Design of a generic machine-readable validation report structure

Mark van der Loo and Olav ten Bosch
Statistics Netherlands

Version 1.0.0 August 15, 2017

Colofon
Version 1.0.0 August 15, 2017
Deliverable of Tasks 14/2 and 15/2 of the ESSnet Validat Integration (`https://goo.gl/hEGdbo`)
European Union Grant agreement N° 11103.2016.01-2016.739
Typeset with LaTeX

Contents

1 Introduction

Data validation is at the core of every production chain in official statistics. Whether it is the input received from a survey, a bunch of data records transferred from an administrative source or the result from some internet scraping, data must be checked against our expectations in order to process it and turn it into reliable statistics. For a more detailed explanation of data validation in general and the principles we identified in validation, we refer to the handbook of data validation from the ESSnet Foundation, by Di Zio et al. (2015) and the validation principles written down by the ESS validation task force, see ESS (2017, Chapter 4).

Standardisation is important in official statistics and this also applies to validation processes. Especially in the case of cross-organisation validation, where both a data producer and a data receiver check the same data against some commonly agreed validation rules, standardisation is crucial to prevent different interpretations of the results. The more harmonised validation reports are, the better the understanding across organisations is. This has been recognized by the ESSnet on Validation project (Validat Integration). A Work Package (WP2) was defined to attack this issue. This report is one of the deliverables of Work Package 2. It contains the result of the work carried out by various project partners and focusses on the standardisation of the output of a validation process: the *validation report*.

Obviously we are interested in the question what information can and should be included in a validation report and the optimal way to express this information. In this document we develop a *generic* structure for expressing validation results. We have the ambition to develop a validation report structure that can be used in *any validation task* in *any organisation* in any *statistical domain*, for validation of *microdata* as well as *aggregated data*. To make it applicable in machine to machine communication contexts as well as in human contexts we design a machine-readable as well as a human readable format. This report focusses on the machine-readable version.

The approach we have taken is a combination of a top-down approach and a bottom-up approach. In the bottom-up approach a number of example validation reports from member states and Eurostat were collected and studied to identify common elements. This resulted in a long-list of validation report elements categorized into several classes such as rule metadata, process metadata, aggregates etc. This led to some thinking about the basic concepts that are used in validation reports across the ESS. We used the results from the bottom-up approach to develop a more formal top-down approach expressed in this report. Step by step we built a validation report structure that is generic enough to be used in a wide range of validation contexts in many institutes, expressive

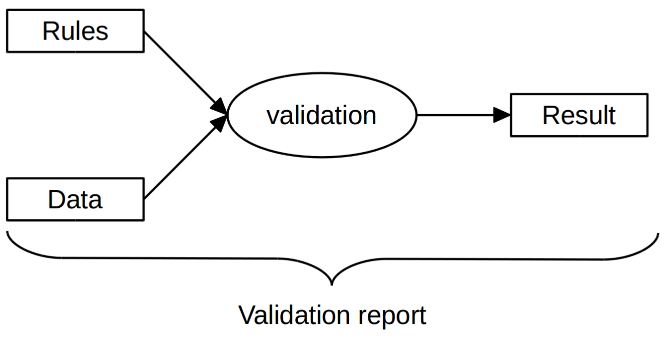


Figure 1: Information elements involved in creating a validation result, relevant for validation reports.

enough to support the most recognized validation report elements recognized from the bottom-up approach and flexible enough to be adopted in a regional validation context while containing the standardized elements from the generic format. Validation tools or other statistical tools producing validation reports as a side product should be able to use this structure for their validation output.

In this deliverable we address the following aspects: In Chapter 2 we ask ourselves what we should expect from a generic validation report. After some elaboration on the variety in richness of a validation report, we define a number of generic demands. In Chapter 3 the idea of validation events and aggregation events will be explained. Chapter 4 explains how to identify validation results and other core elements to be used in the definition of the validation report structure. In Chapter 5 we dive deeper into the modeling of aggregates to be contained in a validation report. In Chapter 6 we formally define the generic validation report structure that can carry both basic and aggregated results. Chapter 7 describes the technical implementation of the concepts developed in the earlier Chapters.

2 Requirements for a validation report

Figure 1 gives a high-level overview of a data validation procedure. At the input side, we find the data to be validated and the validation rules that the data are supposed to satisfy. At some point in time, the data are confronted with the rules and validation results are created.

The purpose of a validation report is to convey validation results and information on the validation procedure. The procedure as a whole generates and processes a lot of information that can possibly be included in such a report. For example, the validation rules may be endowed with metadata such as descriptions and

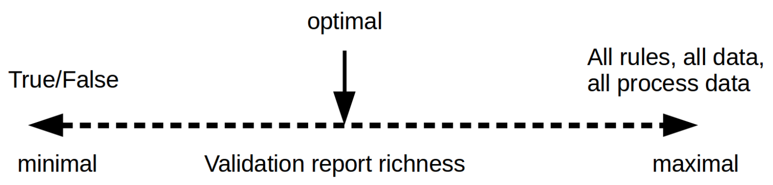

optimal
True/False
All rules, all data, all process data
minimal
Validation report richness
maximal

Figure 2: Possible content of validation reports on a conceptual scale of 'richness'.

severity level and for the validation procedure it may be interesting to record a timestamp and the used software.

A relevant question to ask is therefore what information should be included in a validation report. Conceptually we can explore the extreme possibilities on a scale such as depicted in Figure 2. On the left, we find a minimal report containing a single result only: True, meaning that all data passed all rules, or False, meaning that not all data passed all rules. On the right extreme, the report conveys all data and metadata associated with the validated data, the validation rules, the validation procedure and the results.

Before moving to a formal discussion of the information items involved with data validation, it is useful to discuss a number of demands on a validation report that will serve all uses and users. First of all, we take as a given that a result that cannot be identified with the data and rules it pertains to is useless. If the reports are to be used for communication across organizations, or if they are to interpreted separately from the process that triggered the validation, the relation with the rules and data needs to be included. In fact, this can also be seen as a direct consequence of the validation principle *Well-documented and appropriately communicated validation errors* (ESS, 2017). This leads us to the following demand.

