data-quality-rules Documentation

De Nederlandsche Bank

Jul 04, 2023

Contents:

CHAPTER 1

data-quality-rules

[This repository has been archived on 4/7/2023]

This is Data Quality Rules repository for the Solvency 2 and the Verzekeraars Nationale Staten (VNS) quantitative reporting templates (QRTs). In this repository De Nederlandsche Bank (DNB) publishes additional validation rules to improve the data quality of the QRTs. This repository can be used by entities who submit supervisory reports to DNB.

The use of the validation rules published in this repository is not mandatory. You may choose to submit supervisory reports without prior evaluation of these validation rules. If you choose to use the validation rules published in this repository then you cannot derive any rights from this.

The documentation can be found here.

Data Quality Rules for the Financial Assessment Framework (FTK) are not supported in this version. If you are interested in this please contact DNB.

1.1 License

The Data Quality Rule repository is free software under MIT/X license.

1.2 Features

Here is what the package contains:

- (Statistical) validation rules for Solvency 2 and VNS
- Source code to extract csv- and Pandas pickle-files from XBRL instance files
- Source code to evaluate validation rules on Pandas pickle-files

- Command line interfaces and notebook tutorials to get you started

Currently, the patterns and rules published here contain a limited number of EIOPA-validation rules and the code to evaluate the complete set of EIOPA-validation rules is not yet included.

CHAPTER 2

Installation

We provide two installation methods: online and offline installation. For the online installation you need a connection to internet to download the required packages. For the offline installation you need to download a limited number of files (specified below) and then you can install everything without a connection to internet. After installation of the repository, you need to download the taxonomies and examples files you want to work with. We provided a script to download these files and put them in the correct location.

For both methods we recommend to have Anaconda (> 5.3.1) installed.

2.1 Online installation

Clone the project:

```
git clone https://github.com/DeNederlandscheBank/data-quality-rules.git
```

Then start with a clean environment:

```
conda create -n your_env_name python=3.6
```

And activate the environment:

```
conda activate your_env_name
```

Make sure you are in the root of the cloned project. Install the code and the required packages:

```
pip install -e .
```

2.2 Offline installation

We included all the required packages in the project, so you should be able to do an offline installation.

To do an offline installation you need some files from the internet downloaded in advance:

- the zip file with the data-quality-rules repository from https://github.com/DeNederlandscheBank/data-quality-rules.git;
- the zip files with the taxonomy and example instances from the EIOPA website (https://dev.eiopa.europa.eu/Taxonomy/Full/2.7.0/S2/EIOPA_SolvencyII_XBRL_Taxonomy_2.7.0_hotfix_with_External_Files.zip; and https://dev.eiopa.europa.eu/Taxonomy/Full/2.7.0/S2/EIOPA_SolvencyII_XBRL_Instance_documents_2.7.0.zip)
- the zip files with the taxonomy and example instances from the DNB website (https://www.dnb.nl/media/fivpjdpj/vns_taxonomy_1-1-0.zip; and https://www.dnb.nl/media/o10oswji/vns_sample_instances_1-1-0.zip)

2.2.1 Install data-quality-rules repository

Extract the zip file from the data-quality-rules repository to the desired location.

Then start with a clean and empty environment:

```
conda create -n your_env_name --offline
```

And activate the environment:

```
conda activate your_env_name
```

Install the following packages:

```
conda install pkgs/vc-14-0.tar.bz2

conda install pkgs/vs2015_runtime-14.0.25420-0.tar.bz2
```

Then install the following packages:

```
conda install pkgs/python-3.6.6-he025d50_0.tar.bz2

conda install pkgs/pip-18.1-py36_1000.tar.bz2

conda install pkgs/setuptools-40.6.3-py36_0.tar.bz2
```

(if you get an error you need to copy the required packages from internet)

Make sure you are in the root of the cloned project. Then install the project with the packages in pkgs/.:

```
pip install -e . --no-index --find-links pkgs/
```

2.2.2 Copy taxonomy and instance files

Copy the Solvency 2 and VNS XBRL taxonomy file and the Solvency 2 and VNS XBRL instance examples (both zip files) to the directory data/downloaded files.

2.3 Taxonomy and example instance files

For Solvency 2 and VNS execute (in the root of the project):

```
python src/solvency2_data.py
```

This downloads the Solvency 2 and VNS XBRL taxonomy and the corresponding example instance files and extracts them in the proper directories.

For FTK, execute (in the root of the project):

```
python src/ftk_data.py
```

This downloads the FTK taxonomy and the corresponding example instance files and extracts them in the proper directories.

