INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

CLUSTERING METHODOLOGIES APPLIED TO SHORT-TERM ENSEMBLE FORECASTING: AN EXERCISE IN DATA MINING.

A Dissertation

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

By

Ahmad A. I. Alhamed

Norman, Oklahoma

2000

UMI Number: 9988308

UMI Microform 9988308

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, Mi 48106-1346

Clustering Methodologies Applied To Short-Term Ensemble Forecasting: AN EXERCISE IN DATA MINING.

A Dissertation APPROVED FOR THE SCHOOL OF COMPUTER SCEINCE

BY

S. Lakshmivarahan

Michael B. Richman

Anindya Das

Sridhar Radhakrishnan

Dedication

To my respected mother for her prayers, sacrifice, and support.

To my wife Huda for her continuous encouragement, sacrifice, and love.

And to my little lovely children Munira, Abdullah, and Abdulaziz.

ACKNOWLEDGMENT

I wish to express my gratitude and thanks to my teacher and advisor Dr. S. Lakshmivarahan, George Lynn Cross Research Professor, for his scientific guidance. I am grateful for his kindness, help, and continuous encouragement during the years I spent at the University of Oklahoma.

I thank Dr. Michael Richman, School of Meteorology at the University of Oklahoma. His help, comments and suggestions are greatly appreciated. I wish to thank the member of my advisory committee Dr. S. Dhall, Dr. S. Radhakrishnan, and Dr. A. Das for their support and help.

Dr. David Stensrud, National Severe Storms Laboratory (NSSL) is gratefully acknowledged for his review, feedbacks and comments that enrich the meteorological aspects of this research. I also thank him for his subjective analysis and validation on the SAMEX data (chapter 4). Thanks are also due to Dr. Joseph Schaefer, Dr. Russell Schneider, Mr. Mike Baldwin of Storm Prediction Center (SPC) and Dr. Harold Brooks and Dr. Chuck Doswell of NSSL for their participation in debating on several key issues that lie at the core of this work. I also thank Dr. Henry Neeman, a research scientist with the Center for Analysis & Prediction of Storms (CAPS) at the University of Oklahoma for his help in obtaining SAMEX data sets and providing the program reader. All the computations related to this project were performed on the Environmental Computing and Applications System (ECAS) at the University of Oklahoma.

The author is supported by a scholarship from the General Organization for the Vocational Training and Technical Education, Riyadh, Saudi Arabia.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In ensemble forecasting system, the divergence of the ensemble members during evolution may lead to the identification of clusters of different sizes in the final state. Based on the size of each cluster, probabilities can be assigned objectively to different outcomes. Hence, identifying the clustering structures of the ensemble members is very significant in the ensemble forecasting system which in turn may lead to better forecasts for public consumption. In this work, we use a fundamental methodology from data mining, namely clustering, to detect and identify the clustering structures of the ensemble members during their evolution. To achieve this goal, in this research, we use two tools from the multivariate data analysis: cluster analysis and principal component analysis. Various clustering methodologies and cluster validation techniques are applied to the output of two short-term ensembles. The first one is a uni-model ensemble, which is the output of the Eta model, consists of 10 members. The second is a mutli-model ensemble consists of 25 members. The multi-model, known as the Storm And Mesoscale Ensemble Experiment (SAMEX), is a project coordinated by CAPS at the University of Oklahoma. Four models (ARPS, Eta, RSM, MM5) are used to construct the SAMEX ensemble data. The ultimate goal of our research is to automate the aspects of clustering the ensemble members in the ensemble forecasting system. Such automation is implemented by developing an expert system. Our research provides the infrastructures for building this expert system.

CHAPTER 1

INTRODUCTION

Clustering objects according to their similarities is one of the basic tools that is often used in research in biology, psychology, medicine, marketing, and meteorology (Anderberg 1973, Basilevsky 1983, Everitt 1993, Jain and Dubes 1988, Wilks 1995, Okada 1996, and Mirkin and Muchnik 1996). The process of organizing data into sensible clusters, or groups, is one of the most important modes of understanding and learning. Clustering has a variety of applications in image processing, and pattern recognition (Jain and Dubes 1988). In this research, we deal with the applications of clustering in meteorology and, particularly, in the ensemble forecasting system.

Ensemble forecasting represents a very different philosophical view of numerical weather prediction (NWP) than the traditional approach of producing the best single forecast possible. Instead of single forecast throughout the time period of prediction, the new prediction system provides a collection, or ensemble, of forecasts for the same period of time (Buizza 1997, Molteni *et al.* 1996, Brooks *et al.* 1995, and Houtekamer and Derome 1995). The ensemble approach addresses one of the uncertainties of the forecast process by providing information on probabilities of different outcomes (Brooks *et al.* 1995 and Houtekamer and Lefaivre 1997).

Two types of ensemble forecasting exist, short-term and medium-term ensemble forecasting. Basically, they differ on the length of the time for which the ensemble provides a prediction. This research is concerned with the short-term ensemble forecasting. Mittelstadt (1995) defined ensemble forecasting as "the process of

introducing small perturbations to the initial conditions and examining their growth in order to determine the predictability of model forecasts". So in this approach, there is no single valid forecast but rather a collection of possible solutions are provided.

The ensemble refers to the set of model solutions, or members, such that each member is created by introducing small perturbation to the initial condition, IC (Du and Mullen 1997, Hamill and Colucci 1997 and 1998). These members, who are initiated from slightly different initial conditions, follow different trajectories as they evolve, and none of them may be the "true" state of the atmosphere, Figure 1-1. The large circle (Finishing Time) in Figure 1-1 represent the envelope of uncertainty about the true state of the atmosphere. In general, the mean of the ensemble should provide a better forecast if most of the members agree within a given tolerance. More importantly, the divergence of the ensemble members during evolution may lead to creation of some clusters in the finishing time. Each of these clusters, dashed circles in Figure 1-1, may contain different number of members. Hence, probabilities can be assigned objectively to different scenarios based on the size of each cluster (Tracton and Kalnay 1993). Thus, clustering methodologies are used, in the process of the ensemble forecasting, to group the members of ensemble that are similar in some respect into clusters.

The aim of our research is to detect and identify the clustering structures of the ensemble members during their evolution. Various clustering methodologies are applied to the model output. The models are the Eta model (Black 1994), the Regional Spectral Model (RSM; Juang and Kanamitsu 1994), the Advanced Regional Prediction System (ARPS; Xue *et al.* 2000), and the Pennsylvania State University - National Center for Atmospheric Research Mesoscale Model version 5 (MM5; Dudhia 1993; Grell *et al.*

1994). In this research, the trajectories of ensemble members initiated from different initial conditions are analyzed.

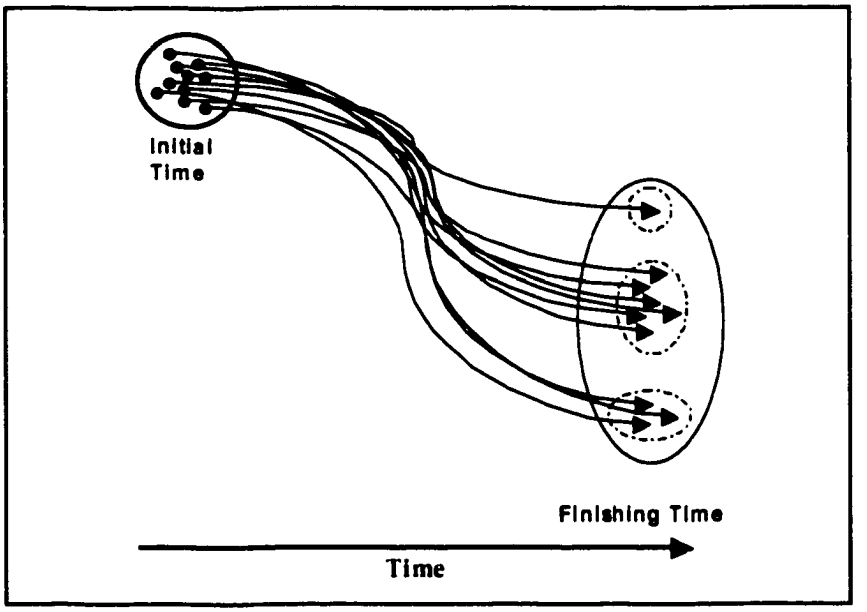


Figure 1-1: The trajectories of the members of an ensemble during their evolution. "T" and the dotted line represent the truth.

Clustering, distributing objects into several groups so that objects in one cluster are similar and objects from different groups are dissimilar, is one of the dominant techniques of exploratory data analysis. To achieve the aim of this research, we use two multivariate data analysis techniques: **cluster analysis** and **principal component analysis**. Cluster Analysis (CA) is the formal study of algorithms and methods for grouping objects (Jain and Dubes 1988). An **object**, such as member in one ensemble, is an entity in a data set described by a set of measurements called **attributes, features,** or **observations**. Cluster analysis is a tool used for estimating similarities and, thus finding out which objects in one set are similar, so that the similar objects can be grouped into a cluster.

Principal Component Analysis (PCA) is a data reduction and linear transformation statistical technique. It transforms a set of correlated variables into a set of uncorrelated variables (Basilevsky 1983, Hotelling 1933, Jackson 1991, Preisendorfer 1988, and Wilks 1995). The new variables are linear transformations of the original variables. The new variables, in addition to being uncorrelated, account for decreasing portions of the variance of the original variables, hence the data reduction (Everitt 1993).

Clustering has been applied to atmospheric research for about three decades (Key and Crane 1986, Ronberg and Wang 1987, Mo and Ghil 1988, and Komarov *et. al.* 1994). Gong and Richman (1995) carried out an intercomparison of various clustering techniques on a well-studied data set. The authors conducted a detailed survey of the literature on clustering applications in atmospheric research. A comprehensive list of these applications can be found in Gong and Richman (1995).

Clustering is a classical data mining problem. Its techniques and algorithms are regarded as a fundamental methodology to be used in data mining (Cios *et. al.* 1998, Agrawal *et. al.* 1998, Ramakrishnan and Grama 1999, Ganti *et. al.* 1999, Karypis *et. al.* 1999, Witten and Frank 2000, and Berson *et. al.* 2000). Data mining applications aim to detect, interpret, and predict the qualitative and/or quantitative patterns in the data (Ramakrishnan and Grama 1999). The principal component analysis is often used in data mining as a preprocessing technique for major operations such as objects projection, attribute extraction, and data reduction (Cios *et. al.* 1998).

The ultimate goal of this research is to automate the aspects of clustering the ensemble members in the ensemble forecasting system. Achieving this goal could lead

eventually to automating the process of making decision regarding weather prediction. Such automation is implemented by developing an expert system (Backer 1995). The expert system should utilize the clustering methodology and draw a conclusion based on a set of stored rules, called the knowledge base, and other information supplied by user. This research provides the infrastructure for building such an expert system. In addition, it provides the guidelines for the developer about the applicability of the various clustering techniques to the context of meteorology and ensemble forecasting, in particular.

This dissertation is organized as follow. In chapter 2, on Mathematical Preliminaries, we describe the properties of the data set including its representation, scaling and normalization. Summaries of the clustering algorithms and the basic ideas from the principal component analysis are given in this chapter. A description of the uni-model called the Eta-model short-term ensemble data and discussion of the results of the analysis of ten fields – precipitation, pressure, temperature, and convective available potential energy at the surface level, height at 500 mb, wind at 250 mb, absolute vorticity at 250, 500, and 850 mb, and pressure vertical velocity at 700 mb – are given in chapter 3. Analyses of the sensitivity of the clustering algorithms with respect to variation in parameters are also given in this chapter. In chapter 4, the multi-model short-term ensemble data (SAMEX) are described, followed by a discussion of the results of cluster analyzing, roughly, the same set of fields. Chapter 5 presents the theory of cluster validation and discusses many of the common approaches and algorithms for validating the clustering solution resulting from a clustering algorithm. Applications of the validation techniques on SAMEX data are discussed in chapter 6.

Concluding remarks including a summary of key results, and potential extensions are given in chapter 7. The computer format of the model output, called GRIB format, is described in Appendix A. The program listings of the software, we developed to implement this research, are given in Appendix B.

CHAPTER 2

MATHEMATICAL PRELIMINARIES

2.1 Data Set

Let $X=[x_{ij}]$, $1 \le i \le m$, $1 \le j \le n$, be the ***data matrix***. Each column represents an ***object***, and each row represents an ***attribute***. In the meteorological context, objects and attributes are sometimes called ***variables*** and ***observations*** respectively. The words objects and variables are used interchangeably to refer to the data units (ensemble members, for instance) and the words attributes and observations are used interchangeably to refer to the set of measurements that describe the data unit. Thus, x_{ij} denotes the i^{th} observation on the j^{th} variable. The aim is to discover similarities among objects (Romesburg 1984). Since each object can be viewed as a vector of m-dimensions, where m is the number of attributes, and each row can also be viewed as a vector of n-dimensions, where n is the number of objects, we will use the following notations:

$x_{\bullet j}$: refers to the j^{th} object.

$x_{i\bullet}$: refers to the i^{th} attributes across the n object.

2.1.1 Scales for Data.

Scale of measurement for observations is a rule of assigning numbers to represent different states of the variables being observed. There are four typical scales that researchers use to measure the attributes. ***Nominal scale*** merely distinguishes values of objects, as being equal or not equal. For example, a binary variable takes true/false values, and gender takes Male/Female values. When scale induces an ordering of the

variables, it is called ***ordinal scale***. In the ordinal scale, in addition to distinguishing whether two variables are equal or not, the state of inequality is further refined to distinguish whether one is greater or smaller than the other is. Rating on a scale of 1 to 10 and grades in a course say A, B, C, D, and F are examples of ordinal scale. These two scales, nominal and ordinal, are usually called ***qualitative scales***. There are two other scales called ***interval*** and ***ratio*** scales. These are also known as ***quantitative scales***. Interval scale assigns a meaningful measure of the **difference** between two variables. Ratio scale is an interval scale with a meaningful and fixed zero point. Thus, in addition to the differences, we can also talk about the ratio of values within this scale. The data set used in our study is all in the ratio scale.

2.1.2 Normalizing the data matrix.

The measurement unit can arbitrarily affect the similarities among objects. By normalizing the attributes and recasting them in dimensionless units, this arbitrary affect is removed. Normalizing the data matrix makes each attribute contributes more equally to the overall resemblance (Romesburg 1984). There are several methods for normalizing data matrix. Each method transforms the data matrix X into new data matrix Z of the same size. The new matrix, Z, called ***normalized*** data matrix, can then be used in data analysis.

Let

$$\bar{x}_{\bullet j} = \frac{1}{m}\sum_{i=1}^{m} x_{ij} \tag{2.1}$$

and

$$\sigma_{\bullet j} = \left[\frac{1}{m-1}\sum_{i=1}^{m}\left(x_{ij} - \bar{x}_{\bullet j}\right)^2\right]^{\frac{1}{2}}. \tag{2.2}$$

Clearly, $\bar{x}_{\bullet j}$ and $\sigma_{\bullet j}$ are the **unbiased** estimates of the **sample mean** and the **sample standard deviation** for the j^{th} object, respectively.

The simplest type of normalization subtracts the object mean from all attributes of this object:

$$z_{ij} = x_{ij} - \bar{x}_{\bullet j} \qquad \text{for } 1 \le i \le m. \tag{2.3}$$

Z obtained using this type of normalization is the **centered** data matrix, and in the meteorological context is also called the ***difference field***. The second type of normalization translates and scales the objects so that they have zero mean and unit variance:

$$z_{ij} = \frac{x_{ij} - \bar{x}_{\bullet j}}{\sigma_{\bullet j}} \qquad \text{for } 1 \le i \le m. \tag{2.4}$$

Other types of normalization include scaling by the maximum of each object as in (2.5), by the minimum and maximum of each object as in (2.6) or by the maximum of the entire data matrix as in (2.7):

$$z_{ij} = \frac{x_{ij}}{\underset{1 \le i \le m}{MAX}\{x_{ij}\}}, \qquad -1 \le z_{ij} \le 1 \tag{2.5}$$

$$z_{ij} = \frac{x_{ij} - \underset{1 \le i \le m}{MIN}\{x_{ij}\}}{\underset{1 \le i \le m}{MAX}\{x_{ij}\} - \underset{1 \le i \le m}{MIN}\{x_{ij}\}}, \qquad 0 \le z_{ij} \le 1 \tag{2.6}$$

$$z_{ij} = \frac{x_{ij}}{\underset{\substack{1 \le i \le n \\ 1 \le j \le m}}{MAX}\{x_{ij}\}}, \qquad -1 \le z_{ij} \le 1. \tag{2.7}$$

Sometimes there may be a need to transform the original data matrix by identifying and removing highly unusual values of data called ***outliers***. An outlier is

any value of data that is atypical with respect to other values in the data set (i.e., the values that are quite deviant are labeled outliers) (Romesburg 1984).

Normalizing the original data matrix is an optional step in data analysis. Data analysis based on clustering or principal component analysis can be implemented using either original or normalized data matrix. However, when data matrix is normalized, the computational cost of the normalizing method must be considered in the overall computational complexity of the entire clustering procedure. Table 2-1 lists the number of operations required for each of the normalization methods discussed in this section. It assumed that each of the basic operations (i.e., addition, subtraction, multiplication, and division) takes one unit time. The square root is indicated separately in the table since it is a complex operation. The number of operations is shown as a function of m and n for a given mxn data matrix where m is the number of rows (attributes) and n is the number of columns (objects).

Table 2-1: The number of operations required to implement different normalization methods for a given mxn data matrix.

Normalization method	No. of operations
Centering, or difference field (2.3)	$2mn + n$
Normalization by the mean and standard deviation of each column, (2.4)	$7mn + 3n$ and n square root
Normalization by the maximum of each column, (2.5)	$2mn$
Normalization by the maximum and minimum of each column (2.6)	$4mn + n$
Normalization by the maximum of the entire data matrix (2.7)	$2mn$

2.2 Cluster Analysis (CA) Overview

Cluster analysis (CA) is a statistical technique that is used to discover the property of a collection of objects to group together based on certain similarity measures. There are

two classes of methods for performing cluster analysis: ***hierarchical*** and ***partitional*** methods. A hierarchical method operates on the similarity matrix to construct a tree depicting specified relationships among the objects (Anderberg 1973). It organizes the objects into nested sequence of clusters. Partitional, or non-hierarchical, methods generate one single partition of the objects in an attempt to classify the objects. In these later methods, the number of clusters is specified in advance.

2.2.1 Hierarchical Cluster Analysis (HCA).

There are two algorithmic approaches that can be used to carry out the hierarchical methods: ***agglomerative*** and ***divisive***. In the agglomerative approach, each object is placed in its own cluster and these atomic clusters are gradually merged into larger and larger clusters until all objects are in a single cluster. In contrast, the divisive approach starts with all objects in one cluster and subdivides them into smaller clusters. In this study, we use agglomerative algorithms to implement the hierarchical cluster analysis.

In order to discover a natural clustering of a set of objects, the first step is to compute the similarities between objects. To measure the similarities between objects, a **resemblance coefficient** is computed for each pair of objects. A resemblance coefficient measures the overall resemblance - the degree of similarity - between each pair of objects (Romesburg 1984). The pair wise resemblance between the n objects are arranged in the form of a matrix where the $(i, j)^{th}$ entry denotes the resemblance coefficients between objects i and j. A resemblance matrix is a square, $n \times n$, matrix where n is the number of objects. The concept of resemblance is symmetric; that is, the resemblance between objects i and j is the same as the resemblance between objects j and i. A resemblance coefficient can be of two types either ***similarity*** coefficient or

dissimilarity coefficient. A similarity coefficient means the larger value, more similar the objects are, while dissimilarity coefficient means the less value, more similar the objects are.

Dissimilarity Coefficient. The most commonly used dissimilarity measure is the ***Euclidean distance*** coefficient, e_{jk}, which measures the distance between two objects, say j and k. It can be defined as follows:

$$e_{jk} = \sqrt{\sum_{i=1}^{m} (x_{ij} - x_{ik})^2} \qquad 0 \le e_{jk} \le \infty, \tag{2.8}$$

$$= \sqrt{(x_{\bullet j} - x_{\bullet k})^T (x_{\bullet j} - x_{\bullet k})} = \|x_{\bullet j} - x_{\bullet k}\|_2 \tag{2.9}$$

where $\|y\|_2$ denotes the Euclidean norm of y. Other dissimilarity coefficients include ***Manhattan distance*** and "***sup***" ***distance***. The Manhattan distance is defined as follow:

$$h_{jk} = \sum_{i=1}^{m} |x_{ij} - x_{ik}| = \|x_{\bullet j} - x_{\bullet k}\|_1 \qquad 0 \le h_{jk} \le \infty, \tag{2.10}$$

and is also called the 1-norm. The "sup" distance is defined as follow:

$$p_{jk} = \underset{1 \le i \le m}{MAX} |x_{ij} - x_{ik}| = \|x_{\bullet j} - x_{\bullet k}\|_\infty \qquad 0 \le p_{jk} \le \infty, \tag{2.11}$$

and is also known as the ∞-norm.

Similarity Coefficient. The most commonly used similarity measure is the ***correlation coefficient***, r_{jk}. In term of normalized data under Eq. (2.4), define an $n \times n$ matrix R as follows.

$$R = \frac{1}{m-1} Z^T Z \ , \tag{2.12}$$

where

$$r_{jk} = \frac{1}{m-1}\sum_{i=1}^{m} z_{ij} z_{ik} \,. \qquad (2.13)$$

R is the sample ***correlation matrix***, and r_{jk} is the sample ***correlation coefficient*** between objects j and k and $r_{jk} = 1$ for all $j=k$. Note that under Eq. (2.3), R is the sample ***covariance matrix***. The ***Cosine Coefficient***, c_{jk}, is another similarity coefficient which is defined as

$$c_{ij} = \frac{\sum_{i=1}^{m} x_{ij} x_{ik}}{\left(\sum_{i=1}^{m} x_{ij}^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{m} x_{ik}^2\right)^{\frac{1}{2}}} \qquad \text{where} \qquad -1.0 \le c_{jk} \le 1.0 \,. \qquad (2.14)$$

Clearly, c_{jk} is the cosine of the angle α between the vectors $x_{\bullet j}$ and $x_{\bullet k}$ respectively. c_{jk} can also be rewritten as

$$c_{jk} = \cos\alpha = \frac{x_{\bullet j}^{T} x_{\bullet k}}{\|x_{\bullet j}\|_2 \, \|x_{\bullet k}\|_2} \,. \qquad (2.15)$$

Thus, the smaller the angle α, the larger the similarity.

Another similarity measure, called ***Euclidean Similarity*** (EucSimi), can be derived from the Euclidean distance coefficient as follows (Elmore and Richman 2000).

Let $E=\{e_{ij}\}$, $1 \le i,j \le n$ be the Euclidean distance matrix, define the Euclidean Similarity matrix $\hat{E} = \{ \hat{e}_{ij}, 1 \le i,j \le n \}$ where

$$\hat{e}_{ij} = 1 - \frac{e_{ij}}{\underset{1 \le i,j \le n}{MAX}\, e_{ij}} \,, \qquad 0 \le \hat{e}_{ij} \le 1.0 \quad \text{for } i,j = 1,2, \ldots, n \,. \qquad (2.16)$$

This measure allows dissimilarity-based PCA analyses, where dissimilarity measure cannot be used directly.

The similarity matrix is the input for the hierarchical clustering algorithm. Its computational cost must be considered in the overall computational complexity of the entire clustering procedure. Table 2-2 lists the number of operations required for each of the similarity/dissimilarity measures discussed in this section. It assumed that each of the basic operations (i.e., addition, subtraction, multiplication, and division) takes one unit time. The square root is indicated separately in the table since it is a complex operation. The number of operations is shown as a function of *m* and *n* for a given *mxn* data matrix where *m* is the number of rows (attributes) and *n* is the number of columns (objects).

Table 2-2: The number of operations required to compute the *nxn* similarity matrix using different similarity/dissimilarity measures for a given *mxn* data matrix.

Similarity/dissimilarity measure	**No. of operations**
Euclidean distance	$3m\binom{n}{2}$ and $\binom{n}{2}$ square roots
Euclidean Similarity	$(3m+3)\binom{n}{2}$ and $\binom{n}{2}$ square roots
Correlation	$(8m+9)\binom{n}{2}$ and $\binom{n}{2}$ square roots

A. The Basic framework for the Agglomerative Algorithms

This section describes a framework for agglomerative algorithms (Anderberg 1973).

Let s_{ij} be the similarity between objects i and j as defined by one of the resemblance measures. Since the similarity is symmetric ($s_{ij} \equiv s_{ji}$), the complete schedule of similarities for all $\binom{n}{2} = \frac{1}{2}n(n-1)$ possible pair wise combinations of

objects may be arrayed in lower triangular resemblance matrix. Once the resemblance matrix is defined, the process of clustering is straightforward. A general procedure for agglomerative clustering based on resemblance matrix is as follows (Anderberg 1973).

INPUT: **An $n \times n$ similarity matrix, S_{nxn}.**

OUTPUT: **Clustering tree (dendrogram).**

STEP 1 **Place each object in a separate cluster to construct n clusters each of which contains only one object. Use the integer numbers 1 to n to label the clusters.**

STEP 2 **Find the most similar pair of clusters in the resemblance matrix. Let C_i and C_j be the most similar pair with s_{ij} as the similarity measure between i and j where $i < j$.**

STEP 3 **Merge the two clusters C_i and C_j and reduce the number of clusters by 1. Label the new cluster C_i and update the resemblance matrix to reflect the revised similarities between the new cluster C_i and other existing clusters other than C_j. Delete the row and column of S that corresponds to the cluster C_j.**

STEP 4 **Repeat step 2 and step 3 a total of (n-1) times. At each stage record the elements of each cluster and keep track of all similarity measures at each stage to have a complete record.**

From this framework, it can be readily noticed that different agglomerative methods can be implemented by varying the procedure used for defining the most similar pair at step 2 and for updating the revised resemblance matrix at step 3. Defining

the most similar pair depends on whether the resemblance measure is similarity or dissimilarity coefficient. If similarity coefficient is used then the largest value in the similarity matrix indicates the most similar pair. On the contrary, if dissimilarity coefficient is used then the smallest value in the dissimilarity matrix indicates the most similar pair. Updating the resemblance matrix depends on the specific clustering method used. Let C_i and C_j be the most similar clusters, so they will be merged into a new cluster, say C_t, that's, $C_t = C_i \cup C_j$. Let $\delta(t,r)$ be the function or method that compute the similarity, S_{tr} , between the new cluster C_t and an existing cluster C_r, that is

$$\delta(t,r) = S_{tr} . \tag{2.17}$$

Three of the most well known hierarchical clustering methods that can be used in (2.17) are

$$\delta(t,r) = \min_{\substack{i \in t \\ j \in r}} \{S_{ij}\} \tag{2.18}$$

$$\delta(t,r) = \max_{\substack{i \in t \\ j \in r}} \{S_{ij}\} \tag{2.19}$$

$$\delta(t,r) = \frac{1}{|t||r|} \sum_{\substack{i \in t \\ j \in r}} S_{ij} . \tag{2.20}$$

Functions in (2.18), (2.19), and (2.20) are the functions used in the ***single linkage*** method (***SLINK***), the ***complete linkage*** method (***CLINK***), and the ***average linkage*** method, respectively. The average linkage method is known ***as unweighted pair-group method using arithmetic average*** (***UPGMA***). Figure 2-1 illustrates how the SLINK method updates the resemblance matrix.

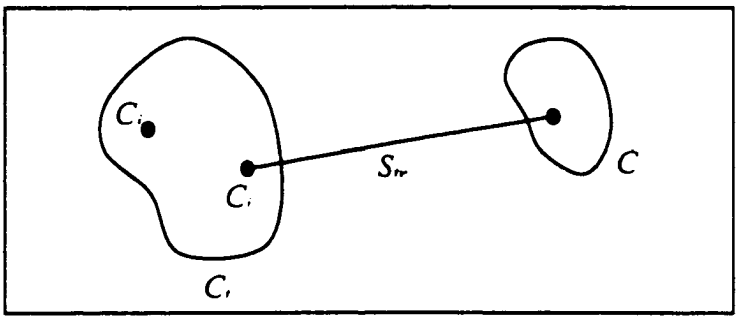


Figure 2-1: Graphical illustration of the SLINK method.

The general steps in a typical hierarchical agglomerative cluster analysis are now illustrated using the following example.

Example 2.1:

Step 1: Obtain or construct the data matrix. Let X be the data matrix for this example.

$$X = \begin{bmatrix} 5 & 12 & 15 & 20 & 25 \\ 15 & 25 & 10 & 23 & 12 \end{bmatrix}$$

Step 2: Standardize the data matrix. This is an optional step since the clustering algorithm can be applied on either the data matrix or on the standardized form. For simplicity of illustration the data matrix X is used directly in this example.

Step 3: Compute the resemblance matrix. The Euclidean distance coefficient (2.9) is used as a dissimilarity coefficient, hence, the dissimilarity matrix is the Euclidean distance matrix $E=\{e_{ij}, 1 \le i < j \le 5\}$.

$$E = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & & & & \\ 12.21 & 0 & & & \\ 11.18 & 15.3 & 0 & & \\ 17 & 8.25 & 13.93 & 0 & \\ 20.22 & 18.38 & 10.2 & 12.08 & 0 \end{bmatrix} \end{array}$$

Step 4: Execute the clustering method. The average linkage (UPGMA) method (2.20) is used to illustrate the clustering procedure in this example. At the beginning, each object is in a separate cluster. The clustering algorithm performs n-1 iterations (4 in this example). At each iteration, the dissimilarity matrix is searched and the most similar two clusters are merged into one cluster and the number of clusters is reduced by one. The dissimilarity matrix is then updated by deleting one row and one column. Equation (2.20) is computed between the components of the newly merged cluster and any other existing cluster in order to prepare the dissimilarity matrix for the subsequent iteration. Using the previously computed dissimilarity matrix E, the UPGMA method proceeds in the following iterations.

Iteration 1: Objects 2 and 4 are the most similar pair. Note that since the Euclidean distance is a dissimilarity coefficient, then the smallest value means the objects (or clusters) are more similar. Objects 2 and 4 are merged to form one cluster (24) => 1, (24), 3, 5.

$$e_{(24)1} = \frac{1}{2}[e_{21} + e_{41}] = \frac{1}{2}[12.21 + 17] = 14.61$$

$$e_{(24)3} = \frac{1}{2}[e_{23} + e_{43}] = \frac{1}{2}[15.3 + 13.93] = 14.62$$

$$e_{(24)5} = \frac{1}{2}[e_{25} + e_{45}] = \frac{1}{2}[18.38 + 12.08] = 15.23$$

$$E = \begin{array}{c} \\ 1 \\ (24) \\ 3 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & (24) & 3 & 5 \end{array} \\ \begin{bmatrix} 0 & & & \\ 14.61 & 0 & & \\ 11.18 & 14.62 & 0 & \\ 20.22 & 15.23 & 10.2 & 0 \end{bmatrix} \end{array}$$

Iteration 2: Objects 3 and 5 are the most similar, so they are merged to form one cluster (35) => 1, (24), (35)

$$e_{(35)1} = \frac{1}{2}[e_{31} + e_{51}] = \frac{1}{2}[11.18 + 20.22] = 31.4$$

$$e_{(35)(24)} = \frac{1}{4}[e_{32} + e_{34} + e_{52} + e_{54}] = \frac{1}{4}[15.3 + 13.93 + 18.38 + 12.08] = 14.92$$

$$E = \begin{array}{c} \\ 1 \\ (24) \\ (35) \end{array} \begin{bmatrix} 1 & (24) & (35) \\ 0 & & \\ 14.61 & 0 & \\ 31.4 & 14.92 & 0 \end{bmatrix}$$

Iteration 3: Object 1 and (24) are the most similar, so they are merged to form one cluster (124) => (124), (35)

$$e_{(124)(35)} = \frac{1}{6}[e_{13} + e_{15} + e_{23} + e_{25} + e_{43} + e_{45}] = 15.18$$

$$E = \begin{array}{c} \\ (124) \\ (35) \end{array} \begin{bmatrix} (124) & (35) \\ 0 & \\ 15.18 & 0 \end{bmatrix}$$

Iteration 4: Of course only two clusters remain, so they are combined into a single cluster (12345) that contains all objects. The clustering dendrogram (or tree) can be drawn from these 4 iterations. Figure 2-1 shows the clustering dendrogram corresponding to the clustering structure obtained for this example.

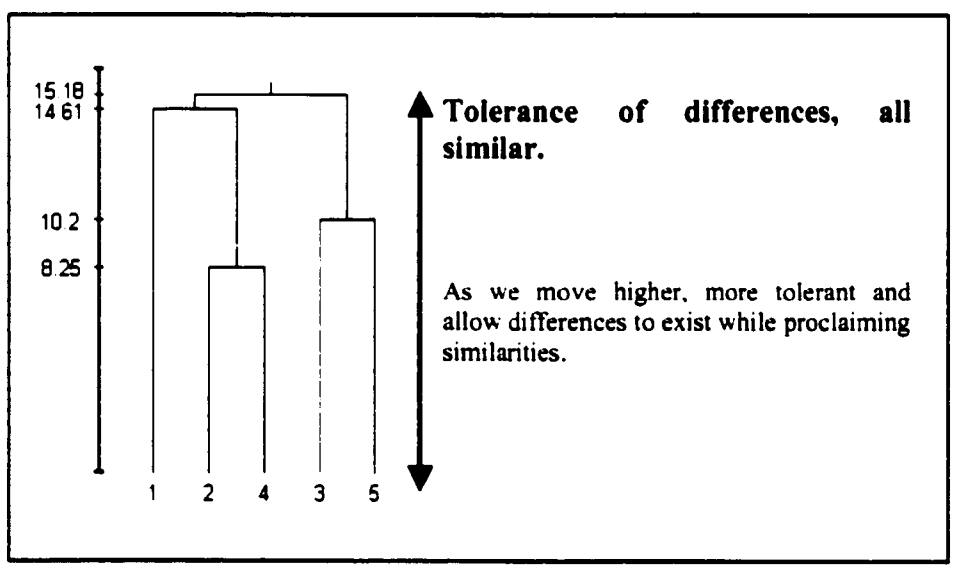


Figure 2-2: Clustering trees for Example 2.1

Ward's method, also called the ***minimum variance*** method, is another popular hierarchical clustering method. This method is based on the notion of square error. In Ward's method, at each clustering step, the value of the sum-of-square-error must be computed for every possible merger of two clusters. The merger that produces the smallest increase in the value of sum-of-square-error is taken to be the clustering of this step. This process is repeated in each step until one large cluster, which contains all objects is obtained. Initially, each cluster contains only one object; hence the value of the sum-of-square-error at the beginning is zero. Several studies found that the Ward's method is the best among the hierarchical clustering methods and its clustering result outperforms all other methods (McIntyre and Blashfield 1980; Morey *et al.* 1983; Jain and Dubes 1988; Breckenridge 1989; and Gong and Richman 1995).

The four methods discussed in this section are used in step 3 of the general procedure for the agglomerative clustering to update the resemblance matrix after each merge of the two most similar pair. The update is done by defining the resemblance between a newly formed cluster, (i, j), and an existing cluster, r, with n_r objects. A general formula for implementing this updating has been proposed by Lance and Williams (1967) for most of the commonly hierarchical clustering methods. This formula is known as Lance and Williams' recurrence formula and is given as follow.

$$S[(r),(i,j)] = \alpha_i S[(r),(i)] + \alpha_j S[(r),(j)] + \beta S[(i),(j)] + \gamma |S[(r),(i)] - S[(r),(j)]| \quad (2.21)$$

Different clustering methods are implemented using this general formula by varying the values of α_i, α_j, β, and γ parameters. The parameters values for the clustering algorithms discussed in this chapter are given in Table 2-3 (Lance and Williams 1967; Milligan 1979; Jain and Dubes 1988; and Everitt 1993). The computational complexity of any

hierarchical clustering method is $O(n^2)$, where n is the number of objects for a given nxn similarity matrix (Cios *et. al.* 1998).

Table 2-3: Parameter values of the Lance and Williams' recurrence formula for several clustering algorithms

Clustering Method	α_i	α_j	β	γ
SLINK	½	½	0	-½
CLINK	½	½	0	½
UPGMA	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
Ward's	$\frac{n_i+n_r}{n_i+n_j+n_r}$	$\frac{n_j+n_r}{n_i+n_j+n_r}$	$\frac{-n_r}{n_i+n_j+n_r}$	0

Cophenetic Correlation

Once the clustering dendrogram is obtained from the hierarchical clustering algorithm, the question now is how well the resultant tree fits the data. For such purpose, the **cophenetic correlation** coefficient (*CP*) has been proposed for the interval and ratio scale data (Romesburg 1984). The cophenetic correlation coefficient *CP* is the Pearson product-moment correlation coefficient between the entries of the observed resemblance matrix *S* and the entries of the cophenetic matrix. The cophenetic matrix, *P*, is an nxn symmetric matrix constructed using the clustering dendrogram found by the algorithm. The entry p_{ij} in the cophenetic matrix is the level in the dendrogram at which objects i and j are first merged in the same cluster. The cophenetic correlation coefficient CP is written as follow.

$$CP = \frac{\sum s_{ij}p_{ij} - \frac{1}{M}\sum s_{ij}\sum p_{ij}}{\left[\left(\sum s_{ij}^2 - \frac{1}{M}\left(\sum s_{ij}\right)^2\right)\left(\sum p_{ij}^2 - \frac{1}{M}\left(\sum p_{ij}\right)^2\right)\right]^{\frac{1}{2}}}, \quad -1 \le CP \le 1 \qquad (2.22)$$

Since the two matrices are symmetric, only the upper triangular matrices are considered in (2.22). That is, the entries below the main diagonals. Thus $M = \frac{1}{2}n(n-1)$ and all sums are over the set $\{(i,j): 1 \le i < j \le n\}$. The better match between the dendrogram and the data is when CP is close to 1.

Example 2.2:

The cophenetic correlation coefficient for example 2.1 is the product-moment correlation coefficient between the entries of the observed dissimilarity matrix and the entries of the cophenetic matrix. Here is the dissimilarity matrix from example 2.1 and the cophenetic matrix constructed from the clustering tree in Figure 2-1.

$$S = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & & & & \\ 12.21 & 0 & & & \\ 11.18 & 15.3 & 0 & & \\ 17 & 8.25 & 13.93 & 0 & \\ 20.22 & 18.38 & 10.2 & 12.08 & 0 \end{bmatrix} \end{array}, \quad P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & & & & \\ 14.61 & 0 & & & \\ 15.18 & 15.18 & 0 & & \\ 14.61 & 8.25 & 15.18 & 0 & \\ 15.18 & 15.18 & 10.2 & 15.18 & 0 \end{bmatrix} \end{array}$$

The cophenetic correlation coefficient (2.22) is:

$$CP = 0.66$$

2.2.2 Nonhierarchical, Partitional, Cluster Analysis (nHCA).

Hierarchical clustering algorithms organize the objects into a nested sequence of clusters. An important characteristic of hierarchical methods is the visual impact of the tree, which enables a data analyst to see how objects are being merged into clusters at successive levels of similarity. Nonhierarchical, or partitional cluster analysis (nHCA) algorithms, on the other hand, are designed to cluster objects into a single classification of k clusters, where k either is specified a priori or is determined as part of the clustering method (Anderberg 1973). As seen in the previous section, hierarchical methods

generally require only the resemblance matrix among the objects, whereas partitional methods expect the data in the form of a data matrix, raw or normalized. It is generally assumed that the attributes have been measured on a ratio scale. The problem of partitional clustering may be stated as follows (Jain and Dubes 1988). Given n objects, determine a partition of the objects into k clusters such that the objects in a cluster are more similar to each other than to objects in different clusters. The value of k may or may not be specified. An explicit clustering criterion is also to be specified.

The basic idea of partitional clustering method is to choose some initial partition of the set of objects, and a criterion. Then evaluate the criterion for all possible partitions containing k clusters, alter the memberships of clusters if needed. Finally, pick the partition that optimizes the criterion. The first difficulty encountered is the selection of the initial partition of the objects into clusters. The second difficulty is the selection of a criterion. Criteria are highly dependent on problem parameters and must be simple for computational reasons but complex enough to reflect various data structures (Jain and Dubes 1988).

Initial Partition. An initial partition can be formed by first specifying a set of k seed points. A set of k seed points can be used as cluster nuclei around which the set of n objects can be grouped (Anderberg 1973). The k seed points can be the first k objects in the data set, or they can be chosen subjectively or randomly. One other scheme is by taking any desired partition of the objects into k mutually exclusive clusters and computes the cluster centroids as seed points.

Once a set of k seed points have been chosen, an initial partition can be easily obtained. One method to obtain an initial partition is to assign each object to the cluster

built around the nearest seed point. This point in this method remains stationary throughout one pass over all objects. Another method to obtain initial clustering is to let each seed point to form a cluster of one member. Then assign objects one at a time to the cluster with the nearest centroid; after an object is assigned to a cluster; update the centroid so that it is the true mean vector for all the objects currently in that cluster. Hierarchical clustering of the set of objects can also be used to obtain an initial partition. In most cases a hierarchical clustering method can produce an excellent initial partition. Moreover, the analyst could use his judgment to sort the set of objects into an initial partition, or rely on some random allocation schemes.

A. Criteria for Partitional Clustering.

Let $X = [\, x_{\bullet 1}, x_{\bullet 2}, \ldots\ldots\ldots, x_{\bullet n}]_{m \times n}$. The problem is to partition X into k clusters, such that $X = C_1 \cup C_2 \cup \ldots\ldots \cup C_k$, and $C_i \cap C_j = \phi$ the null set, $1 \le i, j \le k$, and $i \ne j$.

Let $|C_i| = n_i$, and $\sum_{i=1}^{k} n_i = n$, where n_i is the number of object in the i^{th} cluster. Let m_i be the mean vector or center of cluster C_i that is defined as the centroid of cluster C_i,

$$m_i = \frac{1}{n_i} \sum_{x_{\bullet j} \in C_i} x_{\bullet j} \tag{2.23}$$

where $x_{\bullet j}$ is the j^{th} object belonging to cluster C_i. Let $m_{\bullet}$ be the pooled mean of all objects in X,

$$m_{\bullet} = \frac{1}{n} \sum_{j=1}^{n} x_{\bullet j}. \tag{2.24}$$

One of the most often used criterions is the sum-of-squared error criterion.

Sum-of-squared error Criterion. The square-error J_i for cluster C_i is the sum of the squared Euclidean distance between each object in C_i and its cluster centroid m_i,

$$J_i = \sum_{x_{\bullet j} \in C_i} \left\| x_{\bullet j} - m_i \right\|_2^2 \tag{2.25}$$

$$= \sum_{x_{\bullet j} \in C_i} \left(x_{\bullet j} - m_i \right)^T \left(x_{\bullet j} - m_i \right). \tag{2.26}$$

The square-error, J_e, for the entire clustering containing k clusters is the sum of square-error of the individual clusters,

$$J_e = \sum_{i=1}^{k} J_i . \tag{2.27}$$

The objective of a partitional clustering algorithm based on the sum-of-square-error criterion is to find a partition that minimizes J_e. The resulting J_e has also been referred to as the minimum variance partitional (Jain and Dubes 1988).

B. Partitional Clustering algorithms

***K*mean Algorithm**

The most used partitional clustering algorithm is known as *K*mean algorithms, which may be described as follows (Anderberg 1973).

INPUT: **An $m \times n$ data matrix ($X_{m \times n}$), Number of clusters (k), and k seed points.**

OUTPUT: **k clusters.**

STEP 1 **Begin with an initial partition of the objects into k clusters for a chosen value of k.**

STEP 2 **Take each object in sequence and compute the distances to all cluster centroids; if the nearest centroid is not that of object's parent**

cluster, then reassign the object and update the centroids of the losing and gaining clusters.

STEP 3 Repeat step 2 until convergence is achieved; that is, continue until a full cycle through the objects fails to cause any change in cluster membership.

The idea in *K*mean algorithm is to find a clustering of the objects so as to minimize the total within-cluster sums of squares, given in (2.25) and (2.27). This algorithm sequentially processes each object and reassigns it to another cluster if doing so results in minimizing the total within-cluster sums of the squares. The computational complexity of the *K*mean algorithm for a given *mxn* data matrix is $O(mnkl)$, where m is the number of attributes (rows), n is the number of objects (col), k is the number of clusters desired, and l is the number of iterations. The number of iterations depends on the size of the clustering problem and, in practice, its value is specified by the user (Jain and Dubes 1988, and Cios *et. al.* 1998).

Nucleated Agglomerative Algorithm (NA)

This algorithm provides a convenient way for analyst to look into a range of partitions instead of only one partition as *K*mean does. It is implemented in agglomerative fashion, which generates a range of partitions starting from a given maximum number of initial clusters to a given minimum number of clusters. The envisioned final number of clusters should lie in this range (Mather 1976). This algorithm operates into two phases. In the first phase, it generates the clustering structure corresponding to the given

maximum number of clusters. Phase 2, iteratively merges a pair of clusters until clustering structure corresponds to the given minimum number of clusters is obtained.

INPUT: An $m \times n$ data matrix ($X_{m \times n}$).

OUTPUT: Clustering solutions from the pre-specified maximum number of clusters to the pre-specified minimum number of clusters.

Phase I

Step 1: Specify the maximum and minimum number of clusters, (k_{max}, k_{min}) and choose k_{max} seed points as the set of centroids for the initial partition.

Step 2: For each object $x_{\cdot j}$ compute the Euclidean distance (2.9) to all centroids and allocate $x_{\cdot j}$ to the cluster having the nearest centroid to attain initial partition.

Step 3: Compute the centroids of each cluster using (2.23).

Step 4: Objects are moved in turn to other cluster if the total within square distance is reduced. That's objects j is moved from cluster p to cluster q if

$$\frac{n_q}{n_q+1}\sum\left(x_{\cdot j}-m_q\right)^2 < \frac{n_p}{n_p-1}\sum\left(x_{\cdot j}-m_p\right)^2. \qquad (2.28)$$

If move is made, recompute the centroids of the losing and gaining clusters.

Step 5: Repeat Step 4 until no further move is made. That's, continue until a full cycle fails to move any object.

The final configuration of this phase provides the clustering solution for k_{max}.

Phase II

Step 1: **Reduce k by 1, initially $k = k_{max}$**

Step 2: **Merge the pair of clusters that minimizes the increase in the square deviation of the objects from their cluster centroids, which is defined as follows:**

$$\frac{n_p n_q}{n_p + n_q} \sum (m_p - m_q)^2, \quad p=1, \ldots, k\text{-}1; \quad q=p+1, \ldots, k. \tag{2.29}$$

This formula is computed for all pair of clusters and the pair (p,q) of the minimum are merged into one cluster. The centroid of the new cluster is computed.

Step 3: **Repeat Step 1 and Step 2 until $k = k_{max}$.**

2.3 Principal Component Analysis (PCA) Overview

The principal component analysis (PCA) is one of the widely used multivariate statistical methods in the field of data analysis. This technique is used, traditionally, with more than one purpose in mind. On one hand, it transforms a set of correlated variables into a new set of uncorrelated ones. Also, it reduces a data set containing a large number of variables to a data set having fewer new variables, such that the new variables extract a large fraction of the variance contained in the original variables (Wilks 1995). However, we use this technique to provide us the ability to cluster the variables (objects) in our data set by taking advantage of the wealthy information embedded in the principal component ***loading*** after rotating these loadings using an analytical rotation algorithm. The rotational algorithms generally seek to satisfy the

concept of simple structure (Richman 1986). Gong and Richman (1995, p. 904) consider the rotated PCA (rPCA) to be cluster analysis. They consider the simple structure rotation is the stage that distinguishes the PCA from purely statistical technique; its aim is to maximize variance, from one that looks for finding greatly related groups of correlation vectors in m space (Gong and Richman 1995). The elements of each principal component loading relate the new transformed variables to the original variables. Each principal component loading is obtained by scaling an eigenvector by the square root of its corresponding eigenvalues. Rotating the loadings and examining the resultant structure enables us to assign each variable in the data set to its proper cluster.

2.3.1 Basic equation

Let $X=[x_{ij}]$ $1 \le i \le m$, $1 \le j \le n$, be a $m \times n$ data matrix where i is the observation and j is the variable. The Grammian, Σ, is a real symmetric matrix defined as

$$\Sigma = X^T X \text{ , (usually } \Sigma = \frac{1}{m-1} X^T X \text{).} \tag{2.30}$$

Let (λ_i , v_i) denote the eigenvalue and eigenvector pair for Σ that is,

$$\Sigma v_i = \lambda_i v_i \qquad i = 1, 2, \ldots\ldots, n. \tag{2.31}$$

The system of n equations in (2.31) can be collectively represented as

$$\Sigma V = V \Lambda \tag{2.32}$$

where $V=[v_1, v_2, \ldots v_n,]$ is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues that is,

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \mathrm{K} & 0 \\ 0 & \lambda_2 & & 0 \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ 0 & 0 & & \lambda_n \end{bmatrix}.$$

Since Σ is real and symmetric, V is orthogonal (Basilevsky 1983). Hence

$$V^T V = I = V V^T. \tag{2.33}$$

Thus (2.32) can be rewritten as

$$V^T \Sigma V = \Lambda. \tag{2.34}$$

By substituting (2.30) in (2.34) we get

$$\left.\begin{aligned} V^T X^T X V &= \Lambda \\ (XV)^T (XV) &= \Lambda \end{aligned}\right\} \tag{2.35}$$

Define $Y=XV$. Then (2.35) becomes

$$Y^T Y = \Lambda \tag{2.36}$$

Notice that vectors (columns) in X are correlated but vectors in Y are uncorrelated.