Demand 1 (Identification)**.** *A validation report shall convey validation results such that they can be identified with the validation procedure, the validation rules used, and the validated data.*

A validation procedure usually involves multiple rules and each rule may concern different subsets of variables, records or reference datasets. Every confrontation of a rule with the data can be seen as a validation (sub)procedure yielding a validation report. To gather all results, validation reports should be able to be combined to an overall report.

Demand 2 (Closure under combination)**.** *Two validation reports shall be combinable in such a way that the result is again a validation report that includes all information that separate reports contained.*

This includes cases where multiple procedures are involved, possibly related to varying datasets, rule sets and actors involved in the validation procedures.

Finally, depending on intended use, one may be interested in the details of each step in each validation procedure, or in a more aggregated view of one or more validation events. Indeed, a quick view on some example validation reports from multiple NSI's and multiple statistical domains shows that most of them contain some kind of aggregated results within the validation report itself. Therefore, we also demand the following.

Demand 3 (Closure under aggregation)**.** *A validation report can be aggregated such that the result is again a validation report.*

Here, many types of aggregation may be relevant, including counting the (relative) number of passes and fails, finding the procedure that yielded the maximum number of fails (or passes), and so on.

Both the second and third demand mention the term *closure* which may not be a familiar term to all readers. The term 'closure' or 'algebraic closure' refers to the property of a set being invariant under certain operations. For example, because the sum of any two natural numbers is again a natural number, we can say that the natural numbers are closed under addition of two numbers. For our purposes it is important that the result of combining or aggregating a validation report is also a validation report, meaning that it has exactly the same structure (but possibly different content) before or after aggregation. If this is not the case, we run the risk of defining new data structures for each aggregate or combination of reports.

Upon closer examination the creation of aggregate results closely parallels the creation of validation results. From an abstract point of view both types of results are created by an event where an expression is evaluated after substituting its variables with values coming from a given dataset. Their explicit differences derive from their use in the context of validation reports: validation results are created by confronting statistical data with validation rules while aggregation results are created by evaluating an aggregation function using a set of validation results.

In the following section the idea of expression evaluation will be discussed in a little more detail. Next, the metadata elements will be specified towards validation and aggregation.

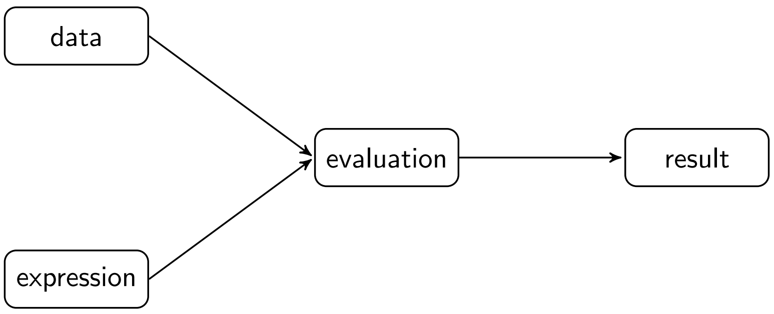


Figure 3: Concepts involved in evaluating a (validating or aggregating) expression.

3 Evaluation events

Figure 3 depicts the concepts involved in creating a result by evaluating a validating or aggregating expression. Conceptually, the data consists of values bound to variable names. The *expression* denotes, using a fixed set of syntax rules, a computation on variables present in the *data*. In the *evaluation* event, the following activities take place

1. Read the expression and check whether it is valid syntax. If not: stop execution.
2. Parse the expression: substitute variable names with the corresponding values stored in *data*.
3. Evaluate the expression, creating the *result*.

In actual implementations these processes can be optimized. For example when the same expression is evaluated with multiple data sets, the first step needs to be executed only once. As a demonstration that both validation and aggregation fit in this description, consider the following two examples.

Example 1. *Consider the expression*

```
age >= 0
```

and a data record given by

Name	*Age*	*Sex*
Joe	*17*	*male*

In the first step, the evaluator reads in the expression and approves it after checking with the syntax. In the second step, the variable names are identified and replaced with matching values from the dataset. This yields `17 >= 0`*.*

Finally, in the third step the proposition `17 >= 0` *is evaluated and the result returned.*

Example 2. *Consider the following table of validation results (we use 1 for* `True` *and 0 for* `False`*).*

Name	*validation_result*
Alice	*1*
Bob	*1*
Carol	*0*

The expression counting the number of passes is given by

`sum(validation_result).`

In the first step the evaluator reads in the expression and approves it after checking with the syntax. In the second step, the variable names are identified and the expression is expanded[1] *to* `1 + 1 + 0`*. This expression is then evaluated and its result returned.*

Obviously, to interpret a result, both the input data and the expression used must be known. However, the value of the final result may also depend on the (version of) the evaluator. Especially since the Handbook on Validation explicitly includes the possibility of validation being done by expert review. In that case, the 'expression' may be a manual or handbook with recommendations on evaluating a certain dataset and the result is a validity assessment by an expert. But even in the case of formalized expressions, interpreters may differ. Formal programming syntax standards often leave certain details to interpreter developers. This inevitably leads to platform-dependent results.

It is therefore proposed here to include metadata elements that identify data, expression, and the evaluating event for all results reported in a validation report, whether they are aggregates or validation results. Since validation results have a particular type and meaning that differs from aggregation results it will be useful to differentiate their metadata as well.

4 Identifying validation results

To get an idea of the (meta)data necessary to identify a validation result, consider the data, validation rules and validation results shown in Table 1. When a dataset is confronted with a validation rule, there are three possible outcomes.

Table 1: Example data, validation rules, and results for six validation events.

