CRAN Repository Policy

Version $Revision: 6738 $

CRAN Repository Maintainers

Preamble

This document describes the policies in place for the R (`https://www.R-project.org/`) package repository hosted by the Comprehensive R Archive Network (`https://CRAN.R-project.org/`). In what follows, this CRAN package repository (`https://CRAN.R-project.org/web/packages/`) will be referred to as "CRAN".

CRAN is maintained by the efforts of volunteers (the "CRAN team") and the resources of the R Foundation (`https://www.R-project.org/foundation/index.html`) and the employers of those volunteers (WU Wien, TU Dortmund, U Oxford, U Auckland). Having a package distributed by CRAN is subject to a set of policies, and submitting a package (including an update) to CRAN indicates agreement to these policies.

CRAN hosts packages of publication quality and is not a development platform. A package's contribution has to be non-trivial.

Distributing code or documentation is subject to legal requirements, and CRAN operates in many jurisdictions. One of the aims of these policies is to ensure that the distributors meet their legal obligations of diligence without excessive work.

The time of the volunteers is CRAN's most precious resource, and they reserve the right to remove or modify packages on CRAN without notice or explanation (although notification will usually be given).

All correspondence with CRAN must be sent to `CRAN-submissions@R-project.org` (for submissions) or `CRAN@R-project.org` (for published packages) and not to members of the team, in plain text ASCII and not HTML.

Source packages

- The ownership of copyright and intellectual property rights of all components of the package must be clear and unambiguous (including from the authors specification in the `DESCRIPTION` file). Where code is copied (or derived) from the work of others (including from R itself), care must be taken that any copyright/license statements are preserved and authorship is not misrepresented.

  Preferably, an '`Authors@R`' field would be used with '`ctb`' roles for the authors of such code. Alternatively, the '`Author`' field should list these authors as contributors.

  Where copyrights are held by an entity other than the package authors, this should preferably be indicated via '`cph`' roles in the '`Authors@R`' field, or using a '`Copyright`' field (if necessary referring to an `inst/COPYRIGHTS` file).

  Trademarks must be respected.

  ('All components' includes any downloaded at installation or during use.)
- The package's `DESCRIPTION` file must show both the name and email address of a single designated maintainer (a person, not a mailing list). That contact address must be kept up to date, and be usable for information mailed by the CRAN team without any form of filtering, confirmation .... Forwarding mail from the maintainer address increasingly results in confusing non-delivery notifications to the original sender, so is best avoided.

  The maintainer warrants that (s)he is acting on behalf of all credited authors and has their agreement to use their material in the way it is included in the package (or if this is not possible, warrants that it is used in accordance with the license granted by the original author).

  Additional `DESCRIPTION` fields could be used for providing email addresses for contacting the package authors/developers (e.g., '`Contact`'), or a URL for submitting bug reports (e.g., '`BugReports`').

  Citations in the '`Description`' field of the `DESCRIPTION` file should be in author-year style, followed by a DOI or ISBN for published materials, or a URL otherwise. Preferably, the year and identifier would be enclosed, respectively, in parentheses and angle brackets.
- Source packages may not contain any form of binary executable code.
- Source packages under an Open Source license must provide source or something which can easily be converted back to source (e.g., `.rda` files) for all components of the package (including for example PDF documentation, configure files produced by `autoconf`). For Java `.class` and `.jar` files, the sources should be in a top-level `java` directory in the source package (or that directory should explain how they can be obtained).

  Such packages are not permitted to require (e.g., by specifying in '`Depends`', '`Imports`' or '`LinkingTo`' fields) directly or indirectly a package or external software which restricts users or usage.

  The package's license must give the right for CRAN to distribute the package in perpetuity. Any change to a package's license must be highlighted when an update is submitted (for there have been instances of an undocumented license change removing even the right of CRAN to distribute the package).

Packages with licenses not listed at `https://svn.r-project.org/R/trunk/share/licenses/license.db` will generally not be accepted.

