**PhUSE 2010**

**Paper RG03**

# How much is it? Validation of Open-Source-Software Using the example of R

Peter Bewerunge, HMS Analytical Software, Heidelberg, Germany

## ABSTRACT

This paper discusses the use of R[1] as an Open Source Software (OSS) in the validated environment. After an introduction to principle aspects of integrating R into processes that have to be validated, the conformance to the terms Installation Qualification (IQ) and Operational Qualification (OQ) is addressed. Also the integration of R into other software systems (Java or C# applications) is described.

## INTRODUCTION

Validation is a key element during the whole software life cycle process. Its purpose is to build quality into a software system and to ensure its reliability. For this reason, it is understandable that companies rely on well-established commercial software products.

High license fees and the call for flexible ad hoc analyses, especially in research departments, are just two arguments that lead to the use of OSS. For this reason, the software environment R for statistical computing and graphics has made its way into pharmaceutical companies. However, once particular regulations are required, e.g. in the context of a FDA submission, the code must be re-implemented for example with C++ or C#. This is cost and time intensive.

One would be lucky to pass down R code developed in a research department to a validated environment. In general, I believe it is a matter of could one find arguments to trust an OSS or not. Here one of the key feature of OSS that the code is visible to everyone can help to make this decision. The possibility of critical code reviews, evaluation of risks and re-engineering of flawed or missing code may make this a suitable alternative to expensive commercial software.

This paper focuses on general strategies to extend trust in the OSS R and in particular attempts to bring R closer to the terms Installation Qualification, Operational Qualification and calling R from other software systems (e.g. Java-, C++ or C#-applications).

## GENERAL PROS AND CONS OF OPEN SOURCE SOFTWARE

Open Source in the context of Software means, that the utilized program code is available to everyone. This allows for code development in a public collaborative manner. Here are some general pros and cons about OSS:

**PROS**

- the source code is available to all and can be modified
- many open source projects are free of charge
- no need to start from scratch, but rather use existing open source libraries
- the amount of widely spread developers increases the probability to identify bugs
- generally, a community of developers are able to provide support
- makes license management easier

**CONS**

- no support hotline or customer service for urgent requests
- often do not focus on user interfaces and lacks good documentation (not user friendly)
- open source libraries focus on the developer issues, and not on those of the users

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

- often no information is available about which module has been tested and how much of the code is covered (That is not true for R. Here, the test files are visible for everyone too.)

## R UNDER THE GENERAL PUBLIC LICENCE (GNU)

Although the open source idea appears to be gratuitous, this is not the case in reality. Developers can and do charge for OSS. R is available free of charge under the GNU General Public License, which intends to guarantee the 4 freedoms:

a) everybody is allowed to use the software for any purpose
b) copies of the source code can be made free of charge or for a fee
c) the source code is open to everybody and may be modified
d) it is allowed to distribute the modified code for a fee, but the source code must be open for the customer

For more detailed information, go to the website of the GNU GPL[2].

## PRINCIPLE ASPECTS OF USING R IN A VALIDATED ENVIRONMENT

People often claim that they cannot use R, because it is not validated. But how can one be sure, that for example, Microsoft Excel or SAS® Analytics is validated? This raises the question, what should actually be validated: The software or the process the software is used in?

In most cases, great importance is attached to the validation of a process. But to be honest, all software that is used in a validated environment also needs a minimum validation. This has become a sensible issue especially for OSS. Even R has to pass the Installation Qualification (IQ), Operational Qualification (OQ) and Performance Qualification (PQ). Potentials and risks concerning this will be discussed later in this article.

Other arguments such as the FDA not allowing the use of R cannot be substantiated in any FDA guidance document. The FDA gives suggestions but does not prescribe at any time a specific software product. To quote Mat Soukup, Acting Team Lead at the FDA: “Regulations do NOT prevent use of R” (Figure 1).

