More Than 25 Years of CRAN

WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Kurt Hornik

How it all began

Once upon a time . . .

How it all began

Once upon a time . . .

Almost 30 years ago . . .

How it all began

Once upon a time . . .

Almost 30 years ago . . .

FL and KH had fallen in love with R (and gotten write access to the R sources)

How it all began

Once upon a time . . .

Almost 30 years ago . . .

FL and KH had fallen in love with R (and gotten write access to the R sources)

The base system was rather small, and extensions providing add-on functionality were needed.

Before R these was S . . . contributed extensions for S available from Statlib, bur rather inconvenient to use. Typically easy to port to R (date, chron, htest, . . .), but how to use conveniently?

And of course, what about extensions newly written for R?

How it all began

FL and KH (actually, the whole “Center for Computational Intelligence” at Technische Universität Wien) were proud & happy users of Debian (testing) and its package management system

So went went about doing something similar for R:

- Implement a package management system (tools for build, check and install)
- Set up a repository for distributing packages

What’s in a name?

We already knew and used

The Comprehensive TeX Archive Network (CTAN)
The Comprehensive Perl Archive Network (CPAN)

so it seemed obvious to go for

The Comprehensive R Archive Network (CRAN)

What’s in a name?

We already knew and used

The Comprehensive TeX Archive Network (CTAN)
The Comprehensive Perl Archive Network (CPAN)

so it seemed obvious to go for

The Comprehensive R Archive Network (CRAN)

Actually, “comprehensive” was more meant as

all kinds of things for R (code, docs, data, . . .)

and not

all things for R

(in particular, *not* providing the only CRAN-style package repository).

Why Debian testing matters

Debian always has at least three releases in active maintenance:

stable:	the latest officially released distribution. Production release for general users.
testing:	packages in a queue for stable. For users who like having more recent versions of software.
unstable:	where active development occurs: for users who like to live on the edge.

Debian releases rather infrequently $\Longrightarrow$ testing worked nicely for us

Except when it got frozen for a new release: *DON'T DO THAT*.

And except for things that took very long to make it from unstable to testing (e.g, due to issues with reverse dependencies): *DON'T DO THAT*.

CRAN design principles

Altogether, this suggested to have

one "rolling" repository for "stable" releases of R packages

CRAN design principles

Altogether, this suggested to have

one "rolling" repository for "stable" releases of R packages

- no hosting of development versions of packages (use forges instead)

CRAN design principles

Altogether, this suggested to have

one "rolling" repository for "stable" releases of R packages

- no hosting of development versions of packages (use forges instead)
- collaborative development through controlled releases and active repository maintenance

CRAN design principles

Altogether, this suggested to have

one "rolling" repository for "stable" releases of R packages

- no hosting of development versions of packages (use forges instead)
- collaborative development through controlled releases and active repository maintenance
- no releases for the whole repository

CRAN design principles

Altogether, this suggested to have

one "rolling" repository for "stable" releases of R packages

- no hosting of development versions of packages (use forges instead)
- collaborative development through controlled releases and active repository maintenance
- no releases for the whole repository

Clearly, very nice if you want timely access to state-of-the-art software without too much breakage.

Particularly nice when the state-of-the-art changes rapidly

CRAN design principles

We then have the following.

Theorem: *Let CRAN be a "rolling" repository of mostly "working" packages. Then CRAN must be actively maintained.*

CRAN design principles

We then have the following.

Theorem: *Let CRAN be a “rolling” repository of mostly “working” packages. Then CRAN must be actively maintained.*

Proof: obvious.

CRAN design principles

We then have the following.

Theorem: *Let CRAN be a “rolling” repository of mostly “working” packages. Then CRAN must be actively maintained.*

Proof: obvious. If a package *p* no longer “works” due to changes outside of *p* (other package, R itself, toolchain/systems, . . .), then *p* must get updated, and this will not happen automagically.

CRAN design principles

We then have the following.

Theorem: *Let CRAN be a "rolling" repository of mostly "working" packages. Then CRAN must be actively maintained.*

Proof: obvious. If a package p no longer "works" due to changes outside of p (other package, R itself, toolchain/systems, ...), then p must get updated, and this will not happen automagically.

Corollary: *Under the above conditions, let p be a package in CRAN. Then p also needs to be actively maintained.*

CRAN design principles

We then have the following.

Theorem: *Let CRAN be a "rolling" repository of mostly "working" packages. Then CRAN must be actively maintained.*

Proof: obvious. If a package p no longer "works" due to changes outside of p (other package, R itself, toolchain/systems, . . .), then p must get updated, and this will not happen automagically.

