Systematic Review of Versioned Files for Modeling and Simulation Quality Control: The "review" R Package

Timothy Bergsma, Jeannine Fisher, Marc Gastonguay, Jeffrey Hane, William Knebel, Matthew Riggs, James Rogers

Metrum Research Group, Tariffville, CT

Objectives

To create systematic mechanisms for (1) documenting the review of version-controlled files in a production modeling and simulation environment and for (2) analyzing review documentation to support administrative decisions for regulatory compliance of modeling and simulation processes.

Methods

Software was implemented as the package `review` (current version 2.4.1): a publicly available [1] extension of the R programming language [2]. Functions were designed to support Subversion [3]: an open source version control system. Scope of operations was designed to be limited to a directory checkout and all its descendants. A name, format, and location were chosen for a review log to capture information critical to the review process. Primary functions were developed to register acceptance of the current versions of particular files, and to detect the need for further review due to file revision. Secondary functions were developed to support assignment of review tasks within the constraints of the review log format. Integrated use of review with Subversion was documented. Support was added for calculating version status for files generated by other files.

Results

In the checkout directory, the function `logCreate` creates `QClog.csv` with columns `file`, `origin`, `revf`, `revo`, `reviewer`, and `time`.

```
library(review)
```
```
review 2.4.1 loaded
```

```
writeLines(system(intern = TRUE, "svn co file:///repo demo"))
```
```
A    demo/figure.pdf
A    demo/code1.R
A    demo/code2.R
Checked out revision 1.
```

```
setwd("demo")
```

```
dir()
```
```
[1] "code1.R"    "code2.R"    "figure.pdf"
```

```
logCreate()
```

```
dir()
```
```
[1] "QClog.csv"  "code1.R"    "code2.R"    "figure.pdf"
```

The user adds `QClog.csv` to the repository using Subversion techniques. Each row in `QClog.csv` documents acceptance of a particular revision of a particular file in the checkout directory or a subdirectory. The user generates new rows with the function `logAccept`, specifying file names. Optionally, the user may specify `origin`, which names a file responsible for the contents of `file` (e.g,textttorigin is a script that generates the data table `file`; if not specified, `origin` defaults to `file`.). The field `reviewer` is populated with the R login of the current user (`Sys.info()["login"]`). The field `time` is populated with the current system time expressed as GMT.

```
logRead()
```
```
[1] file     origin   revf     revo     reviewer time
<0 rows> (or 0-length row.names)
```

```
logAccept("code1.R")
```

```
logAccept("figure.pdf", origin = "code1.R")
```

```
logRead()
```
```
        file  origin revf revo reviewer                 time
1    code1.R code1.R    1    1     timb 2011-03-17 12:46:58 GMT
2 figure.pdf code1.R    1    1     timb 2011-03-17 12:46:58 GMT
```

The function `logAssign` creates a record accepting revision zero of a particular file on behalf of a specified reviewer. Since revision zero of a file never exists, the idiom serves as a convenient and unambiguous mechanism to request or assign review of a particular file in a multi-user environment; it is also an opportunity to register a non-default value of `origin`, if any. After `QClog.csv` is checked in to the Subversion repository, other users may check it out and detect user-specific assignments with `logAssignments`.

```
logAssign("code2.R")
```

```
logRead()
```
```
        file  origin revf revo reviewer                 time
1    code1.R code1.R    1    1     timb 2011-03-17 12:46:58 GMT
2 figure.pdf code1.R    1    1     timb 2011-03-17 12:46:58 GMT
3    code2.R code2.R    0    0   anyone 2011-03-17 12:46:58 GMT
```

```
logAssignments(reviewer = "anyone")
```
```
[1] "code2.R"
```

```
system("svn ci -m assignments")
```

