STATISTICAL COMMISSION and
ECONOMIC COMMISSION FOR EUROPE

Working paper No. 29
English only

CONFERENCE OF EUROPEAN STATISTICIANS

UNECE Work Session on Statistical Data Editing
(27 – 29 May 2002, Helsinki, Finland)

Topic (iv): Impact of new technologies on statistical data editing

TREE-STRUCTURED SELF-ORGANIZING MAPS FOR IMPUTATION

Contributed paper

Submitted by Statistics Finland[1]

Abstract: In this presentation I will discuss one modern approach to imputation. Many traditional methods of imputation use some kind of classification trying to get observations with missing values into as homogenous groups as possible. Self-organizing maps (SOM) is an iterative method for classification and can thus also be used in finding the imputation classes. Imputations are made within clusters, located by corresponding neurons, in several ways that can be based on both traditional and neural methods. Respectively for editing problems, SOM modelling helps in error localization. It can be trained to observe different abnormalities, and in the ideal situation the erroneous observations are, for example, distinguished in their own clusters.
Tree-Structured SOM (TS-SOM) is a modification that reduces computational complexity of the basic SOM, for example. TS-SOM methods are included in the versatile program named NDA, Neural Data Analysis, which was made by the research group on Software Engineering and Computational Intelligence of the University of Jyväskylä, Finland. Imputation methods have been implemented into NDA in co-operation with the research group of Statistics Finland. This presentation is based on research work in the Euredit FP5 project, and the methods are still under heavy development.

Keywords: Kohonen algorithm, tree-structured self-organizing maps, neural data analysis, regression and hot-deck imputation, editing.

I. INTRODUCTION

1. The research group on Software Engineering and Computational Intelligence (SECI) of the University of Jyväskylä (JyU), Finland, developed a software called the Neural Data Analysis environment (NDA) with *Pasi Koikkalainen* as the group leader, and the NDA is naturally still under heavy development. The software provides a generic application platform for computational intelligence with many proven examples of real world applications. The main emphasis of the software is to aid methodological development of knowledge discovery, data analysis and modelling in general. Techniques are mainly based on neural networks, but the system also includes a large data set of data manipulation and visualization techniques, fuzzy sets and so on. However, the most important method used in the NDA is the Tree-Structured Self-Organizing Map [6], usually abbreviated as TS-SOM. TS-SOM is a modification of the Self-Organizing Maps (SOM) [3].

2. During the development of the NDA it was noticed that one of the main problems with the use of neural networks in practice is incomplete data with missing and erroneous units. Modern neural oriented methods for error localization and imputation are under research and development especially in the EU/FP5 project EUREDIT[2] (*The Development and Evaluation of New Methods for Editing and Imputation*) in which JyU and Statistics Finland are working together to develop SOM based techniques for editing and imputation.

---

[1] Prepared by Pasi Piela (pasi.piela@stat.fi).
[2] For more information about the Euredit project: *http://www.cs.york.ac.uk/euredit/* .

This paper gives a rough introduction to the TS-SOM methodology for imputation. Practical examples with result tables, in which data sets are derived from the Euredit project, are also presented.

II. TREE-STRUCTURED SELF-ORGANIZING MAP MODELLING

3. The basic SOM defines a mapping from the input data space $\boldsymbol{R}^n$ onto a latent space typically consisted of a two-dimensional array of nodes or neurons [3]. A parametric reference or weight vector $\boldsymbol{w}_i$ is chosen for each neuron i from the discrete set $i = \{1, 2, \ldots, N\}$. Now, let $\mathbf{x} \in \boldsymbol{R}^n$ be a stochastic, random data vector. Usually the smallest of the Euclidean distances $|| \mathbf{x} - \mathbf{w}_i ||$ is made to define the best matching unit (BMU) for the vector $\mathbf{x}$, denoted by the subscript b, that is, $b = \text{argmin}\{|| \mathbf{x} - \mathbf{w}_i ||\}$. Then SOM algorithm updates the weight vectors of the BMU and *in its neighbouring neurons i* as follows:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + h_{bi}(t)\left[\mathbf{x}(t) - \mathbf{w}_i(t)\right]$$

