Laboratory of Data Analysis
University of Jyväskylä

EUREDIT - WP4.5, WP5.5 Internal reports

D4.5.2,D5.5.2 - Description of Software for Error Localication and Imputation based on The Tree-Structured Self-Organizing Map (Version 2)

Ismo Horppu - University of Jyväskylä
Pasi P. Koikkalainen - University of Jyväskylä
(Draft version 13. March 2002)

Contents

Summary

This report is a summary of software for Data Editing and Imputation, developed in the university of Jyväskylä (JyU) by the research group on Software Engineering and Computational Intelligence (SE&CI). Software has been build on the top of NDA (Neural Data Analysis) software platform that we have been building over ten years in various projects, both academic and industrial.

In EurEdit project new methodology for data editing and imputation has been implemented in the NDA kernel, as well as a new user interface, specially made for editing and imputing, has been build.

This report describes a second version of the software. First version was distributed to ONS and StatFi in last December (2001). The second version is based on the feedback (mainly from StatFi) and on our own ideas howto improve the system.

More information about the methodology is given in the related documents D5.5.1 (Imputation methodology), D4.5.1 (Editing methodology), and related appendixes.

Background information about the NDA software platform can be found from the PhD thesis of Erkki Hkkinen (2001): Design, Implementation and Evaluation of the Neural Data Analysis Environment, Studies in Computing, University of Jyvskyl. This monograph includes also one chapter about imputation methodology.

The NDA software package and its extensions for imputation and editing are available for all EurEdit members without any cost. For more information and howto get the software, send mail to (nda@mit.jyu.fi).

1 Introduction

Many real world problems fail to satisfy the assumptions often made by classical statistical methods. The reasons are

- Data size or dimension is too large
- Model or factors are unknown, iid assumption does not hold.
- Spotty data, missing data, outliers, errors, etc.

We think that the use of modern statistical methods for pattern recognition and data analysis is the best way to handle such problems. Since the methods are strongly based on computer power, information visualization and human support, we have build a software prototype, called NDA (Neural Data Analysis)(**?**) to support this work.

The original goal for the development of NDA was to distribute neural network technology among enterprises. For that reason, we have experience of a wide range of applications based on NDA, varying from customer analysis to computer vision and material research.

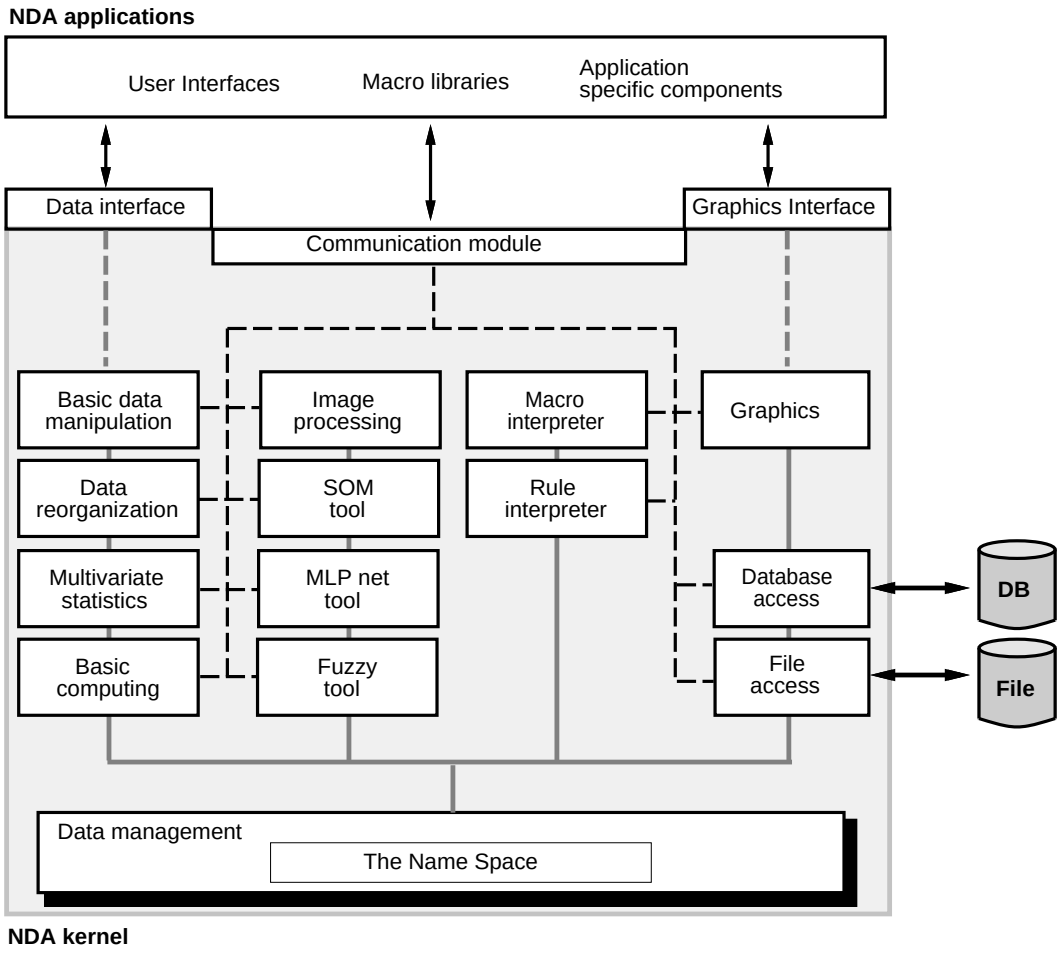
1.1 NDA basics

A fundamental principle of NDA software has been build in its modular architecture and the idea of reusing components. Our software architecture, which is depicted in Fig. **??** is one realization of this idea. It contains two type of mechanisms of reuse.

i) Layer architecture separates the kernel and algorithms from applications. Many different types of applications can be build on the level of customized user interfaces, that share the same kernel.

ii) New algorithmic modules can be added, as well as new algorithms, to the kernel. They communicate and share information with old algorithms via named data structures.