Variables			**Rules**	
				`IF Age < 15 THEN`
Nr	Age	hasjob	`Age >= 0`	`hasjob == 'no'`
1	36	`yes`	1	1
2	53	`NA`	1	`NA`
3	11	`yes`	1	0

A rule can be satisfied, yielding 1, a rule can be failed, yielding 0, or a rule cannot be evaluated because of missing data, yielding `NA`.

In the example three records on age and work status are checked against two rules: age must be larger than or equal to zero, and persons under 15 years old cannot have a job. In each case the demand on age can be checked and each record passes this test, yielding 1 (`True`) as validation result. For the second rule, the first record passes the check since age equals 36 and the person has a job which is allowed by the rule. In the second case the job status is not available (`NA`). Hence the rule cannot be checked and the returned value is `NA` as well. Finally, in the third record there is an 11-year old with a job which is a combination that is not allowed by the rule, yielding 0 (`False`).

The above discussion leads to the following definition of a validation result

Definition 1 (Validation result)**.** *A validation result is a value from the set* $\{0, 1, \texttt{NA}\}$.

Validation results are obtained as outcomes of evaluating a validation rule on a data set. We also follow the common convention which identifies 0 with `False` and 1 with `True`. The numeric representation will make it easier to define aggregators in later chapters.

A recurring point of discussion is whether `NA` should be an allowed validation result. The main other options are to either interrupt execution, or to interpret the result as failed when one of the data points necessary for evaluating a rule is missing. Because statistical data often suffers from missing data the first option would yield many interruptions of a statistical process flow. The only way to prevent that would be to make sure that each rule guards against it explicitly by building in clauses that detect missing values. The second option (`NA` implies `False`) would yield a loss of information. More importantly, it assumes (possibly unwarranted) that the user of the result follows this interpretation.

We deem both alternatives undesirable and therefore follow the definition above, which also agrees with the definition of a formal data validation function in

[1]Conceptually of course. In practice accumulation will be more efficient.

the Methodology on Validation handbook published by the Validat Foundation ESSnet (Di Zio et al., 2015). In practice, `NA` is likely to be more important for validating (raw) micro data than for aggregated data. In the latter case missing values may be forbidden but in that case an explicit validation rule checking for missing data can be applied.

Often, but not always, the dataset that is used to evaluate a validation rule is also the data under scrutiny. As an example consider a set (column) of numerical values $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. In a structural business survey for example, $\boldsymbol{x}$ may consist of profit values for different enterprises. The rule

$$\textsf{mean}(x) \geq 0$$

is a check on the whole column. The dataset is considered invalid if the mean profit is negative. This does not mean that all values are necessarily erroneous, it just means that this particular combination of profit values cannot be accepted. So in this case the data used to compute the result is also the data that is being validated. Now consider a commonly used rule, based on a method of Hiridoglou and Berthelot (1986), evaluated for each value x_j $(j = 1, 2, \ldots, n)$:

$$\max\left(\frac{x_j}{x^*}, \frac{x^*}{x_j}\right) \leq h. \tag{1}$$

Here, x^* is a reference value, usually $\textsf{median}(x)$, and h is a fixed parameter. To evaluate this rule x^* must be computed which implies that the whole of $\boldsymbol{x}$ must be known. This means that in a formal sense the rule is a validation of $\boldsymbol{x}$ as a whole. After all, changing any value x_i may influence the result for even when $i \neq j$. The aim is of course to evaluate the rule n times, once for each value x_j where the goal is to evaluate the value of x_j, not the combination of values $\boldsymbol{x}$ n times.

For this reason it is proposed that the validation report provides explicit room to communicate cases where by intention, the validated data is not identical to the data used in the evaluation of a rule.

Definition 2 (validation)**.** *A* validation *is a tuple (e, d, f, v), where e identifies the physical validation event, d identifies the data points to which the result pertains, f identifies the evaluated validation rule, and v is the generated validation result.*

We leave open the possibility that the components e, f and d consist of informative tuples to identify subcomponents of the event, data, or rule. In particular, d will contain information on the data used to evaluate the rule as well as the data under scrutiny. Note that the above 'definition' is not a definition in the mathematical sense. Rather it is a convention that can be tested in a (software) environment where data, rules, and validation events have been labeled and stored.

5 Aggregation

A stocktaking of aggregates desired by users of validation reports yielded a wide range of wishes[2]. Commonly reported aggregates include the total number or fraction of rules violated or the number or fraction of records that violate a certain rule. Aggregates need not have a numerical value. For example, in the case of record-wise validation rules, interesting aggregates include 'the validation rule that is violated most often by a dataset' or the 'record violating most (record-wise) validation rules'.

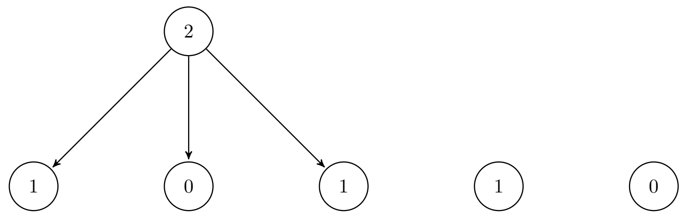
Before discussing how we endow machine-readable validation reports with identifiable and combinable aggregates, it is useful to point out some of the subtleties that arise when thinking of aggregation in a general way. The first subtlety has to do with counting rules and counting validation results. Consider as an example the rules `turnover >= 0` and `mean(profit) >= 0`. When applied to a set of n business records containing the variables `turnover` and `profit`, the first rule is evaluated n times in n validation events, yielding n validation results. The second rule is evaluated in a single validation event, yielding a single validation result, regardless of the number of records in the dataset. This means that an aggregate such as 'the total number of rules violated' is ill-defined. It is only meaningful to talk about 'the number of events that resulted in `False`'.

The second subtlety is that a set of aggregates can again be combined to form new aggregates. For example, in a first step we may compute the fraction of events yielding `False` per economic sector. In a next step we count for how many sectors this number is less than or equal to 0.05.

