DATA ANALYSIS SYSTEM WITH STATISTICAL KNOWLEDGE BASE

MARCH 1995

Kikuo YANAGI

The Graduate School of Natural Science and Technology
(Doctor Course)
OKAYAMA UNIVERSITY

(1)

DATA ANALYSIS SYSTEM
WITH
STATISTICAL KNOWLEDGE BASE

MARCH 1995

Kikuo YANAGI

The Graduate School of Natural Science and Technology
(Doctor Course)
OKAYAMA UNIVERSITY

Contents

1 Introduction

There are several popular packages for statistical analysis, but most of them require that the users must have a sufficient statistical knowledge. Though, even the user who has a little statistical knowledge can use those packages for statistical analysis, and he gets some results. But they may have no meaning in statistical analysis. According to growing of the computer environment, those packages are more popular and such 'miss using's are more.

On the other hand, many expert systems have been developed recently, on various domains. An expert system has some knowledge, and make some decision instead of a user. Also, in statistical analysis, a number of expert systems have been developed or proposed. They are designed for the users who have a little statistical knowledge, and when such users use the system, it supports and advises users for making no miss using. Discussions with respect to the expert system for statistical data analysis are found in REX (Gale, 1986), Student (Gale, 1986), RASS (Nakano et al., 1990), SCSH (Hayashi, 1993) and SCSK (Hayashi and Tarumi, 1994)

An expert system consists of a knowledge base, an inference engine and a calculation engine. A calculation engine executes an analysis and makes a result according to a decision which was made by an inference engine. Therefore, implementations of a knowledge base and an inference engine are important to develop an expert system. Discussion of a knowledge base and an inference engine are found in previous system and in Thisted (1986), Spiegelhalter (1986), Minami et al. (1993a, 1993b, 1994) and Yanagi (1994).

As another type of approach, Afifi and Clark(1990) have shown that attributes of variables were important factors in deciding on an appropriate analysis. According to the distinction of the variable type (continuous or discrete) and the role of variable (dependent or independent), data are classified into several groups. That is to say, if we have sufficient information for each variable, we can select an appropriate method for such data. Actually, this concept is similar to the Object Oriented Programming (OOP) style. In the OOP design, the group is denoted as a class, and the relationship of the classes is constructed in term of hierarchical structure.

Classes of statistical data are classified according to the attributes of variable, and classes of matrix and classes of result are defined properly. In our proposed expert system, all values, graphics and texts are considered as instances of a certain class and the statistical analysis is a 'method' in the class, and it is activated by 'message sending' style.

In this thesis, we discuss a knowledge base, an inference engine and a calculation engine for developing an expert system which make results of statistical analysis from sufficient information of the data automatically.

In chapter 2, we explain an Object-Oriented Programming technique and define Matrix classes for matrix calculation using an OOP technique. And we develop a system for matrix calculation using Matrix classes and MAT language for the system. Matrix calculation are basic operations for statistical data analyses, Matrix classes are used in calculation engine.

In chapter 3, we define Data classes which are also used in calculation engine. These classes have 'methods' which mean statistical data analysis methods. The matrix classes which are defined in Chapter 2, are used to implement these methods. And we develop the '*Stat*' system for statistical data analysis using Data classes and Matrix class. The MAT language can be used in 'Stat' system, it is possible to apply a new analysis method which is not implemented in 'Stat' system by programming with MAT language.

There are three major models to implement of an inference engine, "frame model", "network model" and "blackboard model". But in chapter 4, we propose a new model using an OOP technique. From using the model, classes for knowledge can be used both a knowledge base and an inference engine, and such a knowledge can be used in other statistical system. We define 'Knowledge class' in this chapter, and show an expression of some a knowledge using these classes.

Last, in chapter 5, we explain a method for detecting influential subjects using the clustering method as an example of knowledge. And we explain a clustering method using the linked lines rotation graphics, which is the one method for clustering.

2 Object-Oriented Programming and Matrix Classes

2.1 Introduction

An object-oriented programming (OOP) technique is a new paradigm of programming technique and it aims to reduction of work of programming. This technique dose not only give a boon for programmer, but also for the users who use a system programmed using the OOP technique.

In section 2.2, we explain the the OOP technique simply, and in section 2.3, we define 'Matrix classes' as example of the OOP technique. After we define Matrix classes, in section 2.4, we design MAT language for matrix calculation using Matrix classes and develop MAT system with this language.

2.2 Object-Oriented Programming

There are many useful concepts in the OOP technique that can be utilized to create a hierarchical class structure. We show five main concepts for understanding the OOP technique.

Class: A class is concept like a set. But relationship between two classes is allowed only exclusive or conclusive. If Class B is conclusive to Class A, Class A is called *super class* of Class B and Class B is called *subclass* of Class A. Hence, classes are illustrated by tree structure. A class on top of tree structure is called *Object class*, and only this class has no super class. Every class has methods and properties. In this thesis, a name of a class is started by capital character.

An element of certain class is called *instance* of that class.

Method: A method is concept like a function or a subroutine in ordinary programming, but methods are characterized by the class. Hence, after the class is determined, those methods are only limited to that instance of particular class. In this thesis, a name of method is started by small character.

It is called '*send message*' to execute a method. We can get information of properties of instance or change properties only by sending message.

Polymorphism: There are some methods which have same name but we obtain different results since they belong to different classes. Therefore, it is possible in some classes to define a method which is similar but different. It makes reduction of work to define methods have same name and similar effect in several classes, because programmer need not consider difference of classes.

Inheritance: Since subclass inherits all the properties and methods of its super class, we define the method used in most of the classes based on super class only once. Off course, we can redefine the method which inherited from its super class in certain subclass (*overriding*).

Encapsulation: We must consider that classes mean machine parts. When we add a new method to a certain class, we need to change only the program of the class and not necessarily modifying other classes without special cases. Also, when we define new subclass, we need to write only the program of the subclass. It is simple to add some methods or to modify them.

The OOP technique has more useful feature, refer to Cox (1986), Wiener and Prinson (1988) and Prinson and Wiener (1991).

2.3 Matrix class

Because matrix calculation is necessary for almost statistical analysis method, we define Matrix class first. It has two subclasses 'Vector class' and 'SquareMatrix class' and the later includes 'SymmetricMatrix class' as its subclass. In Figure 2.1, we show a tree structure of Matrix class and its subclasses.

Matrix: This class is a set of $n \times m$ matrices.

method: transpose, product, addition, subtraction, combine, sweep out and other matrix operations and functions

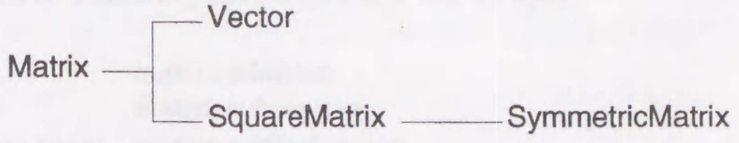


Figure 2.1: Matrix class and its subclass

Vector: This class is a set of vectors. For example mean vector is belong to this class.

method: inner product

SquareMatrix: This class is a set of square $(n \times n)$ matrices.

method: determinant, inverse

SymmetricMatrix: This class is a set of symmetric matrices. For example variance-covariance matrix and correlation matrix are belong to this class.

method: calculate eigenvectors and eigenvalues

Off course, from inheritance concept of the OOP, all subclasses of Matrix class have matrix operation and function methods defined in Matrix class.

2.4 MAT language

We design the programming language 'MAT' for matrix calculation as the application of the 'Matrix class'. It is used when the matrix calculation of some statistical analyses is necessary.

2.4.1 Feature of MAT

There are feature of MAT following.

Definition of matrix Put comma(,) after each element and separate each rows with semi-colon (;). For example,

$$(1,\ 2,\ 3;\ 4,\ 5,\ 6)$$

means

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}.$$

Matrix operator Following matrix operator can be used.

+	matrix addition
−	matrix subtraction
. or space	matrix multiplication
'	transpose of matrix
^	matrix exponent, only positive integer or −1 are allowed.

Table 2.1: Matrix functions

Let A, B are matrices, x is vector and i, j, n are integer.

inverse(A)	return inverse matrix
det(A)	return determinant
rank(A)	return rank
trace(A)	return trace
transpose(A)	return transpose
rows(A)	return the number of rows of matrix
columns(A)	return the number of columns of matrix
hcombine(A, B)	return combined matrix of two (horizontal), i.e. hcombine $(A,\ B) = \begin{pmatrix} A & B \end{pmatrix}$
vcombine(A, B)	return combined matrix of two (vertical), i.e. vcombine $(A,\ B) = \begin{pmatrix} A \\ B \end{pmatrix}$
eigenvalues(A)	return eigenvalues of symmetric matrix A
eigenvalues(A, n)	same, but only n values from largest
eigenvectors(A)	return eigenvectors of symmetric matrix A
eigenvectors(A, n)	same, but only n vectors from largest eigenvalues
sweepout(A, i, j)	return the matrix swept out with pivot (i, j) element
norm(A)	return L_2-norm of matrix
diag(A)	return a vector which elements are diagonal elements of the matrix A
diag(x)	return a matrix which diagonal elements are elements of the vector x and other elements are 0
ident(n)	return $n \times n$ identical matrix
one(n, m)	return a $n \times m$ matrix which all elements are 1
one(n)	return a $n \times n$ matrix which all elements are 1
zero(n, m)	return a $n \times m$ matrix which all elements are 0
zero(n)	return a $n \times n$ matrix which all elements are 0

Element For a matrix A, $(i,\ j)$ element of A is represented `A[i, j]`.

Matrix functions Matrix functions are shown Table 2.1.

Scalar operation and its extension Following scalar operation can be used and extend to matrix operation. For $n \times m$ matrix $A = (a_{ij})$ and $B = (b_{ij})$, $n \times 1$ vector $x = (x_i)$, $1 \times m$ vector $y = (y_j)$ and scalar z, definition of operation is

$$\begin{aligned} A \circ B &= (a_{ij} \circ b_{ij}), & B \circ A &= (b_{ij} \circ a_{ij}), \\ A \circ x &= (a_{ij} \circ x_i), & x \circ A &= (x_i \circ a_{ij}), \\ A \circ y &= (a_{ij} \circ y_j), & y \circ A &= (y_j \circ a_{ij}), \\ A \circ z &= (a_{ij} \circ z), & z \circ A &= (z \circ a_{ij}), \end{aligned}$$

where $\circ$ is one of $+$, $-$, $*$, $/$ which mean respectively addition, subtraction, multiplication, deviation.

Extension of scalar functions Extension of scalar functions (sin, log, exp, ...) are defined by

$$\text{function}(A) = \left(\text{function}(a_{ij}) \right).$$

Programming statement MAT has following control statements.

conditional statement 'if' statement. Syntax 'if' statement are shown in following.

```
if condition then
    (statements condition is true)
end if

if condition then
    (statements condition is true)
else
    (statements condition is false)
end if
```

repetition statement Statement are shown in Following.

```
for(initialize;condition;re-initialize)
    .....
end for
```

repeat while condition is true

```
while(condition)
    .....
end while
```

repeat while condition is true

```
do
    .....
while(condition)
```

repeat while condition is true

```
eachelement(variable,matrix)
    .....
end each
```

repeat for each element of matrix into variable

```
eachrow(variable,matrix)
    .....
end each
```

repeat for each row of matrix into variable

```
eachcolumn(variable,matrix)
    .....
end each
```

repeat for each column of matrix into variable

User interface functions User interface is satisfied by functions in following.

```
print(variables or string, ...)
```

show variable or string

```
input(variable)
```

put into variable by user

definition of function Definition of user function can be present. following show syntax of function

```
function function_name(list of variables)
    ....
    return(value)
    ....
end function
```

The return value of a function is defined to use return statement, and not execute all statements after that. Multiple 'return' statement with 'if' statement can be used.

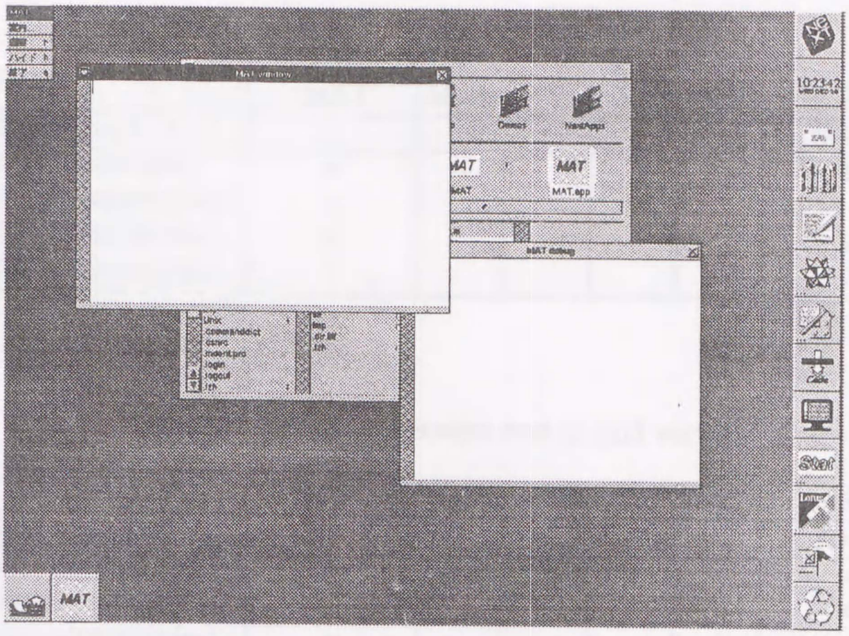


Figure 2.2: MAT system

We develop MAT system with MAT language (Figure 2.2). The system is developed in NeXT computer with Objective-C language, InterfaceBuilder, ProjectBuilder and class library 'Appkit' (Application Kits). To get information of these tools, refer to NEXTSTEP Developer's Library (1993).

2.4.2 Comparison with other language for matrix calculation

There are some famous packages or languages for matrix calculation. We show comparison of our MAT language with *Mathematica*, *SAS/IML* and *S*, in Table 2.2 to 2.6. A space mean $*$ in *Mathematica*, and a space mean . in our MAT language.

2.4.3 Example of MAT language

We show function FA (Factor analysis) in Figure 2.3 for example grogram of MAT language. An input variable `datacorr` is a correlation matrix of some multivariate data and an output of this function is a factor loading vectors. In executing this program, system shows a list of eigenvalues and requests to input the number of factors.

