Notes for Committing RSiena to R-Forge

Tom Snijders

February 5, 2020

This is a short set of guidelines for committing the RSiena and RSienaTest packages to R-Forge. It is an excerpt of the *Notes for Developers of RSiena* (file `RSienaDeveloper`). More complete information can be found there. Both of these files are in the directory `RSienaTest\doc` in the package source.

It is wise to have a look at *Notes for Developers of RSiena*. For anything in this document that you don't understand, it may be illuminating. Further, its section on personal/private keys was useful to me.

1 Basics

The connection to R-Forge is made by TortoiseSVN. To check and compile the package, you need the most recent version of R and RTools. Note that RTools must be in the path of your computer.

2 The code; correspondence between packages

Notes for Developers of RSiena mentions coding standards for R as well as C++. Try to respect them. Add comments in your code and use helpful names (in camel case).

Keep the line endings in all `.tex` files intact (i.e., do not let an editor take away the hard returns). This is helpful for file comparisons.

The correspondence between RSiena and RSienaTest should be kept intact:

1. Both packages should have the same version number and `ChangeLog`; this means that if (e.g.) you commit a new version of RSienaTest only, you also need to change the version number and `ChangeLog` of RSiena.
2. The moment when things are moved from RSienaTest to RSiena is up to the programmers. Some things, certainly error corrections, should be changed in both simultaneously. When we are unsure about a change in the code, it may be better to first make it in RSienaTest and then decide later, after having obtained more experience, about moving it to RSiena.
3. For aligning files in RSiena with those in RSienaTest I make use (on my Windows machine) of WinMerge (it's free and useful).

The internal structure of both packages should remain intact:

1. Change the version number in the `DESCRIPTION` file and in `\man\RSiena-package.Rd`; add your changes to the top of the `ChangeLog` and, in so far as they are noticeable to users, to the top of appendix B in the manual.

2. When new effects are added, these must be added to Chapter 12 of the manual. The manual is in RSienaTest only.

3. Also new other functionalities, if any, should be added to the manual, unless they are clear enough from the help pages.

4. The manual states the version to which it applies. The best way to keep this up to date is to search in RSiena_Manual.tex for the occurrence of the last/current version number (e.g., 1.2-16) and replace appropriately.

To keep things easy for users of different platforms, when you add a file to the system, in TortoiseSVN set the property **`svn:eoln-style`** to "native". (This is not necessary for .pdf files.) This will avoid the all-too-usual problems when transferring files between Windows and Mac systems.
Recipe: right click on the file name → TortoiseSVN → Properties → New → EOL → Platform-dependent (native).
Remember to do this also for identical files that you create for the second package (e.g., for RSiena if first you added the file to RSienaTest); just copying the files does not copy the Tortoise file properties.

Preferably limit line lengths to a maximum of 80 characters.

3 Checking

It will be much appreciated when you update the description files in the doc directory, or add new files, in cases where you made changes that are important enough to deserve such documentation.

Before committing, it is mandatory to either run
`R CMD check RSiena`
, or do a build, and then run
`R CMD check RSiena_1.1-xxx.tar.gz`
; do the same for RSienaTest. Use the most recent version of R for this (note that successive versions and subversions of R often increase the stringency of checking). Note that the C++ compiler may be different on different systems (e.g., Windows and Mac) which may lead to different warnings.

Do this also for the most recent patched version of R, because R-Forge will do this and often it leads to additional errors and notes, because R becomes progressively stringent.

There is a facility to check Windows packages on CRAN. See
`http://win-builder.r-project.org/`; use the upload page!

A facility to build and check packages on various platforms is at
`https://builder.r-hub.io`; in the 'advanced' page there is a variety of platforms on which you can check.

In case R-hub throws unexplainable problems, you might consult the R-hub issue tracker:
`https://github.com/r-hub/rhub/issues`

Try to make the code so that notes and warnings from this Check are avoided as far as possible, and that you understand those that remain. There should of course be no errors at all. For example, I always get a note for RSienaTest that the package size is big (because the manual is a very big file and the C++ code is big) and often about some superfluous pieces of code (because I want to use them later).

Checks consist of running the examples in the help files (unless excluded by `dontrun`) and in `\tests`, except those that are excluded in the file `.Rbuildignore`. This currently leaves only `\tests\parallel.R`. Sometimes the file `\tests\parallel.Rout.save` must be updated.

It may be useful to add other scripts for checking (but they should not consume much time).

4 Committing

Essential reading is *Writing R extensions*, `https://cran.r-project.org/doc/manuals/r-release/R-exts.html#Checking-packages` (exists also in an *r-devel* version!).

CRAN policies are not constant; see, e.g., Hornik, Ligges and Zeileis. "Changes on CRAN: 2017-12-01 to 2018-06-30", R Journal 10(1), July 2018. `https://journal.r-project.org/archive/2018-1/cran.pdf`

The queue at CRAN is shown in the "CRAN incoming dashboard", `https://cransays.itsalocke.com/articles/dashboard.html`