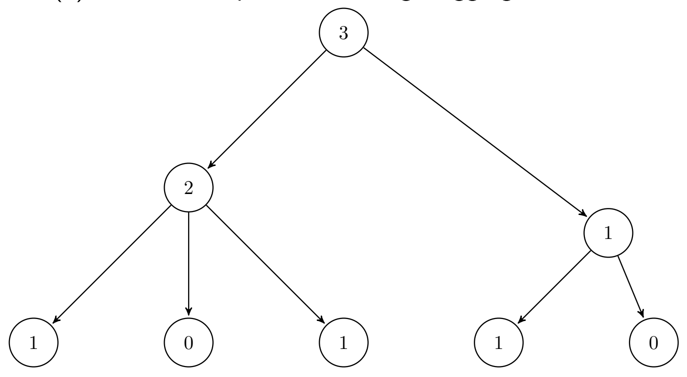
Figure 4 illustrates the above points in a graph-like structure. Suppose we start with a basic report containing five validation results. We can add an aggregate counting the number of passes in a certain subset (Figure 4a). In Figure 4b, a second count is added, as well as an overall count that is composed of the two low-level aggregates. Figure 4c depicts a situation where a subset of validation results is combined to form aggregates of several types: the number of passes and the fraction of passes.

These examples suggest that aggregates can be interpreted as nodes in a directed graph where the edges (arrows) point from the aggregate to the nodes used to compute the contents of the node. Nodes that have no outgoing edges (leaves) correspond one-to-one with validation results. The collection of nodes and edges contains sufficient structure to make the representation of validation reports identifiable, combinable, and (recursively) aggregable. In the following

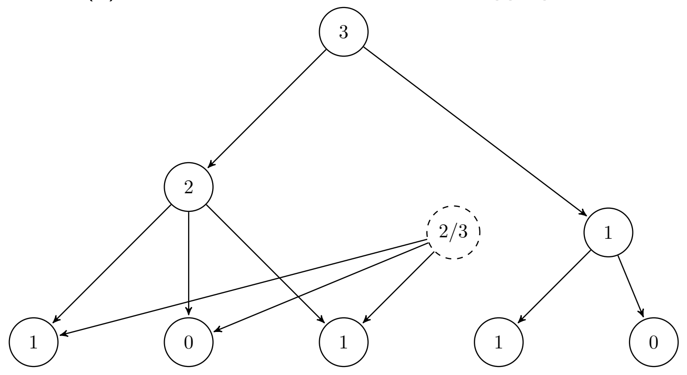
[2]Inventories were taken at the ESSnet Validat Integration meeting in Örebro Sweden (14-16 February 2017) and during the Task Force on Validation meeting in Luxembourg (27 April 2017). Minutes of both meetings are available via CROS-portal .



(a) Validation report with a single aggregated value.



(b) Validation report with multiple aggregates.



(c) Validation report with aggregates of various types.

Figure 4: Structure of validation reports including aggregates, of varying complexity.

paragraph we define more closely the type of graph that can be used to represent an aggregation structure.

5.1 Aggregation graphs

Graphs are well-known mathematical structures that represent connectivity between objects. Indeed, the study of graphs dates back to the 18th century when Leonhard Euler (1741) solved the famous Köningsberger bridges problem. For our purposes, a graph will serve as a model to store validation results, aggregates thereof, and the operations that lead to the aggregated values.

The need for a graph structure rather then a more simple record-like structure arises from the demand that reports be combinable to a new report. To see this, consider the following example. Two institutes validate a dataset with turnover values against the rule `turnover >= 0`. The first institute reports the fraction of passes, while the second institute reports the results for each individual record. If these reports were naively combined, one could misinterpret the fractional value reported by the first institute as relating to the separate values reported by the second institute. A second example occurs when the first institute creates a report containing both the individual results and the aggregated fraction of passes. If this report is to be augmented with new information, for example when new records come in it should be clear from the report that the reported fraction of passes does not concern the records added later.

Graphs basically consist of elements of some set, called *nodes* or *vertices* and connections between them, which are called *edges* or *arrows* if they are directed. Structured information can be stored by endowing the nodes and edges with parcels of data. Depending on the type and number of edges allowed between the nodes, graphs are classified in a broad number of types. Below we define simple directed graph that comes close to our purpose of structuring validation aggregates.

Definition 3. *A* simple finite directed graph *is a pair (V, E) where V is a finite set and E is a set of pairs (v_1, v_2) with $v_1, v_2 \in V$ and $v_1 \neq v_2$.*

The term *directed* means that edges have a direction: the edge (v_1, v_2) can be thought of as an arrow, pointing from v_1 to v_2 and therefore differs from (v_2,v_1). The first element of an edge is referred to as the *start* and the second element is referred to as the *end* of the edge. In a simple directed graph there are no loops (edges whose start and end are the same) and there are maximally two opposite-pointing edges that connect two nodes.

A validation report can be represented by a more special graph. To define it, we first need the concept of a paths and cycles.

Definition 4 (path, cycle)**.** *A* path *is a sequence of edges edges $e_1, e_2, \ldots, e_n$ such that the end of e_j is equal to the start of e_{j+1} for $j = 1, 2, \ldots, n-1$. A* cycle *is a path such that the end of e_n is equal to the start of e_1.*

A path is thus a sequence of connected edges. Since aggregation works bottom-up and not top-down, we define the following graph to represent an extended validation report.

Definition 5. *A* directed acyclic graph *is a simple finite directed graph (V, E) with the condition that E contains no cycles.*

The term ‘directed acyclic graph’ is often shortened to DAG in literature. This particular structure has many technical applications, For example, the DAG stands model for data processing flow in the SPARK model for distributed computing (Gupta et al., 2003).

Simple graphs have a natural rule of composition. The ‘sum’ of two simple (directed) graphs is obtained as the union of the two vertex sets and the union of the two edge sets. For a DAG, this combination rule does not always apply since such a combination could in principle introduce cycles. (As an example, combine the DAGs $(a, b, (a, b))$ and $(a, b, (b, a))$, the result is no longer a DAG). We therefore define the following compatibility rule for directed acyclic graphs.

