Paper 1470-2017

N-Stage Machine Learning Analysis with LUA Procedure Helps Solve Big Data Analysis Problems

Ryo Kiguchi, Eri Sakai, Yoshitake Kitanishi, Akio Tsuji, SHIONOGI & CO., LTD.

ABSTRACT

Data has to be a moderate size when you estimate parameters with machine learning. For data with huge numbers of records like healthcare big data (for example, receipt data) or super multi-dimensional data like genome big data, it is important to follow a procedure in which data is cleaned first and then selection of data or variables for modeling is performed. Big data often consists of macroscopic and microscopic groups. With these groups, it is possible to increase the accuracy of estimation by following the above procedure in which data is cleaned from a macro perspective and the selection of data or variables for modeling is performed from a micro perspective. This kind of step-wise procedure can be expected to help reduce bias. We also propose a new analysis algorithm with N-stage machine learning. For simplicity, we assume N =2. Note that different machine learning approaches should be applied; that is, a random forest method is used at the first step, and an elastic net method is used at the second step. For programming N-stage machine learning, we use the LUA procedure that is not only efficient, but also enables an easily readable iteration algorithm to be developed. Note that we use well-known machine learning methods that are implementable with SAS® 9.4 and SAS product IMSTAT etc.

1. INTRODUCTION

The world contains a large amount of data with various structures such as structured data and/or unstructured data. For example, there are the following two data structures:

- Vertical data
  - Healthcare big data such as receipt data. These data are often called Real World Data (RWD). This kind of data structure features many records and few variables
- Horizontal data
  - Genome big data. This kind of data structure features few records and a great many variables

Kitanishi (2016) summarizes the analysis approaches and analysis types for these data structures, and Table 1 shows the summary of analysis approaches and analysis types based on Kitanishi's summary.

Table 1. Vertical and Horizontal Big Data

Big Data Type	Analysis approaches	Analysis Types (Examples)
Vertical Data (Rule Detection)	Sorting Algorithm Contingency Counting Algorithm	Contingency Table Analysis *Association Rule Mining* Signal Detection Time-to-Event Analysis etc.
Horizontal Data (Feature Selection)	Multiple Comparison	Bonferroni False Discovery Rate (FDR) etc.
	Dimension Reduction	*Principal Component Analysis (PCA)* *- Sparse PCA* *Canonical Correlation Analysis* *Support Vector Machine (SVM)* etc.
	Variable Selection	*Clustering* *Random Forest* *LASSO (Least Absolute Shrinkage and Selection Operator)* *Group LASSO* *Elastic Net* etc.

The analysis approaches and methods depend on the data structure. However, data cleaning (ex. record or explanatory variable selection, missing data imputation etc.) prior to modelling is important to provide appropriate interpretation for an outcome of interest, which does not depend on the data structure. On the other hand, big data often consists of macroscopic and microscopic groups. With these groups, we consider that it is possible to increase the accuracy of estimation by following the procedure in which data are cleaned variables unrelated to model construction are removed from the macro perspective, and the selection of data or variables for modeling is performed from the micro perspective. This kind of stepwise procedure can be expected to help reduce bias and bring an efficient analysis process. For example, remember the gold rush. A method to excavate gold in the mountains from a micro viewpoint would be the more efficient method of finding gold after predicting a mountain where gold is likely to be excavated from a macro viewpoint, rather than digging in several mountains at random. Therefore, we would like to propose N-stage machine learning analysis. This method's details are shown in Chapter 2. We build a good algorithm to implement this analysis. We consider that the good algorithm combines high estimation accuracy, short computation time, and easy parameter tuning. In addition, high readability is also important to build a good algorithm because big data analyses are often done by multiple people. We introduce examples of machine learning and a program language that is used to code the highly readable program. By doing so, we provide you with an easy-to-use analysis algorithm. As examples of machine learning, we introduce random forest, LASSO and elastic net as shown in Table 1 and chapter 3. These machine learning approaches are implementable with SAS® 9.4 (LASSO / elastic net) and SAS® In-Memory Statistics for Hadoop™ (IMSTAT) (random forest). Using these SAS products, we can easily write readable programs. Incidentally, most of the methods shown in Table 1 can be implemented in these SAS products. In addition, the proposed method implements the next step according to the result of previous steps, so it is necessary to construct a program that can simplify macro processing and perform variable operation. Therefore, we utilize the Lua procedure, which was formally implemented in SAS® 9.4. The description of the Lua procedure is shown in chapter 4. Finally, we show the simulation in chapter 5 to consider whether the proposed analysis improves the estimation accuracy and reduces the calculation cost compared with the case of applying machine learning only once.