Vectors in Y are the new representation of X with respect to the basis V.

Assuming that Σ is positive definite, we get

$$\left.\begin{aligned} Y^T Y &= \Lambda = \Lambda^{1/2} \Lambda^{1/2} \\ \Lambda^{-1/2} Y^T Y \Lambda^{-1/2} &= I \\ (Y \Lambda^{-1/2})^T (Y \Lambda^{-1/2}) &= I \end{aligned}\right\} \tag{2.37}$$

$$F^T F = I \qquad \text{where} \qquad F = Y \Lambda^{-1/2}.$$

Now consider

$$\left.\begin{aligned} F &= Y\Lambda^{-1/2} \\ &= XV\Lambda^{-1/2} \\ &= X(V\Lambda^{-1/2}) \end{aligned}\right\} \tag{2.38}$$

Thus, we obtain

$$\left.\begin{aligned} X &= F(V\Lambda^{-1/2})^{-1} \\ &= F(\Lambda^{1/2}V^{-1}) \\ &= F\Lambda^{1/2}V^{T} \\ X &= F\,A^{T} \quad \text{where } A^{T} = \Lambda^{1/2}V^{T} \text{ or } A = V\,\Lambda^{1/2} \end{aligned}\right\} \tag{2.39}$$

where A is called the principal component ***loading*** which denotes the new basis for representation and F is an uncorrelated representation of X called principal component ***scores***. The data matrix X could be centered as in (2.3) or normalized as in (2.4) before computing the Grammian, Σ, in (2.30). If X is centered, the Grammian, Σ, has the interpretation of covariance and it has the interpretation of correlation if it is normalized.

The question now is: how can one cluster using PCA? The principal component loading matrix, A, consists of the eigenvectors where each is scaled by the square root of its corresponding eigenvalue. The eigenvectors define an alternative coordinate system to view the data. There are n elements in each column of A where n is the number of original variables which we need to cluster. Each of these elements relates the original variables, X, to the new derived variable, F. For example, when Σ is the covariance matrix, the principal component loading, A, are the covariance coefficients between the principal component scores, F, and the original variables, X. Geometrically, each of the eigenvector is perpendicular to the others and points to a direction in n-dimensional space of X. Without loss of generality it is assumed that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \ldots \geq$

λ_n . The first eigenvector v_1 is associated with the largest eigenvalue λ_1 and points to a direction where the data exhibit the most variability. The second eigenvector v_2 is associated with the second-largest eigenvalue λ_2 and points to a direction where the data exhibit the next most variability and is orthogonal to the first. Others eigenvectors continue in the same fashion in a decreasing order of their associated variance. Therefore eigenvalues have traditionally played a major role in deciding the number r of principal components to retain in order to extract the largest fraction of the variance with the least number of variables. There are several rules to decide the value of r. A graphically based rule, called ***scree test***, is a plot of the eigenvalues against the corresponding root number. The idea of this test is to locate a point in the plot that separates a steeply sloping portion to the left from the rest of the plot. The root number prior to which the separation occurs is taken to be the value of r (Wilks 1995). Another rule is to define a threshold, say 0.9, corresponds to the percentile of the total variance one wants to extract. Then the value of r is the number, which makes the ratio of the sum of the first r eigenvalues to the sum of all eigenvalues, exceeds the threshold.

$$\frac{\sum_{i=1}^{r} \lambda_i}{\sum_{j=1}^{n} \lambda_j} \geq 0.9. \tag{2.40}$$

Once the value of r is computed, then these columns of the matrix, A, are transformed to new matrix, B, using an analytical rotation algorithm (section 2.3.2). The new matrix B is called the rotated principal component loadings. We cluster the variables by examining the loadings of each variable in the matrix B across all r

retained PCs. The variable is assigned to the PC (or cluster) on which it loads highly and low on other PCs.

2.3.2 Rotation

Our goal of using PCA is neither to maximize the variance nor to compress the data set as the traditional applications of the PCA. It is to recover the underlying clustering structure of the variables by physical interpretation of the principal component loadings A. To achieve this goal, it is often desirable to rotate r components of the principal component loadings of the initial solution to new set of coordinate system. The number r is usually determined by one of the truncation criteria.

The principal component loadings A constitute, geometrically, the coordinates of n points in the loading space, where n is the number of variables. The loadings of the i^{th} row in A constitute the coordinates of a point in the loading space corresponding to the i^{th} variables. By rotating these n points we hope to find a new frame of reference in which the PCs are more interpretable by recovering the clustering structure. In this new frame of reference, we want as many points (or variables) to have near zero projection on the PC axes. Geometrically, if there are clusters of points for this project, we want to rotate the axes so as to pass through or near these clusters. Therefore, each cluster of points would be associated with only one PC axis, hence the clustering structure of these points (or variables) would be obtained. To achieve this, the rotational algorithms seek to produce what is known as simple structure (s.s.). The basic idea of s.s. is to produce rotated vector solution, B, whose loading coefficients are as close to 0 or a salient non-zero value as possible. The idea of the s.s. is attributed to L. Thurstone and its requirements are as follows (Harman 1976, Richman 1986, and Jackson 1991):

1. Each row of B must have at least one zero.
2. Each column of B must have several zeros, minimum is r and the more zeros in a column (up to n-1) the better.
3. For each pair of columns of B, there should be zeros in different rows.
4. If $r \geq 4$, there should be a large number of variables with zeros in both columns and a small number of variables with nonzero loadings in both columns.
5. There should be small number of loadings out of the hyperplane (defined below). Variables are considered to be in the hyperplane if their loadings fall within a certain narrow band on either side of zero.

An exact zero loading is unlikely to exist in any analysis due to the effect of sample errors and other noise on the magnitude of the loadings. For this reason, the notion of hyperplane width is suggested. The hyperplane width (say ± 0.20) is the narrow range on either side or zero. Variables that have loading within this range on one PC are termed "in the hyperplane" of this PC (Richman 1986). Loadings in the hyperplane on a PC loading vector are treated as zero and not considered for cluster membership for that particular vector in the analysis. A proper choice of the hyperplane width is essential for the interpretation of which variables cluster in a principal component analysis. A recent paper by Richman and Gong (1999) aimed to objectify the problem of best identifying the hyperpalne width. Their paper shows that 0.25 is an appropriate hyperplane width for winter precipitation and geopotential height correlation-based PCs. In our analysis, loadings that fall within ±0.25 range are considered in the hyperplane, hence they are treated as zero in the simple structure.

A variable that loads moderately on all PCs (i.e., is not in the hyperplane of any PC) is called complex variable. Variables that load highly on one PC and "in the hyperplane" of other PCs are grouped in one cluster. Complex variable, however, are not assigned to one cluster since their loadings are moderate on all PCs. They introduce overlapping and fuzziness in the clustering solution.

The aim of the simple structure is to produce a new set of components, each one involving a cluster of the original variables with as little overlaps as possible (Jackson 1991). The ideal simple structure is difficult to obtain directly from the 5 rules above, so the rotational algorithms use various criteria to objectify Thurstone's requirements for simple structure.

The degree of resultant simple structure is determined using the notion proposed by Richman (Richman 1986). He qualifies the simple structure as strong, moderate, weak, and random. In the case where there are many variables in the hyperplane with distinct clusters of variables and few complex variables, the simple structure is strong. The simple structure is moderate where there are slightly less variables in the hyperplane than the first case and more complex variables. In case where there are good number of complex variables and indistinct clustering, the simple structure is characterized as weak. The random simple structure is the one with many complex variables and no clusters. These four types that represent the amount of simple structure present in the analysis are schematically illustrated in Richman (1986, p. 312).

The rotational algorithm finds the rotation matrix T by calculating the angle of rotation θ. The rotation matrix T is used to transform the unrotated principal component loadings matrix A to rotated matrix (simple structure) B.

$$B = A\,T \tag{2.41}$$

The matrix B is determined in the rotational algorithm by application of a criterion to ensure the quality of the simple structure. If the criterion is achieved then matrix B is chosen to be the simple structure, otherwise other rotation matrix T is sought by calculating new angle for rotation. The rotation process is called *orthogonal rotation* if the rotation matrix T is orthogonal; that is, if T transforms the unrotated principal component loadings orthogonally to simple structure. The rotation is called *oblique rotation* if T does not transform the unrotated loadings under the orthogonality constraint (that is, T is not orthogonal).

The proposed clustering algorithm using the rotated principal component analysis is now described as follows:

INPUT: **An $m \times n$ data matrix ($X_{m \times n}$).**

OUTPUT: **Clustering solution consists of r clusters, where r is the number of PCs retained in the analysis.**

Step 1: Caiculate correlation matrix R using (2.12)

Step 2: Calculate principal component loading matrix A using (2.39)

Step 3: Rotate r columns of A to simple structure B using (2.41) under the constraint that T is orthogonal (quartimax) and without the orthogonality constraint on T (promax).

Step 4: Examine the loadings in the simple structure (matrix B). Assign each variable to the PC (or cluster) on which it loads highly and low on the other PC.

The goal of using the rotated principal component analysis (rPCA) in this research is to obtain non-overlapping (hard) clustering. Hence, the quartimax (defined below) criterion is used to achieve the rotation of the PC components. Quartimax criterion emphasizes on simplifying the rows of the PC loadings matrix A. Simplifying the rows implies hard clustering which is the goal of this study. In step 3 of the above algorithm, different numbers of PCs are rotated first using quartimax. If strong simple structure is resulted by quartimax, it is taken as the clustering solution. Otherwise, promax (defined below) is used. The simple structure that has the least amount of overlapping is then chosen as the clustering solution.

A. Orthogonal rotation (quartimax)

Orthogonal rotation is the rotation that resulted from transforming the principal component loading matrix A into new matrix B using an orthogonal matrix T. For every variable in X, say $x_{\bullet j}$, the orthogonal rotation in the plane of the principal components s and t through an angle θ transforms the original coordinates (a's) into the new rotated coordinates (b's) according to the following equations:

$$\left.\begin{aligned} b_{js} &= a_{js}\cos\theta + a_{jt}\sin\theta \\ b_{jt} &= -a_{js}\sin\theta + a_{jt}\cos\theta \end{aligned}\right\} \qquad (2.42)$$

In the orthogonal rotation, the communality of any variable is invariant, that is

$$\sum_{s=1}^{r} b_{js}^2 = \sum_{s=1}^{r} a_{js}^2 = h_j^2 \qquad j=1,2,\ldots,n \qquad (2.43)$$

Communality is defined as the amount of the j^{th} variable variance accounted for in the analysis by retaining r PCs. In other words, the communalities of all variables constitute the variability explained by the PCA model.

The aim is to find the angle of rotation θ_{st} for any pair of principal components s and t that will achieve the criterion of the rotational algorithm. Then the product of the transformation of all combinations of the pairs of principal components produces the rotated principal component loading matrix B:

$$B = AT_{12}\, T_{13} \ldots\ldots T_{st} \ldots\ldots T_{(r-1)r} \qquad s=1,2, \ldots, (r-1), \quad t=s+1, s+2, \ldots, r \quad (2.44)$$

The full set of all $\frac{1}{2}r(r-1)$ combinations of pairs of s and t is called *iteration* (Harman 1976). One of the well known orthogonal rotation algorithm is the quartimax algorithm which emphasizes on simplifying the rows of the principal component loadings matrix A in order to attempt to meet Thurstone's requirements for the simple structure. In quartimax, the maximum simplicity of the entire loadings matrix A is then defined as the maximization of the following criterion (Harman 1976):

$$Q = \sum_{j=1}^{n}\sum_{s=1}^{r} b_{js}^{4} \qquad (2.45)$$

where n is the number of variables, r is the number of components, and the b's are the loadings.

The computation procedure of the quartimax is now described. The components (PCs) are rotated 2 at a time according to (2.44), and then the complete iteration is repeated until the value of Q (2.45) is maximized to a given convergence value. The angle of rotation θ, which maximizes (2.45), for a pair of components s and t is calculated as follows (Harman 1976):

$$\theta = \frac{1}{4}\arctan\left(\frac{2\sum_{j=1}^{n}(2a_{js}a_{jt})(a_{js}^{2}-a_{jt}^{2})}{\sum_{j=1}^{n}\left[(a_{js}^{2}-a_{jt}^{2})^{2}-(2a_{js}a_{jt})^{2}\right]}\right) \qquad (2.46)$$

Clustering using rotated principal component analysis is done in this research by first rotating the matrix A orthogonaly using quartimax in the hope of achieving strong simple structure. If orthogonal rotation does not produce such simple structure, the oblique rotation using promax method is then applied to obtain better simple structure without the orthogonality constraint.

B. Oblique rotation (promax)

Oblique simple structure is the one that results from rotation using nonorthogonal rotation matrix T. This, in fact, is a tradeoff between more simple structure and the loss of orthogonality. Hendrickson and White (1964) proposed quick method for rotation to oblique simple structure called *promax*. The idea of promax is to use orthogonal rotation (quartimax, in this study) in order to obtain an approximate first guess at simple structure. Promax, then, raises that guess to some power>1 so that the mid-range loadings get small. That is to say orthogonality leads to a less than ideal simple structure with good number of mid-range loadings, so that by decreasing the number of mid-range loading, one has better simple structure.

The Promax method starts by rotating the matrix of the initial unrotated loadings A to orthogonal simple structure B^*. The rows of B^* are then normalized by the square roots of their communalities h_j where $h_j^2 = \sum_{s=1}^{r} b_{js}^2$. This normalization produces a matrix denoted as L where,

$$l_{ij} = b_{ij}^*/h_i \qquad i=1,2,\ldots,n, \quad j=1,2,\ldots,r .$$

Each column of the matrix L is then normalized by the absolute value of its largest element. The resulted matrix, denoted as U, has the value of 1.0 as the absolute

value of the largest elements. Other elements in each column are scaled by the absolute value of its largest element. The pattern matrix, P, is then defined as the elements of the matrix U raised to a power greater than 1 but retaining the same signs as the original elements.

$$p_{ij} = \frac{\left| u_{ij}^{k+1} \right|}{u_{ij}} \tag{2.47}$$

where $k>1$. A power of 4 is commonly used in (2.47) for non-overlapping clustering. The rationale of the pattern matrix, which defines the ideal simple structure, is that the number of mid-range loading is reduced and the small loadings are forced into the hyperplane by making their magnitudes smaller. The problem now is to find the least square fit of the orthogonal matrix B^* to the pattern matrix P obtained by (2.47). A transformation matrix that transforms B^* to P is needed.

$$T_1 = \left(B^{*T} B^*\right)^{-1} B^{*T} P \tag{2.48}$$

where T_1 is the unnormalized transformation matrix. This equation is the well-known Procrustes equation where B^* is the matrix to be transformed and P is the target matrix (Richman 1986). The columns of T_1 are normalized such that the sums of the squares of their elements are equal to 1.0. This normalization produces the transformation matrix T. The desired oblique simple structure B is then computed as

$$B = B^* T \tag{2.49}$$

The intercorrelations among the components of B can be computed as the inverse of the $T^T T$ matrix.

2.4 Summary and Concluding remarks

In this chapter, the literature related to cluster analysis and principal component analysis is reviewed. A description of the properties of the data matrices of this research and how they are constructed is given in this chapter. This chapter presents the general framework for the hierarchical cluster analysis and reviews the common and new developed resemblance measures. Four hierarchical clustering algorithms, single linkage, complete linkage, average linkage, and Ward's are reviewed. Also, two none hierarchical clustering algorithms, *K*mean and nucleated agglomerative are discussed. In addition, the basic idea of the principal component analysis is discussed and two rotation algorithms, quartimax and promax, are reviewed.

CHAPTER 3

ANALYSIS OF THE UNI-MODEL ENSEMBLE

3.1 Introduction

In this section, we describe the domain and the nature of the data set for the uni-model short-term ensemble, and present the result of applying clustering methodologies to various meteorological fields of interest in ensemble forecasting. Table 3-1 lists the fields we analyze along with their abbreviations and units. These fields are suggested by a group of meteorologists and researchers at the National Severe Storm Laboratory (NSSL) and the Storm Prediction Center (SPC) in Norman, Oklahoma [see the acknowledgment]. The data set for these fields are the output of the Eta forecasting model over the domain starting on 12/2/1997.

Table 3-1: List of all fields analyzed in this study along with their abbreviations and levels (SFC stands for surface level).

Field	Abbreviation	Unit
Precipitation at SFC	PRCP	kg/m^2 *
Pressure at SFC	PRMSL	Pa
Height at 500 mb	HGT	gpm
Temperature at SFC	TMP	K
Wind at 250 mb	WIND	m/s
Absolute vorticity at 250 mb	ABS V	1/s
Absolute vorticity at 500 mb	ABS V	1/s
Absolute vorticity at 850 mb	ABS V	1/s
Pressure vertical velocity at 700 mb	V VEL	Pa/s
Convective available potential energy	CAPE	J/kg

* The accumulative amount of total precipitation for 12 hours period

3.2 A Description of the Ensemble Data

The domain consists of 93x65 grid points (total of 6045 points over the North American continent) where each field is measured at each grid point. The latitudes and longitudes of the four corners of the grid are shown in Figure 3-1.

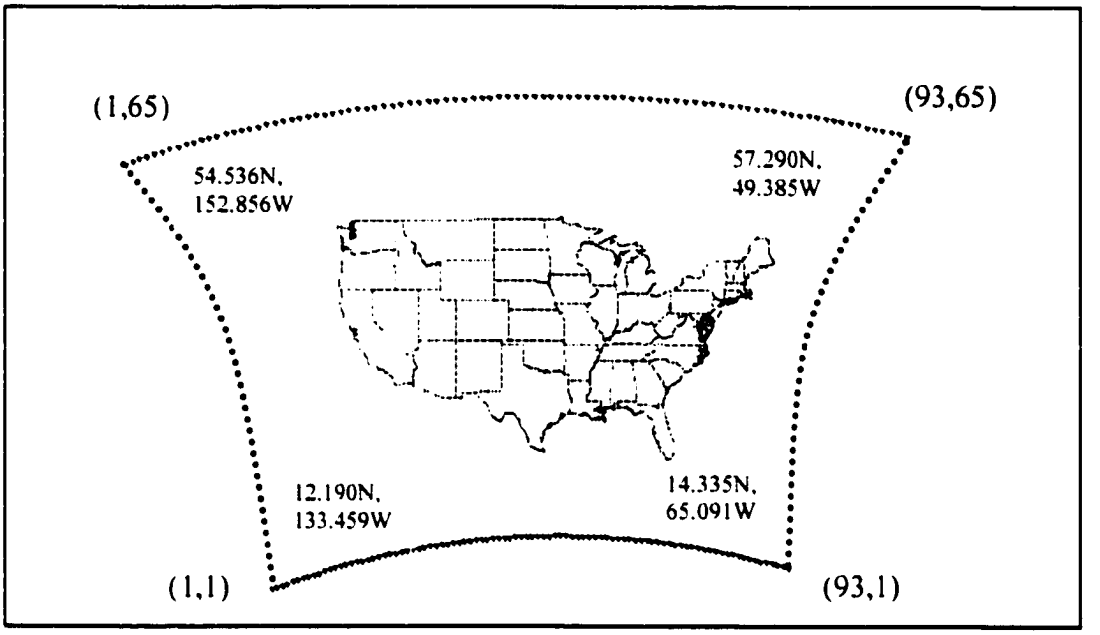


Figure 3-1: The domain of data sets of the fields analyzed in the uni-model ensemble.

The data set for each field, in Table 3-1, represents the output of the Eta model over the North America on 12/2/97 starting at 12:00 PM. The ensemble consists of 10 members, and runs for 48 hours. The short descriptions of the 10 members of the ensemble are given in Table 3-2. The values of each field for these 10 members are computed by the Eta model at nine (9) output times – 0^{th}, 6^{th}, 12^{th}, 18^{th}, 24^{th}, 30^{th}, 36^{th}, 42^{nd}, and 48^{th} hour with an interval of 6 hours apart.

For each field, nine (9) different data matrices are constructed, each of which represents one output time, and the PCA and CA analysis is conducted for each one separately. The precipitation data are available for only 4 output times: the 12^{th} hour, 24^{th} hour, 36^{th} hour, and 24^{th} hour. Each data matrix, $X_{6045x10}$, represents the quantities of one field for the 10 members at one output time. Thus, a variable (member) is

represented by a column with 6045 elements with one for each grid point. The words variable and member are used interchangeably.

Table 3-2: Short descriptions of the 10 members of the ensemble.

Member Number	Description
1	avnt126
2	cnt62
3	edas_erl
4	edas_fnl
5	ngmanl
6	noone62
7	ntwot62
8	opnl80
9	ponet62
10	ptwot62

The grid points are numbered from the left to the right and from the bottom to the top of the grid, Figure 3-1. So, the first 93 rows in the data matrix (rows 1 to 93) correspond to the lowest row of points in the grids that's points (1,1) to (93,1). The second 93 rows in the data matrix (rows 94 to 186) correspond to the points in the second row from the bottom of the grid that's points (1,2) to (93,2). And the last 93 rows in the data matrix (rows 5952 to 6045) correspond to to the points in the highest row of points in the grids that's poins (1,65) to (93,65). The value x_{ij} is the value of the field under study measured at the i^{th} station (grid point) for the j^{th} member. The objective of these analyses is to study the behavior of the ensemble and see how these members cluster.

3.3 Analysis of the precipitation

In this section, we present our results of analyzing the data set for precipitation field (PRCP) at the surface level. Recall, from the previous section, that the precipitation data

are available at only four output times, 12th, 24th, 36th, and 48th hours. So there is one data matrix, $X_{6045x10}$, for each one of these output times. The cluster analysis (CA) analysis and the principal component analysis (PCA) are conducted first, for the precipitation data, based on the correlation matrices (2.12), R_{10x10}, and second based on the Euclidean Similarity matrices (2.16), $\dot{E}_{10x10}$. Then, we apply CA based on dissimilarity measure, Euclidean distance (2.9), E_{10x10}.

3.3.1 CA and PCA based on the correlation

The correlation matrix (2.12), R_{10x10}, of the 10 members is first computed for each output time and then CA is conducted. Table 3-3 shows the correlation matrix for the precipitation data at the 48th hour.

Table 3-3: The correlation matrix for the precipitation data at the 48th hour.

	1	2	3	4	5	6	7	8	9	10
1	1.00									
2	0.87	1.00								
3	0.75	0.82	1.00							
4	0.68	0.73	0.65	1.00						
5	0.67	0.71	0.73	0.62	1.00					
6	0.77	0.80	0.70	0.51	0.58	1.00				
7	0.67	0.70	0.67	0.50	0.58	0.67	1.00			
8	0.82	0.84	0.74	0.82	0.68	0.67	0.59	1.00		
9	0.75	0.76	0.79	0.57	0.64	0.68	0.77	0.64	1.00	
10	0.71	0.69	0.68	0.49	0.64	0.69	0.56	0.62	0.69	1.00

The cluster dendrograms resulted from applying UPGMA method (section 2.2.1) are shown in Figure 3-2a. The clustering trees at all output times for each field are drawn using the same scale (0 to 1 for the correlation-based). Hence, user can assess, visually, the divergence among the members by examining the height of trees as the time goes. The height of the dendrograms in Figure 3-2a is increasing as the time passes which

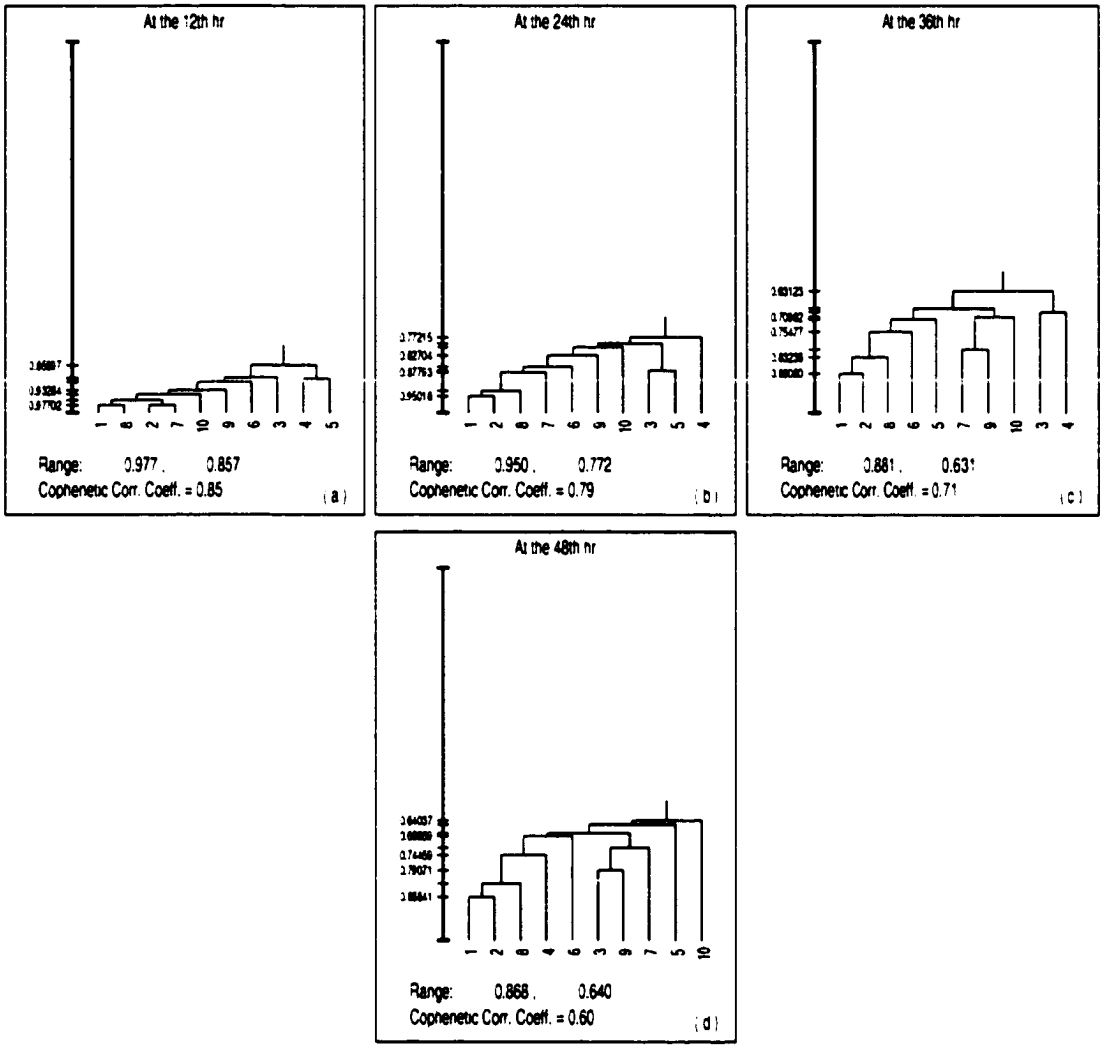


Figure 3-2a: UPGMA dendrograms for the Precipitation field based on the Correlation.

indicates that the coherence is decreasing and divergence is increasing as the time passes. In this figure, the highest tree, which occurs at the 48th hour, merge all variables into one cluster at a correlation level of 0.64. This, in fact, is considered high correlation.

Using the same correlation matrices, the PCA (section 2.3) is then conducted. Table 3-4 shows the variance explained by the first and second eigenvalues for each output time. It is manifest from the table that, as the time passes, the first eigenvalue

decreases and the other contributes more to the variance. This indicates that data exhibit less coherence and more divergence as the time passes. This observation agrees with the increasing heights of the clustering trees in the CA results mentioned above.

Table 3-4: Total variance explained by the first and second eigenvalues of the correlation matrices.

	12^{th} hour		24^{th} hour		36^{th} hour		48^{th} hour	
	i	Variance explained by λ_i	*i*	Variance explained by λ_i	*i*	Variance explained by λ_i	*i*	Variance explained by λ_i
	1	91.19%	1	83.57%	1	72.16%	1	72.08%
	2	3.27%	2	4.04%	2	7.46%	2	7.28%
Total variance explained by λ_1 and λ_2		95.17%		87.61%		79.62%		79.36%

Rotation using quartimax (section 2.3.2 A) indicates that the members are all clustered on one PC, Table 3-5. As the table shows, variable number 4 does not load highly on only one PC; instead it loads moderately on the second PC too. Although the aim here is to associate the variables to the PC on which their loadings are larger (hard clustering), one should be cautious when clustering variables such as variable number 4. The clustering of such variable is not as definite as other variables due to the fact that such variables overlap between two clusters. For this data, rotating more than 2 PCs using quartimax or even using promax does not change the solution. That is, members are all clustered on one PC.

The PCA solution agrees with the CA results with respect to the degree of correlation between the members. The PCA placed all members into one clusters and

CA merge them all into one cluster at relatively high degree of correlation (0.64). However, even under this situation the CA is able to discriminate between the 10 members and produce different clusters (Figure 3-2a (d)). This difference is due to the ability of the CA to distinguish between different clusters even if the variables are all on one dimension.

Table 3-5: The correlation-based rotated loadings of the first 2 PCs using quartimax for the precipitation data at the 48^{th} hour.

n	PC 1	PC 2
1	0.91	0.07
2	0.93	0.10
3	0.89	-0.01
4	0.76	0.56
5	0.80	0.13
6	0.84	-0.21
7	0.80	-0.31
8	0.87	0.38
9	0.87	-0.25
10	0.80	-0.21

When complex variables are found in the simple structure or when some variables load moderately on more than one PC as the case in Table 3-4, the process of hard clustering variables using PCA becomes difficult. Such variables (members) do not load highly on only one PC which makes the process of associating these variables to one cluster is subjective and introduces a fuzziness in the resultant clustering structure. The hierarchical cluster analysis, on the other hand, overcomes this problem by producing a nested hierarchy of clusters (dendrogram). The clustering dendrogram makes the process of clustering easier, but it fails to explain how fuzzy the solution is since the information about overlapping is not provided by the CA method.

3.3.2 CA and PCA based on the Euclidean Similarity

The purpose of this analysis is to provide a dissimilarity-based PCA analysis for the sake of comparison since the Euclidean Similarity (EucSimi; section 2.2) is derived from a dissimilarity measure (i.e., the Euclidean distance).

For each output time, the data matrix is first centered (2.3) and then the Euclidean similarity matrix (2.16), $\dot{E}_{10\times10}$, is computed. Based on the $\dot{E}$ matrix, CA using UPGMA method (section 2.2.1) is applied. The results are then compared to the correlation based CA. Figure 3-2b shows the clustering dendrograms resulted from applying UPGMA method based on the EucSimi for the precipitation data. Since the clustering trees are all drawn using the same scale [0,1]. It can be easily seen that the level at which the first cluster is formed is close to 50% of the range of the EucSimi measure (i.e., [0,1]). This points out the ability of this measure to recover the differences between the members of the ensemble. By comparing this result to the correlation-based for this data set, Figure 3-2a where all members are merged into one cluster at 0.64 degree of correlation, we can see the improvement obtained by using EucSimi as a similarity measure over the correlation with respect to distinguishing the various clusters. However, the clustering structure may not be the same due to the sensitivity of the CA to the change of the resemblance coefficient.

PCA is then conducted using the Euclidean similarity matrices, $\dot{E}$, for the precipitation data. Table 3-6 shows the variance explained by the first and second eigenvalues for each output time. Rotating 4 PCs using the quartimax (section 2.3.2 A) produces strong simple structure that contains no complex variable.

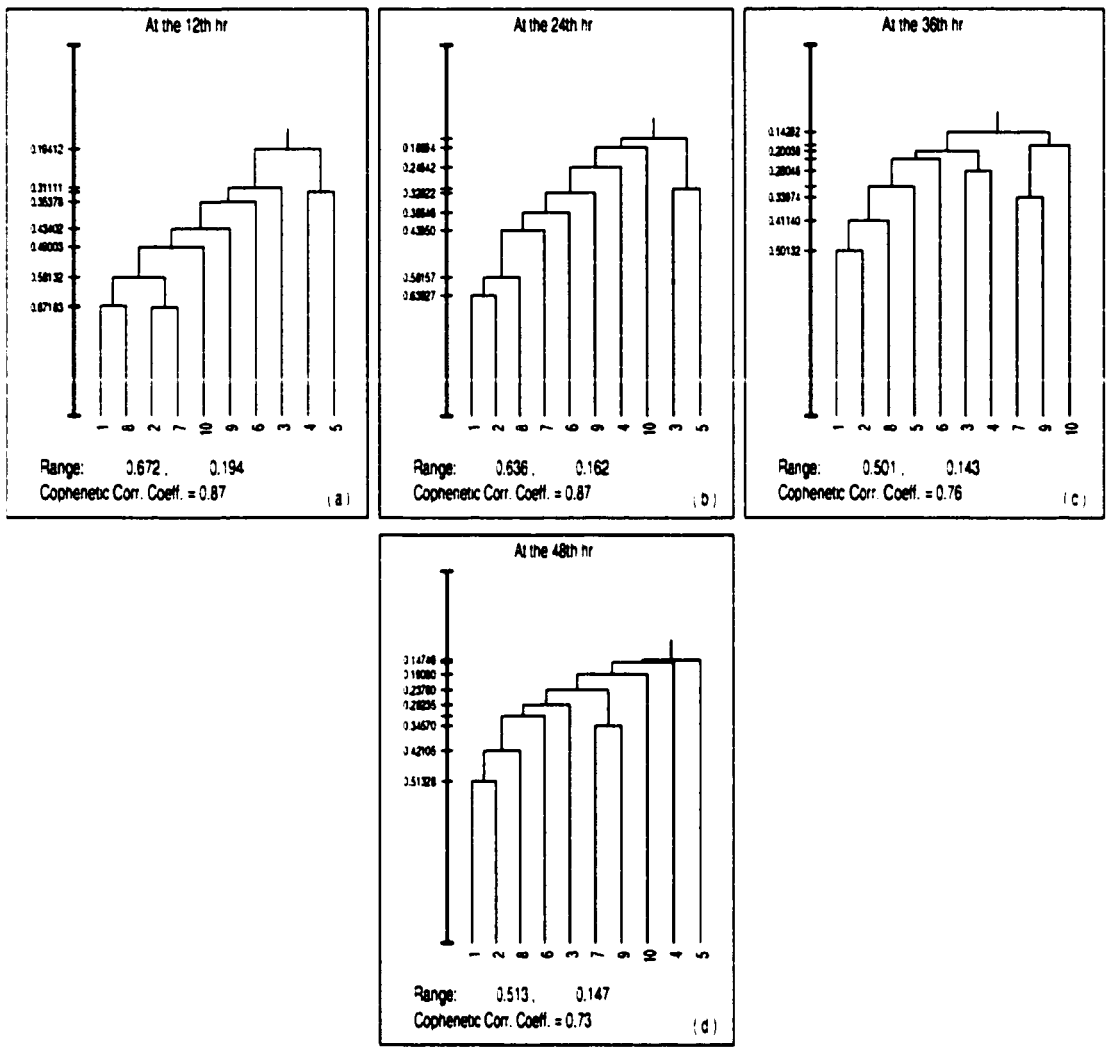


Figure 3-2b: UPGMA dendrograms for the Precipitation field based on the Euclidean Similariy; data is centered using (2.3).

Table 3-6: Total variance extracted by retaining 4 PCs for EucSimi-based PCA.

	12^{th} hour		24^{th} hour		36^{th} hour		48^{th} hour	
	i	Variance explained by λ_i	i	Variance explained by λ_i	i	Variance explained by λ_i	i	Variance explained by λ_i
	1	43.26%	1	36.33%	1	29.42%	1	31.49%
	2	13.04%	2	11.99%	2	13.40%	2	12.75%
Total variance explained by λ_1 and λ_2		56.30%		48.32%		42.82%		44.24%

By examining the quartimax simple structure, Table 3-7, the clustering structure of the 10 members at the 48th hour is as follow: PC1 clusters 7 and 9; PC2 clusters 1, 2, 4, and 8; PC3 clusters 3 and 5; and PC4 clusters 6 and 10. Note that the first and third variables do not load highly on only one PC, instead they overlap between two clusters. The Euclidean Similarity-based PCA solution is different than the correlation-based PCA solution (section 3.3.2). The latter placed all variables into one cluster. On the other hand, both Euclidean Similarity-based PCA and CA (Figure 3-2a (d)) place members 1, 2, 4, and 8 into one cluster.

Table 3-7: EucSimi-based rotated loadings of the first 4 PCs using quartimax for the precipitation data at the 48th hour.

n	PC 1	PC 2	PC 3	PC 4
1	0.31	**0.56**	0.05	0.44
2	0.34	**0.60**	0.14	0.35
3	0.42	0.25	**0.46**	0.15
4	-0.05	**0.78**	0.05	-0.22
5	0.03	0.13	**0.87**	0.08
6	0.33	0.24	-0.22	**0.60**
7	**0.83**	0.05	-0.07	-0.03
8	0.02	**0.79**	0.08	0.11
9	**0.71**	0.08	0.21	0.18
10	-0.02	-0.01	0.18	**0.82**

3.3.3 CA based on the Euclidean distance.

The Euclidean distance measure (2.9) views the members as points in *m*-dimensional space and clustering methods cluster them based on their distances form each other. For each output time, the Euclidean distance matrix, $E_{10\times10}$, is computed for the difference field (2.3), and then used as the dissimilarity matrix for the CA method. Figure 3-2c shows the clustering dendrograms resulted from applying Ward's (section 2.2.1) method on the precipitation data. The nature of Euclidean distance makes it difficult to

assess the degree of divergence between the ensemble members by just looking at the tree of one specific output time. To overcome this problem we draw the clustering trees at all output times for each field using the same scale. Before drawing the clustering dendrograms for all output times of one forecast field, the height of tree H is computed as follow.

Let d_t be the degree of the last merge in the clustering algorithm (i.e., the distance at which all objects are placed in one cluster) at time t.

The height of tree, H is computed as follow.

$$H = \max_{\forall t} \{d_t\} \tag{3.1}$$

That's, the height H is the maximum degree across all output times for the forecast field under study, hence the scales of all trees for one field is the same, [0, H]. Using this method to draw the clustering trees, one can visually assess the divergence among ensemble members with respect to the time by examining the trees height and the level of merges as the time passes toward the end. The dendrograms in Figure 3-2c enable the analyst to study the behavior of the members and identify various clusters by cutting the dendrogram at different levels of dissimilarity. These analyses are very helpful in studying the behavior of the members. Also, they enable us to recover the clustering structures.

3.3.4 CA based on the *K*mean and the Nucleated Agglomerative (NA) algorithms

*K*mean and the Nucleated Agglomerative (NA) algorithms are partitional clustering techniques, Section 2.2.2 B. For such algorithms, the number of clusters and the seed point for each cluster need to be specified in advance.

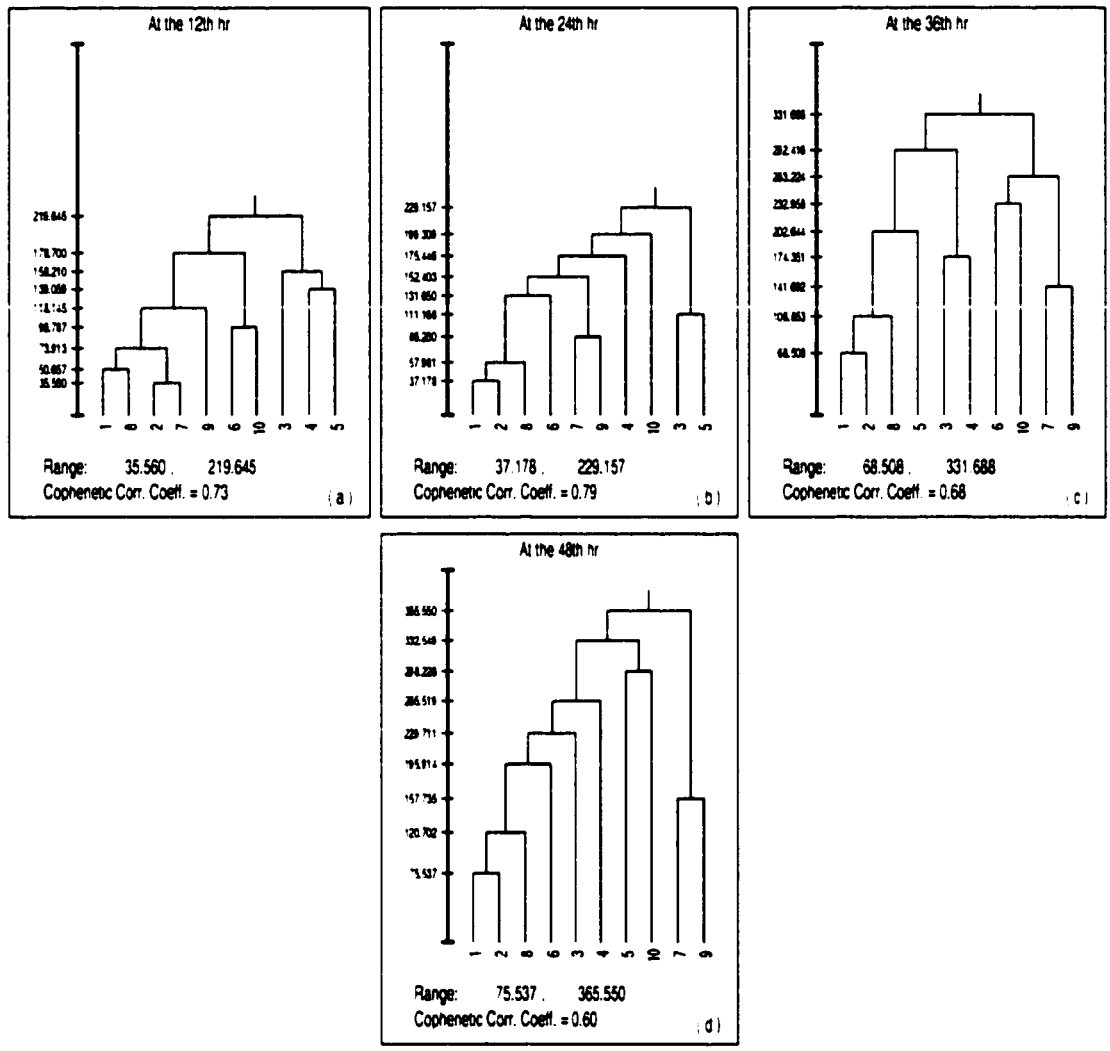


Figure 3-2c: Ward dendrograms for the Precipitation field based on the Euclidean distance; data is centered using (2.3).

The *K*mean algorithm is applied for the precipitation data at the finishing time (48^{th} hour). The number of clusters is specified to be 3, k=3, and the seed points are specified subjectively by examining the Ward's clustering tree for the 48^{th} hour, Figure 3-2c. The resultant *K*mean partition for the finishing time is shown in Table 3-8.

In a similar manner, NA is applied for the same data set, and the number of clusters is chosen to be 3, k=3. The NA clustering solution for the 48^{th} hour is shown in Table 3-9. It is clear from the tables that the two algorithms lead to the same partition

with respect to 3 clusters of each. The common observation that arises from the analysis of the precipitation data is that all clustering algorithms applied to these data agree on placing the members 1, 2, and 8 into one cluster at the finishing time (48^{th} hour).

Table 3-8: Clustering structures resulted from applying *K*mean algorithm to precipitation data at the 48^{th} hour.

Clustering structure		
[1, 2, 4, 8]	[5]	[3, 6, 7, 9, 10]

Table 3-9: Clustering structures resulted from applying NA algorithm to precipitation data at the 48^{th} hour.

Clustering structure		
[1, 2, 4, 8]	[5]	[3, 6, 7, 9, 10]

3.4 Analysis of Pressure

In this section, we present our results of analyzing the data set for pressure field (PRMSL) at the surface level. The data for this field is available at all nine output times, 0^{th}, 6^{th}, 12^{th}, 18^{th}, 24^{th}, 30^{th}, 36^{th}, 42^{nd}, and 48^{th} hours. So, we have nine data matrices, $X_{6045x10}$, each for one output time. The CA analysis and PCA analysis are conducted for each data matrix separately. First, we applied CA and PCA based on the correlation and second based on the Euclidean Similarity. Then, we apply CA based on dissimilarity measure, Euclidean distance.

3.4.1 CA and PCA based on the correlation

The correlation matrix (2.12), R_{10x10}, of the 10 members is first computed for each output time and then CA is conducted. Figure 3-3a shows the clustering dendrograms resulted from applying UPGMA method (section 2.2.1) for pressure data. These dendrograms show the members to be extremely correlated; hence, no clustering structures of the members are recovered.

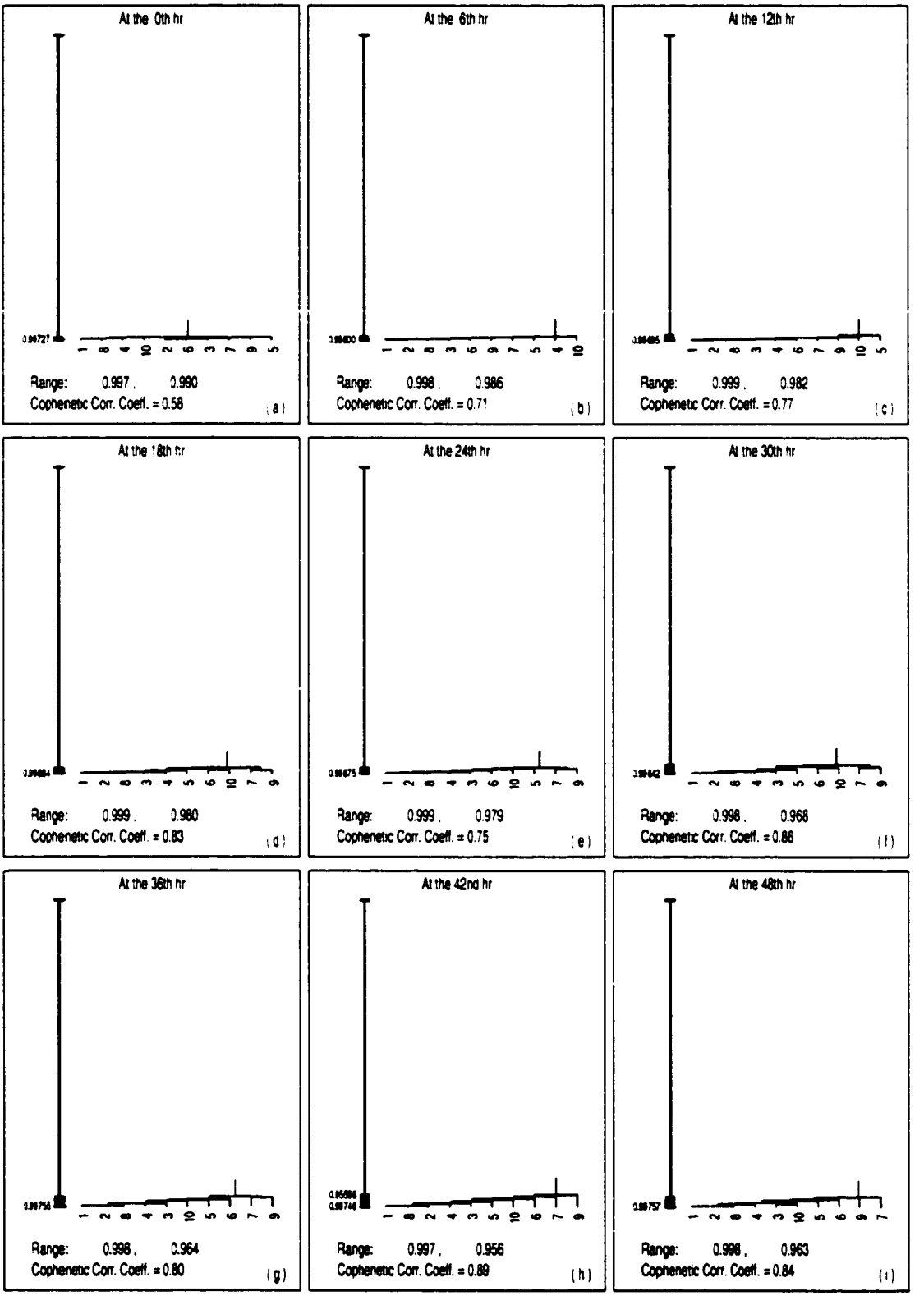


Figure 3-3a: UPGMA dendrograms for the Pressure field based on the Correlation.

