# SCTO Validation Platform

## R add-on Package Function testing

Release date: 2025-01-01

---

This document is an integral component of the SCTO Validation Platform

---

SCTO PLATFORMS

ℹ Document development, review and version history

***Development and Review***

Authored/revised by:

| Name | Date |
|---|---|
| Michael Coslovsky,[1] Alan Haynes,[2] Lisa Hofer,[3] Christina Huf,[4] Christine Otieno,[5] Elio Carreras,[6] | 2025-01-01 |

***Version History***

| Version | Date | Author(s) | Summary of Changes |
|---|---|---|---|
| 1.0 | 2025-01-01 | Michael Coslovsky, Alan Haynes, Lisa Hofer, Christina Huf, Christine Otieno, Elio Carreras | Initial version |

# 1 Purpose

This SOP defines the process and steps that need to be followed when testing the performance of specific functions of R add-on packages within the SCTO-Statistics' platform computerized systems validation policy for R. The process is written to serve as a standardized approach for function testing. A standardized process will allow different organizations to rely on results of tests performed also outside of the internal organizational unit, improving efficiency overall. The centralized process can then be implemented in all associated units as the minimal procedure. Organizations may go beyond the process described herein, as long as they do not contradict the central process.

In lay words, the process should allow a member (with focus on statisticians and data-scientists) of any participating institution (SCTO associated organization, including SAKK, the Department of Biostatistics of the University of Zurich etc.) to understand and reproduce function tests saved and documented on the designated SCTO platform infrastructure. In addition, the process ensures that re-run of tests can be done in a fast and straightforward manner once package updates or function updates occur, so that package updates can be validated easily.

# 2 Abbreviations & Definitions

**SOP**

Standard operating procedure

**R**

The R statistical programming language and environment

**Unit testing**

[1]Head data-analysis, Department Clinical Research, University of Basel
[2]Senior Statistician, Department of Clinical Research (DCR), University of Bern
[3]Statistician, Department Clinical Research, University of Basel
[4]Head Quality Management, Department Clinical Research, University of Bern
[5]IT Quality Manager, Department Clinical Research, University of Basel
[6]Senior Statistical Programmer, SAKK

The testing of one particular element of a program. A function can have multiple units; for example, R's lm() function provides several units as outcome: the coefficients, the standard errors, the coefficient of determination etc.

**Function**

A specific program within the R language

**Function test**

Any action that allows a function to be considered a tested function

**Test author**

Any employee of an SCTO associated organization writing a function test according to this SOP

**Tested function**

A function that underwent the process described in this SOP to assert with determined certainty that it provides the intended outcome

**Tester**

Any employee of an SCTO associated organization executing and documenting a function test according to this SOP

**Reviewer**

Any employee of an SCTO associated organization reviewing a function test from a different test author according to this SOP

## 3 Scope

This SOP defines the process for:

1. Unit testing
2. Self-written functions stored on the SCTO platform infrastructure.
3. Specific functions from software used in the institutes that have been determined as requiring testing.

In scope are functions of the R computing environment, written within or out of the organization, for which the necessity of unit functional testing was determined by a member of a participating institute according to the SCTO computerized systems validation policy for R. The process may be adapted for other statistical software. The SOP focuses primarily on functions used for statistical reporting and data-wrangling.

The appropriate level of testing required is determined within the SOP, but the decision to test the function is out of scope. According to the SCTO R computerized systems validation policy, this decision is determined from the risk associated with the intended use of the function within a specific R product or project, future use and the software packages being used for the R product. The SCTO statistics' platform's business process risk assessment as well as the SCTO computerized systems validation policy for R describe the processes to follow in assessing this risk and in determining whether specific function testing should be performed.

## 4 Process

The decision tree depicted in Figure 1 describes the process a test author undergoes when testing a function output. For purposes of this SOP, the resulting outcome of following the decision tree is considered a "function test" once implemented and documented.

The process starts after the test author determined that a function requires testing. More specifically, the test author determines which outputs of the function require testing. A function test is performed on output level. This means that each output requires its own test.

The decision tree has five levels of complexity:

1) The function output of interest has already been tested on the SCTO platform.
2) There is already test code available in the function package.
3) The function output can be compared to a known result.

4) The function output can be compared to the output of another implementation/program.
5) The function output needs to be validated based on simulations.

In all complexity levels, the test author must decide when a test is appropriate and sufficient. Since this depends on the purpose of the function output and on the risk level of the project, an existing function test might still need to be extended.

