CollapsABEL package review

L.C. Karssen

<2015-05-22 vr>

Contents

1 Introduction

One of the goals of the GenABEL project is to be an environment in which people can work on the implementation of statistical methodology into user-friendly software packages.

In order for the project to be sustainable the packages that are accepted into the GenABEL suite must meet certain standards. Without certain standards/guidelines package maintenance will be difficult and time consuming. Moreover, if the user interface is awkward of if the package lacks documentation users will be less likely to use the package. In short these standard revolve around the following:

- *Maintainability of the package*: is the code understandable? Does it follow our coding standards? Is the code documented?
- *User-friendliness*: What is the quality of the user documentation? Are there any examples? Is the user interface compatible with what is to be expected?

One of the ways to ensure a healthy ecosystem is to have reviews of candidate packages. Such a review would be similar to the peer review done for scientific publications. In order to have good quality package reviews we have put together this document describing in a structured way the minimum questions a package reviewer should ask.

Please consider this document a working draft. Feedback is very much welcomed.

Note that our coding style closely follows the Google style guide and is documented here.

2 Legal issues

2.1 Is the copyright holder clearly mentioned?

Yes, the Author field in the DESCRIPTION file lists two authors. According to "Writing R Extensions" (http://cran.r-project.org/doc/manuals/R-exts.html#The-DESCRIPTION-file, hereafter Rexts) the author is assumed to hold the copyright).

2.2 Is there a clear (standard) license?

Yes, the DESCRIPTION file lists the GPL licence. I would advise to add a version number to it, e.g. "GPL (>=v2)", or >=v3, at the authors' discretion).

2.3 Is the license GNU GPL-compatible?

Yes.

3 Technical quality

3.1 Is the installation procedure clearly documented? Is the code easy to compile and run?

This is an R package, so in principle installation should as simple as running `install.packages(file.tar.gz)` and would be even easier once accepted into CRAN (because then the source code is downloaded automatically).

However, this package depends on many non-R libraries and tools. It is therefore mandatory to read the Installation section of the tutorial before trying to install the package. The authors provide a clearly documented Installation section and have done a commendable job by creating a `install_tools.sh` script that tries to install the dependencies automatically. It should be noted explicitly, however, that superuser privileges are required in order to install the necessary dependencies.

I also recommend to add the following options at the top of the script to ensure the script aborts on errors (otherwise the user might think all went well, or simply miss an error message in the rest of the output):

```
set -e
set -u
```

3.2 [For R packages] Does the package pass CRAN checks (http://cran.r-project.org/doc/manuals/r-devel/R-exts.html#Checking-packages)? At minimum, run "R CMD check ..." and "R CMD check –as-cran ..."

At the time of submission for the review the author informed us on the devel mailing list that currently a warning exists when running, which would be fixed. When running CRAN checks at the time of review, a few warnings/notes remain:

```
 NOTE
Maintainer: ‘Kaiyin Zhong <kindlychung@gmail.com>’

Unknown, possibly mis-spelled, fields in DESCRIPTION:
  ‘Citation’

* checking top-level files ... NOTE
Non-standard files/directories found at top level:
  ‘bk_readme.md’ ‘CollapsABEL.Rproj’ ‘generate_manual.sh’
  ‘install_tools.sh’ ‘install_tools.sh~’ ‘Rd2.pdf’ ‘readme.md.bk’

* checking for missing documentation entries ... WARNING
Undocumented code objects:
  ‘rep_df’
All user-level objects in a package should have documentation entries.
See chapter ‘Writing R documentation files’ in the ‘Writing R
Extensions’ manual.
* checking for code/documentation mismatches ... WARNING
Functions or methods with usage in documentation object 'rep_dat' but not in code:
  rep_dat
```

3.3 Is the package documented? What is the quality of the documentation?

Yes, the package has proper .Rd documentation based on Roxygen2 comments in the code.

However, for some functions like `collapse()` and `collapse_with_id()` the Roxygen2 documentation points to URLs on http://gist.github.com. I don't consider this a good situation. The documentation should be completely within the package. If the only documentation for a function is a link to another site it may very easily lead to missing documentation (in this case when e.g. the user resigns from Github or GitHub is closed).

3.4 [For R packages] Does `help(PackageName)` provide an adequate summary of the package and a review of the major functions?

No. This part of the documentation is missing. See also the next section.

3.5 [For R packages] Does the package use Roxygen2 for documentation?

Yes.

- However, in several cases the roxygen documentation points to github URLs for more information. This is an undesirable solution because package and documentation are disconnected and can (be) change(d) independently. Moreover, if e.g. github decides to discontinue their gist service or charge money for it, the documentation will/may be lost.
- The `importFrom` and `export` lines in the `NAMESPACE` file are not generated by roxygen. See e.g. `http://kbroman.org/pkg_primer/pages/docs.html`. Then the command `devtools::document()` can be used to automatically generate all documentation from the roxygen source. This is done in a file called `CollapsABEL.R` (see e.g. the section *Documenting packages* at `http://cran.r-project.org/web/packages/roxygen2/vignettes/rd.html`)
- The same =CollapsABEL.R file can be used to generate some general information on the package (see previous URL for an example).

3.6 Are examples of usage provided?

No, not within the help of the functions.

3.7 Does the package provide a tutorial/vignette? Can you comment on the tutorial?

Yes, an extensive tutorial has been provided.