2. N-STAGE MACHINE LEARNING ANALYSIS

This analysis method has a data cleaning step and a modeling step. Data cleaning is performed to reduce noise of model construction in the 1^{st} to $(N-1)^{th}$ steps, and model construction is performed in the N^{th} step. In the 1^{st} to $(N-1)^{th}$ steps (data cleaning), we build an algorithm by combining, for example, deletion of noise data and/or variables unrelated to model construction, data supplementation, etc. For simplicity, we assume that N is 2 and that the data cleaning step (1^{st} step) is the deletion of noise variables unrelated to model construction. The target data structure is horizontal. We introduce the concept of 2-stage machine learning analysis. This proposed method is based on a new analysis algorithm in which machine learning is performed twice under different approaches to improve estimation accuracy and reduce calculation cost. Figure 1 shows a conceptual diagram of 2-stage machine learning analysis. First machine learning plays the role of data cleaning, so variables and records are picked up at this step. Note that the 1^{st} step is the step that excludes variables and data that may become noise, so it is not the selection of variables and data for modeling. Therefore, this step must weaken arbitrariness when tuning parameters. Thus, the extraction condition is relaxed in the 1^{st} machine learning. In other words, we set the gray zone. At the next step, for the data after the first step, we build the model with 2^{nd} machine learning to estimate objective variable (an outcome of interest). At that time, we recommend using different machine learning from the first step. You can use the same machine learning, but in that case, it is necessary to be stricter than the selection criteria of the first step. However, the second step criteria are automatically stricter than the first step because 2^{nd} machine learning is used in the ordinary way, but 1^{st} machine learning has looser criteria than the ordinary way.

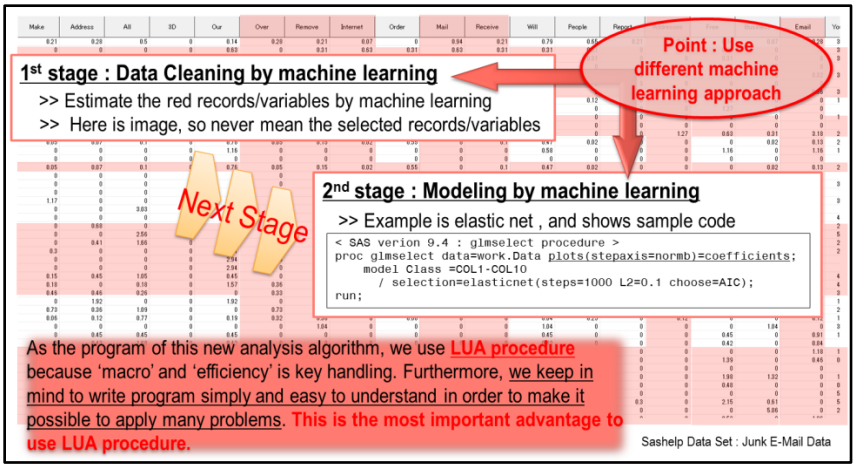

Figure 1. Conceptual diagram of 2-stage machine learning analysis

3. IMPLEMENT MACHINE LEARNING WITH SAS

In this chapter, we introduce three machine learning programming codes that we can implement using IMSTAT and SAS® 9.4 used in practical data analysis: random forest, LASSO, and elastic net.