CHAPTER 3

Usage

The Data Quality Rules repository allows you to evaluate the data quality of Solvency 2, VNS and FTK supervisory reports before you submit them to DNB. You can either apply the rules directly to the XBRL-instance that you want to submit to DNB, or you can apply the rules at an earlier stage in your reporting process.

If you want to apply the rules on internal data sets, then you have to make sure that the format of the data satisfies the data format requirement set out in *Data format requirements*.

If you want to apply the rules to the XBRL-instance you first need to convert the instance to Pandas files (see Converting XBRL instances to Pandas and CSV). The resulting files will be in the correct data format. We recommend to start with this option.

Evaluation of the rule sets is done by a simple command line interface. We added Jupyter notebooks with a step-by-step description of all parts of the code, in case you want to understand the separate steps and include some steps in your internal reporting process.

Currently we have three rules sets available:

- Additional rules for Solvency 2 reports published by DNB (see Evaluating additional Rules)
- Rules for financial data (individual assets and derivates) (see Evaluating financial Data Rules)
- Patterns to compare two subsequent reports (see Evaluating rule sets for report comparison)

All rule sets are evaluated with DNB's data-patterns package. With this package we generated most rule sets that we have published here (based on reports that have been submitted to us).

3.1 Converting XBRL instances to Pandas and CSV

We use a command line interface to convert XBRL instance to Pandas and CSV. You can run the XBRL to Pandas and CSV converter with:

```
python src\instance2csv.py
```

The script will ask for the desired taxonomy, instance-file, output directory, and whether you want verbose column names instead of rc-codes (row-column codes). To evaluate rules you need the templates with the rc-codes. You can also run the script with command line arguments –taxo, –instance, –output and –verbose_labels.

Make sure you use the corresponding version of the taxonomy for your instance.

For each template in the XBRL instance the results are exported to a Pandas pickle-file and a csv-file. A Pandas pickle-file maintains the correct indices, whereas the csv does not, so if you want to access the data read the pickle-files.

The csv-files and the pickle-files are stored in a subdirectory identical to the name of the XBRL-instance (without extension).

All closed axis tables in the XBRL-instance are combined and stored in one pickle-file in the subdirectory where all pickle-files are stored (with the name of the XBRL-instance).

The easiest way to access the data of a separate template is to read the corresponding pickle-file with:

```
df = pd.read_pickle(filename)
```

3.2 Evaluating additional Rules

To run the additional rules use:

```
python solvency2-rules\apply1.py
```

You can run DNBs additional Rules for the following Solvency II reports

- Annual Reporting Solo (ARS); and
- Quarterly Reporting Solo (QRS)

The command line interface will ask the entrypoint (ARS or QRS), the directory where the reports (in pickle-files) are stored, the output type (confirmation, exceptions or both) and the output directory. You can also run the script with command line arguments –entrypoint, –report_dir, –output_type and –output_dir.

We distinguish 2 types of tables

- With a closed-axis, e.g. the balance sheet: an entity reports only 1 balance sheet per period
- With an open-axis, e.g. the list of assets: an entity reports several ‘rows of data’ in the relevant table

To evaluate the patterns we use a PatternMiner-object (part of the data_patterns package), and run the analyze function.

3.3 Evaluating financial Data Rules

To run the financial data rules use:

```
python solvency2-rules/apply2.py
```

The command line interface will ask the directory where the reports (in pickle-files) are stored and the output directory. You can also run the script with command line arguments –report_dir, output_type and –output_dir.

With output_type you can choose to output confirmations only, exceptions only, or all results.

If, given the output_type, no output was generated (for example, no exceptions when the output type is exceptions only), then no Excel output file is generated.

The rules are related to the following tables:

- S.06.02.01.01 (Information on positions held)
- S.06.02.01.02 (Information on assets)
- S.06.02.01.01 (Information on positions held) and S.06.02.01.02 (Information on assets)
- S.08.01.01.01 (Information on positions held) and S2.08.01.01.02 (Information on derivatives)
- S.08.01.01.02 (Information on derivatives)

3.4 Evaluating rule sets for report comparison

The idea of the report comparison rules is a bit more difficult than the additional validation rules from DNB and the financial data validation rules. The report comparison rule sets evaluate differences between two report sets (for example ARS-2020 and ARS-2019), whereas the latter rule sets evaluate one single report set (for example QRS-2020-Q1).

The goal is to detect differences in whether data points are reported (datapoint that were included in one period and not in the other) and to detect significant changes in the values of data point between two periods. As such it is not unusual that some data points are included in one period and not in another, and that some data points change significantly between two periods. Because of that we only included datapoints for which it is highly unusual that they are included in one period and not in the other (< 5% of previous reports), and datapoints for which it is highly unusual (< 5% of previous reports) that they changes significantly over two periods (> 10% change).