where $h_{bi}(t) = \alpha(t) H\left(\frac{\mathbf{v}_b - \mathbf{v}_i}{\sigma}\right)$ defines the neighbourhood kernel $H(\cdot)$ over the lattice points and the learning rate $(0 < \alpha(t) < 1)$ at the iteration step t. One common, Gaussian, kernel function is $H(v) = exp(-|| \mathbf{v} ||)$. Thus, each iteration starts with a new random sample. The idea is to start with a large neigbourhood and reduce it along with the value of the learning rate parameter. *Specifically, the SOM algorithm can be interpreted as a discretized approximation procedure for computation of principal curves or surfaces* [8].

4. The map is usually created in a non-stochastic way by using the so-called *batch algorithm* in which all the data points are associated with their BMUs in each iteration, and then all the prototypes are updated at a time.

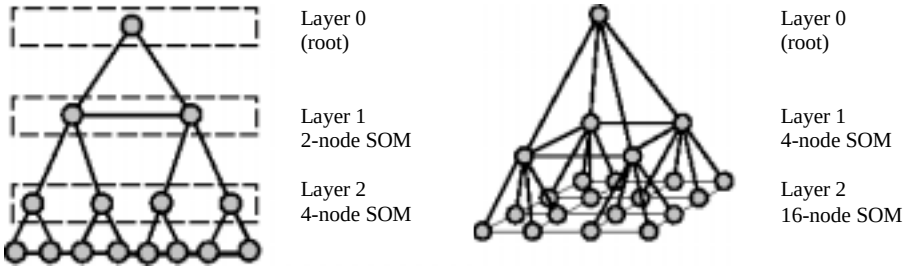
5. The Tree-Structured Self-Organizing Map is made of several SOMs arranged to a tree structure (see Figure 1). The topmost layer ($L = 0$) has one neuron. Layer 1 has four neurons in two-dimensional and two neurons in one-dimensional case. Thus, each neuron has its own associated subgroup of data, four subgroups on layer 1 but one group, the data set itself on layer 0. The subgroup forms the cluster of which centroid is the weight vector of the best matching unit b, $\mathbf{w}_b$. Furthermore, the centroid on layer 0 defines the mean of all data.

6. The training is repeated layer by layer using knowledge about the BMU of the frozen layer l–1 in the search of the BMU on the next layer l. That is, the search of the BMU for the layer l is restricted into a small set of neurons: sons and sons of neighbours of the BMU of the previous layer. This clearly reduces the *computational complexity* when compared to the basic SOM. We will discuss this later.

7. The training is usually made with *the batch algorithm* [4]. During each epoch the BMUs are searched for all data vectors using the tree search, and then new centroids $\mathbf{m}_i(t)$ are the weight vectors $\mathbf{w}_i(t)$ computed using the rule:

$$\mathbf{w}_b(t+1) = \frac{1}{N_b + \sum_{i \in N_c(b)} \alpha N_i} \left(N_b m_b(t) + \sum_{i \in N_c(b)} \alpha N_i \mathbf{m}_i(t) \right)$$

Figure 1. Illustrations of one and two-dimensional TS-SOM structures.



8. Where $N_c(b)$ is a set of indices of neighbours of b, and N_i is the number of data records in the Voronoi region (cluster) i. The smoothing is partially controlled through the parameter $\alpha \in [0..1]$. One side advantage here is that the size of the neighbourhood can be kept constant, and the usual problem with the basic SOM, namely, the relation between the neighbourhood size and the learning rate parameter during each epoch, does not exist.

III. TS-SOM BASED IMPUTATION

9. A natural approach to the missingness problem is now imputation within the clusters located by associated neurons. A simple starting point is to derive analogous processes from the classical imputation methods. Nearest neighbour imputation can be made by filling missing components of the data vector from the nearest data vector within the same cluster. Group means imputation (replacing the missing value by the average value of the observations belonging to the same class/subgroup) can be made by taking the centroid of the cluster, $\boldsymbol{m}_b$, and by replacing the missing component j of the data vector $\mathbf{x}_i$ by corresponding $\boldsymbol{m}_b(j)$, which is actually the fastest way to impute.