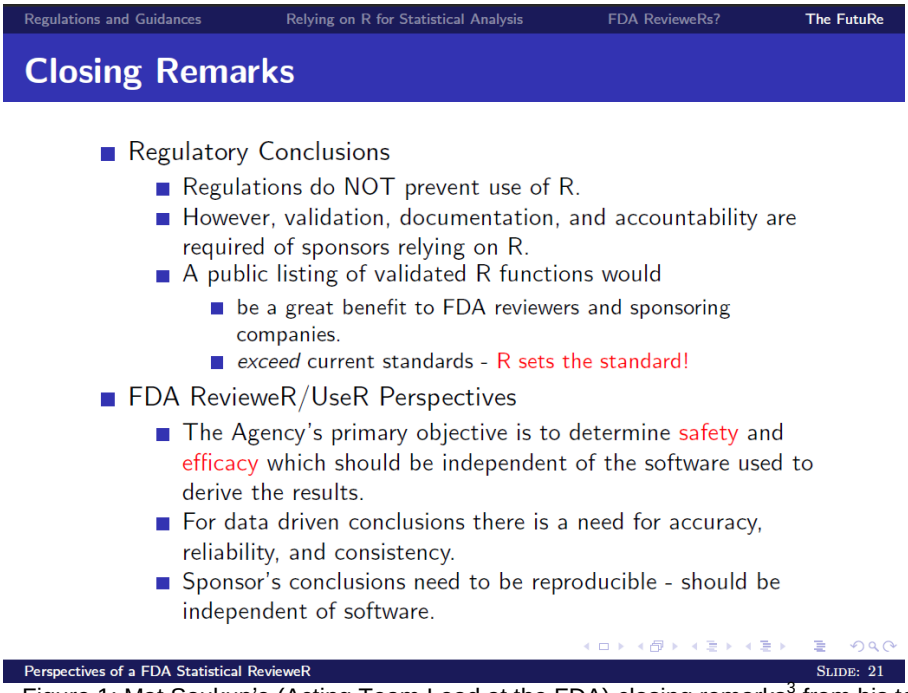



Figure 1: Mat Soukup’s (Acting Team Lead at the FDA) closing remarks[3] from his talk “Using R: Perspectives of a FDA Statistical RevieweR” at the useR-Conference 2007.

[2] http://www.gnu.org/licenses/gpl.html
[3] http://www.r-project.org/conferences/useR-2007/program/presentations/soukup.pdf

## INSTALLATION-QUALIFICATION OF R

The IQ-step of R should ensure that R is installed correctly. Below prerequisites, installation guidance and evaluation of the delivered tests by R are reviewed.

#### OPERATING SYSTEMS AND MACHINES

R runs at Unix-like, Windows and Mac families of operating systems. For Windows machines R runs on Windows 2000 and later (including 64-bit versions of Windows) and on ix86 and x86_64 chips.

#### OBTAINING R

Download the latest version of the R installation file (R-v*ersion-number*-win32.exe) from CRAN (Comprehensive R Archive Network)[4] .

#### SOFTWARE PREREQUISITES

Before the installation of R one should install the latest version of Rtools[5]. This set of tools contains among other things command lines tools like cat, cp, cut, date, diff, echo grep, gzip, make etc. which are needed for the installation tests.

#### INSTALLATION

Execute the downloaded R installation file (R-v*ersion-number*-win32.exe) as Administrator. Follow the instructions during the installation. Select an installation folder, choose "Full installation" and at the end select the checkbox for "Save version number in registry". Afterwards the R software will be installed. R comes along with three bundles of packages. This are the basic-, base- and recommended- packages

#### TESTING

The R Core Team provides test for the IQ within the `library('tools')`.The bundle of packages that are tested are called basic, base, and recommended packages.

**Basic packages**

The basic package includes functions that are used for arithmetic operations, complex numbers, R expressions, linear models, regular expressions etc. Call for this test routine:

```
testInstalledBasic(scope = "both")
```

**Base packages**

The base package includes functions that are used for dates, grid graphics, regression splines, ANOVA, data set examples etc. Call for this test routine:

```
testInstalledPackages("base")
```

**Recommended packages**

The recommended package includes functions that are used for bootstrapping classification, cluster analysis, kernel smoothing survival analysis etc. Call for this test routine:

```
testInstalledPackages("recommended")
```

**What is tested?**

Each package has a different number of test scripts which can be found in the directory *installation-path*\library*package-name*\tests). Each test script (*package name.*R) has an undefined number of test cases where different functions are called. During the tests, print outs of the test scripts are saved into a test output file (*package name*.Rout.save). At the end, the test output file will be compared with a reference file (*package name*.Rout) by text matching. This approach is illustrated in Figure 2.

