Comparing Data Quality Dashboard results from consecutive ETL iterations: two new visualizations and one utility script

Elena G. Lara[†], Maxim Moinat[†]
[†] The Hyve, Utrecht, The Netherlands

Background

Data quality assessment of an observational health data set is an important aspect when deciding whether the data is suitable to answer a selected research question[1]. Accordingly, in 2019 the Data Quality Dashboard (DQD)[1] was introduced to the OHDSI (Observational Health Data Sciences and Informatics) toolkit. The DQD performs a series of checks on the converted data set that may pass or fail, depending on a pre-specified threshold.

The DQD has been widely used to assess the quality of an individual OMOP CDM (Observational Medical Outcomes Partnership, Common Data Model) instance. However, in practice, the conversion to OMOP requires multiple ETL (extract-transform-load) or source data iterations, between which the quality is expected to vary. In order to facilitate the interpretation of those differences, we developed two visualizations comparing the DQD results. Furthermore, the thresholds for DQD checks are often changed during the ETL iterations. However, the tables containing these thresholds are wide and contain additional settings in them. In order to avoid having to edit them manually, we developed a utility to simplify the editing of these thresholds. We applied these newly developed methods to the UK Biobank ETL project, part of the European Health Data Evidence Network (EHDEN) COVID19 rapid data partner call[2], and in collaboration with University College London (UCL).

Methods

We developed two scripts to create visualizations that yield more insight from the DQD results. The first shows the concept mapping coverage across all OMOP domains in a bar plot. The information is taken directly from a DQD results file and produces one plot per DQD result. The second visualization script produces a scatterplot comparing the results of the two DQD runs. It selects the checks for which the percentage of records that satisfy that check have changed between iterations, as well as their fail or pass status. Lastly, we developed a utility script that allows the users to customize the DQD check thresholds. We created a more human friendly format for editing new thresholds (Table 1). Our script combines this new table with the original DQD check tables to produce a customized thresholds file accepted by DQD's check execution.

These three extensions are built on top of the current DQD GitHub repository. Currently, the code described above is available in a fork on The Hyve's Github organization[3]. We used the R libraries jsonlite[4], in order to access the DQD thresholds and results tables, and dplyr[5] to manipulate those tables. The UK Biobank ETL was developed using delphyne[6], a Python framework, and we produced DQD results at four stages to evaluate the data quality. We report here on the comparison between the second and third iteration.

Level	Check Name	cdmTable Name	cdmField Name	fkTable Name	Concept Id	Unit Concept Id	Thres-hold	Notes
Field	*Plausible ValueLow*	*PERSON*	*Year_of_ birth*				*100*	*"one note"*
Field	*isForeign Key*	*MEASURE-MENT*	*person_id*	*PERSON*			*1*	
Concept	*Plausible ValueLow*	*MEASURE-MENT*	*MEASURE-MENT_ CONCEPT_ ID*		*2212333*	*8554*	*56*	*"note in another table"*

***Table 1**. Example of the information needed to find the specific checks in the DQD threshold files. This information can be found in the description of each check through the DQD web interface.*

Results

During the UK Biobank ETL project, we compared the DQD results of two successive ETL iterations. The percentage of records passing the checks had modestly improved in the second iteration (Fig. 1a). However, two outliers on the top of the graph show that a few checks had a worse performance. For example, the completeness of the *visit_detail_id* in the *device_exposure* table went from 0% to 100% of records failing the check. While the checks still passed, this change of percentage did prompt us to investigate and update the ETL accordingly.

The concept coverage barplot provided additional insights about the completeness of the code mappings. This ETL iteration achieved a high coverage throughout all domains and units (Fig. 1b) in terms of records mapped to standard concepts. The number of unique terms mapped was low for measurement and observation units (1.82% and 1.00%) and for measurement (19.42%). The coverage moderately increased from the previous iteration (Supplementary figure).

The *editThresholds* utility facilitated managing the thresholds for subsequent runs of DQD on the converted UK Biobank data. It also allowed us to more easily identify which thresholds have been changed compared to the default values, as all the customized thresholds are in a separate file.