For the PCA analysis (section 2.3), eigenvalues for each correlation matrix are computed. One common property about this field is that the first eigenvalue for every correlation matrix is extremely large comparing to the others, and hence the total variance is almost explained by the first PC, Table 3-10. This, in fact, indicates that the variance in the data is mostly concentrated in one direction and the data are gathered around the first PC. Since 97% or more of the total variance can be extracted by retaining only one PC, then all members are placed into one cluster. This, in fact, is total agreement with the CA results.

Table 3-10: First eigenvalue and variance explained by the first PC (correlation-based) at each output time for the pressure data set.

Output time	λ_1	Variance explained by λ_1 (%)
0	9.92	99.2%
6	9.91	99.1
12	9.91	99.1
18	9.87	98.7
24	9.86	98.6
30	9.80	98.0
36	9.78	97.8
42	9.74	97.4
48	9.78	97.8

Rotation using quartimax (section 2.3.2 A) indicates that the members are all clustered on one PC, Table 3-11. As the table shows, all variables are entirely load on the first PC. For this data, rotating more than 2 PCs using quartimax or even using promax does not change the solution. That is, members are all clustered on one PC.

This observation let us believe that when the variance explained by the first eigenvalue is extremely large, then all variables are extremely correlated and form only

one cluster. Therefore, when the aim is to recover non-overlapping clustering structure then evaluating the variance explained by the first eigenvalue decide whether or not one can rotate well if $\lambda_1 >> \lambda_2$.

Table 3-11: The correlation-based rotated loadings of the first 2 PCs using quartimax for the pressure data at the 48th hour.

n	PC 1	PC 2
1	1.00	-0.02
2	1.00	0.01
3	0.99	-0.09
4	0.99	-0.05
5	0.98	-0.08
6	0.98	0.08
7	0.98	0.20
8	1.00	-0.02
9	0.98	0.05
10	0.99	-0.06

3.4.2 CA and PCA based on the Euclidean Similarity

For each output time, the data matrix is first centered (2.3) and then the Euclidean similarity matrix (2.16), $\acute{E}_{10\times10}$, is computed. Based on the $\acute{E}$ matrix, CA using UPGMA method (section 2.2.1) is applied and the results are shown on Figure 3-3b. These results are compared to the correlation-based CA for this data set (section 3.4.1). As the figure shows, the improvement that is obtained by using the Euclidean Similarity as a similarity measure over the correlation is manifest for this data set. Euclidean Similarity is able to discriminate the members and assign them to various clusters whereas the correlation measure is unable to reveal any clustering structure.

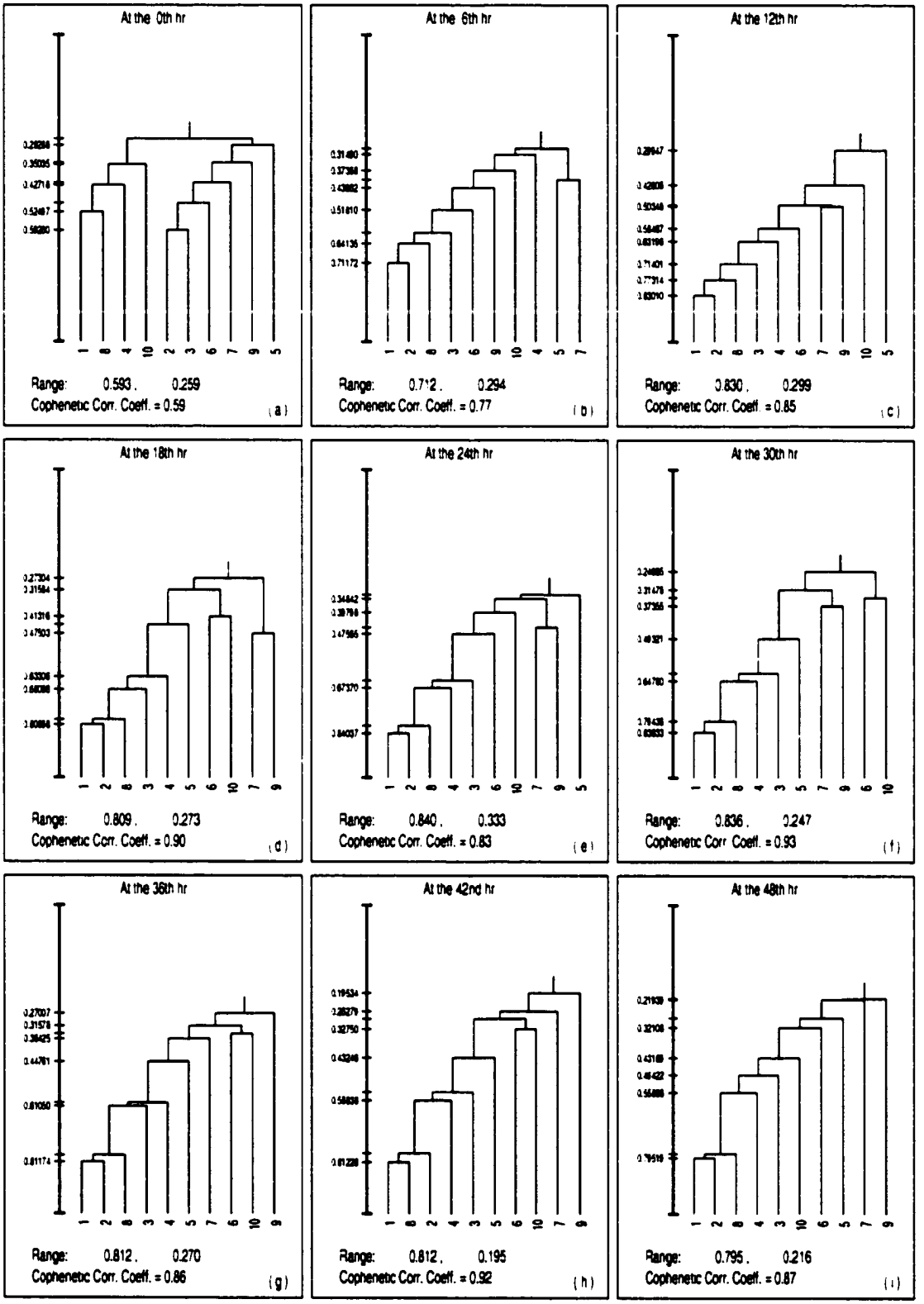


Figure 3-3b: UPGMA dendrograms for the Pressure field based on the Euclidean Similarity; data is centered using (2.3).

The pressure data set is then analyzed using PCA (section 2.3) based on the Euclidean similarity matrices. Rotating 3 PCs using the quartimax (section 2.3.2 A) leads to strong simple structure. By examining the quaritmax simple structure, Table 3-12, the clustering structure of the 10 members at the 48th hour is as follow: PC1 clusters 1, 2, 3, 4, 8 and 10; PC2 clusters 5 and 9; and PC3 clusters 6 and 7. Rotating the 3 PCs using promax (not shown) improves the quality of the simple structure slightly but does not change the clustering solution. Note that the variables number 5, 6, and 9 overlap between 2 clusters since they load moderately on two PCs. In the Euclidean Similarity-based CA solution (Figure 3-3b (i)) these members join the large clusters but at lower level of similarity.

Table 3-12: The Euclidean Similarity-based rotated loadings of the first 3 PCs using quartimax for the pressure data at the 48th hour.

n	PC 1	PC 2	PC 3
1	**0.87**	-0.05	0.14
2	**0.85**	-0.01	0.25
3	**0.73**	0.18	-0.29
4	**0.75**	-0.24	-0.03
5	0.49	**-0.69**	-0.10
6	0.42	-0.05	**0.57**
7	0.28	0.08	**0.80**
8	**0.84**	-0.04	0.15
9	0.41	**0.63**	0.02
10	**0.64**	0.37	-0.19

This solution is similar to the results of the EucSimi-based CA especially for the PC1 cluster. On the other hand this solution is not comparable to the correlation-based

PCA solution. The later one puts all members into one cluster due to extreme degree of correlation among the members.

3.4.3 CA based on the Euclidean distance.

For each output time, the Euclidean distance matrix, E_{10x10}, is computed for the difference field (2.3), and then used as the dissimilarity matrix for the CA method. Figure 3-3c shows the clustering dendrograms resulted from applying Ward's method (section 2.2.1) on the pressure data. These dendrograms enable the analyst, as apposite from correlation-based dendrograms, to study the behavior of the members and identify various clusters by cutting the dendrogram at different levels of dissimilarity. These analyses are very helpful in studying the behavior of the members.

3.4.4 CA based on the *K*mean and the Nucleated Agglomerative (NA) algorithms

The *K*mean algorithm (section 2.2.2 B) is applied for the finishing time of the pressure data set. The number of clusters is chosen to be 3, k=3, and the seed points are specified subjectively by examining the Ward's clustering tree for the 48th hour, Figure 3-3c. The resultant *K*mean partition for the finishing time is shown in Table 3-13.

Table 3-13: Clustering structures resulted from applying *K*mean algorithm to pressure data at the 48th hour.

Clustering structure		
C_1	C_2	C_3
[1, 2, 3, 4, 8, 9, 10]	[5]	[6, 7]

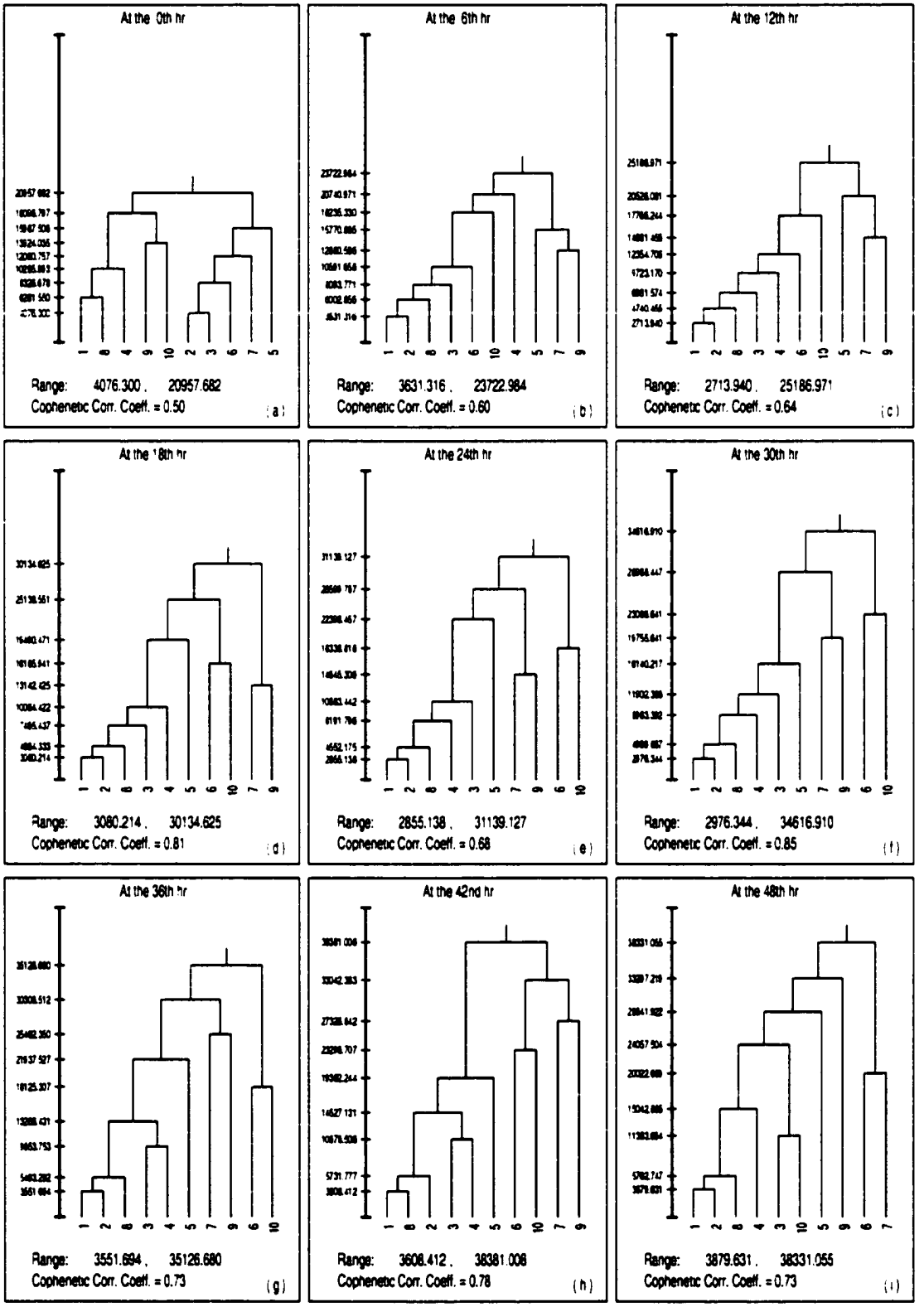


Figure 3-3c: Ward dendrograms for the Pressure field based on the Euclidean distance; data is centered using (2.3).

In a similar manner, the Nucleated Agglomerative (NA) algorithm (section 2.2.2 B) is applied for the same data set, and the number of clusters is chosen to be 3, k=3. The NA clustering solution for the 48th hour is shown in Table 3-14.

Table 3-14: Clustering structures resulted from applying NA algorithm to pressure data at the 48th hour.

Clustering structure		
C_1	C_2	C_3
[1, 2, 3, 4, 5, 8, 10]	[9]	[6, 7]

With respect to 3 clusters, both the *K*mean and NA algorithms lead to very similar partitions. The common observation that arises from the analysis of the pressure data is that all clustering algorithms applied to these data agree on placing the members 1, 2, 3, 4, 8 and 10 into one cluster at the finishing time (48th hour).

3.5 Analysis of the other fields

In this section, we present our results of analyzing the data sets for eight other fields. These fields include height (HGT) at 500 mb, temperature (TMP) at the surface level, wind (WIND) at 250 mb, absolute vorticity (ABS V) at 250, 500, 850 mb, pressure vertical velocity (V VEL) at 700 mb, and Convective Available Potential Energy (CAPE).

First, the PCA analysis (section 2.3) and CA analysis (section 2.2) based on the correlation (2.12) are conducted for three fields: height, temperature, and wind. After computing the eigenvalues for each correlation matrix, it turns out that the first eigenvalue of the correlation matrix of each output time for all these three fields is

extremely large compared to the others, and hence the total variance is nearly explained by the first PC itself. This, in fact is the same observation found when analyzing the pressure data, see section 3.4. Table 3-15 shows the first eigenvalue and percentage variance explained by the first PC at each output time for these fields. In this table and for the sake of comparison, the pressure results from Table 3-10 are included.

Table 3-15: First eigenvalue and variance explained by the first PC at each output time for height, temperature, and wind fields. The result of Table 3-10 is included here for the sake of comparison.

Time	Pressure (PRMSL) SFC		Height (HGT) 500mb		Temperature (TMP) SFC		Wind (WIND) 250mb	
	λ_1	Variance explained by λ_1 (%)	λ_1	Variance explained by λ_1 (%)	λ_1	Variance explained by λ_1 (%)	λ_1	Variance explained by λ_1 (%)
0	9.92	99.2%	9.98	99.8%	9.84	98.4%	9.83	98.3%
6	9.91	99.1	9.98	99.8	9.95	99.5	9.82	98.2
12	9.91	99.1	9.98	99.8	9.95	99.5	9.81	98.1
18	9.87	98.7	9.97	99.7	9.96	99.6	9.79	97.9
24	9.86	98.6	9.96	99.6	9.96	99.6	9.75	97.5
30	9.80	98.0	9.95	99.5	9.96	99.6	9.71	97.1
36	9.78	97.8	9.94	99.4	9.96	99.6	9.66	96.6
42	9.74	97.4	9.93	99.3	9.96	99.6	9.63	96.3
48	9.78	97.8	9.93	99.3	9.95	99.5	9.60	96.0

By analyzing the eigenvalues for the fields in Table 3-4, Table 3-15 and other fields (not shown) in this study, a heuristic stopping rule for correlation-based PCA is now derived. The rule determines the likelihood that there is only 1 cluster in the data by examining the ratio of the first eigenvalue to the second, third, forth etc eigenvalues of the correlation matrix of the data under study. Figure 3-4 shows the plots of the ratio of first eigenvalue to the second eigenvalue for all fields analyzed in this chapter against the underlying number of clusters. The number of clusters is determined subjectively by

examining the simple structures obtained by quartimax (or promax) rotation. As the figure illustrates, only when the ratio is 10.55 or less, more than one cluster is found. Therefore, the ratio $\frac{\lambda_1}{\lambda_2}$ helps to assess the likelihood that there is only 1 cluster in the data. This implies that when there is only 1 cluster then rotation using quartimax (or promax) is not needed. Based on the analysis of Figure 3-4, a heuristic "rule of thumb" can be stated as follows:

> *"If* $\frac{\lambda_1}{\lambda_2} > 10.55$, *Do not rotate. That is, If* λ_1 *is an order of magnitude greater than* λ_2, *no clustering is present so STOP solution (Do not rotate)".*

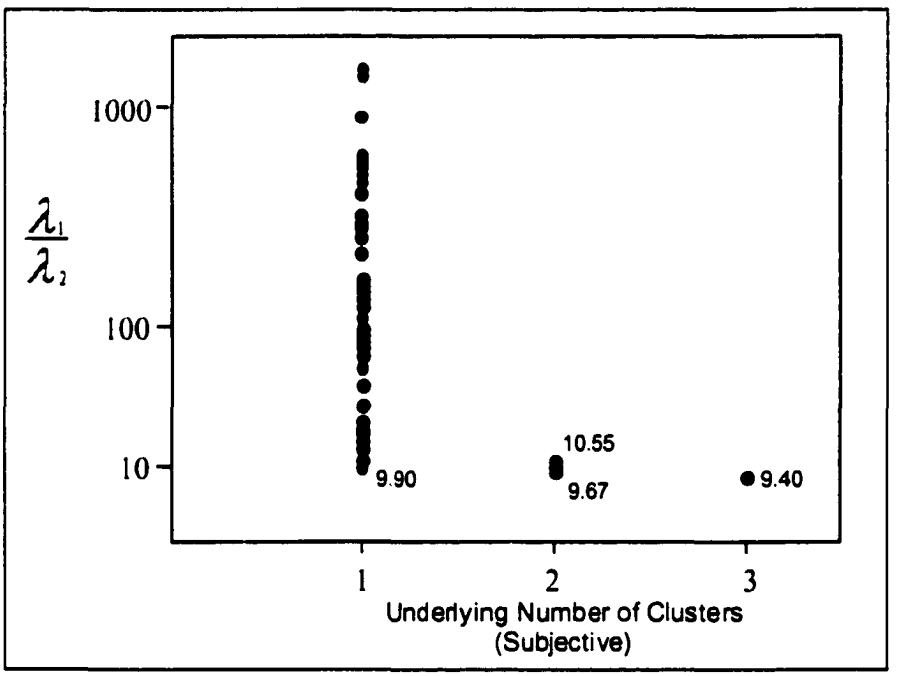


Figure 3-4: Plot of the ratio of first eigenvalue to the second eigenvalue for all fields analyzed in this chapter against the underlying number of clusters.

This rule is depicted graphically in Figure 3-5. Reader should be aware that this is not a general stopping rule. It is a heuristic rule derived based on the observations in this study only, and applicable for the correlation-based PCA and quartimax (or promax) rotation only. Generalization of this rule is out of the scope of this research and require more studies and analyses.

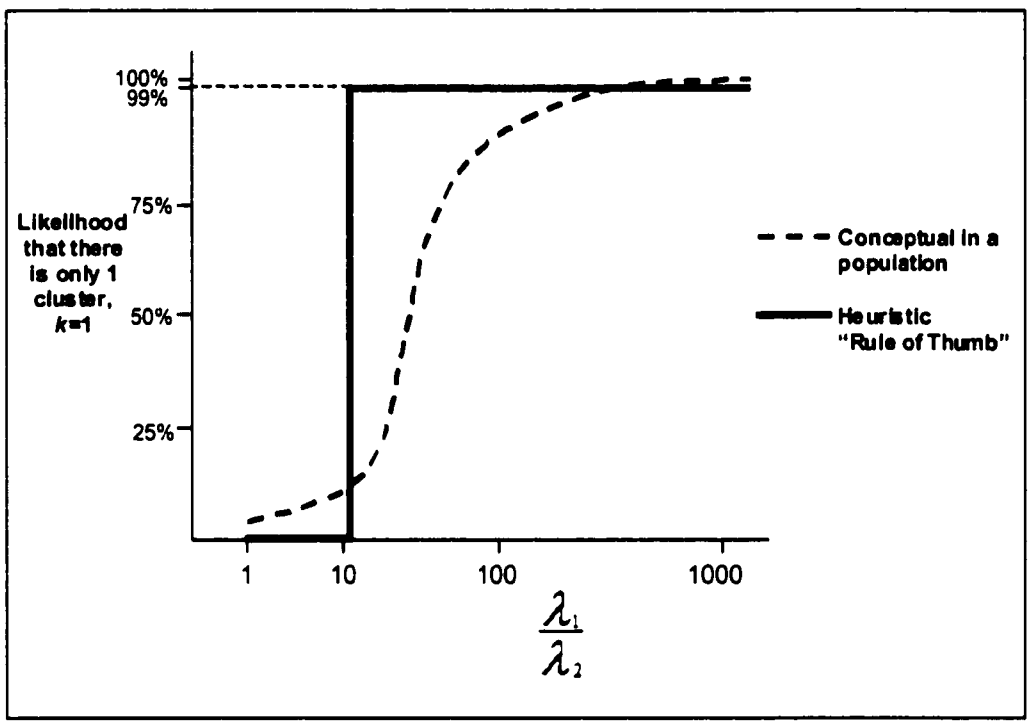


Figure 3-5: Graphical illustration of the heuristic "rule of thumb".

In view of this fact it turns out that the correlation-based PCA analysis and CA analysis of most of the fields in this study are essentially the same as those of pressure field. Also, the clustering dendrograms for these fields are nearly the same as the ones for the pressure data set (Figure 3-3a). This convinces us that correlation measure either with PCA or CA is not always suitable for discriminating the members and recovering their clustering structures for the data studied in this chapter. However, when the

variables are highly correlated the rotated principal component analysis, if more than one cluster is found, gives more useful information about the data than those given by the cluster analysis. In such cases, using PCA, the degree of concentration of the data can be realized by analyzing the variance explained by the first eigenvalue across the time. Also when oblique solution is found by PCA, the separation between the PCs (or clusters) can be measured using the correlation matrix of the rotated PCs which is computed by the promax algorithm.

We then turn to dissimilarity-based cluster analysis to study the behavior of the 10 members under distance measure. For each one of these three fields and other five fields – mentioned above- cluster analysis based on Euclidean distance is conducted for the difference field. For each field, the data matrices are first centered using (2.3). Then a distance matrix based on Euclidean distance (2.9) is computed for each output time and then used as the dissimilarity matrix for the CA method. The clustering method used in these analyses is Ward's method (section 2.2.1). These analyses enable us to recover the clustering structures embedded in the data and help to study the behavior of the members for each field. Figure 3-6 to Figure 3-13 show the clustering dendrograms resulted from applying Ward's method on the data sets of these fields. These dendrograms clearly reveal the clustering tendencies inherent in these data sets.

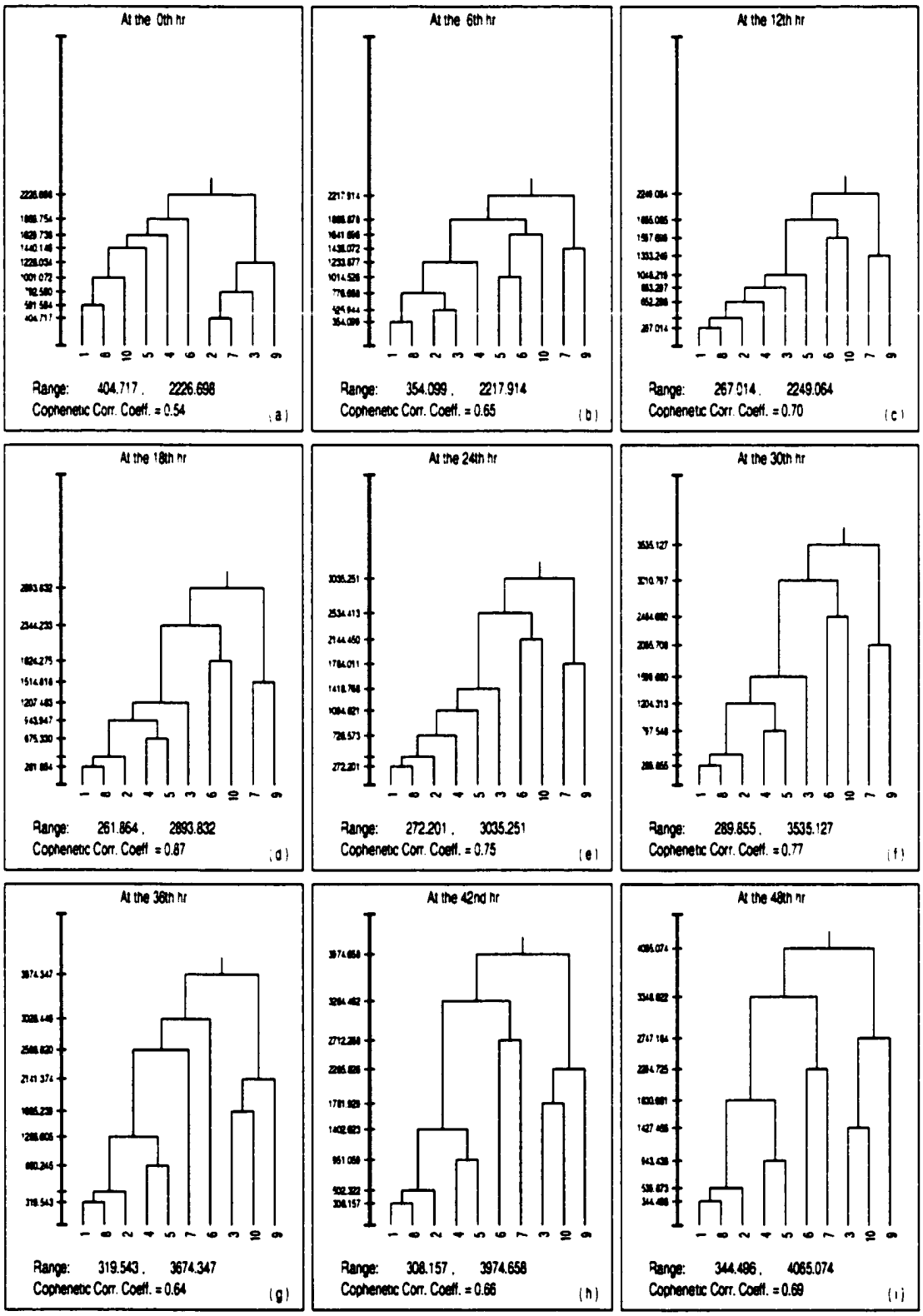


Figure 3-6: Ward dendrograms for the Height 500mb field based on Euclidean distance; data is centered using (2.3).

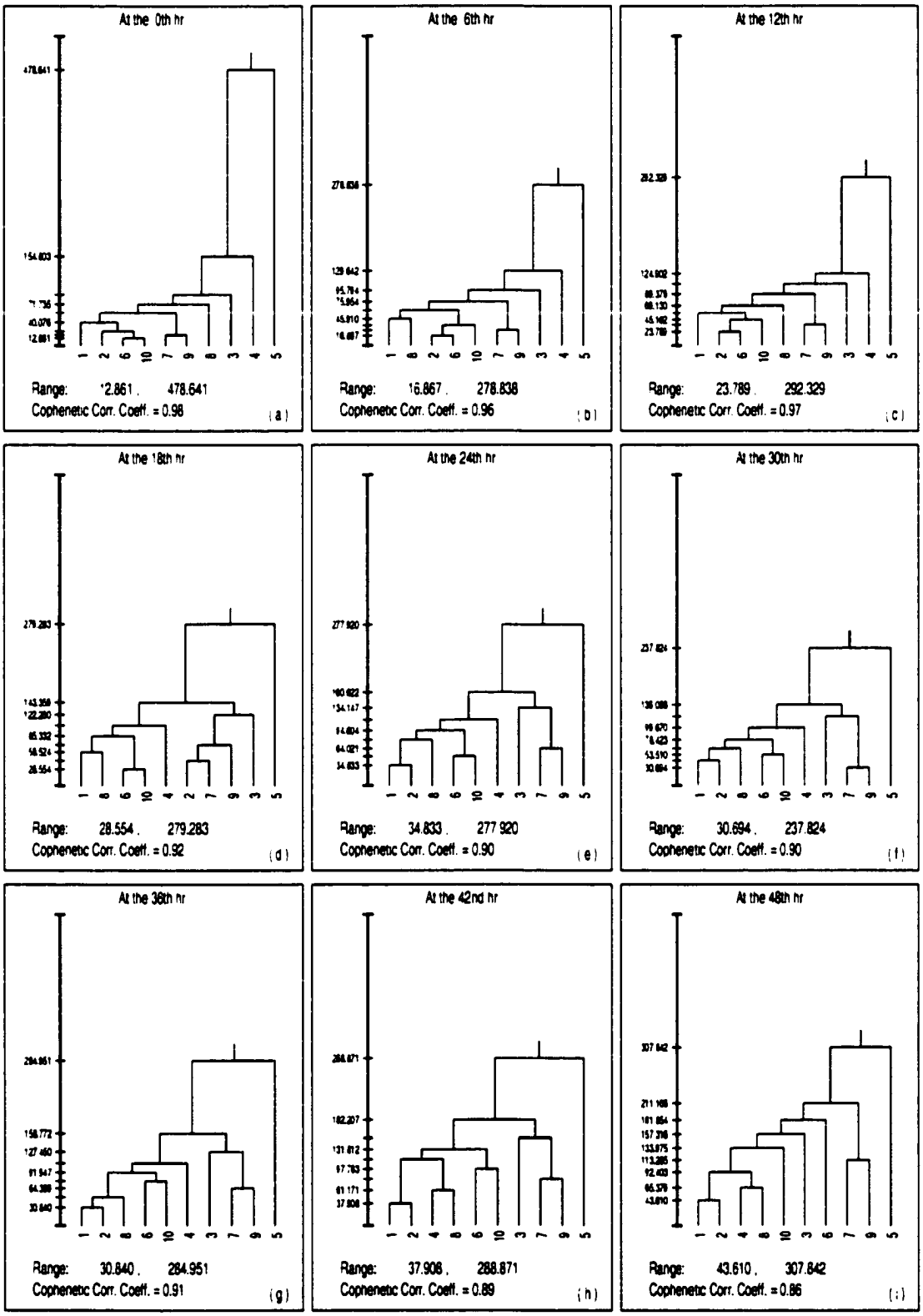


Figure 3-7: Ward dendrograms for the Temperature field based on Euclidean distance; data is centered using (2.3).

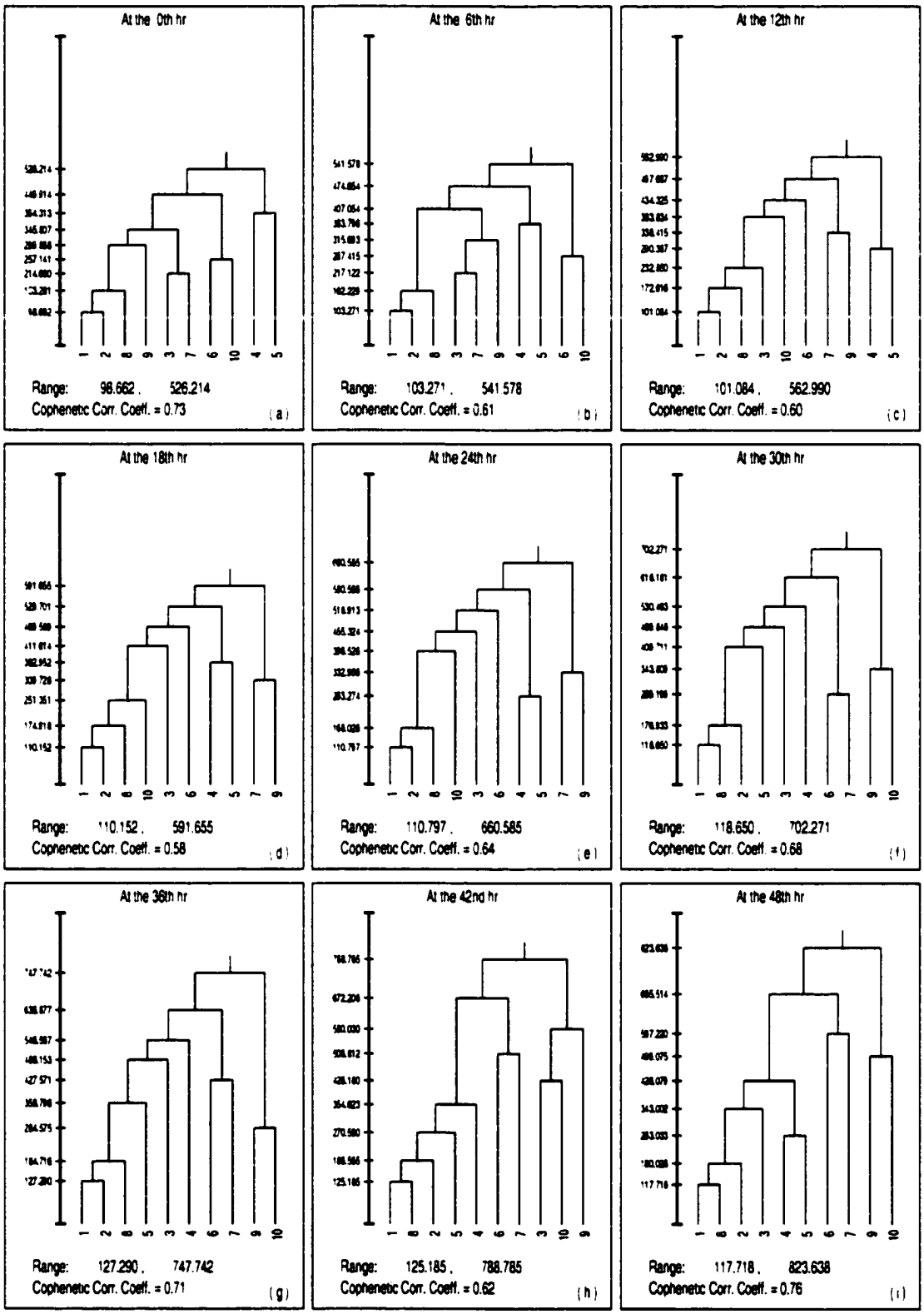


Figure 3-8: Ward dendrograms for the Wind 250mb field based on Euclidean distance; data is centered using (2.3).

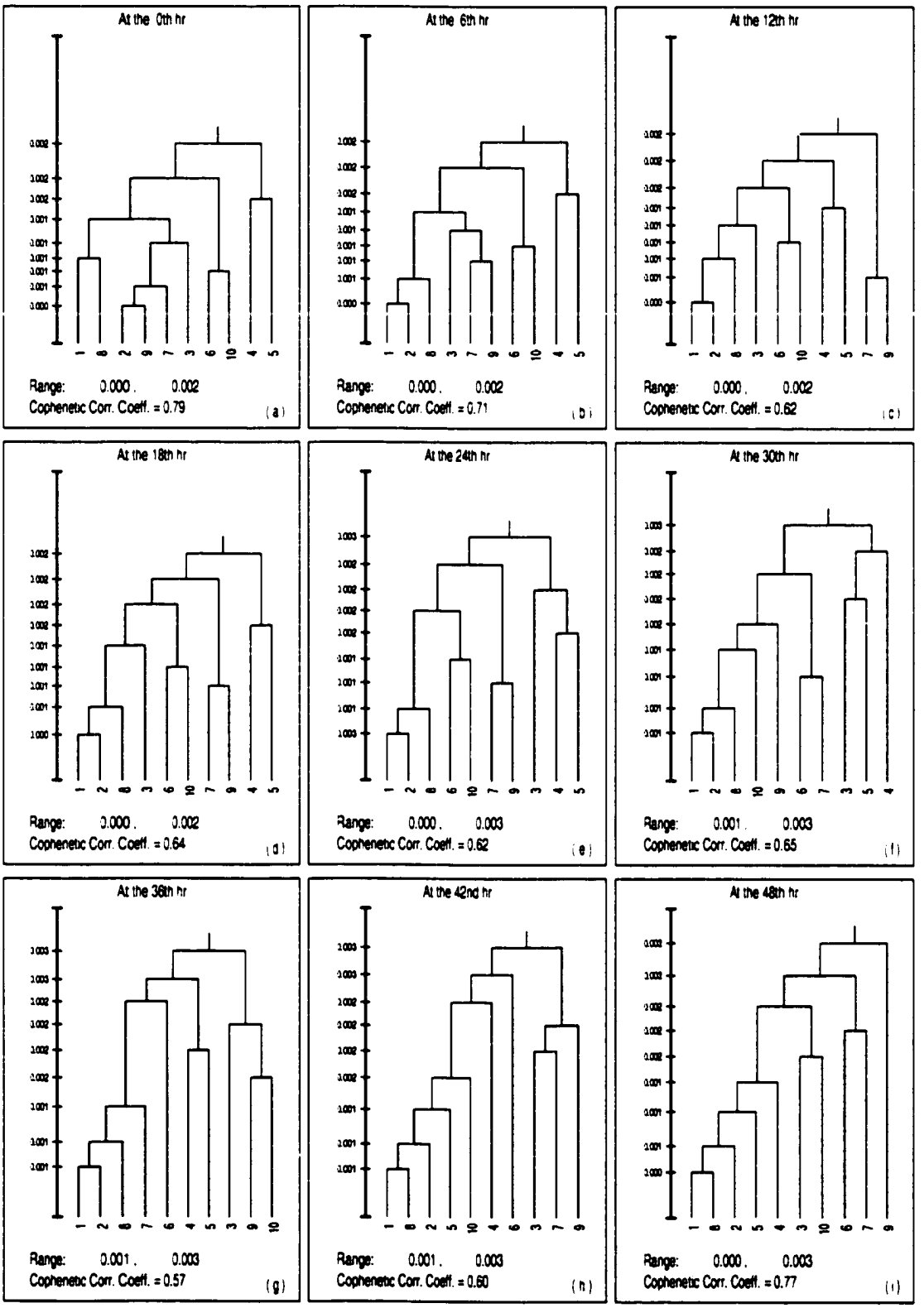


Figure 3-9: Ward dendrograms for the Abs. Vorticity 250mb field based on Euclidean distance; data is centered using (2.3).

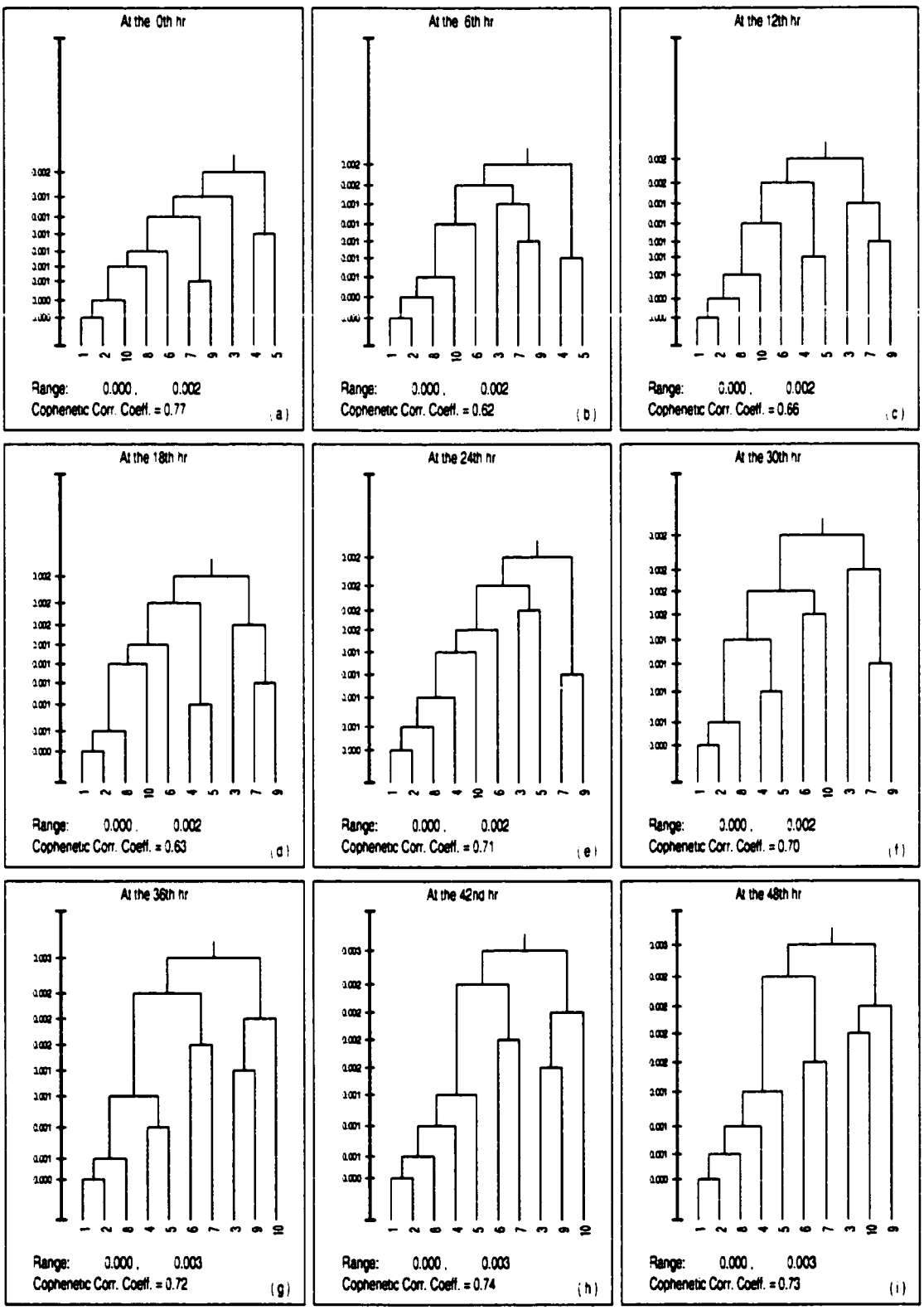


Figure 3-10: Ward dendrograms for the Abs. Vorticity 500mb field based on Euclidean distance; data is centered using (2.3).

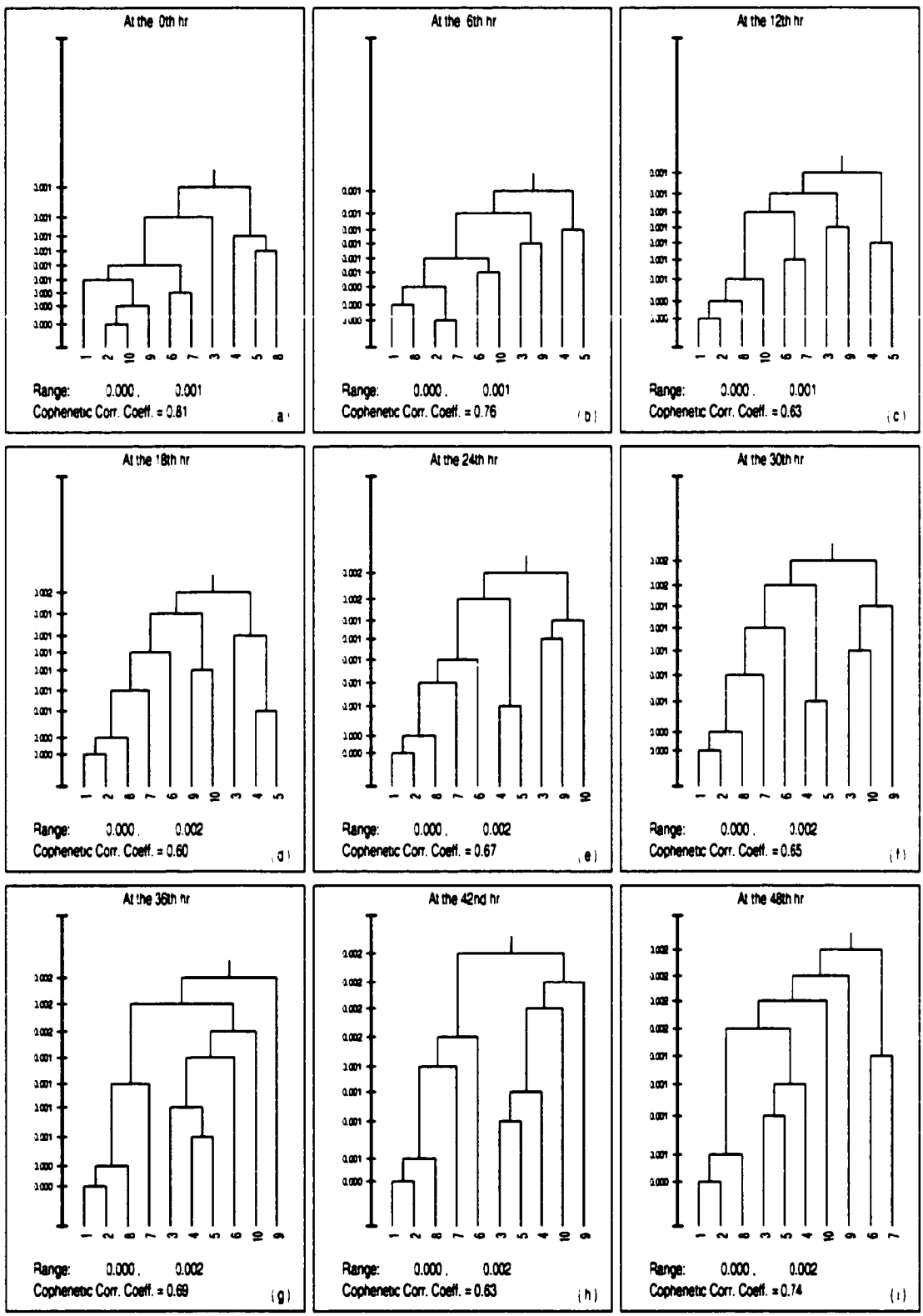


Figure 3-11: Ward dendrograms for the Abs. Vorticity 850mb field based on Euclidean distance; data is centered using (2.3).

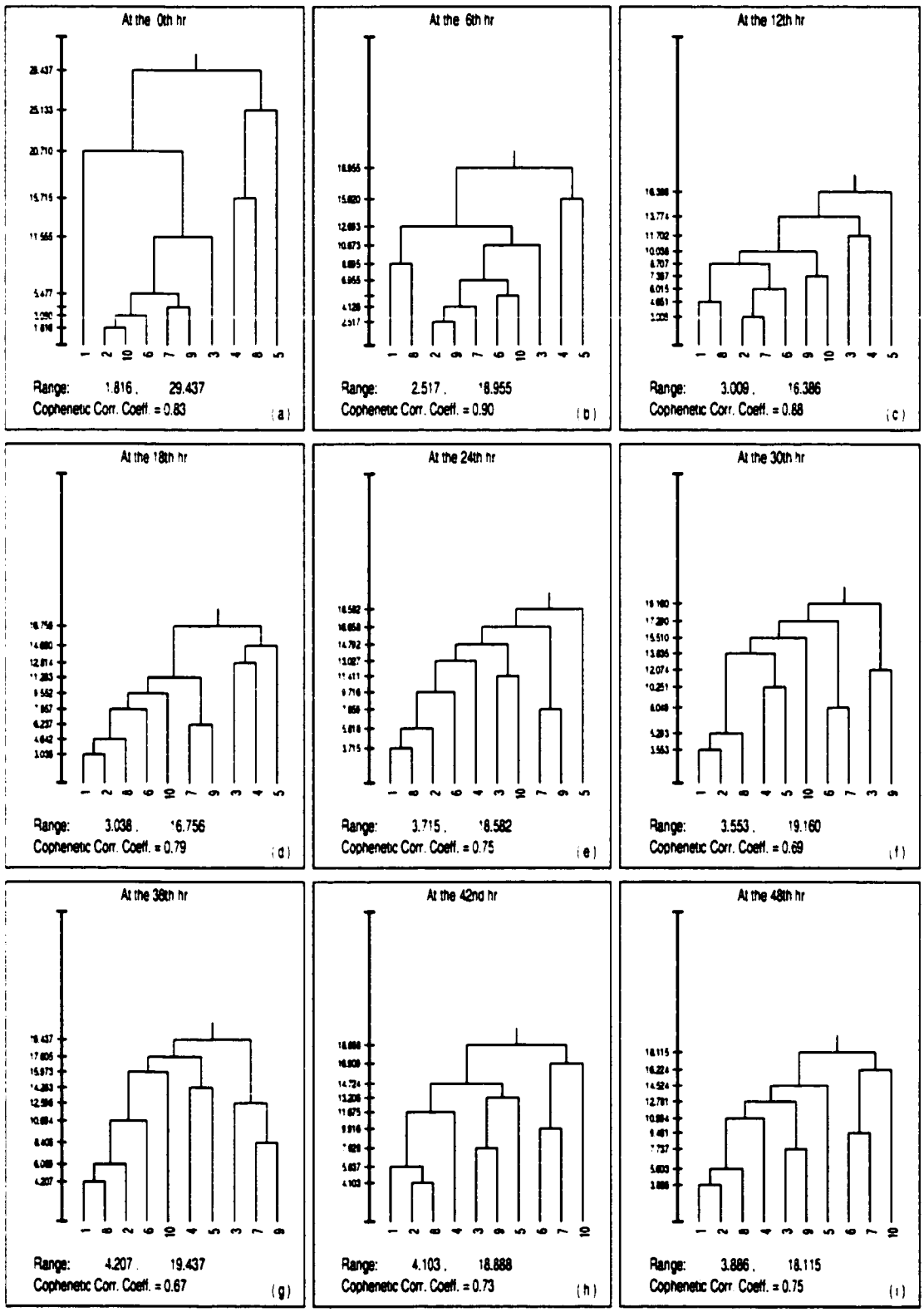


Figure 3-12: Ward dendrograms for the Pressure Vertical Velocity 700mb field based on Euclidean distance; data is centered using (2.3).

