Macro-selection and micro-editing: a prototype

Wim Hacking, Statistics Netherlands

Introduction

Micro-editing procedures have a number of problems: often there are many checks with narrow tolerances resulting in too many mistakes that need to be resolved manually by analysts. The analysts cannot assess the relative importance of these errors. Each marked item has the same weight and needs the same amount of time for correction. However, many errors have a neglible impact on the final estimates: either they are small or they cancel out.

In general the boundaries for micro-checks are set conservatively, i.e. only error-less records are accepted. An example of a frequently used edit is that the relative change compared to the last period cannot exceed ± 10%. Generally, this approach results in a considerable amount of over-editing.

To address the issue of over-editing one can use macro-editing; this method is already in use at a number of places at Statistics Netherlands and other statistical offices, based on applications tailored for (a number) of statistics. An important difference compared to micro-editing is its use of aggregates, to assess the relative importance of a set of records. This is done by comparing concept-publications with those of the last period**, as a final check**. The largest deviations are either displayed on top or marked visually (e.g. by using colours). Based on suspect publication data the underlying data can be viewed, checked for inconsistencies and corrected; immediately after the correction at the micro-level one can check whether the reviewed publication data fits better in the time series. Another example of macro-editing is outlier detection where the (multivariate) distribution determines which records are suspect and possibly need to be corrected. Such a distribution can be obtained by comparing with t-1 data, but also by considering two (or more) related variables in one (or multiple) scatter plots; in these plots outliers may be marked based on visual inspection. Alternatively, these outliers may be determined automatically and subsequently be displayed, e.g. by colouring these points inside a scatter plot. A third example is prioritising (based on measures supplied by the analyst) of records: records having the highest plausibility index are listed first. This allows more direct and efficient manual editing. Some simulation studies (Granquist) rapport an efficiency gain between 35% and 80%.

These macro-editing approaches not only result in less work for the analist**,** but also in a more meaningful analysis and editing process. He can always see the implications of his corrections at the micro-level on the output-aggregates.

These macro-editing techniques can also be used for the evaluation of (incoming) register data: based on appropriate aggregates one can compare the current version of the register with previous ones at the macro-level. After finding differences at the macro-level one can zoom in on more detailed levels and, finally, inspect and correct records.

Although applications of the principles above exist at various statistical offices but no general software has been made yet, which can be applied to different statistics, to our knowledge. During the last half year a prototype has been developed at Statistics Netherlands that has been designed to serve as such a tool. This doesn't mean that the tool is finished; in a certain way it's just a starting point to discover which macro-detection strategies work in practice and which not. This discovery process is still going on. Currently, we are working with data from road-transport and production.

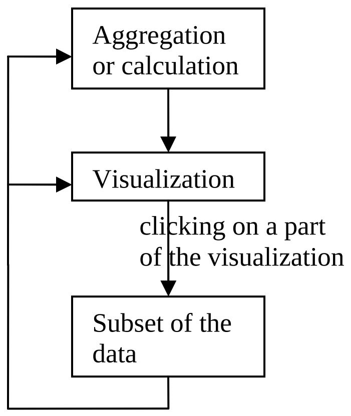
Main idea

Macro-selection is always based on a collection of records to make a selection of those records that contribute the most to errors observed at some aggregate level. In order to asses this error one must have a reference set: this may be the data set itself (e.g. using outlier detection) or (aggregated) reference data (using *t-x* data or data strongly correlated to the variables under study). The reference datasets can be used to make a prediction D_p of the current data D_t and the relative differences between the prediction and the actual values can be used to look for possible errors. For this, the program has to be able to describe and access reference data, either micro-data or aggregated data**.**

During the macro-selection process, the data needs to be displayed in various ways to enable the analyst to zoom in or analyze the data. Looking at outliers and deviations at the aggregate level the analyst can either zoom in or apply another visualization, e.g. other variables in a scatter plot. Finally, at the most detailed level, the analyst needs to be able to edit a single record. To configure all of these possibilities, a project file can be specified in which the analyst needs to specify the elements of the editing process and, finally, the interactive process itself:

1. The input of the editing process: a data model of the microdata input and of the reference (aggregate) data.
2. The variables and their derivation in the aggregate(s) (e.g. plausibility functions, distribution moments, …)
3. Visualizations: what data needs to be displayed, which colours needs to be used, etc.
4. The behaviour of the macro-edit program when interacting with the analyst. For example, when the analyst selects a cell from an aggregate: should a sub-aggregate be displayed or a scatter plot?