3.8 Is the source code of the tutorial/vignette provided?

Yes, as a MarkDown file (one of the `.md` files that the `--as-cran` checks complains about).

3.9 Does the package make use of unit/integration/etc. tests?

Yes, via the `testthat` package. I haven't checked which part of the code is (not) covered by these tests.

3.10 [For R packages] Does the package make use of RUnit testing?

No, but `testthat` checks are provided for some functions.

3.11 Does the code comply with the GenABEL coding standards?

Mostly.

- Indentation is sometimes done with two spaces, sometimes four, sometimes tabs. Sometimes mixes of these within one file or even function.
- Helpful comments in the code.

3.12 Is the code readable/understandable?

Yes.

3.13 Does the code contain explanatory comments?

Yes.

3.14 Were the design and methods implemented in package discussed during the development process (e.g. on the genabel-devel mailing list)?

No.

4 Content

4.1 Does the package address a problem in the domain of statistical genomics?

Yes. The package provides functions for the (generalised) compound double heterozigosity test.

4.2 Is it streamlining analyses not covered elsewhere in the GenABEL suite? If not, does it improve the analysis already covered?

This package is an extension/upgrade of the compound heterozygosity functionality already present in the GenABEL package.

An important question that should be discussed on the mailing list is whether the existing functions should be removed in favour of this package.

4.3 Should it become a separate package or rather be incorporated into an existing package?

Given the size of the package and the fact that it doesn't use existing packages from the GenABEL suite I recommend to keep this a separate package.

4.4 Does the package use any of the data types defined in other GenABEL packages?

No. It does not depend on other packages in the GenABEL suite.

4.5 Does the package use code/functions/data defined in other GenABEL packages?

No, see above.

5 Recommendations

This package shows considerable effort and is in a very good state. It also implements useful statistical genetics methods.

5.1 What are the major issues that should be addressed?

5.1.1 General

- The package depends on many non-R libraries. The `install_tools.sh` script downloads several of these, but they should nonetheless be mentioned in the SystemRequirements field of the `DESCRIPTION` file (see e.g. the curl package or the XML package on CRAN for an example).
- The Description field in the `DESCRIPTION` file is non-informative.
- The DESCRIPTION file contains a `Citation` field. This field isn't mentions in the "Writing R Extensions manual" (Rexts), nor in the Debian Policy that it links to. A separate CITATION file is mentioned in Rexts. Running `R CMD check --as-cran` flags this as `NOTE` (which CRAN maintainers don't like to see).
- Running `R CMD check --as-cran` with the development version of R (`2015-03-31 r68131`) lead to some NOTES and WARNINGs (see above). These need to be fixed.

5.1.2 Installation documentation

- The "Simple way" section should mention explicitly that superuser privileges are required.
- The `XML` R package requires the xml2-config binary (On Debian and Ubuntu this can be found in the libxml2-dev package). This should be mentioned.
- The `git2r` R package required by `devtools` needs openSSL development files (Debian/Ubuntu package `libssl-dev`). This would be good to mention.
- The `wget` part in the simple install command is incomplete as the downloaded file is called `1uJInXK`. The command should be `wget http://bit.ly/1uJInXK -O install_tools.sh`.
- The `install_tools.sh` script should test for the existence of the `bedcoll` and `plink2` binaries before overwriting them. Otherwise the user's existing `/usr/local/bin/plink` binary maybe overwritten without their knowledge, for example.
- The user should be able to specify an installation directory on the command line for the Armadillo, bedcoll and plink2. Now the last two are installed in `/usr/local/bin/` which may not be desirable. Moreover, as a system administrator I wouldn't mind installing some `.deb` or `.rpm` packages, but I wouldn't like a tool that dumps stuff in directories like `/usr/local` that isn't packaged properly. This may work on a single server, but would be problematic on a cluster, for example. Giving the sysadmin or user without root privileges an option to install these 'external' dependencies in another directory is very important.
- The URL for `plink2` in the `install_tools.sh` script is invalid (the date in the directory has changed). A more stable approach is needed. Maybe use `wget` or `curl` to parse the https://www.cog-genomics.org/plink2/ web page for the current stable URL? Or, better and more simple from a maintenance point of view: make it a SystemRequirement in the `DESCRIPTION` file and if the `plink` or `plink2` binary can't be found simply print an error message from the R functions that use it. Then installation of these tools is left to the user (or his/her sysadmin).

5.1.3 Documentation

- Add the documentation for all functions to the Roxygen2 comments instead of using links to gists.github.com.
- Roxygen documentation for the `NAMESPACE` file is missing. I recommend to fix that so that `devtools::document()` can be used to generate all documentation. The file `collapsABEL.R` should be created for this purpose. This file should also contain some general introduction/help for the manual and `help(collapsABEL)`.

5.2 What other (optional) suggestions do you have for the author?

- The "Install R packages" subsection of the tutorial uses `require(devtools)`. `require()` should only be used inside functions, it is better to use `library()` instead.
- CollapsABEL depends on several other R packages by the same author (`txtutils`, `manqq2`, `rbed2`), which have to be installed from the author's BitBucket repository. Ideally these packages would be available on CRAN to make for a smoother install process.
- The roxygen documentation of the functions could use some examples so that the user doesn't need to go and dig up the tutorial in order to get a quick idea of how the function works.
- A bit more consistency in the indentation style would be great (sometimes two spaces, sometimes four, other times a TAB).