3.1 RANDOM FOREST

Random forest is a machine learning algorithm that uses many decision trees or regression trees learned by randomly sampled training data. In other words, the idea is the combination of a decision tree or regression tree and bootstrapping. The advantages of this method are that learning is early, over-training does not easily occur, etc., but the most important advantage is that it is possible to calculate the "importance," which is a feature quantity in the learning step. With these advantages, random forest is commonly used as a variable selection method. Random forest can be implemented by using the RANDOMWOODS statement of IMSTAT. The sample code is shown below.

```
proc imstat DATA=LASRLIB.Dataset;
   RANDOMWOODS OUTCOME/                                          /*Response variable*/
       INPUT     = (COL1 COL2 COL3 COL4 COL5 COL6)             /*Explanatory variable*/
       NOMINAL   = (COL3 COL5 COL6)                             /*Category variable*/
       M         =4                        /*Sampling number of explanatory variable*/
       LEAFSIZE  = 5                                              /*Size of Leaf*/
       MAXBRANCH = 2                                    /*Maximum number of branch*/
       MAXLEVEL  = 10                                          /*Depth of tree*/
       BOOTSTRAP = 0.8                                       /*Default:1-exp(-1)*/
       NTREE     = 3000                                          /*Number of tree*/
       SAVE      = TEST;                                /*Store result of random forest*/
   STORE TEST[2] (where="Importance>0.1",cols=VarName)=Importance;
                                                    /*Store information of importance*/
run;
quit;
```

Using the RANDOMWOODS statement, we can easily program a random forest. However, random forest has the disadvantages that it has many parameters and cannot be learned well if the records are small. We must pay attention to these disadvantages.

3.2 LASSO & ELASTIC NET

As described in the Introduction, this study assumes the application of horizontal data such as genome big data. As a characteristic of genome big data, the data feature some groups with strong correlation among explanatory variables. However, the correlation may accidentally arise by variables not included in the group because the number of records is small. Especially, the variables that are correlated by chance are difficult to remove as a group. Because genomic big data contains many noise variables unrelated to model construction, it is necessary to delete the noise variables beforehand to improve the prediction accuracy of the model, and it is better not to focus on "variables" but rather on “groups” with strong correlation. Therefore, we focused on LASSO, which identifies important variables, and elastic net, which can consider correlation. Next, we would like to introduce these methods.

First, we introduce LASSO. The estimator $\hat{\beta}$ of a general linear regression model using LASSO is expressed as follows:

$$\hat{\beta} = \mathrm{argmin}_{\beta}(y - X\beta)^2 + \lambda|\beta|\}.$$

The second term on the right side is called a penalty term. By adjusting the tuning parameterλ, shrinkage estimation is possible, so variable selection is possible. However, in the case of $\mathrm{p} > \mathrm{n}$ where n is the number of data and p is the number of explanatory variables, LASSO can select no more than n explanatory variables. If there are groups with strong correlation among explanatory variables, LASSO tends to select only one variable from each group and ignore the other variables. This characteristic is an inconvenient restriction on the genome big data. Thus, next, we introduce elastic net, which can select the groups. Elastic net can consider the group effects due to correlation between variables in addition to the variable selection and continuous shrinkage estimation, which are the characteristics of LASSO. The estimator $\hat{\beta}$ of a general linear regression model using elastic net is expressed as follows:

$$\hat{\beta} = (1 + \lambda_2) \times \mathrm{argmin}_{\beta}\{(y - X\beta)^2 + \lambda_2\|\beta\|^2 + \lambda_1|\beta|_1\},$$

where λ_1 is a turning parameter with shrinkage estimation, and λ_2 is a turning parameter with group effects. LASSO and elastic net can be implemented with the glmselect procedure of SAS® 9.4. The sample codes are shown below.

Lasso [SAS® 9.4]

```
proc glmselect data=work.Data plots=all ;
         model OUTCOME=COL1-COL10
           / selection=lasso(steps=1000 choose=AIC) ;
run ;
```

Elastic net [SAS® 9.4]

```
proc glmselect data=work.Data plots=all ;
         model OUTCOME=COL1-COL10
           / selection=elasticnet(steps=1000 L2=0.1  choose=AIC) ;
run ;
```

As one of the advantages using the glmselect procedure, it is possible to visually grasp the transition of variable selection. Figure 2 shows how explanatory variables COL1 to COL10 are incorporated into the model by applying LASSO for certain data. By using such visualizations, we will find it easier to discover new knowledge.