Chart 1: Software architecture of the Neural Data Analysis Environment.



When the existing components of the kernel are sufficient for an application, then development can be done using NDA macro language only. Then a new application is a collection of macros that are executed by the user via command interface or a case specific user interface.

In the this presentation a new user interface has been build to support the tasks of data editing and imputation. Most of the functionality is based on old kernel algorithms, but also new variations of tree-structured map or editing and imputation have been implemented.

1.2 Building an editing and imputation environment with NDA

The software for data editing and imputation is our attempt to cover all aspects of typical *data production process* (DPP), as we think it is done in official statistics and industrial data management.

Our first idea of DPP was a chain of predefined operations from data imput to display of results. This was a failture, since any predefined way of doing data edits and imputations will most certainly fail, regardless of how good the underlying algorithms are. Therefore a more flexible environment has been build.

In the second version of our software we consider the following tasks as fundamental requirements of what our software should be able to do.

a) Data manipulation, reorganization and visualization are common tasks in data analysis. These are already implemented in the NDA, but some effort has been done, and still need to be done, to make these operations user friendly.

b) Use of external knowledge, such as edit rules, must be supported. We have done this with a simple rule converter that translates rules of Euredit data sets to out NDA type of expressions.

c) Variable selections and case spesific edit/imputation operations should allowed. The user should be able do them with minimal effort and see easily all the choices and selections he has done.

d) Experimenting and playing with data should be easy.

e) There should be support for the visualization of results, summaries etc.

The current realization of these goals is briefly outlined in the following sections. The reader should note that this is still work in progress.

2 Main functionality of the edit and imputation software.

The data production process (DPP) application software can be used to edit and impute erroneous and incomplete data sets. As explained in the previous sections, most of the functionality if is implemented with the Neural Data Analysis (NDA) software package, which is computer architecture and operating system independent code. The graphical user interface of the application is built with Borland Delphi/Kylix combiation, which is, making the program available for both Windows and Linux operating systems. Fig. 2 shows the main screen of the application.

The user interface collects information from the user and gives feedback for him/her. Then it produces a macro file of the DPP. The idea is that the whole DPP can be executed as such a NDA macro file, where a separate parameter file is used to store needed parameters for the process. This allows us to re-execute operations easily, when doing, for example resampling of data for model validation.

Different phases, such as logical edits, the coding of variables, data transforms, data reductions, imputation procedures and validation of results, are implemented as sub-macros, which allows easy customization of them. This makes the application more flexible than traditional, predined systems.

In the main view of Fig 2 are visualized tools that one can use to configure general setting of the system, select how to do edits, variable transforms, dummy variable codings, data modelling (compression), imputation of incompleta variables. General settings include, for example the selection of data sets and generic parameters.

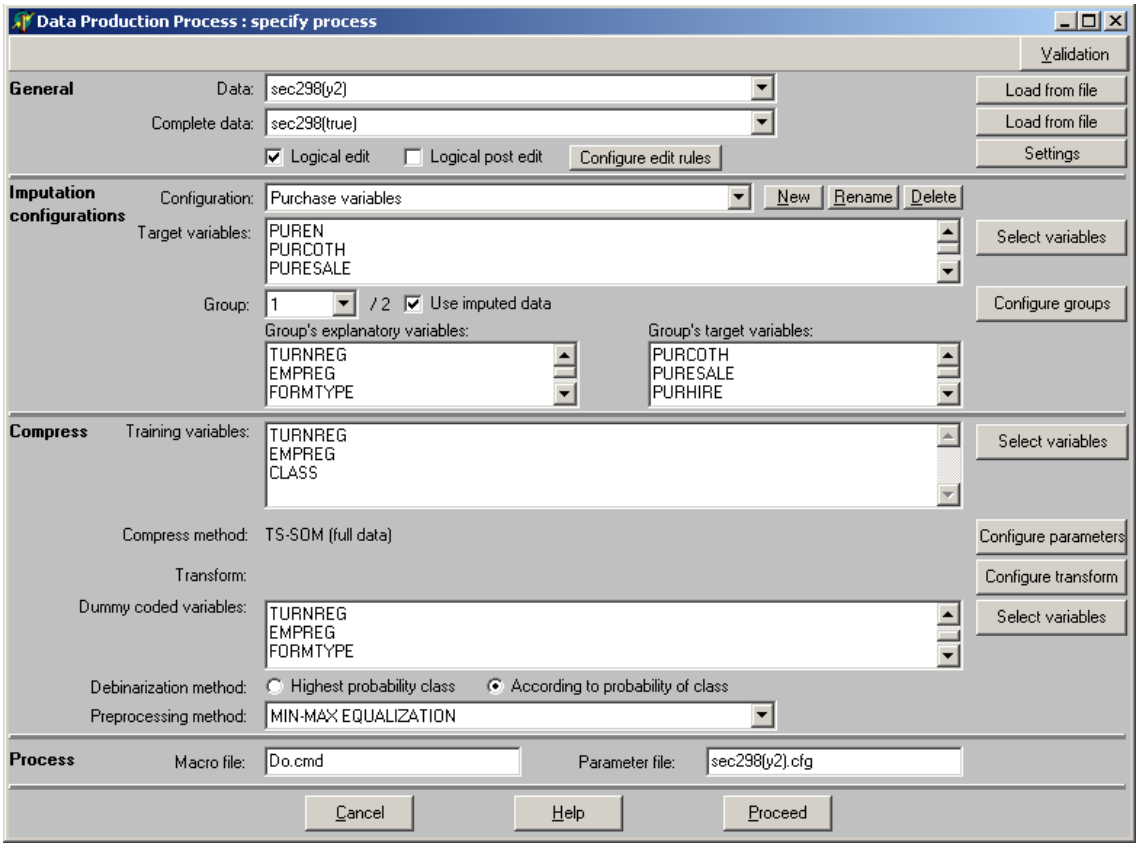
Some other settings are configured in the settings screen that is shown in Fig. 3.