10. Besides previous simple imputation models, it is obvious that more complex, regression based, imputation modelling should be taken under consideration when trying to take advantage of TS-SOM mapping, and thus take values for missing components from these models which are generalizations of the local distributions of each data cluster. The MLP network with the backpropagation algorithm is often used to model the relationship between complete and imputation variables (MLP theory and methodology, see [1]). From the SOM mapping point of view, the MLP networks can be trained separately in each cluster of the final TS-SOM layer. Thus, we have as many *local* MLP models as we have neurons on the last TS-SOM layer.

11. However, MLP training as presented above only takes into account the resulting condition of the TS-SOM mapping. But Koikkalainen [5] has shown that TS-SOM can be seen as a hierarchical Gaussian mixture model where each neuron is a Gaussian generator. It is thus possible to use this idea in creating an imputation model by observing the posterior probabilities of the neurons. Häkkinen [2] presents imputation methods based on this idea, calling them iterative methods with probabilistic imputation. We will not discuss these methods here any further, just state that new methods applying the idea of probabilistic distributions of building TS-SOM into imputation are being tested and developed.

IV. REDUCING COMPUTATIONAL COMPLEXITY: TS-SOM BASED IMPUTATION IN THE DANISH LABOUR FORCE SURVEY DATA SET

12. One of the data sets for the evaluation and development of imputation methods in the project Euredit is the Danish Labour Force Survey (1996), consisting of Danish population register records for individuals selected for interview. The very carefully created synthetic version of this data set was given as training and development data for imputation methods. It is structurally very simple and will be used here. The data consist of only 14 variables with little information, four relating to type of response. Annual income (DKK) is the only variable needed to impute, the missingness rate being 26.8%. The data are at person level having 200,000 observations, information about households is not available here.

These 13 auxiliary variables are categorical, except person's age. They have 2 to 4 classes, except AREA (area of living), BUSINESS (last employment) and EDUCATION, for example, have 4 classes:

1 = Private Industry	1 = Primary school only
2 = Other private business	2 = Craftsman, Skilled labour, High school only
3 = Government employed	3 = Long education, school teacher, university, etc.
-9 = Not applicable	-9 = No information

13. The complete data vectors are used here as training data. That is, we select all 146,323 observations with a known value of income. Another alternative would be selecting all the observations and training TS-SOM for these without the variable INCOME, but this is not necessarily appropriate in this simple case due to losing information on the relationship between INCOME and explanatory variables.

14. First we make data analysis by building TS-SOM for our training data set by NDA software and viewing its different layers. For example, it is easy to observe graphically the basic statistics of INCOME for different clusters and compare them to the statistics of background variables.

15. Variables are scaled for nearest neighbour imputation (this thus assumes that each variable has the same weight), where for the missing component $\mathbf{x}_i(j)$ that belongs to a cluster b

$$\mathbf{x}_i(j) = \mathbf{x}_n(j), \quad n = \text{argmin}_{k \in cl_b} (\| \mathbf{x}_i - \mathbf{x}_k \|), \tag{4.1}$$

so that all categorical variables are binarized. Thus, for the variable l of m classes we have corresponding m (0/1) variables. The continuous background variable, AGE, is scaled to [0, 1] based on min/max ranges:

$$\mathbf{x}' = \frac{\mathbf{x} - min(\mathbf{x})}{max(\mathbf{x}) - min(\mathbf{x})}.$$

16. However, in the case of several skewed distributed continuous variables there are naturally other appropriate methods for equalization. Also, when missingness of several components occurs the distances in (4.1) can be weighted by simply comparing the number of missing components and the number of variables for every data vector.

