Quick guide on VTL 2.0

(version January 2019)

Introduction

VTL is the acronyms of Validation and Transformation Language, defined by the VTL Task Force under the umbrella of the SDMX[1] Technical Working Group. The language contains the operators to be used to define the rules to validate statistical data.

This document has the objective to give to the readers an overview of VTL version 2.0 (released on April 2018) and some operators that can be useful in the validation process. For every operator, a description will be provided and an example or a rule. The rules are also explained using real set of data, in appendix. For all the examples and rules has been used a fictive statistical domain "Intra-EU travellers".

To study VTL deeper and to have a complete view of all the operators that are part of VTL 2.0, access to the link: https://sdmx.org/?page_id=5096.

VTL Basic information

The data validation is a phase of the statistical business process in which data are checked through rules in order to ensure the data quality (i.e. the GDP must be greater than zero).

Rules are usually defined in a textual format, sometimes in local language, and subsequently translated in a software language to be executed. VTL is an intermediary formal language between the textual format and the programming language with the scope to facilitate the use but also the sharing and the exchange of rules also with a view to exchanging machine to machine.

[1] SDMX is the acronyms of Statistical Data and Metadata eXchange a standard for the exchange of statistical data and metadata. For further information see the link: http://sdmx.org

Data structure and Dataset

Crucial in VTL is the structure of data. Indeed also, if usually a rule in a textual form does not require necessarily the definition of a structure: for example the rule: "if age less than 14 implies a civil status unmarried" has no structure specification, it is clear that both age and civil status are variables describing data to be checked. VTL requires something more, a specification of a Data Structure that is the representation of data in terms of Components (variables), domains associated to the Components and the roles of Components inside the set of data (this will be specifier better in following paragraph). In table 1, is reported the Data Structure of Intra-EU travellers survey (hereafter called INEUTRAV) that will be used for the examples in the guide.

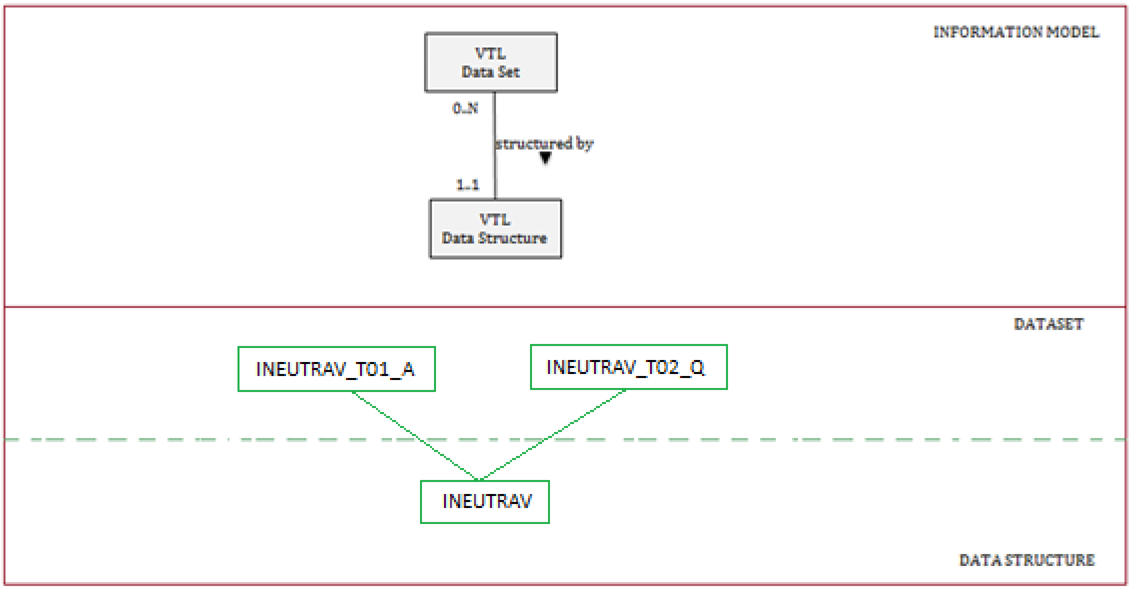
Components	Description	Validity domain (code list, datatype and/or range)	Type
TABLE	Table-ID	CL_TABLE: T01,T02	Identifier
FREQ	Frequency of data	CL_FREQ: A (Annual), Q (Quarterly)	Identifier
TIME_PERIOD	Period (year or quarter) of data	2008, 2009, 2008-Q1, 2008-Q2, …	Identifier
REPORTING	Reporting country	CL_REPORTING: BE, BG, CZ, DK, DE, EE, IE, EL, ES, FR, HR, IT, CY, LV, LT, LU, HU, MT, NL, AT, PL, PT, RO, SI, SK, FI, SE, UK	Identifier
PARTNER	Partner country (country of departure or of destination)	CL_PARTNER: EU28,BE, BG, CZ, DK, DE, EE, IE, EL, ES, FR,HR, IT, CY, LV, LT, LU, HU, MT, NL, AT, PL, PT, RO, SI, SK, FI, SE, UK	Identifier
DIRECTION	Direction of the trip (incoming or outgoing)	CL_DIRECTION: IN (INcoming from PARTNER), OUT (OUTgoing to PARTNER)	Identifier
AGE	The age group to which the travellers belong	CL_AGE : TOTAL, Y0_18 (from 0 to 18), Y0, Y1, Y2…, Y18 Y19_64, Y19, Y20, …, Y64 Y65_MAX, Y65, …., Y122 UNK,	Identifier
ADJUST	Seasonal adjustment	CL_ADJUST: N: Not adjusted S: Seasonally adjusted	Identifier
OBS_VALUE	Number of travellers	>=0	Measure
OBS_STATUS	Status of the observation	empty, P (Provisional)	No viral Attribute

Table 1- INEUTRAV Data Structure

- The “Components” column of the Table 1 contains all the components that are used to describe the data for the Intra-EU travellers survey.
- The “Description” column of the Table 1 contains a description of every component.
- The “Validity domain” column of the Table 1 contains all the codes (i.e. A, Q for FREQ) or the range and data type (integer non-negative number for OBS_VALUE) that every component can assume. The code list is defined only for coded [2]concepts.
- The “type” column of the Table 1, distinguish components in:
    - *Identifiers* – component identifying data,
    - *Measures* – values of the observed phenomena
    - *Attributes* – components providing added information about observed data.

We will see better, later on, the importance of this last distinction.

A logical subset of the Data Structure is the Data Set that is the structure on which the VTL rules are defined. The Data Set is not a physical data file or a table in a database but a logical organization of data deriving by the Data Structure. In some case it can coincide with the Data Structure itself, in other cases can be a subset of the Data Structure in which one or more components are constrained to specific values depending on the dissemination policy. For example in our case, the INEUTRAV Data Structure represents the structure for two Data Sets: INEUTRAV_T01_A, INEUTRAV_T02_Q that are distinguished by the FREQUENCY variable values (in the first Data Set Annual, in the second one Quarterly).



As well as the Data Structure, from which it derives, the Data Set uses the distinction between Identifiers, Measures and Attributes. In general, VTL uses the Identifiers to individuate data and is applied to Measures and Attributes. A single element of a Data Set is called Data Point (for further explanation see the VTL Information Model).

