TS04

Running OpenCDISC from SAS

Mark Crangle
Introduction

- The OpenCDISC validator is a tool used to check the compliance of datasets with CDISC standards
  - Open-source
  - Metadata driven
  - Java based
  - Can validate a range of CDISC datasets
    - Freely available and created by team of experts
    - Validation rules can be easily modified
    - Usable on a variety of operating systems
    - Solution can be applied to many types of dataset

- Tool is packaged with a graphical user interface but can also be run from a command line interface (CLI)
Commonly a package of datasets is only checked for CDISC compliance once all are complete.

If any issues are identified then the dataset is updated and re-validated.
- Inefficient as can result in rework
- Changes could affect other dataset/analysis programs that use the data

More efficient solution would be to check CDISC compliance at the same time as other validation activities.

Issues can be identified and fixed early.

The OpenCDISC Validator can be configured to work more accurately for individual datasets.

The CLI can be used directly from a SAS session.
- Tool can be run alongside other validation activities
- Results can be read back in to a SAS dataset, analysed and reported with other validation output
The OpenCDISC Validator

The OpenCDISC validator can be used to create the Define.xml that is required for electronic submission.
The OpenCDISC Validator

The OpenCDISC validator can be used to create the Define.xml that is required for electronic submission. As mentioned above, the tool can be used to validate different types of CDISC standard items.
The OpenCDISC Validator

As mentioned above, the tool can be used to validate different types of CDISC standard items. It is possible to include datasets in xpt, xml or delimited format.
It is possible to include datasets in xpt, xml or delimited format. Datasets to be checked are specified here.
The OpenCDISC Validator

Datasets to be checked are specified there. The configuration specifies the checks that will be performed on the data and the metadata that these checks will be using.

The validator is packaged with configuration files but these can be edited or new files created.
The configuration specifies the checks that will be performed on the data and the metadata that these checks will be using. The validator is packaged with configuration files but these can be edited or new files created.
The validation file for the dataset specified can be supplied in CSV. Excel format is the easiest to read but the CSV file can be more easily read into a SAS dataset if required.

The tool is packaged with different versions of the ADaM, SDTM and SEND Controlled Terminology which can be selected here.
Validator Configuration

- The rules the validator uses are specified in xml configuration files
- The tool is packaged with preset configuration files for each release of CDISC standards

- Configuration file has section for global rules and then one for each dataset type
  - In ADaM configuration, one for ADSL and one for BDS
  - Each section is further divided into the variable metadata and rules specific to that datatype.
The configuration file can be viewed in a web browser to show the validation rules in a user friendly format.
Validator Configuration – Setting for Individual Datasets

- Some of the checks are used to check consistency between datasets but these are not needed for checking individual datasets
  - Checking an ADaM package contains the required ADSL dataset. (AD0001)
  - Checking subjects present in a dataset are also included in ADSL/DM. (AD0053)

- To de-activate a specific check the xml file needs to be edited in a text editor
- Rules are applied to each dataset type by the `<val:ValidationRuleRef>` tag
  `<val:ValidationRuleRef RuleID="AD0001" Active="Yes"/>

- To de-activate a check set Active to No
- Once the file is saved the change can be easily seen in the web-browser view
Command Line Interface

