Faster ML(ops) With r2u

CRAN as Ubuntu Binaries

Dirk Eddelbuettel

Chicago ML and MLOps User Groups Fall Meeting

25 Sep 2024

https://dirk.eddelbuettel.com/papers/ml_mlops_r2u_sep2024.pdf

ILLINOIS

Problem Setting For This Talk

Key Issues For *MLOps*

- We want *fast* automated task execution
- That is *easy* to setup and maintain
- And *reliable* without debugging nightmares
- Plus an additional “won’t break” aspect (more later)

Specific Context

Setup

- We will focus on R here as it 'my tool of choice'
  - But nothing that we present is *specific* to R
  - One could create a similar for Python, Julia, Rust, ...
  - Essentially any language with a repository
- We will also focus on `.deb` binary packages
  - Again nothing specific on this choice
  - `.rpm` and others such as `brew` packages could be used
  - Again 'my tool of choice'
- We will equate 'Deployment' with Linux
  - Common in cloud and server use

So why CRAN ?

CRAN rules R

- Excellent repository with stringent quality control
- “Everything builds at @HEAD”
- This enables scripting and automation
- But on Linux by default compilation from source
  - we talk about p3m.dev a little later
- And compilation from source
  - can be slow (for non-trivial tasks)
  - can be error prone (dependencies ...)

Linux Distributions Offer Help

Features

- Group packages for easier installation
- Organize dependencies (e.g. by analysing dynamic libraries, and more)
- Provide ‘cohorts’ aka ‘releases’
- Exist in a range
  - ‘vertical’ with one platform as e.g. Linux distributions
  - ‘horizontal’ as e.g. Conda across platforms
- We will focus on the vertical case

Linux Distributions Offer Help

Why Focus Here?

- The example of Ubuntu is quite intriguing
- It is generally “pretty good, pretty complete, pretty current”
- As well as polished and maintained well-enough (incl security)
- Hence default at eg GitHub Actions and some other instances
- Broad support for ‘other’ software in Ubuntu .deb form
  - Intel supports it for ‘oneAPI’ (i.e. MKL, TBB, ...)
  - NVidia supports it to support its GPUs

This is not to start an argument. Similar efforts can be done and are being done for Fedora and OpenSUSE.

Package Manager

In a Distribution

- Generally runs as system user, can install system dependencies
- Can cover everything that is offered within the distribution
- Can cover version dependencies and possibly version pinning
- But cannot cover what is not packaged: often the relevant applications

In an Application

- Application-level package managers are common and excellent
- They cover a lot of ground getting *other packages*
- They can cover package dependencies and possibly pinning
- They generally cannot install system dependencies

So Promise In Approach To “Join” Both

- Integrate the application packages into the distribution repositories
- Now both united and dealt with jointly the system package manager
- Application level requirements satisfied by system level manager

Past Attempts

- Several incomplete attempts in the past, max’ed out at 6k out 21k
- Some details in our ‘binary R packages’ arXiv paper (and earlier r2u talks)
- But now r2u is the first that got it working

apt install r-cran-whatever

- Every CRAN package, prefixed with r-cran- and name lower-cased
- The distribution build process wrapped around the R installation
- So it is guaranteed the package can actually be loaded
- Thus every dependency formally declared, resolvable – and tested
- Supported (currently) for the three Ubuntu LTS releases on x86_64
- More details at `https://eddelbuettel.github.io/r2u`

Alternatives

p3m.dev

- Compelling system offering breath across packages and OS choices
- Generally provides binaries
  - But sometimes only source, unclear ex ante if one gets source or binary
  - This can differ for the same package across LTS releases
- Not integrated with the package manager: system deps are ‘harder’
  - Mechanism to obtain required commands, but not automated / integrated
- But useful and e.g. it does provide our inputs here
- Overall ‘not bad at all’ but we can do better

This would be a fair place to compare to PyPI, Conda,... but that is out scope here – and I do not really use those.

Advantages of System Integration

Missing Libraries are Installed

- Installing, say, a PostgreSQL package will install the Postgres library

Used Libraries are Never Removed

- Package manager *knows* that the client package uses libpqN
  - So when the system updates the library from releases N to N+1
  - Package is no longer left broken by removing the dependency

In the narrowest sense you could argue that an ML training run is ephemeral and system updates never happen. Fine. It is still damn convenient on all other systems.