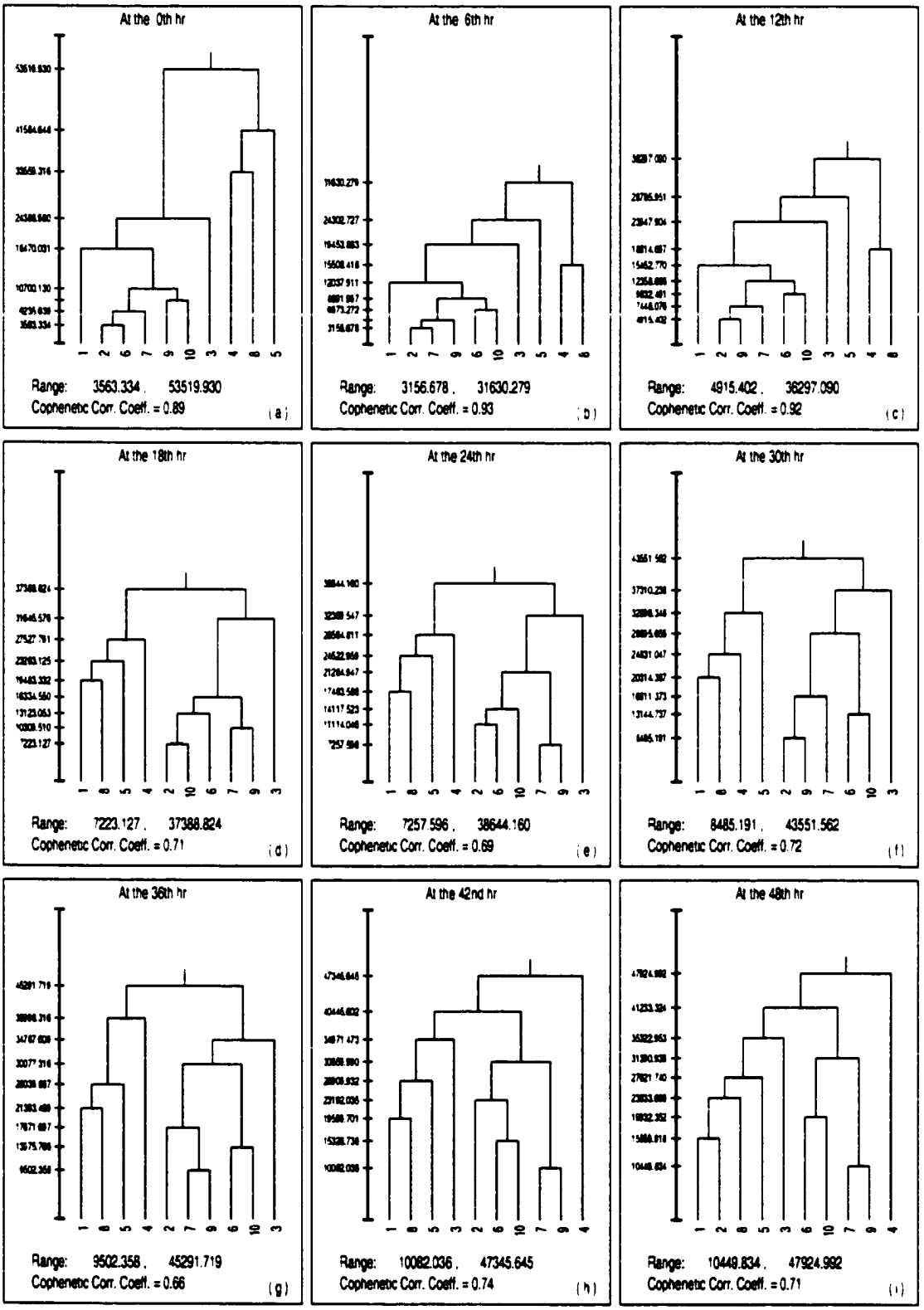


Figure 3-13: Ward dendrograms for the CAPE field based on Euclidean distance; data is centered using (2.3).

The PCA (section 2.3) is, then, applied on the Euclidean similarity matrices (2.16) of the finishing time of two fields: temperature and absolute vorticity at 850 mb. In the case of temperature, a strong simple structure is found by rotating 2 PCs using quartimax (section 2.3.2 A). According to this solution, Table 3-16, all members, except member no. 5, are placed into one cluster. This structure is very similar to the one obtained by Ward's method, Figure 3-5(i). In that solution, member no. 5 is shown to be isolated from other members.

Table 3-16: Rotated loadings of the first 2 PCs using quartimax for the temperature data at the 48^{th} hour.

n	PC 1	PC 2
1	**0.86**	0.04
2	**0.88**	0.05
3	**0.77**	0.03
4	**0.81**	0.00
5	0.06	**0.99**
6	**0.73**	0.01
7	**0.76**	0.12
8	**0.85**	-0.01
9	**0.76**	-0.02
10	**0.80**	-0.10

For the Absolute Vorticity (850 mb), rotating the first 4 PCs using quartimax (section 2.3.2 A) leads to moderate simple structure, Table 3-17. By examining this solution, the clustering structure of the 10 members is as follow: PC1 clusters 1, 2, 3, 4, 5, and 8; PC2 clusters 6 and 7; PC3 clusters 9; and PC4 clusters 10. In this solution, the variables 3 and 6 are nearly complex variables since they load moderately on almost all PCs. This structure is very consistent with the dendrograms in Figure 3-9(i).

Table 3-17: Rotated loadings of the first 4 PCs using quartimax for the absolute vorticity (850 mb) data at the 48th hour.

n	PC 1	PC 2	PC 3	PC 4
1	**0.75**	0.23	0.08	0.11
2	**0.71**	0.39	0.11	0.16
3	**0.48**	-0.27	0.22	0.37
4	**0.66**	-0.28	0.05	-0.29
5	**0.58**	-0.28	-0.04	0.24
6	0.32	**0.47**	-0.44	0.09
7	0.24	**0.74**	0.21	-0.03
8	**0.75**	0.12	-0.01	0.02
9	0.17	0.12	**0.86**	0.01
10	0.14	0.01	-0.01	**0.89**

3.6 Sensitivity Analysis

In this section, we perform a sensitivity analysis for PCA and CA with respect to several parameters. As for PCA, we perform this analysis with respect to using different Grammian Σ matrices (section 2.3.1). On the other hand, several parameters are used to perform sensitivity analysis for CA. These parameters include scaling method, clustering method, distance measures, and selection of seed points for partitional clustering algorithm. We use the precipitation (section 3.3) and pressure data sets (section 3.4) for this sensitivity analysis.

3.6.1 Sensitivity of PCA

Recall from section 3.3 and section 3.4 that we use the correlation matrix as the Grammian Σ matrix (2.30) for the PCA analyses of the precipitation and pressure data sets. In the case of precipitation, clustering tendencies seen through PCA analysis agrees with those observed by the CA. But there is no such agreement between the PCA

result and the CA result for the pressure data. Moreover, the dissimilarity-based PCA analyses of these two fields led to different results from those found using the correlation. This, in fact, means that PCA analysis does not necessary give the same results if different Grammian Σ matrix is used.

In view of the observed behavior of the first eigenvalue of the correlation matrices for the pressure data and some other fields analyzed in section 3.5, we decide to test the first eigenvalue if different Grammian matrix is used or if the data matrix is normalized differently. First, we conduct PCA analyses for the pressure, height, temperature, and wind fields data based on covariance matrix. Table 3.18 shows the first eigenvalue and variance explained by the first PC at each time instance resulted from applying PCA analyses based on covariance matrices. The result from using correlation matrix is included in this table for the sake of comparison. This table shows that using either correlation or covariance lead to the same results for these data sets.

Table 3-18: Variance explained by the first PC at each time instance for several fields resulted from applying PCA based on correlation (R) and covariance (C) matrices.

T	**Pressure (PRMSL)**		**Height 500mb**		**Temperature**		**Wind 250mb**	
	R	C	R	C	R	C	R	C
0	99.2%	99.2%	99.8%	%99.8	98.4%	99.2%	98.3%	98.3%
6	99.1	99.1	99.8	99.8	99.5	99.6	98.2	98.2
12	99.1	99.1	99.8	99.8	99.5	99.6	98.1	98.1
18	98.7	98.7	99.7	99.7	99.6	99.7	97.9	97.9
24	98.6	98.6	99.6	99.6	99.6	99.7	97.5	97.5
30	98.0	98.0	99.5	99.5	99.6	99.7	97.1	97.1
36	97.8	97.7	99.4	99.4	99.6	99.6	96.6	96.6
42	97.4	97.4	99.3	99.3	99.6	99.6	96.3	96.3
48	97.8	97.8	99.3	99.3	99.5	99.6	96.0	96.0

Second, the PCA is conducted for normalized data. That is, X is normalized before Grammian is computed. We normalize X using (2.5), (2.6), and (2.7). The first eigenvalue and variance explained by the first PC at each output time resulted from applying PCA analyses based on Grammian for normalized X is shown in Table 3-19. These results, in fact, agree on that the first eigenvalue is extremely large and the variance explained by the first PC is nearly 100%. This means that changing the Grammian by merely centering or normalizing does not affect the overall result of the PCA analysis.

Table 3-19: First eigenvalue and variance explained by the first PC at each output time for pressure fields resulted from applying PCA based on Grammian for normalized X.

Time	X is normalized using (2.5)		X is normalized using (2.6)		X is normalized using (2.7)	
	λ_1	Variance explained by λ_1 (%)*	λ_1	Variance explained by λ_1 (%)	λ_1	Variance explained by λ_1 (%)*
0	9.72	100%	5.84	99.9%	9.69	100%
6	9.70	100	5.93	99.9	9.66	100
12	9.69	100	5.86	99.9	9.65	100
18	9.69	100	5.82	99.9	9.64	100
24	9.68	100	5.62	99.9	9.63	100
30	9.66	100	5.27	99.9	9.63	100
36	9.60	100	4.57	99.9	9.57	100
42	9.57	100	4.30	99.9	9.53	100
48	9.53	100	3.91	99.9	9.49	100

* indicates that values are shown for five digits accuracy.

3.6.2 Sensitivity of CA

In this subsection, we discuss our results of analyzing the sensitivity of hierarchical (section 2.2.1) and partitional (section 2.2.2) cluster analysis. The sensitivity of hierarchical CA is studied with respect to the scaling method and clustering method. The sensitivity of the partitional CA algorithms is studied with respect to different schemes that used to specify the seed points.

Sensitivity with respect to scaling method

In section 3.4, we present our result of applying Ward's method based on Euclidean distance on centered data matrix. In order to study the sensitivity of CA to various scaling methods, we perform CA analysis for three normalized versions of the pressure data set. These versions are obtained by normalizing the data matrices using (2.5), (2.6), and (2.7). Euclidean distance (2.9) and Ward's (section 2.2.1) are used as the dissimilarity coefficient and clustering method in these analyses. Figure 3-12a – Figure 3-12c show the clustering dendrograms for the normalized data matrices using (2.5), (2.6), and (2.7), respectively.

The quality of clustering is about the same but the clustering structures are different due to different scaling methods. In other word, the clustering structures are recovered regardless the shape of the data but the obtained structures themselves are not always the same.

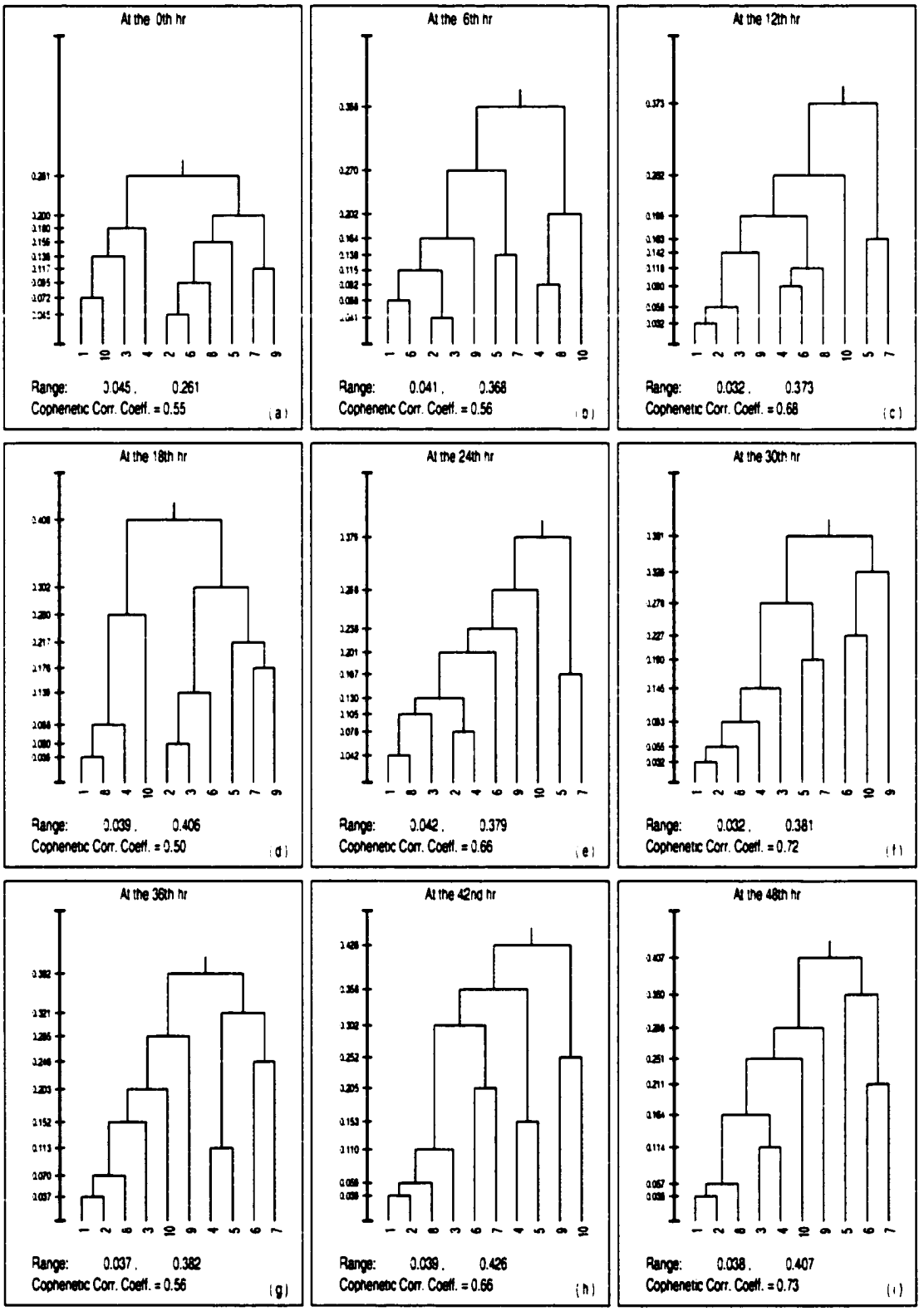


Figure 3-14a: Ward dendrograms for the Pressure field based on the Euclidean distance; data is centered using (2.5).

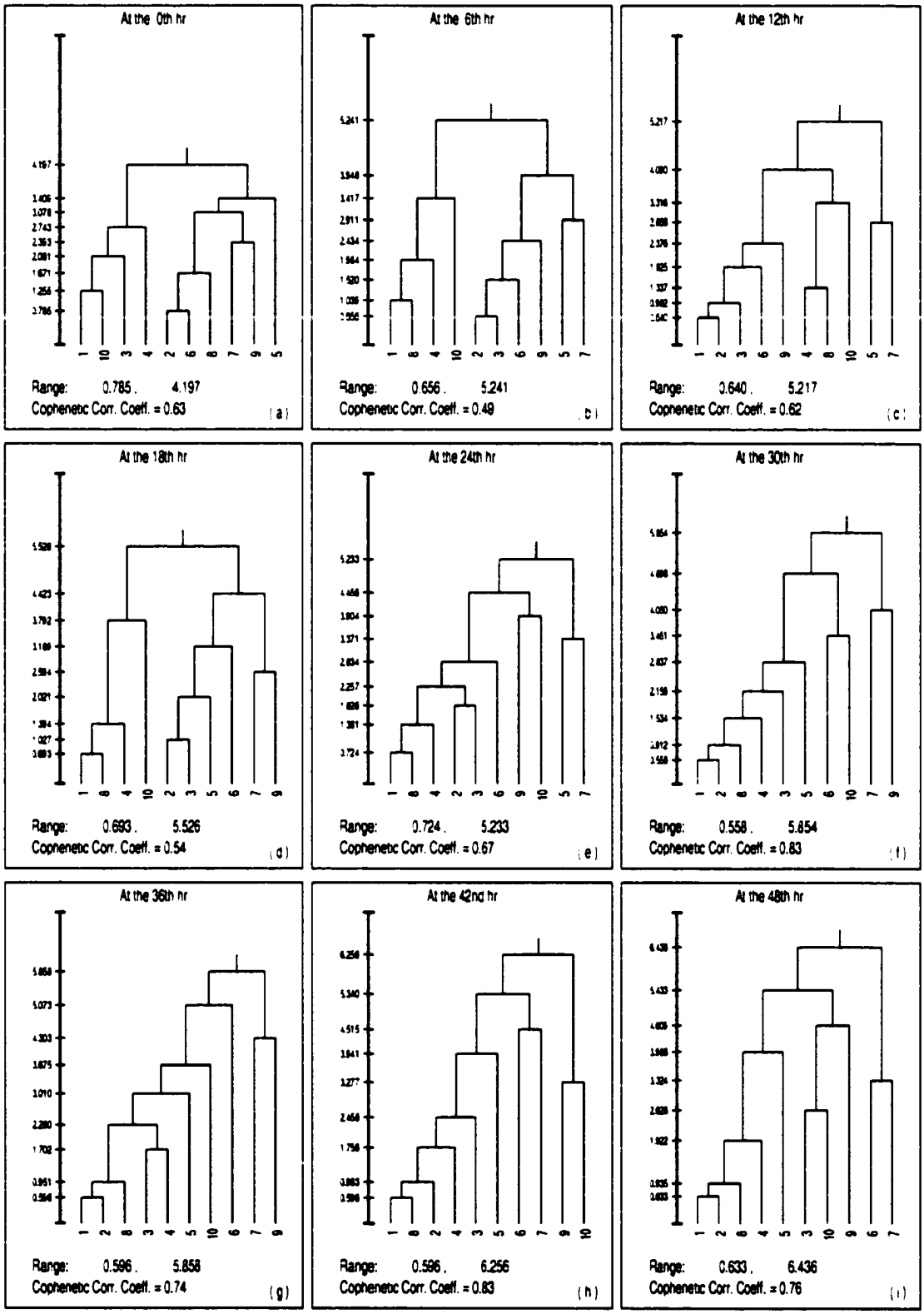


Figure 3-14b: Ward dendrograms for the Pressure field based on the Euclidean distance; data is centered using (2.6).

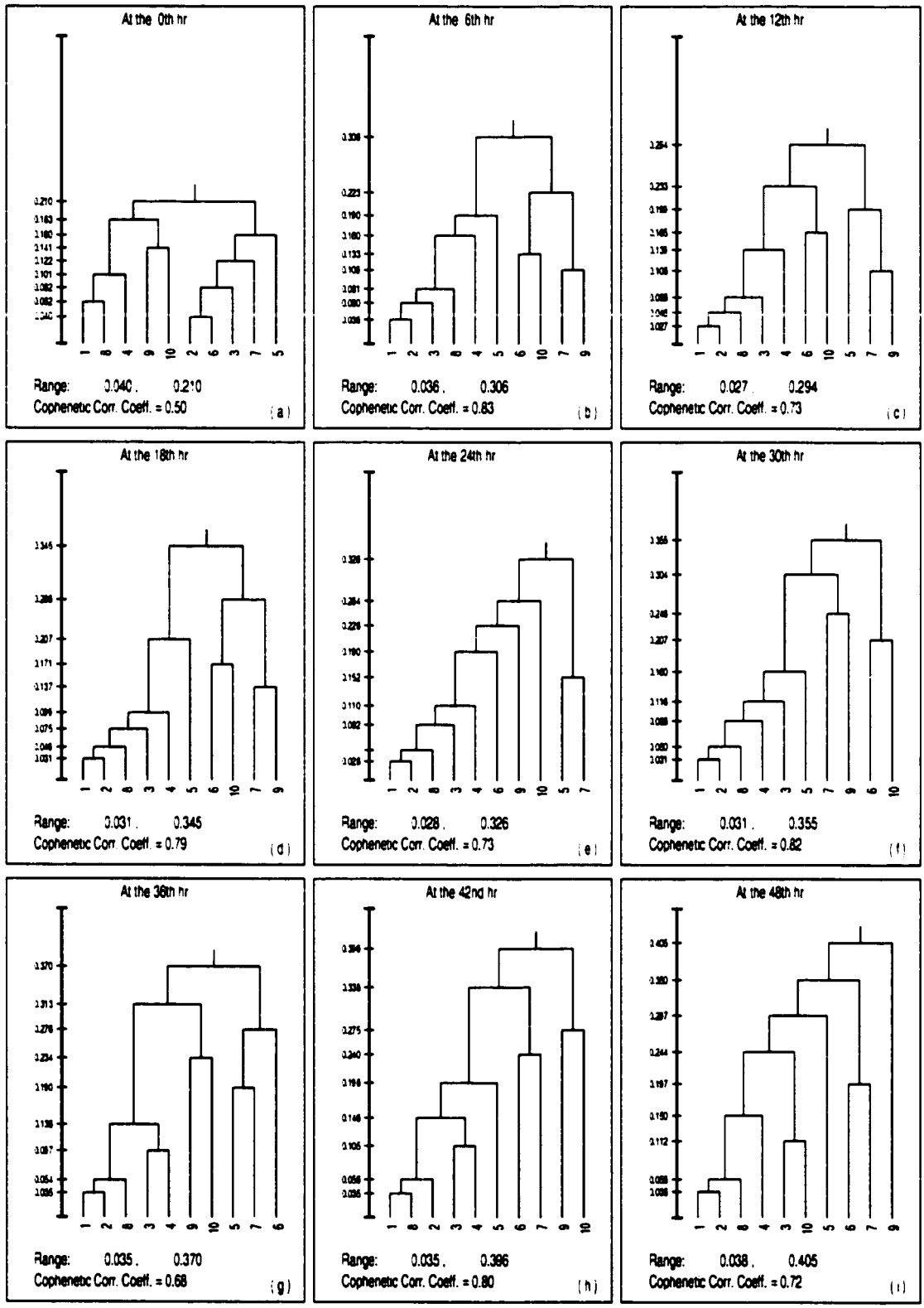


Figure 3-14c: Ward dendrograms for the Pressure field based on the Euclidean distance; data is centered using (2.7).

Sensitivity with respect to clustering method

In section 3.4, we present our result of applying Ward's method based on Euclidean distance on centered data matrix. In order to study the sensitivity of CA to various clustering methods, we perform CA analysis for the centered pressure data, based on Euclidean distance (2.9), using three other methods: SLINK as in (2.18), CLINK as in (2.19), and UPGMA as in (2.20). Figure 3-13a – Figure 3-13c show the clustering dendrograms resulted from applying SLINK, CLINK, and UPGMA methods, respectively. Again, the clustering structures are recovered regardless the method used for clustering the data but the obtained structures themselves are different. Gong and Richman (1995) in earlier work observed this sensitivity of CA to the change of the clustering method.

Sensitivity with respect to distance measure

In order to study the effect of using different distance measures on the overall result of the CA, we conduct CA analysis for the pressure data set using three different distance coefficients: Euclidean distance (2.9), Manhattan distance (2.10), and 'sup' distance (2.11). CLINK is used as the clustering method in these analyses. Figure 3-14a – Figure 3-14c show the resultant clustering dendrograms of these analyses.

As the figures show, the clustering structures are recovered for the three measures, however the obtained structures themselves are not always the same. Therefore, CA is sensitive to the change of the dissimilarity coefficient.

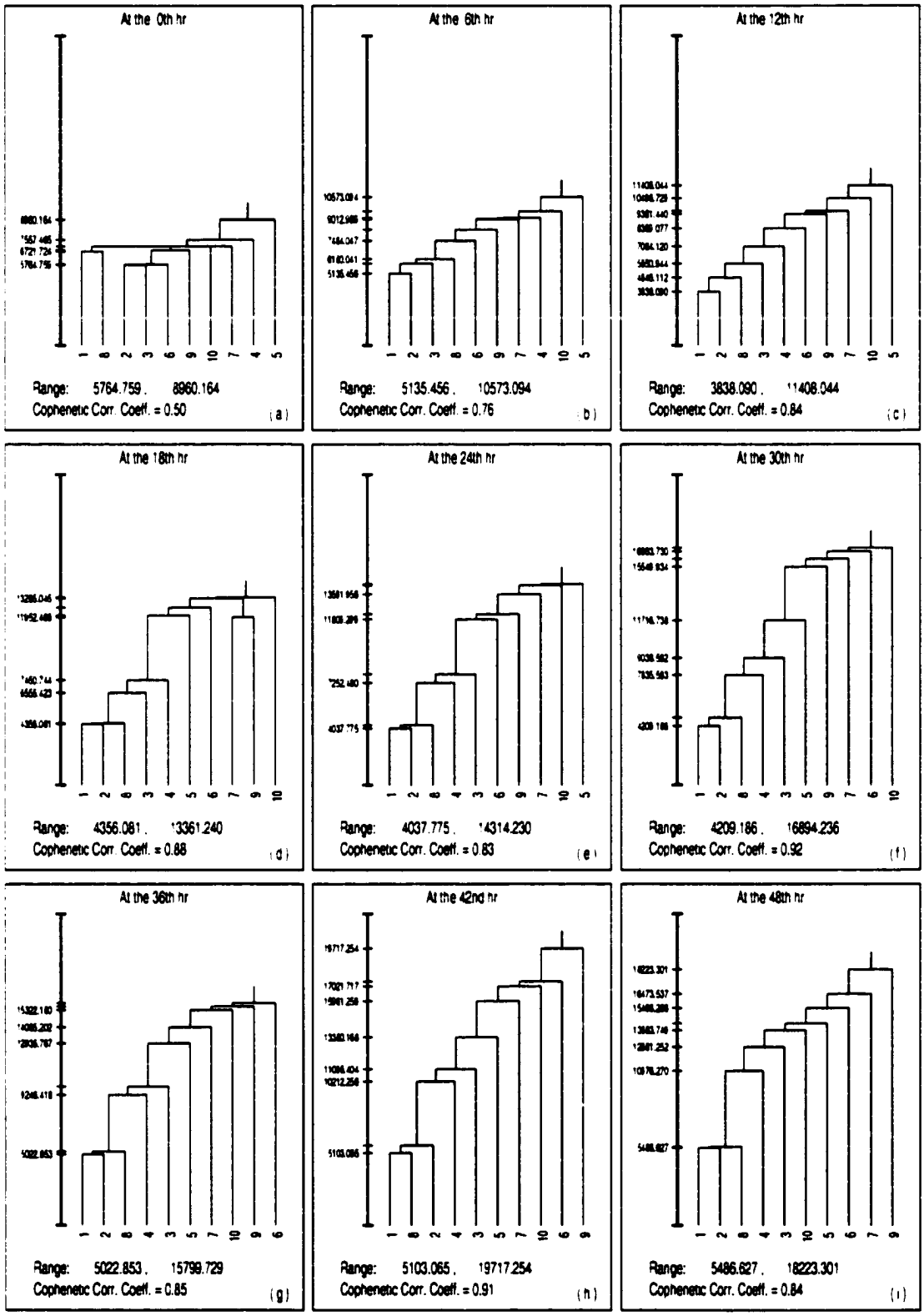


Figure 3-15a: SLINK dendrograms for the Pressure field based on the Euclidean distance; data is centered using (2.3).

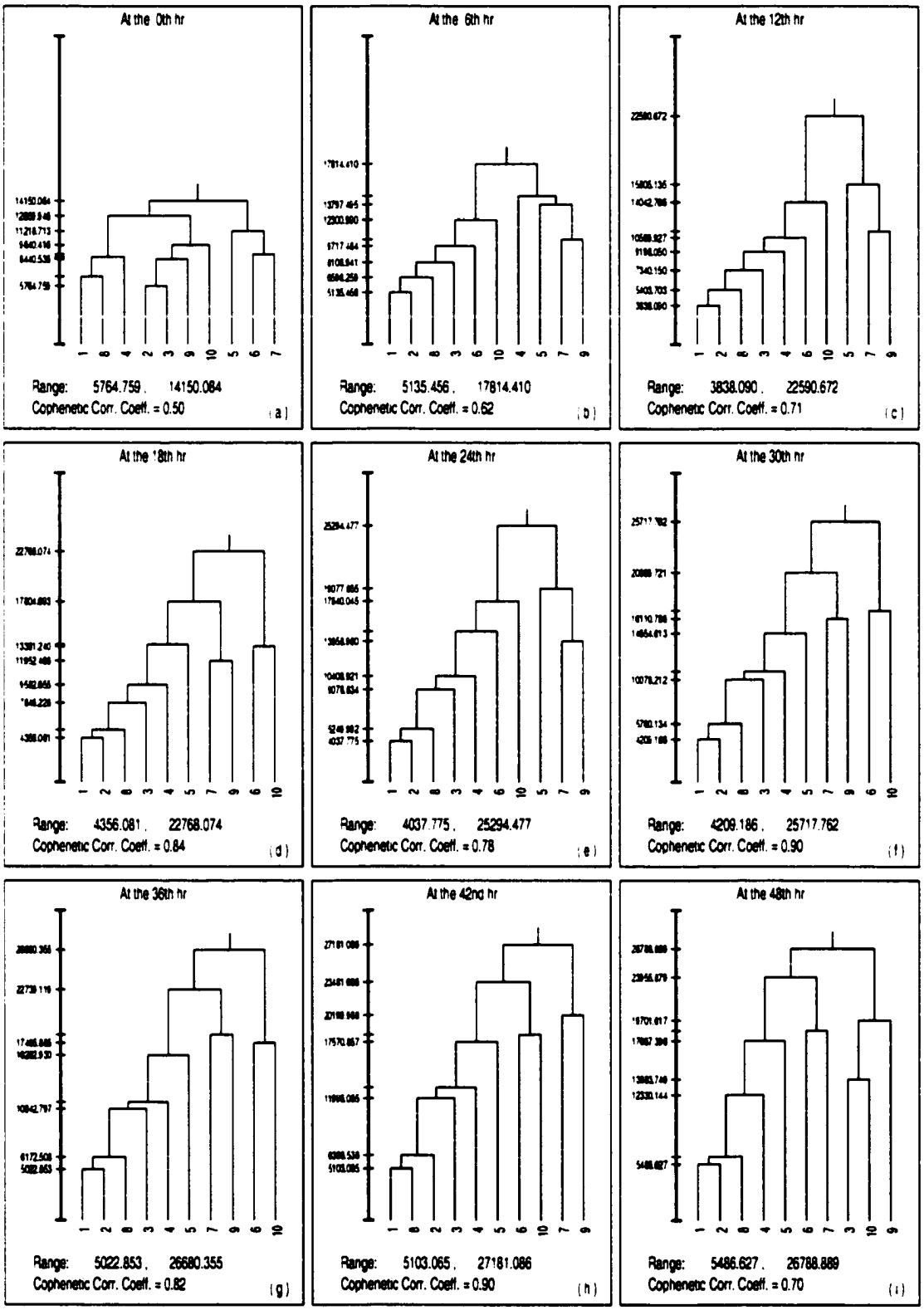


Figure 3-15b: CLINK dendrograms for the Pressure field based on the Euclidean distance; data is centered using (2.3).

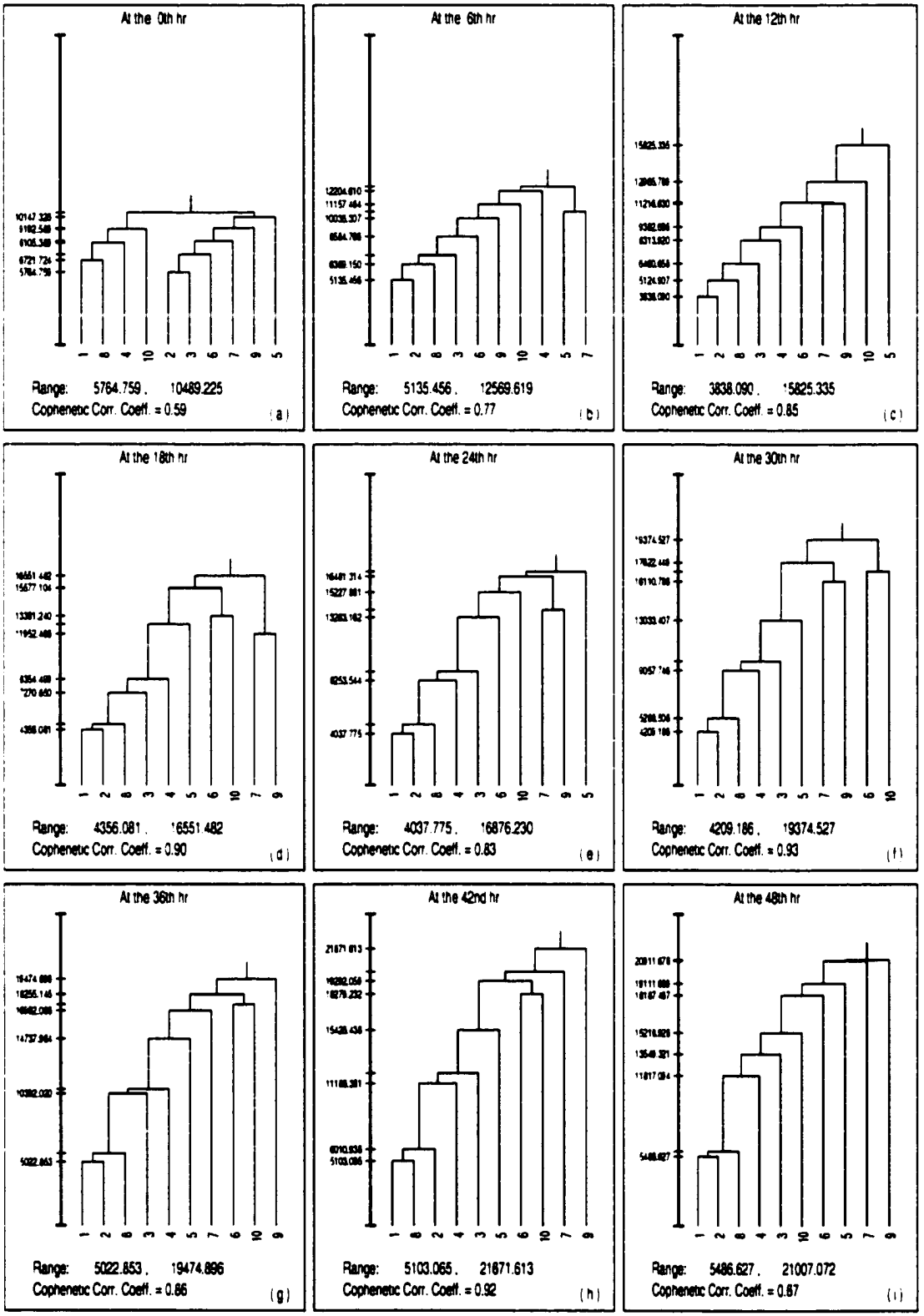


Figure 3-15c: UPGMA dendrograms for the Pressure field based on the Euclidean distance; data is centered using (2.3).

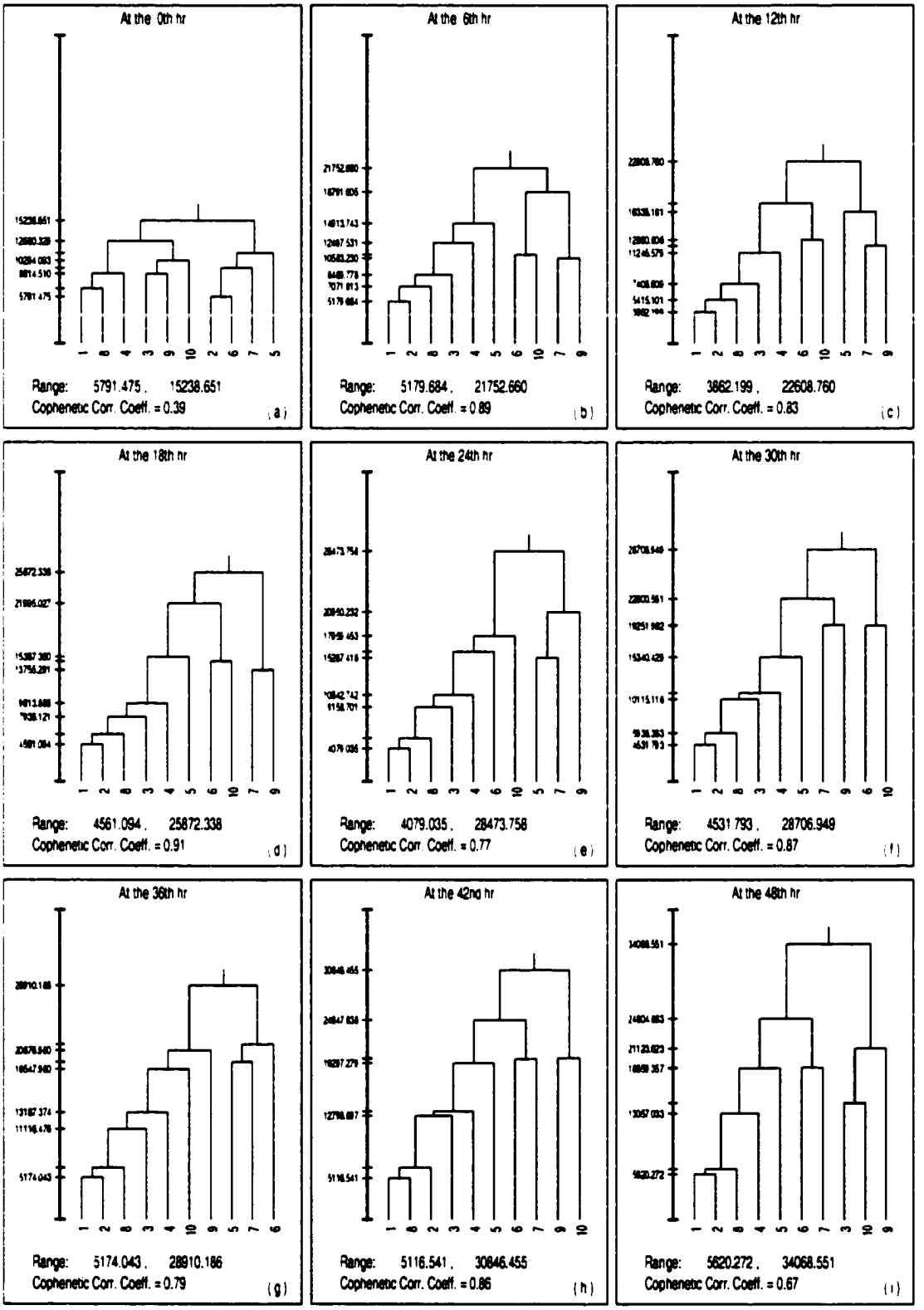


Figure 3-16a: CLINK dendrograms for the Pressure field based on the Euclidean distance.

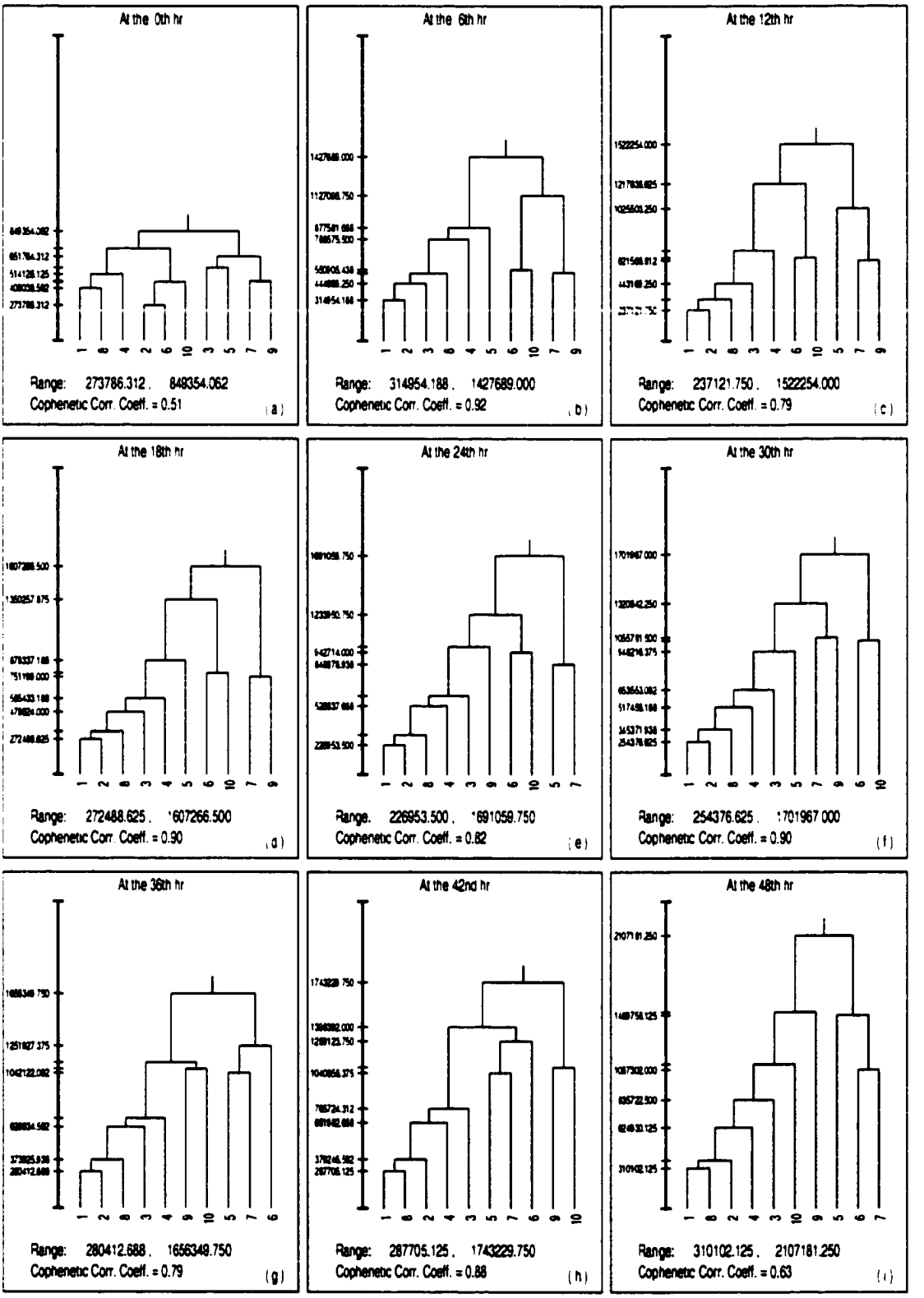


Figure 3-16b: CLINK dendrograms for the Pressure field based on the Manhattan distance.

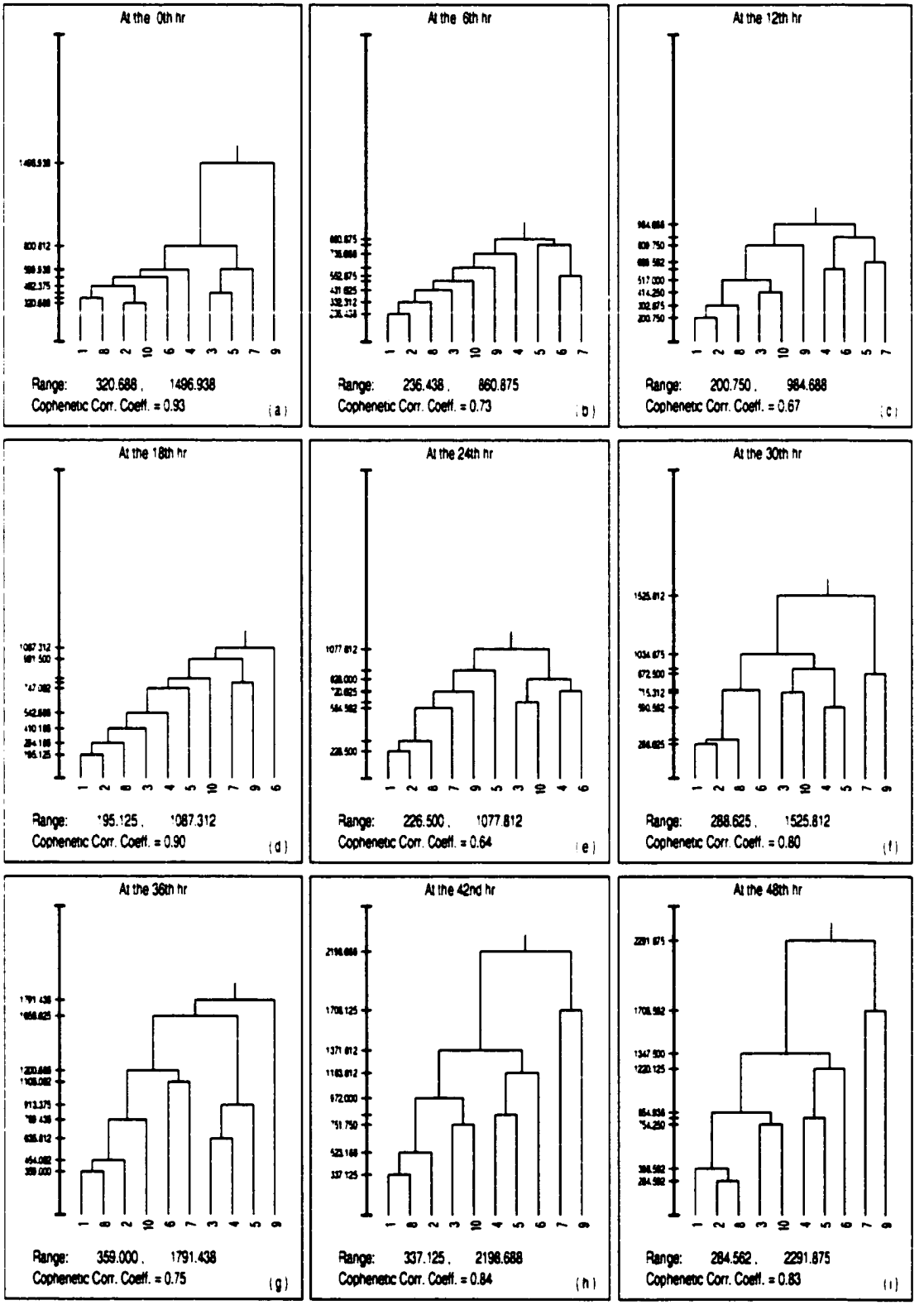


Figure 3-16c: CLINK dendrograms for the Pressure field based on the Sup distance.

Sensitivity of the *K*mean algorithm

*K*mean algorithm (section 2.2.2 B) is a partitional clustering technique. Recall from section 2.2.2 that the partitional methods work directly on the data matrix, hence there is no need for a resemblance matrix. However the number of clusters and the seed point for each cluster need to be specified a priori.

For each output time of the pressure data set, the *K*mean algorithm is applied. The number of clusters is fixed to be 3 and the seed points are specified using three different schemes. The first scheme (Scheme 1) is to specify the first members (column) to be the seed points for the three clusters, that's the first column is the seed point for the first cluster and so on. The other two schemes are to specify the seed points randomly (Scheme 2) or subjectively (Scheme 3). The results of applying *K*mean algorithm to pressure data set at the 0^{th} hour are shown in Table 3-20, and at the 48^{th} hour are shown in Table 3-21.

Table 3-20: Clustering structures resulted from applying *K*mean algorithm to pressure data at the 0^{th} hour.