Table 2.2: Operators between matrix and matrix

	MAT	*Mathematica*	*SAS/IML*	*S*
Matrix product	.	.	*	%*%
Sum(elementwise)	+	+	+	+
Difference(elementwise)	−	−	−	−
Product(elementwise)	*	*	#	*
Quotient(elementwise)	/	/	/	/

Table 2.3: Operators between matrix and vector

	MAT	*Mathematica*	*SAS/IML*	*S*
Matrix product	.	.	*	%*%
Sum(elementwise)	+	+	×	×
Difference(elementwise)	−	−	×	×
Product(elementwise)	*	*	#	×
Quotient(elementwise)	/	/	×	×

Table 2.4: Operators between vector and vector

	MAT	*Mathematica*	*SAS/IML*	*S*
Inner product	.	.		%*%
Sum(elementwise)	+	+	+	+
Difference(elementwise)	−	−	−	−
Product(elementwise)	*	*	#	*
Quotient(elementwise)	/	/	/	/

Table 2.5: Operators between scalar and matrix, vector

	MAT	*Mathematica*	*SAS/IML*	*S*
Sum(elementwise)	+	+	+	+
Difference(elementwise)	−	−	−	−
Product(elementwise)	., *	*	*	*
Quotient(elementwise)	/	/	/	/

Table 2.6: Operators between scalar and scalar

	MAT	*Mathematica*	*SAS/IML*	*S*
Sum(elementwise)	+	+	+	+
Difference(elementwise)	−	−	−	−
Product(elementwise)	.*, *	*	*	*
Quotient(elementwise)	/	/	/	/

```
function FA(datacorr)
        n=columns(datacorr)
        cc=data_corr
        cc_inv=inverse(datacorr)
        eigen_cc=eigenvalues(datacorr)

        print("eigenvalues")
        for(i=1;i<=n;i=i+1)
                print(eigen_cc[i])
        end for
        print("input number fo factor ")
        input(m)

        diags=1-1/diag(cc_inv)
        do
                for(i=1;i<=n;i=i+1)
                        c[i,i]=diags[i]
                end for;

                eigenval=eigenvalues(c,m)
                eigenvec=eigenvectors(c,m)
                aHat=eigenvec*sqrt(eigenval)
        while(norm(diag(c)-diag(aHat aHat'))<0.00001)

        return(aHat);
end function
```

Figure 2.3: A program of FA (factor analysis) written in MAT language

2.5 Conclusions

In this chapter, we explained what is the OOP technique and its usefulness. And we defined Matrix classes and MAT language. Because of encapsulation of classes, we can use Matrix class to develop other systems. Then we use this class in following chapters.

We develop the MAT system in NeXT computer, because NEXTSTEP which is the only OS programmed using the OOP technique, in that time.

In following chapter, we define classes for statistical data to develop a data analysis system using the OOP techniques.

3 Classes for Data Analysis

3.1 Introduction

In chapter 2, We show that the OOP technique is useful technique for programming. We planed to develop a statistical software using the OOP technique. We use five basic concepts which are explained in chapter 2 as follows:

Classes: Statistical data are classified by means of tree structure based on the attributes of variables. After the classification, the appropriate method of analysis is determined for such data class.

Methods: For any statistical data class, methods are defined as a statistical analysis or a data manipulation method.

Polymorphism: For example, a method named 'regression' means regression analysis when the data set belongs to 'ContinuousVariable class' and it means quantification method I when the data set belongs to 'CategoricalVariable class'. So, the user can use our system with a little vocabulary and select a message consciously according to data's class.

Inheritance: For example, if a method 'printing basic statistics' is defined in a super class, every subclass of the class or its subclasses have the same method.

Encapsulation: Because it is easy to add some methods or to modify them, it is also easy to add new statistical analysis method or new class for statistical data.

In this chapter, we define various classes for statistical analysis, 'Value class' for basically data and 'VariableVector class' and 'Data class' for statistical data and 'Result class' for output of statistical analysis and other classes (Graphics class, Text class, Distribution Function class, etc.), but we show four mainly classes following.
Remark: There are two kinds of methods, one is a factory method (class method) to generate an instance, the other is an instance method to operate to the instance. Every class must have at least one factory method, but most of factory methods are 'new'. So that, in this thesis, we show classes and instance methods but omit factory methods.

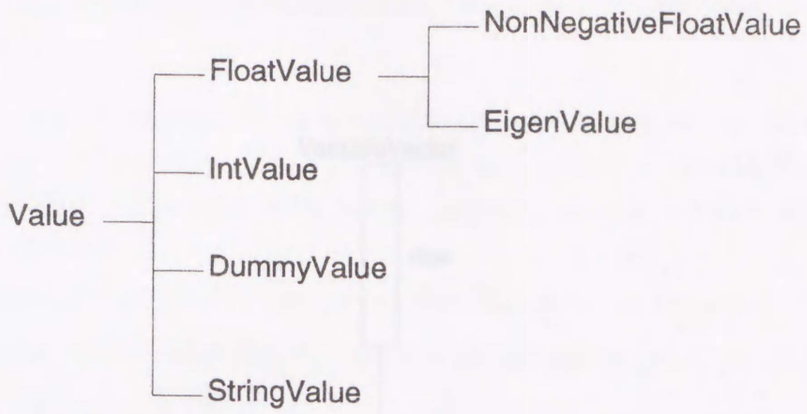


Figure 3.1: Value class and its subclasses

3.2 Value class

In the first, we treat the following values. The instance of this class has not only one value but also has a flag to indicate whether the value is missing value or not. We show a structure of this classes in Figure 3.1

FloatValue: It is an ordinal floating value.

NonNegativeFloatValue: It is used to a case weight variable.

EigenValue: This class is used in principal component analysis (PCA), factor analysis (FA) or other analysis using eigenvalue problem and has information of proportion.

IntValue: It is an ordinal integer value and used for the item-category variable.

DummyValue: It takes only zero or one value.

StringValue: It is an ordinal character string and used for name labels.

3.3 VariableVector class

Data of one variable. In Figure 3.2, we show a tree structure of VariableVector class and its subclasses. Because there are many other subclasses and we design new subclasses according to necessity, we show only six subclasses as follows.

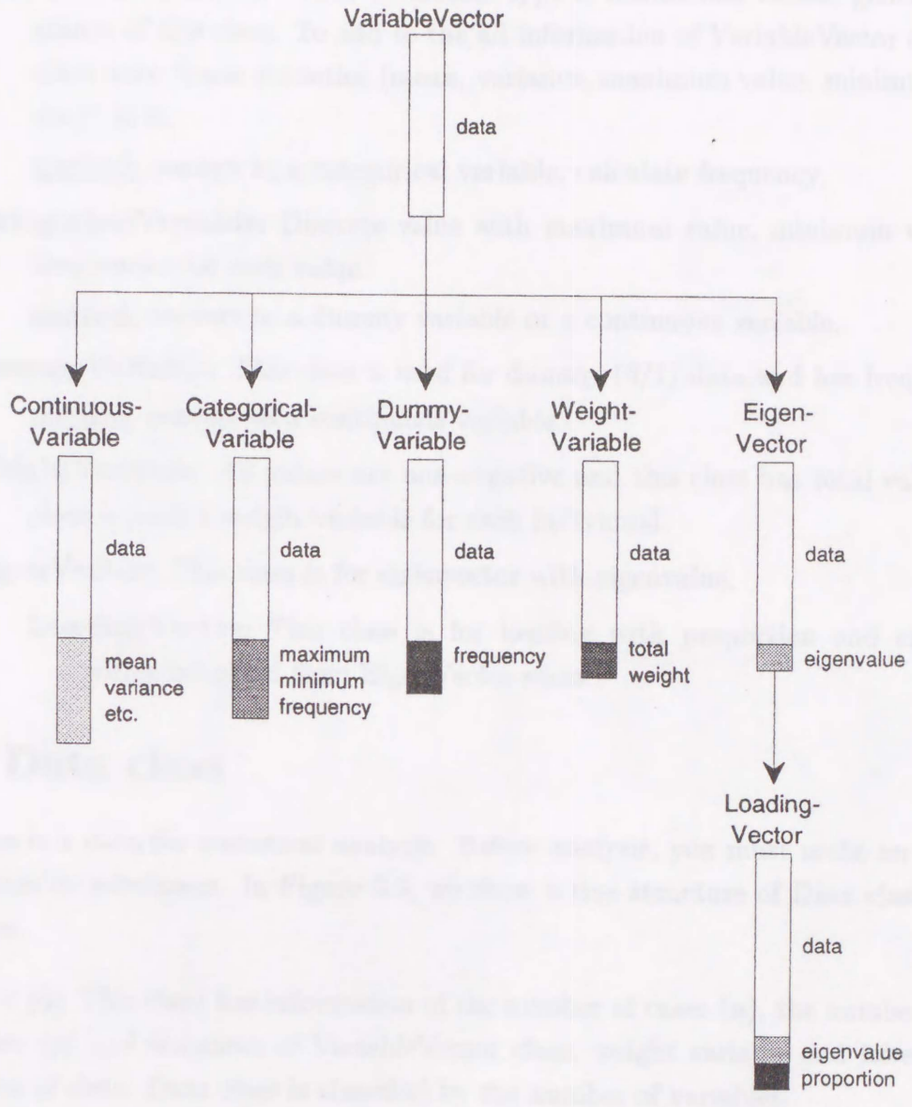


Figure 3.2: VairableVector class and its subclasses

VariableVector: It has data (value), the variable name and the number of cases and its subclasses are ContinuousVariable class, CategoricalVariable class, DummyVariable class and so on.

ContinuousVariable: When a variable type is continuous values, generate a instance of this class. To add to the all information of VariableVector class, this class have 'basic statistics (mean, variance, maximum value, minimum value, etc.)' in it.

method: convert to a categorical variable, calculate frequency.

CategoricalVariable: Discrete value with maximum value, minimum value and frequencies for each value.

method: convert to a dummy variable or a continuous variable.

DummyVariable: This class is used for dummy (0/1) data and has frequencies.

method: convert to a continuous variable.

WeightVaribale: All values are non-negative and this class has total value. This class is used a weight variable for each individual.

EigenVector: This class is for eigenvector with eigenvalue.

LoadingVector: This class is for loading with proportion and eigenvalue which inherited from EigenVector class.

3.4 Data class

This class is a data for statistical analysis. Before analysis, you must make an instance of this class or subclasses. In Figure 3.3, we show a tree structure of Data class and its subclasses.

Data($n \times p$): This class has information of the number of cases (n), the number of variables (p) and instances of VariableVector class, weight variable and labels of the cases of data. Data class is classified by the number of variables.

method: variable transformation, combine other data, split to other data, delete some variables, display basic statistics.

DataOneVariable($n \times 1$): The data belong to this class has only one variables. This class has information of the number of variables, weight variable and labels. different from VariableVector class.

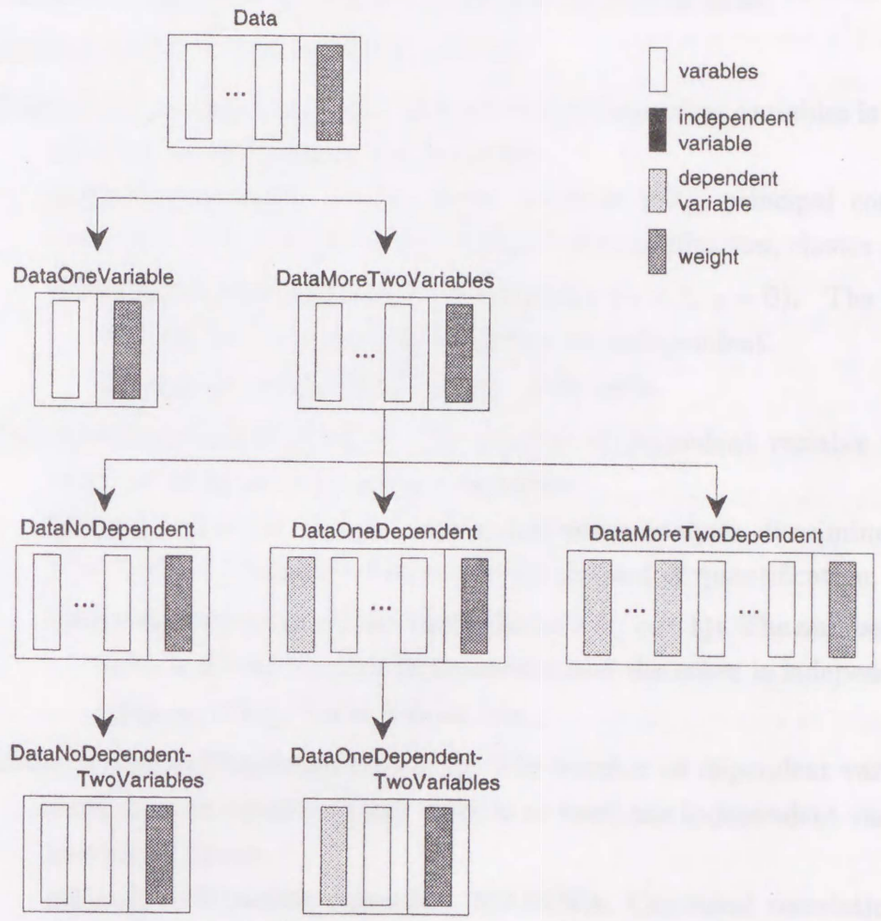


Figure 3.3: Data class and its subclasses

method: print frequency, graphics (histogram, probability-plot), test for normality, check of data.

DataMoreTwoVariables($n \times p$, $p \geq 2$)**:** The number of variables of data is more than or equal to 2. This class is classified by the number of dependent variables (q) and the number of independent variables as follows.

Remark: The number of independent variables must be one or more, but the number of dependent variables is allowed to be zero or more.

method: graphics (scatter matrix, scatter)

DataNoDependent ($q = 0$)**:** The number of dependent variables is 0, hence all variables are independent variables.

method: correlation matrix, factor analysis (FA), principal component analysis (PCA), Hayashi's third method of quantification, cluster analysis.