[2] A coded component is a component assuming values in a code list.

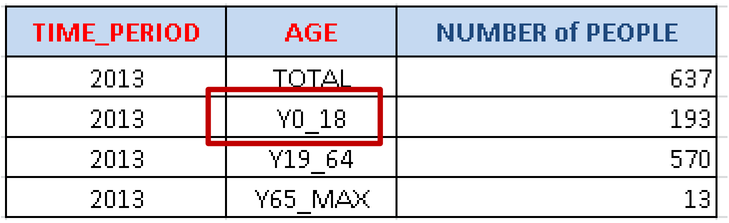
Identifiers

The Identifiers are important to identify data and to manage the join between different Data Set. Indeed, for operators applying:

- on a single dataset (Unary operators): the Identifiers are only used to <u>identify the records</u>,

  For example if we need to check if people having AGE=Y0_18 in 2013 are more than 100. In this case the Identifiers AGE is used to recognize Data Points (people having age between 0 and 18) that should be (OBS_VALUE) more than 100.

  It is evident that, in order to identify Measures, the combination of the values of the Identifiers must be <u>unique</u>.



TIME_PERIOD	AGE	NUMBER of PEOPLE
2013	TOTAL	637
2013	Y0_18	193
2013	Y19_64	570
2013	Y65_MAX	13

- on more than one Data Set (Binary or N-ary operators): the Identifiers are used first of all <u>to join</u> Data Sets (in case more than one are involved) and then, to identify Data Points in the joined Data Set. The join can be:
  - <u>automatic</u>, based on the match of the values of common Identifiers (same name and value domain) to all Data Set. The constraint in this case is that among all the input Data Sets, one must have as Identifiers, all the Identifiers without repetition (superset) of the other Data Sets. The joined Data Set resulting will have as Identifiers those of the "superset" Data Set.
  - or can be done using the <u>*join*</u> operator_ that foreseen different type of join (inner, left, full and cross) and a clause "using" with which is possible to specify the Identifiers to be joined.

Measures

When an operator is applied on Data Set(s), it is applied to all the Measures having the correct datatype for the operator[3]. If the operator applies to more Data Sets, it is applied to all the Measures having the same name and datatype for all the input Data Sets. The Measures are returned with the same input name except for some operators changing the Value Domain of the Measures or in case of renaming.

For example: ds1 + ds2 sums the Measures of the two Data Sets (ds1, ds2) having the same name and the same Value Domain and returns only those Measures summed.

If it is necessary to specify a single Measure, it is possible to use the membership operator (see next paragraph).

Attributes

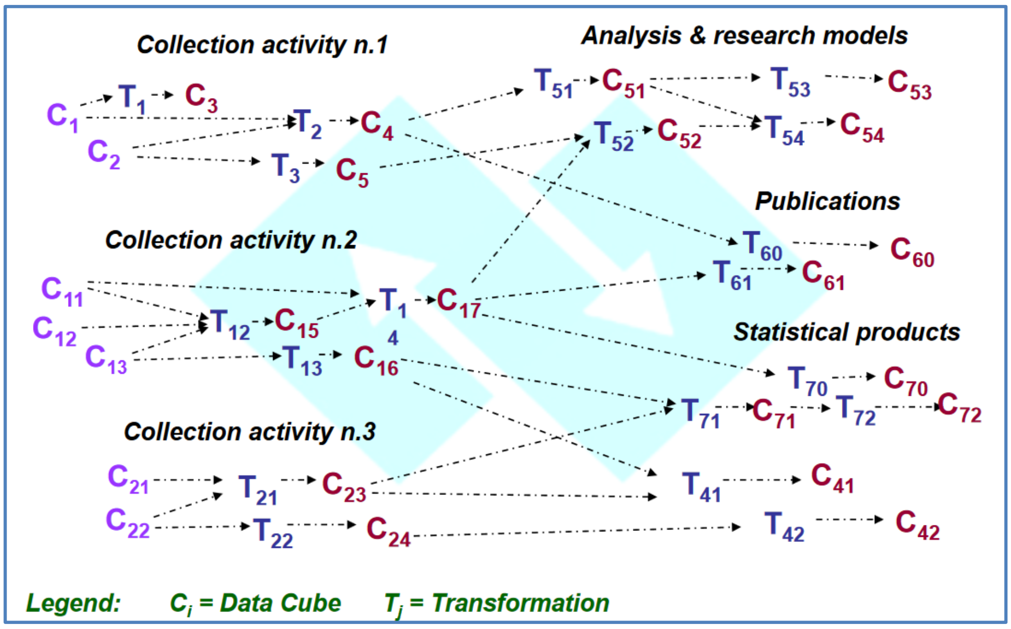
The Attributes are used sometimes to select Data Points where to apply the operator (for example a rule can be valid only for confidential data, OBS_CONF= "C"), sometimes they can represent a value to which the operator can be applied. When the operator is applied on Measures the Attributes can be or not in the resulting Data Set, depending on the importance of the attribute expressed in VTL through the characteristic

[3] For example numeric operators must be applied on numeric measures

to be “viral”. In the examples of the document we will suppose all the attributes to be not “viral” so they will be not included in the resulting dataset.

VTL Transformation

A VTL Transformation (including in the VTL transformation also the validation rules) can be composed by one or more VTL statements. It takes in input one or more Data Sets and transform them in a single, final, output Dataset that can be reused by other transformations, so that it is possible to see the set of transformation as a graph[4].



Initial input Data Sets and final output one, are called “persistent”, instead, Data Sets used inside the VTL statements are called “temporary”.

Operators and functions

VTL operators are organized in groups depending on their field of application. In this document it will be described, in a general way, some groups specifying the most important operators for validation, belonging to each group.

General purpose

The general purpose groups have no specific field of application.

Persistent assignment <-	***Description***: The **<-** operator allows to save the structure of a temporary Data Set making it persistent.
	Return: an output persistent Data Set having the same components of the input Data Set and the same Data Points.
	Example: save the structure of the INEUTRAV_T01_A Data Set DS_R **<-** INEUTRAV_T01_A

[4] Taken by “SDMX technical standards - Data validation and other major enhancements” by Vincenzo del Vecchio, Global Conference 2013.