Scheme	Clustering structure		
Scheme 1	[1, 4, 8]	[2, 3, 6, 9, 10]	[5, 7]
Scheme 2	[1, 4, 8]	[2, 6, 10]	[3, 5, 7, 9]
Scheme 2	[1, 4, 8]	[2, 3, 6, 9, 10]	[5, 7]

Table 3-21: Clustering structures resulted from applying *K*mean algorithm to pressure data at the 48^{th} hour.

Scheme	Clustering structure		
Scheme 1	[1, 2, 4, 5, 8]	[3, 9, 10]	[6, 7]
Scheme 2	[1, 2, 4, 5, 8]	[3, 9, 10]	[6, 7]
Scheme 2	[1, 2, 4, 5, 8]	[3, 9, 10]	[6, 7]

As shown in these tables, the clustering structures at the 0^{th} hour are affected by changing the seed points. However, the same structure obtained at the 48^{th} hour regardless of the seed points the algorithm starts with. The clustering structures at the other hours, not shown, are not always the same. This means that the *K*mean algorithm is sensitive to changing the seed points and could produce different clustering structures if different schemes are used to specify the seed points. Similar behavior was earlier observed in Gong and Richman (1995). The authors note that *K*mean is sensitive to the initial partition.

Sensitivity of the Nucleated Agglomerative (NA) algorithm

The sensitivity of the Nucleated Agglomerative (NA) algorithm (section 2.2.2 B) is conducted using the finishing time of the precipitation data set. The number of clusters is chosen to be 3 and the seed points are specified using three different schemes. The first scheme (Scheme 1) is to specify the first members (column) to be the seed points for the four clusters The other two schemes are to specify the seed points randomly (Scheme 2) or subjectively (Scheme 3). The results of applying NA algorithm to precipitation data set at the 48^{th} hour are shown in Table 3-22. As shown in this table, the partitions are affected by changing the seed points. This means that the NA algorithm is sensitive to changing the seed points and could lead to different clustering solutions if different schemes are used to specify the seed points. Based on this observation that both *K*mean and NA algorithms are sensitive to changing the seed points, and since we do not have any prior information about the ensemble data studied in this research, we believe that the hierarchical clustering algorithms are good way to pick the seed points for the partitional methods.

Table 3-22: Clustering structures resulted from applying NA algorithm to precipitation data at the 48th hour.

Scheme	Clustering structure		
Scheme 1	[3, 6, 7, 9, 10]	[1, 2, 4, 8]	[5]
Scheme 2	[1, 2, 3, 6, 7, 9, 10]	[4, 8]	[5]
Scheme 3	[3, 6, 7, 9, 10]	[1, 2, 4, 8]	[5]

3.7 Summary and Concluding remarks

This chapter demonstrates the applications of the principles of clustering methodologies to a uni-model short-term ensemble data. The ensemble studied in this chapter consists of 10 members and the goal is to analyze the trajectories from these members and see how they cluster during evolution. Part of this work in this chapter was presented in Alhamed and Lakshmivarahan (2000).

Two statistical techniques are examined: principal component analysis and cluster analysis. Based on the analyses of several meteorological fields, it is observed that the similarity-based principal component analysis and similarity-based cluster analysis do not provide useful clustering tendencies in all cases. They are very useful only when variance in the data set is not concentrated in one dimension. However, when the variables are highly correlated the rotated principal component analysis gives more useful results than those given by the cluster analysis. Using PCA, the degree of concentration of the data can be realized by analyzing the variance explained by the first eigenvalue across the time. Also when oblique solution is found by PCA, the separation between the PCs (or clusters) can be measured using the correlation matrix of the rotated PCs which is computed by the promax algorithm. On the other hand, dissimilarity or distance based cluster analysis provides useful information in all cases.

For the distant-based hierarchical cluster analysis, a method has been developed for drawing the clustering trees for each field that takes into consideration the different states of the ensemble members during their evolution. The clustering trees for each field are drawn using the same scale. Therefore, the tightness or divergence among the ensemble members for one field can be realized visually by examining the trees height and the level of mergers as the time goes from the initial time to the finishing time.

A heuristic, rule of thumb, stopping rule for correlation-based PCA is derived to help analyst to decide whether rotation of PC is needed. The rule determines the likelihood that there is only one cluster in the data by examining the ratio of the first eigenvalue to the second one. When there is only 1 cluster, then the rotation using quartimax is not needed.

Moreover, the clustering algorithms are shown to be sensitive in the context of ensemble forecasting system. The sensitivity of the hierarchical cluster analysis is studied with respect to the changes of several parameters, such as scaling method, resemblance coefficient, clustering method. Also, the nonhierarchical clustering algorithms are found sensitive to the change of the seed point selection.

CHAPTER 4

ANALYSES OF MULTI-MODEL ENSEMBLE USING SAMEX DATA

4.1 Introduction

In this chapter clustering techniques are applied to the problem of short-term ensemble forecasting in meteorology based on a multi-model data. The multi-model data used in this chapter is known as the Storm And Mesoscale Ensemble Experiment, or SAMEX for short. SAMEX is a multi-institutional numerical weather prediction experiment, coordinated by the Center for Analysis and Prediction of Storms (CAPS) at the University of Oklahoma, that occurred during May of 1998. SAMEX represents a collaboration among CAPS, the National Sever Storms Laboratory (NSSL), and the National Centers for Environmental Prediction (NCEP). One goal that SAMEX seeks to achieve is to apply short-term ensemble forecasting techniques and related statistical verification strategies to multi-model ensemble forecasts. Another goal is to explore the usefulness of model diversity in ensemble forecasting. More information on SAMEX can be found in Hou *et al.* (2000).

The interest here is to apply various cluster analysis, CA, (section 2.2) methods and principal component analysis, PCA, (section 2.3) to several types of meteorological fields, which are the output of the SAMEX multi-model project for 25 forecasts valid over the same time period using either different models and different initial conditions. These techniques allow to objectively compare the forecast fields from the different models that participated in SAMEX and explore their similarities. Subjective clustering is also used to evaluate the ability of the algorithms to produce clusters that make sense meteorologically.

While ensemble forecasting has largely focused upon different methods for generating the ensemble member initial conditions (Mullen and Baumhefner 1988; Toth and Kalnay 1993, 1997; Buizza and Palmer 1995; Houtekamer and Derome 1995; Molteni et al. 1996) the role of model physics variability in ensembles is now also being recognized as important for both medium-range (Buizza et al. 1999; Harrison *et al.* 1999) and short-range forecasting (Stensrud *et al.* 2000). The SAMEX data provides an excellent opportunity to further explore and compare both initial condition and model variability in the creation of a short-range ensemble forecast (Alhamed, Lakshmivarahan, and Stensrud 2000). Using CA and PCA methods, the trajectories from the model forecasts that constitute the ensemble are analyzed to learn how the members of the ensemble cluster as they evolve with time.

The ten fields analyzed in this chapter are listed in Table 4-1. These fields are suggested by a group of meteorologists and researchers at the National Severe Storm Laboratory (NSSL) and the Storm Prediction Center (SPC) in Norman, Oklahoma [see the acknowledgment].

Table 4-1: List of the fields analyzed in SAMEX data.

Field	Description	Unit
ACCPPT	Accumulated precipitation in mm	mm
MSLP	mean sea-level pressure in pascals	pascals
CAPE	Convective available potential energy in J/kg	J/kg
HGT500	500 mb height in m	m
TEMP2M	2 m temperature in K	K
WIND250	250 mb wind velocity in m/s	m/s
ABSVORT250	250 mb absolute vorticity in 1/s	1/s
ABSVORT500	500 mb absolute vorticity in 1/s	1/s
ABSVORT850	850 mb absolute vorticity in 1/s	1/s
DEWP2M	2 m dewpoint in K	K

4.2 A Description of the Ensemble Data

During May 1998, SAMEX personnel collected and stored forecast output from 4 different numerical weather prediction models at three hourly intervals out to 36 hour. The models are the NCEP Eta model (Black 1994), the NCEP Regional Spectral Model (RSM; Juang and Kanamitsu 1994), the CAPS Advanced Regional Prediction System (ARPS; Xue et al. 2000), and The Pennsylvania State University - National Center for Atmospheric Research Mesoscale Model version 5 (MM5; Dudhia 1993; Grell *et al.* 1995) run at NSSL. Forecasts were started each day at 0000 UTC. Each model used a different domain and had different horizontal and vertical resolutions, but the model output fields were interpolated to a common grid that represents the largest shared region from all the models[1] (David Stensrud, 2000, personal communication). This domain consists of 117x81 grid points (a total of 9477 points) covering a large part of the United States of America at 30 km grid spacing (Figure 4-1). The southwest corner of the domain is located at 27.0844 degree north, 110.068 degree west and the northeast corner is located at 45.1552 degree north, 68.10911 degree west.

Three of these models, the Eta, RSM, and ARPS, used the identical model configuration with different initial and boundary conditions. The Eta and RSM were started with a control and 4 bred mode perturbations from the global forecast system (Toth and Kalnay 1993, 1997). The Eta and RSM bred perturbations start with the same initial and boundary conditions, but the RSM also allowed for regional breeding at higher resolution than the global breeding. The ARPS was started with a control initial condition from the Eta model and 4 initial and boundary conditions using the scaled

[1] Note that the MM5 forecasts used a two-way nesting procedure and the MM5 output used is only from the inner domain. The domains from all the models extended well bevond the contiguous 48 states.

lagged average forecasting approach (Hou *et al.* 2000). This approach is an extension of the lagged average forecasting method of Hoffman and Kalnay (1983).

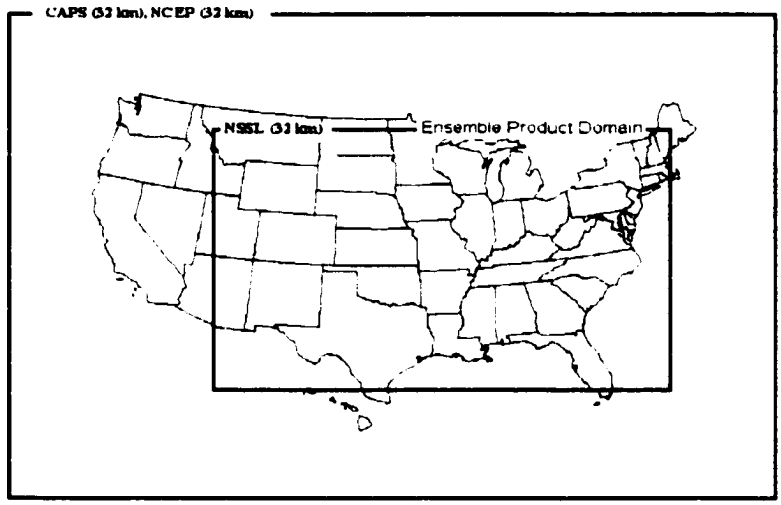


Figure 4-1: Domain of the SAMEX data

The MM5 forecasts also used the Eta model forecasts for the control initial and boundary conditions. Random coherent structures were added to 9 of the initial and boundary conditions to mimic analysis uncertainty (Errico and Baumhefner 1987; Stensrud *et al.* 2000). This approach has slower error growth on the boundaries when compared with the other techniques, but still produces growing modes (Stensrud *et al.* 2000). In addition, 10 different configurations of MM5 were used, one for each forecast. Different physical parameterization schemes for the planetary boundary layer and deep convection were used, and the values of moisture availability were altered. The three convective schemes used were the Betts-Miller (Betts and Miller 1986), Kain-Fritsch (Kain and Fritsch 1990), and Grell (Grell 1993) schemes, and the two planetary boundary layer schemes used were the Blackadar non-local closure scheme (Zhang and Anthes 1982) and the Burk-Thompson 1.5 order closure scheme (Burk and Thompson

1989). Moisture availability was calculated using an antecedent precipitation index from daily rainfall totals over the previous three months and interpolated to the MM5 grid. Values of moisture availability were varied by ± 10% while maintaining the mean value over the domain as in Stensrud *et al.* (2000).

The data set for each forecast field examined in this study represents the output of 4 forecast models starting at 0000 UTC May 29, 1998. Therefore, the ensemble consists of the 25 members as listed in Table 4-2.

Table 4-2: List of institutions and models for the SAMEX ensemble members.

Institution	Model	Member No.
CAPS	ARPS4.3.5	1 – 5
NCEP	Eta	6,8,10,12,14
NCEP	RSM	7,9,11,13,15
NSSL	MM5v2	16 - 25

While the data analysis is not focused on verification of the ensemble data, as in Hou *et al.* (2000), a brief overview of the evolution of the atmosphere on this day is helpful (Alhamed, Lakshmivarahan, and Stensrud 2000). At 0000 UTC 29 May a low pressure center was located in northeastern Canada with a northeast-to-southwest oriented quasi-stationary frontal boundary draped across the northern plains states. Deep convection was present to the north of this frontal boundary in southern Minnesota and was also present over a majority of the southeastern states. The convection over Minnesota organized into a derecho that moved quickly eastward across the northern states, reaching western Michigan by 0500 UTC 29 May and New York by 1400 UTC the same day. The number of damaging wind reports exceeded 170, the number of severe hail reports exceeded 100, and sixteen severe weather watches

were issued during the 36 hours period over which the model forecasts are valid. Deep convection persisted throughout most of the southeastern states throughout this entire time period, while the convection in the northern U.S. shifted eastward with time. At the 36^{th} hour lee cyclogenesis was occurring with a low center in western South Dakota. Thus, even though no distinct midlatitude cyclone passed through the model domain, the weather events on this day were numerous and difficult to forecast.

For each forecast field 13 different data matrices are constructed, one for each output time, with a CA and PCA conducted separately for each output time. The data matrix, X_{mxn}, m=9477, n=25, represents the quantities of the field at each grid point from the 25 ensemble members. Thus, each member is represented by a column with 9477 elements with one element per grid point. The objective of these analyses is to study the behavior of a multi-model ensemble and see how members from different models cluster.

4.3 Analysis of the Accumulated Precipitation (ACCPPT) Field

4.3.1 CA and PCA Based On the Correlation

The correlation matrix (2.13) , R_{25x25}, is first computed for each output time and then used as a similarity matrix for the CA method in order to determine the hierarchical clustering structures of the 25 members of the ensemble. Figure 4-2 shows the clustering dendrograms resulting from applying UPGMA (section 2.2.1) method on the ACCPPT data. The height of the trees and the various levels at which members are merged into clusters indicate that the members are extremely diverging as the time passed. As the Figure 4-2 shows, the members of each model tend to build their own cluster before merging with members from other models.

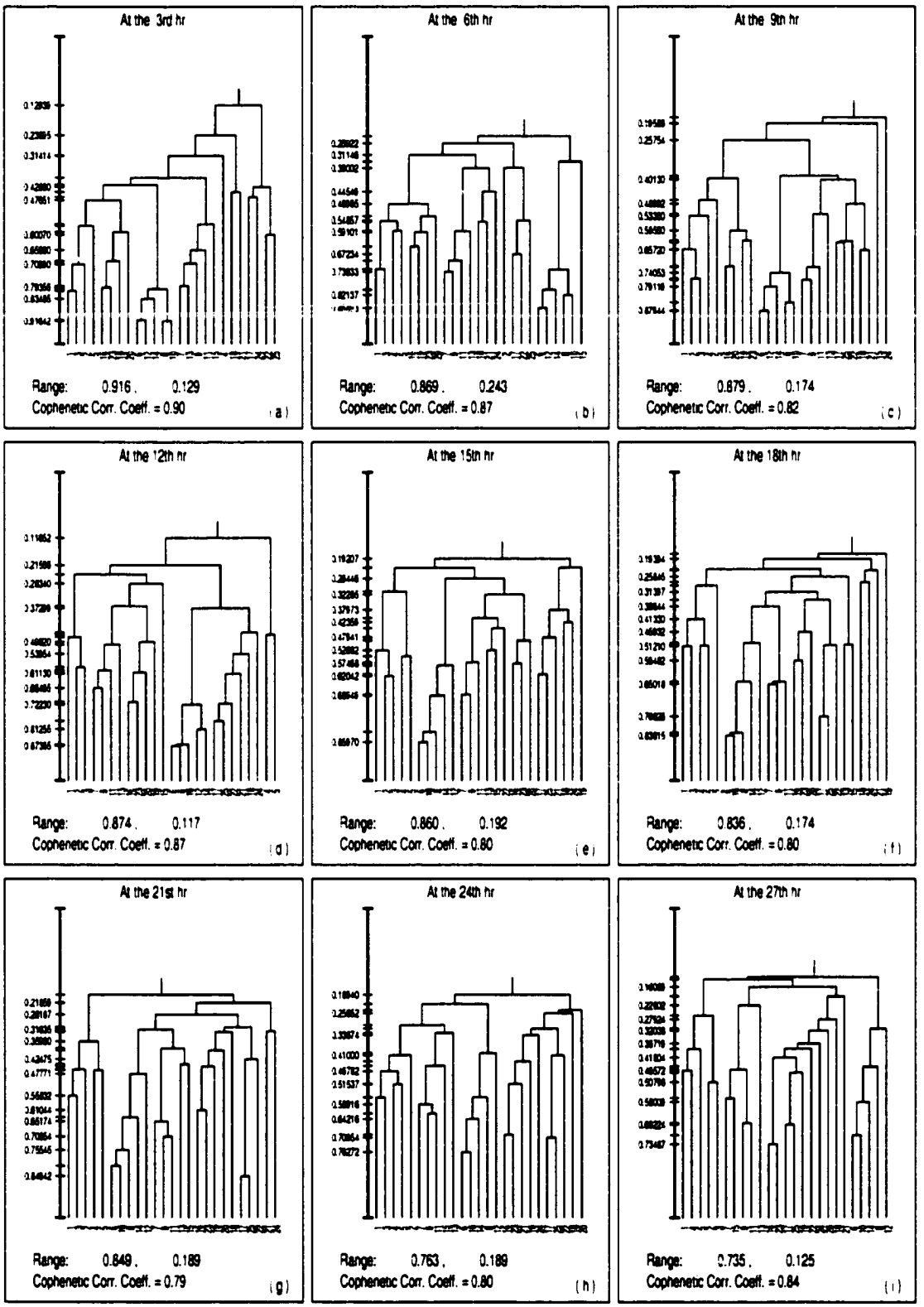


Figure 4-2: UPGMA dendrograms for the Accumulated Precipitation based on the Correlation.

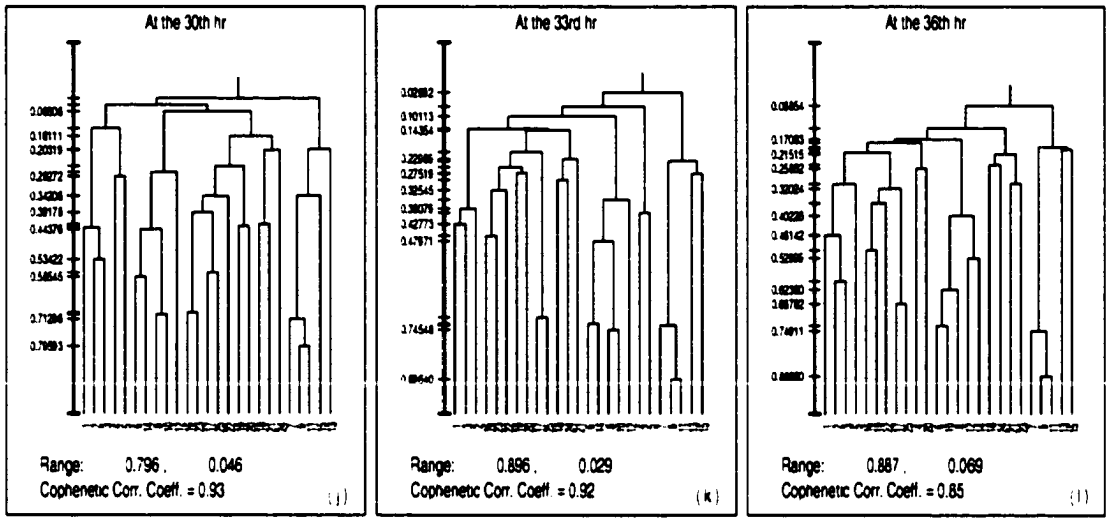


Figure 4-2: Continued

In fact, if the clustering trees of the 24th, 27th and 30th hour are cut to form 4 clusters, the resultant clusters represent the 4 models. At the other output times, when members from different models join each other to form a cluster, the merge occur at a very low level of correlation indicating the lack of inter-similarity between the models. This tendency points out the coherence between the members from one model and the isolation between the members from different models.

To investigate the meteorological significance of these results, plots of the 3 hourly accumulated rainfall valid at the 30th hour are shown (Figure 4-3). A visual examination of these data indicates that the rainfall patterns cluster by model first. Even though the Eta model and RSM forecasts are initialized using different bred modes, the precipitation forecasts are amazingly similar in location of the precipitation regions from one forecast to another. For example, all the Eta model forecasts have a local precipitation region in western Illinois and along the Gulf coast. Similarly, all the RSM forecasts have a local precipitation region along the Minnesota - South Dakota border

with varying amounts of precipitation along a bowing line from Iowa to New York. Looking at the other forecasts, note that all the ARPS forecasts produce a precipitation region in Kansas and produce much less precipitation over the Gulf of Mexico when compared with the Eta model and RSM forecasts. Finally, all the MM5 forecasts produce precipitation from southern Indiana eastward into Pennsylvania, but have a greater variety of precipitation patterns and amounts elsewhere in the domain. Examination of other time periods (not shown) produce similar conclusions. The subjective analyses are largely in agreement with the CA results in that the forecasts cluster first by model, indicating a strong coherence in forecasts from the same numerical weather prediction model (David Stensrud, 2000, personal communication).

This result suggests that the model framework and physical parameterization schemes dominate the evolution of the precipitation forecast. Recall that both the Eta model and the RSM use the breeding of growing modes technique, based upon the global model output, to produce the different initial and boundary conditions for their 5 member ensembles. Yet these models rarely cluster together first, and when they do cluster together the level of correlation is low (Figure 4-2). These results agree with those of Stensrud *et al.* (2000) who show that model physics is a very important component of short-range ensemble systems and can dominate over initial condition uncertainty when the large-scale forcing for upward motion is weak. However, the large-scale forcing for this case is not weak, as a relatively strong short-wave was passing through the upper midwest. Yet model physics still appears to play a very important role in the evolution of the model forecasts of precipitation (Alhamed, Lakshmivarahan, and Stensrud 2000).

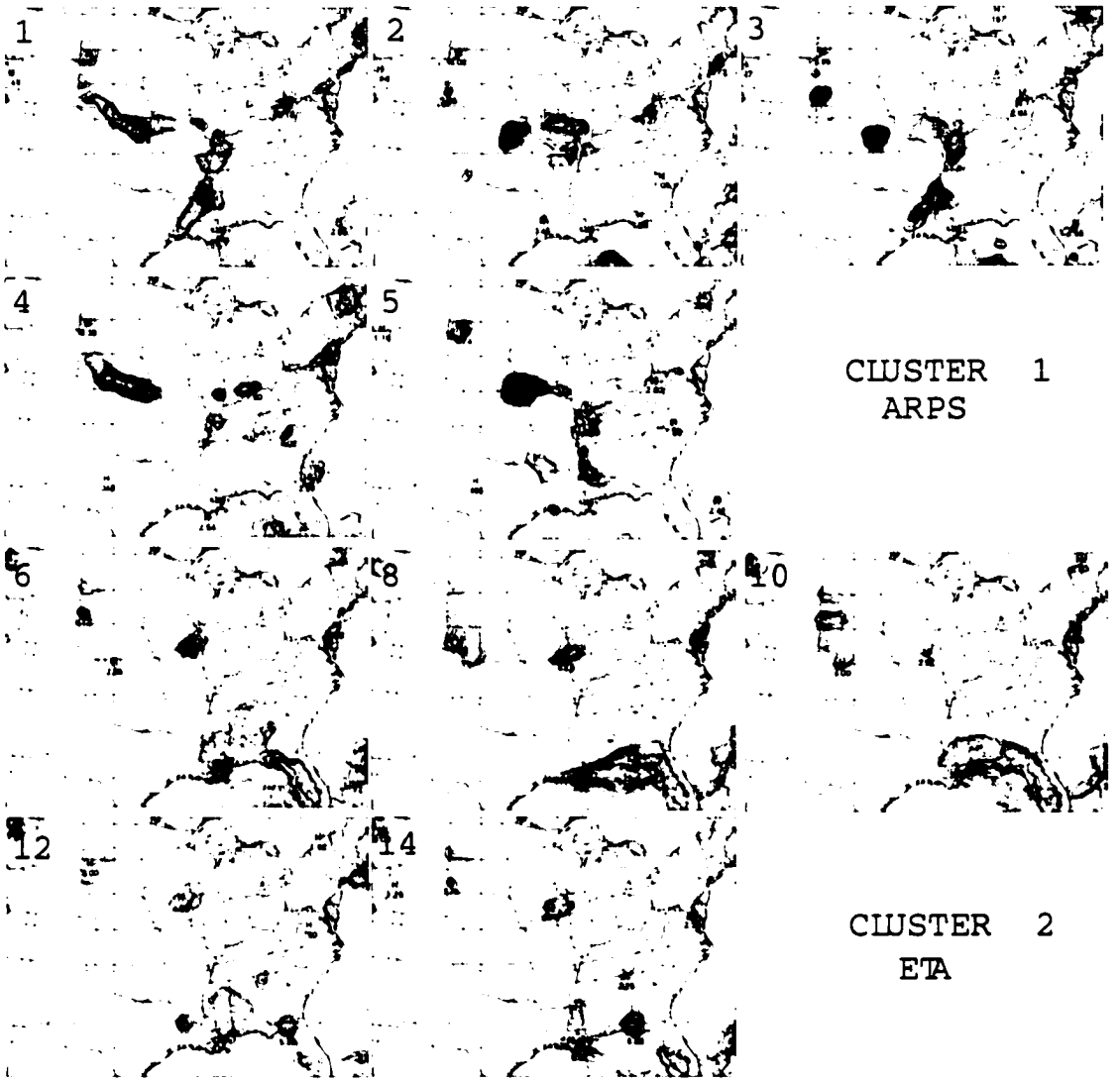


Figure 4-3: 3-h accumulated precipitation valid at 30th hour from all 25 ensemble members grouped subjectively into 4 clusters. Numbers in the upper-left hand corner indicate the ensemble member as defined in Table 4-2. Isolines every 1 mm.

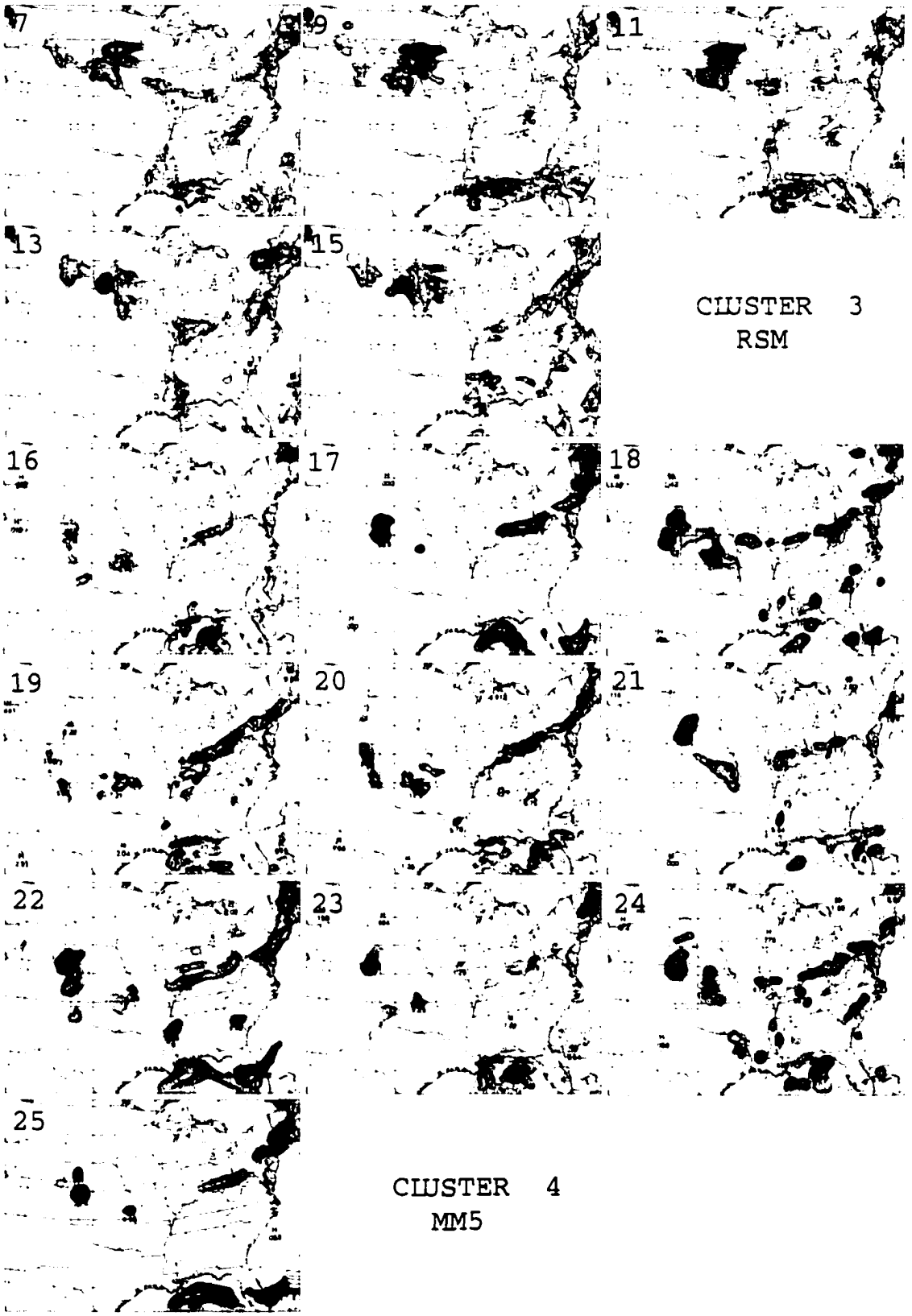


Figure 4-3: Continued.

To further explore this ensemble behavior, the eigenvalues of the correlation matrices for the 13 data matrices are computed. The variances that explained by the first eigenvalues for the 13 correlation matrices indicate that the data are not concentrated on one dimension. Hence, the difference between the variance explained by the first eigenvalue at the 3rd hour, 39.2%, and at the 36th hour, 20.9%, points out the slight divergence of the members of the ensemble as the time passes.

However, this behavior of the multi-model data set is not the same for the individual models. When the correlation matrices for the individual models are constructed and their eigenvalues are computed, a large spread in the variance explained by the first eigenvalue is found from the four models (Figure 4-4). The precipitation forecasts from the Eta model are most alike, and the forecasts from MM5 are most different, with the RSM and ARPS in between. Note that three of the models have percentages of variance explained above 50% throughout most of the forecast time period. This again highlights the important role played by model physics in the

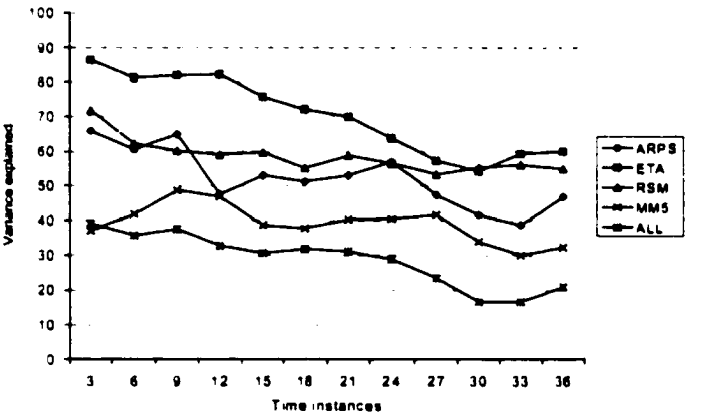


Figure 4-4: Comparison of the variance explained by the first eigenvalue across the four models for the ACCPPT

generation of precipitation forecasts, since the one individual model with variations in model physics has the lowest variance explained by the first eigenvalue.

Clustering the 25 members of the ensemble using PCA (section 2.3) yields a similar result as found with CA when quartimax (section 2.3.2 A), an orthogonal rotation algorithm, is used. Using quartimax, a moderate simple structure is obtained by rotating the first 4 PCs (Table 4-3). Several variables in this solution (e.g., 1, 4, 13, 14, and 15) do not load highly on one PC, hence their clustering is fuzzy. The loading of variable 14, for instance, is divided nearly equal on PC3 and PC4. That is, this variable is overlapped between the Eta cluster and the ARPS cluster. Although this variable (i.e., 14) is placed into the Eta cluster by both CA and PCA, visual examination of the contour plots for the variables at the 36^{th} hour (not shown) indicates that variable 14 is similar to variable 5 from the forth cluster. This observation points out the ability of PCA to reveal the overlapping of variables between the resultant clusters.

Examination of the membership of the 4 clusters in Table 4-3 indicates that the members of each cluster are all belong to a single model; there is no inter-model clustering found. The results of PCA analyses of the precipitation output at 27^{th}, 30^{th}, and 33^{rd} hours (not shown) lead to the same solution, that is, 4 clusters represent the 4 models.

The results from both the correlation-based CA and PCA agree on the tendency of the members from one model to be in one cluster. Therefore, the correlation between the forecast members from a single model is usually higher than the correlation between forecast members from different models in this data set.

Table 4-3: Rotated loadings of the first 4 PCs using quartimax for the precipitation data at the 36th hour based on the correlation.

n	PC 1	PC 2	PC 3	PC 4
1	0.43	0.07	0.15	**0.62**
2	0.09	0.03	0.04	**0.81**
3	0.27	0.05	0.07	**0.62**
4	0.22	0.07	0.16	**0.29**
5	0.12	0.01	0.14	**0.62**
6	0.06	0.02	**0.85**	0.21
7	0.01	**0.89**	0.04	0.03
8	0.35	0.10	**0.75**	-0.13
9	0.05	**0.94**	0.03	0.02
10	0.19	0.13	**0.86**	-0.07
11	0.05	**0.91**	0.09	0.02
12	-0.25	-0.02	**0.65**	0.33
13	-0.05	**0.30**	0.19	0.15
14	-0.12	-0.01	**0.60**	0.55
15	0.01	**0.32**	0.18	-0.01
16	**0.67**	0.10	0.13	0.16
17	**0.60**	0.01	0.13	0.10
18	**0.46**	-0.02	-0.06	0.06
19	**0.59**	0.10	0.18	0.27
20	**0.44**	0.05	0.04	0.14
21	**0.38**	0.01	0.06	-0.04
22	**0.50**	0.09	0.21	-0.06
23	**0.56**	0.04	0.07	0.14
24	**0.45**	-0.02	0.18	-0.08
25	**0.65**	0.02	0.09	0.12

4.3.2 CA and PCA Based On the Euclidean Similarity (EucSimi)

The purpose of this analysis is to provide a dissimilarity-based PCA analysis for the sake of comparison since the Euclidean Similarity (EucSimi) (2.16) is derived from a dissimilarity measure (i.e., the Euclidean distance).

For each output time, the data matrix is first centered (2.3) and then the Euclidean similarity matrix, $\hat{E}_{25\times25}$, is computed. Figure 4-5 shows the clustering dendrograms resulted from applying UPGMA method (section 2.2.1) based on the EucSimi for the ACCPPT data. The overall result is different from its correlation-based counterpart (section 4.3.1). Although the tendency of one model to form one cluster still exists to some extent, inter-models cluster are found in several cases. At the finishing time, one large cluster containing members from different models is formed at relatively early stage of the hierarchy, which points out the similarity between these members.

When the Euclidean similarity clusters are compared with the model forecast fields at the 30th hour (Figure 4-3), it is more challenging to explain the cluster groupings. Euclidean similarity is influenced by the precipitation amplitudes in addition to location (Elmore and Richman 2000), such that the clusters may in part reflect the differences in the precipitation maximum values. For this output time the Eta model and ARPS typically have lower maximum values of precipitation than the RSM and MM5. Note that the Eta model and ARPS are grouped together before the other two models (except for ARPS member 5, which has the highest precipitation maximum of all the ARPS forecasts). Therefore, which cluster measure one prefers may depend upon how important are precipitation amplitudes in comparison with precipitation patterns.

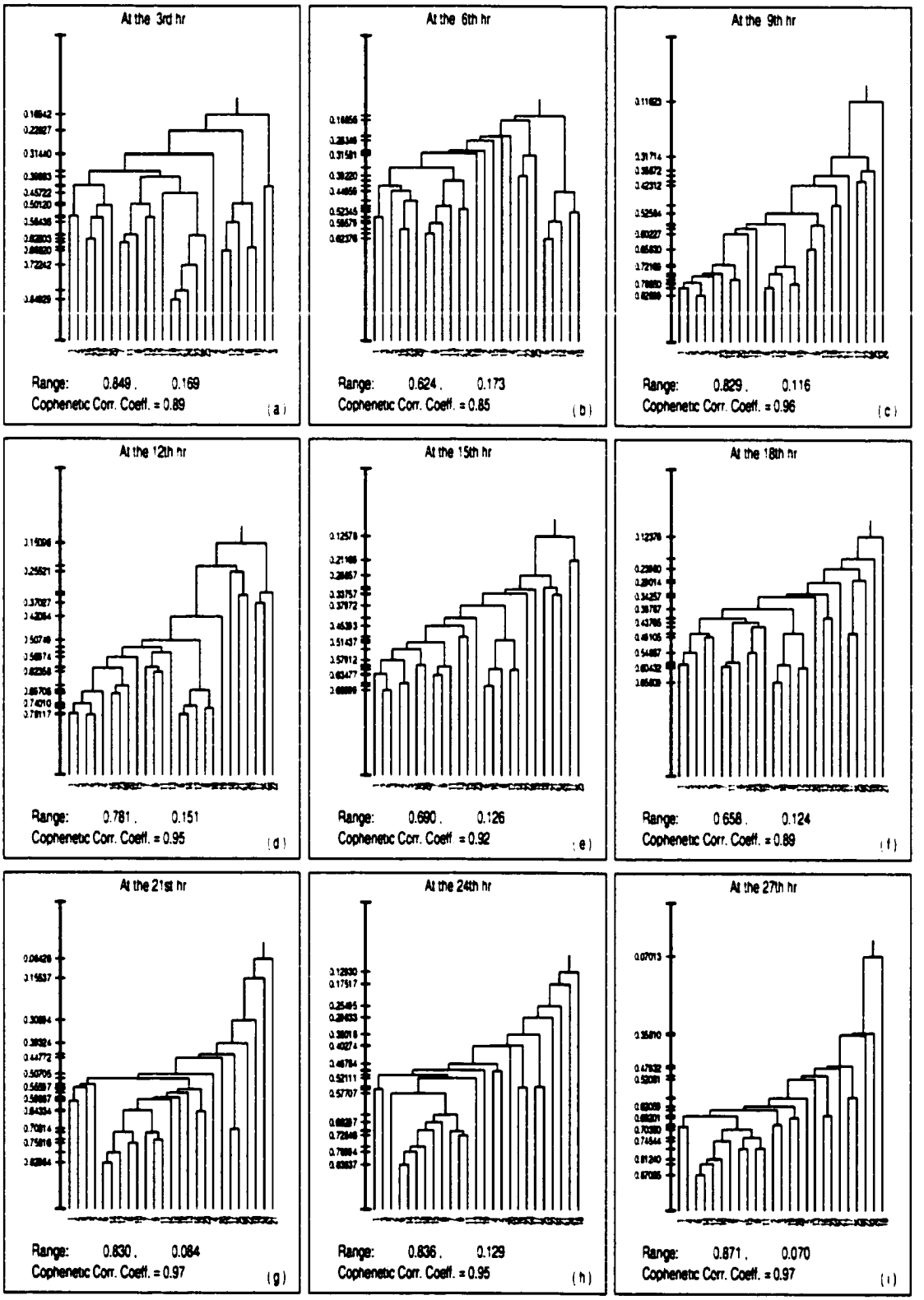


Figure 4-5: UPGMA dendrograms for the Accumulated Precipitation based on the Euclidean Similarity; data is centered using (2.3).

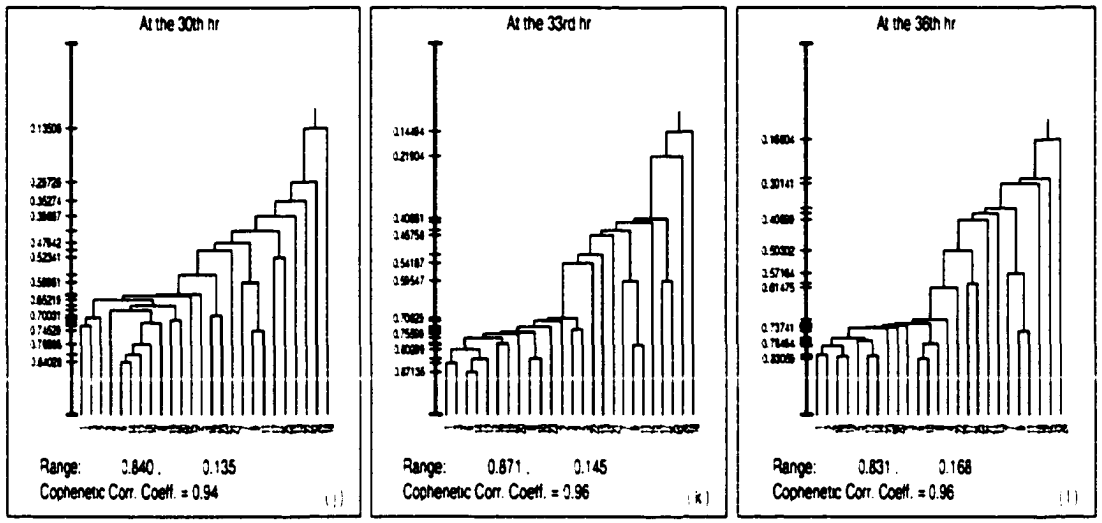


Figure 4-5: Continued

A PCA (section 2.3) is then conducted using the same Euclidean similarity matrices. A strong simple structure is obtained by rotating 4 PCs using the promax algorithm (section 2.3.2 B) for the precipitation data at the 36th hour. By examining the resulting promax simple structure, Table 4-4a, the clustering structure of the 25 members at the 36th hour is as follows: PC1 clusters 1-6, 8, 10, 12-16, and 19-23; PC2 clusters 7, 9, and 11; PC3 clusters 24; and PC4 clusters 17, 18 and 25. It is seen that the first cluster also has members from different models. Yet there remain instances where the forecasts from a single model tend to cluster together, even though they are part of a larger cluster. The members of RSM model are separated into 2 clusters, and members of MM5 model are scattered into 3 clusters. The correlation matrix of the rotated 4 PCs at the 36th hour, Table 4-4b, shows that the 4 clusters are well separated. The overall results of the PCA analysis are very similar to those from the CA-based analysis using the Euclidean Similarity measure.

Table 4-4a: Rotated loadings of the first 4 PCs using promax for the precipitation data at the 36th hour based on the Euclidean Similarity.

n	PC1	PC2	PC3	PC4
1	**0.88**	0.00	0.02	-0.05
2	**0.89**	-0.01	0.02	0.00
3	**0.84**	0.00	0.05	0.00
4	**0.83**	0.01	0.04	0.00
5	**0.86**	-0.01	0.01	-0.03
6	**0.90**	-0.02	0.01	0.04
7	0.01	**0.82**	-0.05	0.05
8	**0.84**	0.03	0.10	0.03
9	0.11	**0.85**	0.02	0.00
10	**0.87**	0.02	0.05	0.04
11	0.12	**0.81**	0.00	-0.01
12	**0.92**	-0.02	0.02	0.08
13	**0.54**	0.02	-0.28	0.07
14	**0.92**	-0.02	0.01	0.05
15	**0.57**	-0.06	-0.40	0.25
16	**0.84**	0.01	0.03	-0.09
17	0.33	0.00	-0.10	**-0.59**
18	0.15	-0.06	-0.04	**-0.62**
19	**0.85**	0.01	0.02	-0.08
20	**0.81**	0.01	0.03	-0.07
21	**0.48**	-0.01	0.22	-0.08
22	**0.62**	0.05	-0.01	-0.19
23	**0.82**	0.01	0.08	-0.05
24	0.35	-0.02	**0.86**	0.17
25	0.30	0.00	-0.14	**-0.66**

Table 4-4b: the correlation matrix of the 4 rotated PCs (promax) for the precipitation data set at the 36th hour.

	1	2	3	4
1	1.00	0.48	-0.03	-0.43
2	0.48	1.00	-0.09	-0.19
3	-0.03	-0.09	1.00	-0.22
4	-0.43	-0.19	-0.22	1.00

4.3.3 CA Based On the Euclidean distance

The data matrices are first centered (2.3) by subtracting the column mean from each element in the column. For each output time, a distance matrix, E_{25x25}, based on Euclidean distance (2.10) is computed and then used as the dissimilarity matrix for the CA method. Figure 4-6 shows the clustering dendrograms resulted from applying Ward's method (section 2.2.1) on the ACCPPT data set.

It is expected that the resultant hierarchies from applying Ward's method will be different from their UPGMA counterparts (section 4.3.1). That is, in fact, due to the sensitivity of cluster analysis to the changes of the clustering methods or the resemblance measures. By examining the clustering trees, it is not possible to obtain a clustering structure that corresponds to the 4 models by cutting the trees to form 4 clusters at any output time. Although the 4 models clustering structure does not exist, the tendencies observed in the previous analyses still exist. That is, the tendency of the members of one model to build their own cluster before merging with members from other models. However, the inter-models clustering can be seen in several cases. The nature of the Euclidean distance makes it difficult to assess the degree of similarity between the members of a given cluster. But inter-model clusters are formed at earlier stages of the hierarchies of Ward's comparing to the correlation-based. These results, and those from the similarity measures, suggest that the Euclidean-based CA and PCA view the members of the ensemble differently and reveal the inter-similarity embedded in the 4 models which, collectively, construct the ensemble. However, the correlation-based clustering solution is closer to the truth as shown by the expert (section 4.3.1).

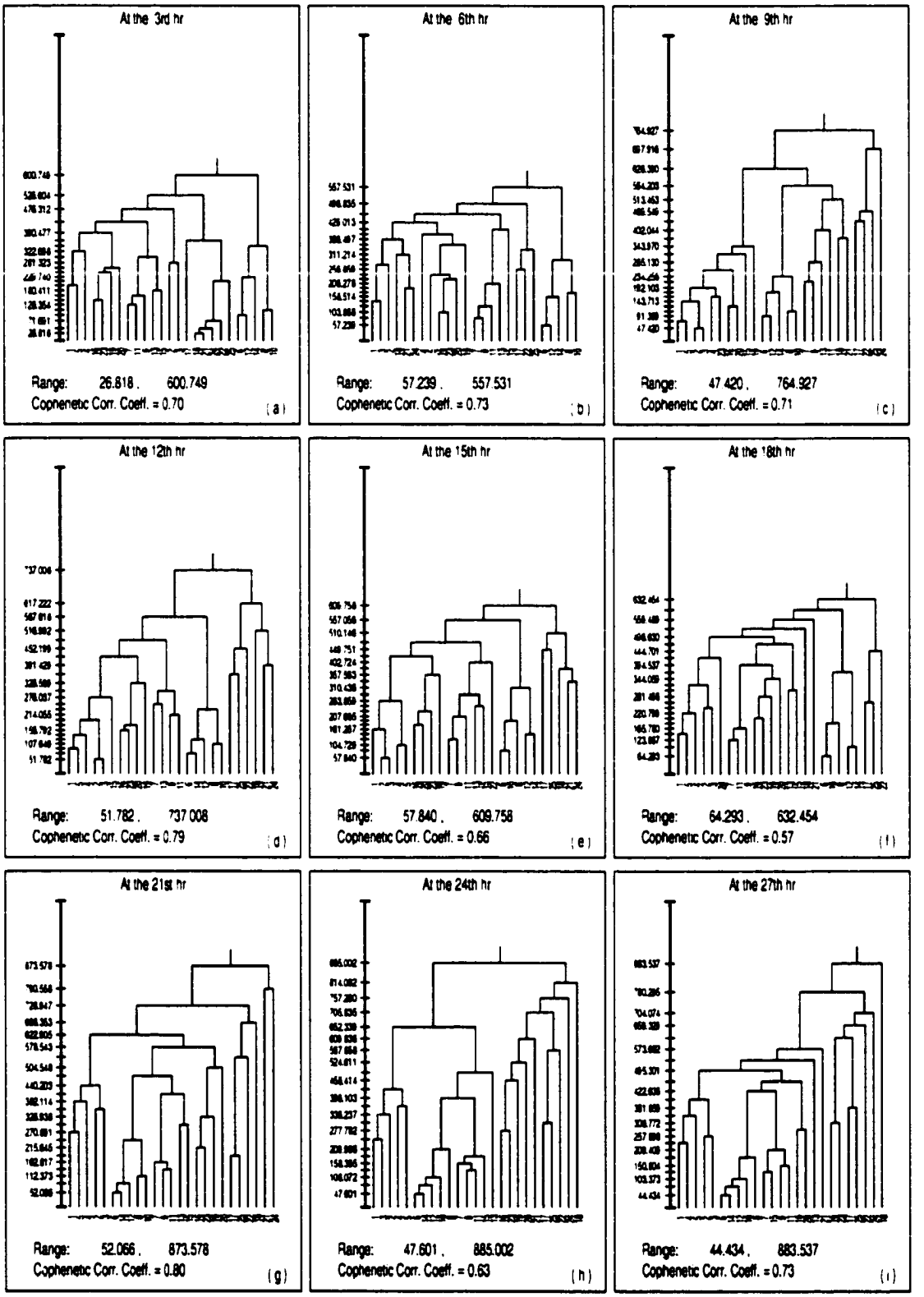


Figure 4-6: Ward dendrograms for the Accumulated Precipitation based on the Euclidean distance; data is centered using (2.3).