DataNoDependentMoreTwoVariables ($n \times 2$, $q = 0$)**:** The number of variables is 2, and both variables are independent.

method: χ^2 test of independent. cross table.

DataOneDependent ($q = 1$)**:** The number of dependent variable is 1 and other variables are independent variables.

method: regression analysis, logistic regression analysis, discriminant analysis, ANOVA, Hayashi's first or second method of quantification.

DataOneDependentTwoVariable ($n \times 2$, $q = 1$)**:** The number of variables is 2 (one variable is dependent and the other is independent).

method: t-test, Fisher's exact test.

DataMoreTwoDependent ($q \geq 2$)**:** The number of dependent variables is more than or equal to 2 and there is at least one independent variable as mentioned above.

method: multivariate-regression, MANOVA, Canonical correlation, path analysis, structural model.

Remark: Statistical analysis methods do not always belong to each class, but belong to its subclass which is classified by variable type. For example, the method 'Hayashi's third method of quantification' is belong to the subclass of DataNoDependent class in which all variables are dummy(0/1) variables, and the method 'PCA' is belong to one in which all variables are continuous variables.

3.5 Other classes

3.5.1 Result class

After an analysis we get an instance of this class. Every result of the analysis (for example score, loading graphics and so on) belong to this class and all of them are reusable. For example, after PCA, we get 'PCAresult' which is an instance of this class. The instance contains a principal component score, loading vectors, and so on. So we send message 'get' with 'score' to 'PCAresult', we can get the score which is an instance of Data class. If we want a scatter of the score, we send message 'scatter' to the score to get graphical result which is another instance of Result class.

3.5.2 Classes for display

There are two type classes for display. One is dependent on NeXT computer, the other is independent. The computer dependent classes are defined as subclass of certain class in class library 'Appkit'. TextWindow class is subclass of ScrollView and it is used to display some text and ViewWindow class is subclass of View and it is used to display some graphics. When we develop a system using our classes in other computers, it is necessary to implement TextWindow class and ViewWindow class for other computers.

The computer independent classes are TextManager class for displaying some text and ViewManager class for displaying some graphics. ViewManager class has subclass for every graphical statistical analysis method. For example, ScatterPlot class is used for 'scatter plot' and Histogram class is used for 'histogram'. These classes make a instance of computer dependent classes and use it. For example, when a instance of Histogram is received message to display, make a instance of ViewWindow and send message to draw a graphics (lines and letters). Hence computer dependent classes has methods for only basically drawing.

3.6 Examples

We consider the principal component analysis as an example of data analysis using our classes. Since most of the matrix calculation methods are available, most of statistical analysis using matrix calculations can be developed easily by using our classes.

This method belongs to subclass of DataNoDependent class ($n \times p$ data, $p \geq 2$ and all variables are independent) in which all variables are continuous variables.

(1) Construct a new matrix from the instance of Data class and new vectors from its mean and standard deviation (s.d.).

(2) Standardize the data matrix using a mean vector and a s.d. vector.

In step (2), use only matrix calculations to standardize data matrix. Most of matrix calculations are prepared as methods in Matrix class.

(3) Calculate eigenvalues and eigenvectors.

Output of the step (3) is an instance of Data class with EigenVector class. This is considered a subclass of VariableVector class with an eigenvalue. When the message 'print basic statistics' is sent to the instance, a table of eigenvalues and eigenvectors is displayed.

(4) Determine the number of components using the table and delete superfluous components. We obtained a factor loading matrix.

(5) Multiply the standardized data matrix by the factor loading matrix to get scores and generate an instance of Result class including both scores and loading.

In step (5), both scores and factor loadings are instances of Data class included in PCAresult.

(6) Send message 'print' and 'plot' to scores and loadings to print and display the scatter plot of scores and loadings.

Here, we obtain tables and graphics of the corresponding scores and loadings.

3.7 'Stat' system

We develop 'Stat' system using Data classes and Matrix class in NeXT computer with Objective-C language, InterfaceBuilder, ProjectBuilder and class library 'Appkit'.

3.7.1 Executing example

We treat the Principal Component Analysis for the executing example. The data is the amount of consumption of foods for person per one year in 1954 to 1956 (Wakimoto et al, 1992) (Table B.1).

In the 'File Viewer' of NeXT computer, we select our system 'Stat' and double click this icon. Then Stat system is started, the menu appears at the upper left corner, and the 'Main Window' appears at the center (Figure 3.4).

In the first, we must decide the data, so, we click 'de File' and 'open' commands in the menu. We select the data file and double click it.

Our system reads the file, and creates an instance, which belongs to the class of 'DataNoDependent'. The instance is registered in the Main Window and applicable methods to the instance are displayed in the right side in the window.

Now, we apply the method 'PCA' to the instance. We apply the PCA to the standardized data, so we click the 'correlation' button (Figure 3.5).

The system executes the PCA method, and creates the new instance 'PCA(Cor)', which belongs to the class of 'PCA Result'. It consists of nine objects of the third column in the window, and we can use two methods('Delete' or 'Display') to the instance.

We send a message 'Display' to the instance of 'PCA Result', then we get the default outputs, which are eigenvalues and their proportional values, scatter plot of the loadings and the PCA scores. In this case the first three eigenvalues exceed one, which is an average value of the proportion, then only these pairs are displayed (Figure 3.6).

The PCA score is a member belonging to the 'PCA Result', and it is an instance of 'DataNoDependent' class. So, we can apply the method 'Display' to this instances, then we can see their values. This result contains all components (Figure 3.7).

3.7.2 MAT in Stat

MAT language, in chapter 2, can be used in 'Stat' system. To use MAT language, we apply the method 'MAT' to the instance. We click 'MAT' in method area (Figure 3.8), then a 'MAT' window open (Figure 3.9).

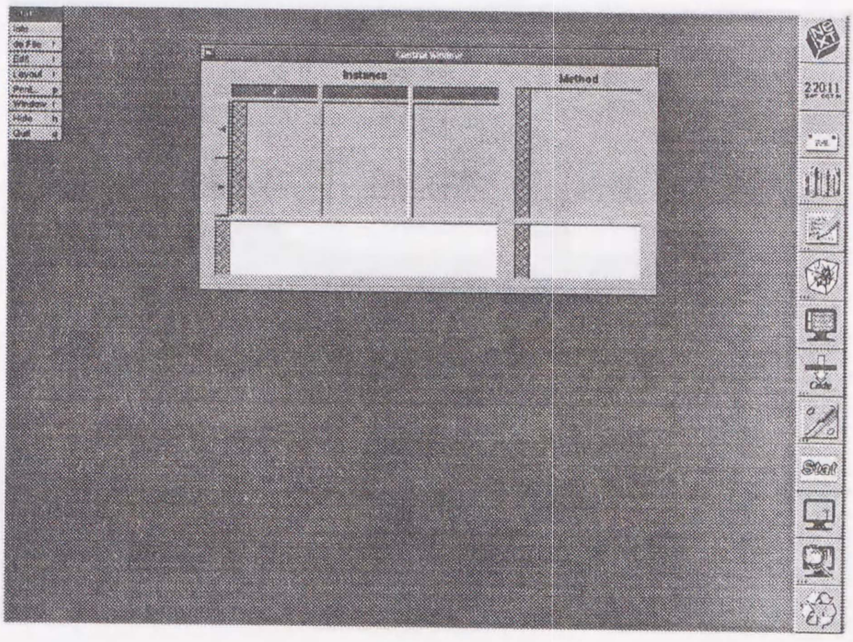


Figure 3.4: Main Window of Stat System

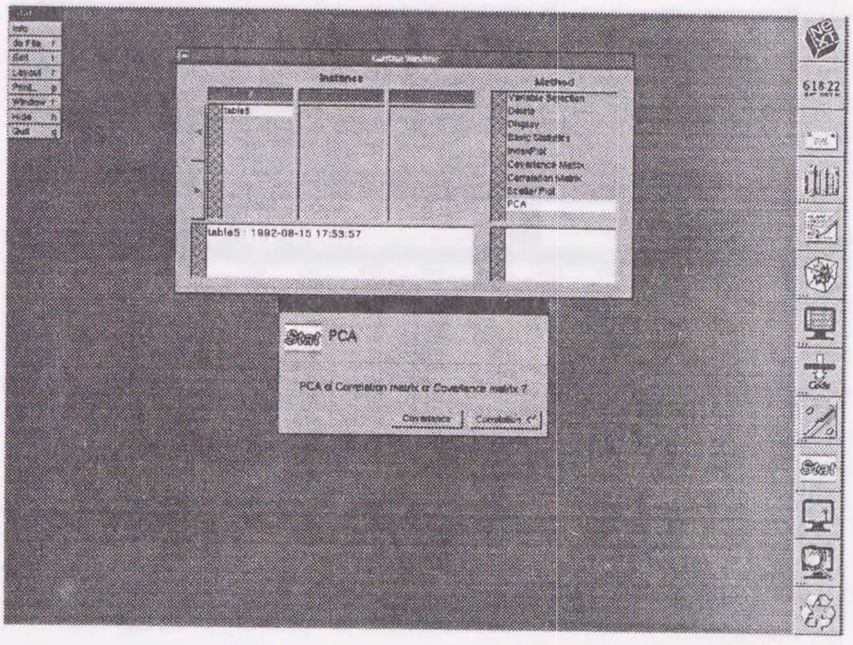


Figure 3.5: Selecting a PCA method

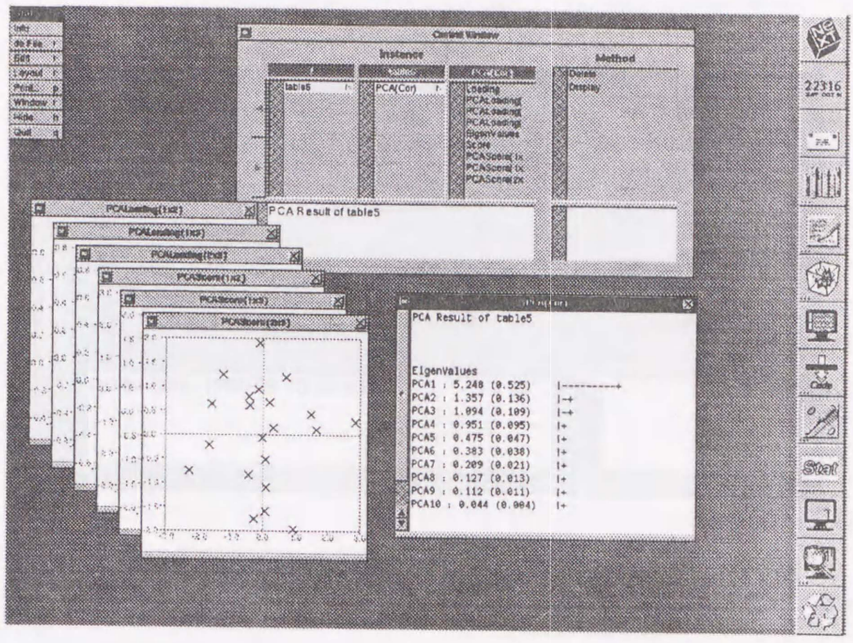


Figure 3.6: Results of PCA

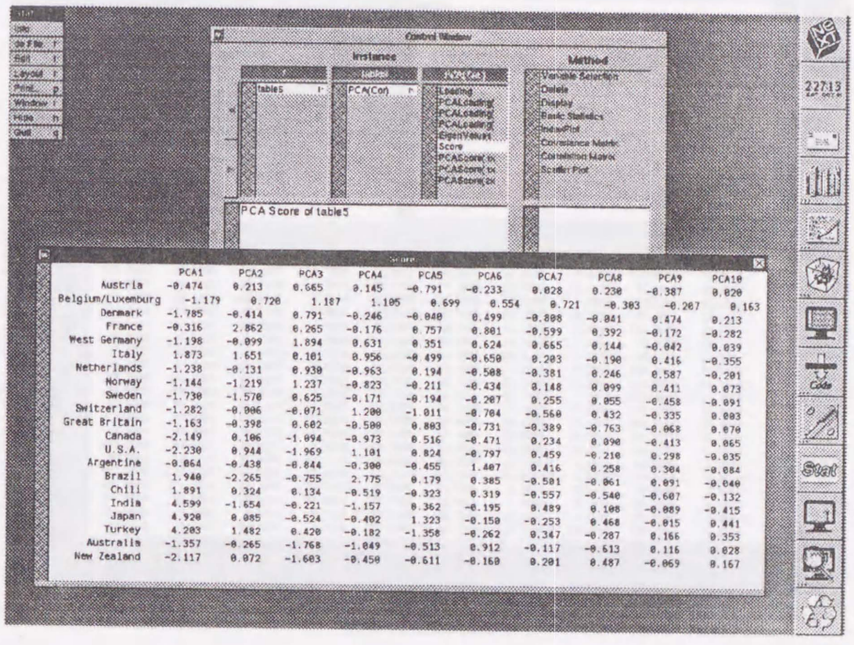


Figure 3.7: Displaying PCA Scores

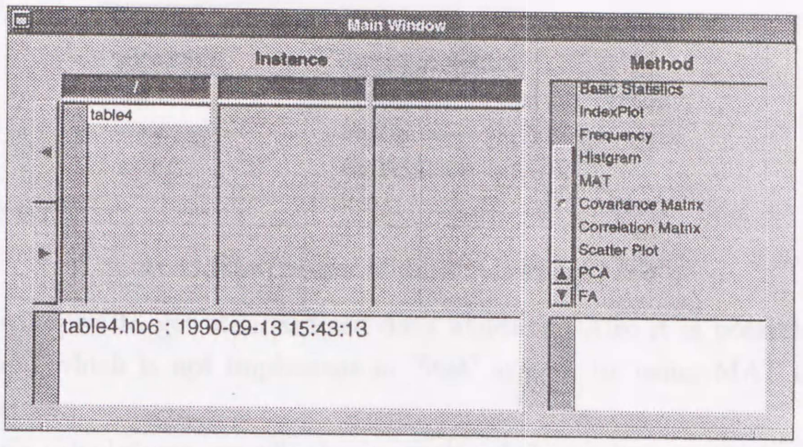


Figure 3.8: Main Window of 'Stat'

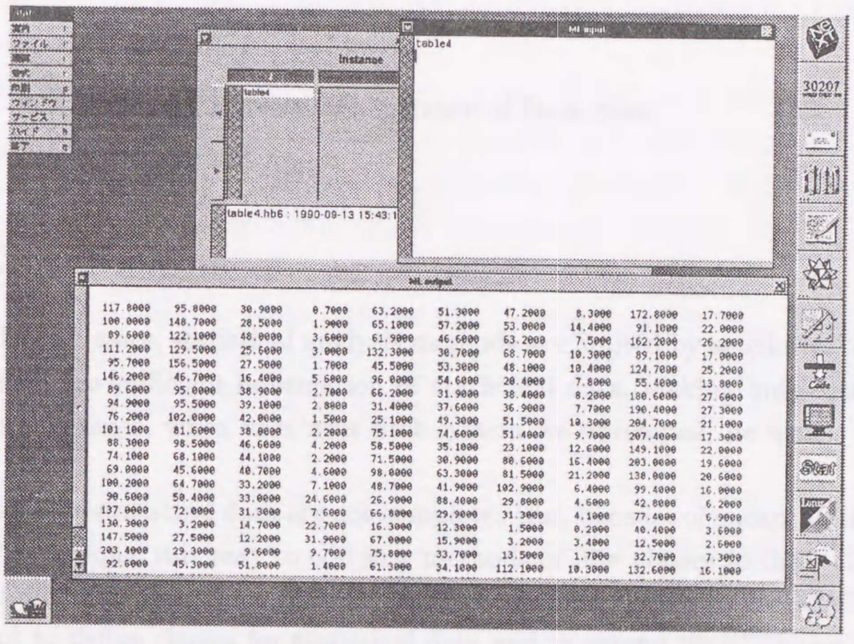


Figure 3.9: MAT Window of 'Stat'

When the MAT window start, following variable is already defined.

variable name	content
a name of data	data matrix
`average`	average vector
`sd`	standard deviation vector
`cov`	variance-covariance matrix
`corr`	correlation matrix

Therefore, we type

```
stddata=('name of data' -average)/sd
```

in the MAT windows to get standardized data `stddata`. Also it is possible to use new analysis method which is not implement in 'Stat' system by using MAT language and previous variables.