Assignment, “:=”	**Description**: The assignment operator, indicated with a “:=”, assigns to a Data Set (left side) an expression (right side).
	Return: an association between a Data Set and an expression.
	Example: the Data Set DS_R is the Data Set INEUTRAV_T01_A, filtered by the variable AGE equal to “TOTAL”. DS_R:= INEUTRAV_T01_A [filter AGE= “TOTAL”]
Membership, “#”	***Description***: the membership operator, “**#**”, selects a component into a Data Set.
	Return: A Data Set having all the Identifiers of the input Data Set and one Measure component represented by: 1) the Measure specified by the operator if the Component selected is a Measure, 2) a new Measure assuming the values and the Value Domain of the component specified if the Component selected is not a Measure. The name of the Measure in the case 2) is assumed to be the default name corresponding to the Value domain of the component specified.
	Rule 1): represent the component OBS_VALUE of the dataset Intra_EU_travellers. INEUTRAV_T01_A#OBS_VALUE ***Rule 2)***: represent the component AGE of the dataset Intra_EU_travellers. INEUTRAV_T01_A#AGE

Join operators

The Join operators is a set of operators that allows performing joins among Data Sets that are different from the automatic join mentioned in the “Identifier” paragraph. The joined Data Set can be after manipulated through VTL operators, row by row using the **calc** clause or entirely using the **apply** clause. Is it also possible to **filter** Data Points, **rename** and **drop** or **keep** components and finally aggregate Data Points.

join operators	***Description***: the join operators joins two or more Data Sets and allows to work on the resulting joined Data Setusing different clauses. The possible joins are: • inner_join: all Data Points of all Data Sets matching for Identifiers specified in the **using** clause or on common Identifiers (one Data Set must include all the others Data Sets Identifiers), • left_join: all Data Points of leftmost Data Sets, matching for Identifiers specified in the **using** clause or on all Identifiers (Identifiers must be the same for all the Data Sets) • full_join: inner join, plus the Data Points of all the Data Sets that do not match with the others (**using** is not allowed and Identifiers must be the same for all the Data Sets) • cross_join: cartesian product of the data points of the two Data Sets (**using** is not allowed, no constraints on Identifiers). In the body of the join is possible to: • **filter** data

	• perform Data Sets operations (**apply**) for the whole Data Set or compute new components (**calc**) using VTL operators row by row, or aggregate Data Points (**aggr**) • **drop** or **keep** components • **rename** components
	Return: a Data Set having as Components the superset of all the identifiers of the input Data Sets, as Measures or Attributes all those specified in the body and having as Data Points all deriving by the type of join used and by eventual filters in the body.
	Rule 3)*:* Sum the number of travellers (OBS_VALUE) of PARTNER "BE" and PARTNER "DE" DS_R:=**inner_join(**INEUTRAV_T01_M[filter REPORTING="BE"] **as** DS_BE, INEUTRAV_T01_M [filter REPORTING ="DE"] **as** DS_DE **using** TABLE, FREQ, TIME_PERIOD, PARTNER, DIRECTION, AGE, ADJUST **calc** OBS_VALUE=DS_BE#OBS_VALUE + DS_DE#OBS_VALUE, **identifier** REPORTING= "DE_BE" **drop** DS_BE# REPORTING, DS_DE# REPORTING **)**

Rulesets

The Rulesets have been introduced in version 1.1 to allow the definition of a set of rules in one single block. The Ruleset can be related only to single Data Point (datapoint ruleset) or to more Data Points at the same time through a hierarchical relationship (hierarchical ruleset) defined for a coded component.

datapoint (horizontal) ruleset	***Description***: the **datapoint** Ruleset allows to define one or more rules involving Data Points (horizontal) of the Data Set. The Data Point ruleset can be used with the check_datapoint operator, to verify Data Points respecting the rules. For every rule in the datapoint Ruleset it is possible to specify a subset of Data Points on which the rule is applied, filtering them through a condition on one Component. For every rule it also optionally possible to define a name (e.g. "rule1"), an error code and error level to be returned for erroneous Data Points.
	Return: a datapoint Ruleset
	Rule 4): accepted codes for FREQ are • For TABLE=T01: A • For TABLE=T02: Q **define datapoint ruleset** *TABLE_Periodicity* (variable *TABLE*, *FREQ*) **is** **when** *TABLE* = "T01" **then** *FREQ*="A" **errorcode** "T*able* T01 should contain only Annual series (FREQ=A)" **errorlevel** "Error";

	when *TABLE* = "T02" **then** *FREQ*="Q" **errorcode** "Table T02 should contain only Quarterly series (FREQ=Q)" **errorlevel** "Error" **end datapoint ruleset**
hierarchical (vertical) ruleset	***Description***: the **hierarchical** Ruleset allows to establish relationships based on hierarchical combinations of values for a coded component (vertical). A hierarchical combination of values is a set of comparison using: **=** , **>** ,**<** , **>=** ,**<=** between a code (parent) and a combination of codes (children) involving only the “+” or “- “ operators. The hierarchical Ruleset can be used with the check operators to verify if the comparison are respected (see rule 9) or with the hierarchy operator to compute aggregation using only Ruleset equality comparison. For every rule in the hierarchical Ruleset it is possible to specify a subset of Data Points on which the rule is applied, filtering them through a condition on one Component. For every rule it also possible to define a name, an error code and error level for erroneous data points.
	Return: a hierarchical ruleset

	Rule 5)*:* The sum of partial values of AGE must coincide with the total.
	
	define hierarchical ruleset HR_ AGE_GROUP_AGGREGATE (variable rule **AGE**) **is**
	TOTAL = Y0_18+ Y19_64+ Y65_MAX+ UNK;
	Y0_18 = Y0+…+ Y18;
	Y19_64 = Y19+…+ Y64;
	Y65_MAX = Y65+…+ Y122;
	end hierarchical ruleset

Aggregate and analytic invocation

The aggregate operators contain statistical functions like sum, count and average that can be used to aggregate Data Points of a Data Set.

The aggregation methods are:

- the group by, group except or group all that allows to groups over a set of identifier components,
- the grouping using a condition

aggregate invocation	***Description***: the **aggregate invocation** executes an aggregate operator. It is possible to aggregate Data Points by one or more Identifiers using: • group by - Identifiers to be included • group except - Identifiers to be excluded • group all - one Identifier used in an aggregative expression.

	The group all can be used for example with the time_agg operator. In this case, Data Points are aggregated over the time, from a smaller to a larger period. Aggregate operators are: **avg**, **count, max, median, min, stddev_pop, stddev_samp, sum, var_pop, var_samp**
	Return: returns a Data Set having the identifiers specified in the grouping clause, all the Measures to which the statistical function is applied and as data points all those returned by the type of grouping.
	Rule 6): Calculate the number of Data Points of Data Set INEUTRAV_T01_A, for every combination of identifier values (TABLE, FREQ, TIME_PERIOD, REPORTING, PARTNER, DIRECTION, AGE, ADJUST) Statistical function DS_R:= **count(**INEUTRAV_T01_A **group by** TABLE, FREQ, TIME_PERIOD, REPORTING, PARTNER, DIRECTION, AGE, ADJUST) ; ***Rule 7)***: Calculate the sum of INEUTRAV_T02_Q to obtain the annual value of the OBS_VALUE. DS_R:= **sum (**INEUTRAV_T02_Q **group all time_agg (** "A " **))**
analytic invocation	***Description***: the **analytic invocation** executes an analytic operator. It is possible to analyse data on a <u>sliding window</u> eventually ordered. The "window" of Data Points can be specified through: • a partition of Data Points through one or more Identifiers, • a range of data points or through a definition of a number of preceding and following data points of the current one to be grouped. Analytic operators are: **avg**, **count, max, median, min, stddev_pop, stddev_samp, sum, var_pop, var_samp, first_value, lag, last_value, lead, rank, ratio_to_report**
	Return: a Data Set having as components all Identifiers of the input Data Set, all the Measures to which the statistical function is applied and as Data Points all those returned by the sliding window grouping.
	Rule 8): calculate the average between the yearly data and the preceding one in the annual dataset A11 DS_R:=**avg(**INEUTRAV_T01_A **over (order by** TIME_PERIOD **data points between** 1 **preceding and current data point))** windowing