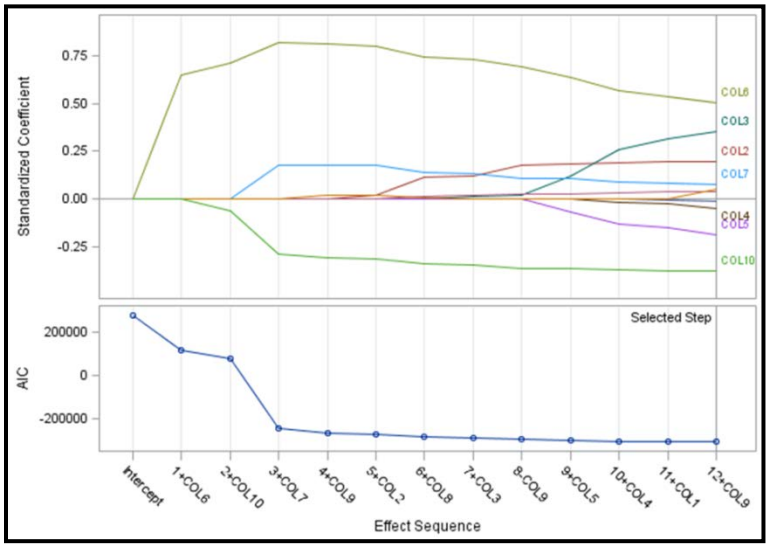

Figure 2. Horizontal axis shows the order of selected variables. Vertical axis is the standardized value. The upper figure is the coefficient value, and the lower figure is AIC

Incidentally, by using the glmselect procedure, we can also implement the forward selection method, backward elimination method, group LASSO, etc. Please use the methods appropriate to your situation.

4. LUA PROGRAM LANGUAGE

The Lua procedure (Lua) is a formally implemented procedure is SAS® 9.4. Lua makes SAS programming easier than ever, and Lua's data structure is supported, thus breaking the limitations of previous SAS macros. Lua has many features, but the best feature is the flexible in-memory data structure, which is called Tables. A depiction of the data structure is shown Figure 3.

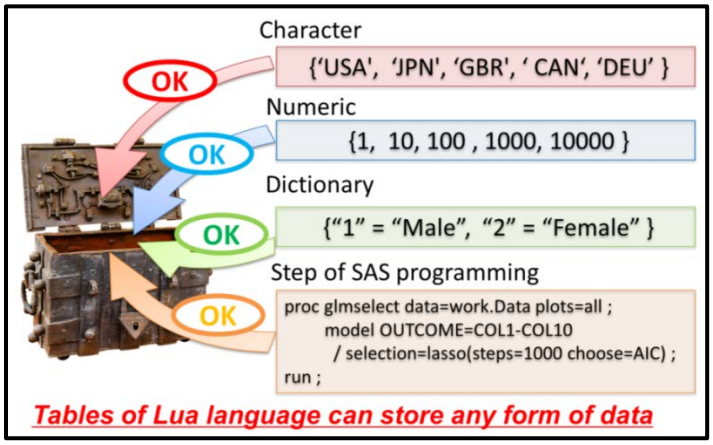

Figure 3. Lua's data structure. Tables of the Lua language can store any form of data

Any data type can be stored—for example, numeric type, character type, function, and step of SAS programming. You can code the data at the top of Figure 3 with Lua as follows.

```
proc lua;
   submit;
   local Country_name = {'USA', 'JPN', 'GBR', 'CAN', 'DEU'}
            -- define Tables. Restore character values in local variable "Country_name"
    endsubmit;
run;
```

This data structure is very convenient and leads to simplification and improved readability of your program. To realize the variability program, which is simplified macro processing using Lua, we will show some Lua features and sample codes.

4.1 SIMPLE MACRO PROCESSING

As shown in Figure 3, Lua's variable can store the SAS program. Thus, we can execute the program with sas.submit to prestore the program in Lua's variable. Although this can be implemented by using macros in the conventional SAS program, it is obvious that you can write a simpler program by using Lua. We write two programs, which are the conventional SAS program (using macros) and SAS program using Lua, to run the processing shown in the following figure and compare the readability. Comparing two programs, you will find that Lua is superior from the viewpoint of readability.