After matrix calculations, we will obtain results of the analysis method as some matrices. Thus, the MAT window has new function `data` to convert from a instance of Matrix class to a instance Data classes, because only Data classes have methods for graphical expression or statistical analysis.

Table 3.1: New matrix functions

Let A is matrices.

data(A)	convert to a instance of Data class

3.8 Conclusions

To use our Data classes, statistical analysis methods are limited by attributes of variables. Hence, if there are sufficient information of statistical data, making 'miss using' can be decease. In this sense, when user uses Stat system we developed, he makes little 'miss using'.

Our classes for statistical data are not complete. But, because of encapsulation concept of the OOP technique, it it easy to add new methods or new classes to these classes. And it is easy to add new statistical analysis methods to these classes to use MAT language. We interrupt to define classes for statistical data and to extend 'Stat' system for a time.

In following chapter, we design a knowledge base and a inference engine to develop the data analysis system with knowledge. Off course, we use our classes defined in previous chapter and this chapter to develop that system.

4 Knowledge Base and Inference Engine

4.1 Introduction

A knowledge base is a set of knowledge. An inference engine makes decision using knowledge in the knowledge base. In this thesis, knowledge is every thing what we consider when we use software for statistical analysis or how to use software.

Both a knowledge base and an inference engine are necessary to developing expert system. A quality of a knowledge base determines a quality of expert system and a power of an inference engine determines a power of expert system.

In this chapter, we discuss both a knowledge base and an inference engine using the OOP technique.

4.2 Inference Engine

There are three major models for implementation of an inference engine, "frame model", "network model" and "blackboard model".

On a frame model, every data is called frame. It has *slots* which keep a several properties of itself. When an inference engine makes some decision, it access slots of object and according to these properties and some knowledge in knowledge base. On a network model, every knowledge is called node and linked each other. According to links of knowledge, an inference engine makes decision. On a blackboard model, there are several inference engines. which have own knowledge base. Every inference engine is looking at 'blackboard' and if there is a question in blackboard and the engine has its solution, erase the question and write the solution to blackboard according to knowledge of it.

We propose a new model "Object-Oriented" model. A Data is defined as class, then it has *methods* which is statistical analysis method. From polymorphism concept of the OOP technique, there are methods which have same name but different result. So, when instance of Data class receive a message, it return a certain result according to own method without inference engines. In this sense, a method can be role of a knowledge.

An inference engine makes a decision sending message. The engines sends a message to instance of Data to get properties and makes a decision using own knowledge. Therefore,

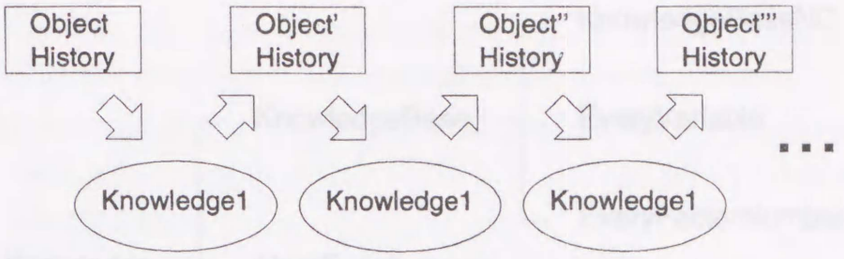


Figure 4.1: Concept of working of Knowledge class

if a knowledge is defined as class and has a method for inference, a knowledge can be used role of an inference engine. A knowledge base is set of knowledge and also it is a knowledge.

Our proposed system using this model. A knowledge has role of inference engines and sending message for inference, it makes a decision of what message send to data. So, we defined Knowledge class for knowledge and inference engines. Also, this idea is effective following sense.

- A knowledge base has to be made only once, since instance of class can be archive to files.
- A knowledge is separated from a statistical system, so a maintenance of knowledge can be held without maintenance of the system.
- And it is possible that several system use same knowledge.

4.3 Knowledge class

When a instance of Knowledge class or its subclass receive a message with data and instance to use as history, and the instance do something according to its own knowledge and return result and history which is added what to do in it. A knowledge means statistical analysis, decision making, transformation, or other. Figure 4.1 show a concept of working of Knowledge class.

Figure 4.2 show knowledge class and its main subclasses.

4.4 Subclasses of Knowledge class

KnowledgeBase This class has a sequence of instances of Knowledge class or subclass.

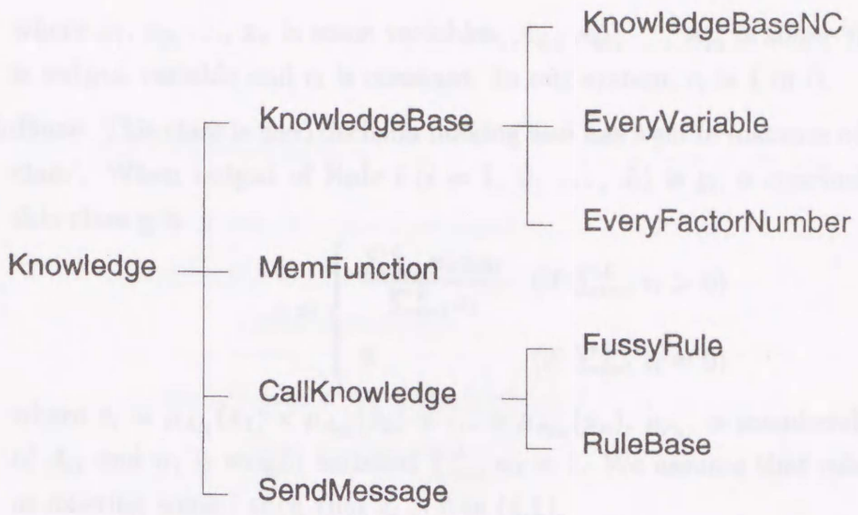


Figure 4.2: 'Knowledge class' and its subclasses

They receive message sequentially like Figure 4.1. At last, it returns a data which last instance returned.

KnowledgeBaseNC This class is subclass of 'KnowledgeBase class'. Difference between two class is a instance of this class returns a data which it received first.

EveryVariable This class is also subclass of 'KnowledgeBase class'. A instance of this class sending message with a every variable.

EveryFactorNumber This class is used for Principal Component Analysis (PCA), Factor Analysis (FA) or other using factor numbers. A instance of this class sending message with i (i is the number of factor, $i = 1, 2, \ldots, p$, p is a number of variables).

MemFunction This class is implementation of membership functions.

CallKnowledge This class has a set of instances of 'KnowledgeBase class' or subclass. They receive message message independently. It returns a set of data which every instance returned.

FussyRule This class is used decision making. Rule is following style :

Rule i: IF x_1 is A_{i1} and x_2 is A_{i2} and ... and x_{ni} is A_{ni} and then $y_i = c_i$

where $x_1, x_2, \ldots, x_n$ is some variables, $A_{i1}, A_{i2}, \ldots, A_{in}$ is some 'fuzzy set', y_i is output variable and c_i is constant. In our system, c_i is 1 or 0.

RuleBase This class is used decision making and has a set of instance of 'FussyRule class'. When output of Rule i ($i = 1, 2, \ldots, L$) is y_i, a conclusion value of this class y is

$$y = \begin{cases} \dfrac{\sum_{i=1}^{L} w_i z_i y_i}{\sum_{i=1}^{L} z_i} & (\text{if } \sum_{i=1}^{L} z_i > 0) \\ 0 & (\text{if } \sum_{i=1}^{L} z_i = 0) \end{cases} \tag{4.1}$$

where $z_i = \mu_{A_{i1}}(x_1) \times \mu_{A_{i2}}(x_2) \times \ldots \times \mu_{A_{in}}(x_n)$, $\mu_{A_{ij}}$ is membership function of A_{ij} and w_i is weight satisfied $\sum_{i=1}^{L} w_i = 1$. We assume that rules are made as existing some i such that $z_i > 0$ in (4.1).

This class returns one of signal 'GO', 'WAIT', 'STOP' as result.

SendMessage This class has a message to send to a data object for executing some analysis or getting some information of the object.

Assume that a RuleBase in sequence of a KnowledgeBase and the KnowledgeBase is belong to a CallKnowledge. When a RuleBase returns signal 'GO', the KnowledgeBase sent message to the following knowledge in its sequence of knowledge. When signal 'STOP', the KnowledgeBase is canceled and returns a signal 'STOP' as a result. And when signal 'WAIT', the KnowledgeBase returns a signal 'WAIT' as a result and waits to send message to the following knowledge until all KnowledgeBase in CallKnowledge return a signal 'STOP' or 'WAIT'. This design makes our system to return statistical analysis result at least one but not many.

4.5 Examples

For a example of expression using these classes of knowledge, we consider Principal Component Analysis (PCA) with standardized data. Figure 4.3 is a main KnowledgeBase. In this figure, a class name of knowledge is shown in parenthesis.

Figure 4.4 is detail knowledge with respect to the KnowledgeBase 'det-num-pc' for determining the number of factors. This is called its own knowledge with the number of factor i (i is the number of factor, $i = 1, 2, \ldots, p$, p is a number of variables). Figure 4.5 and 4.6 show membership functions 'mFEigenMore1' and 'mFcumPropMore80' where

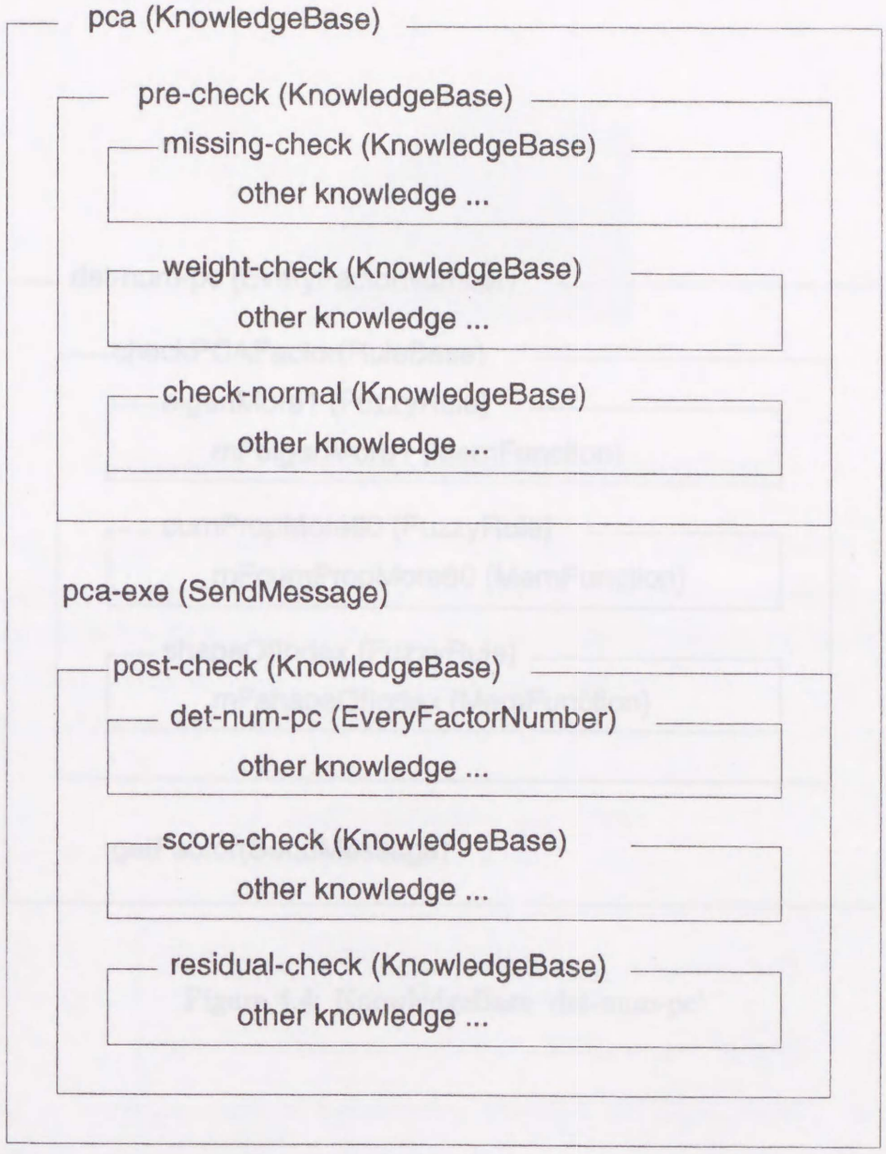


Figure 4.3: KnowledgeBase 'PCA'