Validation

The operators in this group are specifically used for validation purpose. These operators take in input a rule (check) or a set of rule (check_datapoint and check_hierarchy) with which validate Data Points.

check _datapoint	***Description***: the **check_datapoint** operator is used to validate a Data Set using a datapoint ruleset.
	Return: return a dataset having as Identifiers all those of the input Data Set and • all the Measures of the input Data Set if the option **invalid** is chosen, • only the Boolean Measure named "bool_var" representing the result of the validation, if option **all** is chosen, • the Boolean Measure together with all the input Measures if option **all_measures** is chosen. Added Components representing the rule name, errorcode and errorlevel, arereturned.
	Rule 9): validate the Data Set INEUTRAV_T01_A using the datapoint ruleset defined in the *Rule 4)* DS_R:=**check_datapoint(**INEUTRAV_T01_A, TABLE_Periodicity, **invalid)**
check_hierarchy	***Description:*** the check_hierarchy operator is used to validate a Data Set using a hierarchical ruleset.
	Return: return a dataset having as Identifiers all those of the input Data Set and • all the Measures of the input Data Set if the option **invalid** is chosen, • only the Boolean Measure named "bool_var" representing the result of the validation, if option **all** is chosen, • the Boolean Measure together with all the input Measures if option **all_measures** is chosen. Added Components representing the rule name, errorcode and errorlevel, are returned.
	Rule 10)*:* validate the Data Set INEUTRAV_T01_A using the hierarchical ruleset defined in the *Rule 5)* DS_R:=**check_hierarchy (**INEUTRAV_T01_A, HR_ AGE_GROUP_AGGREGATE, **invalid)** Name of the hierarchical ruleset of
check	***Description:*** the check operator validates a Data Set through a validation expression.
	As output of the check is possible to define output parameters (errocode, errorlevel and imbalance). The imbalance can be calculated with a numeric expression.
	Rule 11)*:* The values for not empty Measure (OBS_VALUE) for Data Set INEUTRAV_T01_A must be greater or equal to 1 DS_R:=**check** (INEUTRAV_T01_A **[filter** not **isnull(***OBS_VALUE*)**]** >= 1 **errorcode** "Values, when provided, should be higher or equal to 1" **errorlevel** "Warning"**)**

Time Series

This group includes operators specific for time series data.

fill_time_series	***Description***: the **fill_time_series** function is used to include data missed and covering gaps in the series. It inserts, in the Data Set, Data Points that are not present for a time period in a defined frequency. The newly inserted Data Points will have null value for all the Measure Components of the Data Set. Depending on the option specified in the rule, the range for time period to fill series can be related to every series (single) or to the whole Data Set (all).
	Return: a Data Set having the same Components of the input Data Set and having all the Data Points of the input Data Set plus all the missing Data Points in a time period range.
	Rule 12): Insert missing Data Points for the Data Set INEUTRAV_T01_A considering as time period range, the minimum and maximum among all the series. DS_R:= **fill_time_series (**INEUTRAV_T01_A, **all)**
timeshift	***Description***: the **timeshift** operator shifts the time series using a given time-lag. It is implicitly assumed that one Identifier has a time data type. This operator is used when it is necessary to compare data of different years.
	Return: a Data Set having the same Components of the input Data Set and the same Data Points.

	Rule 13): The change in the value from one year to the over should be limited to +/- 10%
	
	INEUTRAV_T01_A_PREC:= **timeshift(**INEUTRAV_T01_A, 1 **)**
	
	DS_R := INEUTRAV_T01_A <= INEUTRAV_T01_A_PREC *1.1

Boolean

The Boolean operators include all the logical operator. These are very common operators like: **and**, **or** and so on, therefore, we will not detail them. The Data Points of the resulting Data Set are all the Data Points of the input Data Set. Input Data Sets that can be used for Boolean operators must have only one measure

String, Numeric and Date

These groups of operators allow to perform operation on strings, numeric or data type components. These are very common operators like: **instr**, **upper** and **lower**, **+**, **-** and so on, therefore, we will not detail them.

Conditional

The conditional operators include all the operators based on a condition. Are part of this group only two operators: the ***nvl*** operator to check if a value is a NULL value or not and the ***if..then..else*** operator to be used when specific conditions affect the result.

Clauses

The clause group derives the name by the peculiar syntax of the operator similar to the SQL clauses. They apply only to the components of the dataset (they cannot be applied to the whole dataset) and are:

- **filter**: to filter data points based on a component value or on a combination of components' values. This clause can be used also to filter data using a datapoint ruleset

- **calc**: to add a component using a computation
- **aggr**: to aggregate a component using the aggregation functions
- **keep**: to specify measures to be kept (the others will be dropt)
- **drop**: to specify measures to be dropt (the others will be kept)
- **rename**: to rename a component
- **pivot**: to transpose several Data Points of the operand Data Set into a single Data Point of the resulting Data Set.
- **unpivot**: To transpose a single Data Point of the operand Data Set into several Data Points of the result Data set
- **sub**: to remove one or more Identifiers, specifying a single value for them

We will provide an example for the subspace operator.

subspace	***Description***: The subspace operator drops one or more Identifier Components assuming fixed values for them. The Data Points in the Data Set are filtered by the values of the Identifiers. The difference with the filter operator is that using the subspace, the Identifiers specified with the subspace operator are dropped by the Data Set while in the filter they are kept assuming the values specified in the filter.
	Return: a Data Set having all Components of the input Data Set except for those indicated in the subscript operators and having as Data Points all those of the input Data Set, filtered by the values of the Components specified in the subspace.
	Rule 14)*:* Checks in all records that OBS_VALUE for AGE="Y0_18" is fewer than the half of OBS_VALUE for AGE="TOTAL" DS_R:= INEUTRAV _T01_A **[sub** *AGE* = "Y0_18" **]** < INEUTRAV_T01_A **[sub** *AGE* = "TOTAL" **]**/2

How to read a validation rule

- Search for how many datasets are involved in the rule and for every of them try to understand clearly the structure of them in terms of identifiers, measures and attributes.
- Depending on the number of datasets involved try to understand the type of join (if it is an automatic join or not) and how it is performed (outer, inner, cross) and data points returned.
- If is used a datapoint ruleset, the rule is applied row wise on all the data points of the dataset or on a subset (depending if it is specified a selection of data points through a condition)
- If it is used a hierarchical ruleset, the rule is applied on groups of records defined by the hierarchies between codes.
- The check operator usually specify the application of the rule (one single rule or ruleset) , by default returns the erroneous records and usually should be the final statement
- Identify when the rule return an error or a warning

How to

Check if the total is the sum of related partials

If total and partials are codes of a component (VAR) of a dataset (DS), this can be done using the hierarchical ruleset

1. Include the equality: total=sum of related partial, in a hierarchical ruleset for example if A is equal to B and C

```
define hierarchical ruleset hr_total( variable rule VAR) is
                A=B+C   errorcode("total different from sum of partials")
end hierarchical ruleset
```

2. check the hierarchical ruleset with the check operator:

```
DS_R:=check_hierarchy(DS, hr_total  invalid)
```

If, instead, the total and partials are different variables (A,B and C) of a dataset (DS)

1. Include the equality in a datapoint ruleset

```
define datapoint ruleset dp_total(variable A, B,C) is
            A=B+C;
end datapoint ruleset
```

2. Check the datapoint ruleset using the check operator

```
DS_R:=check_datapoint(DS, dp_total  invalid)
```

Reuse a rulesets in several statistical domain

Usually the VTL rule are based on a dataset, this means that they are linked to a specific structure and they cannot be reused for different structures. But there are different way introduced in the VTL 1.1 that allows the reusing of rules.