Advantages of System Integration

```
FROM rocker/r-ubuntu:20.04

RUN     apt update -qq \
    && apt install -y r-cran-rcppgsl libgsl-dev \
    && Rscript -e 'install.packages("RcppZiggurat")' \
    && Rscript -e 'library(RcppZiggurat); cat("All good\n")'

## Upgrade from focal to jammy, which means GSL 2.3.* to 2.7.*
RUN     sed -i -e 's/focal/jammy/g' /etc/apt/sources.list \
    && sed -i -e 's/focal/jammy/g' \
       /etc/apt/sources.list.d/c2d4u_team-ubuntu-c2d4u4_0_-focal.list

## 326 packages if we upgrade, so skip now, run if you prefer
#RUN     apt update -qq \
#    && apt upgrade -y

## Now (for expedience just) upgrade RcppGSL, brings
## upgraded libgsl28, removes libgsl27
RUN     apt update -qq \
    && apt install -y r-cran-rcppgsl

## And RcppZiggurat is borked -- this fails because libgsl23 is gone
## so we comment it out not run break the docker build, but see 'manually'
#RUN Rscript -e 'library(RcppZiggurat)'
```

Simple Demo "Proof": Breaks under Ubuntu

We install eg RcppGSL (available as binary) and the GSL, then build RcppZiggurat

We upgrade from 'focal' to 'jammy', this gets us a new libgsl2* version.

And that breaks RcppZiggurat.

Similar for other versioned shared libraries: libicu*, libpq*, ...

Advantages of System Integration

```
FROM rocker/r2u:20.04

## under r2u this installs RcppZiggurat binary and its dependencies
RUN    apt update -qq \
    && Rscript -e 'install.packages("RcppZiggurat")' \
    && Rscript -e 'library(RcppZiggurat); cat("All good\n")'

## Upgrade from noble to oracular, which means GSL 2.3.* to 2.7.*
RUN    sed -i -e 's/focal/jammy/g' /etc/apt/sources.list \
    && sed -i -e 's/focal/jammy/g' /etc/apt/sources.list.d/r2u.list

## Now (for expedience just) upgrade RcppGSL,
## brings upgraded libgsl27, removes libgsl23
RUN    apt update -qq \
    && apt install -y r-cran-rcppgsl

## So RcppZiggurat is not broken as it got upgraded too
## Because the package manager knows it was affected
Rscript -e 'library(RcppZiggurat); cat("All good\n")'
```

Simple Demo “Proof”: Works under r2u

Doing equivalent steps under r2u but with packaged RcppZiggurat

But now upgrading RcppGSL … also gets updated RcppZiggurat: No breakage.

To replicate, Dockerfiles from previous and this slide are at GitHub.

BSPM: Another Key Component

Bridge To Package Manager (by Iñaki Ucar)

- Cleverly 'intercepts' `install.packages()` calls made by R
- So `install.packages(c("xgboost", "mlpack"))` does what you expect
- Translates these into corresponding `apt` calls
  - Now *R users* do not need to know about `apt`
  - Also works with `dnf` and other package managers
- We can just take a package and say 'install dependencies'
- Ideal use case is for example continuous integrations
  - Drop-in setup, no sys admin, no debug
  - Use for example by my r-ci uses it

Actual CI Example

Only whitespace removed to fit display

Can be used as drop-in file `ci.yaml`

“Easy. Fast. Reliable.” for CI

More documentation at r-ci

```yaml
# Run CI for R using https://eddelbuettel.github.io/r-ci/
name: ci
on:
  push:
  pull_request:
env:
  _R_CHECK_FORCE_SUGGESTS_: "false"
jobs:
  ci:
    strategy:
      matrix:
        include:
          - {os: macOS-latest}
          - {os: ubuntu-latest}
    runs-on: ${{ matrix.os }}
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Setup
        uses: eddelbuettel/github-actions/r-ci-setup@master
      - name: Bootstrap
        run: ./run.sh bootstrap
      - name: Dependencies
        run: ./run.sh install_all
      - name: Test
        run: ./run.sh run_tests
```

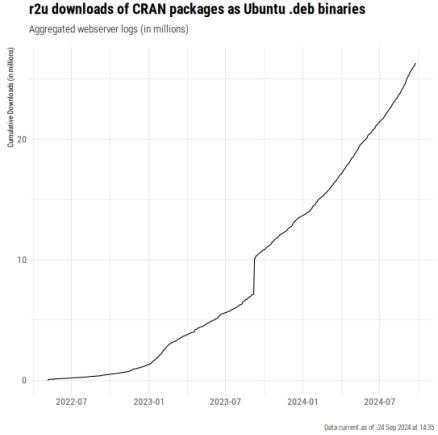
How Does It Work?