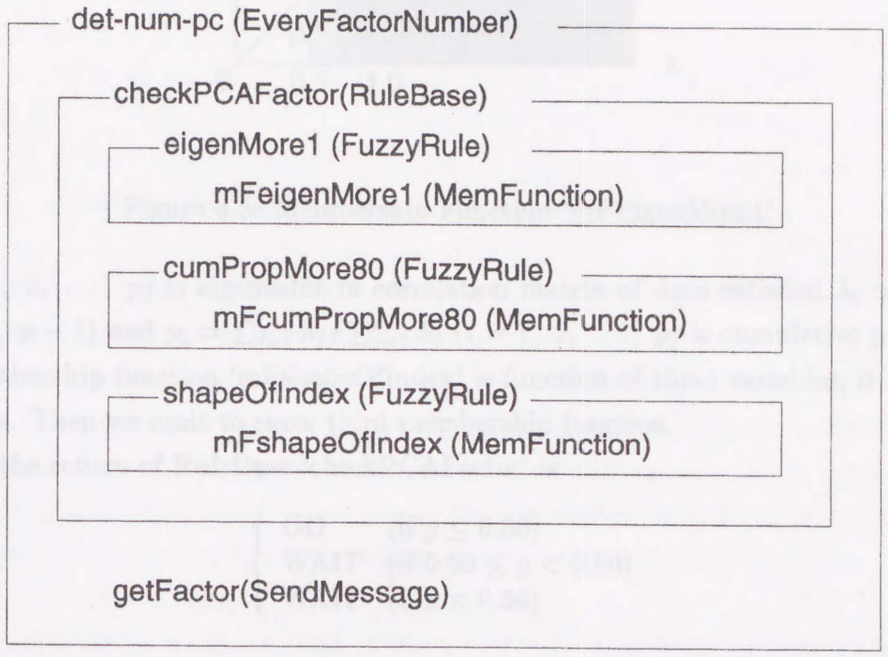


Figure 4.4: KnowledgeBase 'det-num-pc'

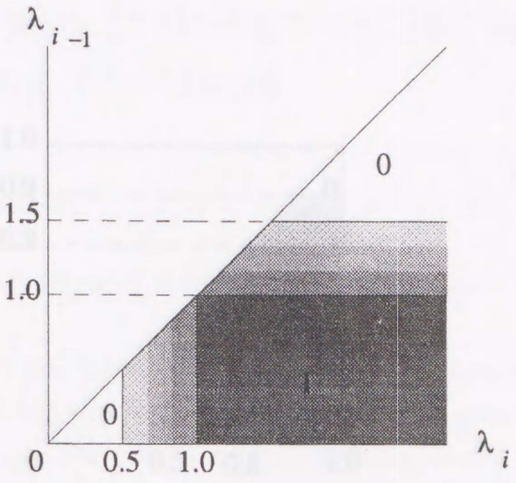


Figure 4.5: Membership Function 'mFEigenMore1'

λ_i $(i = 1, 2, \ldots, p)$ is eigenvalue of correlation matrix of data satisfied $\lambda_i > \lambda_{i+1} (i = 1, 2, \ldots, p-1)$ and $p_i = \sum_{j=1}^{i} \lambda_j / \sum_{j=1}^{p} \lambda_j$ $(i = 1, 2, \ldots, p)$ is cumulative proportion. As a membership function 'mFshapeOfIndex' is function of three variables, it is difficult to express. Then we omit to show third membership function.

Last, the return of RuleBase 'checkPCAFactor' is

$$\begin{cases} \text{GO} & (\text{if } y \leq 0.80) \\ \text{WAIT} & (\text{if } 0.50 \leq y < 0.80) \\ \text{WAIT} & (\text{if } y < 0.50) \end{cases}$$

where

$$y = \begin{cases} \dfrac{(0.4 z_1 y_1 + 0.3 z_2 y_2 + 0.4 z_3 y_3)}{(z_1 + z_2 + z_3)} & (\text{if } z_1 + z_2 + z_3 > 0) \\ 0 & (\text{if } z_1 + z_2 + z_3 = 0) \end{cases}$$

and z_1, z_2, z_3 is return value of membership function 'mFEigenMore1', 'mFcumPropMore80' and 'mFshapeOfIndex', $y_1 = y_2 = y_3 = 1$.

4.6 Conclusions

In this chapter, we proposed new model of a inference engine and a knowledge base. According to this design, we are developing a proto-type system in NeXT computer with

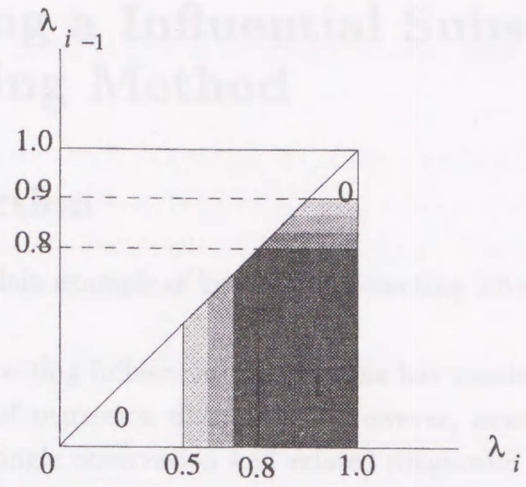


Figure 4.6: Membership Function 'mFcumPropMore80'

InterfaceBuilder, ProjectBuilder, Objective-C language and Appkit. The software consists of only principal component analysis, in now.

This proto-type software is used for specific statistical analysis and examine our system. After that, if we feel to need to modify our classes or system, we re-design classes, re-build knowledge and the system and examine again and again. At the same time, We add knowledge of other statistical analysis to this knowledge base. Because our knowledge base independent from system, it is also necessary to develop a software for making or maintenance knowledge base.

Mouse clicking by a user in 'Stat' system correspond to *send message* by knowledge class in new system. As next step, We have a plan to study how to get knowledge automatically under a man expert using the 'Stat' system.

5 Detecting a Influential Subset Using Clustering Method

5.1 Introduction

In this chapter, we explain example of knowledge, detecting influential observations using clustering method.

The problem of detecting influential observations has considered by a number of researchers in the field of regression diagnostics. However, most of them are concerned with the influence of single observation and related diagnostic measures are numerous. Although those single case diagnostic measures can be easily generalized to multiple case diagnostic measures, the practical application of detecting jointly influential subsets in still problematic because of the presence of the so-call "masking and swamping effects" as well as the prohibitive computation in view of large subsets involved. There have been proposed two major type of practical method for this problem. One is the method based on the cluster analysis with similarities defined by the modified hat matrix. This type was originally proposed by Gray and Ling (1984) and was modified by Hadi (1985). The other is the method based on robust regression such as Rousseeuw(1984, 1990)'s method based on his least median of squares regression. Both are useful for detecting influential subsets in regression analysis, but the ideas behind them can not be applied directly to other multivariate methods.

In sections 5.2, we propose a method of detecting influential subsets using clustering method in multivariate methods where the empirical influence function (EIF) are available. Regarding this topic Tanaka, Castaño-Tostado and Odaka (1990) among other have recommended and illustrated to use principal component analysis (PCA) or canonical variate analysis (CVA) for detecting individuals with relatively large EIF vectors and similar influence patterns, on the basis of the additivity property of EIF.

In section 5.3, we propose a clustering method using a linked lines rotation graphics (LLRG) which is a method of dynamic graphics. This graphics method is proposed by Wakimoto(1993) and based on the idea of the constellation plot proposed by Wakimoto and Taguri(1978). These graphics consist of the linked lines chart based on the p-variate data and the locus which is plotted by rotating the end point of this chart. Observing this graphics, we can find some groups of observations which have similarities.

5.2 Influential subset

5.2.1 Influence functions for single-case and multiple-case diagnostics

To evaluate the influence of each individual, we make use of the idea of influence function or influence curve (Hample, 1974). Let $\theta = \theta(F)$ be a parameter which is expressed as a functional of the cumulative distribution function(cdf) F. Then the theoretical influence function $TIF(x;\theta)$ at x is defined as

$$TIF(x;\theta) \equiv \theta^{(1)} = \lim_{\varepsilon \to 0}(\tilde{\theta} - \theta)/\varepsilon, \tag{5.1}$$

where $\tilde{\theta} = \theta(\tilde{F})$, $\tilde{F} = (1-\varepsilon)F + \varepsilon\delta_x$ is a perturbed cdf of δ_x and δ_x is the cdf of a unit point mass at x. When $/theta$ is expanded in a power series of ε as

$$\theta(\varepsilon) = \theta((1-\varepsilon)F + \varepsilon\delta_x) = \theta + \varepsilon\theta^{(1)} + O(\varepsilon^2), \tag{5.2}$$

the TIF is obtained as the coefficient $\theta(1)$ of the first order term of ε, or equivalently the first differential coefficient of $\theta(\varepsilon)$ at $\varepsilon = 0$.

The empirical influence function (EIF) is obtained by replacing the empirical cdf $\hat{F}$ for F in the definition of TIF (5.1). There we usually focus our interest on the values at $x = x_i$ $(i = 1, \ldots, n)$ given by

$$EIF(x_i;\hat{\theta}) \equiv \hat{\theta}_i^{(1)} = \lim_{\varepsilon \to 0}[\theta((1-\varepsilon)\hat{F} + \varepsilon\delta_{x_i}) - \theta(\hat{F})]/\varepsilon, \tag{5.3}$$

where the estimate $\hat{\theta}$ is defined by $\hat{\theta} = \theta(\hat{F})$.

The sample influence function (SIF) is obtained by taking $\varepsilon = -1/(n-1)$ and omitting the limit (5.1). The SIF is then

$$SIF(x_i;\hat{\theta}) = -(n-1)(\hat{\theta}_{(i)} - \theta), \tag{5.4}$$

where the subscript (i) denotes the omission of the i-th individual.

When we consider the perturbation on F as explained in the above, the covariance matrix corresponding to this perturbation is expressed as

$$\Sigma(\varepsilon) = \Sigma + \varepsilon\Sigma^{(1)} + O(\varepsilon^2), \ \Sigma^{(1)} = (x-\mu)(x-\mu)^T - \Sigma, \tag{5.5}$$

where μ and Σ are unperturbed mean vector and covariance matrix, and the influence function for Σ is given as $\Sigma^{(1)}$.

For evaluating the influence of multiple individuals it is convenient to generalize the influence function in the following manner (Tanaka, 1992). Let us introduce a perturbation on the cdf F as

$$F \to \tilde{F} = (1-\varepsilon)F + \varepsilon G, \tag{5.6}$$

where $G = k^{-1}\sum_{i\in A}\delta_{x_i}$, and $A = \{i_1, , i_2, \ldots, i_k\}$ is an index subset of individuals, and define the generalized influence function for A as

$$TIF(A;\theta) \equiv \theta_A^{(1)} = \lim_{\varepsilon\to 0}(\theta(\tilde{F}) - \theta(F))/\varepsilon. \tag{5.7}$$

Then it is easy to verify that the following relation holds:

$$TIF(A;\theta) \equiv \theta_A^{(1)} = \frac{1}{k}\sum_{i\in A}\theta_i^{(1)} = \frac{1}{k}\sum_{i\in A} TIF(x_i;\theta). \tag{5.8}$$

Thus, the generalized influence function for a given set A is obtained as the average of the ordinary influence function for each member of the set. Similar relation holds also for the EIF.

Let $\hat{F}_{(A)}$ be the empirical cdf based on sample with a subset A omitted. Then

$$\hat{F}_{(A)} = (1 + \frac{k}{n-k})\hat{F} - \frac{k}{n-k}\cdot\frac{1}{k}\sum_{i\in A}\delta_{x_i} \tag{5.9}$$

holds. this implies the perturbation of $\hat{F}$ with $\varepsilon = -k/(n-k)$. Using the approximation up to the first order term of ε, we have

$$\hat{F}_{(A)} = \hat{\theta}_{(A)} \cong \hat{\theta} - \frac{k}{n-k}EIF(A;\hat{\theta}) = \hat{\theta} - (n-k)^{-1}\sum_{i\in A} EIF(x_i;\hat{\theta}). \tag{5.10}$$

In the above equation the two estimates $\hat{\theta}$ and $\hat{\theta}_(A)$ are based on n and $(n-k)$ individuals, respectively. The influence of a set of individual belonging to the set. Therefore, from the right-hand side of (5.10), we should search for the individuals which are relatively influential individually and also have similar influence patterns with each other. For this purpose we can apply PCA, CVA and cluster analysis to obtain ordinary EIF.

5.2.2 Influence measures

Now we consider the influence due to the perturbation scheme (5.6) with parameter ε. The generalized EIF $\hat{\theta}_A^{(1)}$ can be used for computing the changes of estimate $\hat{\theta}$ and other results. But, since the results and their change are in general vector-valued, they should be summarized into scalar measures from some specified aspects of influence for

convenience to evaluate the amount of influence. though the basic idea is common for any multivariate method where the EIF can be evaluated, we concentrate on the case of maximum likelihood factor analysis (MLFA) for illustration assuming an ordinary factor analysis model with multivariate normality, and consider three aspects of influence such as the influence on the estimate $\hat{\Delta}$ for the unique variance diagonal matrix, on its estimate precision $\hat{\Psi}$ and on the goodness of fit statistics X^2. Note that, in MLFA the influence function $\hat{\Delta}$ and $T^{*(1)}$ are available for the unique variance matrices in the so-called common factor decomposition $\Sigma = T^* + \Delta$, $T^* + LL^T$ (L: factor loading matrix) (see Tanaka and Odaka, 1989b; Tanaka, Castaño-Tostado and Odaka, 1990).

