Inspection and Characterization Plan

Science Release and Validation team

November 24, 2022

Version 0.2

1 Purpose of the document

This document is aimed at describing the process to execute the test suite necessary to characterize new Rubin data releases, in the context of the DESC. It covers the following points:

- Describe the goal of characterization tests for DESC.
- The list of tests, methods, tools and criteria (where applicable) to characterize and validate the Rubin dataset, for the purposes of DESC analyses.
- The actual procedure for the execution of tests and inspection.

1.1 Definitions, reference documents

- Test run: a single execution of the whole SRV characterization framework. Includes tests using DESCQA, other sources, inspection of data and documentation.
- Test Execution Report (TER): a short document where the results of a Test run are summarized.

2 Goals of the characterization process

The overall objective of the inspection and characterization procedure is to **provide a compiled snapshot of the characteristics of the data**. This can respond to three needs:

- Provide a repository of the status of the data set used for the DESC science analyses, used as reference for understanding the results and reporting on papers. This requires a lower frequency of runs (months time scale)
- Explore characteristics of different science samples, within the same data release (weeks time scale)
- Assess issues with the data release, for DESC purposes (days time scale)

It is expected that the Rubin project will provide, alongside with the data itself, documentation and/or reports on the data quality, conducted by their own Verification and Validation tests, through the `analysis_tools`. With the framework that SRV is providing here, the intention is to cater to the interests of the DESC ('fill the gaps'), in case some tests are missing from those provided by Rubin, evaluating the characteristics of 'typical' DESC science samples, testing DESC only value added products, and rechecking some basic metrics in case the data source is different than that of the Rubin Science Platform (e.g., if DESC uses a subset of the data in a different platform or format at NERSC).
A structured compilation of all the relevant tests and inspections will be finally recorded on static Test Execution Reports, which should contain all the information necessary for provenance and regression purposes.
At the same time, it is noteworthy to consider that the execution of these tests are easily run and re-run, and do not require a significant overhead in terms of execution and SRV (DESC) scientists' time, so as to make this framework useful as a quick assessment. A final overall evaluation of the data quality for a 'frozen' version of the data release can be done with a more comprehensive approach, as a final report usable as reference for downstream analyses. Therefore, any science done would be able to use the corresponding TER metrics and references, and minimize any ambiguities as to which version of the data or catalog content is used.
The overall characterization framework is sketched in Figure 1 and summarized below. The tests referred will be described in this document:

1. Data is made visible for an official Release at the Science Platform. It is foreseen that part of this data is copied to NERSC. At this point, **inspection tests** through Jupyter notebooks can be performed quickly to assess overall data health. Examples: RA,DEC coverage; histograms of flags; other simple histograms of the main photometric quantities. It is interesting to verify whether the results are compatible at the Rubin Science Platform and at NERSC. This inspection *could* be complemented by tests being run through other tools, if that seems to be more manageable (examples: running TOPCAT and acquiring data from the Rubin Science Platform through TAP, using Lite IDACs capabilities such as those from Hadoop frameworks holding a version of the data). In all cases, the tests have to be formalized to ensure proper provenance.

2. At the same time, the presence of **adequate documentation** from Rubin side for DESC purposes, can be inspected, complementing it when appropriate or reporting back to the project.

3. **Main software tests will be run through a single framework (DESCQA)**.

4. Tests already embedded in pre-existing frameworks, or coming from downstream analyses are included through DESCQA in the final report, ensuring consistency of the data sets used for the tests, with respect to those developed specifically in the previous point.

5. Finally, besides the TER static reporting, DESCQA will provide **visualization utilities** for the test runs and re-runs that will inform the contents of the TER.