17. An obvious problem in this kind of hot-deck type of imputation is its *computational complexity* (related to computation time) that is $O(N^2)$ due to the full search among N data vectors. By using the above method and TS-SOM the complexity can be reduced to $O(N\log_p M + N^2/M)$ where the complexity of the TS-SOM algorithm is $O(N\log_p M)$ [6], p being the number of sons of each node (in two dimensional case: $p = 4$), and for the imputation within M clusters it is $O(N^2/M)$. In practice, this was clearly observed for the example in question as follows:

Nearest Neighbour Imputation	Computation Time, Pentium® II Processor (500)
without TS-SOM mapping (layer = 0)	> 12 hours
layer 2 as imputation layer (16 neurons)	59 minutes
layer 3 as imputation layer (64 neurons)	20 minutes

Table 1. Computation time of the nearest neighbour imputation in practice. Synthetic Danish Labour Force Data set, N = 200,000.

18. As shown in Figure 2, the imputation does not give good results at the unit level, which was expected because of lack of background information. But for this case the method really seems to work at the aggregate (or data) levels. Specifically, the line for observations with $\hat{Y} = 0$ and $Y^* > 0$ is quite similar to the line for observations with $\hat{Y} > 0$ and $Y^* = 0$, but the lines are long including several wrong imputed

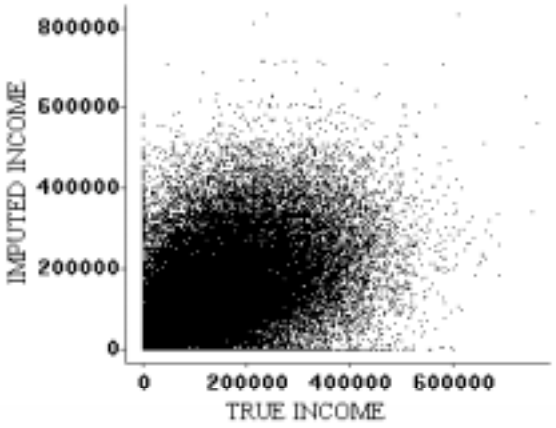
values. Furthermore, the linear regression model (y = true income, x = imputed income) has a surprisingly high estimate of the intercept parameter (102034) but when modelling without intercept quite a good model with high R-square is observed.

19. As Häkkinen [2] points out, there are some obvious problems in this kind of traditional method together with TS-SOM. There might be clusters having only missing values and there is a risk of losing the nearest data vector to another cluster, which might be a problem here as well (see Table 2).

20. Cluster centroid imputation, which replaces the missing values with corresponding values of the centroid of the same cluster, is not appropriate in this case. Totals are clearly over-estimated and variances under-estimated. On the other hand, the structure of TS-SOM gives possibilities to solve problematic situations: the centroid can be interpolated from a parent neuron of the upper TS-SOM layer, or it is possible to use neighbour clusters and neurons as well in finding the appropriate donor or the centroid needed.

21. There are, naturally, a number of ways to impute by MLP models. Table 2 presents results for two structurally different groups of local backpropagation MLP imputation models for modelling dependencies between background and INCOME variables. In Method I, for example, we first build TS-SOM and on its first level we use the backpropagation algorithm to create four local MLP models with two hidden layers of four and six neurons connected sequentially by logistic sigmoid units, one for each neuron (see Table 2). But these kinds of MLP models are clearly problematic for the case in question; they do not give any 0s as estimates of INCOME, and variances are under-estimated.

Figure 2. Scatterplot of imputed INCOME ($\hat{Y}$) and *corresponding* true values (Y*) from nearest neighbour imputation on the 3rd TS-SOM layer.



V. GETTING BETTER IMPUTATION CLASSES: TS-SOM BASED IMPUTATION IN THE UK CENSUS SAMPLE DATA

22. In this second case we present results for an anonymized sample of UK Census data. There are both member level and household level data, and breadwinners of each household are chosen to impute one household level variable, namely ROOMSNUM (*number of rooms*). Member level imputation variables here are AGE and HOURS (hours worked weekly). There are naturally a number of other variables as well. Figure 3 gives an example of SOM visualization for tenure of household space. Note that mapping behind this figure has been made by using many variables of the data.

A. IMPUTATION OF AGE

23. The data for person's age consist of 47,703 units in the York and Humb area. Rate of missingness is 7.6%. Interval of AGE is 0–90, 91, 93 and 95.