Based on the elements defined above the selection process can be depicted as follows:

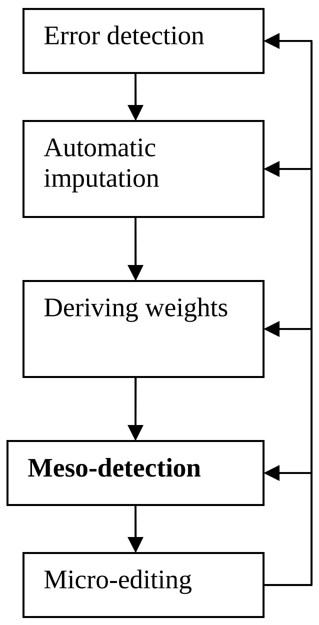

Aggregation or calculation
Visualization
clicking on a part of the visualization
Subset of the data

To make this flexible one needs a flexible description of the data that is being transported from one visualization to another. For example, consider an aggregate by A with possible values (1, 2, 3) and variables AVG(X), AVG(Y) and AVG(Z); the selected cells A=1 are selected from this aggregate can be considered as:

- As one (aggregated) record : (1, AVG(X), AVG(Y), AVG(Z))
- As a selection on the original microdata: all microdata records having A = 1

It means that the information passed on between visualizations must contain information on both aspects.

The are also some additional, practical, requirements: the macro-editing process often requires many records, e.g. millions of records. Even in the presence of such large amounts of records the program should still maintain its interactive character, i.e. the response-time (especially when records have been edited and tables or plots need to be updated) must be small. Secondly, the program must have an open character: the current version already uses some external programs to do its task (Blaise for editing, Bascula for (re)weighting and R for things like outlier detection). At the same time, to the analyst it should appear as if he’s working with one tool. In other words, the integration between the macro-edit tool and the external programs should be quite good. Last, but not least, the tool needs to be flexible, so that it can be used for all kinds of statistics; in the current prototype a text-based user-interface is used to enable this. In a later version, a graphical user-interface will be added.



In the rest of the paper we will describe the elements of the specification in more detail and show some results for **production** data. The specifications are defined in a Blaise/Manipula-like syntax and in each section an example specification is shown and described.

Specifying the input

Microdata is specified as a Blaise datamodel, albeit with a few additional properties per field.

```
DATAMODEL MyData
PRIMARY
      ID
FIELDS
      RitAfst:real
      RitAant:integer
      rittype: integer
      btogew: real, ifmissing(0.0)
RULES
      rittype < ritaant
ENDBLOCK
```

When a value is missing, one can specify what default value should be used instead. In this example, `ifmissing(0.0)` means that whenever the variable `btogew` is missing, it is being replaced by 0.0.

In addition to the (current) microdata, one needs to specify the aggregate data. For this, two sections are added: `AGGREATEBY` and `FILTERBY`:

```
DATAMODEL MyAggData
AGGREGATEBY
      AggDef =  Region * CompanySize
      AggDef2 = Region
      AggDef3 = CompanySize
FILTERBY
      Filter1 = “Profit > 1000”
FIELDS
      Region: string
      CompanySize: string
      avg_distance_now:real
      avg_distance_prev:real
ENDMODEL
```

These aggregate data models have two purposes:

1. to describe *existing* aggregate data (e.g. *t-1* data or any other correlated data); in this case one needs to specify what variables have been used to aggregate (`AGGREGATEBY`); also a previous selection (before the aggregation) can be specified (`FILTERBY`). Precisely one aggregate definition must and at most one filter can be specified.
2. to describe *new* aggregates: in this more than one aggregate and multiple filters can be specified and be combined.

