Consistency variables dashboard documentation

Eurostat grant SMP-ESS-2023-EBS-IBA; 2023-NL-EBS

Statistics Netherlands, 27 June 2025

Frank Aelen, Joey Meijers, Koen van Noortwijk, Maarten Poeth, Ramon Reinders, Ronald van der Stegen

In addition to multisource automatic data editing, some manual data editing will always be necessary. In order to do this most efficiently, in an earlier pilot study at CBS (Vaasen-Otten et al., 2022[1]) score functions were developed to aid top-down analysis by identifying potential influential errors. Similar score functions can also be used for top-down analyses across statistics, to determine potential influential inconsistencies between statistics.

A typical score function for comparing statistics is the difference in value between the two statistics (or between a statistic and administrative data), multiplied by a weight reflecting how often the unit contributes to the compilation of output-level figures – for example, as a donor in imputations or through a sampling weight. The inclusion weight often serves as a good approximation. Finally, the result is divided by an aggregate total to account for the impact of the inconsistency at the output level.

In the pilot study, we implemented these score functions in a consistency dashboard starting from the unit-level scores, which we then aggregated to higher levels to indicate whether there are significant inconsistencies within an (output) aggregate. This way, it is possible to quickly and efficiently drill down from the aggregate levels to the most influential inconsistencies at the unit level. The dashboard was subsequently used by analysts to gain some experience with real production data from multiple statistics. Results of the pilot study were highly encouraging, prompting us to move forward with incorporating it into regular production, also for more statistics than those used in the pilot. However, expansions to the dashboard are needed for that.

The goal of this part of work package 1 of the Eurostat grant, is to further enhance the consistency dashboard, making it more flexible and more broadly applicable. The objective is to enable its use with a wide range of relevant business statistics or other data sources, regardless of the underlying unit types such as enterprise groups, enterprises, local units, VAT units, etc. Additionally, the dashboard should support different levels of aggregate output, including NACE sections, divisions, groups, classes, or other relevant output categories. We also want to ensure that the dashboard aligns with the automatic editing process, for instance by visualizing influential changes made by automatic editing within and across statistics.

This document is set up to describe briefly *how it works* and what is needed to start your own consistency dashboard.

General approach

In order to get the correct output data for the dashboard, a couple of steps need to be taken. In general there is a set of input data which requires statistical data, metadata and configuration data. The statistical data is of course the base data on which the analysis can be done. The metadata in this case is the data belonging to a statistical unit, like company name, NACE mapping etc. The input data need to be given by the user of the dashboard.

Then there is a set of queries, which will use this input data to create the output layers for the dashboard. The queries are written in T-SQL (SQLServer 20). In short, the queries will combine the input and calculate differences and scores for all the units that are available in 2 or more statistics. The queries will create views for the four layers that are used in the R Shiny dashboard. We chose to write all calculations and joins in T-SQL for readability and speed. It is of course possible to rewrite these scripts in other languages.

The query files are supported with comments in the files, declaring what happens in every part of the query.

The queries create four views and a subview, which are the four different layers for the dashboard. The dashboard is written in R Shiny and displays the views with some more functionality, such as coloring for different scores, clicks to more detailed levels of data in order to do a top-down analysis on the data. The R Shiny tools contain docstrings which are created with Roxygen2.

[1] A. Vaasen-Otten, F. Aelen, S. Scholtus and W. de Jong (2022), Towards a New Integrated Uniform Production System for Business Statistics at Statistics Netherlands: Quality Indicators to Guide Top-down Analysis. UNECE Expert Meeting on Statistical Data Editing, 3-7 October 2022 (virtual).