You can compare two quarterly reports or two annual reports, but you cannot compare a quarterly report with an annual report, even if they have corresponding data points.

To run the additional rules use:

```
python solvency2-rules\apply3.py
```

The command line interface will ask the rule set that you want to apply (compare two QRS-reports or compare two ARS-reports), the entity category (Schade, Herverzekeraar, Leven), the two directories where the two reports are located and the output directory. You can also run the script with command line arguments –rule_set, –entity_category, –report_dir_1, –report_dir_2, output_type and output_dir.

With output_type you can choose to output confirmations only, exceptions only, or all results.

You cannot test these rules with the example instances provided by EIOPA because the instances of subsequent periods contain different LEI-codes.

3.5 Rule thresholds

Currently, we have set the threshold on zero decimals, which means that a rule is satisfied if the difference between the expected value and the reported value is lower than 1.5e0. This threshold is set at the level of the rule. This is a slightly different approach than the one applied in XBRL validation rules, where thresholds are set at the level of separate data points. This means that it is possible that an exception to a DQR rule is triggered, where an XBRL rule is not triggered because it applies higher thresholds.

3.6 Logging

Logging is performed for each rule set and is set to **logging.INFO**. Output is written to results\rule-set-1.log, results\rule-set-2.log and results\rule-set-3.log, depending on the rule set that is evaluated. Logging of the data-patterns

package is stored in this file, so you can see the result of each rule and pattern evaluation. Log level is currently not an input but you can change the level in the apply-files in solvency2-rules.

3.7 The format of the patterns and rules files

The input with the rule sets described above are stored in the same format in Excel files. The columns are described here:

- **pattern_id**: the name of the pattern or rule that was applied. A pattern or rule can apply to more than part of the report, but has the same form.
- **cluster**: the group or cluster to which the pattern or rule applies (optional), for example life, non-life or reinsurance.
- **pattern_def**: the definition of the pattern. The definition uses a simply syntax with references to the datapoints that should be relatively easy to read.
- **support**: the number of occurences that satisfy this pattern or rule in reports that were previously submitted to DNB.
- **exceptions**: the number of occurences that do not satisfy this pattern or rule in reports submitted to DNB.
- **confidence**: the support divided by (support plus exceptions). This is an indicator for how exceptional this pattern or rule is. If the confidence is one, then the pattern or rule is in all cases satisfied. If the confidence is lower than one then this could point to an error in the data or an unusual but acceptable circumstance that led to this exception. Only patterns with very high confidences are published in this repository.
- **pattern_status**: the status of the pattern or rule, i.e. blocking taxonomy rule, non-blocking taxonomy rule, validation rule or statistical validation rule.

You can find the documentation of the data-patterns package here.

3.8 The format of the results files

The output of the evaluation of the rule sets are all stored in the same format in Excel files.

- **First columns** describe the index of the report
- **result_type**: true if the pattern or rule is satisfied, false if the pattern or rule is not satisfied
- **pattern_id**: the name of the pattern or rule that was applied. A pattern or rule can apply to one or more parts of the report, but has the same form.
- **cluster**: the group or cluster to which the pattern or rule applies (optional), for example life, non-life or reinsurance.
- **support**: the number of occurrences that satisfy this pattern or rule in the report.
- **exceptions**: the number of occurrences that do not satisfy this pattern or rule in the report.
- **confidence**: the support divided by (support plus exceptions).
- **pattern_def**: the definition of the pattern. The definition uses a simply syntax with references to the datapoints that should be relatively easy to read.
- **P values**: the values of data points in the left hand side of the pattern or rule (in case of an if-then rule: the if part)
- **Q values**: the values of data points in the right hand side of the pattern or rule (in case of an if-then rule: the then part)

3.9 Data format requirements

If you want to apply the rules on internal data sets, then you have to make sure that the data is in the correct format.

3.9.1 Solvency 2

- the template name follows the standard Solvency 2 code, for example S.02.01.02.01 and S.28.02.01.02;
- the file names of the individual templates is the template name plus an extension (.csv or .pickle), for example S.01.02.07.01.pickle;
- the file name of all closed axes templates combined is the instance file name plus an extension, for example qrs_270_instance.pickle (the example instance for qrs);
- the column names and the index names for all templates have the following format: {reporting template name},R????,C???? or {reporting template name},C????, depending on the definition; for example S.02.01.02.01,R0030,C0010 or S.06.02.01.01,C0040;

3.9.2 VNS

- the template name follows the standard VNS code with prefix FTK, for example FTK.T2A or FTK.T4B;
- the file names of the individual templates is the template name plus an extension (.csv or .pickle), for example FTK.T2A.pickle;
- the file name of all closed axes templates combined is the instance file name plus an extension, for example Verzekeraars Nationale Staten Sample Fixed Interval Instance VNS-JR 1.pickle
- the column names and the index names for all templates have the following format: {reporting template name},R???,C???; for example T2A,R030,C010 or T0,R010,C010;

