SCTO Validation Platform

Function testing

Date: December 10, 2024

This document is an integral component of the SCTO Validation Platform

---

SCTO PLATFORMS

ℹ Document development, review and version history

Development and Review

	Name	Date
Authored/Revised by		
Reviewed by		
Released by		

Version History

Version	Date	Author	Summary of Changes
0.1	2024-08-24	Alan Haynes, Elio Carreras, Lisa Hofer, Michael Coslovsky, Christina Huf, Christine Otieno	Initial draft

1 Purpose

This SOP defines the process and steps that need to be followed when testing the performance of specific program functions within the SCTO-Statistics' platform computer system validation. 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 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 biostatistics' department 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 updates of packages or functions can re-run the tests in a fast and straightforward manner, so that package updates can be validated easily.

2 Abbreviations & Definitions

Abbreviation or term	**Definition**
SOP	Standard operating procedure
R	The R computer program
Unit testing	definition
Function	definition
Test author	any employee of an SCTO associated organization writing a function test according to this SOP
Tester	any employee of an SCTO associated organization executing and documenting a function test according to this SOP

Abbreviation or term	Definition
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 R-validation policy. The process may be adapted for other statistical packages. 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 validation policy, this decision is determined from the risk associated with the intended use of the function within a specific product or project, future use and the software packages being used for the product. The SCTO statistics' platform's high level risk assessment as well as the SCTO policy for the use of R in a validated environment describe the processes to follow in assessing this risk and in determining whether specific function testing should be performed.

4 Process

The decision tree below describes the process a test author undergoes when testing a function output. 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:

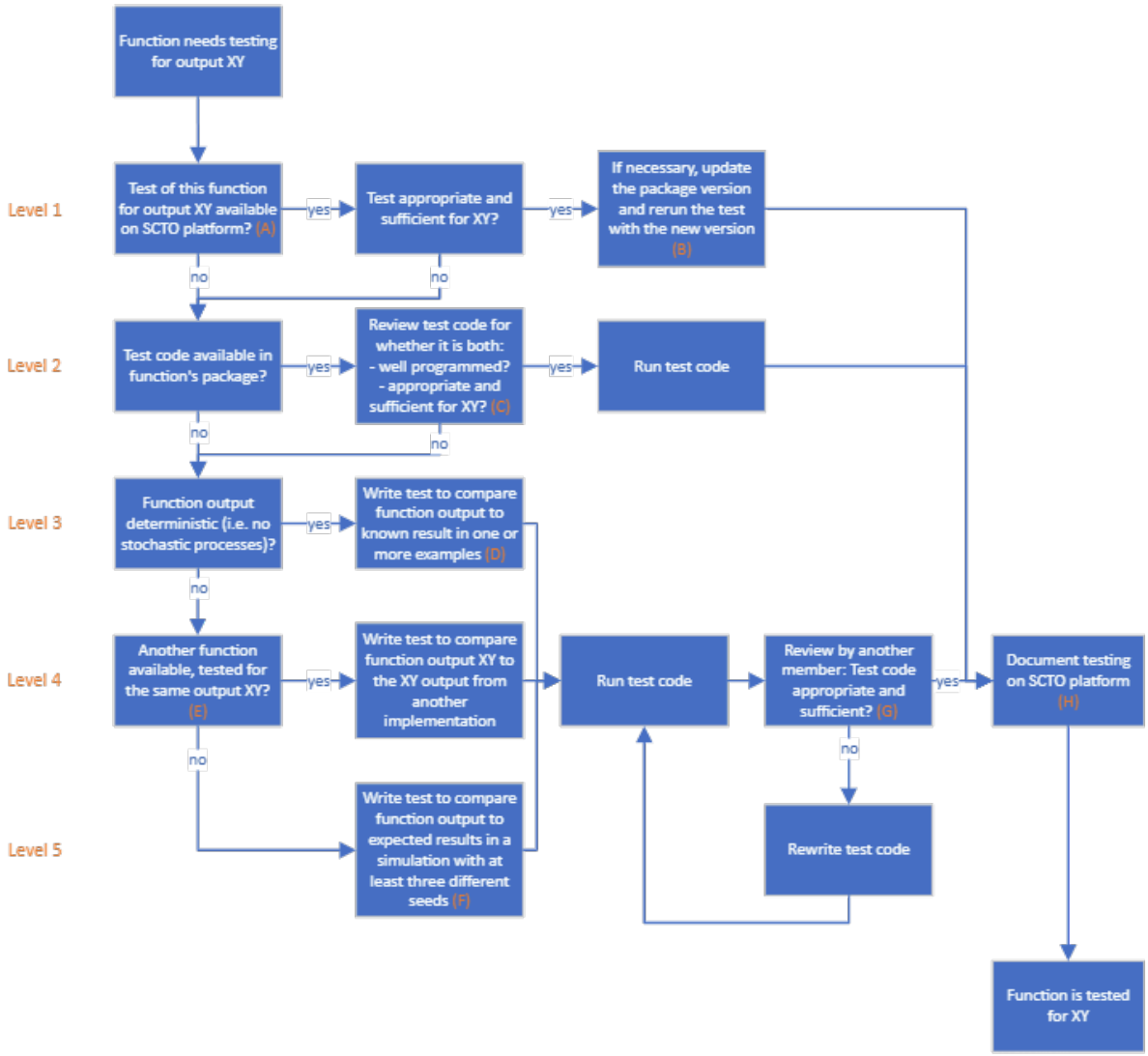
1) The function output of interest has already been tested on the SCTO platform.
2) There is already test code available in the function package.
3) The function output can be compared to a known result.
4) The function output can be compared to the output of another implementation/program.
5) The function 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 XX 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 package versions (Box B). All test code, written by the test author or external package test code used in level 2 (Box C), should follow standard guidelines for good programming practice, such as the *tidyverse* style guide (https://style.tidyverse.org/index.html). If a comparison to a known result in level 3 is needed (Box D), we can either use the solution from our 'hands-on' calculation 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 (E). 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. 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* package on GitHub. 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).



5 Guideline

5.1 Working instructions

The working instructions for the testing process are found under the relevant SCTO GitHub page (https://swissclinicaltrialorganisation.github.io/SCTORvalidation_Rpackage/articles/contribute.html). A copy of the vignette is an associated document to this SOP. Below we provide only a bullet-point description of the relevant steps:

- Function tests are performed and stored on the SCTO's statistics platform's validation_tests repository

- The repository contains both the test script and the data frames used for the tests when relevant
- Approving the test is conditional on a review by a second SCTO statistician from 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 (time 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 repository at https://github.com/SwissClinicalTrialOrganisation/validation_tests. Tools to create the testing structure for each tested package and run the tests are provided in the validation R package.

5.1.1 Submitting the test for incorporation into the framework

Instructions for writing tests and incorporating them into the validation_tests repository are provided here. Prior to incorporation, the test(s) will be reviewed according to the criteria below.

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 package can be reported together in a single report.

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

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.

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 repository at https://github.com/SwissClinicalTrialOrganisation/validation_platform/issues . Select “New issue”. Select the “Add package/function testing results”. Fill out the predefined form, using the output from R (`SCTORvalidation::test("packagename")`).

What to report	Details
Who	Who performed the test?
When	When was the test performed (date)
What	Which function from which 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 package and function containing the tests together with that packages version number.
Degree Type of testing	Which pathway in the decision tree was followed: 1. “re-run prior test” 2. Review existing package test code 3. Deterministic process 4. Comparison to other implementation 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`: • R version • OS • Which other 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 package (e.g. a github issue to the 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.

6 References