Corollary: *Under the above conditions, let p be a package in CRAN. Then p also needs to be actively maintained.*

Proof: trivial.

CRAN design principles

We then have the following.

Theorem: *Let CRAN be a "rolling" repository of mostly "working" packages. Then CRAN must be actively maintained.*

Proof: obvious. If a package p no longer "works" due to changes outside of p (other package, R itself, toolchain/systems, . . .), then p must get updated, and this will not happen automagically.

Corollary: *Under the above conditions, let p be a package in CRAN. Then p also needs to be actively maintained.*

Proof: trivial.

More later on the possible interpretations of "working" . . .

CRAN design principles

Clearly, how CRAN works is rather unusual:

- Allows maintainers of packages (including base packages) to move things forward rather quickly (within reason)
- This can only work if affected maintainers react rather quickly.

This is CRAN's notion of

actively maintained packages

(so both a right and a duty).

CRAN design principles

One of the key ingredients of the social contract established by agreeing to the CRAN Repository Policy.

Challenges:

- maintainers may not be aware of what they contract for (modern world style click on accept without reading)

CRAN design principles

One of the key ingredients of the social contract established by agreeing to the CRAN Repository Policy.

Challenges:

- maintainers may not be aware of what they contract for (modern world style click on accept without reading)
- maintainers may be aware and disagree, but nevertheless want to have their package on CRAN (professional recognition, . . .)

CRAN design principles

One of the key ingredients of the social contract established by agreeing to the CRAN Repository Policy.

Challenges:

- maintainers may not be aware of what they contract for (modern world style click on accept without reading)
- maintainers may be aware and disagree, but nevertheless want to have their package on CRAN (professional recognition, . . .)
- the fine print parts are missing (e.g., does “work” include the “important NOTEs” and what the .u.. are these?)

CRAN design principles

One of the key ingredients of the social contract established by agreeing to the CRAN Repository Policy.

Challenges:

- maintainers may not be aware of what they contract for (modern world style click on accept without reading)
- maintainers may be aware and disagree, but nevertheless want to have their package on CRAN (professional recognition, . . .)
- the fine print parts are missing (e.g., does “work” include the “important NOTEs” and what the `.u..` are these?)
- communication via email sucks

Email madness

CRAN communicates with package maintainers (in order to “activate” them) old-style via emails.

Email madness

CRAN communicates with package maintainers (in order to “activate” them) old-style via emails.

Provides some level of “security” (by default updates can only be performed by someone with access to the maintainer email address).

Email madness

CRAN communicates with package maintainers (in order to “activate” them) old-style via emails.

Provides some level of “security” (by default updates can only be performed by someone with access to the maintainer email address).

But:

- sending emails is getting increasingly complicated

Email madness

CRAN communicates with package maintainers (in order to “activate” them) old-style via emails.

Provides some level of “security” (by default updates can only be performed by someone with access to the maintainer email address).

But:

- sending emails is getting increasingly complicated
- there is no guarantee that emails were actually received

Email madness

CRAN communicates with package maintainers (in order to “activate” them) old-style via emails.

Provides some level of “security” (by default updates can only be performed by someone with access to the maintainer email address).

But:

- sending emails is getting increasingly complicated
- there is no guarantee that emails were actually received
- emails may be undeliverable, e.g., when email addresses change

Email madness

Bounces are bad (no longer an active maintainer)

Non-bounces not necessarily good.

Need to have better tools for tracking maintainers

Need to have better tools for communicating with maintainers

Stability

All nice, but what if you want/need "more stability"? It depends . . .

Having stable repository releases is not the answer to everything (hence, not 42). In particular, as there may be serious bugs that need fixing (or changes needed for system changes).

One can in fact get very stable releases by using Windows or macOS binaries for old versions of R (e.g., binaries for R 4.2 or older were frozen when R 4.4.0 was released)

One can in fact always access older versions of CRAN packages (things no longer in current are in the archive), and could write tools for conveniently accessing these.

Well, not quite: currently only true for *source* packages. Binary builds do not necessarily persist.

Issues with binary packages

In fact, things are even worse.

Binary packages may change without changing their version.

E.g., if the binary for source package foo 1.2.3 gets rebuilt (e.g., because of changes in the toolchain or dependency ABIs), it will still/again show as `foo_1.2.3.zip` or `foo_1.2.3.tgz`.

Bad for stability/reproducibility ("I want the foo for YYYY-mm-dd" cannot be guaranteed for current versions of R).

Issues with binary packages

Part of the problem: no good naming scheme!