These include, the special values for missing data and other uses, bootstrapping parameters of the whole DP process, and listwise deletion spsifications of incomplete data sets. Also, one can select to do both pre and post edits, before and after edits and imputations process.

In the transform selections, user can configure the way how variable(s) are preprocessed. Variables can be, for example, dummy coded, which is typically needed for categorial types. If dummy coding is used the user may also decide the decoding method, which can be either the selection of highest probability class, or random sampling according to the posterior probability of classes.

Data reduction (compression) method is the core of our imputation system. It is a multivariate model of the data, on top of which several imputation systems can be build. The model requires from user variable selections and knowledge of how these shuold be preprosessed. As usual different variables can be used for model building, which mainly conditionalises the data and for imputation that requires prediction of missing variables.

Chart 2: Main view of the application



Different collections of models, preprosessing systems, imputation routines can be stored separately for later use and combined in different order, if needed. Typically one uses the same data model for different imputations (of dirrerent type of variables).

3 Tools for data analysis

Information visualization is fundamentally important for multivariate data analysis. It is also one of the most neglected areas in science, most likely because it is difficult quantify objectively.

With powerful visualization capabilities of the NDA one can display data and its relations in a way that no automated procedure can.

Simple methods for data visualization are, for example scatterplots as depicted in Figs. 4.

For more complex situations self-organizing feature maps can be used to inspect the data, as depicted in Fig. 5. One can, for example to identify groups or correlations between different variables.

The SOM is also useful for viewing error and missingness blocks, as shown in Figs. 6.

Summarization of variable categories is often needed to reduce the number of data records. Also this can be done by using SOM or other similar tools of the NDA. We may search most important variable combinations which are selected and then use our spcial techniques to present whole categories are new varaibles.

Chart 3: An example of a typical parameter setting sub-window.

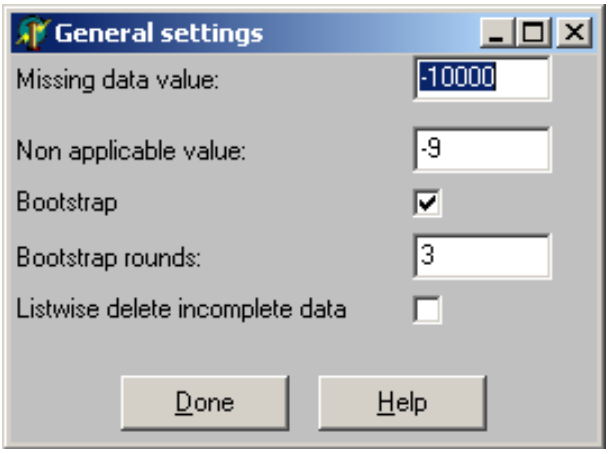
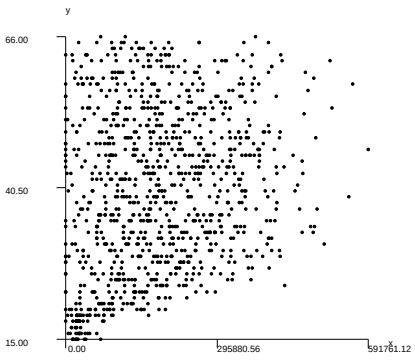
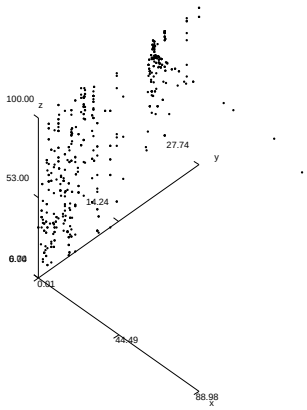


Chart 4: Scatterplots





4 Implementation of edit rules

Logical edits can be used to detect logical errors in the data. For example, the age of child may not be higher than parent.