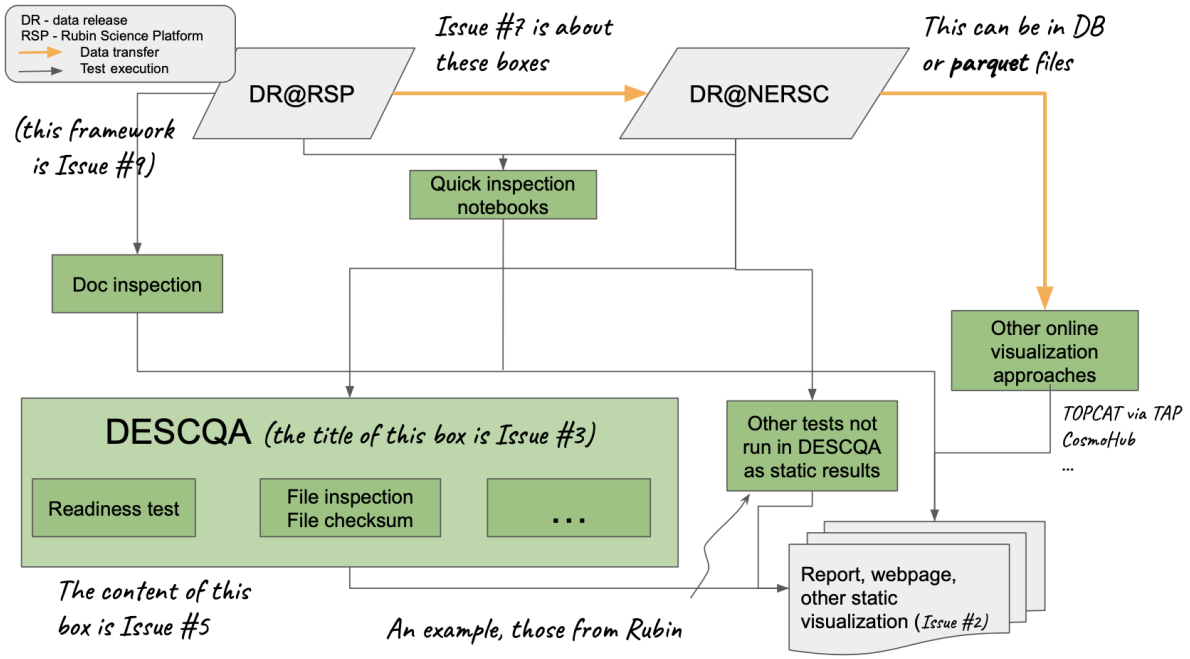

Figure 1: SRV planning diagram

3 Test tool description

Here we include an overall view of the software involved in a single test run of the SRV characterization framework.

3.1 Software tools

A list of references to the software used. The Test Execution Reports will be filled out with the actual versions and other specifics. These will be mainly pointers to GitHub repos holding the code to be executed.
The main software tool is based on DESCQA[1], a framework to validate simulated galaxy catalogs.

3.2 Data sets and formats

An explanation of the characteristics of the model of the data set being tested.

3.2.1 DC2

DC2 coadd catalogs are available as flat parquet files at NERSC. *The TER should include the actual location and details of the data that was used specifically.*

[1] https://github.com/LSSTDESC/descqa

3.2.2 DP0.2

DP0.2 coadd catalogs are available as flat parquet files at NERSC. *The TER should include the actual location and details of the data that was used specifically.*

4 Characterization cases and procedures

4.1 Quick inspection notebooks

Interactive notebooks to be included here that complement or substitute DESCQA executions, that could be run on NERSC and RSP. Also CosmoHub could be an option if the data is available.

4.2 Documentation inspection

Describe what documentation should be present to understand the data.

4.3 Characterization of data set through DESCQA

4.3.1 TC1 - Readiness tests

Purpose:
Run a general 'readiness' test on coadd catalogs, to verify that there aren't any significant issues in data. Plot histograms and scatter plots of listed quantities and perform range, finiteness, mean and standard deviation checks.
Strategy:
Execute an interactive job of the srv_readiness test from DESCQA. These should be done on the overall data set. *Eventually, subsamples can be explored as well.*
The test currently comprises the following checks:

- RA,DEC plot. Check that the footprint in the range expected by the associated data release documentation.
- Differential magnitude histograms in all ugrizy bands, for PSF, aperture and model magnitudes. Check that the depth is in the range expected by the associated documentation.

Procedure:

1. Create a memory allocation on a NERSC node: salloc -N 1 -C haswell --qos=interactive -t 02:00:00
2. From the descqa directory (check version and state on TER), execute ./run_master_parallel.sh -v -c [catalog name] -t srv_readiness.
3. Fill in TER for code version and tested catalog.

4.3.2 TC2 - Shape-related characterization

Purpose:
Run a few diagnostics on core measurements of the shapes on each band.
Strategy:
Execute an interactive job of the srv_shear test from DESCQA. These should be done on the overall data set. *Eventually, subsamples can be explored as well.*
The test currently comprises the following checks:

- Ellipticity plot.
- Moments plot.
- PSF plot.

Procedure:

1. Create a memory allocation on a NERSC node: salloc -N 1 -C haswell --qos=interactive -t 02:00:00
2. From the descqa directory (check version and state on TER), execute ./run_master_parallel.sh -v -c [catalog name] -t srv_shear.
3. Fill in TER for code version and tested catalog.

4.3.3 TC3 - Galaxy density

Purpose:
Measure number of galaxies per square arcmin after successive quality cuts. Plot their sky distribution and write global statistics.
Strategy:
Execute an interactive job over the whole data set of the srv_ngals test from DESCQA.
Procedure:

1. Create a memory allocation on a NERSC node: salloc -N 1 -C haswell --qos=interactive -t 02:00:00
2. From the descqa directory (check version and state on TER), execute ./run_master_parallel.sh -v -c [catalog name] -t srv_ngals.
3. Fill in TER for code version and tested catalog.

4.3.4 TC4 - TXPipe results

Incorporate TXPipe results of tests on the same data set that will complement this report.
Strategy:
Execute an interactive job over the whole data set of the ExternalTest test from DESCQA.
Procedure:

1. Create a memory allocation on a NERSC node: salloc -N 1 -C haswell --qos=interactive -t 02:00:00
2. Configure the ExternalTest yaml file to point to the TXPipe directory to be read.
3. From the descqa directory (check version and state on TER), execute ./run_master_parallel.sh -v -c [catalog name] -t ExternalTest.
4. Fill in TER for code version and tested catalog.

also to be considered RAIL, faro, other analysis WG results.

4.4 Specific comparison tests between catalog versions

Some of the tests above could specifically test new features, fix issues, relevant to DESC

4.5 Regression testing

Here we would include the minimal list of sanity checks that would be rerun each time there is a new version of the data to check that the core characteristics have not changed.

4.6 Validation tests on small areas or subsamples, replicating previous scientific results

5 Test execution reports and visualization

The reports should include date, DESCQA version (and other SW), data set version, people involved in the testing. Then the results would be a summary of an online resource where the complete collection of plots and numbers would be available.