- Package authors should make all reasonable efforts to provide cross-platform portable code. Packages will not normally be accepted that do not run on at least two of the major R platforms. Cases for Windows-only packages will be considered, but CRAN may not be the most appropriate place to host them.
- Packages should be named in a way that does not conflict (irrespective of case) with any current or past CRAN package (the Archive area (`https://CRAN.R-project.org/src/contrib/Archive/`) can be consulted), nor any current Bioconductor (`https://www.bioconductor.org/`) package. Package maintainers give the right to use that package name to CRAN when they submit, so the CRAN team may orphan a package and allow another maintainer to take it over. Package names on CRAN are persistent and in general it is not permitted to change a package's name.

  When a new maintainer wishes to take over a package, this should be accompanied by the written agreement of the previous maintainer (unless the package has been formally orphaned).
- Packages on which a CRAN package depends should be available from a standard repository. The strong dependencies (i.e., packages listed in the '`Depends`', '`Imports`' or '`LinkingTo`' fields) should be available from CRAN or the Bioconductor software repository. If any mentioned in '`Suggests`' or '`Enhances`' fields are not from one of these or the Bioconductor annotation and experiment data repositories, where to obtain them at a repository should be specified in an '`Additional_repositories`' field of the `DESCRIPTION` file (as a comma-separated list of repository URLs) or for other means of access, described in the '`Description`' field.

  A package listed in '`Suggests`' or '`Enhances`' should be used conditionally in examples or tests if it cannot straightforwardly be installed on the major R platforms. ('Writing R Extensions' recommends that they are *always* used conditionally.)

  Orphaned CRAN packages should not be strict requirements (in the '`Depends`', '`Imports`' or '`LinkingTo`' fields, including indirectly). They are allowed in '`Suggests`' if used conditionally, although this is discouraged.
- CRAN versions of packages should work with the current CRAN and Bioconductor releases of packages they depend on and not anticipate nor recommend development versions of such packages (or themselves) on other repositories.
- Packages will not normally be removed from CRAN: however, they may be archived, including at the maintainer's request.

  Packages for which `R CMD check` gives an '`ERROR`' when a new R $x.y.0$ version is released will be archived (or in exceptional circumstances updated by the CRAN team) unless the maintainer has set a firm deadline for an upcoming update (and keeps to it).

  Maintainers will be asked to update packages which show any warnings or significant notes, especially at around the time of a new $x.y.0$ release. Packages which are not updated are liable to be archived.
- Packages should be of the minimum necessary size. Reasonable compression should be used for data (not just `.rda` files) and PDF documentation: CRAN will if necessary pass the latter through `qpdf`.

As a general rule, neither data nor documentation should exceed 5MB (which covers several books). A CRAN package is not an appropriate way to distribute course notes, and authors will be asked to trim their documentation to a maximum of 5MB.

Where a large amount of data is required (even after compression), consideration should be given to a separate data-only package which can be updated only rarely (since older versions of packages are archived in perpetuity).

Similar considerations apply to other forms of "data", e.g., `.jar` files.

Source package tarballs should if possible not exceed 10MB. It is much preferred that third-party source software should be included within the package (as e.g. a `vendor.tar.xz` file) than be downloaded at installation: if this requires a larger tarball a modestly increased limit can be requested at submission.

- Checking the package should take as little CPU time as possible, as the CRAN check farm is a very limited resource and there are thousands of packages. Long-running tests and vignette code can be made optional for checking, but do ensure that the checks that are left do exercise all the features of the package.

  If running a package uses multiple threads/cores it must never use more than two simultaneously: the check farm is a shared resource and will typically be running many checks simultaneously.

  Examples should run for no more than a few seconds each: they are intended to exemplify to the would-be user how to use the functions in the package.
- Packages which use Internet resources should fail gracefully with an informative message if the resource is not available or has changed (and not give a check warning nor error).
- (**Using external C/C++/Fortran/other libraries.**) Where a package wishes to make use of a library not written solely for the package, the package installation should first look to see if it is already installed and if so is of a suitable version. In case not, it is desirable to include the library sources in the package and compile them as part of package installation. If the sources are too large, it is acceptable to download them as part of installation, but do ensure that the download is of a fixed version rather than the latest. Only as a last resort and with the agreement of the CRAN team should a package download pre-compiled software. (See also Using Rust (`using_rust.html`).)