The logical error detection rules are expressed by the expression language of the NDA, which is visualized in Fig. 7. The compilation of edits from other formats is usually quite simple. For example all edit rules of UK-ABI and SARS data sets have been converted to our system.

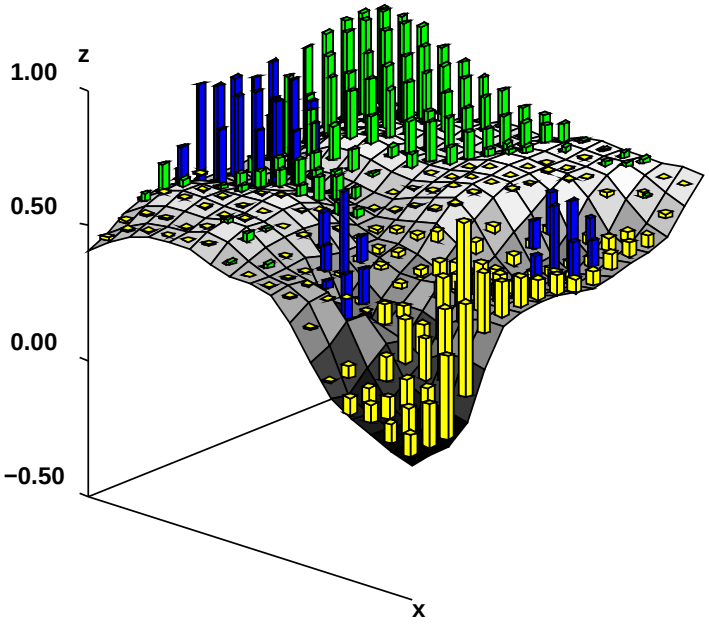
One edit includes four parts which are: `CONDITION`, `CHECK`, `EDIT` and `IMPUTE`. Value of the `CHECK` must be true before edit can happen.

`CONDITION` is evaluated over all records for which `CHECK` is true.

`CHECK` can be used to check rule validity for example with ratio edits. It can also be used for example to put separate short and long form edits to same rules (in such case one/some variable(s) must be used to condition the edit).

`EDIT` is used to specify implicit edit values for the erroneous variable(s) of record. If there are multiple

Chart 5: Joint distribution of 5 variables



edit values for one variable then one of them is randomly chosed.

`IMPUTE` is used to mark erroneous variable(s) of record for imputation.

5 About the coding of variables

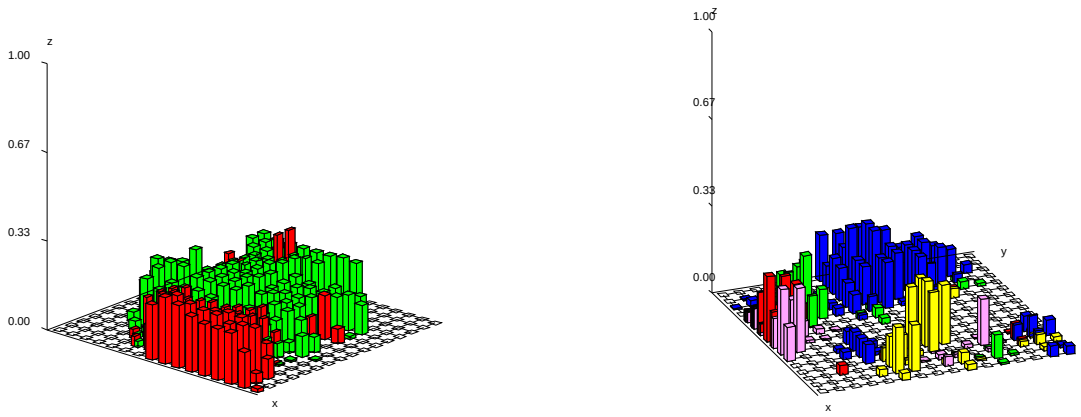
There are several ways to code variables for data analysis. Since most neural network methods expect data to be in vector from, where the elements are floats, some codings need to be used for ordinal and categorial variables.

Discrete variables are often needed to be dummy coded, which means that for each value of the class is created a new dummy (false/true) indicator.

The coding is needed for each discrete category, when using neural networks for data modelling. This is then used as natural measures for record distances etc.

In our software 8, in addition to displaying names of variables, also the description (when available) of variable type, amount of categories, and incompleteness is show when working with variables. This helps the user during the DPP, when making decisions about variable codings.

Chart 6: Joint distribution of missingness of 5 variables (left) and errors of 2 variables (right). Note: value 1 means high amount of (missingness/errors and value 0 means no missingness/errors)



6 Data transformations

All kinds of transformations are needed during the DPP. In the neural networks these are known as preprocessing operations like normalization and standardization. Other possibilities are simple changes of scales, for example.