24. Several tests have been made in trying to get as good results as possible both at aggregate level and at unit level. Specifically, the number of imputed 0s seems to be one satisfactory indicator of the goodness of imputation. We have not yet been able to explain completely why nearest neighbour imputation fails when compared to other methods; there are too many imputed 0s (see Table 3). 0 observations seem to be quite special in these data.

25. First, the best results came from the random imputation within the TS-SOM clusters at the 4^{th} level, which means SOM mapping of 256 clusters; there might exist a few empty clusters. Figure 4 presents a nice visualization of the 4^{th} level SOM map. Naturally, this is a very trivial example but it shows that SOM algorithm has made a good discretization for the data.

26. However, in the method above we have not taken into account households except using household level variables. That is why we chose a special nearest neighbour method within the household – when possible – by using relationship to household head, RELAT, as a sorting variable. In this special method, data are first divided into two parts (part 1: younger people, part 2: older people) and then if the nearest neighbour imputation within household is possible, in other words, if there is at least one known value of AGE for the missing one from the same household, we impute. This leaves 1,261 observations unimputed. Those will be imputed by TS-SOM 4^{th} level random imputation separately from these NN imputed values. This method gives the best results we have got *so far*, see the last row of Table 3.

27. Table 3, as well as other tables, also includes one trivial benchmarking method, namely, random overall imputation which measures the effect of randomness and the degree of difficulty in one easy way. Surprisingly, the estimates of mean and standard deviation are very close to the true one. But luckily, values for DL1 are far away.

28. Interestingly, Piela and Laaksonen [7] have stated that usually a medium-sized classification or regression tree with a small number of explanatory variables, 3 to 5, is the best one for imputation. In this case imputations were made randomly or nearest neighbour hot-decking within the terminal nodes of these classical trees.

B. IMPUTATION OF HOURS

29. Any special problems of replacing missing values of working hours have not been noticed. As seen in Table 4, imputation results are rather good and mean deviations quite small. It should be noted that the true deviation of the HOURs is high, which makes imputation more challenging – besides, it clearly separates average based methods from the others. The number of observations in Table 4 differs from Table 3 due to deleting so-called non-applicable observations.

30. Prediction from the normal distribution that is estimated for all the SOM clusters by calculation using variance and mean gives good results except that it fails to estimate deviations or percentiles. Random imputation within the clusters seems to be reasonable as well. But full random imputation without any other helping method gives good estimates, too.

C. IMPUTATION OF ROOMSNUM

31. The data for ROOMSNUM consist of 19,136 breadwinners of the households in the York and Humb area. Rate of missingness is 6.3% for ROOMSNUM. Values of ROOMSNUM are between 1 and 15, 15 meaning more than 14 rooms in household.

32. The household level variable, Number of rooms, has been tried to impute many times in all versions of the SARS data. TS-SOM based nearest neighbour imputation gives very good results when using five explanatory variables (in finding the correct donor!) mentioned in Table 5. Mean deviations are relatively small and distribution of the imputation data remains very good.

VI. CONCLUSIONS

33. Self-organizing mapping is a widely used and known method for many kinds of data analyses. Efficient tree-structured self-organizing mapping is implemented to the NDA software that gives a powerful tool for data analysis, missingness analysis and imputation with graphical presentation facilities. Classical imputation methods can be used on different levels of the TS-SOM structure, but regression based and more complex methods applying the mathematical idea behind TS-SOM can also be utilized. The SECI group has also presented the TS-SOM training algorithm especially for incomplete data with sampling weights. In the imputation examples it was shown how these modern algorithms can reduce time complexity besides giving fine classification or discretization for the data. TS-SOM can also be useful after imputation tests in trying to find some specific subgroups of high differences between imputed and corresponding true values or other abnormalities, in other words, trying to get the zone in Figure 2 narrower by keeping the fitted regression line (for imputed and their true values) close to $\hat{y} = y^*$. The software will also include editing features in the future. These modern methods as well as software tools are under intensive further development.

REFERENCES

[1] Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, United Kingdom.

[2] Häkkinen, E. (2001). *Design, Implementation and Evaluation of the Neural Data Analysis Environment*. PhD thesis. Jyväskylä University Library, Jyväskylä, Finland.