Definition 6 (Compatible DAGs)**.** *Given two DAGs* $G = (V, E)$ *and* $F = (V', E')$*. These graphs are called* compatible *when the combination*

$$(V \cup V', E \cup E'),$$

is also a DAG.

A practical consequence of the condition in this definition is that software that combines extended validation reports (validation reports including aggregations, to be defined below) should always check for compatibility of the reports upon combination.

One case of compatible graphs occurs when both G and F are edgeless. Two reports that do not contain any aggregates satisfy this condition. Ifa F and G have no nodes or edges in common, they are also compatible.

5.2 Identifying aggregates

Just like validation results, aggregation results are created by evaluating an expression. Also, just like for validation results, the interpretation of to what dataset an aggregate relates need not coincide with the data used to compute the result. Consider a set of validation results $\boldsymbol{v} = (v_1, v_2, \ldots, v_k, v_{k+1}, v_{k+2}, \ldots, v_n)$. Here, $v_1 \ldots v_k$ relates to one subset of the data under scrutiny, for example a certain economic sector, and $v_{k+1}, \ldots, v_n$ relates to another subset. The aggregate

$$a = k - \sum_{j=1}^{k} v_j,$$

counts the number of violations in the first subset. In this case, the aggregate communicates a fact about the set used in the calculation. Now consider

$$a' = \frac{a}{n - \sum_{j=1}^{n} v_j}.$$

This aggregate is the fraction of all failures that occur in the first subset. Although every validation result is used to compute it, such aggregates are usually reported in a table with one column stating the subset (e.g. the economic sector) and one column stating the aggregate value.

For this reason it is proposed that the validation report provides explicit room to communicate cases where by intention, the value set to which an aggregate pertains are not identical to the values used in evaluation of a rule.

Definition 7 (aggregation)**.** *An* aggregation *is a tuple* (e, d, f, a) *where* e *identifies the aggregating event,* d *the data related to the aggregation,* f *the aggregating expression, and* v *the aggregate value.*

We leave open the possibility that the elements e, d and f are again tuples. In particular, d may identify both the data used while evaluating a rule and the data that is the subject of validation. Observe also that the data element (d) fixes the edges of the aggregation graph. Like Definition 2, this is not a definition in a precise mathematical sense. Rather it is something that can be tested in a particular practical software/data environment.

6 Validation reports

Recall that a *validation* is a tuple (e, d, f, v) and an *aggregation* is a tuple (e, d, f, a). Here ($v \in \{0, 1, \texttt{NA}\}$) is a validatin result and a is an aggregate value. In both cases, e refers to the event that where the expression f was evaluated to create the result (v, or a) and d refers to the data evolved in evaluating the expression as well as the data related to the interpretation of the result.

The tuples are constructed to make the values a and v identifiable (Demand 1). The following recursive construction defines validation reports that are combinable (Demand 2) and aggregable (Demand 3).

Definition 8 (Validation report)**.**

1. *The empty set* $\{\}$ *is a validation report.*
2. *If* (e, r, d, v) *is a validation then* $\{(e, d, r, v)\}$ *is a validation report.*

Note that $\{(e, d, r, v)\}$ *is a trivial DAG, with a single node and no edges.*

3. *If* V *and* W *are combinable in the sense of Definition 6, then* $V \cup W$ *is also a validation report.*

4. *If V is a validation report, f is an expression, and S is a subset of V such that f can be evaluated with S. Then*

$$V \cup \{(e_{fS}, d_{fS}, f, f(S))\},$$

is also a validation report. Here, e_{fS} identifies the event that created the aggregate $f(S)$ and d_{fS} identifies S and the data to which the result $f(S)$ pertains.

The first step is a formality, allowing for the edge case of empty reports. By applying the second and third step repeatedly, a report can be populated with identifiable validation results (validations). Observe that as long as we are only adding validations, the validation reports are trivially combinable since it can be interpreted as an edgeless graph (see under definition 6). In step four, an identifiable aggregate (aggregation) is constructed that is compatible with the validation report to which it is added. Remember that an aggregation object also stores the edges to the nodes that were used to construct it so this definition indeed constructs a directed acyclic graph.

Now that we have a conceptual definition of a validation report, that satisfies all three demands, we can move forward and define the identifying pieces of information to be stored in the validations and aggregations on a logical level.

6.1 Logical validation report structure

In the following subsections the information that needs to be stored in validation or aggregation tuples is described explicitly. The descriptions are formatted in a set of tables, each with the following structure.

1. Item: the name of the information item.
2. Format: logical format of the data in the item. Allowed formats are: `string`, `numeric`, `enum` (with categories defined in the description column), `datetime` and `-`. The latter indicates that the format is free, including the possibility to include user-defined objects. A type may be followed by brackets `[]` to indicate an array.
3. Description: a short description of the item. More detailed descriptions might follow after the table.
4. Example: an example.

We distinguish between information which is mandatory and information that is recommended. This is indicated in the caption of each table. Some information items may be extended with user-defined information. Whether this is the case is indicated at the top of each table.

6.1.1 Identification of an expression evaluation event.

Both validation results and aggregates are created by an event e that evaluates an expression.

Table 2: Mandatory identification of a expression evaluation event e . Fields marked with a † are for validation results only. The number of fields is extendable.

Item	Format	Description	Example
time	`datetime`	Time marking the completion of a validation event.	20170212 10:15:30+0100
actor	`string`	Software that or person who created the result.	`R package validate version 0.1.7`

The 'Business Architecture for ESS Validation' (ESS, 2017) defines a service-oriented infrastructure for data validation. In the cases where a client-server model is applied (the server executing validation and sending reports), the following extra information is recommended.

Table 3: Recommended information on a physical validation event e. Fields marked with a † are for validation results only. The number of fields is extendable.