Data transformations are specified using the expression language of the NDA, as shown in Fig. 9. Data is transformed in three phases which are:

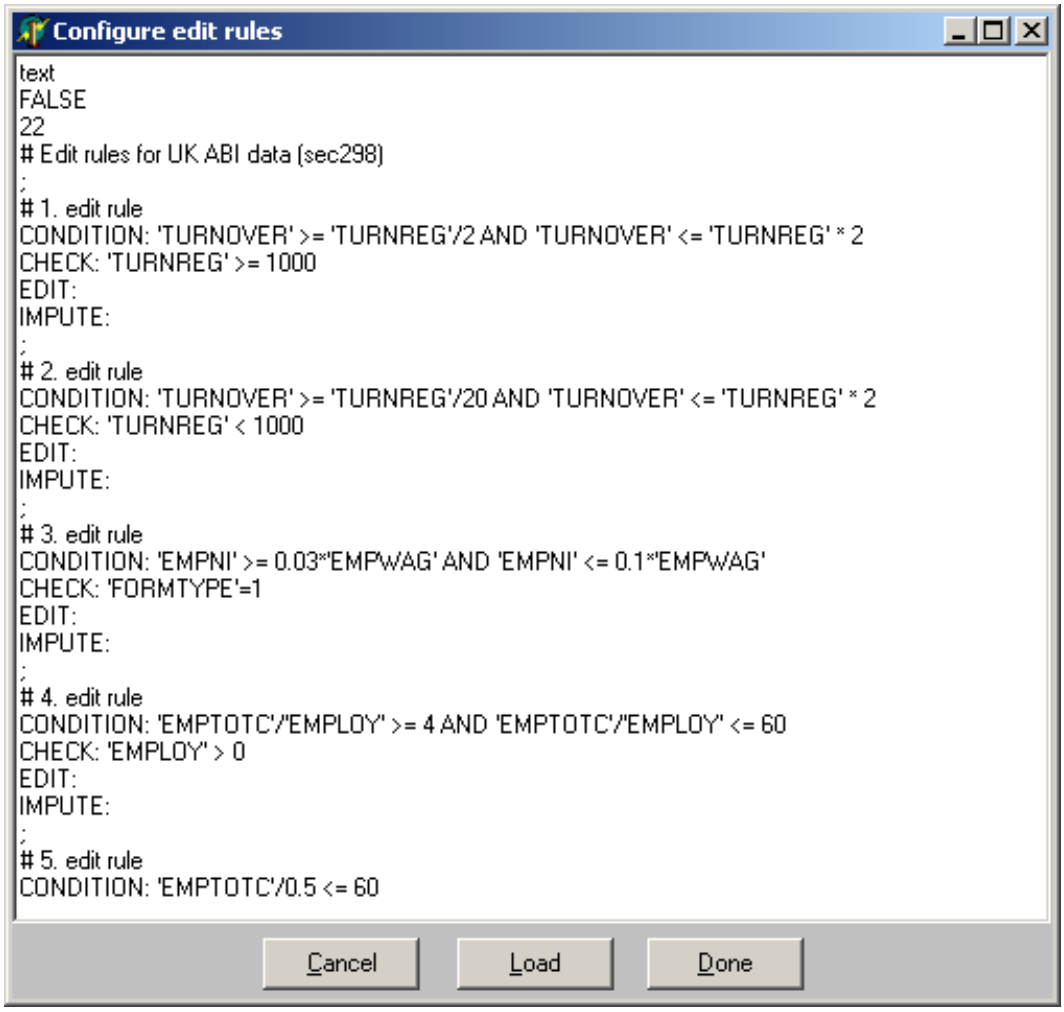
(i) before coding of variables,

(ii) before data reduction, and

(iii) before imputation.

In phase (i) one can for example to reduce amount of classes of discrete variable. For example if there is a hierarchical classification variable whose digits present hierarchy then one can cut the variable to integer (if that is proper). In phases (ii) and (iii) one may for example take logarithm of INCOME variable to make the scale linear (assuming INCOME is exponential).

Chart 7: Configuring edit rules



7 Modelling of data

The modelling phase of the imputation system tries to compress or predict data as well as possible, by dividing it into model part and random part. We typically use self-organizing map for this, as explained in our methodology reports. Task includes, for example the selection of variables for modelling (Fig. 10.

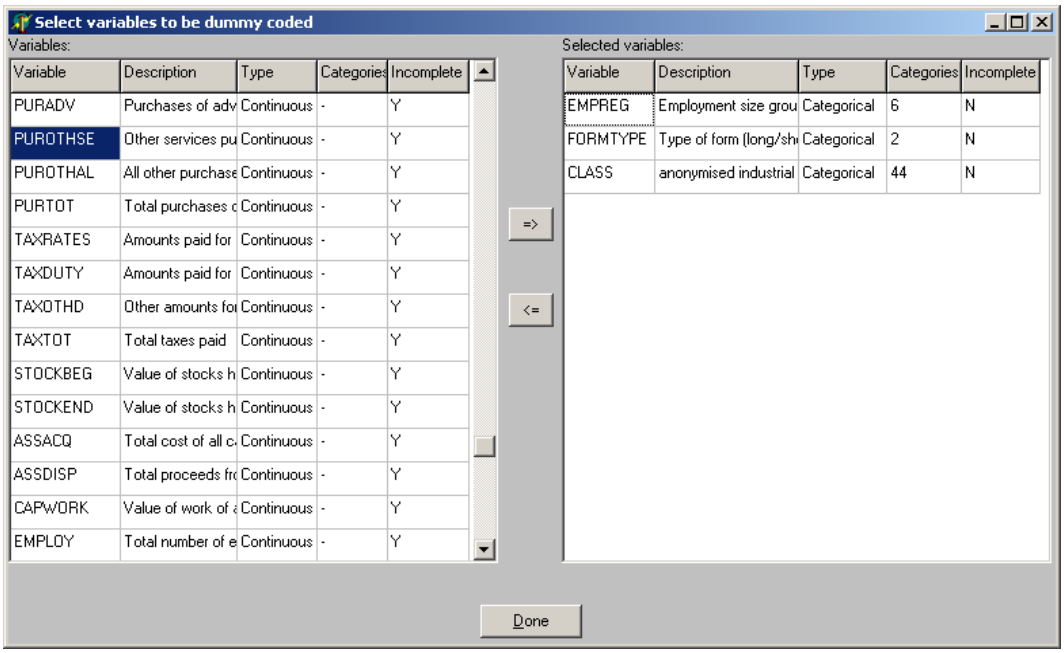
The model can be used with imputation phase in order to improve and/or speedup the process. In order to make variables comparable the data must be preprocessed using a suitable transformation.