[3] Kohonen, T. (1997). *Self-Organizing Maps*. Springer, Berlin, Heidelberg.

[4] Koikkalainen, P. (1995). Fast Deterministic Self-Organizing Maps. In Fogelman-Soulié, F. and Gallinari, P., eds., *Proc. ICANN'95, Int. Conf. on Artificial Neural Networks*, Volume II, pp. 63-68, Nanterre, France. EC2.

[5] Koikkalainen, P. (1999). Tree Structured Self-Organizing Maps. In Oja, E. and Kaski, S., eds., *Kohonen Maps*, pages 121-130. Elsevier, The Netherlands.

[6] Koikkalainen, P. and Oja, E. (1990). Self-Organizing Hierarchical Feature Maps. In *Proc. IJCNN-90-Wash-DC, Int. Joint Conf. on Neural Networks*, Volume II, pp. 279-285, Piscataway, NJ., IEEE Service Center.

[7] Piela, P. & Laaksonen S. (2001): Automatic Interaction Detection for Imputation – Tests with the WAID Software Package. Contributed Paper for the *Federal Committee on Statistical Methodology Research Conference*, Washington, DC Area.

[8] Ritter, H., Martinez, T., and Schulten, K. (1992). *Neural Computation and Self-Organizing Maps: An Introduction*. Addison-Wesley, Reading, MA.

Figure 3. SOM visualization for TENURE.

TENURE classes	
1	own outright
2	own buying
3	private rented (furnished)
4	private rented (unfurnished)
5	rented with job/business
6	rented (housing association)
7	rented public sector / England