Item	Format	Description	Example
agent	-	Actor (person, institute, dpt, ...) responsible for executing the validation event	dpt. of data validation, Eurostat
trigger	-	Actor (person, institute, dpt, ...) responsible for triggering the event	John Statistician, Statistics Netherlands

6.1.2 Identification of an expression

Table 4: Mandatory identification of an expression f. Fields marked with a † are for validation results only. The number of fields is extendable.

Item	Format	Description	Example
language	string	Language and version in which a expression is written	R/validate version 0.1.7
expression	string	Expression defining the rule or aggregate.	`age >= 0`
severity†	enum	`‘error’`, `‘warning’`, or `‘information’`	`‘error’`

The business architecture for ESS validation also allows for certain up- or downgrades of the severity status for individual cases. Furthermore, it is good practice to explain the purpose of complicated expressions in a human-readable description.

Table 5: Recommended values for identification of a validation rule. Fields marked with a † are for validation results only. The number of fields is extendable.

Item	Format	Description	Example
description	`string`	human-readable description of the rule.	Nonnegativity for age.
change†	`enum`	`‘up’`, `‘down’`	`‘down’`
explanation†	`string`	Explanation for change	Nationalization of a large bank.

Of course the ‘status’ field can also be used to add an explanation on why a status was changed.

6.1.3 Identification of data

The validation report identifies two data sets for each reported validation result or aggregate: the set of datapoints that was involved in evaluating the expression and the set of datapoints related to the interpretation of the result. The dataset that is used to evaluate an expression will be referred to as the *source* data while the dataset that is related to the interpretation of the result will be referred to as the *target* data. In many cases these will coincide but see Equation 1 on Page 10 for a counterexample. There, the *source* is the whole column of data (denoted $\boldsymbol{x}$) while the *target* is a single value of the column (denoted x_j).

Table 6: Mandatory identification of validated data.. Fields marked with a † are for validation results only. The number of fields is extendable.

Item	Format	Description	Example
source	string[]	A key or set of keys identifying the data used in evaluating the expression.	{('Dutch inhabitants', 'EU-SILC2016, 'Richard Respondent', 'Income')}
target	string[]	A key or set of keys identifying the data targeted by the expression.	{('Dutch inhabitants', 'EU-SILC2016, 'Richard Respondent', 'Income')}
Convention: if the 'target' field is empty, it is assumed equal to 'source'.			

Since a set of keys that identify a dataset is hard to interpret by humans, we add the following recommendation.

Table 7: Recommended values for identification of validated data.

Item	Format	Description	Example
description	`string`	human-readable description of the data.	Income of a single citizen.

Below, we sketch two possible ways on how the data identification could be implemented.

1. Full specification of data. The methodology handbook on validation prescribes a generic model to identify a single datapoint (Di Zio et al., 2015, Chapter 5). In short, one identifies the value of a data point by fixing

- the population U;
- the event τ that lead to its observation;
- the population unit u from which a property was observed, and
- the attribute X that was measured.

Here, the term 'population' should be interpreted rather generally. It may be the human population of a country or region, but it can also be a population of companies, countries, events, emails, and so on. Similarly, the event that lead to an observation can be the receiving of transmitted data from an institute, or it may be a data collection event based on a survey. In the handbook, a data point is defined as a value (from some domain) paired with a tuple (U, τ, u, X) that identifies it.

When the set of keys consists of a set of (U, τ, u, X)-tuples as defined in the methodological handbook on validation, the report will identify data involved in validation completely free of any context involving the sender, the process, institutes involved and so on.

2. Extra standardization. The key sets can quickly inflate the size of a validation report. Since the format is left open (we only specify keys to be an (array of) strings), it is possible to apply a more practical format at the price of extra standardizing agreements between sender and receiver of the report. Let us illustrate this by sketching a validation procedure in a service-client based infrastructure. We call the client 'Alice' and the server 'Bob'.

1. Alice sends data, consisting of n records and a validation rule to 'Bob'. She also sends a unique string s (for example a hash key) that she has connected with this particular dataset in her administration. The rule she sends is such that it must be evaluated on every record.
2. Upon receiving Alice's message, Bob evaluates the rule on each of the n records. He sets each *source* field equal to the string $s.k$, where k is the key that uniquely identifies the record under scrutiny. The *target* fields are left empty.
3. Bob completes the validation report and sends it to Alice.

The trade-off in the above procedure is that the validation report can only be understood by the sender and receiver, but not by a third party who is unaware of the meaning of the identifying keys s and k.

6.1.4 Result values

Table 8: Mandatory format for the validation result v or aggregation result a. Fields marked with a † are for validation results only. The number of fields is not extendable.

Item	**Format**	**Description**	**Example**
value†	`enum`	1, 0, or `NA`	1
value	`string`	evaluation result	`"7"`

7 Technical standards

In this section we define explicitly the technical data exchange format for the logical structures defined in the previous subsections.

7.1 Object model

The validation report structure consists of records that are either of type validation, or aggregation. The former are schematically represented as follows

$$\begin{aligned}\text{validation} &:= \langle \text{id}, \text{type} = \texttt{"validation"}, \text{event}, \text{rule}, \text{data}, \text{value}\rangle\\ \text{event} &:= \langle \text{time}, \text{actor}, agent, trigger\rangle\\ \text{rule} &:= \langle \text{language}, \text{expression}, \text{severity}, status, decription\rangle\\ \text{data} &:= \langle \text{source}, \text{target}, description\rangle,\end{aligned}$$

where 'id' is a unique identifier, 'type' has a fixed value that labels the object type and 'value' is a validation result (0, 1 or `NA`). In the above scheme, the fields in $italics$ are optional.

Objects of type type 'aggregation' are schematically described as

$$\begin{aligned}\text{aggregation} &:= \langle \text{id}, \text{type} = \texttt{"aggregation"}, \text{event}, \text{aggregate}, \text{data}, \text{value}\rangle\\ \text{event} &:= \langle \text{time}, \text{actor}\rangle\\ \text{aggregate} &:= \langle \text{language}, \text{expression}, description\rangle\\ \text{data} &:= \langle \text{source}, \text{target}, description\rangle,\end{aligned}$$

where 'value' is the aggregate value, represented as a string.