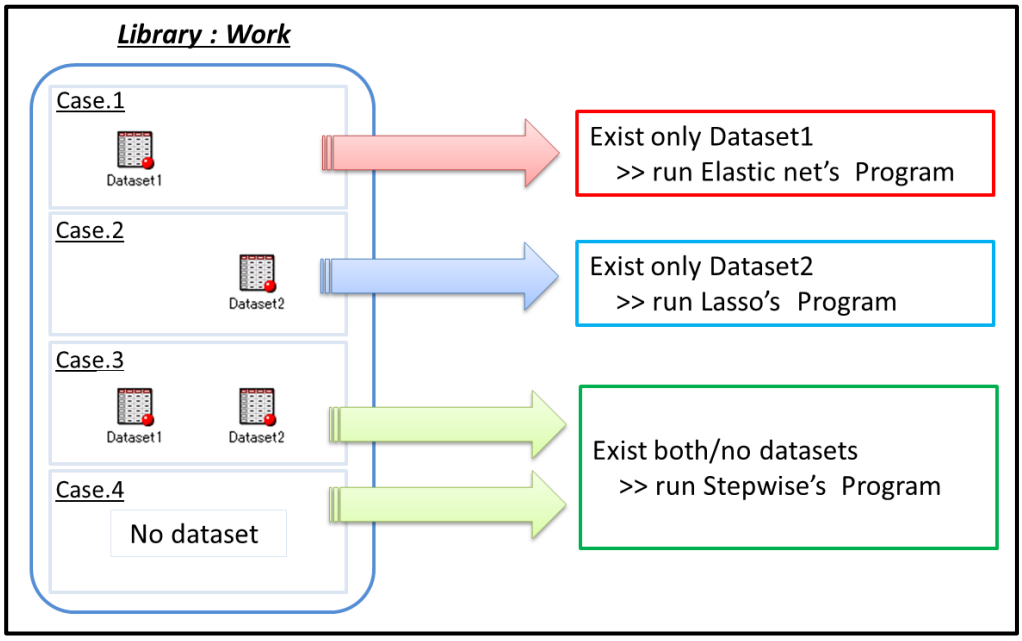

Figure 4. Conceptual diagram of processing to be performed with the following two sample programs

- Conventional SAS program

```
%MACRO ML_MACRO(lib,data1, data2);
  %IF (%SYSFUNC(EXIST(&lib..&data1.) )=1 and  %SYSFUNC(EXIST(&lib..&data2.) )^=1) %THEN
       %DO;
            proc glmselect data=  ---  program to execute Elastic net  ---  run;
       %END;
  %ELSE %IF( %SYSFUNC(EXIST(&lib..&data1.) )= 0 and  %SYSFUNC(EXIST(&lib..&data2.) )= 1) %THEN
       %DO;
            proc glmselect data=  ---  program to execute Lasso  ---  run;
       %END;
  %ELSE %IF( %SYSFUNC(EXIST(&lib..&data1.) )= 1 and  %SYSFUNC(EXIST(&lib..&data2.) )= 1) %THEN
       %DO;
            proc glmselect data=  ---  program to execute Step wise  ---  run;
       %END;
  %ELSE %IF( %SYSFUNC(EXIST(&lib..&data1.) )= 0 and  %SYSFUNC(EXIST(&lib..&data2.) )= 0) %THEN
       %DO;
            proc glmselect data=  ---  program to execute Step wise  ---  run;
       %END;
%MEND;
%ML_MACRO (Work ,  dataset1,  dataset2)
```

・SAS program using Lua

```
[Step1: Store the machine learning program used at Step2 to global variable MLcode]
proc lua ;
     submit;
        MLcode={}
        MLcode.elasticnet = [[proc glmselect data=  ---  program of Elastic net  ---  run; ]]
        MLcode.lasso      = [[proc glmselect data=  ---  program of Lasso        ---  run; ]]
        MLcode.stepwise   = [[proc glmselect data=  ---  program of Stepwise    ---  run; ]]
     endsubmit;
quit;

[Step2: Confirm existence of dataset1/dataset2 and execute program]
proc lua ;
 submit;
  if  sas.exist("work.dataset1"   )~=0 and sas.exist("work.dataset2")==0 then sas.submit(MLcode.elasticnet)
  elseif sas.exist("work.dataset1")==0 and sas.exist("work.dataset2")~=0 then sas.submit(MLcode.lasso)
  else                                                                       sas.submit(MLcode.stepwise)
  end
 endsubmit;
quit;
```