- The GUI is the most commonly used way to run the validator but it is also accessible using a command line interface.
- This is accessed using the file `validator-cli-version.jar` (eg. `validator-cli-1.5.jar`)
- Same options as the GUI are available as parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Valid Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-task</td>
<td>Validate, Generate</td>
<td>Validate data or generate a Define.xml</td>
</tr>
<tr>
<td>-type</td>
<td>SDTM, ADaM, SEND, Define, Custom</td>
<td>Data Standard/Structure to validate</td>
</tr>
<tr>
<td>-source</td>
<td>Path to the source data files</td>
<td></td>
</tr>
<tr>
<td>-config</td>
<td>Path to the xml configuration document specifying the rules/metadata to validate</td>
<td></td>
</tr>
<tr>
<td>-config:define</td>
<td>Path of the define.xml for the study</td>
<td></td>
</tr>
<tr>
<td>-config:cdisc</td>
<td>CDISC Controlled Terminology Version</td>
<td></td>
</tr>
<tr>
<td>-report</td>
<td>Path and filename where the validation report will be saved</td>
<td></td>
</tr>
<tr>
<td>-report:type</td>
<td>Excel, CSV</td>
<td>Report format</td>
</tr>
</tbody>
</table>
• The command line interface can be used through the command prompt
  – Navigate to the folder where the CLI file is located
  – Run the command
    ```
    java -jar validator-cli-1.5.jar
    ```
    adding the parameters needed
• For example
  ```
  java -jar validator-cli-1.5.jar
  -task=Validate
  -type=ADAM
  -source="U:/test data/* .xpt"
  -config="U:/test data/opencdisc-validator/config/config-adam-1.0.xml"
  -config:cdisc=2011-07-22
  -report="U:/test data/opencdisc-validator/reports/report.xls"
  -report:type=Excel
  ```
Using OpenCDISC in dataset validation

- Combine CLI with ability to run system commands from SAS to control the OpenCDISC validator from within SAS
- The edited configuration file is used to eliminate error messages that are caused by only submitting one dataset
- This is handled by a macro that can be called in a dataset validation program

Example

/*Code to create validation dataset*/

ODS RTF FILE = "outputfile.rtf"

PROC COMPARE BASE=dev COMPARE=val;
    ...;
RUN;

%runOpenCDISC(version, dataset, etc.);

ODS RTF CLOSE;
Using OpenCDISC in dataset validation - Steps

- Validator outputs report to CSV file as easier to read into SAS

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Record</td>
<td>Variable</td>
<td>Value</td>
<td>Rule ID</td>
<td>Message</td>
</tr>
<tr>
<td>ADLB</td>
<td>25897</td>
<td>PARCAT3, PARAMCD</td>
<td>Secondary Laboratory, HESSLES</td>
<td>AD0124</td>
<td>Inconsistent value for PARCAT3 within a unique PARAMCD Consistency Error</td>
</tr>
<tr>
<td>ADLB</td>
<td>25899</td>
<td>PARCAT3, PARAMCD</td>
<td>Secondary Laboratory, HHCTS</td>
<td>AD0124</td>
<td>Inconsistent value for PARCAT3 within a unique PARAMCD Consistency Error</td>
</tr>
<tr>
<td>ADLB</td>
<td>25900</td>
<td>PARCAT3, PARAMCD</td>
<td>Secondary Laboratory, HHGBS</td>
<td>AD0124</td>
<td>Inconsistent value for PARCAT3 within a unique PARAMCD Consistency Error</td>
</tr>
</tbody>
</table>

- Read CSV file into SAS dataset

  PROC IMPORT OUT=test DATAFILE = "U:\OpenCDISC ADLB.csv" DBMS=CSV REPLACE;
  GETNAMES=YES;
  DATAROW=2;
  GUESSINGROWS=100000;
  RUN;

- Summarise issues

  PROC FREQ data=report NOPRINT;
  TABLE rule_id*message / OUT=numobs;
  RUN;
Further Processing

• Report could be checked programmatically and further code run conditionally if common issues found
  – Common issues is that PARAM must have the same value within each unique value of PARAMCD (1-1 correspondence)
  – Difficult to see complete issue just by searching through report
  – Check if report contains this issue then run code to show all unique combinations of PARAM/PARAMCD so can easily see where the issue lies.

• Permanent copies of datasets containing OpenCDISC report could be kept to allow tracking of issues
  – Any warnings/notes that have been investigated and deemed acceptable could be flagged so that they are not reported each time
  – Tracking of common issues could be used to identify training needs across department
Conclusion

- CDISC compliance is something that must be considered at all stages of dataset design, development and validation
- Only checking compliance after all datasets are complete is inefficient and could require re-work

- Using the CLI, the OpenCDISC validator can be controlled from within a SAS session to check individual datasets
- This can be done inline with other validation activities
- OpenCDISC report can be processed by SAS to summarise issues
- Reporting can be done alongside output from other validation methods to create a complete record of dataset quality
ANY QUESTIONS?