The above description serves as a quick reference data structure, independent of the implementation language. It encompasses Definitions 1, 2 and 7, with components derived from the logical descriptions in §6.1.

7.2 Implementation

Below we propose a technical format for exchanging data validation reports. There are several common standards allowing for implementation of structured data exchange, including textual formats such as XML (W3C consortium, 2013), YAML (Ben-Kiki et al., 2009), JSON (ECMA, 2013) and popular binary formats such as protobuf (Google, 2017). In principle, any of these formats can be used to serialize the validation and aggregation data structures that were in the previous subsection. Here, we use the JSON format because of simplicity,

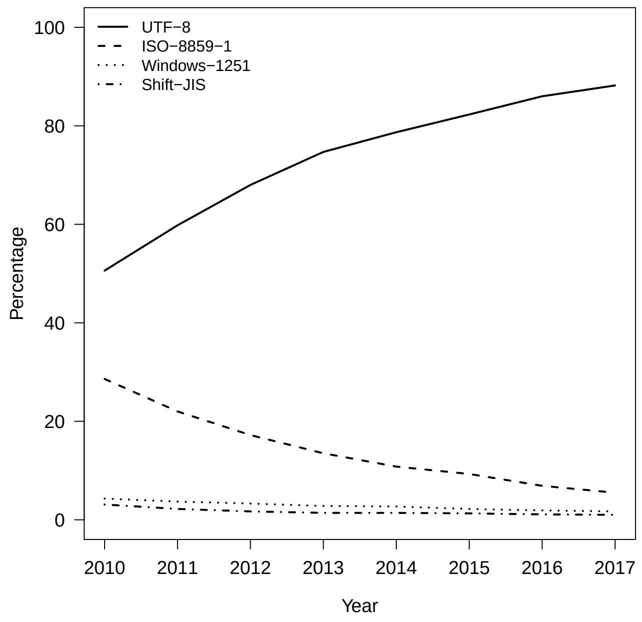


Figure 5: Percentages of encoding standards used on the web (W3techs, 2017).

wide support in many languages, and the availability of a schema definition language (Galiegue et al., 2013). Moreover, JSON strings parse straight into javascript objects, facilitating further processing and visualisation, for example in the popular `d3.js` framework (Bostock et al., 2011). This implementation can be used as a reference and it is left to the user to implement the concepts in another language if so desired.

Although it is a textual format, the JSON standard does not impose restrictions on the encoding used. It is left explicitly to standards built upon JSON to define an encoding (ECMA, 2013, pp ii). In this standard we follow the currently most widely applied standard (see Figure 5) with the following demand.

Table 9: File encoding used for validation reports

Validation reports are encoded in `UTF-8`.

The different data types within a file are to be formatted according to commonly used standards where possible. In particular, data in validation reports are encoded as stated in Table 10.

Table 10: Format of data types in validation reports.

1	Numbers are encoded in a valid decimal ISO/IEC/IEEE 60559:2011 (IEEE 754) format (IEEE, 2008).
2	Date-time data shall be denoted in basic ISO 8601 format `YYYYMMDDThhmmss`$\pm$`hhmm` (ISO, 2004).

The JSON scheme defining is shown in Listing 1. The full code can also be found at github:

https://github.com/data-cleaning/ValidatReport

Listing 1: JSON schema for a validation report.

```
{
  "title"   : "validation report 1.0.0",
  "id"      : "https://goo.gl/PhHurj",
  "$schema" : "http://json-schema.org/draft-04/schema#",
    "type" : "array",
    "items" : {
      "oneOf" : [
        {"$ref" : "#/definitions/validation"},
        {"$ref" : "#/definitions/aggregation"}
      ]
    },
  "definitions" : {
    "validation" : {
      "type" : "object",
      "properties" : {
        "id"          : { "type" : "string" },
        "type"        : { "enum" : ["validation"] },
        "event"       : { "$ref" : "#/definitions/event" },
        "rule"        : { "$ref" : "#/definitions/rule"  },
        "data"        : { "$ref" : "#/definitions/data"  },
        "value"       : { "enum" : ["0", "1", "NA"] }
      }
    },
    "aggregation" : {
      "type" : "object",
      "properties" : {
        "id"          : { "type" : "string" },
        "type"        : { "enum" : ["aggregation"] },
        "event"       : { "$ref" : "#/definitions/event" },
        "aggregate" : { "$ref" : "#/definitions/aggregate" },
        "data"        : { "$ref" : "#/definitions/data"  },
        "value"       : { "type" : "string" }
      }
    },
    "event" : {
      "type" : "object",
      "properties" : {
        "time"    : { "type" : "string" },
        "actor"   : { "type" : "string" },
        "agent"   : {},
        "trigger" : {}
      },
      "required"  : ["time", "actor"]
    },
    "rule" : {
      "type" : "object",
      "properties" : {
        "language"   : { "type" : "string" },
        "expression" : { "type" : "string" },
        "severity"   : { "enum" :
          ["information", "warning", "error"] },
        "description": { "type" : "string" },
        "change"     : { "enum" : ["up", "down"] },
        "explanation": { "type" : "string" }
      },
      "required" : ["language", "expression", "severity"]
    },
    "aggregate" : {
      "type" : "object",
      "properties" : {
        "language"   : { "type" : "string" },
        "expression" : { "type" : "string" },
        "description": { "type" : "string" }
      },
      "required" : ["language", "expression"]
    },
    "data" : {
      "type" : "object",
      "properties" : {
        "source" : {
          "type" : "array",
          "items" : {"type" : "string"}
        },
        "target" : {
          "type" : "array",
          "items" : {"type" : "string"}
        },
        "description" : { "type" : "string"}
      },
      "required" : ["source", "target"]
    }
  }
}
```

8 Examples

In this Section we work out a few examples based on validation reports that are currently implemented in several statistical institutes. We are grateful to the organisations that provided the examples to the ESSnet project.