- Binary version 1.2.3-1 could also be for source version 1.2.3-1
- Binary versions like 1.2.3+b1 or 1.2.3-b1 are not possible: that chance was “missed” once upon a time

(Of course, Debian had that, but looked too complicated) when developing the R package management system all these years ago.)

These issues clearly need to be addressed eventually (better soon, along with enhancements needs for Linux binary packages).

Stability again

You’re still not convinced and want/need stable releases of CRAN?

Stability again

You’re still not convinced and want/need stable releases of CRAN?

You can do this yourself!

Stability again

You’re still not convinced and want/need stable releases of CRAN?

You can do this yourself!

Point I already made many years ago: idea is that “package objects” are unique and persistent, and everyone can create services providing specific/desired “subviews”.

(Perhaps better/easier to only do this for subsets of interest.)

Stability again

You’re still not convinced and want/need stable releases of CRAN?

You can do this yourself!

Point I already made many years ago: idea is that “package objects” are unique and persistent, and everyone can create services providing specific/desired “subviews”.

(Perhaps better/easier to only do this for subsets of interest.)

In particular, everyone could provide a daily snapshot service!

Active CRAN repository management in practice

Based on running *regular checks* (for the packages currently in the repository)

Using either standard builds of current versions of R on Linux, macOS or Windows, or special/enhanced builds (sanitizers, different BLAS/LAPACK subsystems, special toolchains/settings, . . .).

Important problems (ERRORs, WARNINGs and the important NOTEs) from these are reported (via email) to maintainers with a fixed (typically) short deadline for a fix.

These problems can conveniently be seen from the package check results pages (including “additional issues”).

Sometimes (and then often confusingly), there may be issues to be addressed which do not show in the regular check results/pages

Active CRAN repository management in practice

Running these checks is (mostly) automatic. Dealing with problems found is definitely not!

Active maintainers submit updates in time which successfully go through the *incoming checks*.

Not so active maintainers (“non-maintainers”) miss the deadline.

Packages without revdeps then typically get archived.

Packages with revdeps get another reminder (“final deadline”), and if that deadline is missed too, another reminder with the revdep maintainers in cc.

Archival and escalation cost the CRAN team’s scarce time and energy.

Submission checks (aka incoming checks)

Submissions are done via a web interface (or wrappers around this, but the social contract part *is* important, see above).

Submissions are always checked automatically on Debian testing with current LLVM compilers, and Windows using GCC compilers.

Submission checks (aka incoming checks)

- If there are problem, auto-archive.
- Otherwise, if a new package (newbie), inspect in detail by the submission team (currently funded by the R Foundation).
- Packages which got archived currently need a manual processing step.
- If there were additional issues for the current version, these are manually re-checked.
- Otherwise, if there are no revdeps, auto-publish (unless there is a maintainer change).
- Otherwise, check revdeps, and if no new problems in these, auto-publish.
- Otherwise, manual actions by the submission team are needed.

Some basic stats

Number of package objects in current: 20977

Number of package objects in archive: 147361

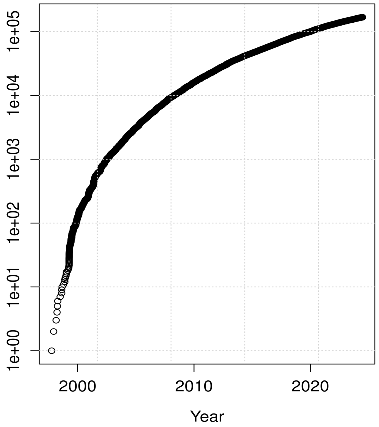
Number of packages only in archive: 6516

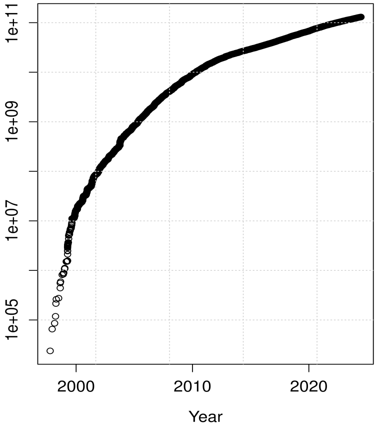
Total size in GB (decimal): 130.21

Total size in GiB (binary): 121.27

Some basic stats

Log number and log size of CRAN packages (current and archive):





Some basic stats

Recency of current packages:

```
2008 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020
   2    1    4   29   38   63  267  594  741  964 1131 1649
2021 2022 2023 2024
1997 3178 5387 4933
```

Some basic stats

Numbers of package updates:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   2.000   3.000   6.121   7.000 217.000 
```

Some basic stats

Numbers of package updates:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   2.000   3.000   6.121   7.000 217.000 
```

In case you want to know . . .