[4] http://cran.r-project.org/

[5] http://www.murdoch-sutherland.com/Rtools/

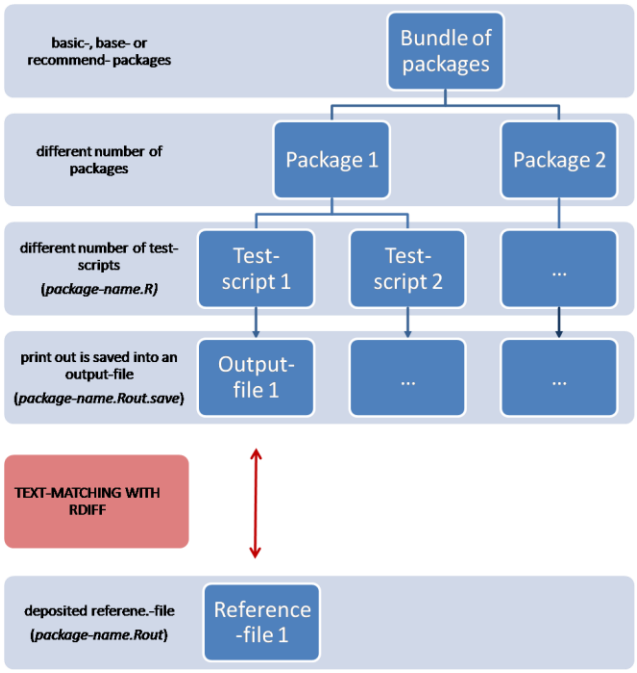



Figure 2: Test approach delivered by the R-Core-Team

If one of the function calls should fail, the test routine stops and a failure file (*package name*.fail) with information about the error are saved.

In terms of validation the complete IQ- procedure has to be well documented by an installation-plan and checklists. Here, every company has to specify requirements on e.g. access to directories, working memory and what tests should be made..

## OPERATIONAL QUALIFICATION OF R

During the OQ-step of R, the correct computation of algorithms and functions should be proven. On the surface, it may seem that the test approach provided by the R core-team (see IQ of R) could also be used for an OQ. Besides test wether a function can be invoked or not the test script also contains tests that prove the accuracy of computations. However, a closer examination of the code reveals a major obstacle: There is no general concept for the entire R functions, as to what a test of such a function should look like.

### DISADVANTAGES OF THE R INTERNAL TEST APPROACH

Some test scripts only test for correct function calls while others contain unit tests, where the result of a function will be compared with a predefined reference result. Still other tests only print out the results (e.g. `summary()`) of a function to the test output file without proof of correctness.

The main drawback that I see lies in the text comparison approach between the test output file and the reference file. Imagine the following scenario which can be observed often in program code. A developer must write a function to multiply two variables, but for some unknown reason, he produces a bug by coding a division:

```
multiplyVars <- function (x,y){
    x/y
}
```

The developer then defines two vectors and uses the `summary()`- function to check the results in the test script:

```
vector1 = c(2,4,6,8,10,12,14,16,18,20)
vector2 = c(2,2,2,2,2,2,2,2,2,2)
summary(multiplyVars(vector1, vector2))
```

In some cases, without cross checking the output from the flawed code is then stored as a reference result in the reference file, shown in Figure 3.

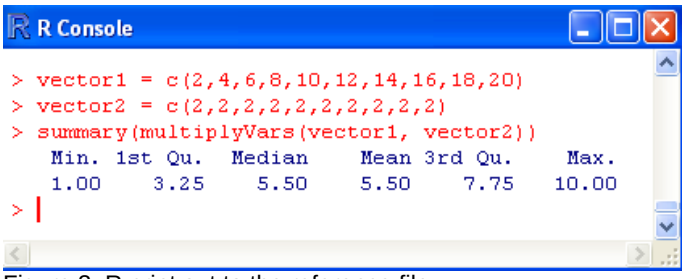


Figure 3: R print out to the reference file

As mentioned above, when the test script file is executed during the IQ, the resulting test output file will be compared to the reference file using text matching. That means, each line of strings of the flawed test output file is matched with the flawed code of the reference file. In the end, this approach misses the detection of this failure.

In some cases, similar failures could even be seen when a developer programmed complex unit tests. The unit tests results indicate a flawed code but the message referring to that has been written unobserved into the reference file. The same message will be written to the test output file during the IQ and again the text matching approach prevents the detection of that bug.

#### EVALUATION OF THE R INTERNAL TEST APPROACH

How should the outlined disadvantages of the test approach, provided by the R-Core-Team, be rated? In fact, they should be watched critically, although to be honest, we only notice them because R is open source. This is no alibi, but should ultimately be seen positively. When we notice a bug as a user of open source software, we are able to modify the code and fix the bug. In doing so with the relevant code, the validation process is enhanced, because we can be sure that the test routine is in best order. That would be not the case for commercial software because we would never get inside the code of the test routines.

#### WHAT WOULD BE THE BEST THING TO DO NOW?

First of all, one should define the algorithms that should be included into operation in the validated environment. Go to the corresponding directory (*installation-path*\library*package-name*\tests) of the package that the function belongs to, and look through the test script file critically. Then raise the following questions:

- What has been tested?
- Are tests missing (code coverage)?
- Do reference results that are cross checked exist?
- Do the test functions produce error messages for e.g. missing input parameters?
- Which tests do I have to develop on my own?

If you miss tests or if you develop own functions, you should make use of **unit tests:**

> *In computer programming, unit testing is a method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure. Unit tests are created by programmers or occasionally by white box testers.*[6]

Functionality for unit testing is implemented in the R packages RUnit and svUnit. For example, RUnit works with so called test suites which are collections of test functions and files relating to one topic. While running the test suites, the status of the unit tests is summarized (Figure.4).