Listing 2: Part of an extended report based on an example from the Swedish Triton system.

```
[
  {
    "id": "agg1",
    "type": "aggregation",
    "event": {
      "time": "20150930T104052+0200",
      "actor": "Triton demo 2",
      "agent": null,
      "trigger": null
    },
    "aggregate": {
      "language": "VBA",
      "expression": "COUNT",
      "severity": "information",
      "description": "Totalt antal fel",
      "status": ""
    },
    "data": {
      "source": null,
      "target": null,
      "description": ""
    },
    "value": "17"
  },
  {
    "id": "A_22",
    "type": "validation",
    "event": {
      "time": "20150930T104041+0200",
      "actor": "Triton demo 2",
      "agent": null,
      "trigger": null
    },
    "rule": {
      "language": "VBA",
      "expression": "Om(B1; PersonNr10; 0;
            T.Substring(KO;9;0;N, KO;1;0;N)) >= (KO;0;T)",
      "severity": "error",
      "description":
          "A_22 Personnummer maste anges med 10 tekken.",
      "status": ""
    },
    "data": {
      "source": "["invarna i Sverige",
                "underskning",
                "5510112641",
                "PersonNr10"]",
      "target": null,
      "description": ""
    },
    "value": "0"
  }
  ... other evaluation results ...
]
```

Listing 3: Part of an extended report based on the assessment of VTL 1.1 - Questionnaire for NAPS statistics (January 2017).

```
[
  {
    "id": "aggr",
    "type": "aggregation",
    "event": {
      "time": "",
      "actor": "",
      "agent": null,
      "trigger": null
    },
    "expression": {
      "language": "VTL 1.1",
      "aggregate": "COUNT",
      "description": "Total nr of revisions above threshold",
    },
    "data": {
      "source": ["WC001", "WC002", "WC003", "WC004"],
      "target": null,
      "description": "EU member states"
    },
    "value": "4"
  },
  {
    "id": "WC001",
    "type": "validation",
    "event": {
      "time": "20150930T104041+0200",
      "actor": "Laura Vignola",
      "agent": null,
      "trigger": null
    },
    "rule": {
      "language": "VTL 1.1",
      "expression": "
        ds_esa_current_filt =
          ds_esa_current [
            filter (PRICE = 'V' and PRICE = 'Y') ];

        ds_esa_previous_filt =
          ds_esa_previous [
            filter (PRICE = 'V' and PRICE = 'Y') ];

        ds_esa_current_pcent =
          ds_esa_current_filt [ STO = 'P3' ] /
            ds_esa_current_filt [ STO = 'B1GQ' ];

        ds_esa_previous_pcent =
          ds_esa_previous_filt [ STO = 'P3' ] /
            ds_esa_previous_filt [ STO = 'B1GQ' ];

        ds_result_5 := check( abs(
          ds_esa_current_pcent.obs_value -
            ds_esa_previous_pcent.obs_value ) ) <= 0.015,
          errorcode('Deviation larger than 1.5% between
            revised figures and previous version for
            the ratio final consumption expenditure / GDP
            at market price'),
          errorlevel('Error') ) ; ",
      "severity": "error",
      "description": "
          For current prices (PRICE='V') and previous year
          prices (PRICE='Y'), the difference between
          - revised figures of final consumption expenditure
            (STO='P3') as a % GDP at market price
            (STO:'B1GQ')
          - and the same ratio for the previous
            transmission should not be higher than 1.5%
          => deviation of 1.5% maximum accepted for the ratio
            (final consumption expenditure /
             GDP at market price) in the revised figures.",
      "status": ""
    },
    "data": {
      "source": [
        [ "prices", "2014Q1-first"   ,"AT",
          "final consumption expenditure"
        ],
        [ "prices","2014Q1-revised","AT",
          "final consumption expenditure"
        ]
      ],
      "target": null,
      "description": ""
    },
    "value": "0"
```

```
  }
  ... the other evaluation results...
]
```

Acknowledgements

The authors are grateful to Tjalling Gelsema, Edwin de Jonge, Dick Windmeijer, and the partners of the ESSnet Validat Integration for comments and suggestions.

References

Ben-Kiki, O., C. Evans, and I. döt Net (2001–2009). *YAML Specification Index*. website.

Bostock, M., V. Ogievetsky, and J. Heer (2011). D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*. website.

Di Zio, M., N. Fursova, T. Gelsema, S. Gießing, U. Guarnera, J. Ptrauskienė, L. Quensel-von Kalben, M. Scanu, K. ten Bosch, M. van der Loo, and K. Walsdorfe (2015). Methodology for data validation. Technical Report Deliverable No. 11/2, ESSNet on validation. pdf.

ECMA (2013). The JSON data interchange format. Technical Report ECMA-404, ECMA International. pdf.

ESS (2017). Business architecture for ESS validation. Technical report, European Statistical System. pdf.

Euler, L. (1741). Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae 8*, 128–140. pdf.

Galiegue, F., K. Zyp, et al. (2013). Json schema: Core definitions and terminology. *Internet Engineering Task Force (IETF)*, 32. json-schema.org.

Google (2008–2017). *Protocol buffers*. website.

Gupta, S., N. Dutt, R. Gupta, and A. Nicolau (2003). Spark: A high-level synthesis framework for applying parallelizing compiler transformations. In *VLSI Design, 2003. Proceedings. 16th International Conference on*, pp. 461–466. IEEE. pdf.

Hiridoglou, M. and J.-M. Berthelot (1986). Statistical editing and imputation for periodic business surveys. *Survey methodology 12*(1), 73–83.

IEEE (2008). *Standard for floating point arithmethic*. IEEE Std. 754-2008, pdf.

ISO (2004). Data elements and interchange formats–information interchange–representation of dates and times. *ISO/TC154*. pdf.

W3C consortium (2008–2013). *XML*. website.

W3techs (2017). Historical yearly trends in the usage of character encodings for websites. website, last referenced May 12, 2017.