1. Influence on the estimate $\hat{\Delta}$: Using EIF $\hat{\Delta}_i^{(1)}$, $\hat{\Delta}_A^{(1)}$ is obtained as $k^{-1}\sum_{i\in A}\hat{\Delta}_i^{(1)}$. The change of the estimate $\delta\hat{d} = \varepsilon \cdot k^{-1}\sum_{i\in A}\hat{d}_i^{(1)}$, where $\hat{d} = (\hat{\Delta}_1^{(1)}1,\ \ldots,\ \hat{\Delta}_p^{(1)}p)^T$, is summarized into

$$D_\varepsilon = [\delta\hat{d}]^T\hat{\Psi}^{-1}[\delta\hat{d}] \tag{5.11}$$

where $\hat{\Psi}$ is the estimated asymptotic covariance matrix for $\hat{d}$, which is given by

$$\Psi = 2n^{-1}\hat{\Xi},\ \hat{\Xi} = (\hat{\xi}_{jl}),\ \hat{\xi}_{jl} = \hat{\phi}_{jl}^2,\ \phi = (\hat{\phi}_{jl}) = \hat{\Delta}^{-1} - \hat{\Delta}^{-1/2}V_1V_1^T\hat{\Delta}^{-1/2}, \tag{5.12}$$

where the column vector of $V_1 = (v_1,\ v_2,\ \ldots,\ v_q)$ are the eigenvectors associated with the largest q eigenvalues of $\hat{\Delta}^{-1/2}(S - \hat{\Delta})\hat{\Delta}^{-1/2}$, and S is the maximum likelihood estimate of Σ.

2. Influence on the precision of $\hat{\Delta}$: We consider the influence on the estimated asymptotic covariance matrix $\hat{\Psi}$ given by (5.12), i.e., $\hat{\Psi} \to \hat{\Psi} + \delta\hat{\Psi}$. Among quantities in the right hand side of the last equation of (5.12), the first order differential coefficient (or influence function) of $\hat{\Delta}$ is derived by Tanaka and Odaka (1989b) and that of $V_1V_1^T$ can be obtained by using the perturbation theory of eigenvalue problem (Tanaka, 1988). Thus, using those $\hat{\Delta}^{(1)}$ and $(V_1V_1^T)^{(1)}$, we have linear approximation as

$$\hat{\Phi}_\varepsilon \cong \tilde{\Phi}_\varepsilon = \hat{\Phi} + \varepsilon\hat{\Phi}_A^{(1)}. \tag{5.13}$$

Applying this approximation to the least equation in (5.12) we can evaluate an approximate $\tilde{\Phi}_\varepsilon$ and define the *COVRATIO*-like measure

$$\widehat{CVR}_\varepsilon = |\tilde{\Phi}_\varepsilon|/|\hat{\Phi}| \tag{5.14}$$

3. Influence on the goodness of fit: The likelihood ratio test statistic for the goodness of fit is given by

$$X^2 = n\log|\hat{\Delta} + \hat{T}^*| - n\log|S| \tag{5.15}$$

The test statistics X^2 follows asymptotically a chi-square distribution with $(1/2)[(p-q)^2-(p+q)]$ degrees of freedom, where p and q denote the number of variables and factors. respectively, under the null hypothesis that given model fits to the data. An approximate $\tilde{X}_\varepsilon^2$ is obtained by substituting linear approximations $\tilde{\Delta}_\varepsilon$, $\tilde{T}_\varepsilon^*$ and $\tilde{S}_\varepsilon$ into their counterparts in (5.15), receptively.

5.3 The clustering method by using linked line rotation graphics

This graphics method has flowing feature.

- A height of each circle show a average of object.
- According to a shape of path, it is possible to make cluster

We show a process of construction of LLRG. we denote the p-variate data of size n by $X_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$, $i = 1,\ 2, \ldots\ , n$. if p is odd, we can add a new variable which has same values to every individual, we suppose that p is even without loss of generality. Also, if there is exist some j such that $x_{ij} < 0$, we can replace x_{ij} by $x_i - \min_j x_{ij}$ $(j = 1,\ 2, \ldots,\ p)$, we suppose that $x_{ij} \geq 0$ $(i = 1,\ 2, \ldots,\ n,\ j = 1,\ 2, \ldots,\ p)$ without loss of generality.

Step 1 Suppose that we have three axes u, v and w intersecting prependicuearly with each other at the origin O in a three dimensional Euclidean space. Let us ensodu a straight line OL which intersects w at O with angle θ $(0 < \theta < \pi/2)$ and rotate OL around axes w, then we have a cone as shown in Figure 5.1. For i-th individual X_i, we plot the points $q_{i1},\ q_{i2}, \ldots, q_{ip}$ such that

$$x_{ij} = \overline{Oq_{ij}} (j = 1, 2, \ldots, p), \tag{5.16}$$

where

$$\begin{aligned}\overline{OR} = \max(&x_{11}\cos\theta,\ \ldots,\ x_{1p}\cos\theta,\ x_{21}\cos\theta,\ \ldots,\ x_{2p}\cos\theta,\\ &\ldots,\ x_{n1}\cos\theta,\ \ldots\ x_{np}\cos\theta)\end{aligned}$$

and

$$\angle b_0Ob_1 = \frac{\pi}{p+2},\ \angle b_0Ob_2 = \frac{2\pi}{p+2}, \ldots, \angle b_0Ob_{\frac{p}{2}} = \frac{p\pi}{2(p+2)},$$
$$\angle b_0Ob_{\frac{p}{2}+1} = \frac{\pi}{p+2} + \pi,\ \angle b_0Ob_{\frac{p}{2}+2} = \frac{2\pi}{p+2} + \pi, \ldots, \angle b_0Ob_p = \frac{p\pi}{2(p+2)} + \pi.$$

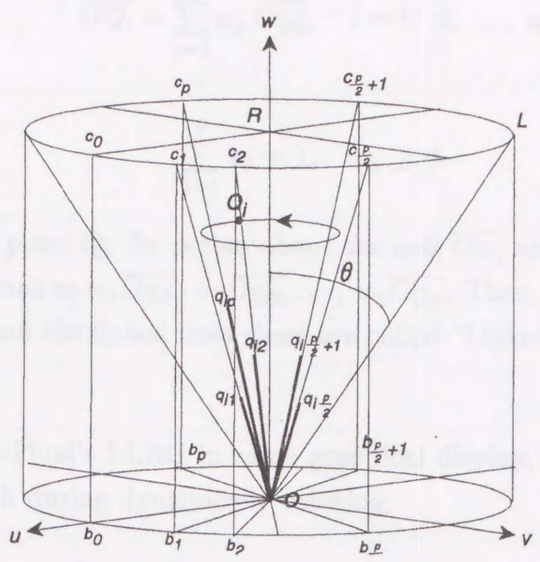


Figure 5.1: Construction of the rotation graphics

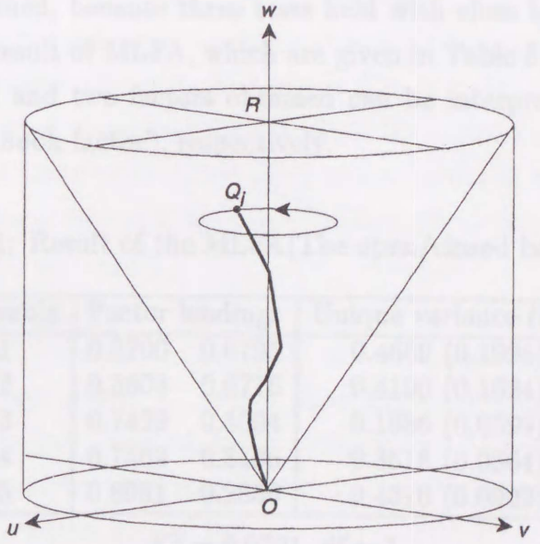


Figure 5.2: Linked lines rotation graphics (one individual)

Step 2 Let Q_i be a weighted gravity of the points $q_{i1}, \ldots, q_{ip}$. Then we have

$$\overrightarrow{OQ_i} = \sum_{j=1}^{p} w_j \overrightarrow{Oq_{ij}}, \quad i = 1, 2, \ldots, n, \tag{5.17}$$

where

$$\sum_{j=1}^{p} w_j = 1, \quad w_j \geq 0.$$

Step 3 We rotate the point Q_i 2π radian about the axis $\overline{Ow}$, and we plot a linked chart of lines of size p such as $w_1\overline{Oq_{i1}}$, $w_2\overline{Oq_{i2}}, \ldots, w_p\overline{Oq_{ip}}$. Then, as in Figure 5.2, a locus of the point Q_i and the linked lines chart are called "Linked lines rotation graphics (LLRG)".

To observe all individual's LLRG in same graphical display, we can find individuals which have similar path during dynamically rotating.

5.4 Example

Tanaka and Odaka (1989) and Tanaka et al. (1990) applied MLFA and their sensitivity analysis procedure to the open/close book data (Mardia, Kent and Bibby, 1979) (B.2). The data consist of $n = 88$ individuals (student) and $p = 5$ variables (tests), and a two-factor model was assumed, because three tests held with close book and two tests held with open book. The result of MLFA, which are given in Table 5.1, show that the model fits the data very well and two factors obtained can be interpreted as the "open book factor" and the "close book factor", respectively.

Table 5.1: Result of the MLFA(The open/closed book data)

Variable	Factor loadings		Unique variance (SE)
1	0.2700	0.6791	0.4659 (0.1995)
2	0.3603	0.6716	0.4190 (0.1624)
3	0.7429	0.5094	0.1886 (0.0599)
4	0.7403	0.3166	0.3518 (0.0864)
5	0.6981	0.2856	0.4310 (0.0902)

$X^2 = 0.0791$, df $= 1$

(SE: standard error, df: degree of freedom)

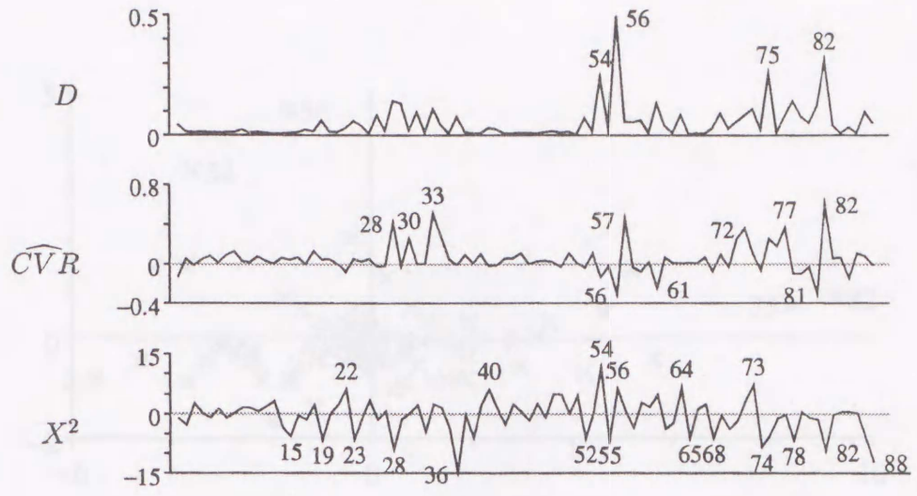


Figure 5.3: Index plots of each influence measures

To investigate the influence of each individual they calculated the EIF for $\hat{\Delta}$ and $\hat{T}^*$ and summarized them into three measures D_i, $\widehat{CVR}_i$ and X^2. The three measure correspond to defined (5.11), (5.14) and (5.15), as single-case diagnostic measures for the i-th individual. The index plot of three measures are given in Figure. 5.3. Also the applied PCA to EIF for $\hat{\Delta}$. Figure 5.4 shows the scatter diagram of the first two principal components, where the two principal components explain about 85.59% of total variance. From those result they have suggested that subset {75, 82} and {54, 56} form influential sets of individuals and that these two subsets affect the result in quite different manner, i.e., the sample without the former subset gives worse results in both of the measures for the precision and for the goodness of fit, while the sample without the latter subset gives better result in the both measure.

Now let us apply LLRG. Figure 5.5 show the LLRG of all data. There are 88 circles and paths in this LLRG. To observe the LLRG of every individual we found two subsets from this data. First, individuals 75 and 82 have large circle and their paths run right side of all paths (Figure 5.6). Second, individuals 54 and 54 have small and high circle and their paths trough is almost straight (Figure 5.7). Both subsets is found as influential sets of individuals from scatter plot of PCA.

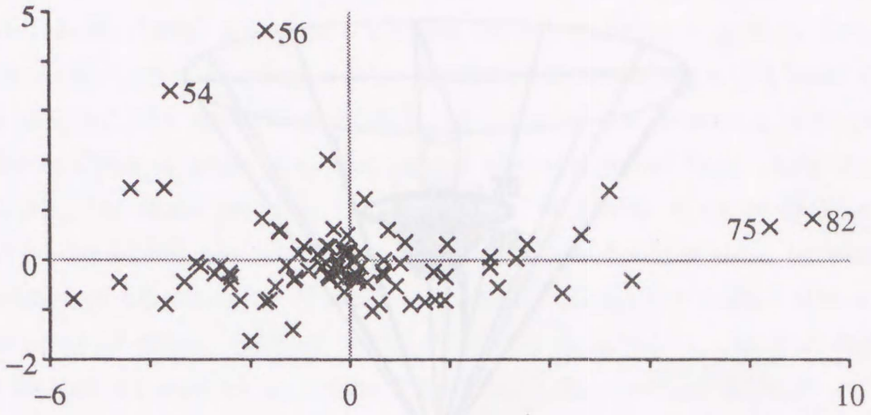


Figure 5.4: Scatter diagram of the first two PC's (EIF)