[6] http://en.wikipedia.org/wiki/Unit_testing

```
Test Suite :
Regressions-Tests - 5 test functions, 0 errors, 0 failures

Details
**************************
Test Suite: Regressions-Tests
Involved directory: /tests
--------------------------
Test file: /tests/runitlm.r
test.lm.Coeff: (2 checks) ... OK (0 seconds)
test.lm.exceptions: (3 checks) ... OK (0 seconds)
test.lm.pValue: (2 checks) ... OK (0 seconds)
test.lm.stdError: (2 checks) ... OK (0 seconds)
test.lm.tStatistic: (2 checks) ... OK (0 seconds)
```

Figure.4: Summary of the unit tests generated with the R package RUnit.

Usually the unit tests should be run once a day.

## CALLING R FROM OTHER SOFTWARE SYSTEMS

Once an algorithm or piece of code crosses the line from research into a validated environment, e.g. when submitting results to the FDA are required, the code must be re-implemented, for example with C++ or C#, into a well documented and validated software application. This is cost and time intensive. For this reason, concepts are needed as to how R can be called from other systems. Once the mentioned IQ and OQ processes for the R software are established, one of the following interfaces to R can be applied:

### R (D) COM SERVER

The R (D) COM Server[7] provides a COM interface for R. In addition there are ActiveX controls that can be included in the application. Once installed, references can be added e.g. to a C# .NET project as well as the namespaces to a class[8]:

```
using STATCONNECTORCLNTLib;
using StatConnectorCommonLib;
using STATCONNECTORSRVLib;
```

For example, to generate twenty random normal numbers, one uses:

```
object Rreturn;
int n=20;
StatConnector connectionR = new StatConnectorSRVLib.StatConnectorClass();
connectionR.Init("R");
connectionR.SetSymbol("n",n);
connectionR.Evaluate("x<-rnorm(n)");
Rreturn = connectionR.GetSymbol("x");
x =(double) Rreturn;
```

In C# the object `x` holds the data vector which was computed by R.

### RWEBSERVICES

The RWebServices package[9] implements functionality to generate JAVA code for webservice methods. Generated classes can be attached to server components such as Tomcat, axis or activeMQ.

### RSERVE

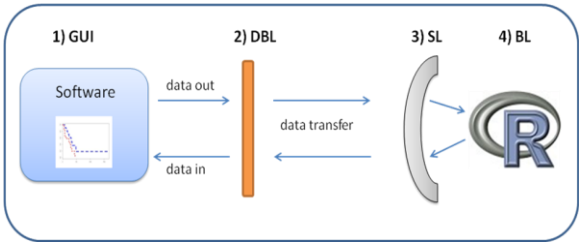
The Rserve[10] package is a TCP/IP server which gives different clients (C, C++, PHP and JAVA) access to R functions. The following Java code illustrates the integration of Rserve:

```
RConnection c = new RConnection();
REXP x = c.eval("R.version.string");
```

[7] http://mirrors.softliste.de/cran/contrib/extra/dcom
[8] Example taken from http://www.codeproject.com/KB/cs/RtoCSharp.aspx
[9] http://bioconductor.org/packages/release/bioc/html/RWebServices.html
[10] http://www.rforge.net/Rserve/index.html

**CALL LOCAL R INSTANCE USING .NET WCF SERVICES**

The .NET WCF technology allows building up a service layer by using the so called WCF services. Similar to the RWebServices packages, R is encapsulated into a webservice. WCF services have the advantage of simply switching from a binary data transfer method to an XML based webservice. The latter will be used to communicate with *R* which is installed on a server in a network.




1) graphical user interface (GUI) (other system)
2) data bridge layer (DBL)
3) service layer (SL)
4) business logic (BL) implemented in *R*

## CONCLUSION

How much is it to validate an OSS like R? Well in the case of R the software package is free. There are no costs for license. But one should spend some time to do code review especially for the tests scripts. Here it should be decided whether the implemented tests cases are sufficient for ones requirements. If some risks are not covered unit-tests should be developed. Money could be saved by integrating algorithms written in R into an existing validated software system. Here re-implementation by e.g. C++ or C# code can be passed over.

In conclusion, R has the potential of being integrated into processes that have to be validated. Although documentation of validation processes was not discussed, hopefully the ideas of this paper will help you to adopt the OSS R as a candidate software for your special requirements.

## RECOMMENDED READING

http://www.r-project.org/doc/R-FDA.pdf

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please contact the author at:

Dr. Peter Bewerunge
HMS Analytical Software
Rohrbacher Str. 26
69115 Heidelberg
Work Phone: +49 6221 6051-58
Fax:+49 6221 6051-7-58
Email: peter.bewerunge@analytical-software.de
Web: www.analytical-software.de