There are some operators or functions that do not use datasets to be defined like he hierarchical ruleset and the datapoint ruleset, they only use the name of components used in them. The dataset is specified only when the ruleset is applied (for example in the check, aggregate or filter operator).

This allows reusing rulesets in different statistical domains.

Another possibility is define a function. A function can include rules defined on parametrized dataset and only when it is used the parameter representing the dataset is instantiated with a real dataset.

This allow reusing the function in several domain applying it to different datasets.

Frequently ask questions

Do the VTL operators and functions always return a dataset?

The rules using dataset(s) as input always return a dataset. The dataset can contains more or one single data point and not necessarily contains both Identifiers and Measures and Attributes. It can have also only one type of component.

The all, any and unique operators that are part of Validation groups, return only a data point and a measure component, the aggregate function can return a dataset with only measures if applied to the whole dataset.

Is VTL a program language?

VTL it not a programming language, it is a language to define rules in a more standardized way and to facilitate the machine-to-machine exchange but it is at higher level of a programming language.

One of the basic requirements of VTL was to be independent from the program language to be implemented in several IT contexts and with several programming language.

However, the approach of most of VTL operators makes the language very close to declarative languages.

Are there any predefined error codes?

The importance of the error code specification arises in particular during the implementation of VTL with a programming language because different type of errors can occur. In the User Manual there is a paragraph representing a guideline on how to express error codes depending on their types:

- compilation errors: errors due to an incorrect use of the syntax or to incorrect data types used in the rule or error due to operator’s constraints not respected
- runtime errors: depending on data like for example the presence of duplicated records or null value for Identifier components
- validation errors: errors due to the violation of the rules by the data

A standardized decoding way is provided to simplify the interpretation of different errors.

Why VTL rules are based on a logical organization of data?

The VTL rules are based on logical organization of data (dataset) and not on physical organization of data for the main reason that rules usually, are not dependent on the physical organization of data, they are only a consequence of relationships between the variables. This allows specifying VTL statements in a way independent by the physical organization of data that can be change from country to country.

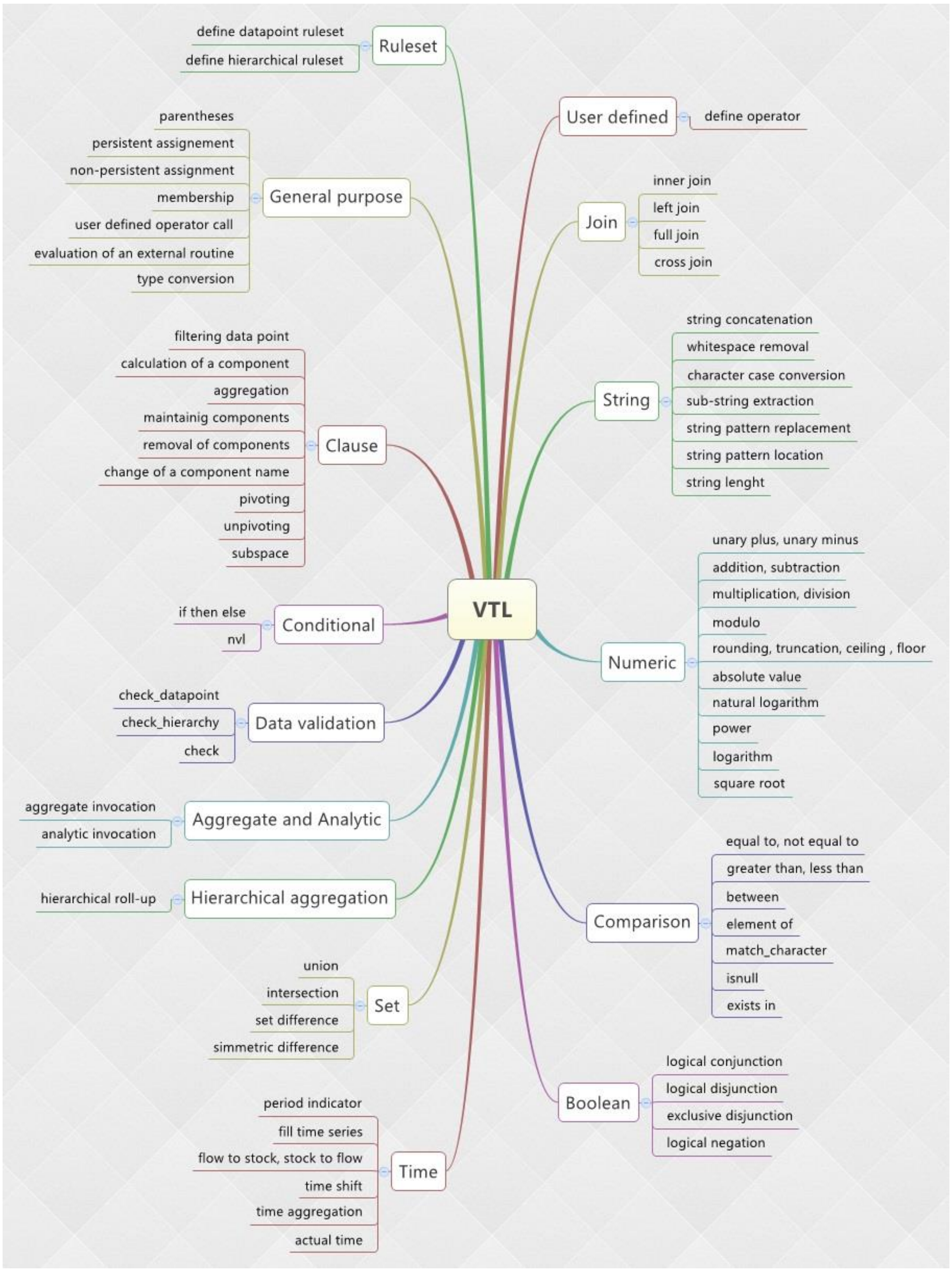
How VTL manage with duplication of records?

A data point in a dataset is duplicated when the same combination of values of the identifier components is duplicated in the dataset. This can occur using the get operator or using some set operators: union and intersect. In this case, VTL provide a type of deduplication function that allows to choice a strategy to avoid duplications. For example the user can decide to take, in case of duplication, all data points of the first dataset or all data points that are the greatest between the values.

Appendix

Picture of all VTL operators and functions

This picture provides a complete view of all the operators and functions of VTL.