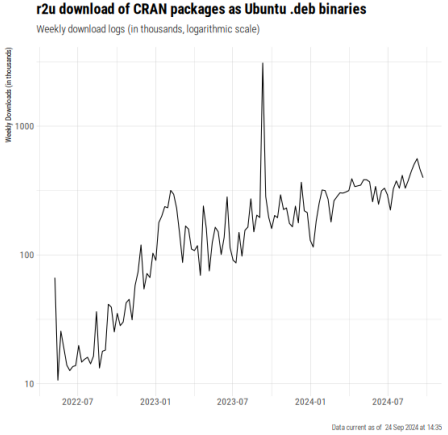
- We run `dpkg-buildpackage` for each package inside a Docker container
  - This gets us proper library dependencies as if Ubuntu built it
  - Proper steps of a genuine distribution package
- We accelerate the builds where we can by using p3m.dev R binaries
  - Which we unpackage inside the directory tree of the build
  - So in most (but not all) cases we can skip the R side of package build
- Small amount of meta data for extra dependencies we need to declare
  - Or a few builds we blacklist for various reasons
- We use standard tools to create a repository for, server it locally
- Internet2 mirror thanks to Tech Support in Liberal Arts & Sciences at Illinois

How Does One Use It? How To Get Started?

- Documentation site has script with four (or five with `bspm`) steps for Ubuntu
  - Used thousands of times in continuous integration at GitHub
  - There is also a dedicated GitHub Action to have this done in one step
- Dedicated Docker containers for deployment `rocker/r2u` for three flavours
- Or 'manually' apply to script steps to a standard Ubuntu system
- Or 'drop in' the CI script from the previous page

Usage Steadily Growing, Now over 26 Million Packages Shipped





Discussion

- r2u shows we can integrate curated 'application repositories' into a Linux distro
- Doing so creates 'total sum greater than sum of parts' effects
  - We get all the benefits of our preferred compute environment (here: Ubuntu)
  - We get all the packages of our preferred application language (here: CRAN for R)
  - The integrations is *fast, easy, reliable* as fully featured binaries are used
- Possible Extension: add arm64/aarch64 for the approx 25% binary packages
- This can serve as model for other languages and/or environments
  - Nothing *fundamentally* limiting this to either Ubuntu or R + CRAN

That's It!

Thank You! And Thanks To

- R (package) authors for creating something wonderful in the commons
- The CRAN team for all they do making it *reliably* accessible
- All past 'cran2deb' members: Albrecht, David, Stefan, Charles, Don, Michael, ...
- posit for p3m.dev, and Iñaki for BSPM
- Rami Dass and Liberal Arts & Sciences IT at UIUC for hosting r2u
- My GitHub sponsors for all the coffee money

And see https://eddelbuettel.github.io/r2u/ for **r2u**

r2u Useful For Non-Apt Binaries

One can also install Ubuntu 24.04 binaries from r-universe and r-multiverse, the

```
rp <- c("https://community.r-multiverse.org/bin/linux/noble/4.4",
        getOption("repos"))
install.packages(c("glaredb", "polars"), repos=rp)
```

On an r2u container, installs two Rust-based R packages not-on-CRAN as *r-multiverse* binary Ubuntu packages along with their dependency `nanoarrow` (here installed from r2u via `bspm`). In 23 seconds.

Appendix II

Some More Screencapture "Movies" Of r2u In Action

- tidyverse and rstan and brms in 33s (gif)
- tidyverse in 20s (gif)
- tidyverse and sf and lme4 (mp4)
- Single-Cell Genomics: 389 packages in 75 seconds (gif)
- ML packages xgboost, lightgbm, mlpack, torch in 10 seconds (gif)

These and more are at `https://github.com/eddelbuettel/images`