  On Windows and macOS static libraries must be used. A separate document, External Libraries for CRAN packages (`external_libs.html`), covers what external libraries are or could be made available.
- The code and examples provided in a package should never do anything which might be regarded as malicious or anti-social. The following are illustrative examples from past experience.
  - Compiled code should never terminate the R process within which it is running. Thus C/C++ calls to `assert`/`abort`/`exit`/`std::terminate`, Fortran calls to `STOP` and so on must be avoided. Nor may R code call `q()`.
  - A package must not tamper with the code already loaded into R: any attempt to change code in the standard and recommended packages which ship with R is prohibited. Altering the namespace of another package should only be done with the agreement of the maintainer of that package.

  - Packages should not write in the user's home filespace (including clipboards), nor anywhere else on the file system apart from the R session's temporary directory (or during installation in the location pointed to by `TMPDIR`: and such usage should be cleaned up). Installing into the system's R installation (e.g., scripts to its `bin` directory) is not allowed.

    Limited exceptions may be allowed in interactive sessions if the package obtains confirmation from the user.

    For R version 4.0 or later (hence a version dependency is required or only conditional use is possible), packages may store user-specific data, configuration and cache files in their respective user directories obtained from `tools::R_user_dir()`, provided that by default sizes are kept as small as possible and the contents are actively managed (including removing outdated material).
  - Packages should not modify the global environment (user's workspace).
  - Packages should not start external software (such as PDF viewers or browsers) during examples or tests unless that specific instance of the software is explicitly closed afterwards.
  - Packages should not send information about the R session to the maintainer's or third-party sites without obtaining confirmation from the user.
  - Packages must not disable the stack-checking mechanism in the R process into which they are loaded.
  - CRAN packages should use only the public API. Hence they should not use entry points not declared as API in installed headers nor `.Internal()` nor `.Call()` etc calls to base packages. Also, `:::` should not be used to access undocumented/internal objects in base packages (nor should other means of access be employed). Such usages can cause packages to break at any time, even in patched versions of R.
  - Packages should not attempt to disable compiler diagnostics, nor to remove other diagnostic information such as symbols in shared objects.
  - Security provisions must not be circumvented, for example by not verifying SSL/TLS certificates.
  - Use of external resources such as websites must be kept to a minimum. In particular, 'rate limit' errors for websites (such as HTTP codes 429 and 403) must be avoided (and do bear in mind that other packages may be using the same resource).

- Changes to CRAN packages causing significant disruption to other packages must be agreed with the CRAN maintainers well in advance of any publicity. Introduction of packages providing back-compatibility versions of already available packages is not allowed.
- Downloads of additional software or data as part of package installation or startup should only use *secure* download mechanisms (e.g., '`https`'). For downloads of more than a few MB, ensure that a sufficiently large timeout is set.

Binary packages

Policies for when a (Windows or macOS) binary package will be distributed:

- all its package dependencies on CRAN are available for that platform. Dependencies from other repositories will be installed at CRAN's discretion.
- any external software needed can easily be installed on the build machine for all the sub-architectures: here "easily" includes not depending on specific versions, nor should the installed binary depend on specific versions.
- it passes `R CMD check` without error for all the available sub-architectures, or at CRAN's discretion, for the most important sub-architecture(s).

Binary packages are not accepted from maintainers: CRAN will only host binary packages prepared by those responsible for the binary areas. Their packages are made automatically by batch jobs and can take a day or two to appear on the CRAN master site (maybe longer to reach CRAN mirrors).

Binary packages are built for the current version of R: they may also be built for the last version in the previous series (e.g. R 3.1.3 when R 3.2.x is current) or for R-devel.

Questions about binary packages should be addressed to those responsible for building them: Simon Urbanek (macOS) and Uwe Ligges (Windows); email addresses '*First.Lastname*`@R-project.org`'.

Submission

When submitting a package to CRAN you should use the submission form at `https://CRAN.R-project.org/submit.html` (and not send an email). You will be sent a confirmation email which needs to be accepted.

You can check that the submission was received by looking at `https://CRAN.R-project.org/incoming/`. Submission difficulties (such as non-receipt of the confirmation email) can be discussed with `cran-sysadmin@R-project.org`.