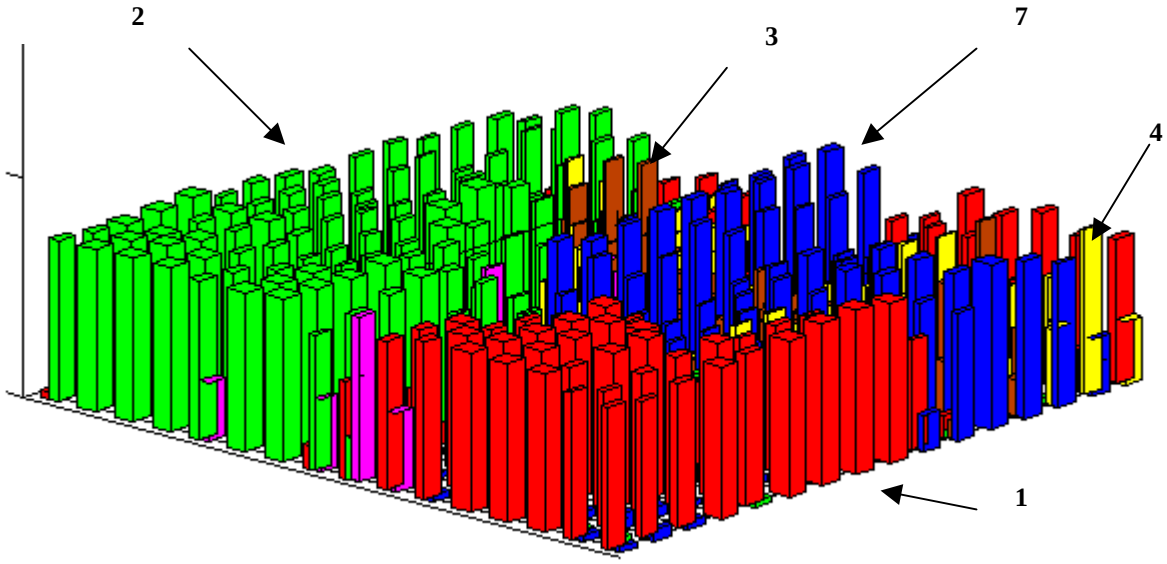
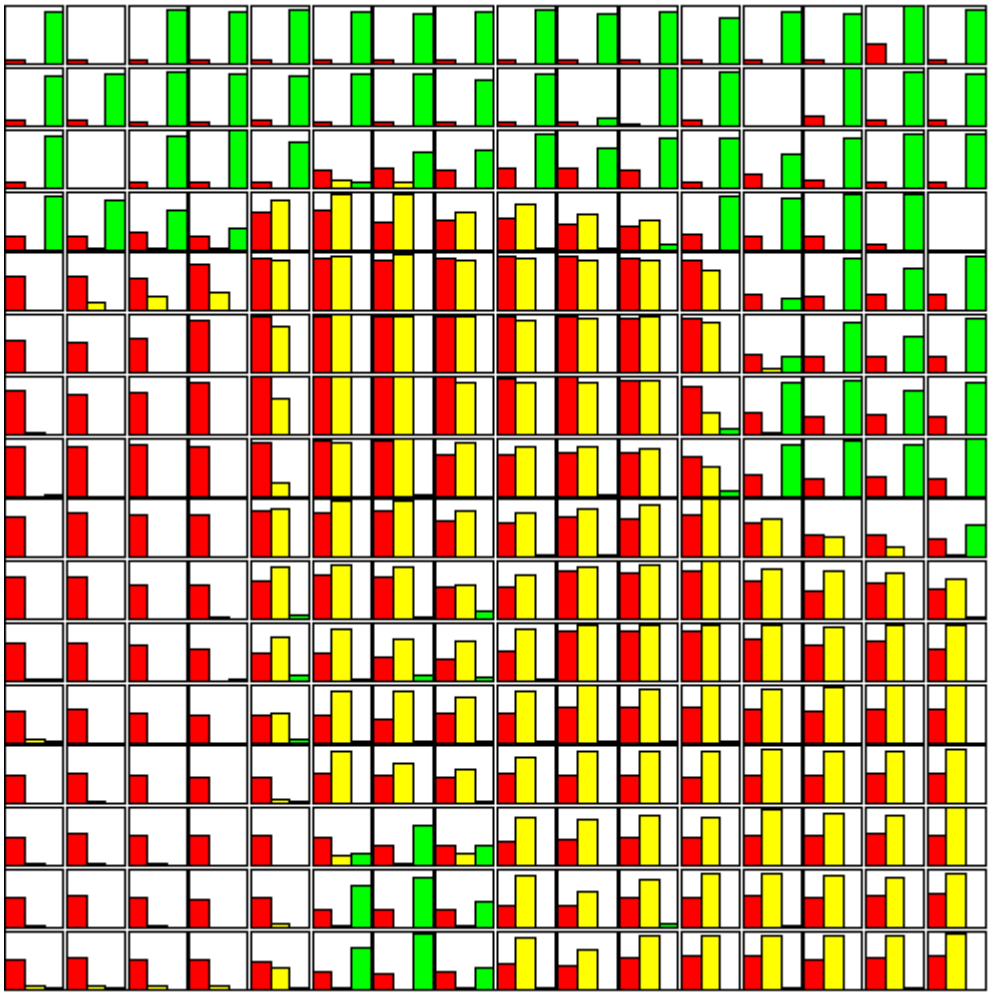


Figure 4. SOM visualization for AGE with RELAT. The left bar indicates the mean of AGE within the cluster, the middle bar gives the proportion of household heads (RELAT = 0) while the right one shows the proportion of children (RELAT = 3).



Method	Mean	Std. Dev.	25% Quantile	Median	75% Quantile	95% Quantile	DL1
True values, N = 53677	158108	107193	76691	140971	221423	362639	0
Random overall	176056	116310	86722	160464	248847	390072	73678
TSSOM L = 4, Centroid	169177	56806	122951	173937	204815	259743	73079
Lin. regression	170287	61498	122344	168978	215989	272202	73678
NN	**158068**	107856	77004	**140105**	220904	363148	90865
TSSOM L = 1, NN	158094	**107855**	**76869**	139715	**221534**	**362674**	91072
TSSOM L = 2, NN	159799	108380	77914	141657	224541	363374	91593
TSSOM L = 3, NN	159408	108253	77779	141616	223804	363712	91639
MLP, method I	167354	69143	107494	173607	225255	280030	**67109**
MLP, method II	166806	68920	109096	162920	219764	277061	67279

MLP activation functions are sigmoidal.