VTL
Ruleset
define datapoint ruleset
define hierarchical ruleset
General purpose
parentheses
persistent assignement
non-persistent assignment
membership
user defined operator call
evaluation of an external routine
type conversion
Clause
filtering data point
calculation of a component
aggregation
maintainig components
removal of components
change of a component name
pivoting
unpivoting
subspace
Conditional
if then else
nvl
Data validation
check_datapoint
check_hierarchy
check
Aggregate and Analytic
aggregate invocation
analytic invocation
Hierarchical aggregation
hierarchical roll-up
Set
union
intersection
set difference
simmetric difference
Time
period indicator
fill time series
flow to stock, stock to flow
time shift
time aggregation
actual time
User defined
define operator
Join
inner join
left join
full join
cross join
String
string concatenation
whitespace removal
character case conversion
sub-string extraction
string pattern replacement
string pattern location
string lenght
Numeric
unary plus, unary minus
addition, subtraction
multiplication, division
modulo
rounding, truncation, ceiling , floor
absolute value
natural logarithm
power
logarithm
square root
Comparison
equal to, not equal to
greater than, less than
between
element of
match_character
isnull
exists in
Boolean
logical conjunction
logical disjunction
exclusive disjunction
logical negation

Rules with figures

In this paragraph all the rules defined in the handbook will be applied to a real set of data. There will be not examples about the VTL DDL language because they are already detailed in the document.

Membership operator

INEUTRAV_T01_A									
TABLE	FREQ	TIME_PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_ VALUE	OBS_ STATUS
T01	A	2013-11	BE	IT	IN	TOTAL	N	637	NULL
T01	A	2013-11	BE	IT	IN	Y0_18	N	77	NULL
T01	A	2013-11	BE	IT	OUT	TOTAL	N	210	NULL
T01	A	2013-11	BE	IT	OUT	Y0_18	N	61	NULL
T01	A	2013-11	DE	IT	IN	TOTAL	N	700	NULL
T01	A	2013-11	DE	IT	IN	Y0_18	N	90	NULL
T01	A	2013-11	DE	IT	OUT	TOTAL	N	240	NULL
T01	A	2013-11	DE	IT	OUT	Y0_18	N	50	NULL

Rule 1): Represent the component OBS_VALUE of the dataset INEUTRAV_T01_A.

DS_R:= INEUTRAV_T01_A#OBS_VALUE

DS_R								
TABLE	FREQ	TIME_PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_VALUE
T01	A	2013-11	BE	IT	IN	TOTAL	N	637
T01	A	2013-11	BE	IT	IN	Y0_18	N	77
T01	A	2013-11	BE	IT	OUT	TOTAL	N	210
T01	A	2013-11	BE	IT	OUT	Y0_18	N	61
T01	A	2013-11	DE	IT	IN	TOTAL	N	700
T01	A	2013-11	DE	IT	IN	Y0_18	N	90
T01	A	2013-11	DE	IT	OUT	TOTAL	N	240
T01	A	2013-11	DE	IT	OUT	Y0_18	N	50

Rule 2): represent the component AGE of the dataset Intra_EU_travellers.

DS_R:= INEUTRAV_T01_A#AGE

DS_R								
TABLE	FREQ	TIME_PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	string_var
T01	A	2013-11	BE	IT	IN	TOTAL	N	TOTAL
T01	A	2013-11	BE	IT	IN	Y0_18	N	Y0_18
T01	A	2013-11	BE	IT	OUT	TOTAL	N	Y19_64
T01	A	2013-11	BE	IT	OUT	Y0_18	N	Y65_MAX
T01	A	2013-11	DE	IT	IN	TOTAL	N	TOTAL
T01	A	2013-11	DE	IT	IN	Y0_18	N	Y0_13
T01	A	2013-11	DE	IT	OUT	TOTAL	N	TOTAL

T01	A	2013-11	DE	IT	OUT	Y0_18	N	Y0_13

Join operators

Rule 3): Sum the number of travellers (OBS_VALUE) of REPORTING "BE" and REPORTING "DE"

DS_R:=**inner_join(**INEUTRAV_T01_A [filter REPORTING="BE"] **as** DS_BE,

INEUTRAV_T01_A [filter REPORTING ="DE"] **as** DS_DE

using TABLE, FREQ, TIME_PERIOD, PARTNER, DIRECTION, AGE, ADJUST

calc OBS_VALUE:=DS_BE#OBS_VALUE + DS_DE#OBS_VALUE,

identifier REPORTING:= "DE_BE"

drop DS_BE# REPORTING, DS_DE# REPORTING, DS_BE#OBS_VALUE, DS_DE#OBS_VALUE, DS_BE#OBS_STATUS, DS_DE#OBS_STATUS **);**

INEUTRAV_T01_A

TABLE	FREQ	TIME_PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_ VALUE	OBS_ STATUS
T01	A	2013-11	BE	IT	IN	TOTAL	N	637	NULL
T01	A	2013-11	BE	IT	IN	Y0_18	N	77	NULL
T01	A	2013-11	BE	IT	OUT	TOTAL	N	210	NULL
T01	A	2013-11	BE	IT	OUT	Y0_18	N	61	NULL
T01	A	2013-11	DE	IT	IN	TOTAL	N	700	NULL
T01	A	2013-11	DE	IT	IN	Y0_18	N	90	NULL
T01	A	2013-11	DE	IT	OUT	TOTAL	N	240	NULL
T01	A	2013-11	DE	IT	OUT	Y0_18	N	50	NULL

The input Data Set INEUTRAV_T01_A, filtered by REPORTING="BE" (INEUTRAV_T01_A [filter PARTNER="BE"]), having alias DS_BE is:

DS_BE

TABLE	FREQ	TIME_PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_ VALUE	OBS_ STATUS
T01	A	2013-11	BE	IT	IN	TOTAL	N	637	NULL
T01	A	2013-11	BE	IT	IN	Y0_18	N	77	NULL
T01	A	2013-11	BE	IT	OUT	TOTAL	N	210	NULL
T01	A	2013-11	BE	IT	OUT	Y0_18	N	61	NULL

The input Data Set INEUTRAV_T01_A, filtered by REPORTING="DE" (INEUTRAV_T01_A [filter PARTNER="DE"]), having alias DS_DE is:

DS_DE

TABLE	FREQ	TIME_PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_ VALUE	OBS_ STATUS
T01	A	2013-11	DE	IT	IN	TOTAL	N	700	NULL

T01	A	2013-11	DE	IT	IN	Y0_18	N	90	NULL
T01	A	2013-11	DE	IT	OUT	TOTAL	N	240	NULL
T01	A	2013-11	DE	IT	OUT	Y0_18	N	50	NULL

The join is performed on the same values of TABLE, FREQ, TIME_PERIOD, PARTNER, DIRECTION, AGE, ADJUST Identifiers and returns a Data Set having:

- all the Identifiers used for the join with no repetition,
- the other Components of the two Data Sets with no repetition and with the measure OBS_VALUE assuming value specified in the **calc** statement:
-

OBS_VALUE= DS_BE#OBS_VALUE + DS_DE#OBS_VALUE

DS_R									
TABLE	FREQ	TIME_PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_ VALUE	OBS_ STATUS
T01	A	2013-11	DE_BE	IT	IN	TOTAL	N	1337	NULL
T01	A	2013-11	DE_BE	IT	IN	Y0_18	N	167	NULL
T01	A	2013-11	DE_BE	IT	OUT	TOTAL	N	450	NULL
T01	A	2013-11	DE_BE	IT	OUT	Y0_18	N	111	NULL