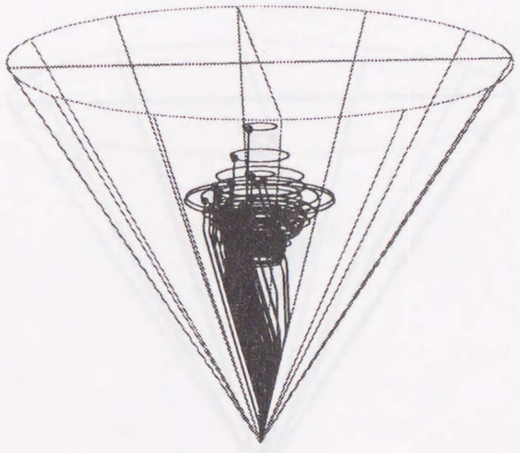


Figure 5.5: LLRG of EIF

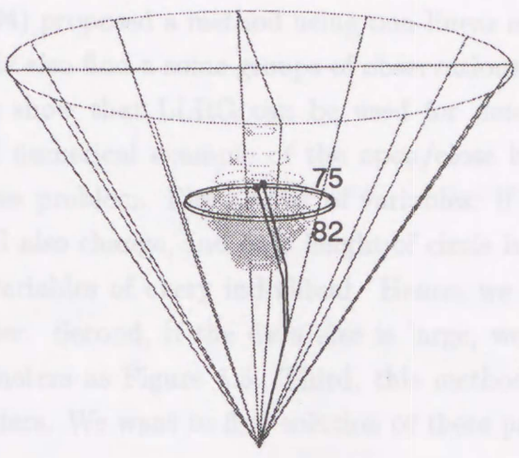


Figure 5.6: Influential individuals {75, 82}

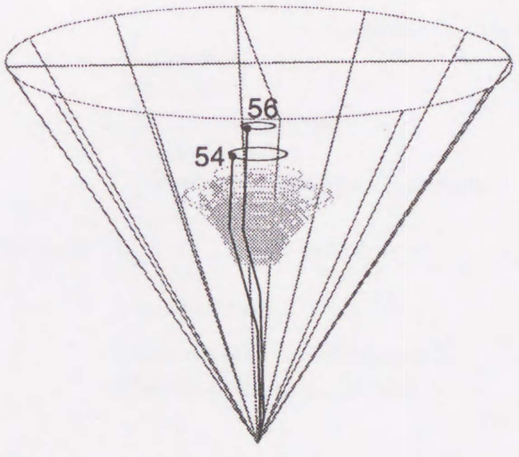


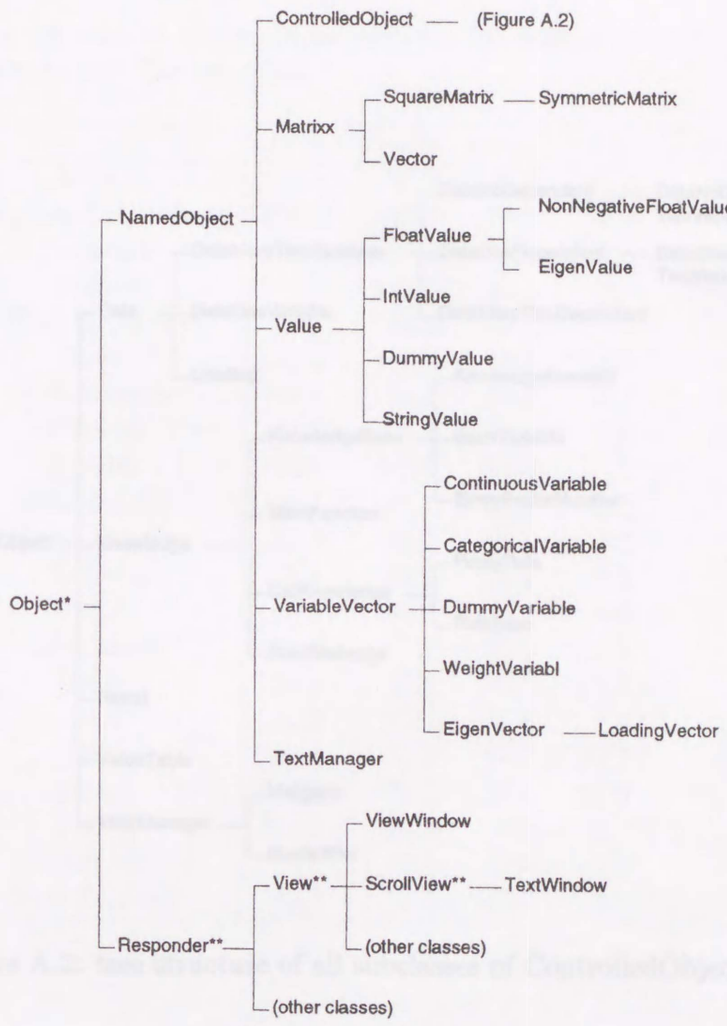
Figure 5.7: Influential individuals {54, 56}

5.5 Conclusions

Several methods to detecting influential subset are proposed. For instance, Fukumori, Yanagi and Tanaka (1994) proposed a method using non-linear mapping. The non-linear mapping method's aim is also find a some groups of observations which have similarities.

In this chapter, We show that LLRG can be used for detecting influential subset through the analysis of numerical example of the open/close book data using MLFA. But, the LLRG has three problem. First, order of variables: if order change, path and circle's size of the LLRG also change, and only height of circle is same because of height mean a average of all variables of every individual. Hence, we must make a algorithm to find the optimal order. Second, if the data size is large, we can find outlier but it is difficult to find all clusters as Figure 5.5. Third, this method depends on subjective judgement to make clusters. We want to find solution of these problem.

Appendix A Tree Structure of All Classes



* Class defined in Objective-C

** Classes defined in Appkit

Figure A.1: tree structure of all classes

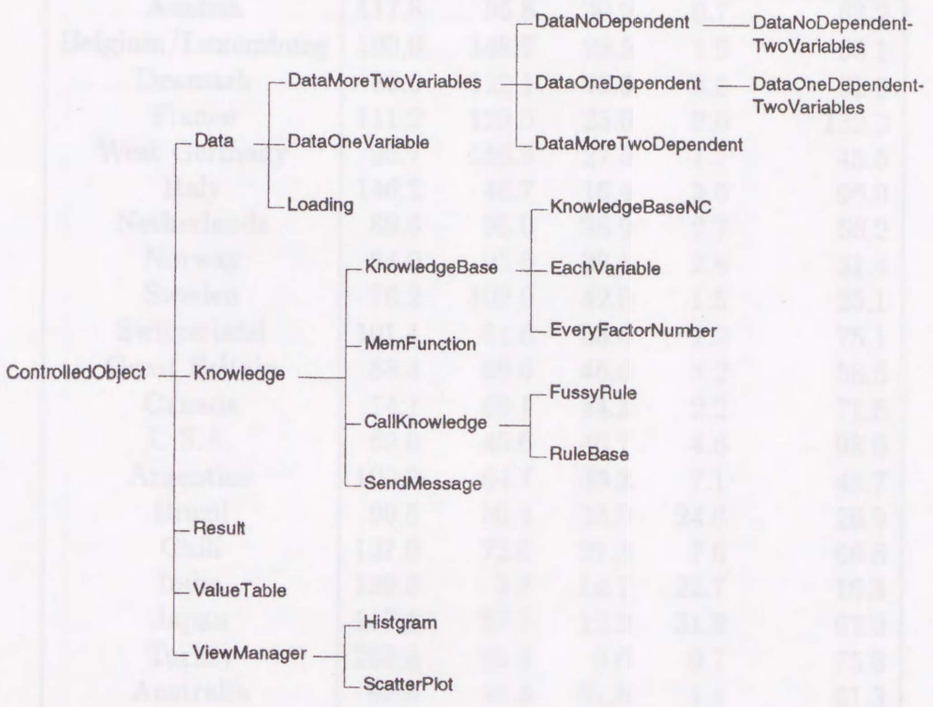


Figure A.2: tree structure of all subclasses of ControlledObject Class

Appendix B Data

Table B.1: The amount of consumption of foods for person per one year (1954–1956) (Wakimoto, Tarumi and Tanaka, 1992)

	corn	potato	sugar	pulse	vegetables
Austria	117.8	95.8	30.9	0.7	63.2
Belgium/Luxemburg	100.0	148.7	28.5	1.9	65.1
Denmark	89.6	122.1	48.0	3.6	61.9
France	111.2	129.5	25.6	3.0	132.3
West Germany	95.7	156.5	27.5	1.7	45.5
Italy	146.2	46.7	16.4	5.6	96.0
Netherlands	89.6	95.0	38.9	2.7	66.2
Norway	94.9	95.5	39.1	2.8	31.4
Sweden	76.2	102.0	42.0	1.5	25.1
Switzerland	101.1	81.6	38.0	2.2	75.1
Great Britain	88.3	98.5	46.6	4.2	58.5
Canada	74.1	68.1	44.1	2.2	71.5
U.S.A.	69.0	45.6	40.7	4.6	98.0
Argentine	100.2	64.7	33.2	7.1	48.7
Brazil	90.6	50.4	33.0	24.6	26.9
Chili	137.0	72.0	31.3	7.6	66.8
India	130.3	3.2	14.7	22.7	16.3
Japan	147.5	27.5	12.2	31.9	67.0
Turkey	203.4	29.3	9.6	9.7	75.8
Australia	92.6	45.3	51.8	1.4	61.3
New Zealand	86.4	47.2	42.9	1.5	72.8

Table B.1: The amount of consumption of foods for person per one year (1954–1956) (Wakimoto, Tarumi and Tanaka, 1992) (Cont.)

	fruits	meat	egg	dairy product	oils and fats
Austria	51.3	47.2	8.3	172.8	17.7
Belgium/Luxemburg	57.2	53.0	14.4	91.1	22.0
Denmark	46.6	63.2	7.5	162.2	26.2
France	30.7	68.7	10.3	89.1	17.0
West Germany	53.3	48.1	10.4	124.7	25.2
Italy	54.6	20.4	7.8	55.4	13.8
Netherlands	31.9	38.4	8.2	180.6	27.6
Norway	37.6	36.9	7.7	190.4	27.3
Sweden	49.3	51.5	10.3	204.7	21.1
Switzerland	74.6	51.4	9.7	207.4	17.3
Great Britain	35.1	23.1	12.6	149.1	22.0
Canada	30.9	80.6	16.4	203.0	19.6
U.S.A.	63.3	81.5	21.2	138.0	20.6
Argentine	41.9	102.9	6.4	99.4	16.0
Brazil	88.4	29.8	4.6	42.8	6.2
Chili	29.9	31.3	4.1	77.4	6.9
India	12.3	1.5	0.2	40.4	3.6
Japan	15.9	3.2	3.4	12.5	2.6
Turkey	33.7	13.5	1.7	32.7	7.3
Australia	34.1	112.1	10.3	132.6	16.1
New Zealand	47.2	105.3	13.6	215.8	19.4

Table B.2: the open/close book data (Mardia, Kent and Bibby, 1979)

	V1	V2	V3	V4	V5
1	77	82	67	67	81
2	63	78	80	70	81
3	75	73	71	66	81
4	55	72	63	70	68
5	63	63	65	70	63
6	53	61	72	64	73
7	51	67	65	65	68
8	59	70	68	62	56
9	62	60	58	62	70
10	64	72	60	62	45
11	52	64	60	63	54
12	55	67	59	62	44
13	50	50	64	55	63
14	65	63	58	56	37
15	31	55	60	57	73
16	60	64	56	54	40
17	44	69	53	53	53
18	42	69	61	55	45
19	62	46	61	57	45
20	31	49	62	63	62
21	44	61	52	62	46
22	49	41	61	49	64
23	12	58	61	63	67
24	49	53	49	62	47
25	54	49	56	47	53

	V1	V2	V3	V4	V5
26	54	53	46	59	44
27	44	56	55	61	36
28	18	44	50	57	81
29	46	52	65	50	35
30	32	45	49	57	64
31	30	69	50	52	45
32	46	49	53	59	37
33	40	27	54	61	61
34	31	42	48	54	68
35	36	59	51	45	51
36	56	40	56	54	35
37	46	56	57	49	32
38	45	42	55	56	40
39	42	60	54	49	33
40	40	63	53	54	25
41	23	55	59	53	44
42	48	48	49	51	37
43	41	63	49	46	34
44	46	52	53	41	40
45	46	61	46	38	41
46	40	57	51	52	31
47	49	49	45	48	39
48	22	58	53	56	41
49	35	60	47	54	33
50	48	56	49	42	32

Table B.2: the open/close book data (Mardia, Kent and Bibby, 1979) (Cont.)

	V1	V2	V3	V4	V5
51	31	57	50	54	34
52	17	53	57	43	51
53	49	57	47	39	26
54	59	50	47	15	46
55	37	56	49	28	45
56	40	43	48	21	61
57	35	35	41	51	50
58	38	44	54	47	24
59	43	43	38	34	49
60	39	46	46	32	43
61	62	44	36	22	42
62	48	38	41	44	33
63	34	42	50	47	29
64	18	51	40	56	30
65	35	36	46	48	29
66	59	53	37	22	19
67	41	41	43	30	33
68	31	52	37	27	40
69	17	51	52	35	31
70	34	30	50	47	36
71	46	40	47	29	17
72	10	46	36	47	39
73	46	37	45	15	30
74	30	34	43	46	18
75	13	51	50	25	31

	V1	V2	V3	V4	V5
76	49	50	38	23	9
77	18	32	31	45	40
78	8	42	48	26	40
79	23	38	36	48	15
80	30	24	43	33	25
81	3	9	51	47	40
82	7	51	43	17	22
83	15	40	43	23	18
84	15	38	39	28	17
85	5	30	44	36	18
86	12	30	32	35	21
87	5	26	15	20	20
88	0	40	21	9	14

References

Afifi, A. A. and Clark, V. (1990), *Computer-Aided Multivariate Analysis 2nd ed.*, Van Nostrand Reinhold Company.

Becker, R. A., Chambers, J. M. and Wilks, A. R.(1988), *The New S Language*, Wadsworth & Brooks/Cole.

Cox, B. J. (1986), *Object-Oriented Programming, An Evolutionary Approach, 2nd ed.*, Addison-Wesley.

Fukumori, M., Yanagi, K. and Tanaka, Y. (1994) Modified Non-linear Mapping and its application to the problem of detecting influential subsets, *Proceeding of the Fifth Japan-China Symposium on Statistics* (Edited by Ichimura, M., Mao, S. and Fan, G.), University education press, 76 – 79.

Gale, W. A. (1986a), REX Review, in *Artificial Intelligence and Statistics* (Edited by Gale, W. A.), Addison Wesley.

Gale, W. A. (1986b), Student Phase 1 – A Report on Work in Progress, in *Artificial Intelligence and Statistics* (Edited by Gale, W. A.), Addison Wesley.

Hadi, A. S. (1985) K-Clustering and the Detection of Influential Subset (Letter to the Editor With Response), *Technometrics*, Vol. 27, 323–325.