Data flow

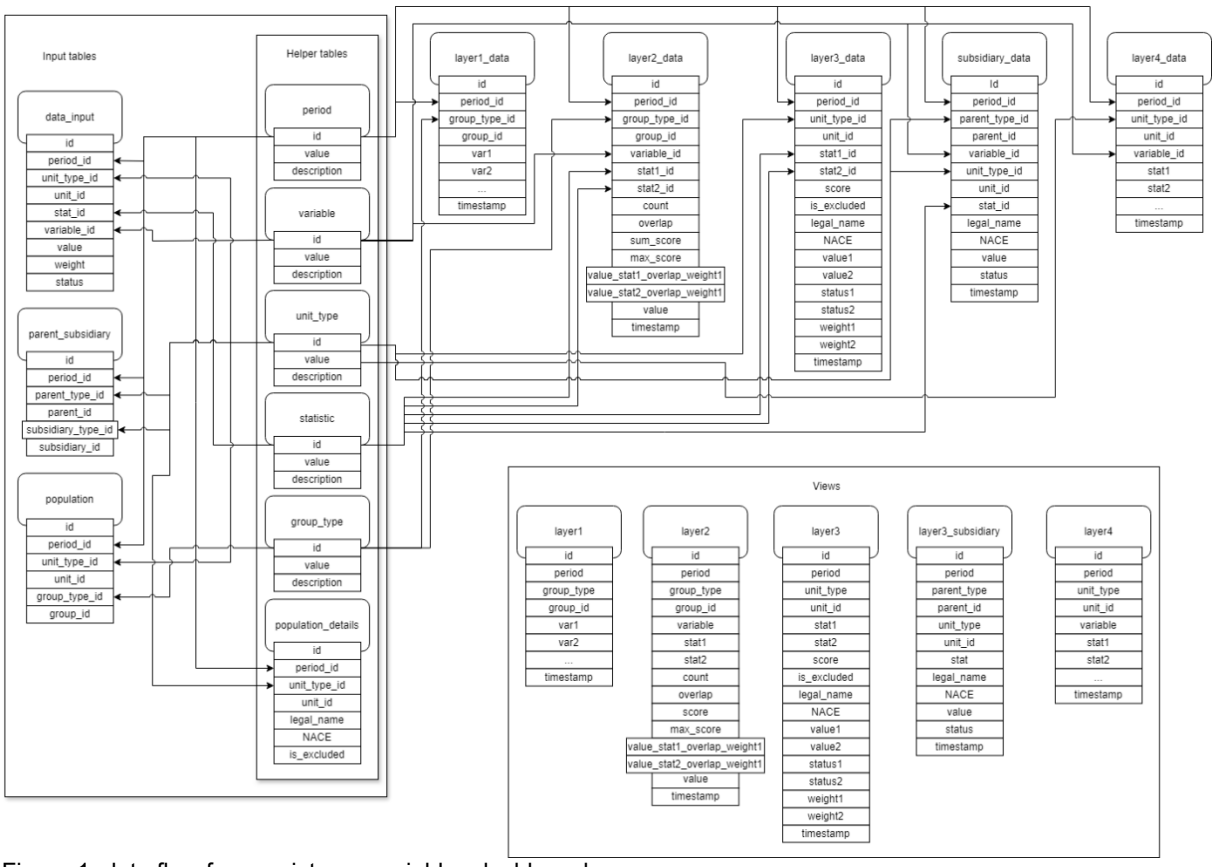

Figure 1: data flow for consistency variables dashboard

To set up the application, the following steps need to be followed:

1. Creating the database.
   firstly, the 1_create_db_table.sql script needs to be run, to create the database structure necessary for the application to run. The resulting database is shown in figure 1.

2. Inserting the input data into the database.
   Secondly the necessary data needs to be inserted into the tables contained in the 'Input tables' box in figure 1. The method for inserting is largely left undefined, as it fully depends on the statistical institute which data needs to be loaded into the dashboard and how that data is provided. It is however necessary to fill the input tables and follow their structure, as can be seen in figure 1, since the further manipulation of the data depends on said structure.

3. Processing the data to be ready for display in the dashboard.
   After inserting the input data, a few provided scripts need to be run, namely: 2_fill_layer1_to_4.sql and subsequently, 3_create_views.sql. The views need to be created after the tables have been filled, due to technical reasons (i.e., a pivot function is used). These scripts create the content for each of the 'layerX_data' tables and some resulting views, which will be used for querying by the dashboard.

4. Running the dashboard.
   The dashboard can then be run, which will query dynamically on the created views.

Input data

Some obligatory input data (values for two or more statistics) is needed next to some helper data. The input that is needed, is in tables in a relational database. The actual input tables are described below and can be seen on the left part of the data flow figure:

data_input			
Field	**Type**	**Key**	**Description**
id	int	PK	
period_id	int	FK	
unit_type_id	int	FK	
unit_id	nvarchar(128)		
stat_id	int	FK	
variable_id	int	FK	
value	float		
weight	float		weight reflecting how often the unit contributes to the compilation of output-level figures – for example, as a donor in imputations or through a sampling weight
status	nvarchar(128)		Status of the value, e.g. RAW, EDIT or IMP

parent_subsidiary			
Field	**Type**	**Key**	**Description**
id	int	PK	
period_id	int	FK	
parent_type_id	int	FK	
parent_id	nvarchar(128)		
subsidiary_type_id	int	FK	
subsidiary_id	nvarchar(128)		

population			
Field	**Type**	**Key**	**Description**
id	int	PK	
period_id	int	FK	
unit_type_id	int	FK	
unit_id	nvarchar(128)		
group_type_id	int	FK	
group_id	nvarchar(128)		

And the following helper tables:

period			
Field	**Type**	**Key**	**Description**
id	int	PK	
value	nvarchar(128)		
description	nvarchar(128)		

variable			
Field	**Type**	**Key**	**Description**
id	int	PK	
value	nvarchar(128)		Name of (consistency) variable
description	nvarchar(128)		

unit_type			
Field	**Type**	**Key**	**Description**
id	int	PK	
value	nvarchar(128)		e.g. enterprise group, enterprise, KAU,…
description	nvarchar(128)		

Statistic			
Field	**Type**	**Key**	**Description**
id	int	PK	
value	nvarchar(128)		Statistic
description	nvarchar(128)		

group_type			
Field	**Type**	**Key**	**Description**
id	int	PK	
value	nvarchar(128)		Group name, e.g. NACE division, group,…
description	nvarchar(128)		

population_details			
Field	**Type**	**Key**	**Description**
id	int	PK	
period_id	int	FK	
unit_type_id	Int	FK	
unit_id	nvarchar(128)		
legal_name	nvarchar(128)		Company name
NACE	nvarchar(128)		
is_excluded	bit		Excludes unit from score calculation on layers 1 and 2 when set to TRUE.