The application supports the following predefined preprocessing methods: min-max equalization, variance based equalization, normalization and whitening.

Latter two do not support (currently) treatment of incomplete data thus with those methods model training information is lost (amount depends on missingness block(s)).

Many data modelling methods have been implemented in the NDA, and almost all of them are available in the DPP application. The methods include our own TS-SOM, K-Means, Fuzzy C-Means and hierarchical clustering methods. New EM-algorithm based versions of TS-SOM centroid and TS-

Chart 8: Selecting variables to be dummy coded



SOM neuron weight imputation are also available. These combine the data modelling and imputation under one procedure.

Yet another use of TS-SOM is that we have developed a version that supports the detection of errors during learning, allowing us to build robust model of data.

8 Imputation systems

The DPP sfotware supports many imputation methods for variables and variable groups.

In the configuration phase user defines the imputation model for selected incomplete variables. Configuration can be named and positioned as shown in Fig. 11.

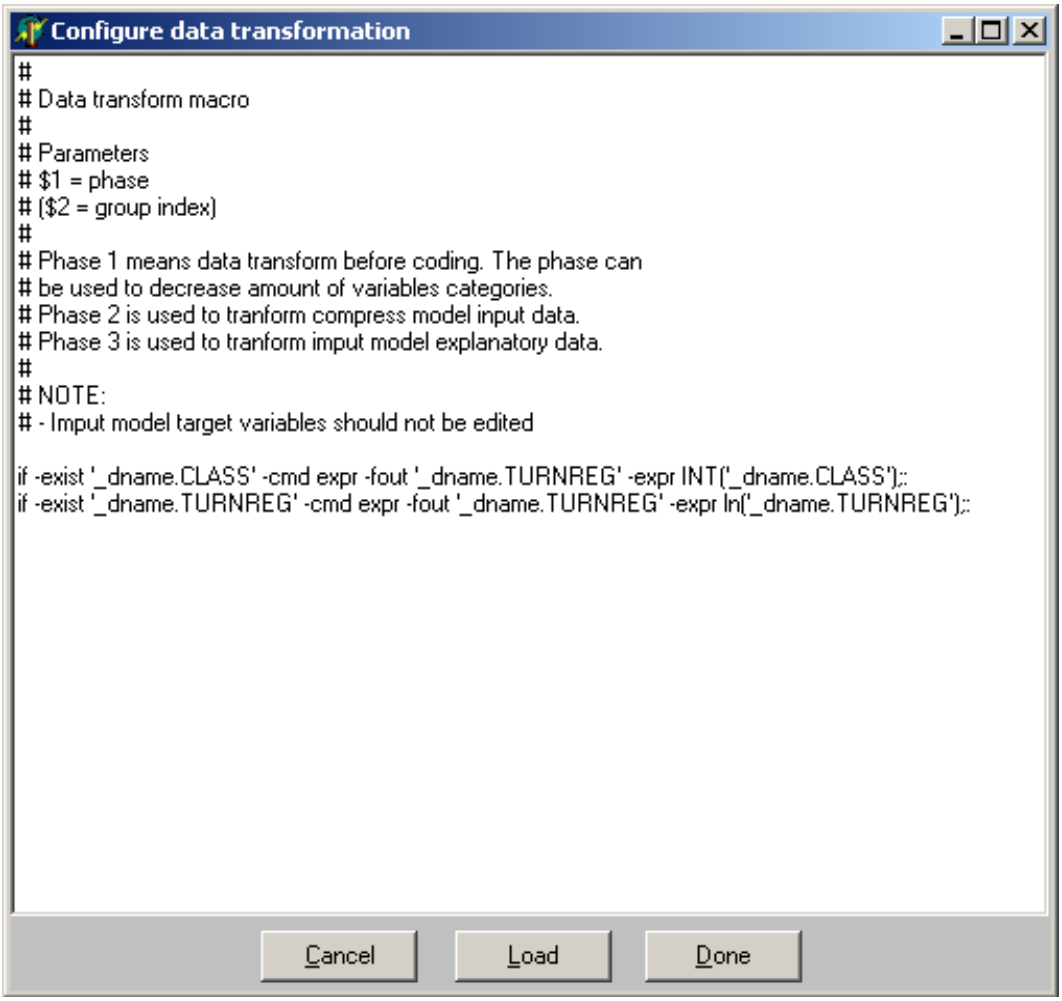
The order of configurations define imputation order of variables. Groups define how the imputation is done as shown in Fig. 12.