Starting at level 1 Box (A), the test author goes through the decision tree as described in Figure 1 below. At level 1 it is examined whether the specific function output has already been appropriately tested on the SCTO platform, while ensuring that the tests are always updated on the newest R add-on package version (Box B). All test code, written by the test author or external R add-on package test code used in level 2 (Box C), should follow standard guidelines for good programming practice, such as the *tidyverse* style guide. If a comparison to a known result in level 3 is needed (Box D), the known result must be provided by the test author. This can be the solution from our 'hands-on' defined calculation of the expected outcome, without use of the function that is being tested ('program from scratch'), within the test script, or the result from a published dataset or analysis. If we compare with output from another implementation or another program, a prerequisite is that this other implementation or program has already been tested (Box E). Note that if this comparison is due to the tested process being stochastic, with no deterministic solution, it should be defined what constitutes an acceptable similarity of results. Finally, in case simulations are used in the testing in level 5 (Box F), make sure to set a seed, to explain the simulation steps with suitable comments and to submit the simulation code together with the final test. A large number of simulations (replications), or several seeds (at least three) should be implemented. The source of the data frame used for the test is irrelevant, i.e., could be simulated or published, as long as the source is documented. To this end, all datasets used for tests are required to be stored in a designated folder in the *validation_tests* SCTO GitHub repository. Published datasets can be stored directly (e.g., as .csv files) while simulated datasets require uploading the script that generated the data. To facilitate generating new tests, these stored datasets can be reused if they have the appropriate properties.

Function tests need to be reviewed by another SCTO member based on the 4-eye principle (Box G). Once the function (output) test is accepted by the reviewer, it will be added to the SCTO list of tested function outputs with the necessary documentation of meta-data (Box H).