Some basic stats

Numbers of package updates:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   2.000   3.000   6.121   7.000 217.000 
```

In case you want to know . . .

```
         rgdal           mgcv RcppArmadillo      spatstat        Matrix 
           152            167           170           212           217 
```

Some basic stats

Numbers of package updates:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   2.000   3.000   6.121   7.000 217.000 
```

In case you want to know . . .

```
         rgdal           mgcv RcppArmadillo      spatstat        Matrix 
           152            167           170           212           217 
```

Update intervals (for packages added before 2023-12-31):

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  20.36  269.85  561.23  924.41 1204.50 9757.00 
```

Note that the Policy says “no more than every 1–2 months”

Regular check problems part one

```
                                     OK NOTE WARNING ERROR FAILURE     n
r-devel-linux-x86_64-debian-clang 15911 5571      42    39       0 21563
r-devel-linux-x86_64-debian-gcc   15919 5556      14    48       1 21538
r-devel-linux-x86_64-fedora-clang 13771 8797      13    28       0 22609
r-devel-linux-x86_64-fedora-gcc   14614 7496      13    25       0 22148
r-devel-windows-x86_64            15490 6129      14    39       0 21672
r-oldrel-macos-arm64              16272 5229      25   177       0 21703
r-oldrel-macos-x86_64             16133 5485      40   165       0 21823
r-oldrel-windows-x86_64           17140 4145      14    35       0 21334
r-patched-linux-x86_64            16055 5370       7    49       2 21483
r-release-linux-x86_64            16019 5374       6    58       1 21458
r-release-macos-arm64             14920 6962      22   131       0 22035
r-release-macos-x86_64            14860 6977     187    79       0 22103
r-release-windows-x86_64          15490 5998     240   171     119 22018
```

Regular check problems part two

```
                                                Status
Check                                            NOTE WARNING ERROR FAILURE
  tests                                             0       0    23       1
  examples                                          0       0    14       0
  re-building of vignette outputs                   0       1    11       0
  compiled code                                   191       5     0       0
  whether package can be installed                  0       5     0       0
  S3 generic/method consistency                     0       1     0       0
  for code/documentation mismatches                 0       1     0       0
  use of S3 registration                            0       1     0       0
  Rd files                                       1990       0     0       0
  LazyData                                       1938       0     0       0
  C++ specification                               530       0     0       0
  package dependencies                            282       0     0       0
  for GNU extensions in Makefiles                 205       0     0       0
  package subdirectories                          160       0     0       0
  for non-standard things in the check directory   97       0     0       0
  DESCRIPTION meta-information                     54       0     0       0
  HTML version of manual                           50       0     0       0
  R code for possible problems                     23       0     0       0
  Rd cross-references                              19       0     0       0
  dependencies in R code                           11       0     0       0
```

Reported issues

(Currently based on “text mining” of the emails. Improvements under way.)

Total number of Issues reported since 2023-01-01: 7222

2nd deadline (final): 719, 3rd deadline (revdeps): 354

Problems for additional issues: 697, with web access: 454

Numbers of issues reported per day:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    3.00    6.00   13.35   11.00  734.00
```

Number of days without issue reports: 28

Quite impressive, given that the time span includes CRAN holidays . . .

Repository actions

Manual repository actions (add/new, archive, unarchive, remove) programmatically tracked since 2022-11-01:

Total numbers of actions:

```
     new    archived unarchived     removed
    3305        2477       1372          18
```

Numbers of actions per day:

```
     new    archived unarchived     removed
    5.51        4.13       2.29        0.03
```

Repository actions

Numbers of actions according to month for the past 12 months:

```
        new archived unarchived removed
2023-07 117       66         43       0
2023-08 197      336         69       0
2023-09 167      118         78       3
2023-10 166      223         88       0
2023-11 167       97         85       0
2023-12 122       42         44       1
2024-01 153      150         77       4
2024-02 203       99         81       0
2024-03 172       98         60       2
2024-04 170      211         55       0
2024-05 184       83         69       1
2024-06 124       27         56       0
```

Repository actions

Numbers of unarchivals:

```
    0      1      2      3      4      5      6      7 
23854   2891    582    145     52     17      5      1 
```