Rulesets

datapoint (horizontal) ruleset

Rule 4): accepted codes for FREQ are

- For TABLE=T01: A
- For TABLE=T02: Q

define datapoint ruleset *TABLE_Periodicity* (variable *TABLE*, *FREQ*) **is**
    when *TABLE* = "T01" **then** *FREQ*="A"
    errorcode "T*able* T01 should contain only Annual series (FREQ=A)"
    errorlevel "Error";

    when *TABLE* = "T02" **then** *FREQ*="Q"
    errorcode "Table T02 should contain only Quarterly series (FREQ=Q)"
    errorlevel "Error";
end datapoint ruleset ;

Rule is applied on datapoints (horizontally)

INEUTRAV_T01_A									
TABLE	FREQ	TIME_PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_ VALUE	OBS_ STATUS
T01	Q	2016	FR	DE	IN	TOTAL	N	200	
T01	A	2016	FR	ES	IN	TOTAL	N	150	

T01	A	2016	FR	ES	OUT	TOTAL	N	160	

hierarchical (vertical) ruleset

***Rule 5)**: The sum of partial values of AGE must coincide with the total.*

define hierarchical ruleset HR_ AGE_GROUP_AGGREGATE (variable rule **AGE**) **is**

TOTAL = Y0_18+ Y19_64+ Y65_MAX+ UNK;

Y0_18 = Y0+…+ Y18;

Y19_64 = Y19+…+ Y64;

Y65_MAX = Y65+…+ Y122;

end hierarchical ruleset

Rule is based on a hierarchy of a list of codes of AGE(vertically)

Rule is applied to the OBS_VALUE measure

INEUTRAV_T01_A									
TABLE	**FREQ**	**TIME_PERIOD**	**REPORTING**	**PARTNER**	**DIRECTION**	**AGE**	**ADJUST**	**OBS_ VALUE**	**OBS_ STATUS**
T01	Q	2016	FR	DE	IN	TOTAL	N	200	
T01	A	2016	FR	ES	IN	TOTAL	N	150	
T01	A	2016	FR	ES	OUT	TOTAL	N	160	

Aggregate invocation

***Rule 6)**:* Calculate the number of Data Points of Data Set INEUTRAV_T01_M, for every combination of identifier values (TABLE, FREQ, TIME_PERIOD, REPORTING, PARTNER, DIRECTION, AGE, ADJUST)DS_R:= **count(**INEUTRAV_T01_A **group by** TABLE, FREQ, TIME_PERIOD, REPORTING, PARTNER, DIRECTION, AGE, ADJUST)

INEUTRAV_T02_Q								
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTION**	**AGE**	**ADJUST**	**OBS_VALUE**
T02	Q	2016-Q3	DE	ES	IN	TOTAL	N	20
T02	Q	2016-Q3	DE	ES	IN	TOTAL	S	15
T02	Q	2016-Q3	DE	ES	OUT	TOTAL	N	21
T02	Q	2016-Q3	DE	ES	OUT	TOTAL	N	25
T02	Q	2016-Q3	DE	ES	OUT	TOTAL	S	16

The rule counts the number of Data Points having the same values for identifiers: TABLE, FREQ, TIME_PERIOD, REPORTING, PARTNER, DIRECTION, AGE, ADJUST.

DS_R								
TABLE	**FREQ**	**TIME_ PERIOD**	**REPORTING**	**PARTNER**	**DIRECTION**	**AGE**	**ADJUST**	**OBS_VALUE**
T02	Q	2016-Q3	DE	ES	IN	TOTAL	N	1
T02	Q	2016-Q3	DE	ES	IN	TOTAL	S	1
T02	Q	2016-Q3	DE	ES	OUT	TOTAL	N	2
T02	Q	2016-Q3	DE	ES	OUT	TOTAL	S	1

***Rule 7)**:* Calculate the sum of INEUTRAV_T02_Q to obtain the annual value of the OBS_VALUE.

DS_R:= **sum (**INEUTRAV_T02_Q **group all time_agg** ("A"**))**

INEUTRAV_T02_Q								
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_VALUE
T02	Q	2016-Q1	DE	ES	IN	TOTAL	N	20
T02	Q	2016-Q2	DE	ES	IN	TOTAL	N	15
T02	Q	2016-Q3	DE	ES	IN	TOTAL	N	21
T02	Q	2016-Q4	DE	ES	IN	TOTAL	N	25
T02	Q	2017-Q1	DE	ES	IN	TOTAL	N	16

Data are aggregated over the TIME_PERIOD component passing from quarterly to annual data.

DS_R								
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_VALUE
T02	Q	2016A	DE	ES	IN	TOTAL	N	81
T02	Q	2017A	DE	ES	IN	TOTAL	N	16

Analitic invocation

***Rule 8)**:* calculate the average between the yearly data and the preceding one in the annual dataset A11

DS_R:=**avg(**INEUTRAV_T01_A **over (order by** TIME_PERIOD **data points between** 1 **preceding and current data point))**

INEUTRAV_T02_Q								
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_VALUE
T02	Q	2016-Q1	DE	ES	IN	TOTAL	N	20
T02	Q	2016-Q2	DE	ES	IN	TOTAL	N	15
T02	Q	2016-Q3	DE	ES	IN	TOTAL	N	21
T02	Q	2016-Q4	DE	ES	IN	TOTAL	N	25
T02	Q	2017-Q1	DE	ES	IN	TOTAL	N	16

DS_R								
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_VALUE
T02	Q	2016-Q1	DE	ES	IN	TOTAL	N	20
T02	Q	2016-Q2	DE	ES	IN	TOTAL	N	15
T02	Q	2016-Q3	DE	ES	IN	TOTAL	N	18
T02	Q	2016-Q4	DE	ES	IN	TOTAL	N	23
T02	Q	2017-Q1	DE	ES	IN	TOTAL	N	20.5

check

***Rule 9)*:** validate the Data Set INEUTRAV_T01_A using the datapoint ruleset defined in the *Rule 4)*

DS_R:=**check_datapoint(**INEUTRAV_T01_A[keep OBS_VALUE], TABLE_Periodicity **invalid)**

INEUTRAV_T01_A									
TABLE	FREQ	TIME_PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_ VALUE	OBS_ STATUS
T01	Q	2016	FR	DE	IN	TOTAL	N	200	
T01	A	2016	FR	ES	IN	TOTAL	N	150	
T01	A	2016	FR	ES	OUT	TOTAL	N	160	

DS_r											
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_ VALUE	ruleid	error code	error level
T01	Q	2016	FR	DE	IN	TOTAL	N	200	NULL	Table T01 should contain only Annual series (FREQ=A)	Error

Rule 10): validate the Data Set INEUTRAV_T01_A using the hierarchical ruleset defined in the *Rule 5)*