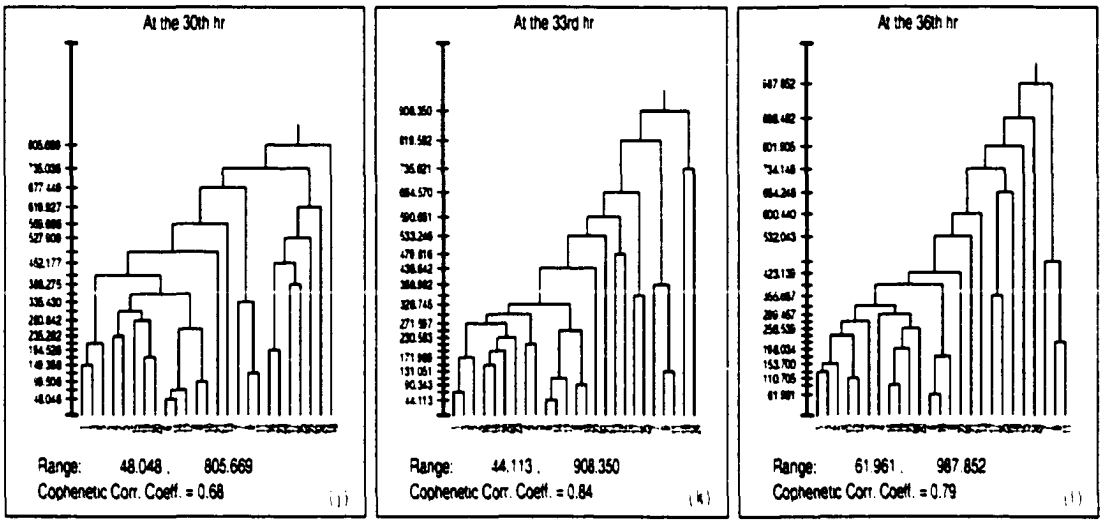


Figure 4-6: Continued

4.3.4 CA Based On the *K*mean and Nucleated Agglomerative (NA) algorithms

Our results thus far are determined only using hierarchical clustering methods. To further explore the ensemble data, we now turn to two non-hierarchical algorithms, *K*mean and Nucleated Agglomerative (NA). Non-hierarchical methods (nHCA) (section 2.2.2) require the number of clusters and the seed point for each cluster to be specified in advance. We first apply *K*mean on the ACCPPT data set and then NA is applied. For each output time of this data set, *K*mean algorithm (section 2.2.2 B) is applied. The number of clusters is chosen to be four, k=4, and the seed points are specified subjectively by choosing one seed from each model. The clustering structures resulted from applying *K*mean are shown in Table 4-5. Although the tendency of the ensemble member to group according to their model showed up, the *K*mean is able to find at least one inter-model cluster for each output time. To test whether the inter-model clusters will break into small clusters each of which having members from one model, the *K*mean is run for the 36th hour data with the number of clusters successively increasing

by one. The seed points again are specified subjectively. With k values of 5 to 19, *K*mean continues to produce inter-model clusters (not shown). In fact, *K*mean produces small clusters-with three members, each of which from different models. With k=20, no inter-models cluster is found by the *K*mean algorithm.

Table 4-5: *K*mean's partitions for the precipitation data set.

T	Clusters			
3	[1,4,5,16,19,20,23]	[6,8,10,12,14]	[7,9,11,13,15,17, 18, 21,22,24,25]	[2,3]
6	[1,2,3,4,5,16,19,20,23]	[6,8,10,12,14]	[7,9,11,13,15,18, 21,24]	[17,22,25]
9	[1,2,3,4,5,15,16,19,20, 23]	[6,8,10,12,14, 21,24]	[7,9,11,13,18]	[17,22,25]
12	[1,2,3,4,5,6,8,10,12,14, 15]	[16,19,20,23]	[7,9,11,13]	[17,18,21,22,24, 25]
15	[1,2,3,4,5,16,19,20,23]	[6,8,10,12,14]	[7,9,11,13,15,22]	[17,18,21,24,25]
18	[1, 2, 3, 4, 5, 12, 14, 19, 24]	[6, 8, 10, 17, 22, 25]	[7, 9,11,13,15]	[16,18,20,21,23]
21	[1,2,3,4,5]	[21]	[6,7,8,9,10,11,12 ,13,14,15,16,19, 20,23]	[17,18,22,24,25]
24	[1,2,3,4,5]	[16,21,23,24]	[6,7,8,9,10,11,12 ,13,14,15,19,20]	[17,18,22,25]
27	[1,2,3,4,5]	[18]	[6,7,8,9,10,11,12 ,13,14,15,19,20, 21]	[16,17,22,23,24, 25]
30	[1,2,3,4,5,6,8,10,12,13, 14,15,19,20,21]	[18]	[7, 9,11]	[16,17,22,23,24, 25]
33	[1,2,3,4,5,6,8,10,12,14, 16,19,20,21,22,23]	[13,15]	[7,9,11]	[17,18,24,25]
36	[1,2,3,4,5,6,8,10,12,13, 14,15,16,19,20,21,22,23]	[24]	[7,9,11]	[17,18,25]

The NA algorithm (section 2.2.2 B) is then applied in the same manner for the same data set. The number of clusters is chosen to be 4, k=4; and the seed points are specified subjectively to span the 4 models. Table 4-6, shows the resultant partitions with respects to 4 clusters. The results of NA algorithm are consistent with the *K*mean counterparts. In all output times, NA finds an inter-model cluster.

Table 4-6: NA's partitions for the precipitation data set.

T	Clusters			
3	[1, 4, 5, 7, 9, 11, 13, 15, 16, 19, 20, 23]	[6, 8, 10, 12, 14]	[2, 3]	[17, 18, 21, 22, 24, 25]
6	[1, 2, 3, 4, 5, 16, 18, 19, 20, 21, 23, 24]	[6, 8, 10, 12, 14]	[7, 9, 11, 13, 15]	[17, 22, 25]
9	[1, 2, 3, 4, 5, 15, 16, 19, 20, 23]	[6, 7, 8, 9, 10, 11, 12, 13, 14, 18, 21]	[24]	[17, 22, 25]
12	[1, 2, 3, 4, 5, 15, 16, 19, 20, 23]	[6, 8, 10, 12, 14]	[7, 9, 11, 13]	[17, 18, 21, 22, 24, 25]
15	[1, 2, 3, 4, 5, 7, 9, 11, 13, 15, 16, 19, 20, 22, 23]	[6, 8, 10, 12, 14]	[18,21, 24]	[17, 25]
18	[1, 2, 3, 4, 5, 7, 9, 11, 13, 15, 19, 24]	[6, 8, 10, 12, 14]	[16, 18, 20, 21, 23]	[17, 22, 25]
21	[1, 2, 3, 4, 5]	[6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 19, 20, 23]	[21]	[17, 18, 22, 24, 25]
24	[1, 2, 3, 4, 5]	[6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 19, 20]	[16, 21, 23, 24]	[17, 18, 22, 25]
27	[1, 2, 3, 4, 5]	[6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 19, 20, 21]	[18]	[16, 17, 22, 23, 24, 25]
30	[1, 2, 3, 4, 5, 6, 8, 10, 12, 13, 14, 15, 19, 20, 21]	[18]	[7, 9, 11]	[16, 17, 22, 23, 24, 25]
33	[1, 2, 3, 4, 5, 6, 8, 10, 12, 13, 14, 15, 16, 19, 20, 21, 22, 23]	[18]	[7, 9, 11]	[17, 24, 25]
36	[1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 16, 17, 18, 19, 20, 21, 22, 23, 25]	[24]	[7, 9, 11]	[13,15]

4.4 Analysis of the Mean Sea-Level Pressure (MSLP) Field

While the 3-hour accumulated precipitation data are likely the most variable of all the data sets saved during SAMEX, the discontinuous nature of the precipitation field makes interpreting the clustering results more difficult. Therefore, other standard output fields, that are smoothly varying, also are examined. In this section, we examine the Mean Sea-Level Pressure (MSLP). Other fields are examined in the subsequent section.

4.4.1 CA and PCA Based On the Correlation

The correlation matrix (2.3), R_{25x25}, is first computed for each output time and then used as a similarity matrix for the CA method. Figure 4-7 shows the clustering dendrograms resulting from applying UPGMA method (section 2.2.1) on the MSLP data. At the initial time (0^{th} hour) there are two main clusters. The first cluster is formed out of the members of ARPS, Eta and RSM models; and the second cluster is built from the members of MM5 model. At the 3^{rd} hour there are three main clusters. The members of ARPS and Eta models form the first cluster, and the other two clusters are formed by the members of RSM and MM5 models, respectively. Starting from the 6^{th} hour, the members of the ensemble follow the same pattern. At each time, thereafter, the members form 3 clusters at high degree of correlation (more than 0.8). The first cluster contains the members of ARPS model, the second cluster contains the members of both Eta and RSM models – both are NCEP's models-, and the third cluster contains the members of MM5 model. Note that the members of Eta model moved from the first cluster at the 3^{rd} hour to join the members from RSM to build the second cluster at the 6^{th} hour. The height of the trees is increasing as the time passes which indicates the divergence. However, this divergence is happening between the 3 clusters themselves and not between the members of the individual cluster. As for the individual clusters, they continue to be formed at the same degree of correlation. This indicates the compactness of the individual clusters and the growing isolation of the 3 clusters as time passes in view of the correlation of the 25 members. Therefore, the members are grouped into 3 clusters corresponding to the 3 institutions that collectively construct the ensemble.

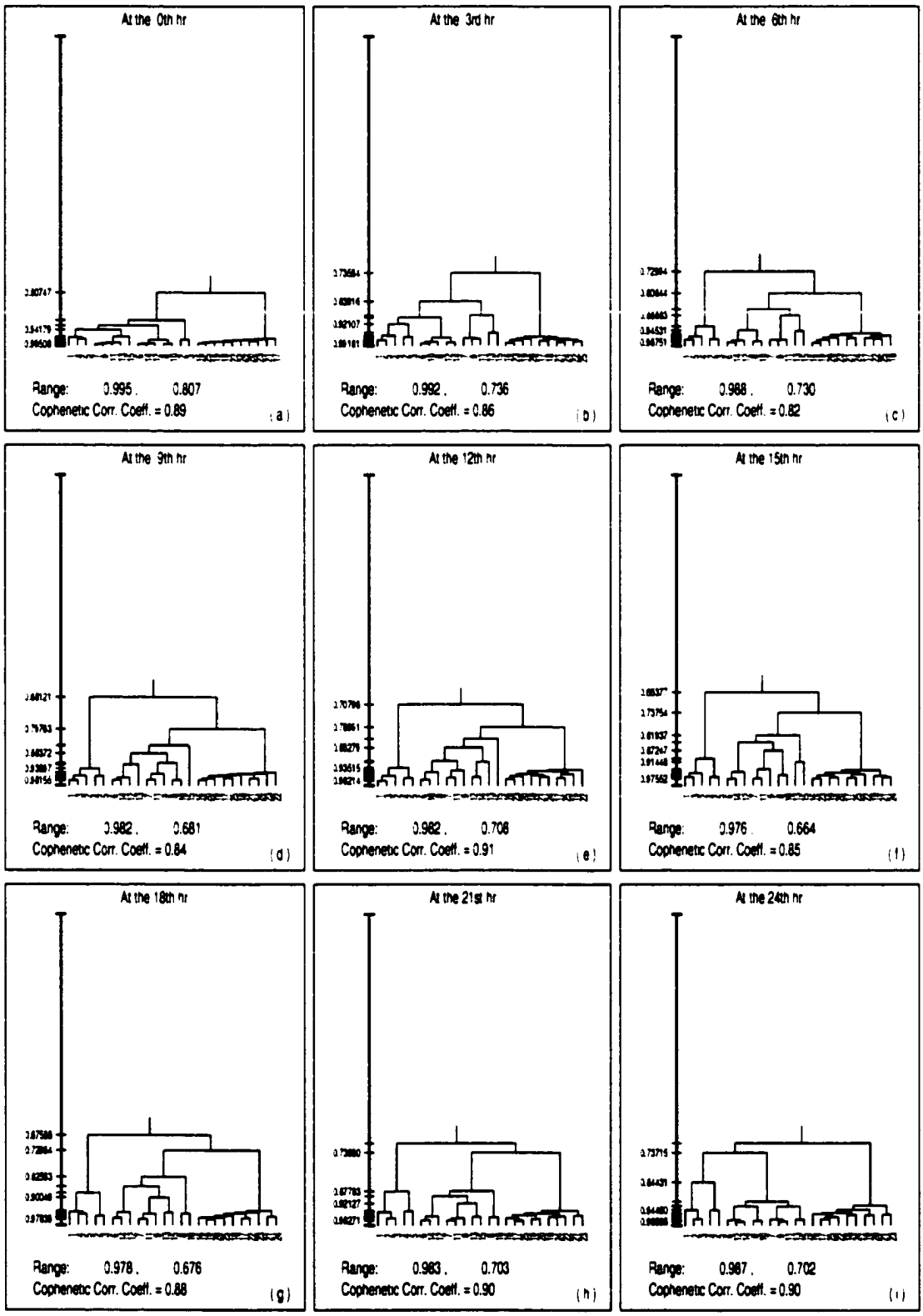


Figure 4-7: UPGMA dendrograms for the Pressure field based on the Correlation.

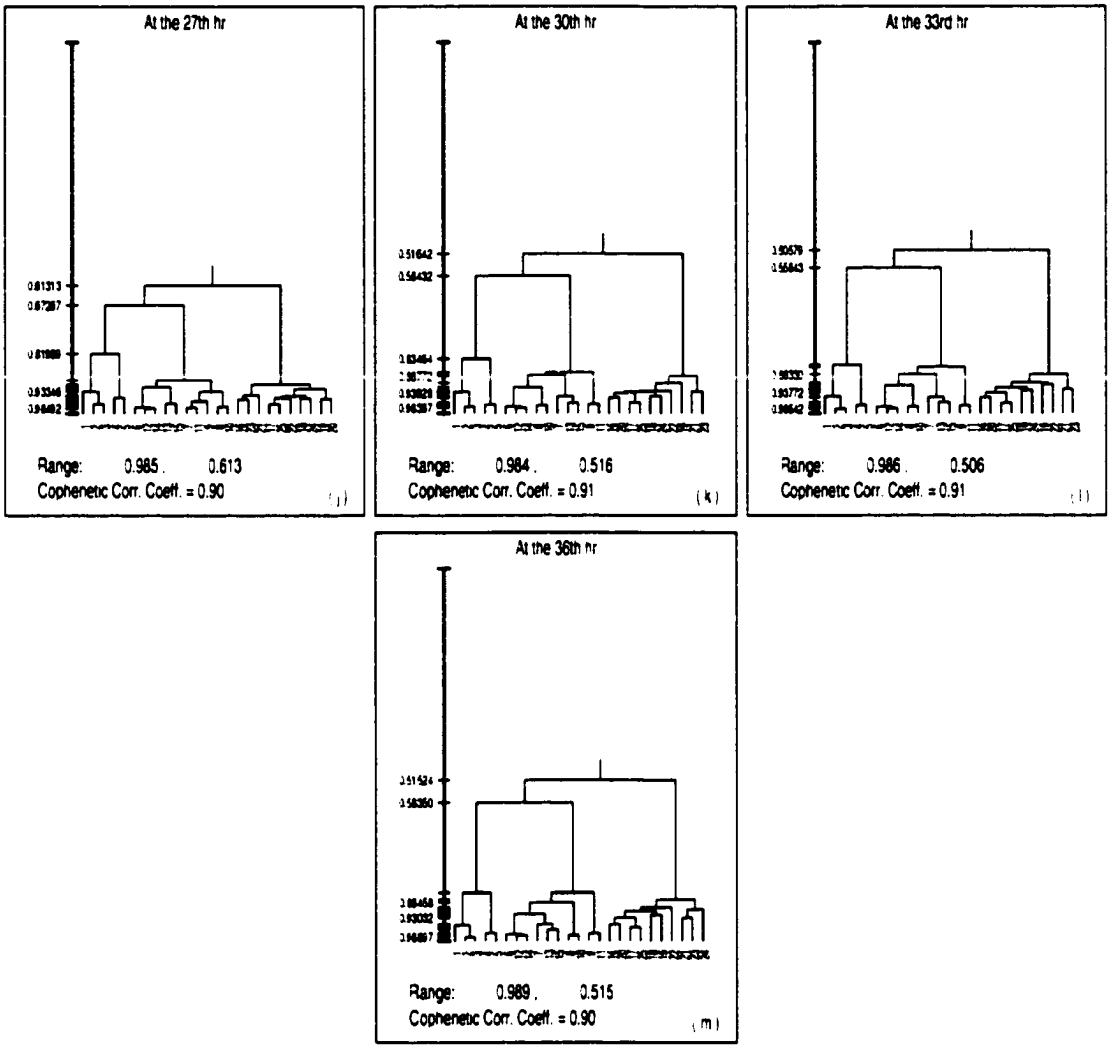


Figure 4-7: Continued

Since mean sea-level pressure is a derived field and not explicitly forecast by the models, the differences in the sea-level pressure derivations could explain the clustering found. A close examination of the fields (Figure 4-8), however, indicates that this conclusion is not warranted. Sea-level pressure fields at the 36th hour from the model forecasts show significant differences in the locations of important, large-scale meteorological features (David Stensrud, 2000, personal communication).

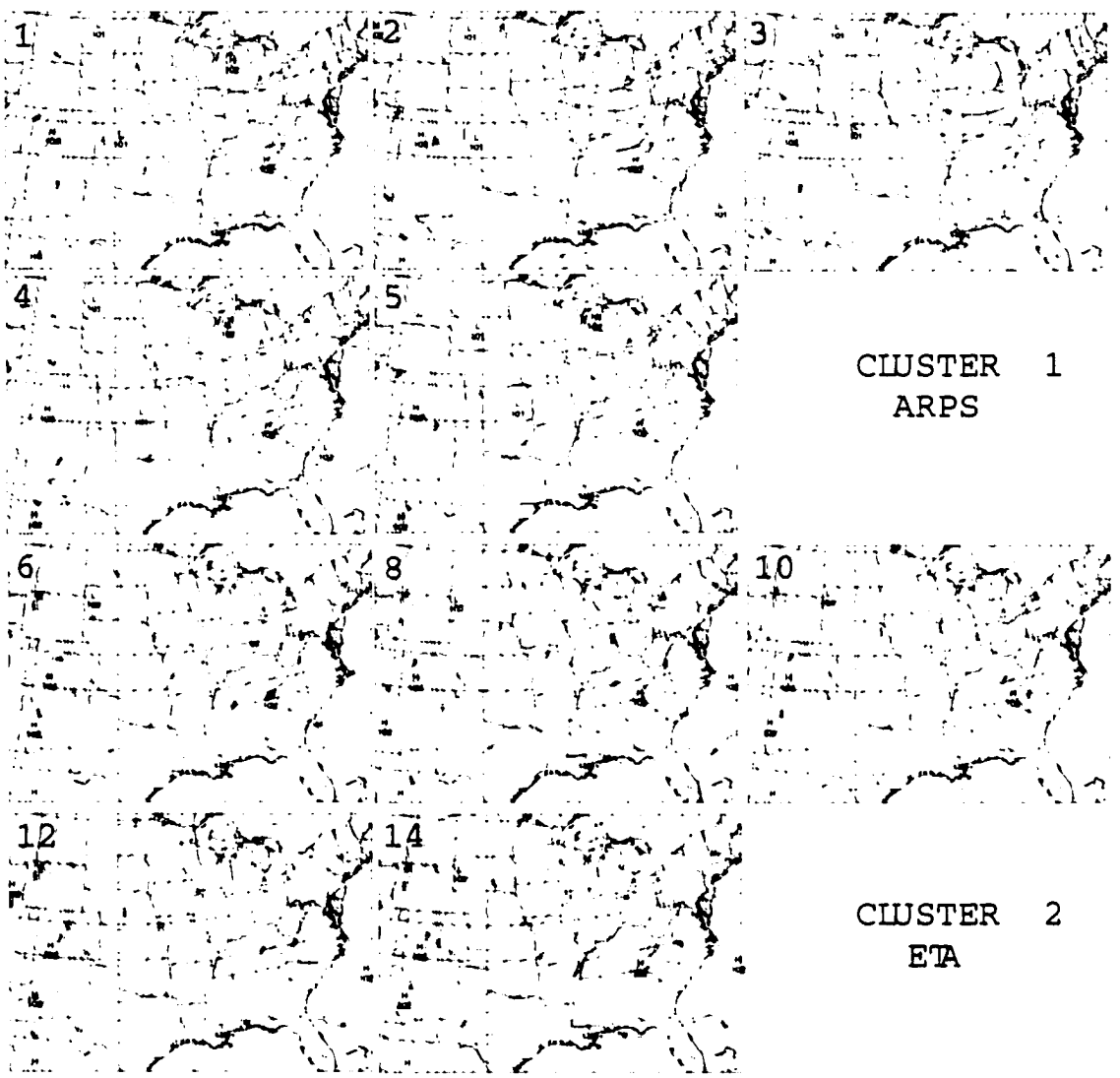


Figure 4-8: Mean sea-level pressure valid at 36th hour from all 25 ensemble members grouped subjectively into 4 clusters. Numbers in the upper-left hand corner indicate the ensemble member as defined in Table 4-2. Isolines every 200 Pa. Relative high and low pressure regions are indicated.

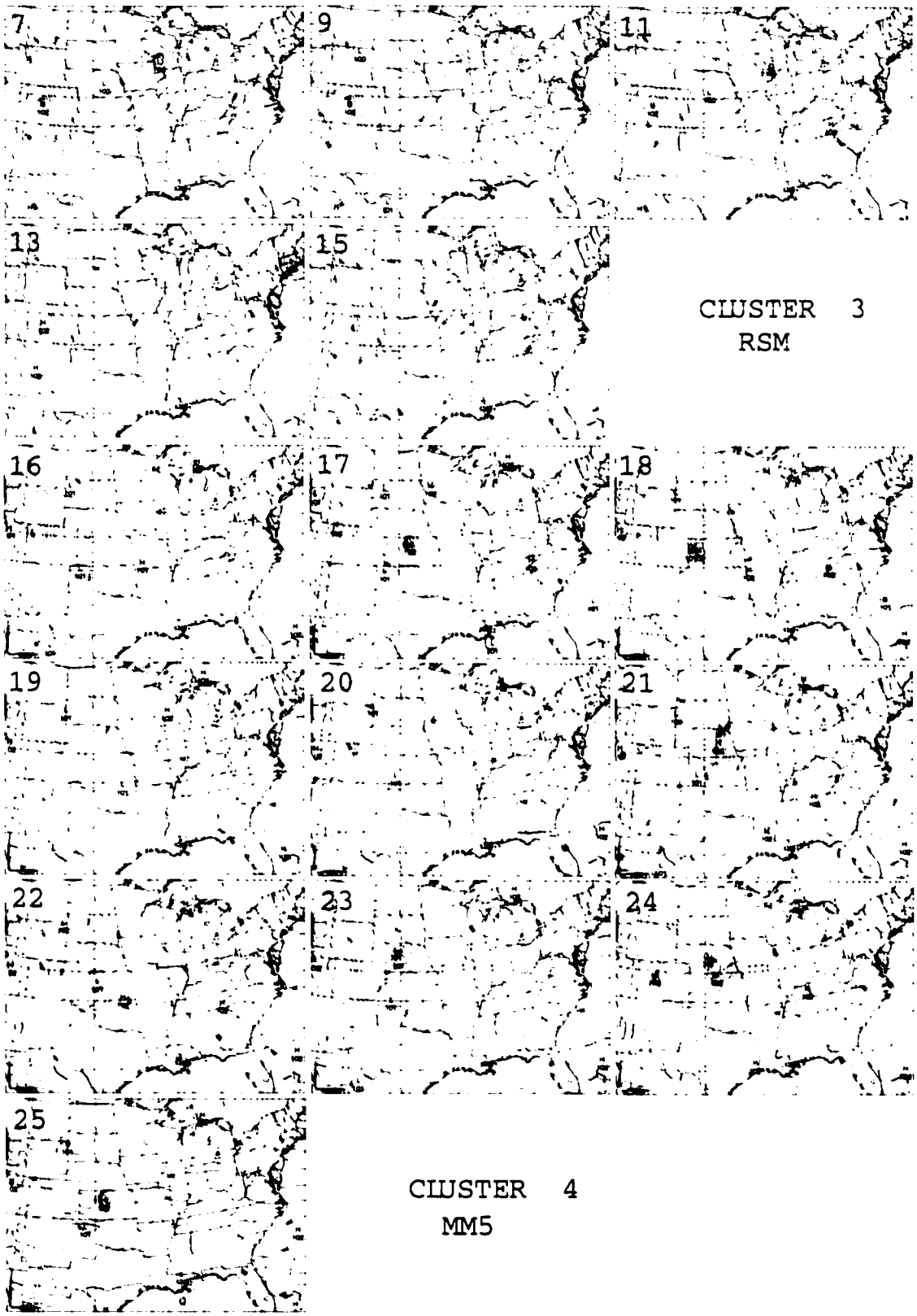


Figure 4-8: Continued.

As shown by Figure 4-8, the ARPS forecasts all indicate a pressure trough to the east of the Rocky Mountains with a distinct low center located somewhere in Kansas. A strong region of high pressure is located over the Great Lakes region. In contrast, both the Eta model and RSM have a single low pressure center in the northern plains region and a much weaker region of higher pressure over the Great Lakes. The MM5 forecasts provide yet a third general solution, with various weaker low pressure centers in the plains states and a more distinct region of high pressure over the Great Lakes region. The greater structure in the sea-level pressure fields over the plains states, seen in the MM5 forecasts, is due in part to the presence of deep convection in the model forecasts in this region, which influences the surface pressure through the inclusion of downdrafts in two of the convective parameterization schemes used and the explicit microphysical parameterization (David Stensrud, 2000, personal communication). The patterns of sea-level pressure from the three clusters are so different that it is difficult to ascribe them to the derivation used; the structures are more indicative of differences in the timing and evolution of large-scale features in the model forecasts (David Stensrud, 2000, personal communication).

The similarity-based PCA is applied for the MSLP data set based on the correlation matrix. The variances that explained by the first and second eigenvalues for the initial and finishing times are shown in Table 4-7. As the table shows slight divergences are occurring between members as the time passes. At the finishing time, a strong simple structure is obtained by rotating the first 3 PCs using promax, Table 4-8a. In fact, retaining 3 PCs lead to extracting 92.9% of the total variance. By examining the simple structure in Table 4-8a, the members are grouped into 3 clusters corresponding

to the 3 institutions that collectively construct the ensemble ([ARPS], [Eta, RSM], [MM5]). The correlation matrix of the rotated 3 PCs at the 36th hour, Table 4-8b, shows that the 3 clusters are moderately separated. The overall results of the PCA analysis are very consistent with the CA analyses of this field.

Table 4-7: Variances explained by the 1st and 2nd eigenvalues for the initial and finishing times for MSLP data.

	var(λ_1)	var(λ_2)	var(λ_1)+var(λ_2)
Initial Time	88.39	7.91	96.30
Finishing Time	67.14	16.26	83.40

Table 4-8a: Rotated loadings of the first 3 PCs using promax for the MSLP data at the 36th hour based on the correlation.

n	PC 1	PC 2	PC 3
1	0.10	0.05	**0.92**
2	-0.03	0.23	**0.84**
3	-0.04	0.19	**0.90**
4	0.14	-0.15	**0.94**
5	0.29	-0.15	**0.84**
6	**0.98**	0.02	-0.02
7	**0.99**	-0.06	0.04
8	**0.78**	0.16	0.11
9	**0.72**	0.10	0.23
10	**0.81**	0.13	0.13
11	**0.81**	0.10	0.12
12	**1.04**	-0.07	-0.08
13	**0.98**	-0.04	-0.01
14	**1.07**	-0.04	-0.16
15	**1.04**	-0.14	0.01
16	-0.01	**0.98**	0.00
17	0.04	**0.96**	-0.04
18	-0.08	**0.99**	0.02
19	-0.31	**1.08**	0.08
20	0.03	**0.91**	0.10
21	0.28	**0.88**	-0.22
22	-0.13	**0.96**	0.12
23	0.32	**0.82**	-0.15
24	0.20	**0.90**	-0.16
25	0.00	**0.91**	0.12

Table 4-8b: the correlation matrix of the 4 rotated PCs (promax) for the pressure data set at the 36th hour.

	1	2	3
1	1.00	0.56	0.54
2	0.56	1.00	0.46
3	0.54	0.46	1.00

4.4.2 CA and PCA Based On the Euclidean Similarity (EucSimi)

For each output time, the data matrix is first centered (2.3) and then the Euclidean similarity (EucSimi) (2.16) matrix, $\hat{E}_{25\times25}$, is computed. Figure 4-9 shows the clustering dendrograms resulted from applying UPGMA method (section 2.2.1) based on the EucSimi for the MSLP data. As the figure shows, the members very clearly form 3 clusters ([ARPS], [Eta, RSM], and [MM5]). So the results are consistent with their correlation-based counterparts. The tendency of the members of one model to form one cluster is apparent in these results.

The dissimilarity-based PCA is, then, applied using the Euclidean similarity matrix of the finishing time. This analysis leads to the same result that obtained by the previous one. Rotating the first 3 PCs using quartimax (section 2.3.2 A) produces a strong simple structure, and rotating the same 3 PCs using promax (not shown) improves the quality of the simple structure but does not change the clustering solution. The 3 clusters are well separated as indicated by the correlation matrix of the 3 rotated PCs (not shown) using promax. By examining the simple structure, Table 4-9, the clustering structure of the 25 members at the 36th hour is as follow: PC1 clusters 16-25; PC2 clusters 6-15; and PC3 clusters 1-5. This, in fact, is exactly the same resultant

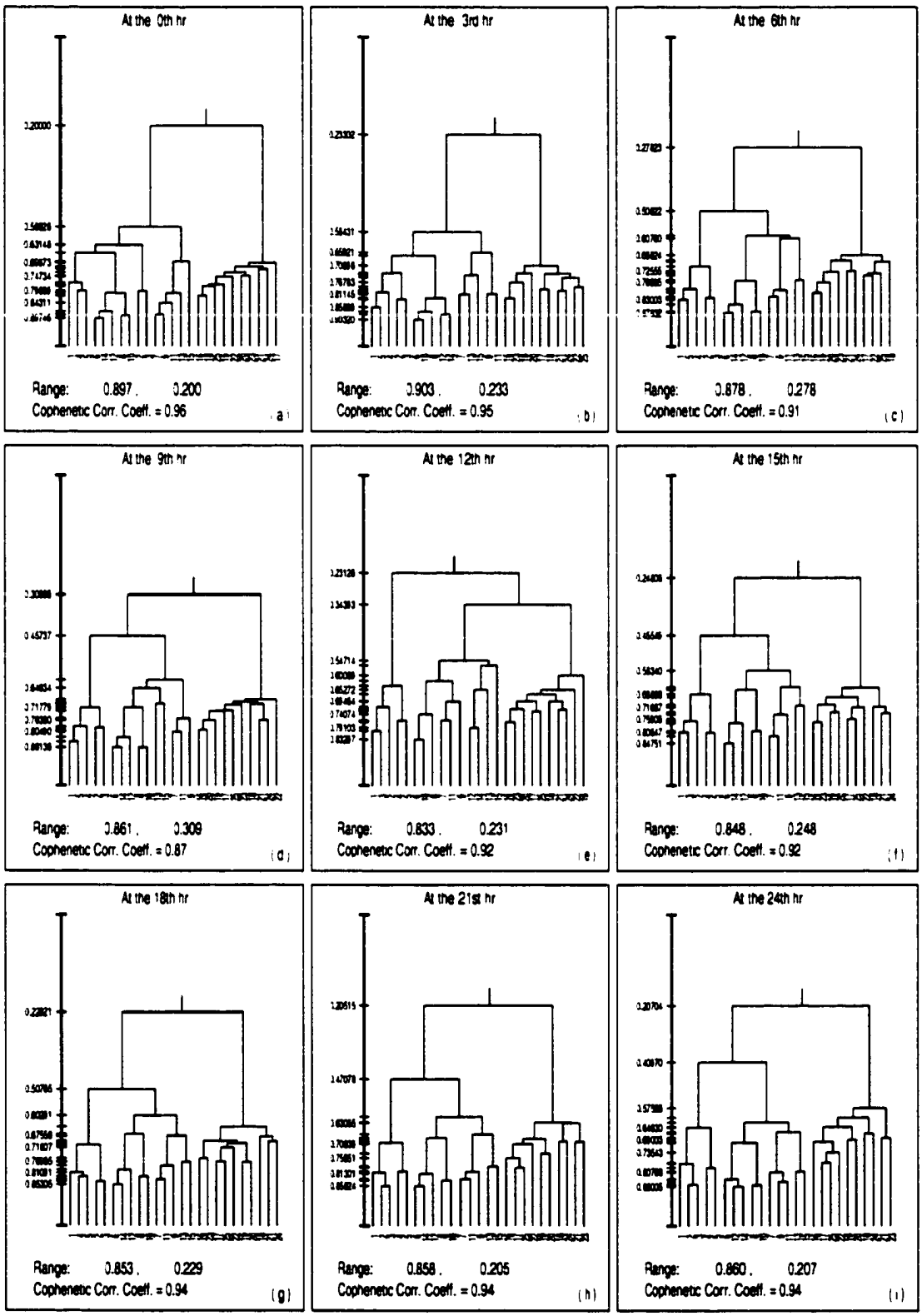


Figure 4-9: UPGMA dendrograms for the Pressure field based on the Euclidean Similarity; data is centered using (2.3).

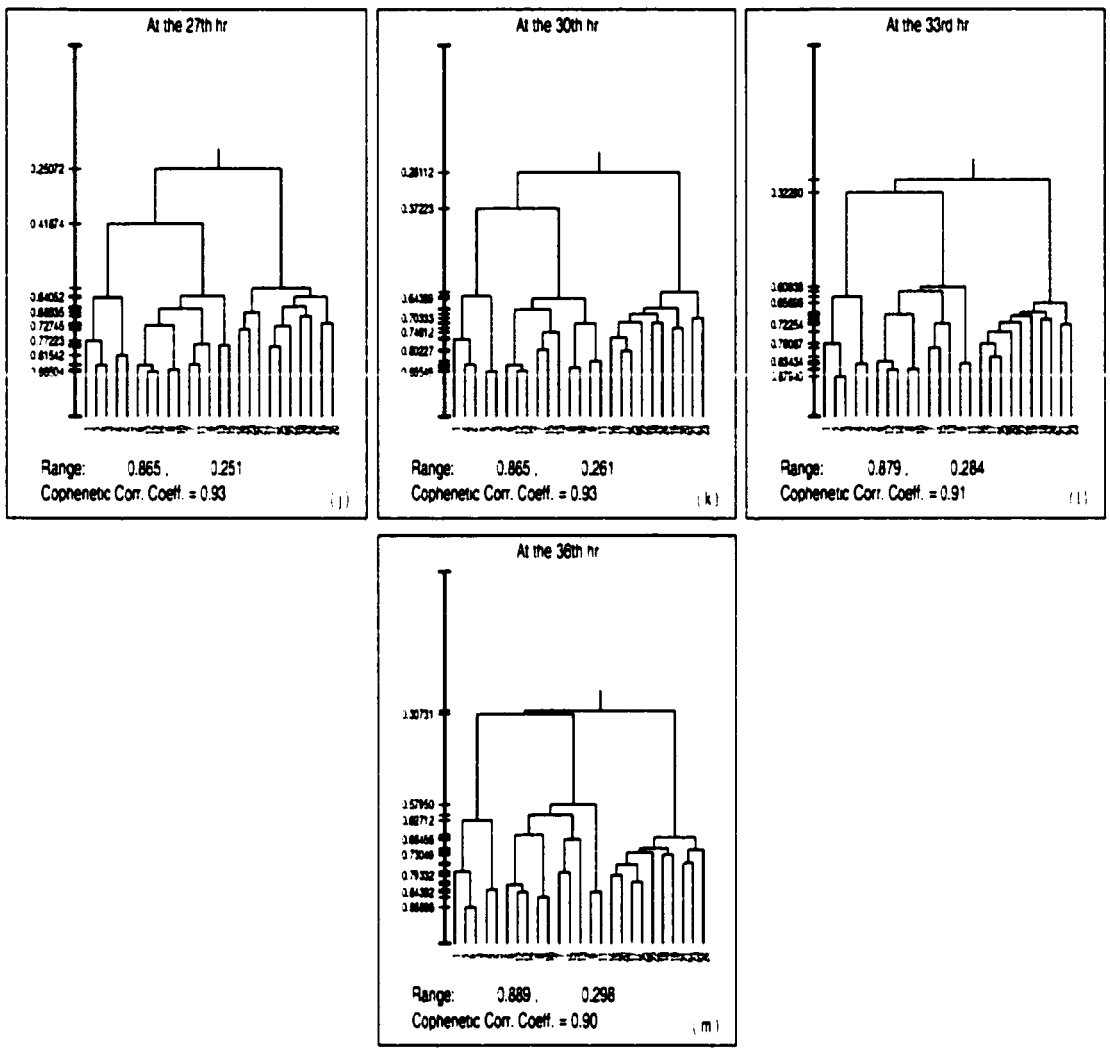


Figure 4-9: Continued

clustering structure from all analyses of this field. There is an agreement among the various clustering method to group the members of the ensemble for the MSLP field into 3 clusters corresponding to the 3 institutions that collectively construct the ensemble (i.e., [ARPS], [Eta, RSM], [MM5]).

Table 4-9: Rotated loadings of the first 3 PCs using quartimax for the MSLP data at the 36th hour based on the Euclidean Similarity.

n	PC 1	PC 2	PC 3
1	0.26	0.22	**0.86**
2	0.34	0.14	**0.77**
3	0.30	0.12	**0.83**
4	0.11	0.19	**0.82**
5	0.16	0.30	**0.78**
6	0.21	**0.86**	0.09
7	0.17	**0.83**	0.16
8	0.36	**0.70**	0.24
9	0.34	**0.64**	0.34
10	0.32	**0.75**	0.23
11	0.31	**0.70**	0.25
12	0.09	**0.86**	0.00
13	0.03	**0.78**	0.00
14	0.12	**0.86**	-0.03
15	0.09	**0.84**	0.10
16	**0.89**	0.13	0.10
17	**0.86**	0.15	0.09
18	**0.85**	0.06	0.08
19	**0.84**	-0.05	0.09
20	**0.86**	0.16	0.16
21	**0.81**	0.25	-0.01
22	**0.82**	0.09	0.18
23	**0.79**	0.31	0.05
24	**0.82**	0.22	0.02
25	**0.85**	0.14	0.17

4.4.3 CA Based On the Euclidean distance

Figure 4-10 shows the clustering dendrograms resulted from applying Ward's method (section 2.2.1) on the MSLP data set based on the Euclidean distance (2.10) matrices, E_{25x25}. The data matrices are first centered (2.3).

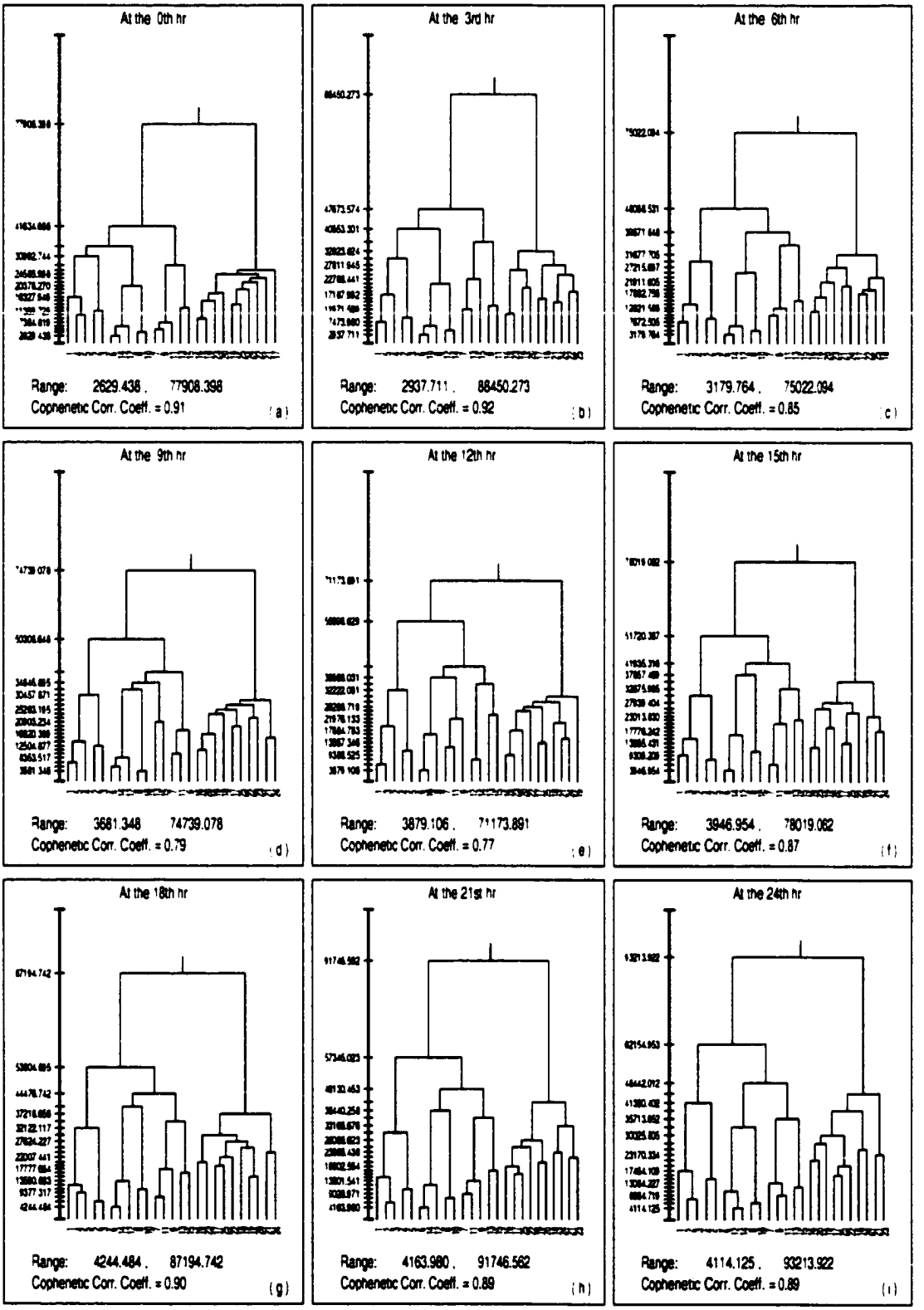


Figure 4-10: Ward dendrograms for the Pressure field based on the Euclidean distance; data is centered using (2.3).

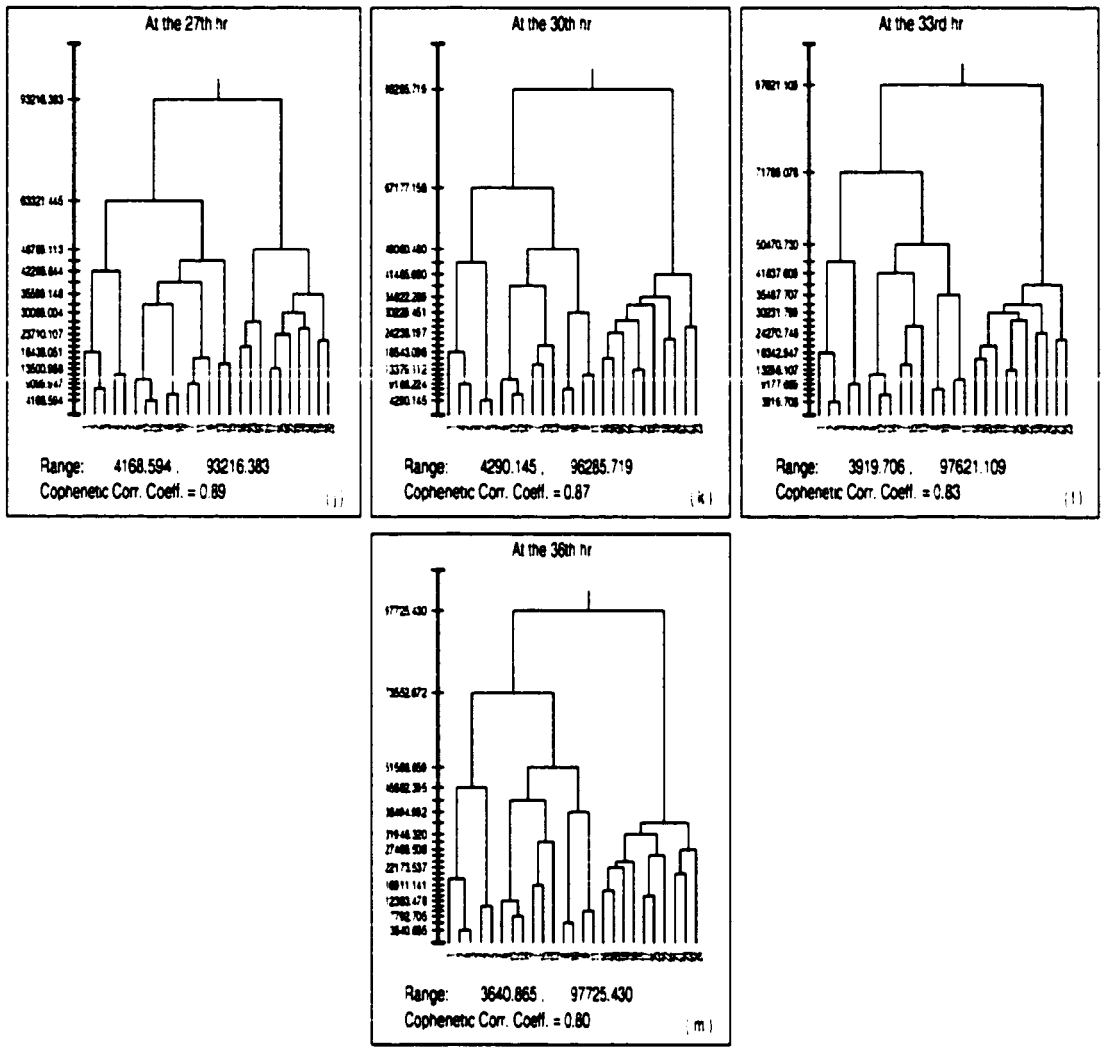


Figure 4-10: Continued

The clustering structures recovered here are exactly the same as those resulted from the correlation-based CA with respect to the 3 clusters pattern – ([ARPS], [Eta, RSM], [MM5]). It is not easy task, due to the nature of the Euclidean distance, to assess the degree of compactness and isolation by merely looking at the height of the trees. However, because all the 13 trees are drawn with the same vertical scale using the method we developed in chapter 3 we can then use the levels of mergence to intuitively

evaluate the degree of similarity among the various clusters. Such an analysis supports our earlier observation that the 3 clusters are compact and isolated from each other.

4.4.4 CA Based On the *K*mean and Nucleated Agglomerative (NA) algorithms

For each output time of the MSLP data set, *K*mean algorithm (section 2.2.2 B) is applied. The number of clusters is chosen based on the previous results to be three, k=3, and the seed points are specified subjectively by choosing one seed from ARPS model, one from [Eta, RSM], and one from [MM5]. The clustering structures resulted from applying *K*mean are shown in Table 4-10. As the table shows, these partitions are consistent with the Ward's results. They show the strong tendency of the ensemble members to build their clusters according to their models.

Table 4-10: *K*mean's partitions for the pressure data set.