Numbers of days between archival and unarchival:

```
     Min. 1st Qu. Median      Mean 3rd Qu. Max.    n
2016    1    2.00   48.0 164.54545  127.00 1110   11
2017    0    6.75   53.5 196.87879  195.75 1949  132
2018    0    7.00   32.5 139.88889  130.00 2096  432
2019    0    6.00   21.0 112.42213   96.50 1935  488
2020    0    7.00   26.0 111.00402   91.00 1523  996
2021    0    8.00   24.0  87.94466   86.00 1025  777
2022    0    8.00   34.0 101.62599  121.00  886 1008
2023    0    7.00   27.0  62.44049   84.00  476  731
2024    0    5.00   15.0  23.54694   30.00  162  245
```

Incoming check actions

Incoming action logs start 2018-05-23, Since then:

Number of actions: 324264 in total (145.87 per day)

Number of submissions: 171630 in total (77.21 per day)

Number of actions according to auto or manual, variant one:

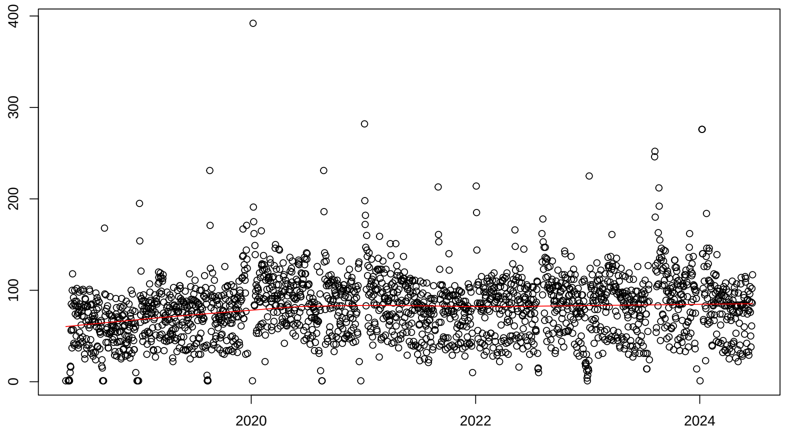
```
  auto manual
196978  99415
```

Number of actions according to auto or manual, variant two:

```
     auto manual_NN manual_UL
   196978     33434     65981
```

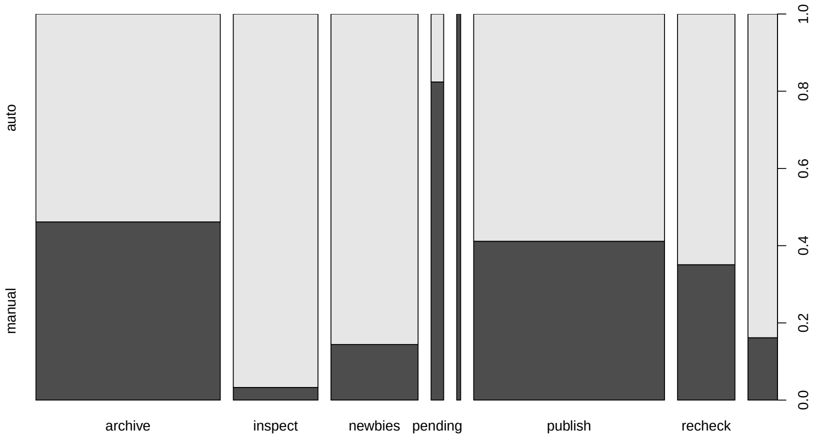
Incoming check actions

Daily numbers of submissions:



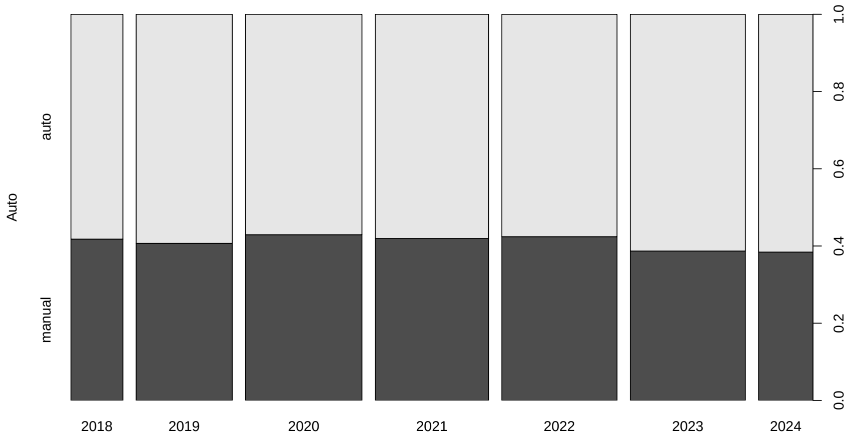
Incoming check actions

Auto-processing rates according to action:



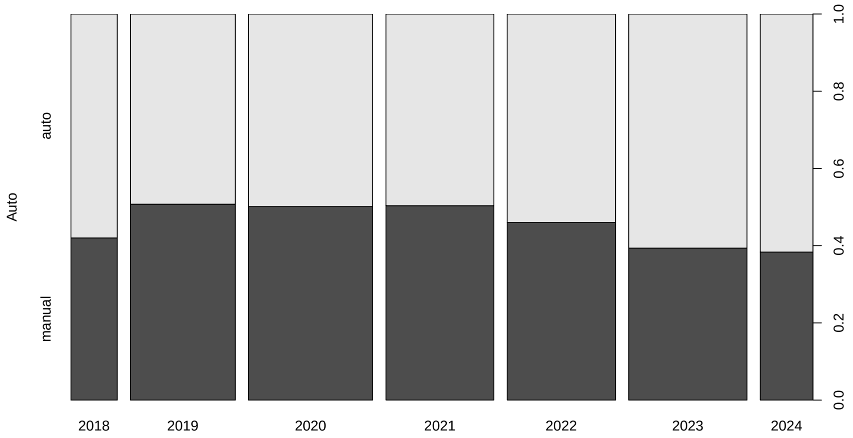
Incoming check actions

Auto-processing rates for action `publish` according to year:



Incoming check actions

Auto-processing rates for action `archive` according to year:



Deadlines

The usual deadline is 2 weeks (some “lesser” issues get one month).

Clearly, 2 weeks is very short: even active maintainers may be away and offline for 2 weeks.

So why not give more time? Some observations:

- Simple problems can be fixed very quickly. Longer deadlines typically result in delaying the trivial fix.

Deadlines

The usual deadline is 2 weeks (some “lesser” issues get one month).

Clearly, 2 weeks is very short: even active maintainers may be away and offline for 2 weeks.

So why not give more time? Some observations:

- Simple problems can be fixed very quickly. Longer deadlines typically result in delaying the trivial fix.
- Hard problems should be fixed rather urgently, but typically cannot. (E.g, memory problems detected by valgrind or sanitizers.)

Deadlines

The usual deadline is 2 weeks (some “lesser” issues get one month).

Clearly, 2 weeks is very short: even active maintainers may be away and offline for 2 weeks.

So why not give more time? Some observations:

- Simple problems can be fixed very quickly. Longer deadlines typically result in delaying the trivial fix.
- Hard problems should be fixed rather urgently, but typically cannot. (E.g, memory problems detected by valgrind or sanitizers.)
- Severity (ERROR, WARNING, NOTE) does not necessarily correspond to importance (a test may simply be wrong and hence not be a real problem, but an Rd xref that no longer works is a real problem).

Deadlines

What are the times between reporting issues and getting these fixed?

Numbers of days between report and next publish (may be unrelated):

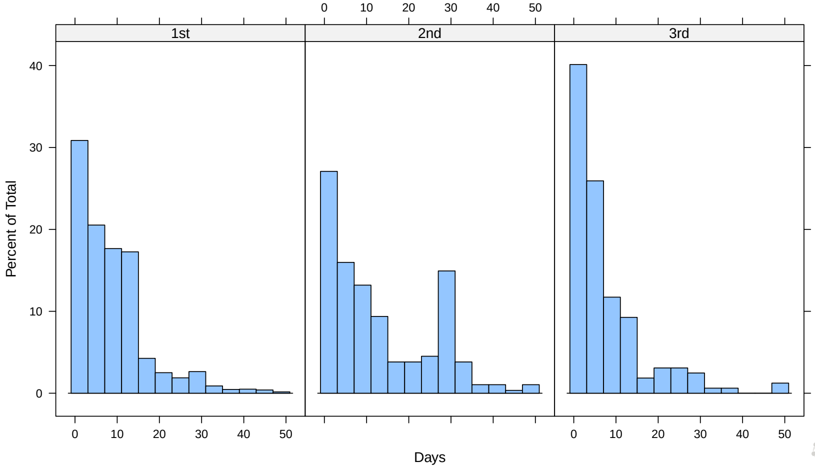
```
    Min. 1st Qu. Median     Mean 3rd Qu. Max.
1st    1       3      8 13.06884      13  370
2nd    1       3     10 15.72203      26  328
3rd    1       2      5 15.79412      12  301
```

Percentage of these days relative to given deadline:

```
    Min. 1st Qu. Median   Mean 3rd Qu.    Max.
1st 2.70   14.29  50.00  83.00   89.47 2371.43
2nd 3.23   14.29  64.29  77.62  100.00 2342.86
3rd 3.33   14.29  35.71 110.75   85.71 2150.00
```

Deadlines

Numbers of days between report and next publish (may be unrelated, hence cut at 50):



CRAN task views

Provide guidance on which packages on CRAN are relevant for tasks related to a certain topic.