MLP Method	1st Hidden Layer	2nd Hidden Layer
I	4 neurons	6 neurons
II	10 neurons	10 neurons

Table 2. NDA test results for INCOME. NN = nearest neighbour imputation, Random = random imputation, TSSOM L = TSSOM clustering, at level L. $L = k$ means that the data have been divided into 4^k clusters/subclasses. Results for Linear Regression by using SOLAS 3.0™ [3].

DL1 is the average difference: $d_{L1}(\hat{\mathbf{Y}}, \mathbf{Y}^*) = \sum_{i=1}^{n} w_i \left| \hat{Y}_i - Y_i^* \right| \Big/ \sum_{i=1}^{n} w_i$, here: $w_i = 1 \; \forall \; i \in \mathbf{N}$.

Method	Mean	Std. Dev.	25% Quantile	Median	75% Quantile	95% Quantile	DL1
True values, N = 3626	37.26	23.05	19	35	55	76	0
Random overall	37.27	22.90	**19**	**35**	**55**	77	13.90
TSSOM L = 2, NN: R E M Re H	39.96	27.37	15	41	61	87	13.16
TSSOM L = 2, Random	37.42	22.89	**19**	**35**	**55**	77	5.14
TSSOM L = 3, Random	37.57	**22.84**	**19**	36	**55**	77	4.80
TSSOM L = 4, Random	**37.26**	22.91	20	**35**	**55**	**76**	4.72
TSSOM L = 5, Random	37.71	23.04	**19**	36	56	77	4.98
NN: Re + TSSOM L = 4, Random	37.00	23.16	**19**	**35**	**55**	**76**	**4.33**

Table 3. NDA test results for AGE. R = ROOMSNUM, E = ECONPRIM, M = MSTATUS, Re = RELAT, H = HHSPTYPE.

Method	Mean	Std. Dev.	25% Quantile	Median	75% Quantile	95% Quantile	DL1
True values, N = 1556	35.44	12.46	32	39	40	50	0
Random overall	**35.61**	12.09	**32**	**39**	**40**	54	12.07
TSSOM L = 3, NN: I E	32.72	14.15	24.5	38	**40**	**50**	9.21
TSSOM L = 3, NN: I E A S Re	33.43	14.73	26.5	38	**40**	52	8.96
TSSOM L = 4, Normal prediction	34.78	8.23	31	37	**40**	45	7.10
TSSOM L = 5, Normal prediction	34.78	8.8	31	37	41	45	**6.92**
TSSOM L = 4, Random	35.11	11.62	**32**	38	**40**	**50**	8.59
TSSOM L = 5, Random	34.94	**12.50**	**32**	38	**40**	**50**	8.75

Table 4. NDA test results for HOURS. Normal prediction = Normal curve prediction imputation. I = ISCO1, R = ROOMSNUM, E = ECONPRIM, M = MSTATUS, Re = RELAT, H = HHSPTYPE, S = SEX, A =AGE.

[3] Solas™ and Solas 3.0™ are trademarks of Statistical Solutions LTD.

Method	DL1	Number of Rooms (%)					
		1-2	**3**	**4**	**5**	**6**	**7+**
True values, *N* = 1214	0	3.95	8.57	23.39	31.55	20.43	12.1
Random overall	1.55	4.78	8.24	**22.89**	32.37	18.78	**12.85**
TSSOM L = 2, NN: H P	1.27	5.44	6.84	22.41	28.25	**20.68**	16.39
TSSOM *L* = 1, NN: E H P T M	1.25	4.86	9.14	21.33	27.35	21.33	15.98
TSSOM *L* = 2, NN: E H P T M	1.23	4.86	7.74	21.00	30.07	22.90	13.42
TSSOM *L* = 3, NN: E H P T M	**1.21**	**3.79**	**8.65**	19.85	**31.71**	21.91	14.07
TSSOM *L* = 5, Centroid	**0.99**	1.31	7.74	36.99	43.82	7.08	3.05

Table 5. Imputation results for ROOMSNUM (number of rooms). Centroid = cluster centroid imputation. P = PERSINHH, E = ECONPRIM, M = MSTATUS, H = HHSPTYPE, T = TENURE.