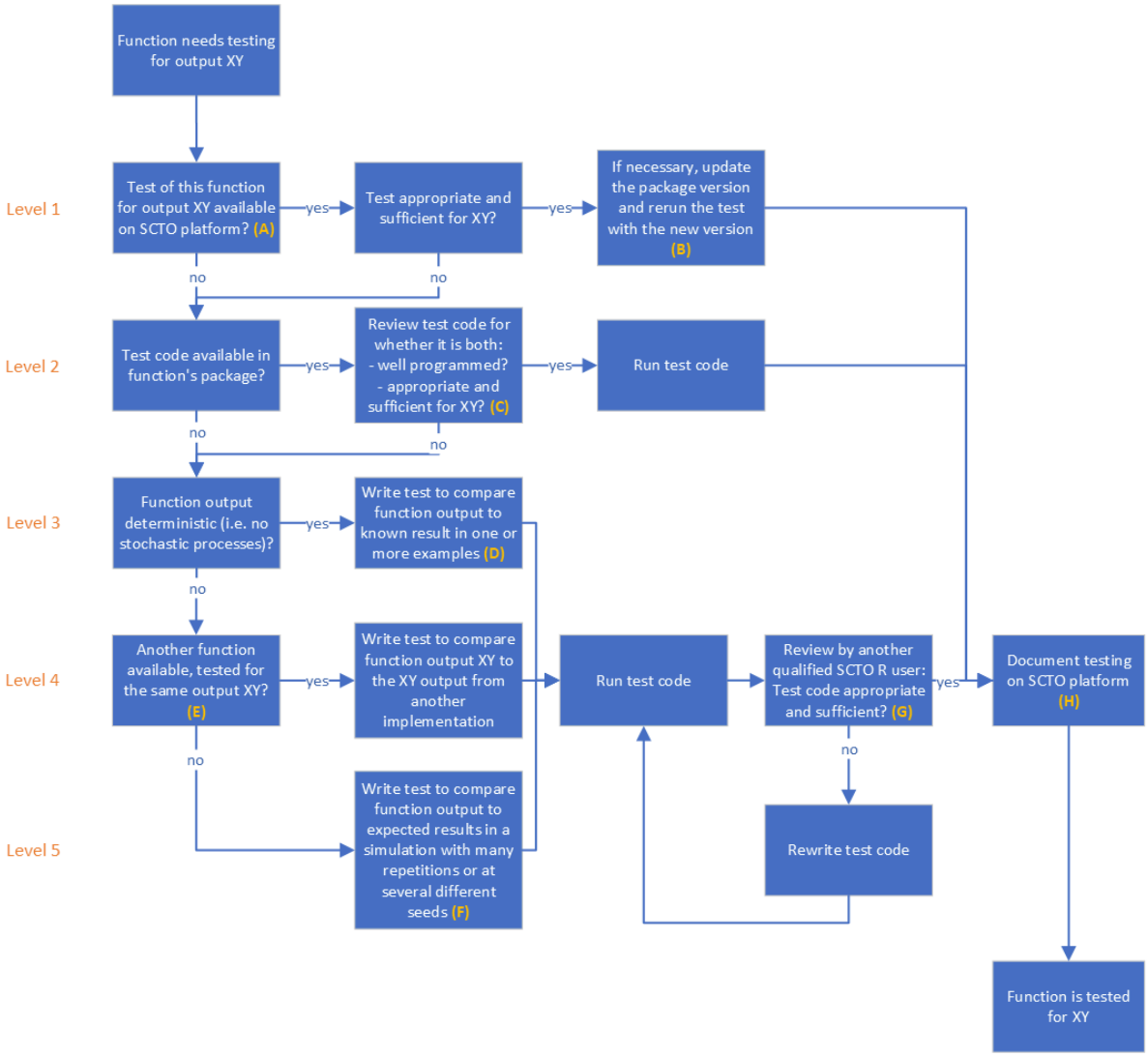


Figure 1: Decision tree for deciding on the type of testing a function requires

# 5 Guideline

## 5.1 Working instructions

> ℹ Suggestion from Alan
>
> (to replace the paragraph just below, which is basically the same thing twice…)
>
> Detailed working instructions for writing tests, reviewing tests, and running tests can be found in the respective work instructions. A summary of the main steps is provided below.

The working instructions for the unit testing process are found under the relevant work instruction. The detailed working instructions can be found on the GitHub page; Below we provide only a bullet-point description of the relevant steps:

- Unit tests of an R function are performed and stored on the SCTO's statistics platform's validation_tests GitHub repository

- The repository contains both the test script(s) and the data frames used for the tests when relevant
- Approving the test is conditional on a review by a second qualified SCTO R user, from a different organizational entity (ideally, a different CTU) than the test author. Reviews are based on the “four eyes” principle.
- The SCTO function-tests platform collects relevant meta-data for the tests including:
  - Test author
  - Test timing (date of submitting the test to the repository)
  - Who reviewed the test code
  - When was the review performed
  - Which function was tested: function name within package and package version
  - Which output of the function was tested
  - What type of testing was performed
  - Test result (pass/fail)
  - Evidence of test (copy of console output)
  - Session information of the testing environment
- The SCTO develops a package that allows easy rerun of package tests upon change of R or package version, as well as generating a report for testing.

Function tests are stored in the validation_tests SCTO GitHub repository. Tools to create the testing structure for each tested package and run the tests are provided in the designated `SCTORvalidation` R add-on package. See the work instruction on writing tests for more information.

#### 5.1.1 Submitting the test for incorporation into the framework

Instructions for writing tests and incorporating them into the validation_tests SCTO GitHub repository are provided here. Prior to incorporation, the test(s) will be reviewed according to the criteria below (“Approving the test”; Section 5.2).

#### 5.1.2 Running tests

The output from test can then be used to complete the function test issue form on GitHub. All functions from a particular R add-on package can be reported together in a single report.

Here, we distinguish between tests that are already implemented within an R add-on package, and those that have been developed as part of the SCTO framework.

See the work instruction on running tests for more information.

### 5.2 Approving the test

A consistency review is performed based on the “four eyes” principle where the reviewer should be chosen from a different participating institution than the author of the test. The review should address the following points:

- Is the listed meta-data complete?
- Is the level of testing specified appropriately?
- Does the test follow the procedure described in this guideline?

This review is done in effect by approving the ‘pull request’ on GitHub and following the working instructions listed under the Reviewing tests work instruction.

The reviewer comments the function test. Only after approval by the reviewer, the function will be listed as tested on the SCTO platform.

### 5.3 Documentation

Test reports are made on the pkg_validation SCTO GitHub repository. Select “New issue”. Select the appropriate issue type depending on whether you’re submitting tests from within the package itself or tests from within the

SCTO framework. For packages testing using tests from the validation_tests SCTO GitHub repository, fill out the predefined form, using the output from R (`SCTORvalidation::test("packagename")`). Where packages from within the package itself are used, the work instruction provides additional information.

| What to report | Details |
|---|---|
| Name of tester | Who performed the test? |
| Test date | When was the test performed (date) |
| Tested function | Which function from which R add-on package and package version. If the function does not come from a package (e.g. it's a script that is stored as a GitHub Gist), the link to the code should be provided. |
| Test details | What precisely was tested? This could be a link to the tests or a reference to the R add-on package and function containing the tests together with that package's version number. |
| Degree Type of testing | Which pathway in the decision tree was followed:<br>1. Re-run prior test<br>2. Review existing R add-on package test code<br>3. Deterministic process<br>4. Comparison to other implementation<br>5. Simulation Was the testing comprehensive, superficial, or something in between? |
| Test result | Pass/fail |
| Evidence of test | Copy/paste of the console output |
| SessionInfo | Relevant parts of `sessionInfo()`:<br>• R version<br>• OS<br>• Which other R add-on packages and versions were loaded? |

Failed tests should also be reported. When a test fails, a bug report should be posted via the appropriate route for that R add-on package (e.g. a github issue to the R add-on package repository, an email to a specific address). The bug report should also be noted alongside the test results, where possible providing a link to the report (e.g. to the GitHub issue), and followed up on by the individual discovering the bug. When the bug has been fixed, the tests can be run again and the successful test result recorded as above.