Specifying the aggregation

Based on the datamodels above, one can define various aggregate functions to detect discrepancies between current and reference data. These functions can be

a. Distribution properties of the microdata, e.g. its variance.
b. Processing properties, such as percentage of item non-response or percentage of previously automatically imputed values in field X
c. Plausibility functions: e.g. the relative change between weighted t-1 and t data:

$$\Delta = \frac{\left|\sum w_{i,t-1}V_{i,t-1} - \sum w_{i,t}V_{i,t}\right|}{w_{i,t-1}V_{i,t-1}}$$

where $w_{i,t(-1)}$ are the weights and $V_{i,t(-1)}$ are the values of one of the variables.

An example of each is contained in the following example section (5a, 5b and 5c)

```
AGGREGATE MyAgg                                               {1}
INPUT
      CurrYear : MyData                                 {2}
      LastYearAgg : MyAggData                           {3}
OUTPUT
      outputagg : MyAggDataWithDifferences              {4}
CELLS
      sum_distance_now := SUM(CurrYear.distance);             {5}
      sum_distance_last := LastYearAgg.sum_distance;

      z := VAR(CurrYear.Profit);                              {5a}
      PercFilled := COUNT(CurrYear.distance)/COUNT(ALL);      {5b}
      difference := ABS(LastYearAgg.sum_distance - AVG(CurrYear.distance *
CurrYear.Weight))/ LastYearAgg.sum_distance {< 0.1};          {5c}
CELLCOMPARE
AggDef2:                                                      {6}
      test1 := avg_distance_now('west')
                        /avg_distance_now('east') > 2.0
                        "There should be more transport in the west of the
                        Netherlands";

ENDAGGREGATE
```

In the text above the following things are specified:

1. We define an aggregate named “MyAggData”

2. Each aggregate has *one* microdata input which contains the current data that needs macro-editing. CurrYear is an internal identifier which can be referenced from the cells section, e.g. CurrYear.Distance. The reference MyData refers to datamodel defined earlier on.
3. Each `AGGREGATE` has one or more additional reference aggregates, that can be joined to the aggregated microdata (i.e. CurrYear). An example of such an aggregate may be *t-1* data.
4. The output refers to the datamodel of the aggregate as calculated in this `AGGREGATE` section.
5. The section `CELLS` contains the derivation of the output variables. These can be simple aggregates (like `SUM(X)`), but can also contains a mix of variables form the various input sources. Note however, that the variables from the microdata can only enter these expressions inside an aggregate function, e.g. `SUM(X).` Available aggregates are `AVG(X),STDEV(X), MIN(X), MAX(X), COUNT(X), SUM(X), MEDIAN(X), ALPHATRIMMEDMEAN(X, AlphaLEFT, AlphaRight, UseMedian), INTERQUARTILE( X)`
6. Finally, an additional set of **ratios** can be derived based on the aggregate data. For example **ratio** `test1` expresses the check that generally `"There should be more transport in the west of the Netherlands"`. The expression `avg_distance_now('west')`indicates the column avg_distance for the aggregate Region=’west’.

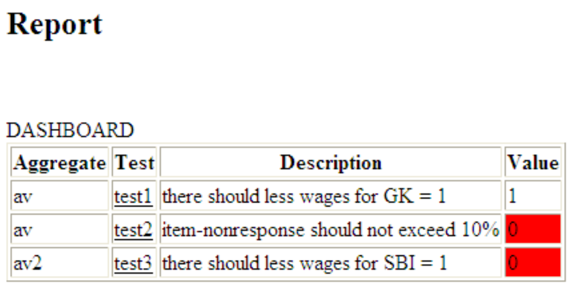
In addition, it’s possible to use conditions inside an aggregate function like `SUM, AVG,` etc.:

```
TotalDistance := SUM(
                           IF RitInd = 'R' THEN
                                 weight * Dist
                            ELSEIF RitInd = 'Z' THEN
                                 weight * Dist
                            ELSE
                                 0.0
                            ENDIF
                           );
```

Specifying the visualizations

In each visualization the analyst can click on certain parts, leading to a sub-selection of the data under study. For example, by clicking on a cell from an aggregate (aggregated by A en B, say), all (micro)records from that cell (from A=1 en B=2, say) are selected. The prototype currently contains three kinds of visualizations:

1. Report: at the end of each aggregate a number of **ratios** can be defined. These numbers from each calculated aggregate are shown in a overview report. Clicking on a **ratio** can be used to show the underlying aggregate.