T	Clusters		
0	[1, 4, 5, 6,8,10,12,14]	[2, 3, 7,9,11,13,15]	[16,17,18,19,20,21,22,23,24,25]
3	[1, 2, 3, 4, 5, 6,8,10,12,14]	[7, 9, 11, 13, 15]	[16,17,18,19,20,21,22,23,24,25]
6	[1, 2, 3, 4, 5]	[6, 8, 10, 12, 14]	[7, 9,11,13,15, 16,17,18,19,20,21,22,23,24,25]
9	[1, 2, 3, 4, 5]	[6, 8, 10, 12, 13, 14]	[7, 9,11,15, 16,17,18,19,20,21,22,23,24,25]
12	[1, 2, 3, 4, 5]	[6, 10, 12, 13, 14]	[7, 8, 9,11,15, 16,17,18,19,20,21,22,23,24,25]
15	[1, 2, 3, 4, 5, 12]	[6, 7, 8, 10, 11, 13, 14, 15]	[9, 16,17,18,19,20,21,22,23,24,25]
18	[1, 2, 3, 4, 5, 12]	[6, 7, 8, 9, 10, 11, 13, 14, 15]	[16,17,18,19,20,21,22,23,24,25]
21	[1, 2, 3, 4, 5]	[6, 7, 8,10,11,12,13,14,15]	[9, 16,17,18,19,20,21,22,23,24,25]
24	[1, 2, 3, 4, 5]	[6, 7, 8, 9,10,11,12,13,14,15]	[16,17,18,19,20,21,22,23,24,25]
27	[1, 2, 3, 4, 5]	[6, 7, 8, 9,10,11,12,13,14,15]	[16,17,18,19,20,21,22,23,24,25]
30	[1, 2, 3, 4, 5]	[6, 7, 8, 9,10,11,12,13,14,15]	[16,17,18,19,20,21,22,23,24,25]
33	[1, 2, 3, 4, 5]	[6, 7, 8, 9,10,11,12,13,14,15]	[16,17,18,19,20,21,22,23,24,25]
36	[1, 2, 3, 4, 5]	[6, 7, 8, 9,10,11,12,13,14,15]	[16,17,18,19,20,21,22,23,24,25]

The NA algorithm (section 2.2.2 B) is then applied in the same manner for the same data set. The range of the number of clusters is chosen to be [4,2], k_{max}=4, k_{min}=2; and the seed points are specified subjectively by chosen one from each model. Table 4-11, shows the resultant partitions with respects to 3 clusters. As the table shows, these partitions are very similar to those of Ward and *K*mean's results. In fact, the clustering solution obtained for the finishing time is exactly the same throughout all different methods we use in this section. This gives more confidence in the recovered solution.

Table 4-11: NA's partitions for the MSLP data set.

T	Clusters		
0	[1, 4, 5, 6, 8,10,12,14]	[2, 3, 7, 9,11,13,15]	[16,17,18,19,20,21,22,23,24,25]
3	[1, 2, 3, 4, 5, 6, 8,10,12,14]	[7, 9,11,13,15]	[16,17,18,19,20,21,22,23,24,25]
6	[1, 2, 3, 4, 5, 6, 8,10,12,14]	[7, 9,11,13,15]	[16,17,18,19,20,21,22,23,24,25]
9	[1, 2, 3, 4, 5, 6,10,12,14]	[7, 8, 9,11,13,15]	[16,17,18,19,20,21,22,23,24,25]
12	[1, 2, 3, 4, 5]	[6,10,12,13,14]	[7, 8, 9,11,15, 16,17,18,19,20,21,22,23,24,25]
15	[1, 2, 3, 4, 5, 6,10,12,13,14]	[7, 8, 9,11,15]	[16,17,18,19,20,21,22,23,24,25]
18	[1, 2, 3, 4, 5]	[6, 7, 8, 9,10,11,12,13,14,15]	[16,17,18,19,20,21,22,23,24,25]
21	[1, 2, 3, 4, 5]	[6, 7, 8, 9,10,11,12,13,14,15]	[16,17,18,19,20,21,22,23,24,25]
24	[1, 2, 3, 4, 5, 6, 8,10,12,14]	[7, 9,11,13,15]	[16,17,18,19,20,21,22,23,24,25]
27	[1, 2, 3, 4, 5, 6, 8,10,12,14]	[7, 9,11,13,15]	[16,17,18,19,20,21,22,23,24,25]
30	[1, 2, 3, 4, 5]	[6, 7, 8, 9,10,11,12,13,14,15]	[16,17,18,19,20,21,22,23,24,25]
33	[1, 2, 3, 4, 5]	[6, 7, 8, 9,10,11,12,13,14,15]	[16,17,18,19,20,21,22,23,24,25]
36	[1, 2, 3, 4, 5]	[6, 7, 8, 9,10,11,12,13,14,15]	[16,17,18,19,20,21,22,23,24,25]

The analyses of the pressure data set demonstrate the strong tendency of the ensemble members to cluster according to their models first before joining others. It should be noticed that this behavior of the ensemble members for this field is likely due

to algorithms used to calculate the mean sea level pressure, and not so much the models themselves (David Stensrud, 2000, personal communication).

4.5 Analysis of Other Fields

This section summarizes the results of analyzing several other fields of interest for meteorologists. In sections 4.3 and 4.4, we analyzed the ACCPPT and MSLP fields. In this section we analyzed all other fields listed in Table 4-1.

4.5.1 CA Based On the Correlation and Euclidean distance

For each field, 13 output times are analyzed starting from the initial time (0^{th} hour) until the finishing time (36^{th} hour) with 3 hours apart. A similarity-based CA is conducted first, and then CA is applied based on a dissimilarity measure.

The CA based on the similarity coefficient is conducted using the correlation matrices (2.13) of the 13 output times as the similarity matrices and then applying UPGMA method (section 2.2.1) on these matrices. The dissimilarity-based CA, on the other hand, is done in the same manner but by using the Euclidean distance (2.10) as the dissimilarity measure and Ward's (section 2.2.1) as the clustering method. In the latter case the data matrices are centered (difference fields) (2.3) before computing the Euclidean distance matrices. The resultant 13 clustering dendrograms for each field are studied to determine the hierarchical clustering structures of the 25 members of the ensemble at each output time. The results of these analyses are summarized in the following subsections.

A. CAPE Field

Figure 4-11a shows the clustering dendrograms resulting from applying UPGMA method on the CAPE data set based on the correlation matrices. Starting from the 3rd hour and until the 33rd hour, the members of the ensemble for this data set follow the 4-model clustering structure – ([ARPS], [Eta], [RSM], [MM5]). That's members of each model form their own cluster before merging with others. In this resultant structure, Eta model cluster is the most compact cluster; while the MM5 model cluster is the least compact one through these hours. At the finishing time, the clustering structure is changed. Members from ARPS and Eta form their own clusters at high degree of correlation, which emphasizes the closeness of the members of these two models. The MM5 members break into 2 groups. One group containing members 16,17,23 and 25 formed separate cluster before joining ARPS cluster. The other group remains separate from all the other clusters. On the other hand, two members of RSM model join the Eta cluster before building one large cluster containing the members of both Eta and RSM models.

Examination of the CAPE fields from the individual model runs again indicates that the clustering is not from using different methods to calculate CAPE, although the methods used are different. The CAPE fields indicate obvious differences in the large-scale structure of the fields that are impossible to explain from different calculation methods. The more northern placement of the low center in the Eta model and RSM forecasts at the 36th hour results in a more northerly extension of the CAPE field as compared with the other models, Figure 4-11b, (David Stensrud, 2000, personal communication).

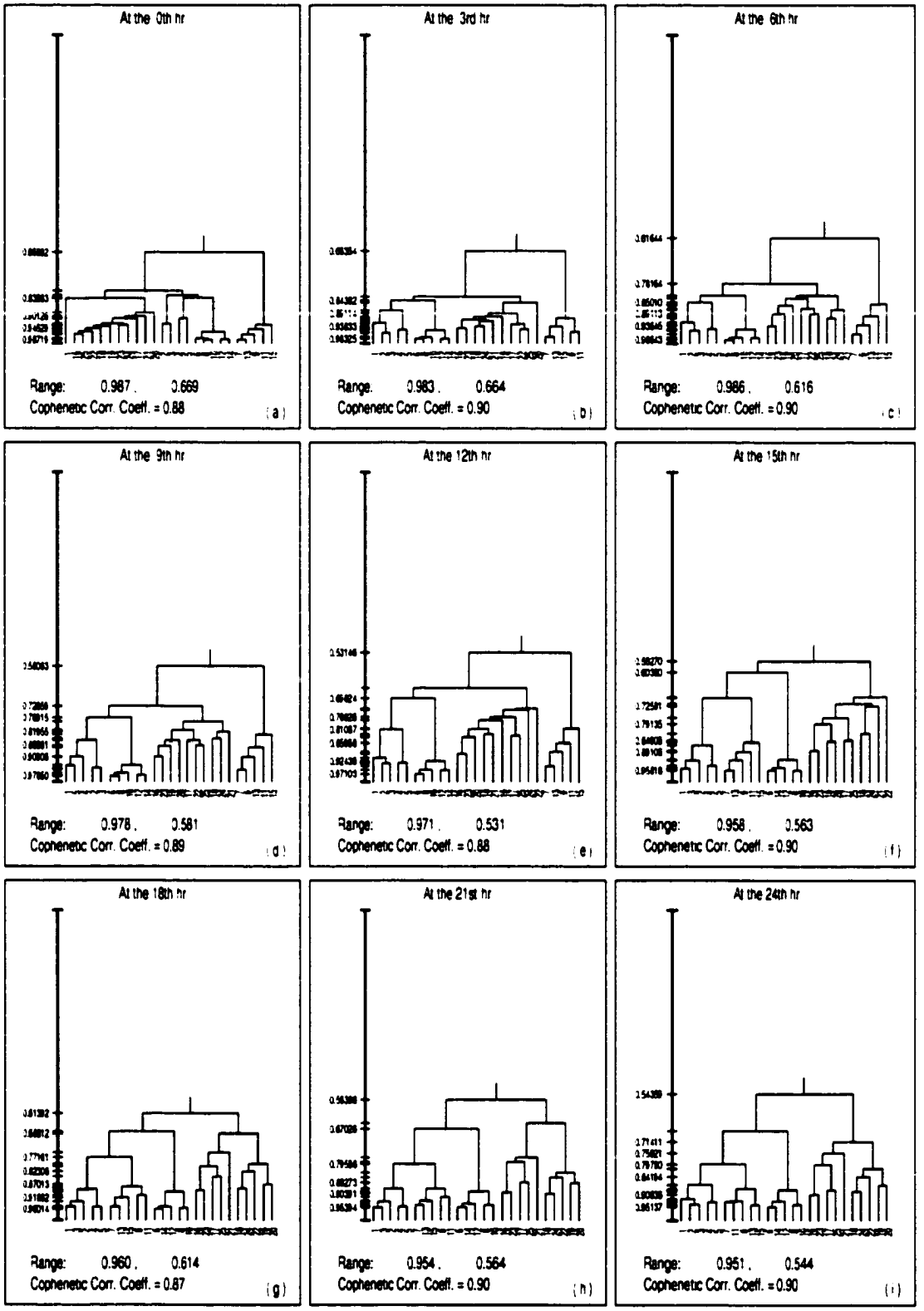


Figure 4-11a: UPGMA dendrograms for the CAPE field based on the Correlation.

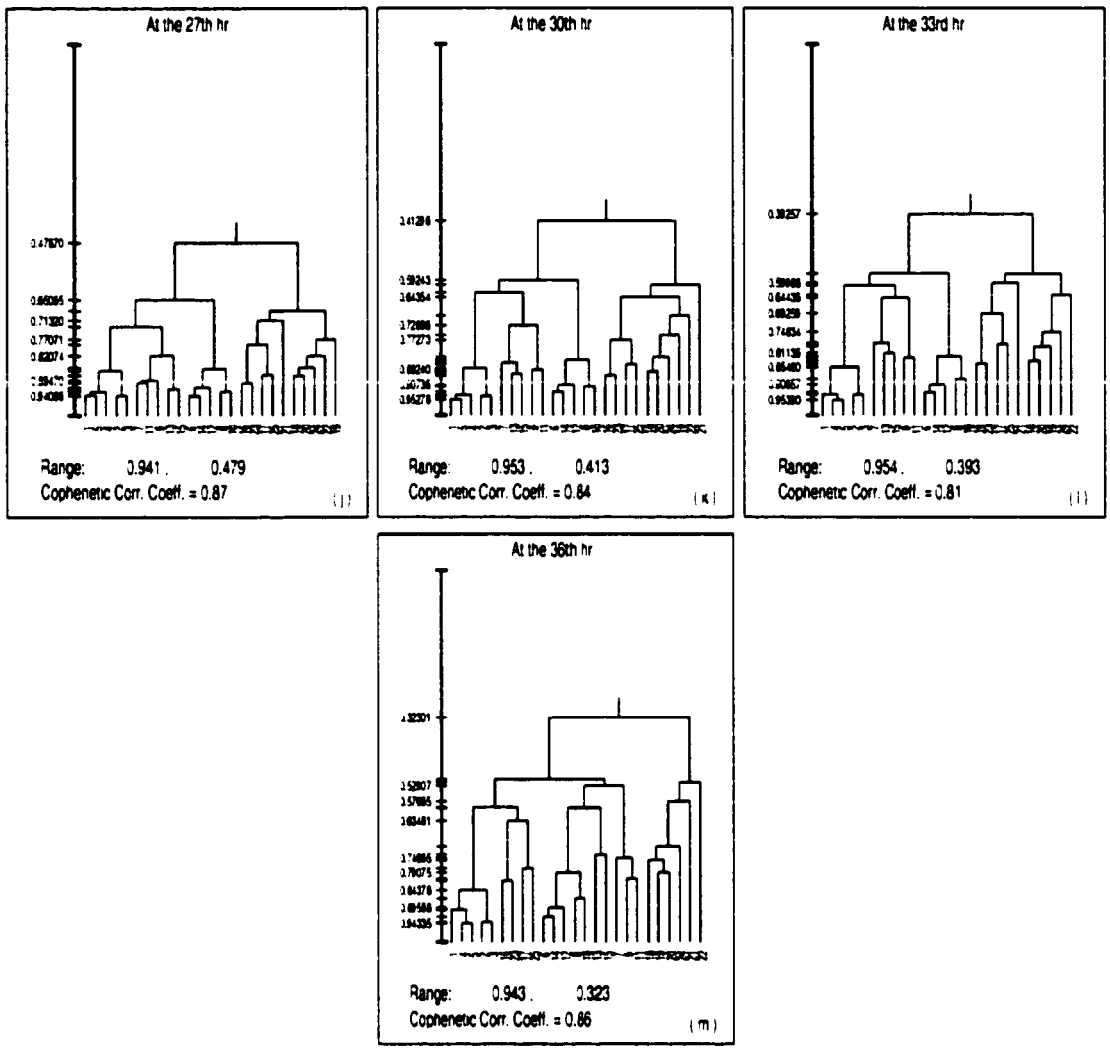


Figure 4-11a: Continued

The ARPS forecasts all have values of CAPE exceeding 1000 J kg^{-1} stretching from the Gulf of Mexico across eastern Texas and northward into Nebraska. In contrast, all the other models have a local minima of CAPE in eastern Texas, producing a vastly different pattern in the fields. Thus, the differences are mainly due to the use of the different models and not the model initialization or CAPE derivation procedures (David Stensrud, 2000, personal communication).

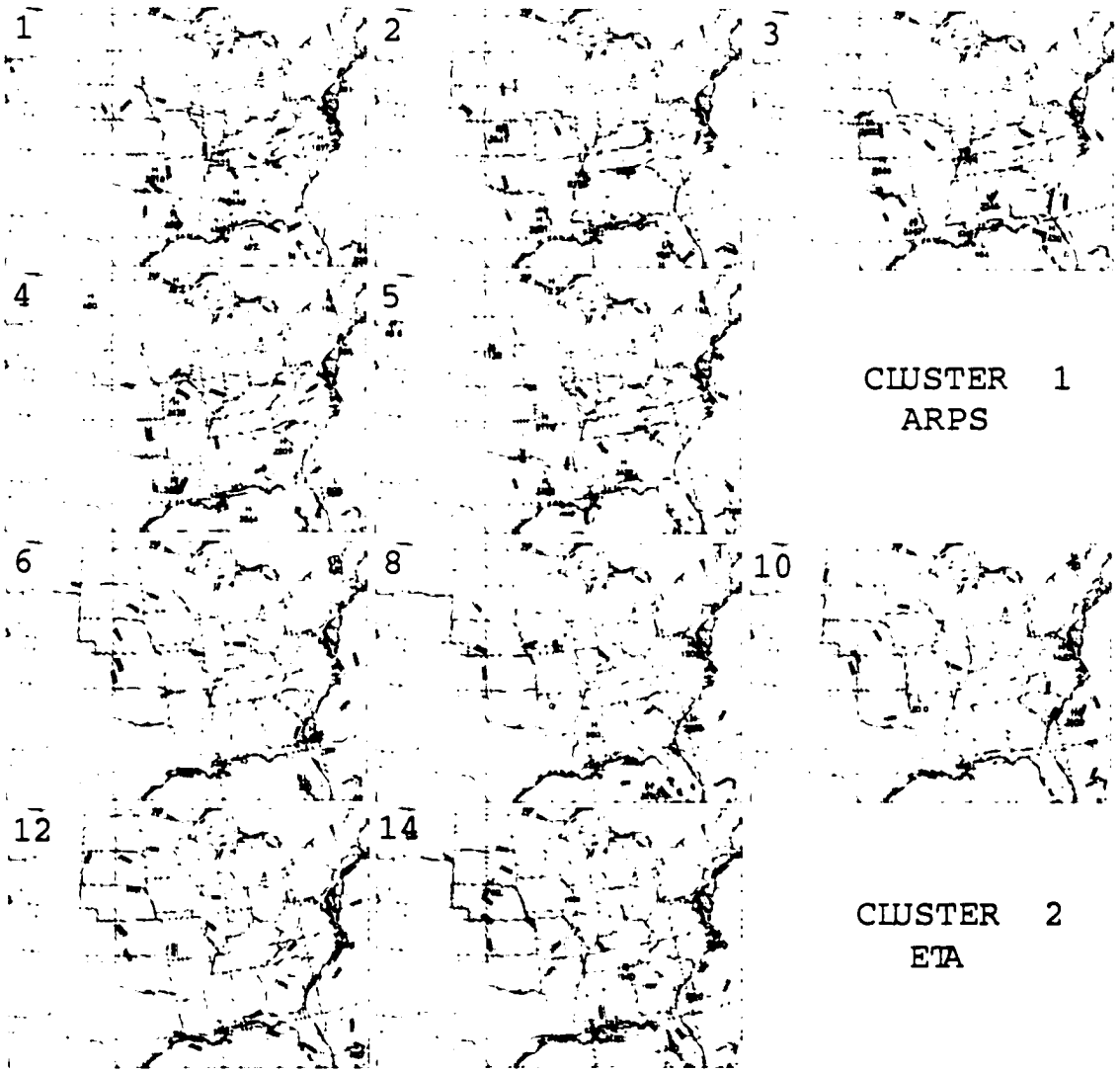


Figure 4-11b: CAPE valid at 36th hour from all 25 ensemble members grouped subjectively into 4 clusters. Numbers in the upper-left hand corner indicate the ensemble member as defined in Table 4-2. Isolines every 500 J kg-1

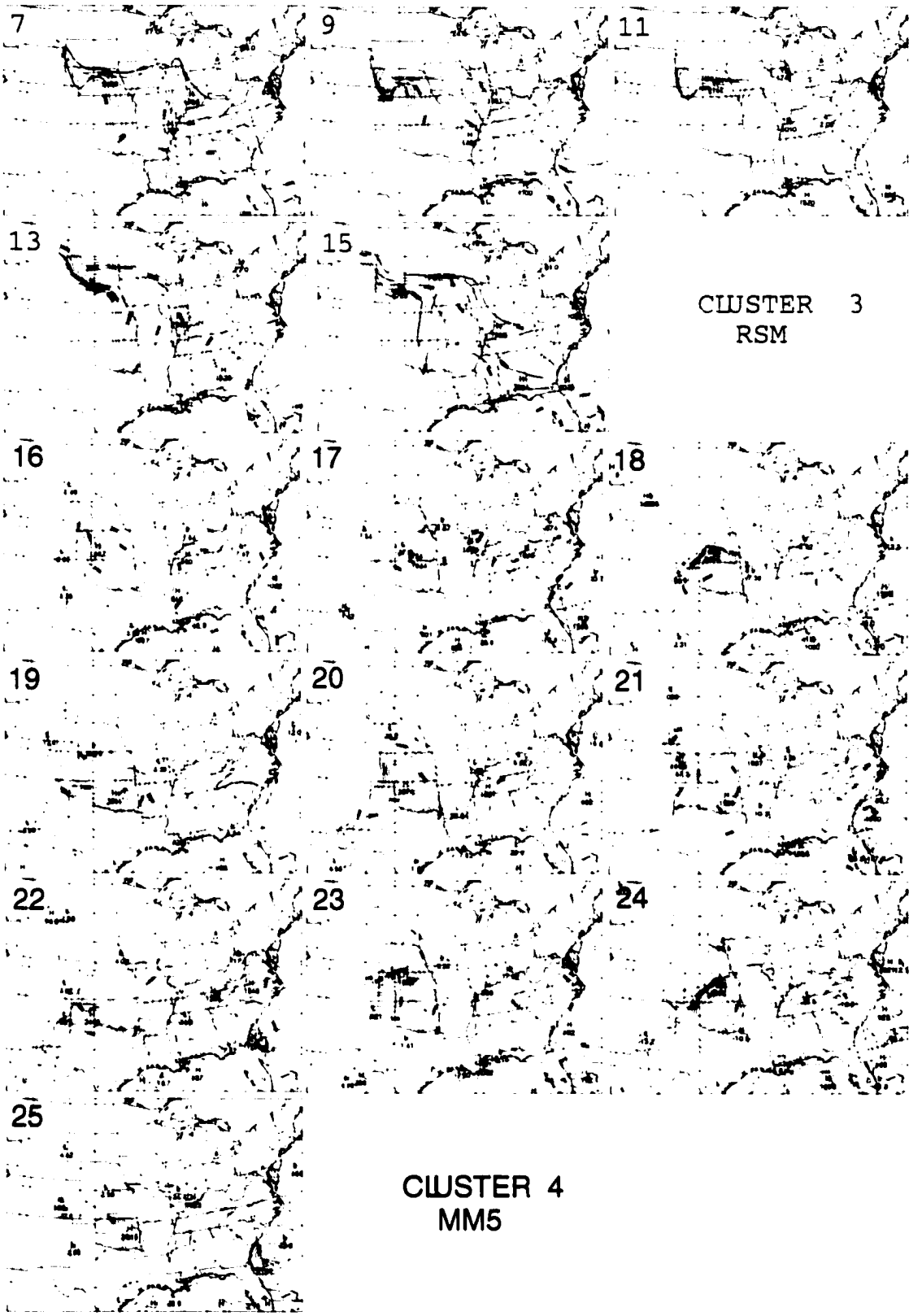


Figure 4-11b: Continued.

The clustering dendrograms resulting from applying Ward's method on the CAPE data set based on the Euclidean distance are shown in Figure 4-11c. At all times the members follow the 4-model clustering structure – ([ARPS], [Eta], [RSM], [MM5]). Hence, there is agreement between this result and the correlation-base result for the time period spanning from 3rd hour until the 33rd hour. The Eta cluster again is shown to be the most compact cluster, followed by the ARPS cluster.

Although the overall results of both analyses differ at the finishing time, no inter-model cluster is found using either clustering algorithm or measure.

B. 500-mb Height Field

Figure 4-12a shows the clustering dendrograms resulting from applying UPGMA method on the height data set based on the correlation matrices. In this analysis, CA fails to recover the clustering structures because members are extremely correlated. In essence, all the trees are so short that no useful information on the clusters can be obtained. By the finishing time, they are joined into one large cluster at 0.96 degree of correlation. However, even under this situation the members are seen to form 3 clusters – ([ARPS], [Eta, RSM], [MM5])- during the last 3 output times (i.e., 30th, 33rd, and 36th hours).

The clustering dendrograms resulting from applying Ward's method on the height data set based on the Euclidean distance are shown in Figure 4-12b. From the beginning until the 27th hour, the resultant clustering structures have dramatic changes with respect to the membership of each resulted clusters. During this period of time, members from ARPS and MM5 tend to form their own clusters at relatively early stage in the hierarchy before merging with members from other models.

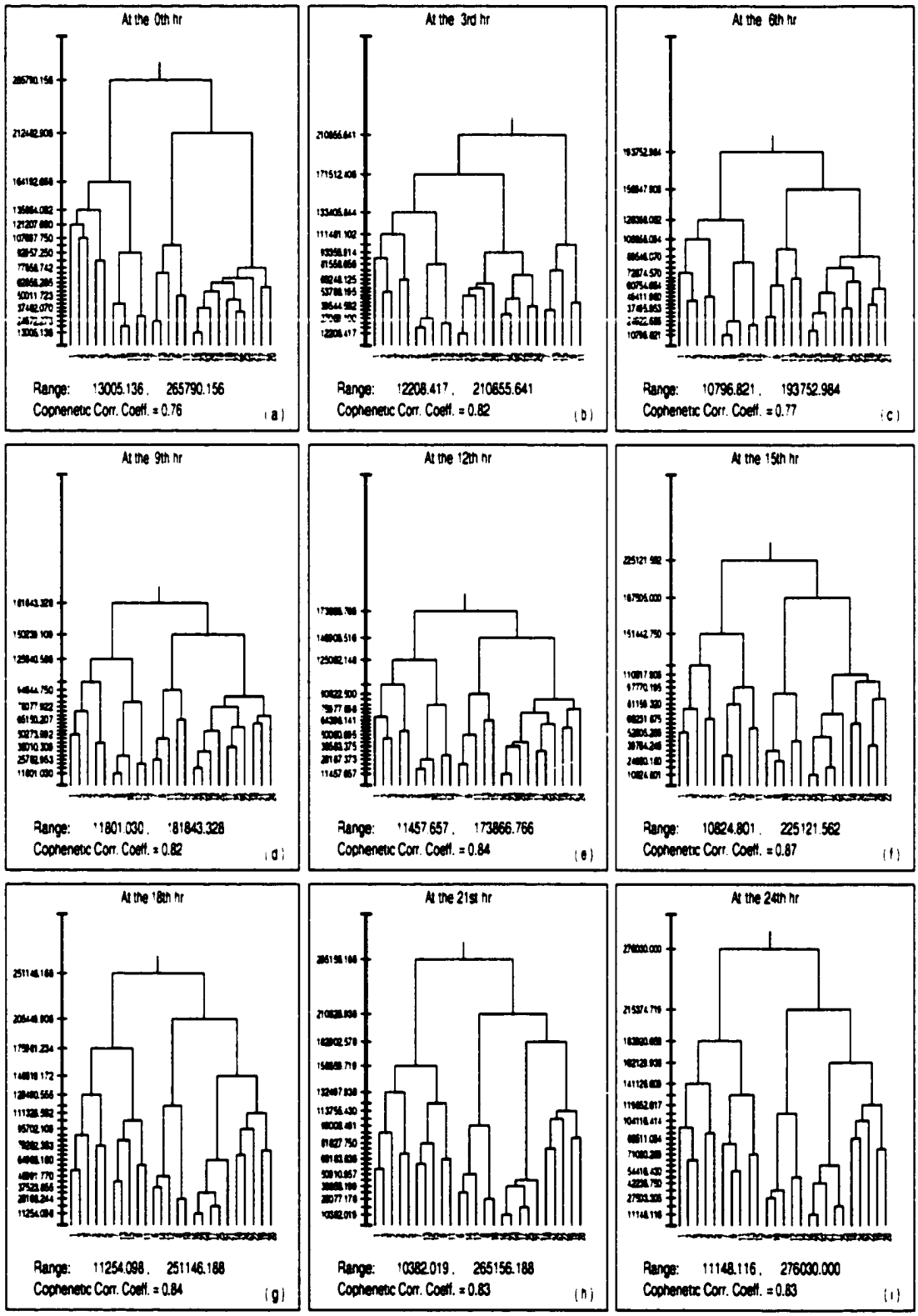


Figure 4-11c: Ward dendrograms for the CAPE field based on the Euclidean distance; data is centered using (2.3).

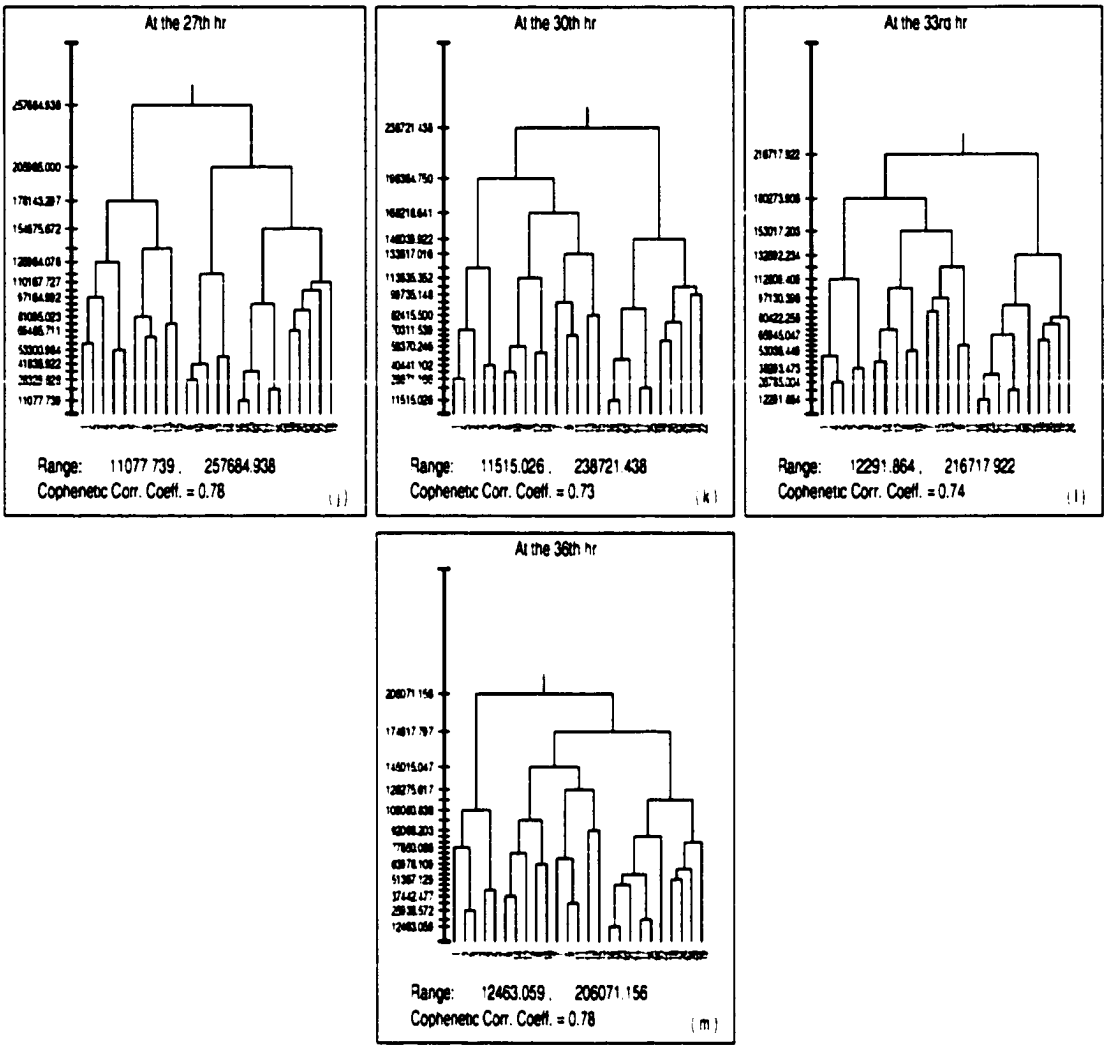


Figure 4-11c: Continued

At a few output times, Eta model members build their own cluster. Starting from the 30th hour onward, the members of the ensemble follow the 3-clusters pattern seen previously (i.e., [ARPS], [Eta, RSM], [MM5]). Inside the [Eta, RSM] cluster, the Eta cluster again is built first before the large cluster is formed; hence no inter-model cluster is found.

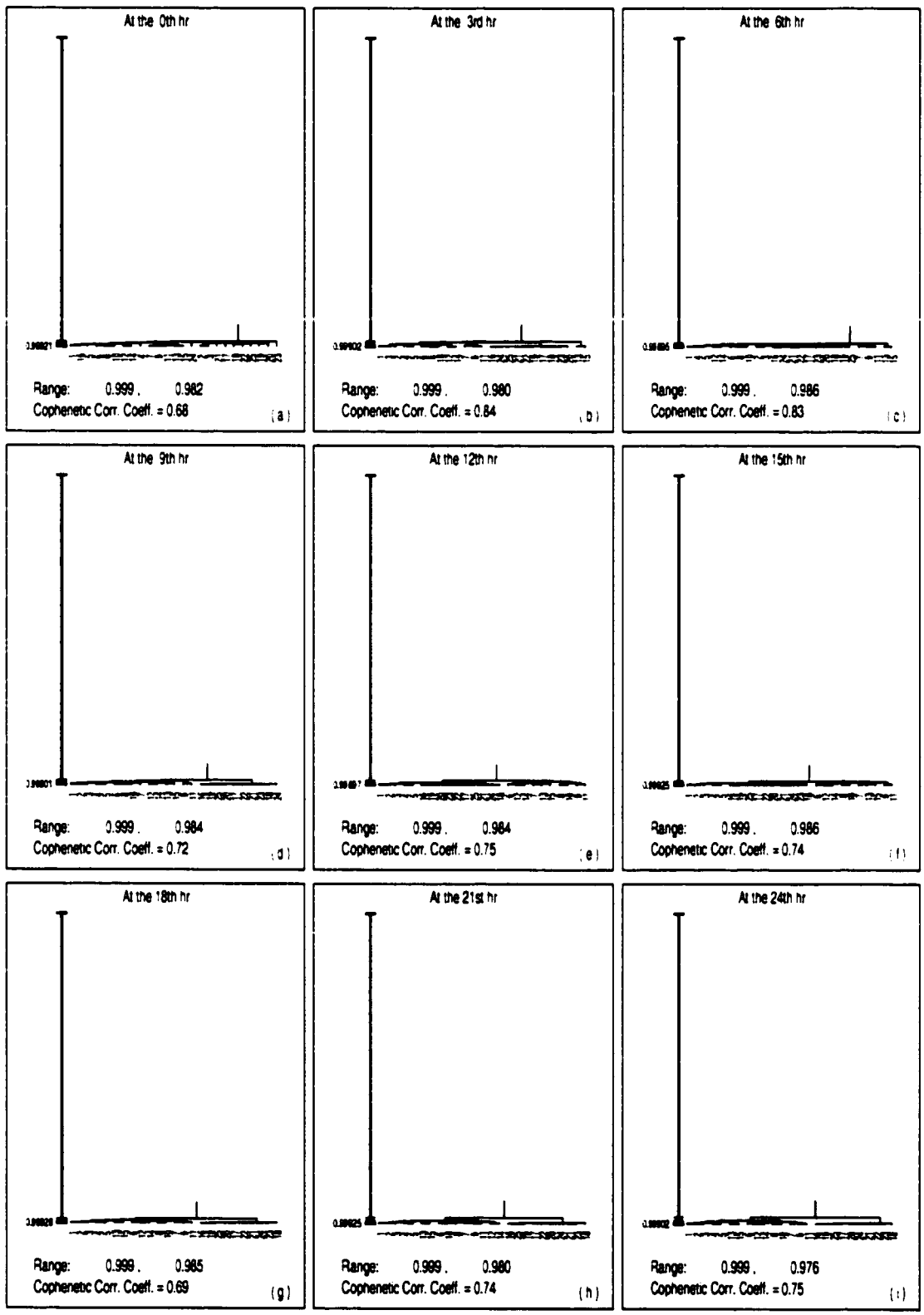


Figure 4-12a: UPGMA dendrograms for the Height 500mb field based on the Correlation.

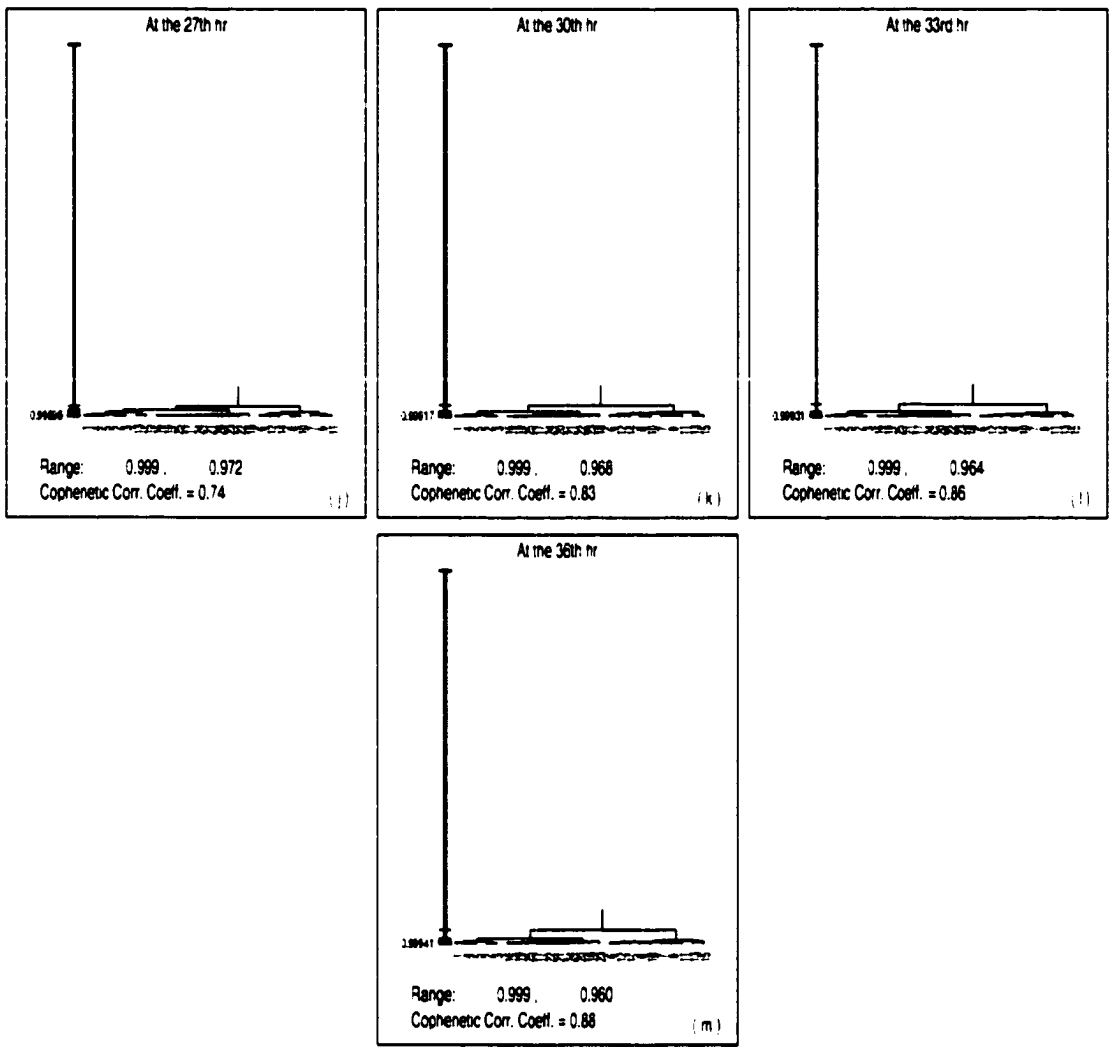


Figure 4-12a: Continued

According to the various levels at which the hierarchies are built (Figure 4-12b), the 3 clusters are reasonably isolated with the MM5 cluster being the most compact one for all output times. Although Stensrud et al. (2000) argue that geopotential height may not be the most useful measure for evaluating ensemble forecast output, the conclusions drawn from these data continue to support the notion that the differences in model physics are dominating the clustering of the ensemble members (David Stensrud, 2000, personal communication).

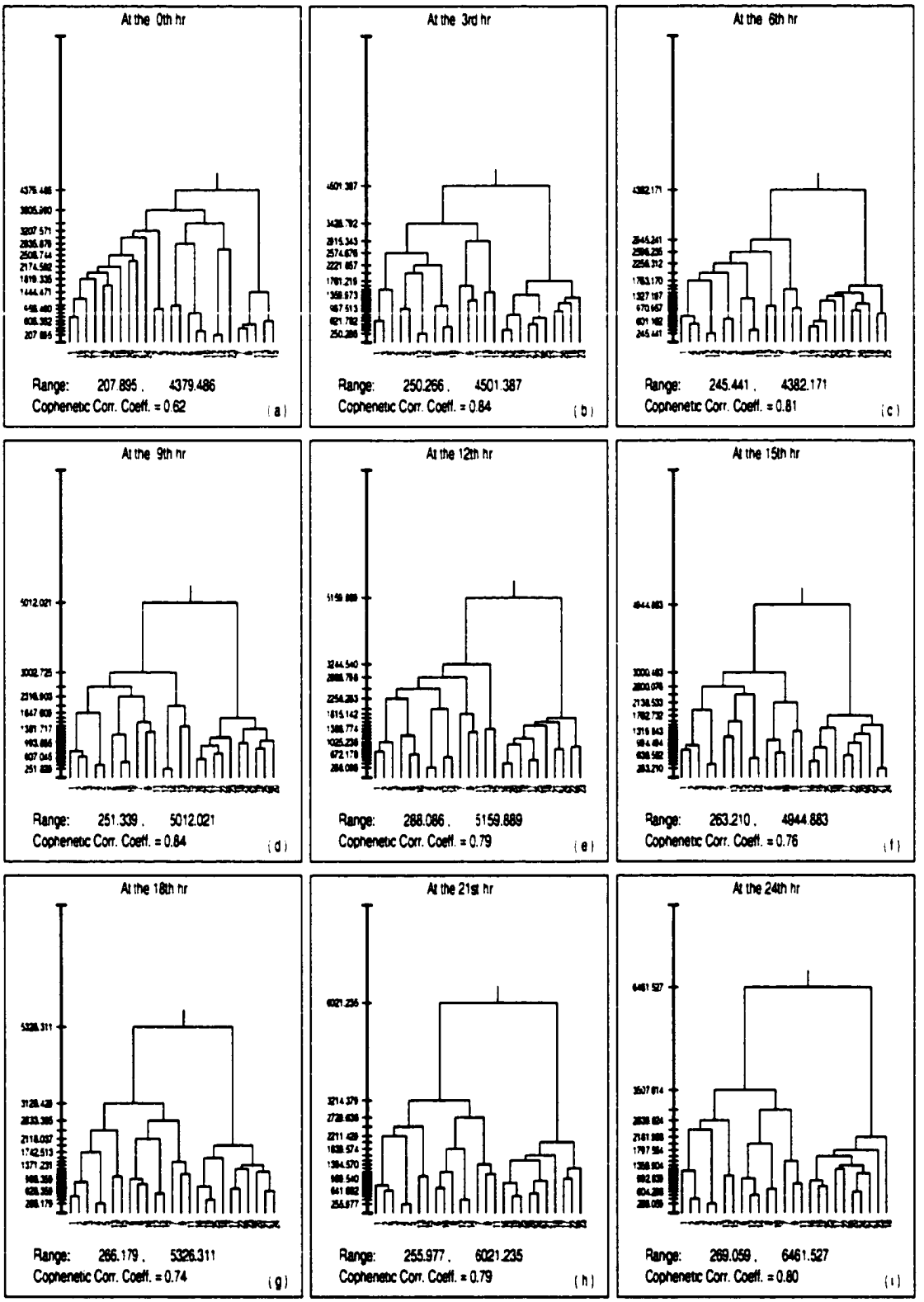


Figure 4-12b: Ward dendrograms for the Height 500mb field based on the Euclidean distance; data is centered using (2.3).

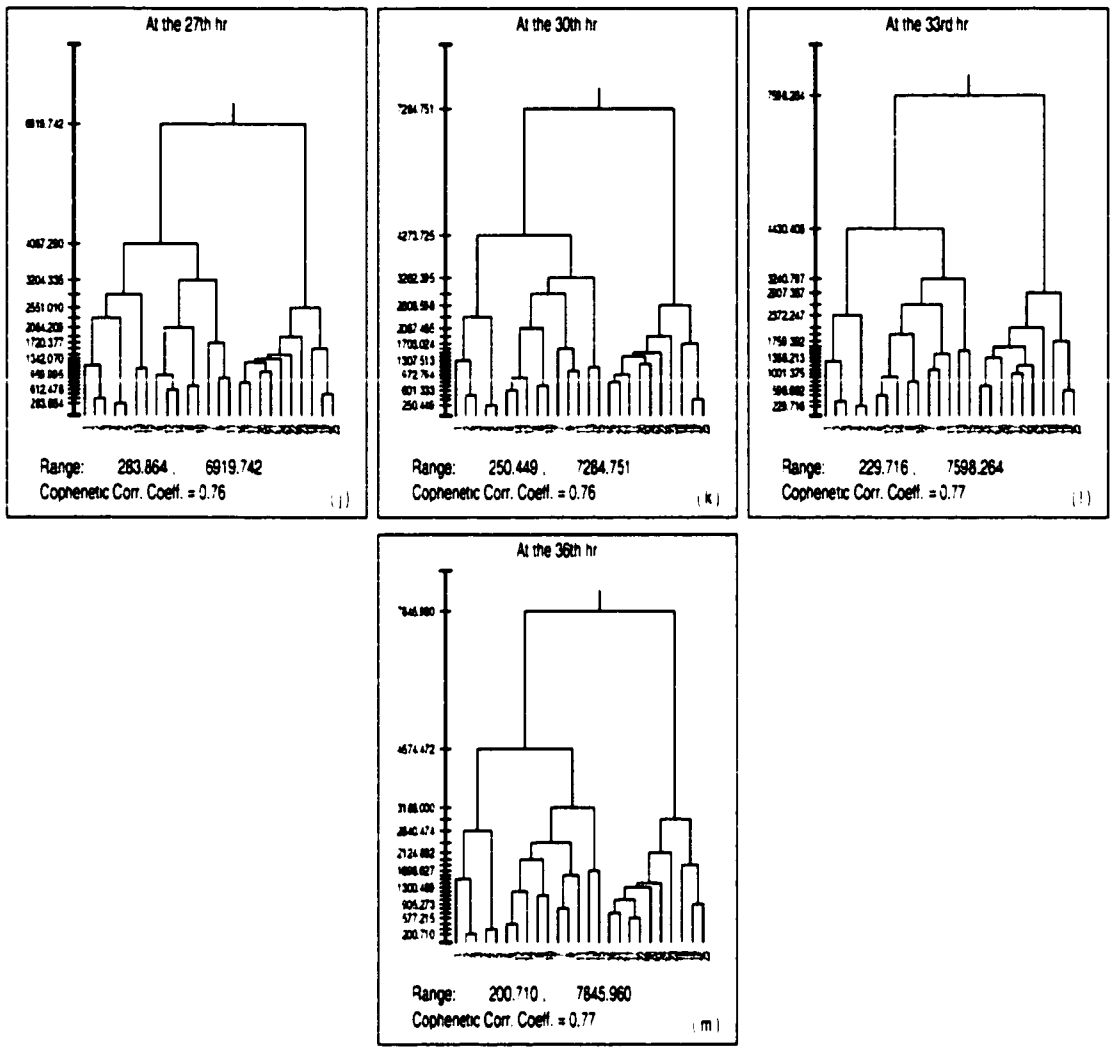


Figure 4-12b: Continued

C. 250-mb Wind Field

Figure 4-13a shows the clustering dendrograms resulting from applying UPGMA method on the wind data set based on the correlation matrices. This results support the conclusions found using the mean sea-level pressure data. Although there is more variation in the clusters over the forecast interval than found with the sea-level pressure data, the members of this ensemble are largely correlated. They are all placed in one large cluster with a correlation of 0.82 by the finishing time.