DS_R:=**check_hierarchy(**INEUTRAV_T01_A, HR_ AGE_GROUP_AGGREGATE **invalid)**

INEUTRAV_T01_A									
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_VALUE	OBS_STATUS
T01	A	2016	FR	IT	IN	TOTAL	N	201	
T01	A	2016	FR	IT	IN	Y0_18	N	101	
T01	A	2016	FR	IT	IN	Y19_64	N	50	
T01	A	2016	FR	IT	IN	Y65_MAX	N	50	
T01	A	2016	FR	ES	IN	TOTAL	N	204	
T01	A	2016	FR	ES	IN	Y0_18	N	100	
T01	A	2016	FR	ES	IN	Y19_64	N	50	
T01	A	2016	FR	ES	IN	Y65_MAX	N	50	

DS_R									
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_VALUE	OBS_STATUS
T01	A	2016	FR	ES	IN	TOTAL	N	204	
T01	A	2016	FR	ES	IN	Y0_18	N	100	
T01	A	2016	FR	ES	IN	Y19_64	N	50	
T01	A	2016	FR	ES	IN	Y65_MAX	N	50	

Rule 11): The values for not empty Measure (OBS_VALUE) for Data Set INEUTRAV_T01_A must be greater or equal to 1

DS_R:=**check** (INEUTRAV_T01_A **[filter** not **isnull**(*OBS_VALUE*)**]** >= 1

errorcode "Values, when provided, should be higher or equal to 1"

errorlevel "Warning"**)**

INEUTRAV_T01_A									
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_VALUE	OBS_STATUS
T02	Q	2016-Q3	FR	ES	IN	TOTAL	N	0	
T02	Q	2016-Q3	FR	ES	IN	TOTAL	S	-15	
T02	Q	2016-Q3	FR	ES	OUT	TOTAL	N	21	P
T02	Q	2016-Q3	FR	ES	OUT	TOTAL	S	16	P

DS_R										
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	bool_ var	errorcode	error level
T02	Q	2016-Q3	FR	ES	IN	TOTAL	N	FALSE	Values, when provided, should be higher or equal to 1	Warni ng
T02	Q	2016-Q3	FR	ES	IN	TOTAL	S	FALSE	Values, when provided, should be higher or equal to 1	
T02	Q	2016-Q3	FR	ES	OUT	TOTAL	N	TRUE		
T02	Q	2016-Q3	FR	ES	OUT	TOTAL	S	TRUE		

fill_time_series

Rule 12): Insert missing Data Points for the Data Set INEUTRAV_T01_A considering as time period range the minimum and maximum among all the series.

DS_R:= **fill_time_series (**INEUTRAV_T01_A, **all)**

INEUTRAV_T01_A									
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_ VALUE	OBS_ STATUS
T01	A	2008	FR	DE	IN	TOTAL	N	200	
T01	A	2009	FR	DE	IN	TOTAL	N	203	
T01	A	2010	FR	DE	IN	TOTAL	N	202	
T01	A	2008	FR	ES	IN	TOTAL	N	150	
T01	A	2010	FR	ES	IN	TOTAL	N	158	
T01	A	2011	FR	DE	OUT	TOTAL	N	210	

DS_R									
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_ VALUE	OBS_ STATUS
T01	A	2008	FR	DE	IN	TOTAL	N	200	
T01	A	2009	FR	DE	IN	TOTAL	N	203	
T01	A	2010	FR	DE	IN	TOTAL	N	202	
T01	A	2011	FR	DE	IN	TOTAL	N	NULL	
T01	A	2008	FR	ES	IN	TOTAL	N	150	
T01	A	2009	FR	ES	IN	TOTAL	N	NULL	
T01	A	2010	FR	ES	IN	TOTAL	N	158	
T01	A	2011	FR	ES	IN	TOTAL	N	210	

timeshift

Rule 13)*: The change in the value from one year to the over should be limited to +/- 10%*

INEUTRAV_T01_A_PREC:= **timeshift(**INEUTRAV_T01_A, 1 **)**

DS_R := INEUTRAV_T01_A <= INEUTRAV_T01_A_PREC *1.1

INEUTRAV_T01_A									
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_ VALUE	OBS_ STATUS
T01	A	2008	FR	DE	IN	TOTAL	N	200	
T01	A	2009	FR	DE	IN	TOTAL	N	203	
T01	A	2010	FR	DE	IN	TOTAL	N	202	
T01	A	2011	FR	DE	IN	TOTAL	N	180	
T01	A	2008	FR	ES	IN	TOTAL	N	150	
T01	A	2009	FR	ES	IN	TOTAL	N	190	
T01	A	2010	FR	ES	IN	TOTAL	N	158	
T01	A	2011	FR	ES	IN	TOTAL	N	210	

INEUTRAV_T01_A_PREC									
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_ VALUE	OBS_ STATUS
T01	A	2009	FR	DE	IN	TOTAL	N	200	
T01	A	2010	FR	DE	IN	TOTAL	N	203	
T01	A	2011	FR	DE	IN	TOTAL	N	202	
T01	A	2012	FR	DE	IN	TOTAL	N	180	
T01	A	2009	FR	ES	IN	TOTAL	N	150	
T01	A	2010	FR	ES	IN	TOTAL	N	190	
T01	A	2011	FR	ES	IN	TOTAL	N	158	
T01	A	2012	FR	ES	IN	TOTAL	N	210	

DS_R								
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_VALUE
T01	A	2009	FR	DE	IN	TOTAL	N	TRUE
T01	A	2010	FR	DE	IN	TOTAL	N	TRUE
T01	A	2011	FR	DE	IN	TOTAL	N	TRUE
T01	A	2008	FR	ES	IN	TOTAL	N	TRUE
T01	A	2009	FR	ES	IN	TOTAL	N	FALSE
T01	A	2010	FR	ES	IN	TOTAL	N	TRUE
T01	A	2011	FR	ES	IN	TOTAL	N	FALSE

sub

***Rule 14)**: Checks in all records that OBS_VALUE for AGE=”Y0_18” is fewer than the half of OBS_VALUE for AGE=”TOTAL”*

DS_R:= INEUTRAV _T01_A **[sub** *AGE* = "Y0_18"**]** < INEUTRAV_T01_A **[sub** *AGE* = "TOTAL"**]**/2 ;

INEUTRAV_T01_A									
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	AGE	ADJUST	OBS_VALUE	OBS_STATUS
T01	A	2016	FR	ES	IN	TOTAL	N	200	
T01	A	2016	FR	ES	IN	Y0_18	N	120	
T01	A	2016	FR	ES	OUT	TOTAL	N	100	
T01	A	2016	FR	ES	OUT	Y0_18	N	45	

INEUTRAV _T01_A [sub AGE = "Y0_18"]								
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	ADJUST	OBS_VALUE	OBS_STATUS
T01	A	2016	FR	ES	IN	N	120	
T01	A	2016	FR	ES	OUT	N	45	

INEUTRAV_T01_A [sub AGE = "TOTAL"]								
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	ADJUST	OBS_VALUE	OBS_STATUS
T01	A	2016	FR	ES	IN	N	200	
T01	A	2016	FR	ES	OUT	N	100	

DS_R								
TABLE	FREQ	TIME_ PERIOD	REPORTING	PARTNER	DIRECTION	ADJUST	bool_var	OBS_STATUS
T01	A	2016	FR	ES	IN	N	FALSE	
T01	A	2016	FR	ES	OUT	N	TRUE	