Additional input

In the SQL-script ***2_fill_layer_1_to_4.sql*** some additional user input is needed:

1. *Source denominator*
   For each variable the source denominator for the score calculation has to be indicated, e.g. *ST_ABC* in the example below.

   ```
   INSERT INTO #source_denominator
   VALUES
         ('VAR_A', NULL,          NULL, NULL)
       , ('VAR_B', 'ST_ABC',      NULL, NULL)
       , ('VAR_C', NULL,          NULL, NULL)
       , ('VAR_D', NULL,          NULL, NULL)
   ```

   If the denominator is empty, the totals from stat1 are used; otherwise, the totals from the specified alternative sources are applied. Users must fill in the variables and alternative denominators (in the first two columns) themselves in the table #source_denominator. The corresponding variable_ids and statistic_ids will be completed automatically.

2. *Maximum score*
   DECLARE @MAXSCORE DECIMAL(10,4) = 2.0
   Score cutoff value: applied to prevent disproportionately large and non-informative scores that may occur when the score's denominator approaches zero. Default is 2.0.

SQL-scripts

The project comes with 3 sql script files, labeled '1_create_db_table.sql, '2_fill_layer_1_to_4.sql' and '3_create_views.sql'.

The first script generates the database tables and makes sure the necessary structure is present in the database. This script needs to be run before actually inserting any input data (as described in the data flow section).

The third script needs to be run after all data is inserted in the database and all manipulations have been done, and makes sure the views are created that need to be present for the dashboard application to query on.

The second script is where the data preparation and processing happens and the input data is transformed such that it conforms to our data model in which we support 4 'layers' of information. These layers each represent a depth layer in the dashboard, in which layer1 is the overview layer from which to drill down up to the most detailed unit information on layer 4. Each of the layers has a dedicated table in the database containing information relevant for this layer as well as links to information stored in helper tables that will need to be presented in the dashboard.

The script is structured in such a way, that first some data manipulations are done and stored in intermediary temporary tables, after which the actual content of the layerX_data tables is calculated and inserted, based on the information in the input tables, as well as the intermediary temporary tables.

When running the provided scripts, please note that you will need to make sure the input tables are filled (after creating the database tables) and there is no script provided to do so, since it will depend on the particular input source that is used. A working example is provided in this project to give some idea about the required data. There is a directory *example_data* that comes with the repository with a ready-to-use SQLite database and the scripts to set up a SQL Server database. It is your choice for one of those.

The scripts themselves are designed for Microsoft SQL Server (version 20), but users are responsible for adapting them to other database types if needed.

Output data and views

When data preparation and processing is done, the output is stored in tables and there are five views that can show the output data in different layers. Since the goal is to analyze data in a top down manner, the first layer (*layer1*) will show data in the most aggregated (grouped) form: an overview of maximum scores within an aggregate group (e.g. some NACE-group) for all (consistency) variables. The second layer (*layer2*) shows the maximum scores and summed scores for variables that are compared between two statistical sources. The third

layer (*layer3*) shows the scores for a variables, for all units within the group and for a combination of two compared statistics, including the values, legal names and some extra information. If there is a parent-subsidiary relation between two statistics, then only the score and (summed) values of the parent will appear in this view. The values and extra information of the underlying subsidiaries is shown in the view *layer3_subsidiary*. In *layer4* there is an overview of all values for all variables for a certain unit.
The views are designed in such a way that they can be directly loaded into the R Shiny dashboard, without any further processing.

Dashboard

The dashboard (Graphical User Interface) uses the views from the database directly. The first layer is an exact copy of database *later1*. In the dashboard, colors are added that emphasize high score values and there are drop downs to choose the period and the group type. After clicking on a combination of group and variable, layer 2 will be opened, which is an exact copy of database *layer2*, but then filtered on the chosen group and variable. Colors are again added to emphasize large score values. When clicked on a combination of two statistics in layer 2, layer 3 is opened which is again an exact copy of database *layer3*, but filtered on the chosen combination of statistics and the in layer 1 chosen variable. If a combination of statistics is chosen where the first statistic is on parent level and the second statistic on subsidiary level, the rows can be expanded to show the data of *layer3_subsidiary*, filtered for that specific parent_id. Layer 4 in the dashboard a filtered overview from the database. This is the filtered version of *layer4* on the chosen unit_id.