4.2 MAKING THE VARIABILITY PROGRAM

In the algorithm of the proposed method, it is important to execute the SAS code variably because we want to change the SAS code of 2nd machine learning according to the result of 1st machine learning. Lua can also use processing to handle SAS code variably. An implementation example is shown below. Suppose that we have a dataset to store currency per one US dollar of each country or area and the ISO code (Figure 4). Using this dataset, we write and compare two programs, the conventional SAS program and SAS program using Lua, to calculate other currencies based on the currency of each country or area shown in Figure 5. SAS program using Lua is longer than the conventional SAS program, but it is a better program from the viewpoint of variability. Suppose that the new record “Country/Erea = ‘NewZealand’, ISO CODE = ‘NZD’, Currency per USD = 1.38” is given to the dataset. SAS program using Lua does not need to be edited, but the conventional SAS program must be edited such that % NEW_VALURE('NewZealand') is added. Of course, using a FOR statement, you can write a program that does not require editing, but we do not recommend this because you must exclude America from the handling of the program, and the program becomes more complex. Similar operation is possible with a CALL EXECUTE or DOSUBL statement implemented in SAS® 9.3. However, although this may be subjective, we consider Lua to be more readable.

	Country/Erea	ISO CODE	Currency per USD
1	Japan	JPY	112.38
2	America	USD	1
3	Euro	EUR	0.93
4	Britain	GBP	0.8
5	Koria	KRW	1133.02
6	China	CNY	6.85
7	Australia	AUD	1.31
8	Canada	CAD	1.3
9	Switzerland	CHF	0.99
10	SouthAfrica	ZAR	13.29

Figure 4. Dataset to store currency per one US dollar of each country or area and the ISO code (https://www.oanda.com/lang/ja/currency/converter/)

	Country/Erea	ISO CODE	Currency per USD	Currency per JPY	Currency per EUR	Currency per GBP	Currency per KRW	Currency per CNY	Currency per AUD	Currency per CAD	Currency per CHF	Currency per ZAR
1	Japan	JPY	112.38	1	120.8387	140.475	0.0992	16.4058	85.7863	86.4462	113.5152	8.456
2	America	USD	1	0.0089	1.0753	1.25	0.0009	0.146	0.7634	0.7692	1.0101	0.0752
3	Euro	EUR	0.93	0.0083	1	1.1625	0.0008	0.1358	0.7099	0.7154	0.9394	0.07
4	Britain	GBP	0.8	0.0071	0.8602	1	0.0007	0.1168	0.6107	0.6154	0.8081	0.0602
5	Koria	KRW	1133.02	10.082	1218.3011	1416.275	1	165.4044	864.9008	871.5538	1144.4646	85.2536
6	China	CNY	6.85	0.061	7.3656	8.5625	0.006	1	5.229	5.2692	6.9192	0.5154
7	Australia	AUD	1.31	0.0117	1.4086	1.6375	0.0012	0.1912	1	1.0077	1.3232	0.0986
8	Canada	CAD	1.3	0.0116	1.3978	1.625	0.0011	0.1898	0.9924	1	1.3131	0.0978
9	Switzerland	CHF	0.99	0.0088	1.0645	1.2375	0.0009	0.1445	0.7557	0.7615	1	0.0745
10	SouthAfrica	ZAR	13.29	0.1183	14.2903	16.6125	0.0117	1.9401	10.145	10.2231	13.4242	1

Figure 5. Dataset of other currencies based on currency of each country or area

・ Conventional SAS program

```
%MACRO NEW_VALURE(Country);
  data _null_;
    set work.meta(where = (Country="&Country.")  );
    call symputx("value", value);
    call symputx("unit", unit);
  run;

  data work.meta;
    set work.meta;
    &Country._value = round(value/&value., 10**-4);
    label &Country._value = "Currency per &unit.";
  run;
%mend NEW_VALURE;

%NEW_VALURE(Japan)    %NEW_VALURE(Euro)   %NEW_VALURE(Britain)     %NEW_VALURE(Koria)  %NEW_VALURE(China)
%NEW_VALURE(Australia)%NEW_VALURE(Canada) %NEW_VALURE(Switzerland) %NEW_VALURE(SouthAfrica)
```

