers provide over twenty-thousand packages for around a dozen architectures.
- Packages and package management “just work”: with arguably the most advanced and robust package management system, and a tremendous build and test infrastructure.
- Ubuntu has taken Debian, added a fair amount of spit and polish, as well as regular bi-annual releases, and has rapidly gained mind- and well as market-share as the Linux distribution to beat.
- We also note that the CRAN backend is implemented on Debian.

Why build Debian R packages?

Combining R and Debian

Bates, Eddelbuettel and Gebhard (UseR! 2004) listed a number of reason that still hold:

- **Dependencies** are resolved automatically: *it just works*
- **Convenience** of installing binary packages via `apt-get`
- **Quality control** as build daemons, automated rebuilds, porting, ... all ensure that everything is pretty much buildable all the time
- **Scalability** as building one binary package and scripting installation on a cluster beats doing lots of manual installations
- **Common platform** as Debian forms the base for Ubuntu and several other derivative or single-focus distributions
- **Different architectures** ranging from small arm or MIPS based systems to amd64, sparc64, hppa or even s390 mainframes
- **Audience** given the reach of Debian and Ubuntu, large number of users can be reached with little effort

Comparing two approaches

What have we learned?

Eddelbuettel, Vernazobres, Gebhard and Möller (UseR 2007) implemented a system which provides a basis for comparison:

Then

- Top-down approach
- Monolithic and large Perl program
- Meta-information encode directly as Perl hashes in program
- Re-implementing chunks of what R does in parsing archives
- Not very robust

Now

- Bottom-up approach
- Collection of R and shell scripts, also lots of SQL
- Re-using R internal infrastructure as much as possible
- Influenced by CRANberries and its 200 lines of R code to monitor and summarize CRAN changes

Technology Overview: Big Picture

Key components

Our cran2deb system is implemented as a collection of small tools:

- cran2deb itself is a wrapper script calling out to about twenty other ’worker’ scripts implementing the principal commands
    - ’worker’ scripts are written in R (for littler), Korn/Bash shell, and in the Plan9 shell rc
    - these scripts are small: the largest is 4 kb and only seven are larger than 1 kb
    - this is recursive: ’help’ is one of these scripts scanning for doc-strings in the other scripts
- cran2deb is also an R package that is being called by some of the R scripts; the R package has just over 1500 lines of code, and it calls out to R functionality from package utils and tools.

Technology Overview

A walk through: some details

What does cran2deb do:

- pulls new meta-data from CRAN via `available.packages()`
- detects new (or changed) packages and builds each one via:
  - map declared R dependencies onto cran2deb packages
  - map free-form SystemRequirements onto Debian packages
    - Rules for this shared among packages—many packages “just work”.
  - add any undeclared dependencies (this applies to just 36 packages and often entails only loading, say, MASS).
  - build each package in its own isolated, clean, fresh, up to date build environment via pbuilder: this looks like a fresh install of Debian and ensures correctness of dependencies.
- checks package quality via Debian’s lintian.

Technology Overview

A walk through: some more details

What does cran2deb do (cont.):

- uses RSQLite backend for cran2deb state: everything from package meta-information, blacklist of bad packages, to build logs.
- checks for a free license of a package before its built:
    - initially: handcrafted regular expressions to match licenses.
    - some packages ignore “Writing R extensions” guidelines concerning the License: field: how many ways to write GPL?
        - initialised vs. its expansion (GPL vs. GNU general public license)
        - license vs. licence
        - see `http://www.gnu.org/GPL`
        - (v, version) (2.0, 2) or (higher, later, newer, greater, above)
        - typos of the above
        - file LICENSE: contents reformatted in arbitrary ways
    - now: strip white space and perform other harmless transforms and match SHA1 checksums to determine license; likewise for contents of LICENSE file.

Technology Overview

Continued

Re-use, re-duce, re-cycle:

- R ’s infrastructure is used to obtain the R view of the world: what packages and where, first approximation to dependencies.
- All this uses the Debian build infrastructure, notably the pbuilder chroot environment and the package management system
- cran2deb sets the build environment up by invoking the proper Debian scripts
- the ‘production line’ of packages is fully automated via cron and report status summaries by email
- per-package patches are allowed (currently eleven packages have mostly trivial patches)
- source code is available via the r-forge subversion repository and archive

Building 1700+ package

Summary from a package views

It's easy: basically *everything* builds and is available as a Debian package (complete with full dependencies) — apart from:

- 17 packages that are *not free enough*:[1] mclust, mclust02, ConvCalendar, SDDA, conf.design, isa2, optmatch, rankreg, realized, rngwell19937, tnet, spatialkernel, Bhat, PTAk, PredictiveRegression, RLadyBug, mapproj
- 1 package that is obsolete: xgobi
- 2 package that break building packages via cran2deb:[2] dprep, EngrExpt
- 1 package that is not built for 'other' reasons:[3] sabreR

[1] Generally these do not allow commercial use, modification and/or distribution with the exception of ConvCalendar which gives no modification or distribution rights.

[2] They take down the cronjob; we are stumped as to why.

[3] It contains binary code.

Building 1700+ package

Continued

- 47 packages that have *unsatisfied dependencies*:[4] ROracle, Rlsf, Rsge, CarbonEL, VhayuR, gputools, klaR, wgaim, svGUI, RScaLAPACK, caMassClass, Rcplex, ADaCGH, DAAGbio, GFMaps, GOSim, Metabonomic, classGraph, gcExplorer, logilasso, pcalg, celsius, multtest, hopach, GExMap, LMGene, PCS, SubpathwayMiner, gene2pathway, PhViD, SNPMaP, qdg, lsa, mpm, sisus, metaMA, clustTool, clustvarsel, SpectralGEM, bayesCGH, crosshybDetector
- 8 package that (as of end of June) fail for unclassified reasons: IDPmisc, Rsymphony, SuppDists, aroma.apd, aroma.core, aroma.affymetrix, cmprskContin, mvgraph

But everything else—currently 1770 packages—builds and is available via `apt-get` and other package management frontends!

[4] Some require other commercial software, some require software we classified as non-free, some require BioConductor packages.

Status and credits

Ready for wider deployment and testing

Who do we owe, and where is it at:

- The ground-work was provided during Google Summer of Code (GSoC) 2008 under the umbrella of the Debian project. We thank Google for the GSoC support.
- Currently we are using a (small) Xen-instance on a server at WU Wien to host two Debian pbuilder chroots and an archive. We thank WU Wien/CRAN for hosting and cpu cycles.
- 1700+ packages for i386 and amd64 on Debian testing
- In daily use for the last few weeks!

So just add one of these URLs:

```
deb http://debian.cran.r-project.org/cran2deb/debian-i386 testing/
deb http://debian.cran.r-project.org/cran2deb/debian-amd64 testing/
```

Question to be addressed

For cran2deb to migrate out of beta testing

- **Licenses:**
  - What can or cannot be (re-)distributed by CRAN and its mirrors?
  - What can or cannot be used (and/or modified) by all users?
- **Externtal dependencies**
  - BioConductor is the single largest source: BioBase, RGraphviz, etc
  - Other external libraries or tools not in Debian
  - Commercial external dependencies: SGE, LSF, Oracle, Vhayu
- **Scope**
  - Builds for other architectures?
  - Builds for other Debian flavours such as Ubuntu?
  - Builds of other repositories: BioConductor? R-Forge?