Hampel, F. R. (1974), The influence curve and its role in robust estimation, Journal of American Statistic Associate, Vol. 69, 383–393.

Hayashi, A. (1993), A Consultation System for Statistical Analysis (doctor thesis).

Hayashi, A. and Tarumi, T. (1992), A Consultation System for Statistical Analysis on Hypertool, in *Proceedings of 10th Symposium on Computational Statistics* (Edited by Dodge, Y. and Whittaker, J.), Springer-Verlag.

Hayashi, A. and Tarumi, T. (1994), SCSK: A Statistical Consultation System Based on Knowledge Linkages, in *Short Communication in COMPSTAT 94* (Edited by Dutter, R. and Grossman, W.), Physica-Verlag.

Mardia, K, V., Kent, J. T. and Bibby, J. M. (1979), *Multivariate Analysis*, Academic Press.

Minami, H., Mizuta, M. and Sato, Y. (1993a), A Knowledge Supporting System for Data Analysis, Journal of the Japanese Society of Computational Statistics, Vol. 6, No. 1, pp. 85–94.

Minami, H., Mizuta, M. and Sato, Y. (1994b), A Framework of Knowledge Base for Data Analysis Supporting System and It's Implementation, Bulletin of The Computational Statistics of Japan, Vol. 6, No. 1, 2, pp. 37–48 (in Japanese).

Minami, H., Mizuta, M. and Sato, Y. (1994), A Multivariate Data Analysis Supporting System with Assumption-based Researching Function, Japanese Journal of Applied Statistics, Vol. 23, No. 2, pp. 63–79 (in Japanese).

Moon, S. H., Yanagi, K. and Tanaka, Y. (1992) A Clustering Algorithm of for Detecting Influential Subset in Multivariate method, Journal of the Japanese Society of Computational Statistics, Vol. 5, No. 1, pp. 21–31.

Nakano, J., Yamamoto, Y. and Okada, M. (1991), A Knowledge-based Multiple Regression Analysis Supporting System, Japanese Journal of Applied Statistics, Vol. 20, No. 1, pp. 11–23 (in Japanese).

NeXT Computer, Inc. (1993), *NEXTSTEP Developer's Library*, Addison Wesley.

Prinson, L. J. and Wiener, R. S. (1991), *Objective-C, Object-Oriented Programming Techniques*, Addison Wesley.

Rousseeuw P. J. (1984), Least Median of Squares Regression, J. Amer. Statist. Assoc., Vol. 70, 871-880.

SAS Institute Inc.(1985), *SAS/IML User's Guide(Ver. 5 ed.)*, SAS Institute Inc.

Spiegelhalter, D. J. (1986), A Statistics View of Uncertainly in Expert system, in *Artificial Intelligence and Statistics* (Edited by Gale, W. A.), Addison Wesley.

Tanaka, K.(1990), *An Introduction to Fuzzy Theory for Applicative Use*, Rasseru-Sha (in Japanese).

Takana, Y. (1988), Sensitivity analysis in principal component analysis: influence on the subspace spanned by principal components, Comm. Statist. Vol. A17, 3157 – 3175.

Takana, Y. (1992), Sensitivity analysis in multivariate methods: Principals, methods and software, Technical Report No. 52, Okayama Statisticians Group.

Takana, Y., Castaño-Tostado, E. and Odaka, Y. (1990), Sensitivity analysis in factor analysis: methods and software, in *COMPSTAT 90* (Edited by Momirović, K. and Mildner, V.), Physica-Verlag, 205–210.

Takana, Y. and Odaka, Y. (1989a), Influential observations in principal factor analysis, Psychometrika, Vol. 54, 475–485.

Takana, Y. and Odaka, Y. (1989b), Sensitivity analysis in maximum likelihood factor analysis, Comm. Statist., Vol. A18, 4067–4084.

Tanaka, Y., Tarumi, T. and Wakimoto, K. (1984). *Handbook of Statistical Analysis with Programs for Personal Computers, (2)*, Kyoritsu Publishing Company (in Japanese).

Tarumi, T. and Yanagi, Y. (1994), Data Class for Statistical Software, in *Proceedings of 2nd French-Japan Statistical Conference.*

Thisted, R. A. (1986), Representing Statistical Knowledge for Expert Data Analysis System, in *Artificial Intelligence and Statistics* (Edited by Gale, W. A.), Addison Wesley.

Wakimoto, K. and Taguri, M.(1978). Constellation graphical method for representing multi-dimensional data, *Annals of the Institute of Statistical Mathematics*, Vol. 30, No. 1, 97–104.

Wakimoto, K., Tarumi, T. and Tanaka, Y. (1984). *Handbook of Statistical Analysis with Programs for Personal Computers, (1)*, Kyoritsu Publishing Company (in Japanese).

Wakimoto, K., Tarumi, T. and Tanaka, Y. (1992), *Handbook of Statistical Analysis with Programs for Personal Computers, (6)*, Kyoritsu Publishing Company (in Japanese).

Wakimoto, K.(1993), Linked Lines Rotation Graphics for Looking at Multivariate Data, Journal of the Japanese Society of Computational Statistics, Vol. 6, No. 1, pp. 1–9.

Wiener, R. S. and Prinson, L. J. (1988), *An Introduction to Object-Oriented programming and C++*, Addison Wesley.

Wolfram, S. (1991), *Mathematica : a system for doing mathematics by computer 2nd ed.*, Addison-Wesley.

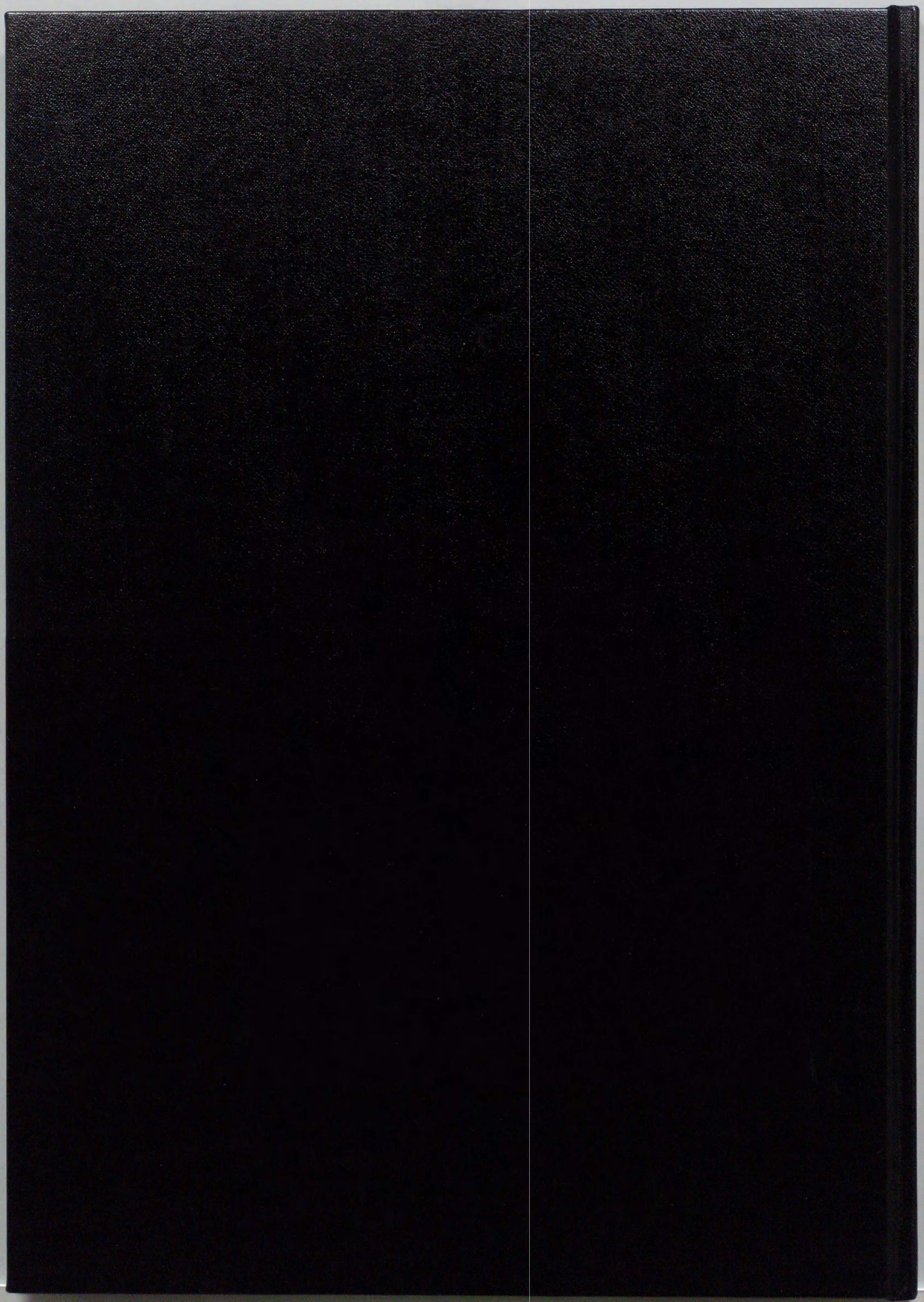
Yanagi, K., Kataoka, T. and Wakimoto, K. (1994), The Cluster Method by Using Linked Lines Rotation Graphics, Bulletin of The Computational Statistics of Japan, Vol. 7, No. 1, pp. 21 – 28 (in Japanese).

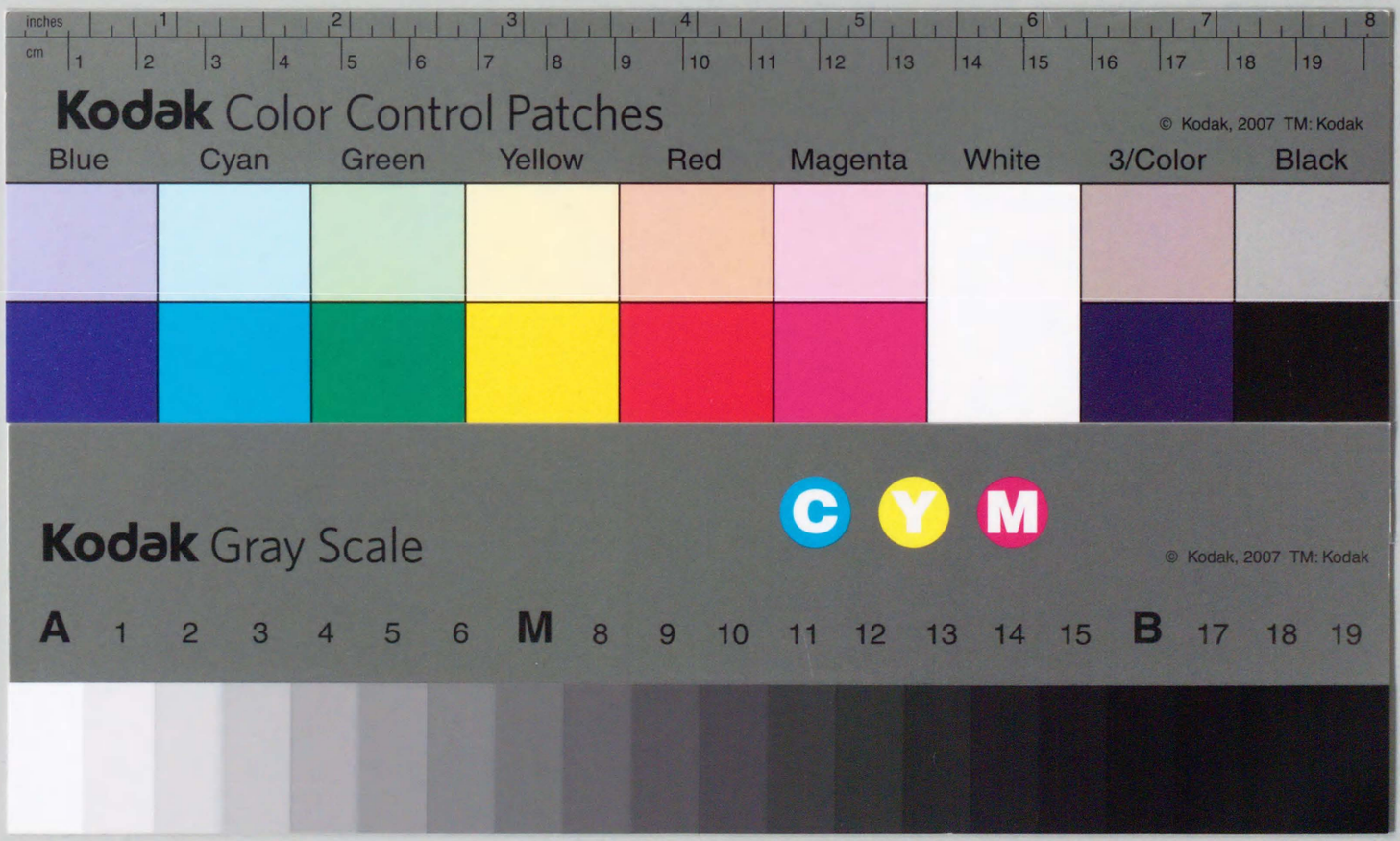
Yanagi, K. (1994), The Software with Knowledge Base for Statistical Analysis, *Proceedings of The Eighth Japan and Korea Joint Conference of Statistics*, 169 – 174.

Acknowledgements

I would like to express my deeply gratitude to Professor Tomoyuki Tarumi, Okayama University, for his hearty teaching, guidance and continuous encouragement in my study. I also want to express my gratitude to late Professor Kazumasa Wakimoto, Okayama University, and Professor Yutaka Tanaka, Okayama University for their valuable suggestions and encouragement.

I wish to express my deeply thanks Dr. Atsuhiro Hayashi, Okayama Prefectural University, for his hearty advice and encouragement. I wish to express my thanks to Dr. Sung Ho Moon, Pusan University of Foreign Studies, Mr. Takeshi Kataoka, Sanyo Gakuen University, and all members of Okayama Statistical Association for their helpful assistance.




inches
cm
Kodak Color Control Patches
© Kodak, 2007 TM: Kodak
Blue
Cyan
Green
Yellow
Red
Magenta
White
3/Color
Black
Kodak Gray Scale
C
Y
M
© Kodak, 2007 TM: Kodak
A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19