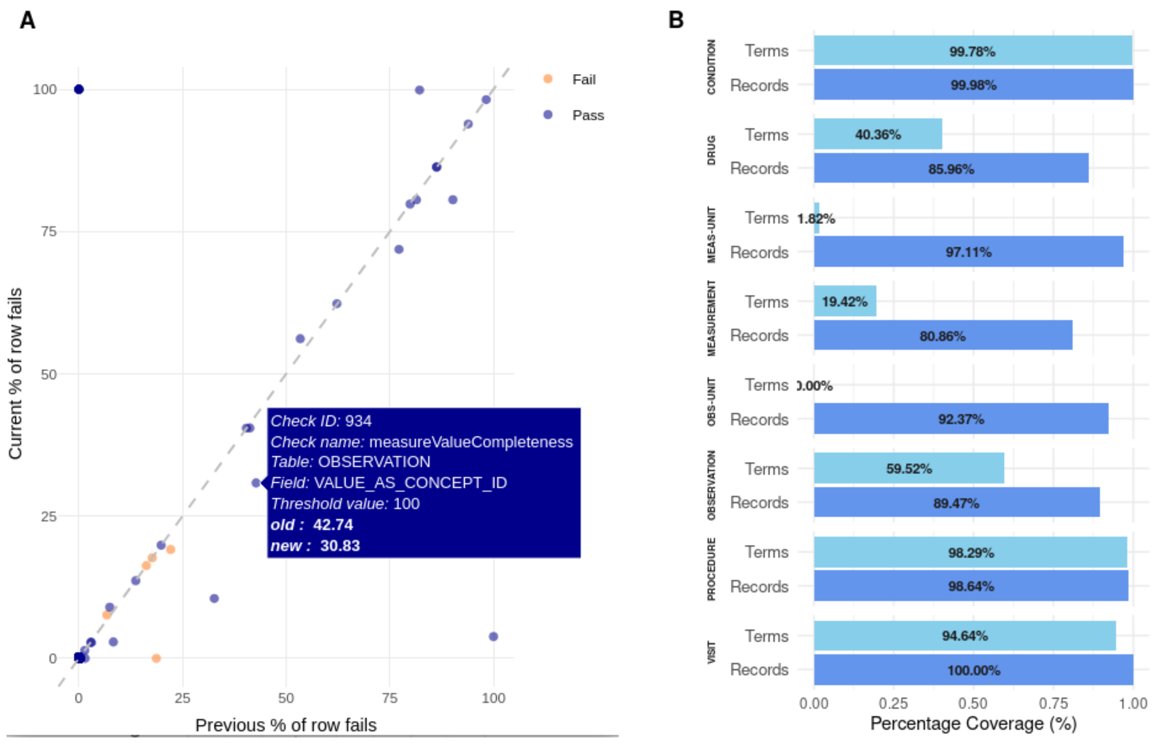

***Figure 1**. **(a)** Scatter plot comparing the DQD outputs from two ETL iterations. Each dot represents one check that has a different percentage of row fails between the iterations. This percentage of row fails is marked in the x-axis for the earliest run, and in the y-axis for the latest run. Thus, every point below the diagonal dashed line will have improved, and vice versa. **(b)** Barplot for the mapping coverage in an ETL. In light blue: the percentage of terms used; in darker blue: the coverage of all records using those concepts.*

Conclusions

In conclusion, the described scripts facilitate the comparison between the DQD results of two ETL iterations. The *compareResults.R* does a direct comparison between the checks, which allows to find all the cases that have changed between runs. The *conceptMappingCoveragePerDomain.R* script displays the quality of concept mappings. By placing the output from two different iterations next to each other, one can detect the changes in the concept mapping coverage across data domains. Additionally, our new utility makes it easier to edit the check thresholds for a new run. Our future work will be focused on merging these scripts into the main DQD repository, making them easily accessible to the OHDSI community. Finally, the new figures could be incorporated into the DQD Shiny application to have them side-by-side the main dashboard.

References

1. EHDEN COVID19 Rapid Data Partner Call, April 2020, retrieved May 2021. https://www.ehden.eu/open-calls/04-2020-covid19-data-partner-call/
2. Blacketer C, Defalco F, Ryan P, Rijnbeek P. Increasing trust in real-world evidence through evaluation of observational data quality. *medRxiv 2021*.
3. Branch of the DataQualityDashboard on Github (accessed June 17th 2021), https://github.com/thehyve/OHDSI-DataQualityDashboard/tree/utilities/extras
4. Ooms J, Lang DT, Hilaiel L. jsonlite: A simple and robust JSON parser and generator for R. 2014.
5. Wickham H, François R, Henry L, Müller K. dplyr: A grammar of data manipulation. 2018.
6. Delphyne user documentation (accessed June 15th 2021), https://delphyne.readthedocs.io/en/latest/