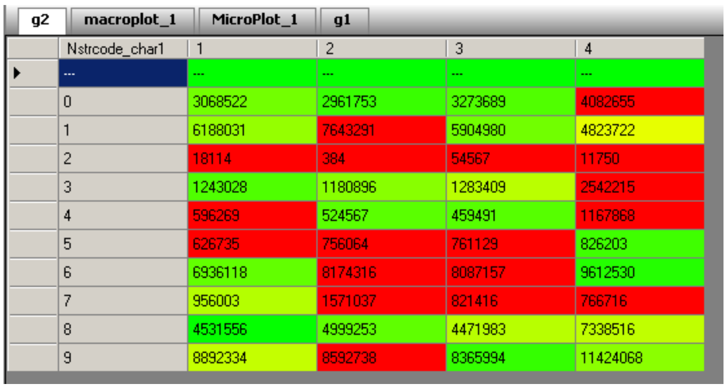
Report

DASHBOARD

Aggregate	Test	Description	Value
av	test1	there should less wages for GK = 1	1
av	test2	item-nonresponse should not exceed 10%	0
av2	test3	there should less wages for SBI = 1	0

In this report three ratios are shown for two aggregate (*av* and *av2*). Two of the ratios are not OK and are shown in red.

2. (Pivot) tables: used to show either aggregated or micro data. It’s possible to give cells from variable V_1 a background colour based on a variable V_2.

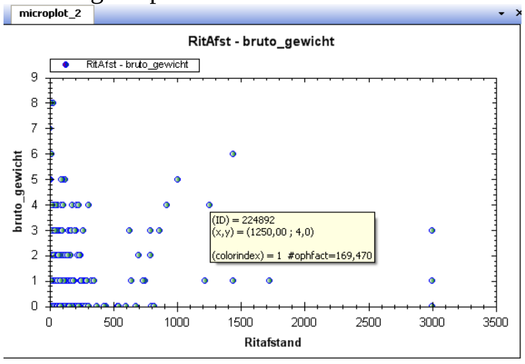


g2 | macroplot_1 | MicroPlot_1 | g1

Nstrcode_char1	1	2	3	4
---	---	---	---	---
0	3068522	2961753	3273689	4082655
1	6188031	7643291	5904980	4823722
2	18114	384	54567	11750
3	1243028	1180896	1283409	2542215
4	596269	524567	459491	1167868
5	626735	756064	761129	826203
6	6936118	8174316	8087157	9612530
7	956003	1571037	821416	766716
8	4531556	4999253	4471983	7338516
9	8892334	8592738	8365994	11424068

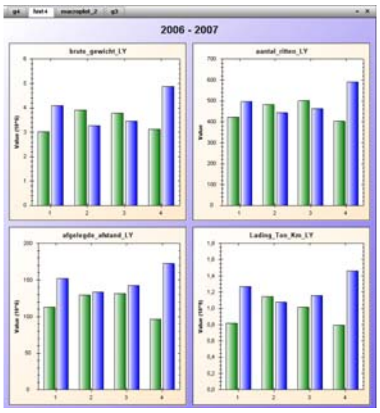
This pivot table shows `SUM(distance)` as a function of the two aggregation variables `Nstrcode_char1` (0…9) and `Quarter` (1…4). The color is a based on the relative difference with the corresponding aggregated value from last year, i.e. 0% is mapped on green and 100% (or more) is mapped onto red.