One can define explanatory and target variables for each imputation group. In addition to these imputation methods and their parameters, imputation count, which defines single or multiple imputation, and the usage of the distribution model can be specified.

If data model is not used, then imputation is done over all the data. Otherwise imputation is done within conditional selection of data, e.g. within nearest data clusters. In this contest an observation can be forsed to go only to those clusters that are representative enough. This is done with a global imputation parameter. The criterion guides incomplete records to clusters that are both close to the observed part of sample and meet an additional criterion, e.g. prior probability of cluster is large enough.

Our software supports currently the following imputation methods: mean, nearest neighbour, prediction by normal distribution, random donor, robust mean, stochastic mean, stochastic robust mean, MLP regression and MLP multivariate regression.

Chart 9: Configuring data transformation



9 Validation

If complete (true) data is available then individual level values, like mean error (ME), mean square error (MSE) and root mean square error (RMSE) are calculated. On agregate level, values like mean and deviation are also calculated. Currently on part of Ray's criteria are implemented, but more will be added in near future.

Software allows customization of validation which means that user can define what values are calculated, as shown in Fig. 13. The application calculates absolute and percentage difference between agregate level values of imputed and comparison data.

If complete data is not available then the data is compared to original distribution. If bootstrapping of process is selected then all values are bootstrapped. Thus average and deviation of each parameter is calculated. If logical edit rules were available then post edit can be done. It can be used to determine whether process increases or decreases logical errors.

With true values one can also visualize difference of imputed and true values, for example by SOM surface.

.

Chart 10: Selecting training variables

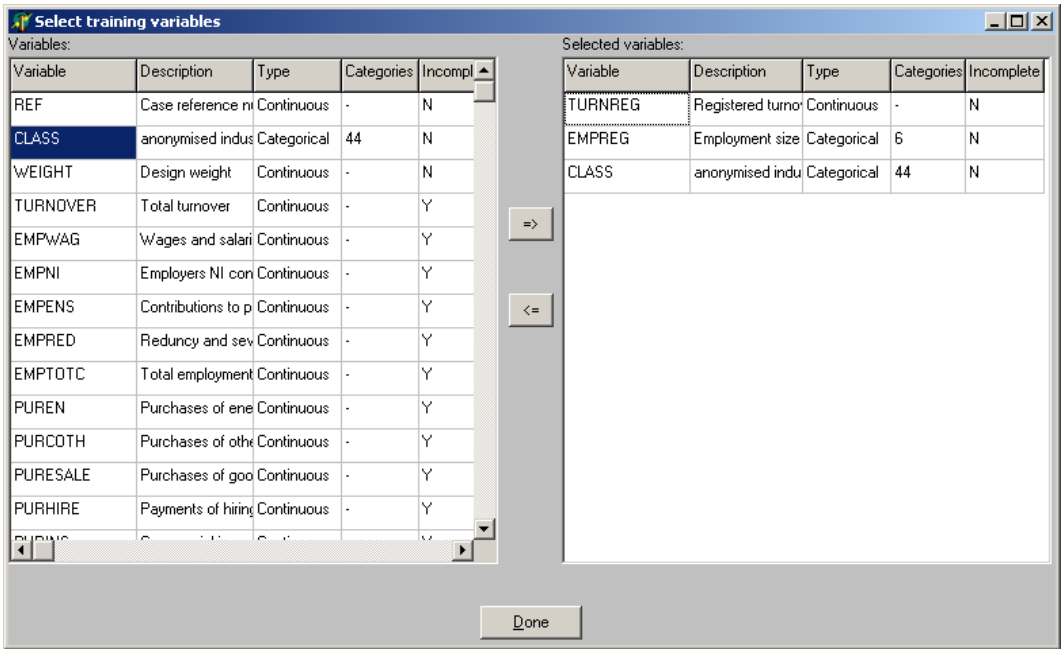
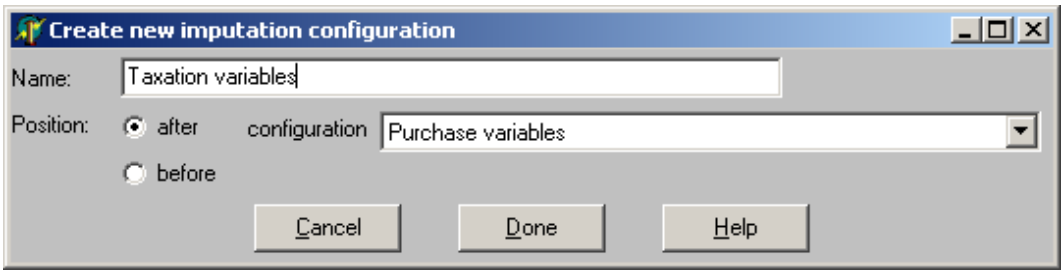


Chart 11: Creating new imputation configuration



10 Process time analysis and bootstrapping

Process tools include also statistics of time consumption and the amount of found logical errors.

With bootstrapping one gets confidence intervals for these parameters. Time consumption can be used to detect how chosen algorithms scale with size of data. Amount of logical errors can be used to detect whether process corrects or creates logical errors. Time measurement accuracy is one second.

11 Final notes

This is very early draft, please understand....