In more detail:

- Uploads must be source tarballs created by `R CMD build` and following the *`PACKAGE_VERSION`*`.tar.gz` naming scheme. This should be done with current R-patched or the current release of R.
- Please ensure that `R CMD check --as-cran` has been run *on the tarball to be uploaded* before submission. This should be done with the current version of R-devel (or if that is not possible and explained in the submission, current R-patched or the current release of R.) For new submissions in particular, please take care to include an informative `Description` field. See `submission_checklist.html` for details.

  In principle, packages must pass `R CMD check` without warnings or significant notes to be admitted to the main CRAN package area. If there are warnings or notes you cannot eliminate (for example because you believe them to be spurious) send an explanatory note as part of your covering email, or as a comment on the submission form.

  For interpretation of the URL checks, see `URL_checks.html`.
- Authors can avoid a lot of the all too frequent rounds of updates by checking carefully for themselves. It should be normal for those without Windows machines of their own to use the winbuilder (`https://win-builder.R-project.org/`) service to check a package before submission. There is a lot of helpful advice on writing portable packages in "Writing R Extensions" (`../../manuals.html#R-exts`).
- For a package update, please check that any packages depending on this one still pass `R CMD check`: it is especially expected that you will have checked your own packages. Reverse dependencies can conveniently be checked using `tools::check_packages_in_dir(reverse = list())`, and changes in check status subsequently be analyzed using `tools::check_packages_in_dir_changes()`. A listing of the reverse dependencies of the current version can be found on the CRAN web page for the package, or be obtained via `tools::package_dependencies(reverse = TRUE)`. If possible, check reverse strong dependencies, reverse suggests and the recursive strong dependencies of these (by `tools::package_dependencies(reverse = TRUE, which = "most", recursive = "strong")`).
- If for some reason the submission has to be made by someone else (for example, a co-author) this needs to be explained, and the designated maintainer will need to confirm the submission.
- Explain any change in the maintainer's email address and if possible send confirmation from the previous address (by a separate email to `CRAN-submissions@R-project.org`) or explain why it is not possible.

If the package needs special treatment (for example if vignettes can only be run or rebuilt on the maintainer's machine or take a very long time), say so on the submission form.

- Do not email the package itself.
- Once uploaded, no further submissions of that package should be made whilst the uploaded version is pending processing (which may take a few days) and you have not received a reply from a CRAN maintainer.

Re-submission

Re-submission is done in the same way as submission, using the 'Optional comment' field on the web form (and not a separate email) to explain how the feedback on previous submission(s) has been addressed.

Updates to previously-published packages must have an increased version. Increasing the version number at each submission reduces confusion so is preferred even when a previous submission was not accepted.

In more detail:

- Submitting updates should be done responsibly and with respect for the volunteers' time. Once a package is established (which may take several rounds), "no more than every 1–2 months" seems appropriate.
- Before submitting a package update, consult the CRAN check page at '`https://CRAN.R-project.org/web/checks/check_results_NAME.html`', substituting NAME by the name of your package. In particular, wait for that page to be fully updated after publication of a version (which can take at least 48 hours) before submitting any corrections.

  For packages which have been archived since February 2018, a snapshot of the CRAN results page at the time of archival will be available under `https://cran-archive.r-project.org/web/checks/`. (Note that only a few of the links from the snapshot will work: normally those to listed 'Additional issues' will.)

  A package showing issues for `macos-arm64` or an 'M1mac' additional issue should be checked using the macbuilder (`https://mac.r-project.org/macbuilder/submit.html`) service prior to re-submission.
- If an update will change the package's API and hence affect packages depending on it, it is expected that you will contact the maintainers of affected packages and suggest changes, and give them time (at least 2 weeks, ideally more) to prepare updates before submitting your updated package. Do mention in the submission email which packages are affected and that their maintainers have been informed. In order to derive the reverse dependencies of a package including the addresses of maintainers who have to be notified upon changes, the function reverse_dependencies_with_maintainers (`https://developer.R-project.org/CRAN/Scripts/depends.R`) is available from the developer website.