・ SAS program using Lua

```
proc lua;
    submit;
       local data = sas.open("work.meta")
       sas.where(data, "Country not in ('America')")
       local nobs = sas.nobs(data)
       sas.symput("obs",nobs)
       for i = 1, sas.symget("obs") do
          while sas.next(data) do
          sas.where(data, "_N_ = i")
          sas.submit_([[data work.meta_@Country@; ]],{Country= sas.get_value(data,"Country")})
          sas.submit_([[set work.meta;]])
          sas.submit_([[@Country@_value = round(value/@value@, 10**-4);;]],
                              {Country= sas.get_value(data,"Country"), value= sas.get_value(data,"value")})
          sas.submit_([[label @Country@_value = "Currency per @unit@";]],
                               {Country= sas.get_value(data,"Country"), unit= sas.get_value(data,"unit")})
          sas.submit([[run;]])
          end
       end
       sas.submit([[data Result;  merge meta_:; run;]])
       sas.close(data)
    endsubmit;
run;
```

5. SIMULATION

Details of the simulation are shown in the presentation. Based on the contents of chapters 3 and 4, we build an algorithm of 2-stage machine learning analysis. In the simulation, we use horizontal data such as genome big data. We plan to consider whether the proposed method increases estimation accuracy and reduces computational cost compared to the case of using one machine learning applied only once.

6. CONCLUSION

We proposed an N-stage machine learning analysis because we consider that it is possible to perform more accurate modeling by combining existing methods, although a useful analysis method has been studied for genome data including many noise variables unrelated to model construction. The points of this analysis are to use different machine learning and to construct the model in the N^{th} step after preremoving noise information unrelated to model construction in the 1^{st} to $(N-1)^{th}$ steps. In addition, the Lua using this analysis makes the SAS program more concise, more readable, more flexible, and more elegant. Big data analyses are often done by multiple people and so indications of a good algorithm include the calculation time, operation cost (server cost, labor of parameter tuning etc.), and readability. Lua optimizes the task of parameter tuning of operation costs and readability, so we encourage its active use. However, Lua cannot improve the calculation time. However, we think that the calculation time can be improved by the algorithm of the proposed method. Simulation results of the calculation time will be introduced on the day of presentation. Finally, let us show the possibility of this proposed method. In this paper, we introduced the cleaning step of noise variable deletion. However, a more accurate model may be constructed by modifying the analysis of the following procedure. Note that N = 1 to 4 is the cleaning step, and N = 5 is the modeling step.

- ✧ N = 1 : Step of deleting noise variables unrelated to model construction
- ✧ N = 2 : Step of deleting noise records unrelated to model construction

- ✧ N = 3 : Step of imputation of missing data
- ✧ N = 4 : Step of standardizing data
- ✧ N = 5 : Step of modeling

We hope that N-stage machine learning analysis will help you analyze big data.

7. REFERENCES

SAS Institute Inc. 2015. SAS® LASR™ Analytic Server 2.7: Reference Guide. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2015. SAS/STAT® 14.1 User's Guide. Cary, NC: SAS Institute Inc.

Rick Langston 2013. "Submitting SAS® Code On The Side." SAS Global Forum 2013, San Francisco, California. Available at https://support.sas.com/resources/papers/proceedings13/032-2013.pdf.

Paul Tomas 2015. "Driving SAS® with Lua." SAS Global Forum 2015, Dallas, Texas. Available at http://support.sas.com/resources/papers/proceedings15/SAS1561-2015.pdf.

Yoshitake Kitanishi 2016. "PROC IMSTAT Boosts Knowledge Discovery in Big Databases (KDBD) in a Pharmaceutical Company." SAS Global Forum 2016, Las Vegas, Nevada. Available at http://support.sas.com/resources/papers/proceedings16/11780-2016.pdf.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at the following addresses:

Ryo Kiguchi

Shionogi & Co., Ltd.

ryo.kiguchi@shionogi.co.jp

Eri Sakai

Shionogi & Co., Ltd.

eri.sakai@shionogi.co.jp

Yoshitake Kitanishi

Shionogi & Co., Ltd.

yoshitake.kitanishi@shionogi.co.jp

Akio Tsuji

Shionogi & Co., Ltd.

akio.tsuji@shionogi.co.jp