Chart 12: Configuring imputation groups

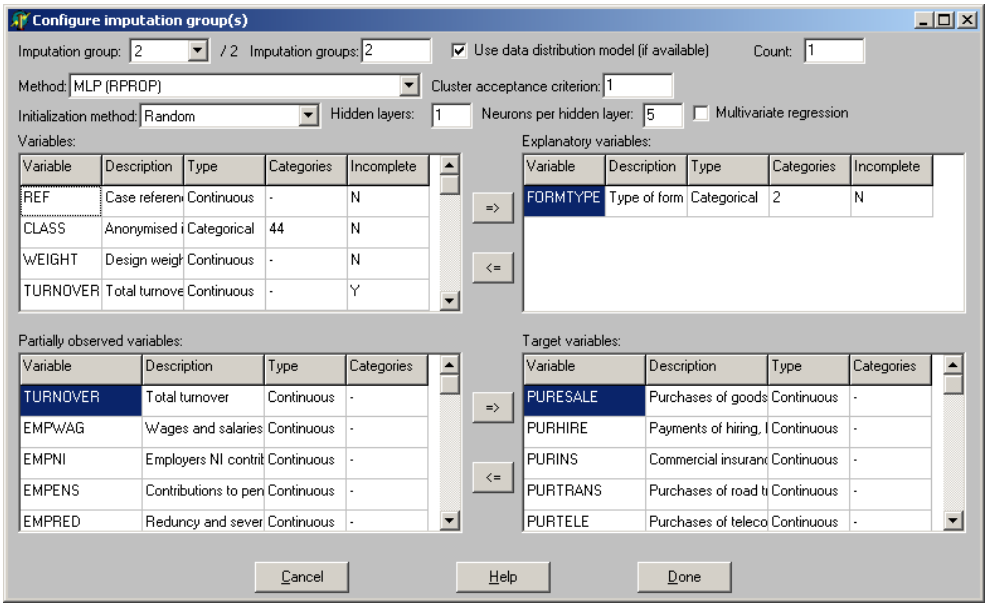


Chart 13: Validation screen

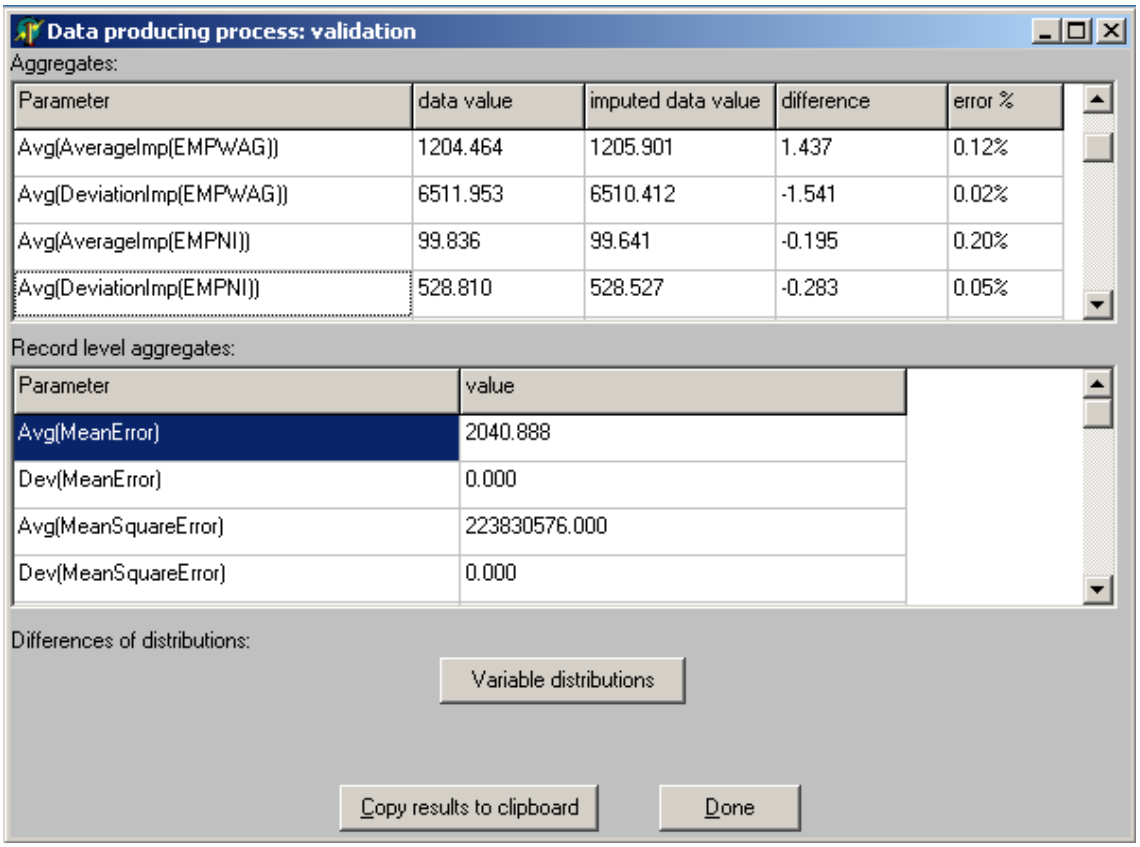


Chart 14: Fig. 13, Fig. 14: Time consumption