

CRAN Repository Policy

(Draft version), Version \$Revision: 2678 \$

CRAN Repository Maintainers

Preamble

This document describes the policies in place for the **R** package repository hosted by the **Comprehensive R Archive Network**. In what follows, this **CRAN package repository** will be referred to as “CRAN”.

CRAN is maintained by the efforts of volunteers (the “CRAN team”) and the resources of the **R Foundation** and the employers of those volunteers (WU Wien, TU Dortmund, U Oxford, AT&T Research). 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.

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).

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 statements are preserved and authorship is not misrepresented.

Trademarks must be respected.

- The package’s ‘DESCRIPTION’ file must show the name and email address of a single designated maintainer (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 . . .

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).

- 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 and configure files produced by `autoconf`).

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).

- 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 can be consulted), nor any current **Bioconductor** 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.

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 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.

- 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.

- 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`, 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 users’ home filespace, 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.
 - 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.

Binary packages

Policies for when a (Windows or OS X) 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: 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.

Submission

When submitting a package to CRAN:

- Uploads must be source tarballs as created by `R CMD build` and follow the ‘`PACKAGE_VERSION.tar.gz`’ naming scheme.
- Please ensure that `R CMD check` has been run *on the tarball to be uploaded* before submission. This should be done with the current release of R or (preferably) R-devel or R-patched. As “Writing R Extensions” says

Please ensure that you can run through the complete procedure with only warnings that you understand and have reasons not to eliminate. In principle, packages must pass `R CMD check` without warnings to be admitted to the main CRAN package area. If there are warnings you cannot eliminate (for example because you believe them to be spurious) send an explanatory note as part of your covering email.

- 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. A listing of the reverse dependencies of the current version can be found on the CRAN web page for the package.
- A submission should be accompanied by an email to CRAN@R-project.org, if possible sent from the maintainer address listed in the package, and using the subject line ‘CRAN submission `PACKAGE VERSION`’, where `PACKAGE` and `VERSION` are the package name and version, respectively.

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.

- Once uploaded, no further submissions of that package should be made whilst the uploaded version is pending processing (which may take a few days).
- Part of the processing is that uploads may be renamed by adding one of the extensions ‘`.save`’, ‘`.pending`’ or ‘`.noemail`’: the presence of such a file is a sign that a submission is still pending (and such a file name should never be uploaded).
- 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.

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](#) service to check a package before submission. There is a lot of helpful advice on writing portable packages in “Writing R Extensions”.

- 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 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` is available from the developer website.