3.9.3 FTK

- the template name follows the standard FTK code with prefix FTK, for example FTK.K101-1 or FTK.K209B;
- the file names of the individual templates is the template name plus an extension (.csv or .pickle), for example FTK.K101-1.pickle;
- the file name of all closed axes templates combined is the instance file name plus an extension, for example DNB-NR_FTK-2019-06_2019-12-31_MOD_FTK-BEL.pickle (the example instance for FTK-BEL);
- the column names and the index names for all templates have the following format: {reporting template name},R???,C??? or {reporting template name},C???, depending on the definition; for example FTK.K101-1,R010,C010 or FTK.K209B,C150;

CHAPTER 4

Contributing

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/DeNederlandscheBank/data-quality-rules/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/DeNederlandscheBank/data-quality-rules/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.

CHAPTER 5

Authors

Development Lead

Willem Jan Willemse <w.j.willemse@dnb.nl>
Expert Centre on Data, Algorithms & Business models
Division Insurance Supervision
De Nederlandsche Bank (DNB)

Contributors

- Robert Verschuren (DNB)
- Annick van Ool (DNB)
- Richard Lieverse (DNB)
- Jan Huiskes (DNB)

Your name could be here, see how to contribute in the text above.

CHAPTER 6

History

6.1 0.1.0 (2020-09-07)

- Development release.

6.2 0.2.0 (2020-09-30)

- Package requirements added to setup.py (all required packages are now on Python Package Index (pypi.org))
- Unittests added for transforming XBRL to CSV (Solvency 2 and FTK) (in tests/)
- TravisCI and tox installed for automatic testing (only online distribution)
- Tutorial for transforming XBRL to CSV adapted to read taxonomies as zip-files instead of extracted files
- FTK script added to download automatically FTK taxonomy and test instances
- FTK bug fix: index column names include FTK. prefix
- Source code for downloading taxonomies moved to src
- README.rst adapted to reflect changes

6.3 0.3.0 (2020-10-19)

- Updated and simplified tutorial Evaluate DNBs Additional Validation Rules
- Added Tutorial to evaluate FTK validation rules
- Added Solvency 2 S.06 (assets) validation rules and updated S.08 (derivatives) validation rules
- Added FTK K208 (assets) and FTK K210 (derivatives) validation rules
- Deleted code to convert XBRL to HTML from Tutorial

- Added code to construct one large Dataframe with all templates with closed axes to Tutorial
- Added description of Data format requirements to README.rst
- README.rst adapted to reflect changes

6.4 0.4.0 (2020-11-16)

- Added Solvency 2 statistical validation rules to evaluate two periods for ARS and QRS
- Added FTK statistical validation rules between to evaluate two periods for FTK-JS and FTK-BEL
- Added tutorials for statistical validation rules
- Added datapoints for Solvency 2 group reports (ARG and QRG)
- Added taxonomies for Solvency 2 (versions 2.5.0, 2.4.0 and 2.3.0) and FTK (versions 2.1.0, 2.0.0, 1.0.3, 1.0.2, 1.0.1, 1.0.0, 0.9.0)
- README.rst adapted to reflect changes

6.5 0.4.1 (2020-11-16)

- Bug fix for evaluation of additional rules for group reports

6.6 0.4.2 (2020-11-26)

- Bug fix for tutorial to evaluate two periods

6.7 0.5.0 (2021-4-29)

- Simplified repo by adding command line interfaces for all rule evaluations
- Added documentation on readthedocs.org
- Updated Solvency 2 taxonomy 2.5 and example instance files
- Updated DNB ruleset: 2021_04-01_set_aanvullende_controleregels_solvency2
- Improved logging

6.8 0.5.1 (2021-5-17)

- Some minor bug fixes
- Release version for DQR 2.0

6.9 0.6.0 (2022-11-14)

- Added generator of closed axis datapoints
- Extended pattern functionalities for SUM, SUBSTR, (NOT) IN
- Included vector functionality with wildcard #
- Updated documentation on readthedocs.org
- Updated package requirements
- Updated Solvency 2 taxonomy 2.6 and VNS taxonomy 1.1.0, with example instance files
- Updated DNB ruleset: 2022_02_23_set_aanvullende_controleregels_solvency2

6.10 0.6.1 (2022-12-15)

- Updated Solvency 2 taxonomy 2.7 and example instance files
- Included DNB ruleset: 2022_02_23_set_aanvullende_controleregels_solvency2

CHAPTER 7

Indices and tables

- genindex
- modindex
- search