Very nice CRAN service provided by the CRAN Task View Initiative (pioneered by Achim Zeileis).

Current number of views: 44 (topics from ActuarialScience to WebTechnologies)

Numbers of packages in views:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  31.00   54.25  105.00  123.95  170.25  394.00 
```

Overall, cover 4574 packages (21.79% of all currently active CRAN packages)

CRAN package DOIs

Starting 2024-06, all current CRAN packages have DOIs.

Registration costs are covered by the R Foundation.

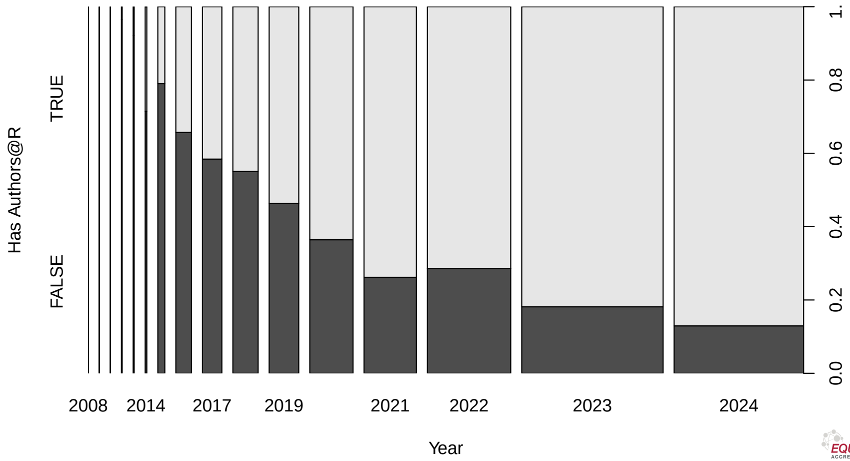
Ideally, registration (with crossref) would provide metadata, including ORCID iDs for authors and DOIs for references.

Ideally maintainers would provide Authors@R.

Ideally maintainers would provide ORCID iDs and DOIs where “possible”.

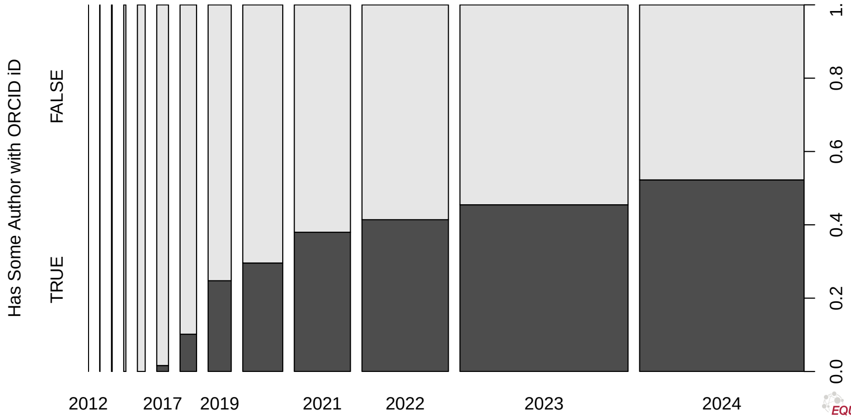
CRAN package DOIs

Percentages of CRAN packages with Authors@R fields (TRUE) according to year of publication:



CRAN package DOIs

Percentages of CRAN packages with Authors@R fields with at least one person with ‘aut’ role and ORCID iD (TRUE), according to year of publication:



CRAN package HTML refmans

R 4.4.0 has added `pkg2HTML()` for creating static HTML package refmans

Want to change CRAN package web pages to use these in preference to the PDF refmans

Work in progress (over the summer?)

Biggest challenge: handling Rd cross-references (xrefs)

Consider `\link{FOO}`. Where should topic FOO be found?

For dynamic help, if not in package itself, try base & recommended packages, then everything else "available" (installed).

Makes some sense, but actually not that much (suppose you have all of CRAN installed).

CRAN package HTML refmans

For static CRAN refmans, we clearly have the topics from *all* CRAN packages available.

Rd xrefs can only *reliably* be resolved if they use package *anchors*:

```
\link[PKG]{FOO}
```

(or `\link[PKG:BAR]{FOO}` if necessary).

So (in due course), all Rd xrefs to topics not in package itself or the base packages should get package anchors.

How much active maintenance will this require?

Note: needs a concerted effort of CRAN *and* Bioconductor.

CRAN package HTML refmans

Total number of Rd xrefs in base and CRAN packages: 1262980

Total number of these Rd xrefs with package anchors: 182135

Where can these packages be found?