3. Plots: currently there are two kinds of *interactive* plots
   a. Scatterplots (shown in (a)): the colour of the plotted points may be coloured based on a third variable and a tool tip text may be defined that shows additional info when hovering over a point.
   b. Barplots (shown in (b)) : in this specific plot 4 variables are show per quarter, with the green bar being the previous value and the blue bar corresponding to the current value.



```
(ID) = 224892
(x,y) = (1250,00 ; 4,0)

(colorindex) = 1  #ophfact=169,470
```

(a)



(b)

Specifying other parts

Editing macro-data

Essential to macro-diting is a close integration with a microdata editor. In this prototype the Blaise editor (DEP) is used. To edit a user-selected record it is converted from database table(s) into a Blaise record; subsquently, the Blaise editor is started. After the editing of the record has finished the Blaise record is converted into the database table(s). The altered fields with old+new values are passed back to Macroview so that it can adjust existing aggregates and visualizations in an efficient way.

Using other external sofware

One of the techniques for macrodetection is the use of outlier-detection. This may require many numerical algorithms which have already been implemented in a number of (open-source) packages. The same holds for the calculation of weights (Bascula, among others). As an example MacroView can specify an interaction with the package R (see references); this is a widely used open-source package allowing matrix-based numerical and statistical calculations. Many additional R packages have been written by the academic community, based on the R scripting language.

Specifying interaction

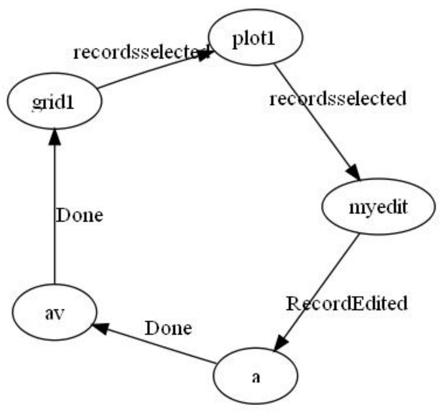
Now that the elements have been specified, the interaction during the analysis can be described:

```
Process

      a(PrevData, PrevDataMacro)
      a.Done -> av(CurrData, PrevDataMacro, CurrDataMacro)
      av.Done -> grid1(Agg.OutputData) AT Test Position Bottom

      grid1.recordsselected -> plot1(grid1.outputdata) at Test Position Top
      plot1.recordsselected -> MyEdit(plot1.outputdata)
      MyEdit.RecordEdited -> av(CurrData, PrevDataMacro, CurrDataMacro)

Endprocess
```

For clarity, the process has been depicted below. The arrows correspond to so called events. In this example there are two events:

- Done: the aggregate has been calculated
- RecordsSelected: the user selected either a cell from the table or a point from the scatterplot.

Note that the process forms a loop as shown below: the effects of the edit cause recalculation of aggregates and result in altered tables and plots being displayed.



Discussion

This paper presents a first prototype attempting to generalize the concept of macro-analysis. Currently only a limited number of visualizations have been implemented. Also, the prototype uses a text-based user-interface, as shown in several sections in this paper; this is meant for prototyping purposes and more advanced users. For less experienced or incidental users a graphical user-interface will be added to aid in the specification of the setup for the macro-editing session. Another addition might be a so called wizard dialog to help the specification of a new setup.

One of the requirements for the macro-edit tool is its interactive character: any of the steps, especially the edit step, shouldn't take too long, i.e. no longer than 2 or 3 seconds under normal circumstances.

However, this may not always be possible: the plausibility functions might be complex or an external program to re-apply imputations or weighting might take some time.

At the moment we're testing the prototype both on economical data (production statistics) and road transport. Trying to serve both statistics will help in generalizing the concepts needed for macro-editing. This may result in more flexible ways to specify plausibility functions, other visualizations or more interactive options when analyzing the data. This iterative process will have the main focus in the near future.

References

Granquist, L. (1994). Macro editing: A Review of some Methods for Rationalizing the Editing of Survey Data. Statistical Data Editing, Volume No. 1: Methods and Techniques.

De Waal, T. and Haziza, D. (2008), Statistical editing and imputation. Handbook of Statistics, Volume 29, Sample Surveys: Theory Methods and Inference, Editors: C.R. Rao and D. Pfeffermann.

The R project for statistical computing: see http://www.r-project.org