An arbitrary number of acceptances can be logged for a particular file. The function `logSummary` sorts the results of `logRead` by `file`, `revf`, and `time`, preserving only the last record per file; it also uses the Subversion interface to calculate the checkout's head revision number for the file (`headf`) and its origin (`heado`). The function `logPending` further reduces the remaining records to those that are out of date: i.e., the number of the head revision is greater than that of the reviewed revision, for either the file or its origin. This is always true for files that have been assigned but never reviewed, since all extant files have revisions greater than zero.

```
logSummary()
```
```
        file  origin revf headf revo heado reviewer                 time
1    code1.R            1     1                timb 2011-03-17 12:46:58 GMT
3    code2.R            0     1              anyone 2011-03-17 12:46:58 GMT
2 figure.pdf code1.R    1     1    1     1     timb 2011-03-17 12:46:58 GMT
```

```
logPending()
```
```
     file revf headf reviewer                 time
3 code2.R    0     1   anyone 2011-03-17 12:46:58 GMT
```

Typically, an author creates a log within a Subversion checkout (`logCreate`) and then requests review (`logAssign`) of critical files. The review log grows as peer reviewers commit acceptance records (logAccept) by means of the Subversion interface. Content issues are negotiated as necessary between the author and reviewer until a version of the file exists that the reviewer can accept.

```
logAccept("code2.R")
```

```
system("svn ci -m reviewed")
```

The author updates his checkout to monitor progress. An empty table returned by `logPending` supports the administrative decision that review is complete. Subsequent alterations to reviewed files cause them to pend once again. Reviewers may take advantage of previous effort by focusing on the differences between previously accepted versions and pending versions (readily tabulated by the `diff` utility of the Subversion interface).

```
logPending()
```
```
[1] file     revf     headf    reviewer time
<0 rows> (or 0-length row.names)
```

```
system("echo new code >> code1.R")
```

```
system("svn ci -m changes")
```

```
logPending()
```
```
        file  origin revf headf revo heado reviewer                 time
1    code1.R            1     2                timb 2011-03-17 12:46:58 GMT
2 figure.pdf code1.R    1     1    1     2     timb 2011-03-17 12:46:58 GMT
```

```
logAccept("code1.R")
```

```
logPending()
```
```
        file  origin revf headf revo heado reviewer                 time
2 figure.pdf code1.R    1     1    1     2     timb 2011-03-17 12:46:58 GMT
```

The `review` package was written in 2008. Version 2.4 has been in production use since January 2010.

Conclusions

Independent review of critical files is a good general practice for verifying correctness of pharmacometric analyses. Tracking review activity poses characteristic challenges for documents that are created and maintained electronically. Regulatory guidance for the pharmaceutical industry emphasizes capture of unambiguous metadata regarding management of electronic records, such as time, agent, activity, and object version [4]. The `review` package for the R programming language integrates with Subversion to address these needs: each captured record is explicitly an act of administrative acceptance, where time of acceptance is captured in GMT, agent is captured as the user's login, and version identifier is supplied by the Subversion interface. Further, the system hosting Subversion unambiguously identifies the agent modifying the log, as noted elsewhere [5]. In 21CFR11 parlance [4], `review` records have many of the properties of electronic signatures and serve a similar role. If one chooses to think of `QClog.csv` itself as an electronic record, then use of Subversion meets FDA's definition of audit trail [6].

The `review` package provides mechanisms both for documenting review activity and for analyzing the resulting documentation. Functions `logCreate`, `logAssign`, and `logAccept` create and modify `QClog.csv` interactively. Functions `logRead`, `logSummary`, and `logPending` render `QClog.csv` in various formats. In particular, `logPending` reports files that were selected for review but have not been reviewed yet, as well as files that have been revised since last review. The review package relies on Subversion software to distribute changes among users and to track file version information.

References

[1] http://cran.r-project.org/web/packages/review

[2] http://www.r-project.org

[3] http://subversion.apache.org

[4] 21CFR11 Sec. 11.10, 11.50. http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfCFR/CFRSearch.cfm?CFRPart=11&showFR=1

[5] R: regulatory compliance and validation issues. A guidance document for the use of R in regulated clinical trial environments. Technical report, The R Foundation for Statistical Computing (2008).

[6] http://www.fda.gov/OHRMS/DOCKETS/98fr/04d-0440-gdl0002.pdf

 Copies of this poster are available at www.metrumrg.com/publications ACoP 2011, San Diego, CA, April 3–6, 2011