```
  base   CRAN   BioC   rems   none 
 42167 139000    888     11     69 
```

Number of CRAN packages with no Rd xrefs: 7170

Number of CRAN packages with Rd xrefs needing package anchors: 2434

(I.e., with at least one Rd xref to topic not in package itself or the base packages and without a package anchor.)

CRAN package HTML refmans

Distribution of numbers of package anchors needed:

```
   1    2    3    4    5    6    7    8    9   10   11   12   13   14 
 699  395  227  172  117  123   73   75   54   47   44   35   21   22 
  17   18   19   20   21   22   23   24   25   26   27   28   29   30 
   9   19   18   13    8   18   10   11    9   12    7    8    1    8 
  33   34   35   36   37   38   39   40   41   42   44   45   46   47 
   5    5    5    1    2    3    6    6    1    2    6    1    3    4 
  53   54   55   56   57   58   59   62   64   66   67   68   69   70 
   3    1    2    2    2    4    2    3    2    1    3    2    1    1 
  76   80   81   84   86   89   90   93   95  101  110  112  118  127 
   2    1    1    1    1    1    2    2    1    1    1    1    2    2 
 132  149  151  155  156  157  160  161  169  182  197  209  217  237 
   1    1    1    1    1    1    1    1    1    1    1    1    1    1 
 426  460  971 1393 
   1    1    1    1 
```

Activate yourself

Keep your maintainer email address up-to-date and working

Activate yourself

Keep your maintainer email address up-to-date and working

Provide Authors@R in your package DESCRIPTION

Activate yourself

Keep your maintainer email address up-to-date and working

Provide Authors@R in your package DESCRIPTION

Provide ORCID iDs for the persons in your Authors@R and inst/CITATION.

Activate yourself

Keep your maintainer email address up-to-date and working

Provide Authors@R in your package DESCRIPTION

Provide ORCID iDs for the persons in your Authors@R and inst/CITATION.

Provide DOIs where “possible”

Activate yourself

Keep your maintainer email address up-to-date and working

Provide Authors@R in your package DESCRIPTION

Provide ORCID iDs for the persons in your Authors@R and inst/CITATION.

Provide DOIs where “possible”

Add Rd xref package anchors where now necessary

Activate yourself

Keep your maintainer email address up-to-date and working

Provide Authors@R in your package DESCRIPTION

Provide ORCID iDs for the persons in your Authors@R and inst/CITATION.

Provide DOIs where “possible”

Add Rd xref package anchors where now necessary

Contribute to services (CRAN task views, . . .)

Activate yourself

Keep your maintainer email address up-to-date and working

Provide Authors@R in your package DESCRIPTION

Provide ORCID iDs for the persons in your Authors@R and inst/CITATION.

Provide DOIs where “possible”

Add Rd xref package anchors where now necessary

Contribute to services (CRAN task views, . . .)

Provide services based on CRAN

Activate yourself

Keep your maintainer email address up-to-date and working

Provide Authors@R in your package DESCRIPTION

Provide ORCID iDs for the persons in your Authors@R and inst/CITATION.

Provide DOIs where “possible”

Add Rd xref package anchors where now necessary

Contribute to services (CRAN task views, . . .)

Provide services based on CRAN

Help with core CRAN services

Activate yourself

Keep your maintainer email address up-to-date and working

Provide Authors@R in your package DESCRIPTION

Provide ORCID iDs for the persons in your Authors@R and inst/CITATION.

Provide DOIs where “possible”

Add Rd xref package anchors where now necessary

Contribute to services (CRAN task views, . . .)

Provide services based on CRAN

Help with core CRAN services

Donate to the R Foundation

Coordinates

Kurt Hornik
Institute for Statistics and Mathematics
Department of Finance, Accounting and Statistics
WU Wirtschaftsuniversität Wien
Welthandelsplatz 1, A-1020 Wien

Tel:	+43/1/313-36x4756
Email:	`Kurt.Hornik@wu.ac.at`
WWW:	`https://statmath.wu.ac.at/~hornik`