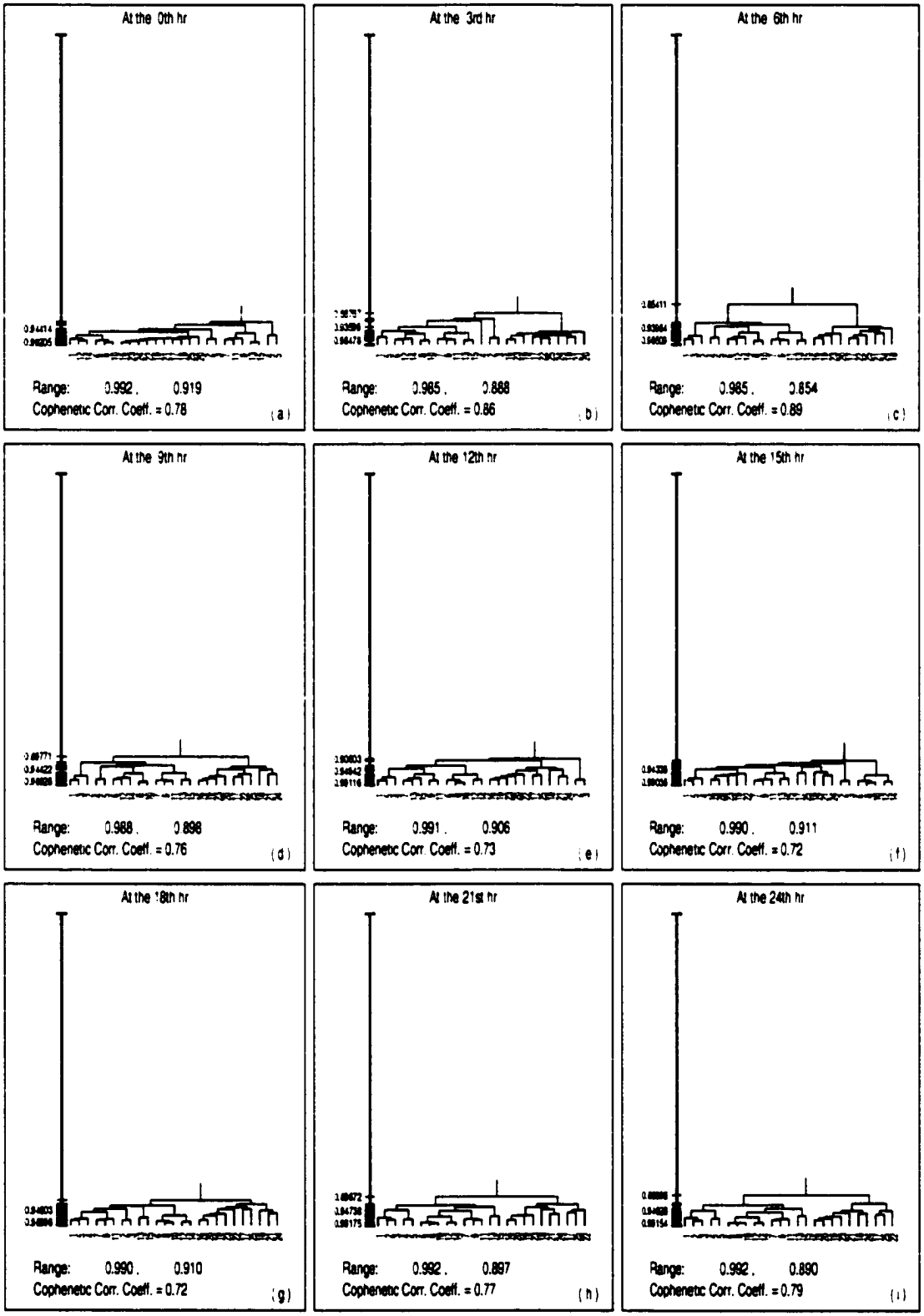


Figure 4-13a: UPGMA dendrograms for the Wind velocity 250mb field based on the Correlation.

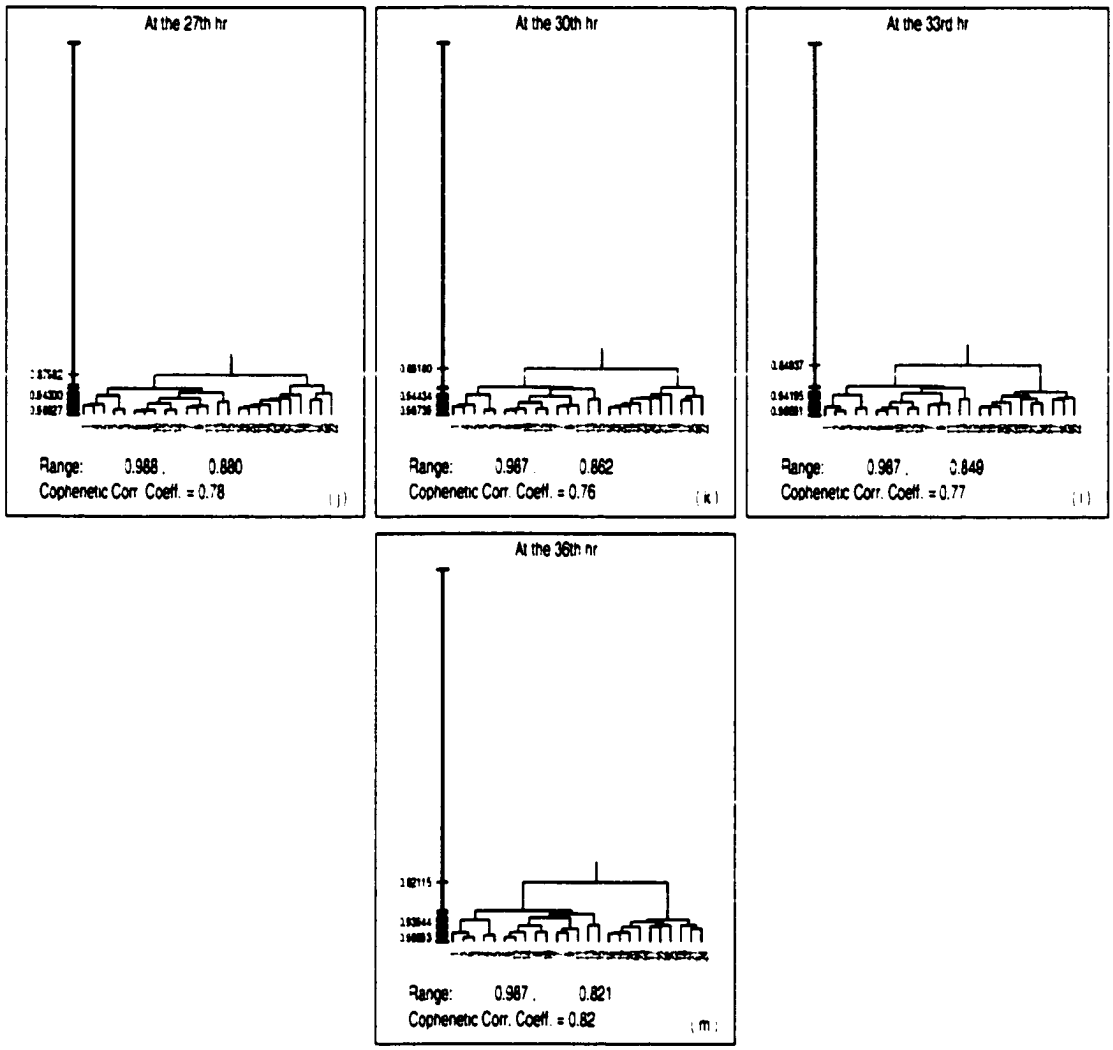


Figure 4-13a: Continued

Starting from the 0^{th} hour until the 18^{th} hour, the clustering structure changes from time to time. During this period of time, MM5 tends to build itself in many occasions. However, from the 21^{st} hour onward the members follow the same 3-clusters pattern seen using the MSLP data (i.e.,[ARPS], [Eta, RSM], [MM5]). Inside the [Eta, RSM] cluster, the Eta cluster is built first before building the large cluster; hence no inter-model cluster is found.

Examination of the 250-mb wind fields from the model forecasts again shows differences in the large-scale features predicted (not shown). The ridge over the Rocky Mountains in the Eta model and RSM is farther east than in the ARPS or MM5, and the winds are stronger in the Eta model and RSM than in the other two models. In contrast, the trough over the eastern United States is deeper in the ARPS and MM5 forecasts than in the Eta model and RSM forecasts, which have a more zonal flow pattern. Thus, the 3 clusters found using the CA are supported by a subjective analysis of these data and illustrate that the models are producing different evolutions of large-scale atmospheric features even on time-scales as short as 36 hours or less. It is also interesting to note that even with bred mode perturbations in the Eta model and RSM, the forecast 250-mb wind speeds from these two models are most alike; the model variability is exceeding the initial condition variability for this day (David Stensrud, 2000, personal communication).

The clustering dendrograms resulting from applying Ward's method on the wind speed data, based on the Euclidean distance (Figure 4-13b), show that the members follow a 4-model clustering structure (i.e.,[ARPS], [Eta], [RSM], [MM5]). At the 36^{th} hour, members form 3 clusters of [ARPS], [Eta, RSM], and [MM5]. Inside the [Eta, RSM] cluster, the Eta cluster again is built first before building the large cluster; hence no inter-model cluster is found, supporting the conclusion found from the correlation measure.

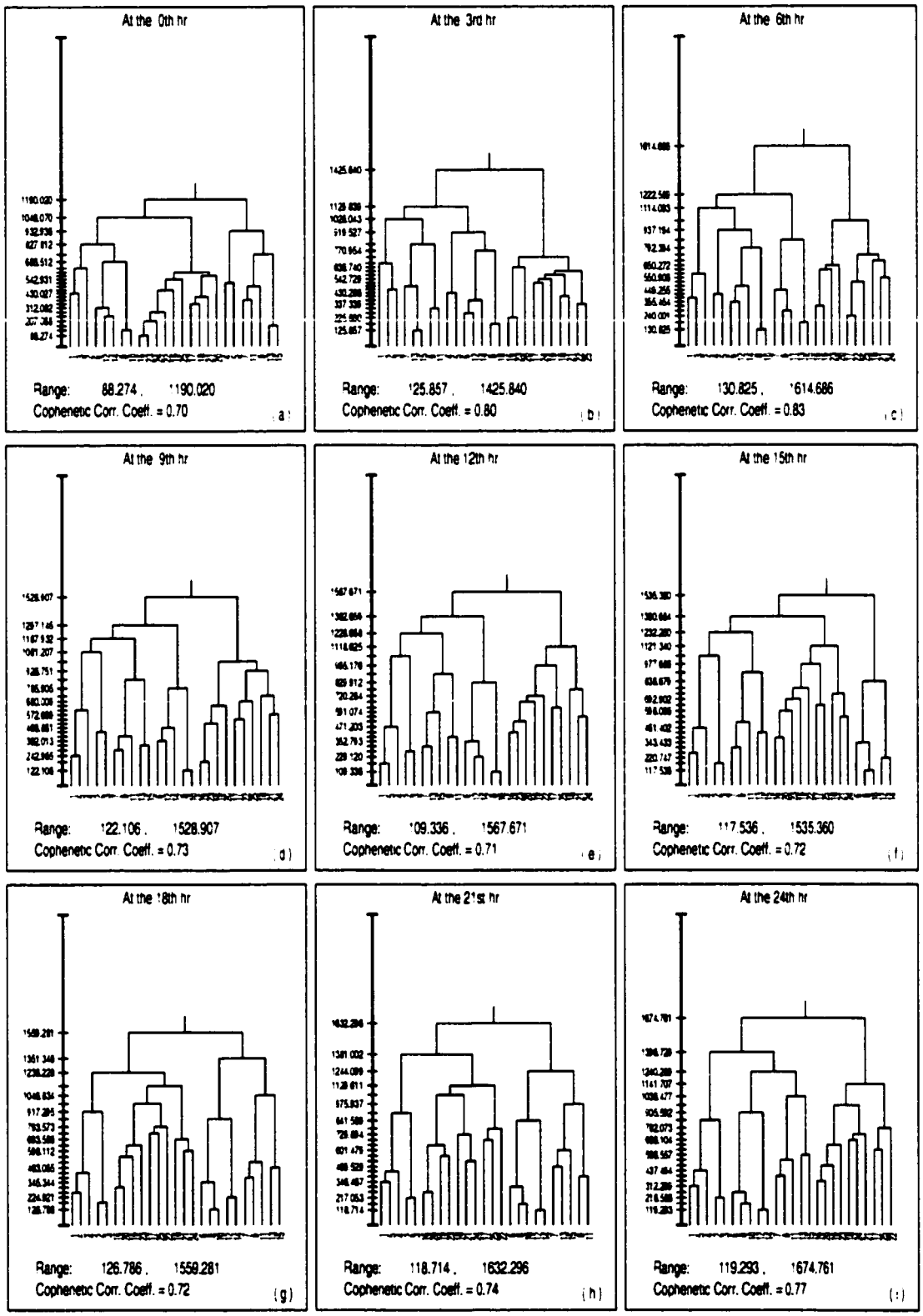


Figure 4-13b: Ward dendrograms for the Wind velocity 250mb field based on the Euclidean distance; data is centered using (2.3).

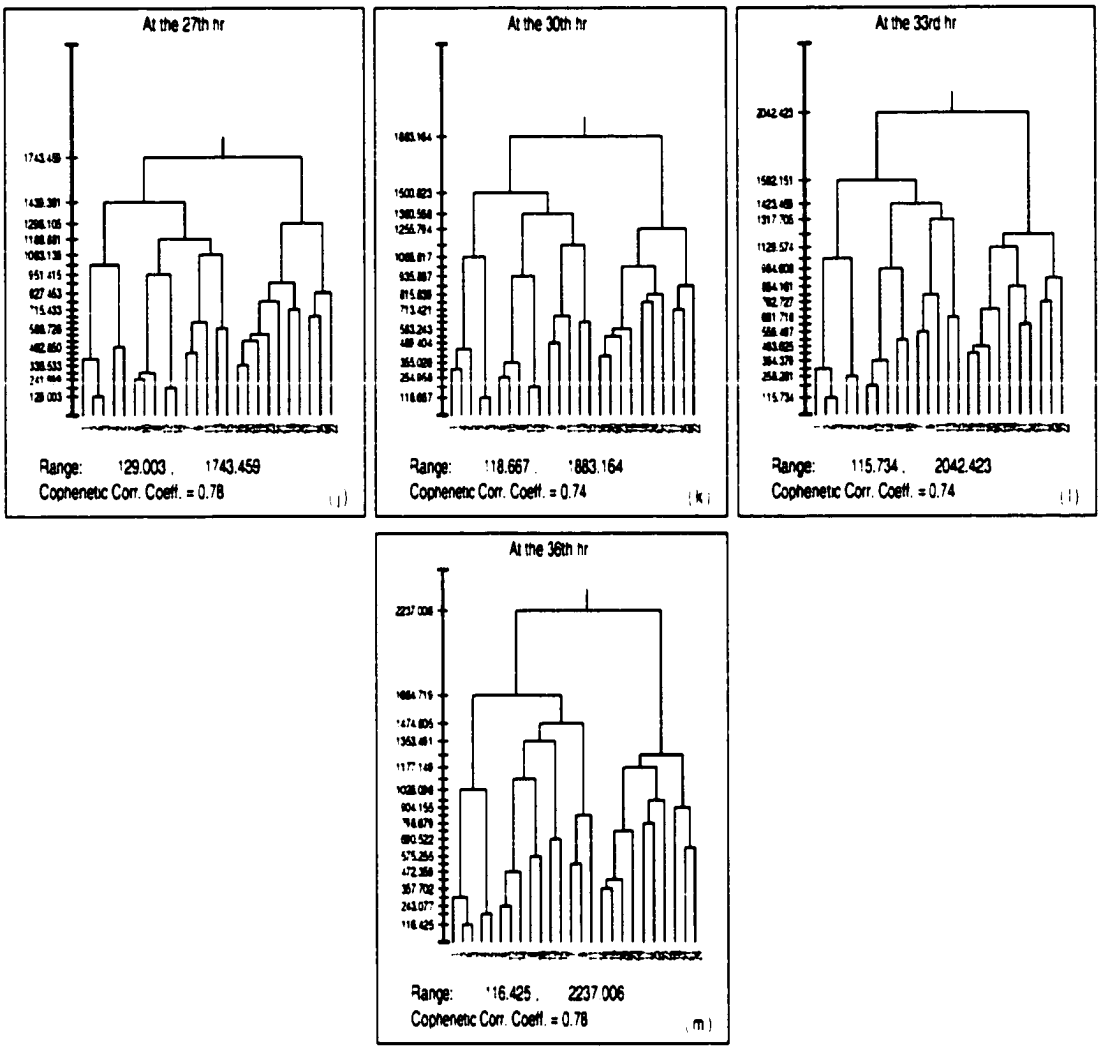


Figure 4-13b: Continued

D. 2-m Temperature Field

Figure 4-14a shows the clustering dendrograms resulting from applying UPGMA method on the temperature data set based on the correlation matrices. As of the finishing time, all members of the ensemble are joined into one large cluster at 0.86 degree of correlation. Whereas, at the initial time, they are merged at 0.82 degree of correlation. Thus the 25 members are largely correlated and getting more correlated and close to each other as the time passes.

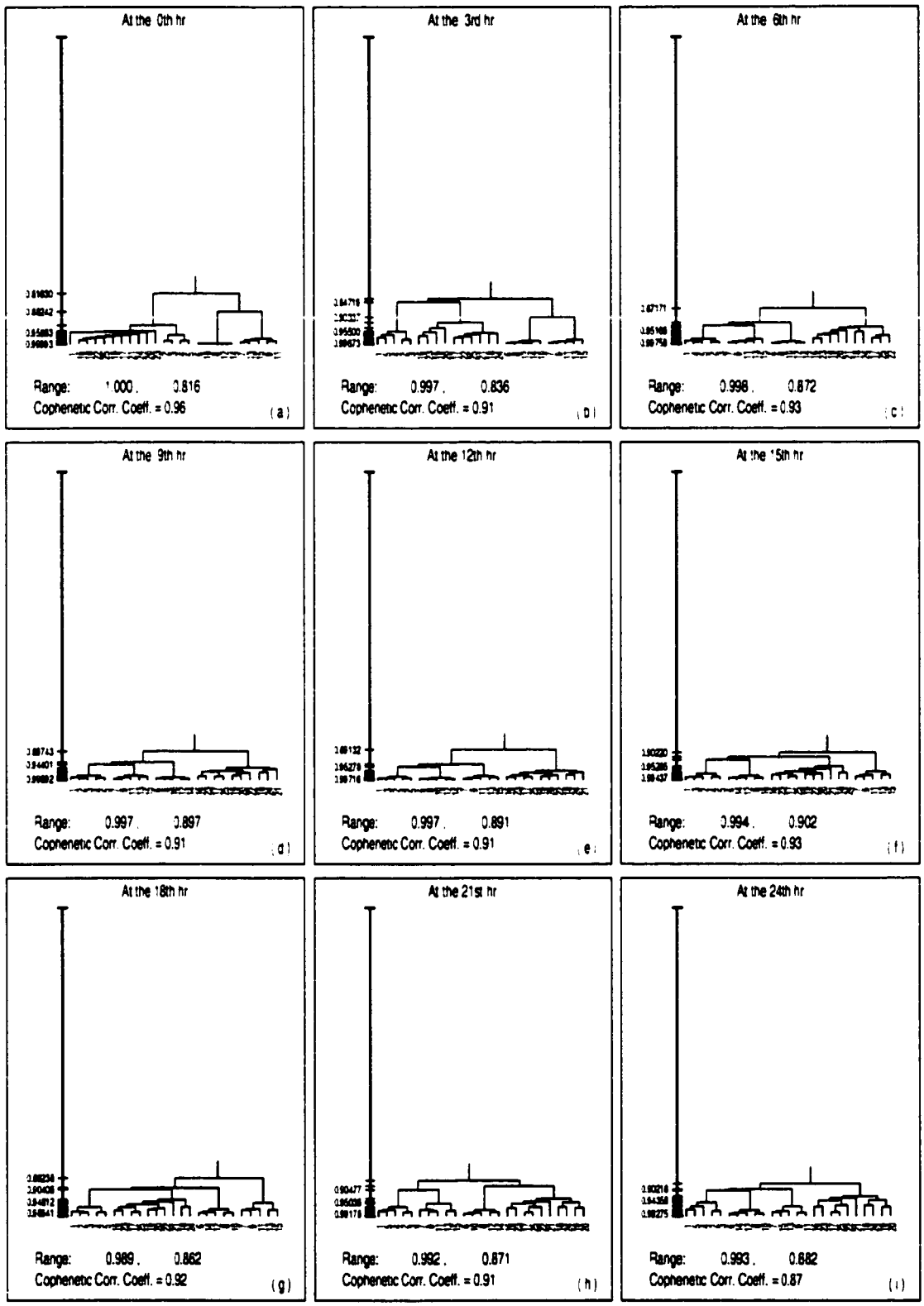


Figure 4-14a: UPGMA dendrograms for the Temperature field based on the Correlation.

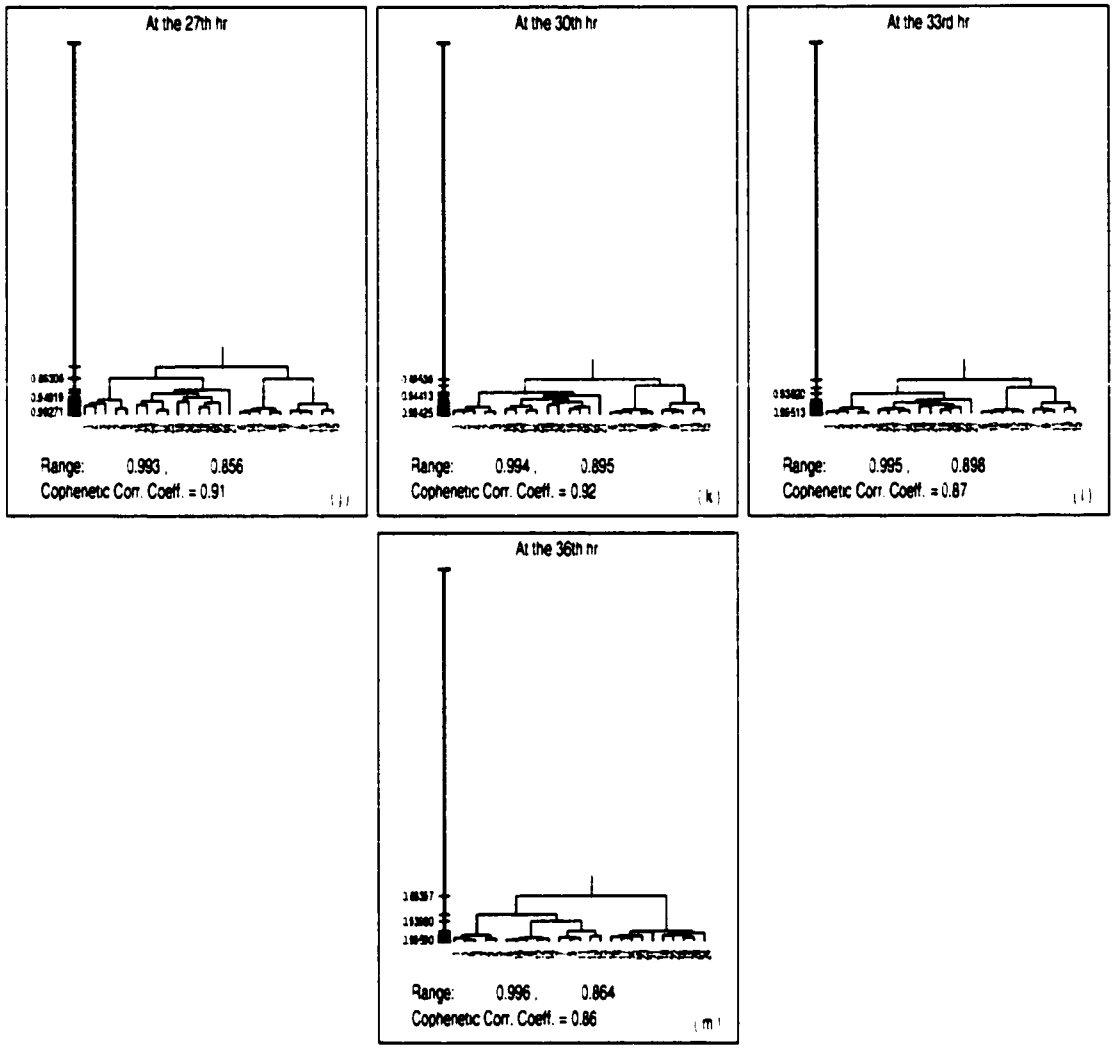


Figure 4-14a: Continued

Starting from the 3rd hour and until the end, the members follow the 4-model clustering structure (i.e., [ARPS], [Eta], [RSM], [MM5]).

The clustering dendrograms resulting from applying Ward's method based on the Euclidean distance are shown in Figure 4-14b. At all times, the members follow the 4-model clustering structure (i.e., [ARPS], [Eta], [RSM], [MM5]). By looking at the levels of mergence, the resultant 4 clusters are reasonably compact and isolated. The Eta cluster seems to be the most compact cluster at all times.

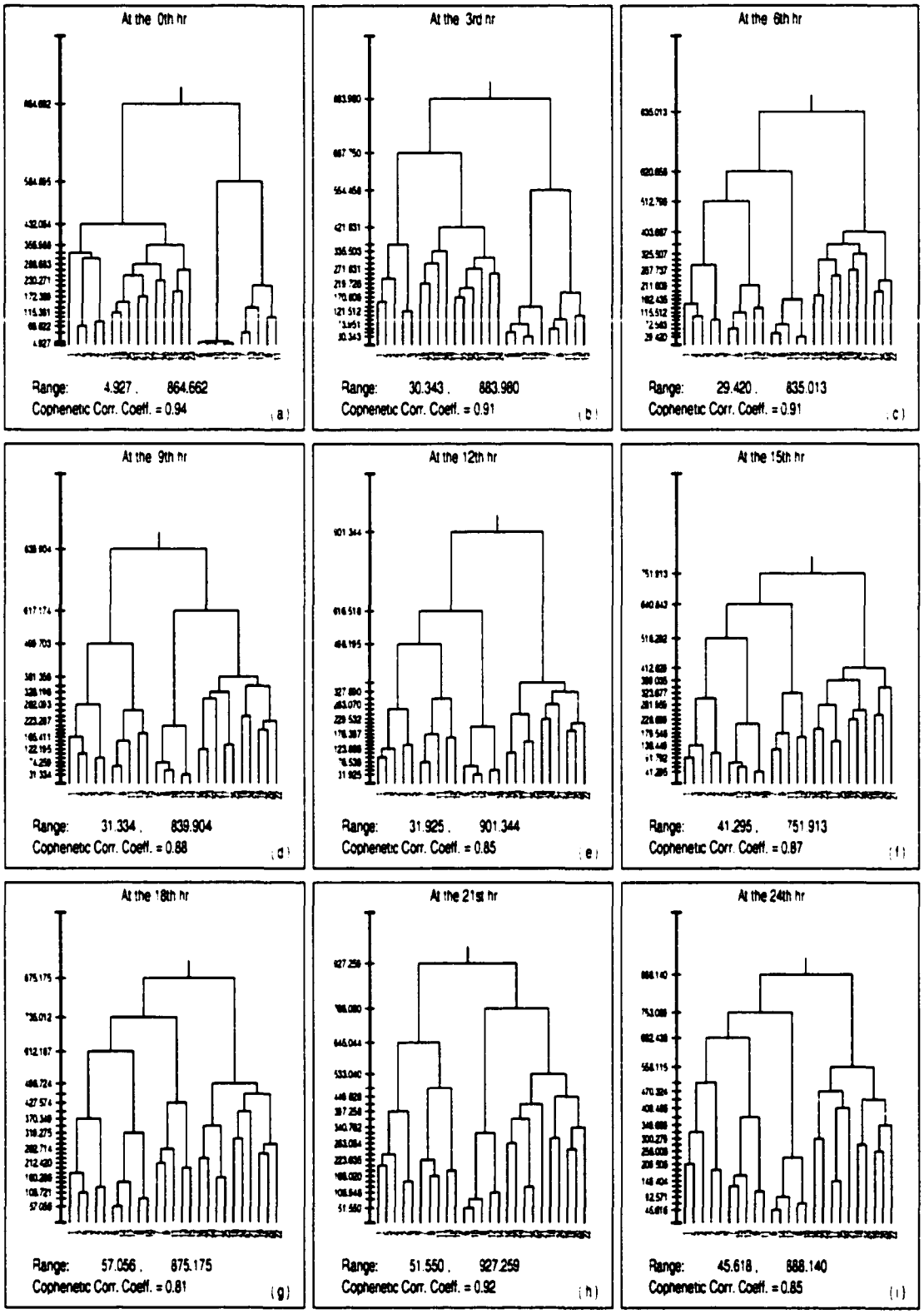


Figure 4-14b: Ward dendrograms for the Temperature field based on the Euclidean distance; data is centered using (2.3).

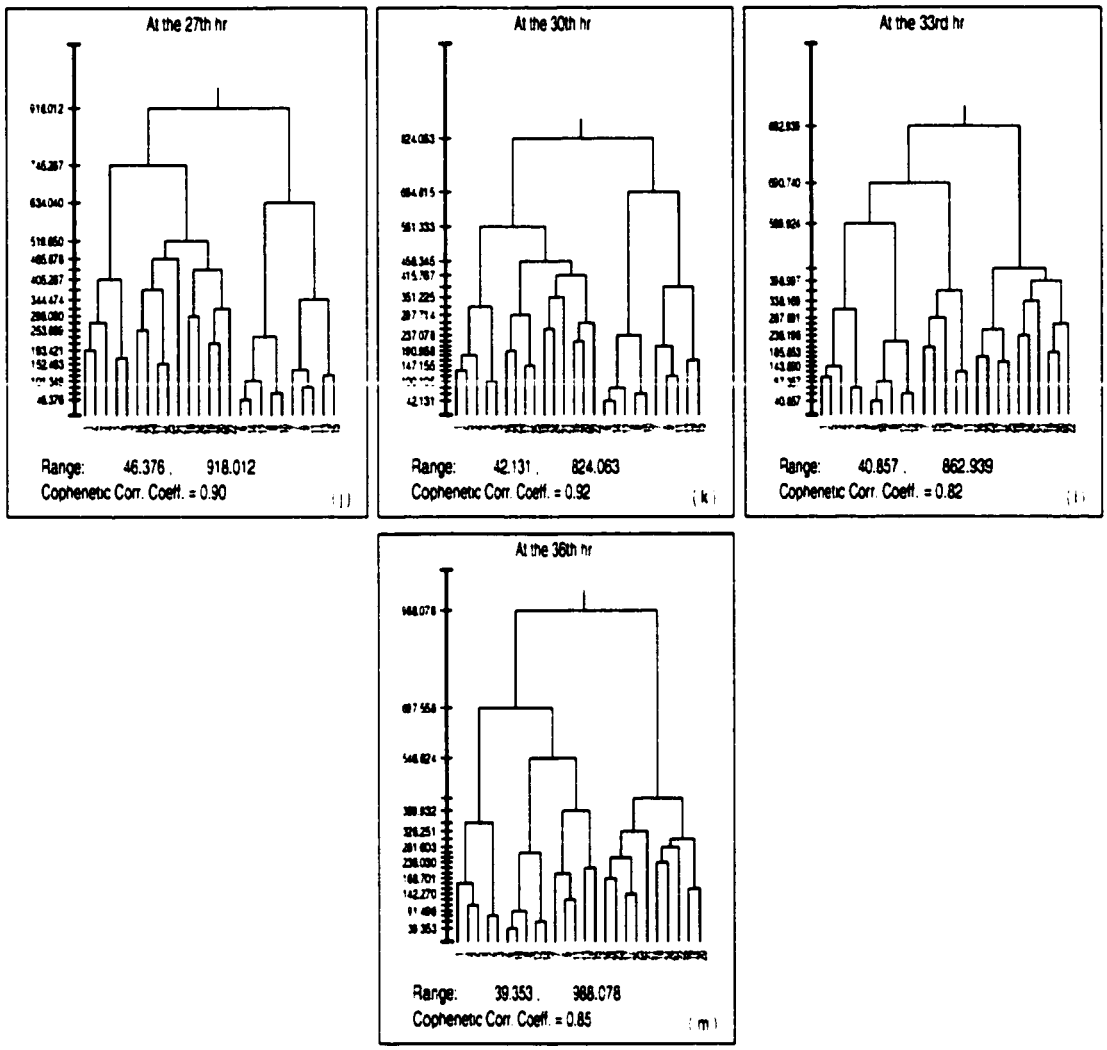


Figure 4-14b: Continued

E. 250-mb Absolute Vorticity (ABSVORT250) Field

Figure 4-15a shows the clustering dendrograms resulting from applying UPGMA method on the ABSVORT250 data set based on the correlation matrices. Throughout the times the Eta members formed their own cluster at more than 0.8 degree of correlation, hence their cluster is compact. MM5 cluster always build itself at low level of correlation when approaching the finishing time.

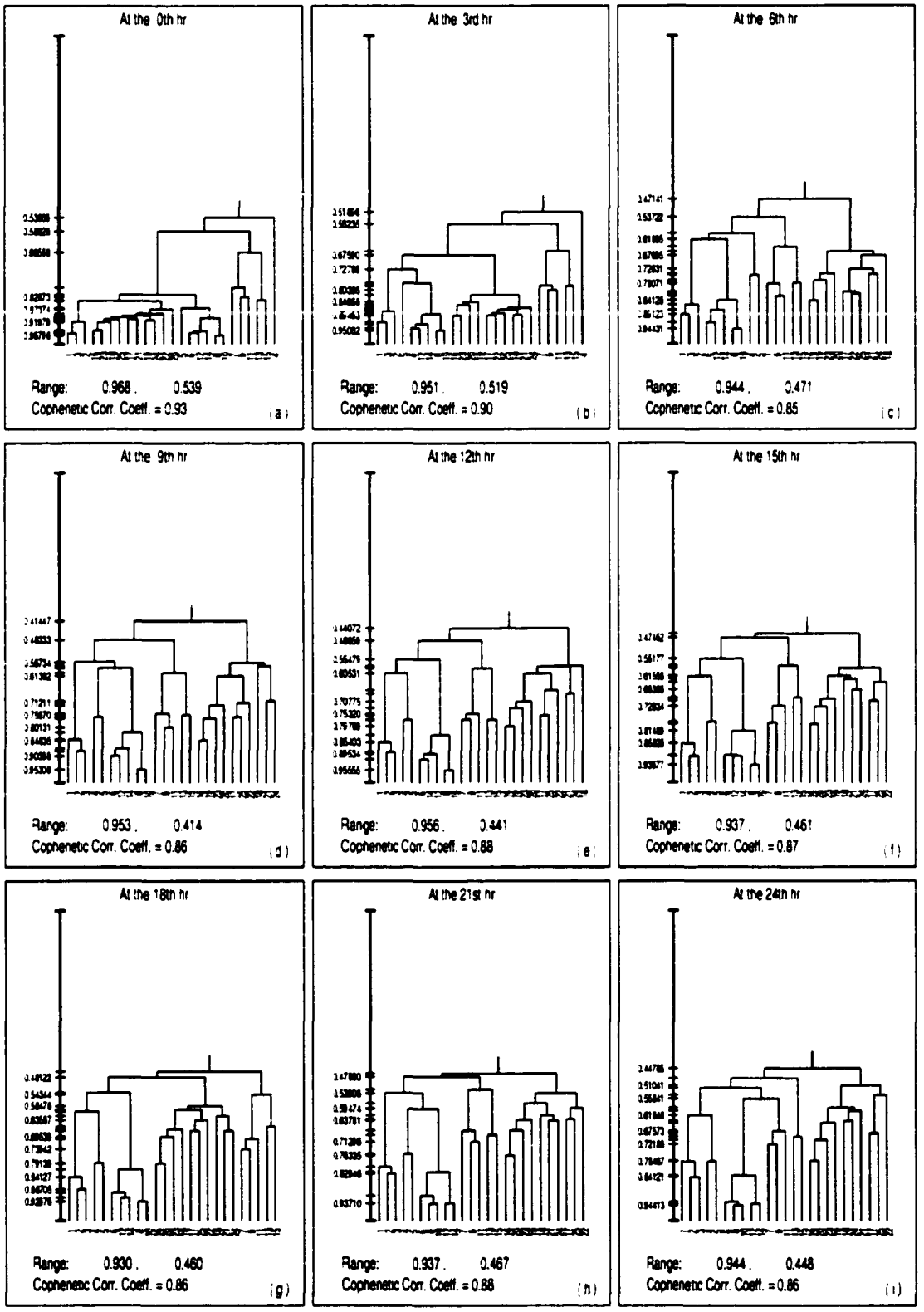


Figure 4-15a: UPGMA dendrograms for the Abs. Vorticity 250mb field based on the Correlation.

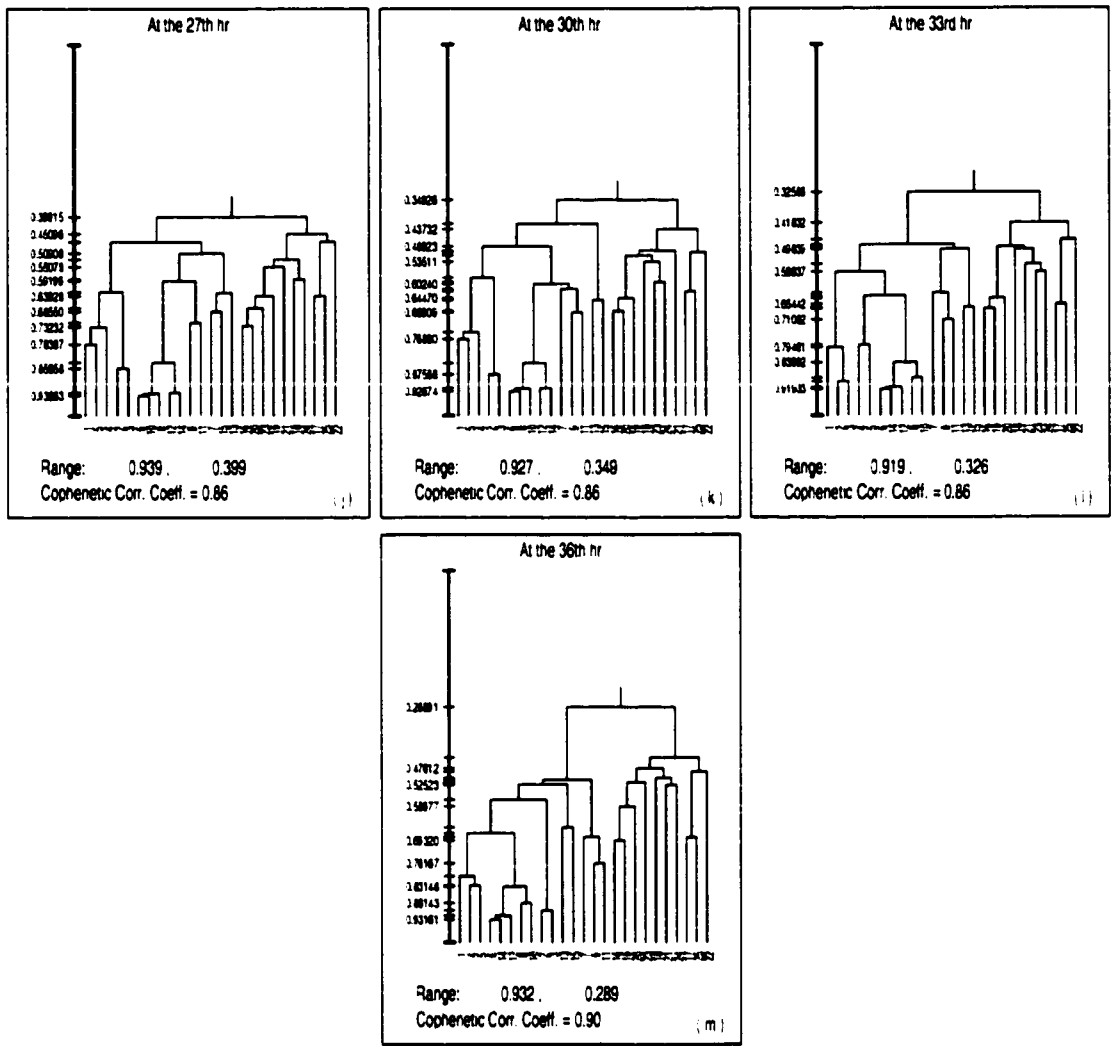


Figure 4-15a: Continued

Therefore, this cluster is isolated but not compact. Other members have no consistent behavior throughout the time. In a few cases they formed clusters that represent the models they belong to but in most cases they are scattered around into small clusters. However, no inter-model cluster is formed at acceptable degree of correlation.

The clustering dendrograms resulting from applying Ward's method on the ABSVORT250 data set based on the Euclidean distance are shown in Figure 4-15b.

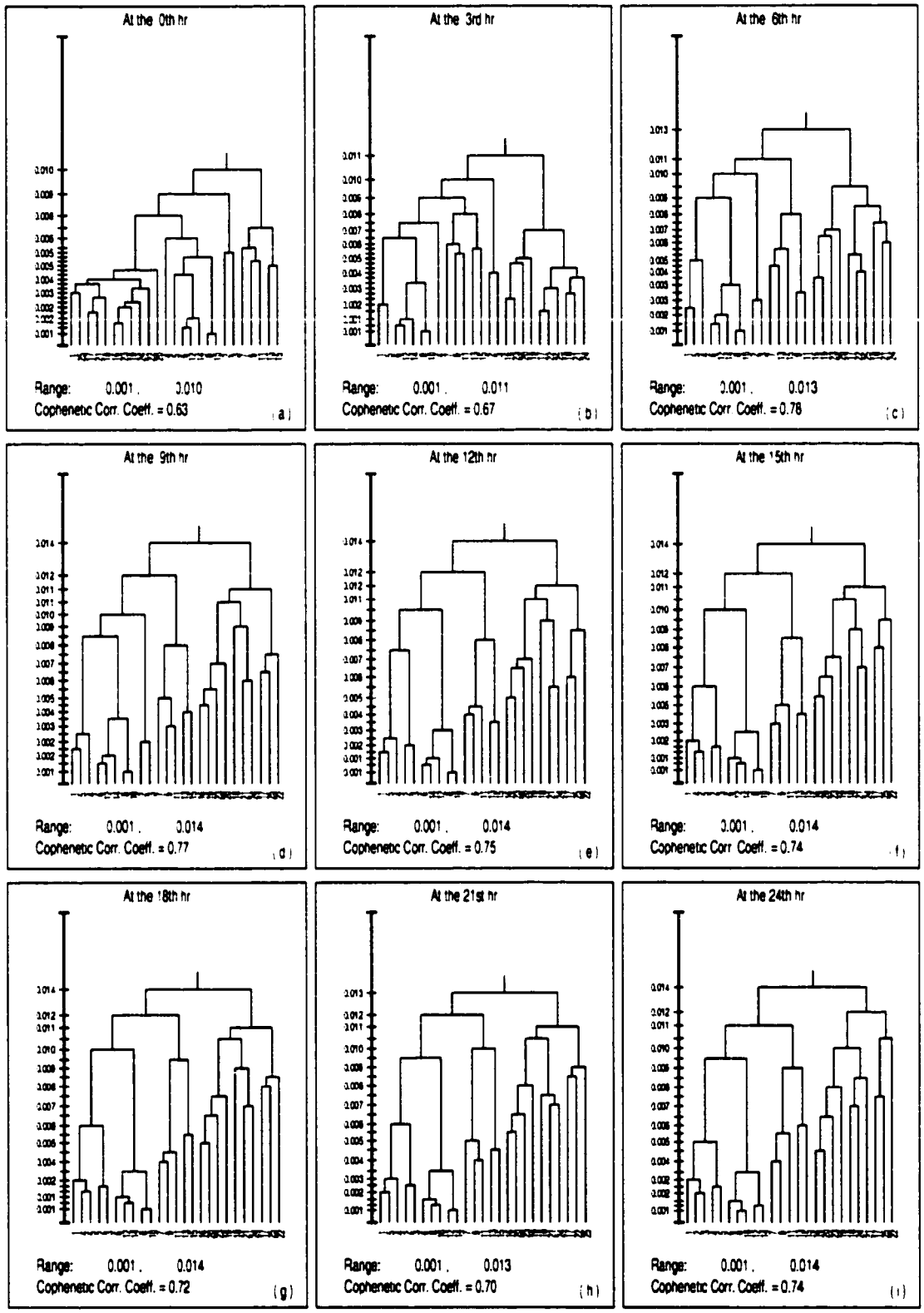


Figure 4-15b: Ward dendrograms for the Abs. Vorticity 250mb field based on the Euclidean distance; data is centered using (2.3).

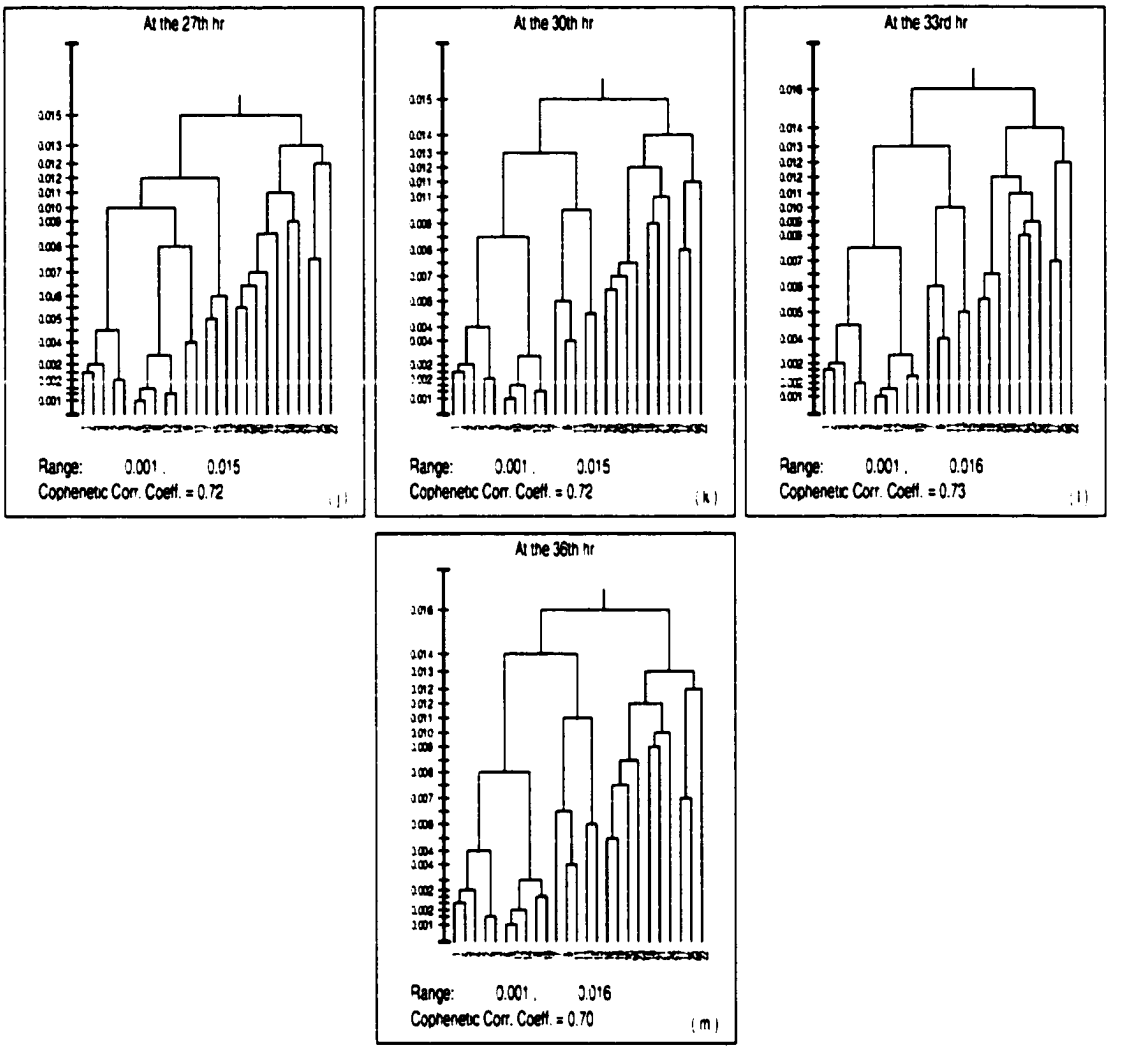


Figure 4-15b: Continued

Starting from the 9th hour and until the end, the members follow the 4-model clustering structure. The Eta cluster is the most compact cluster then comes the ARPS cluster. These two clusters seem to be isolated from the others. The overall results of the two analyses for the ABSVORT250 field are different. However, both analyses find no inter-model cluster and the both build [Eta] and [ARPS] as separate clusters.

F. 500-mb Absolute Vorticity (ABSVORT500)Field

Figure 4-16a shows the clustering dendrograms resulting from applying UPGMA method on the ABSVORT500 data set based on the correlation matrices.

From the beginning until the 21st hour, Eta members and MM5 members formed their own clusters. Other members have no consistent behavior but no inter-model cluster is found out of the individual members of the ARPS and RSM models. These members either build a cluster out of the members belonging to one model or scatter into small clusters before joining others to form large cluster. From the 24th hour until the end, the 4-model clustering structure is the recovered solutions. The Eta cluster is observed to be the most compact cluster at all times.

The clustering dendrograms resulting from applying Ward's method on the ABSVORT500 data set based on the Euclidean distance are shown in Figure 4-16b. During the first a few times, the members of each model build their own cluster except the ARPS members. The ARPS members are broken into two small clusters but no inter-model cluster is formed out of this break up. Starting from the 15th hour until the end, the members follow the 4-model clustering structure. The ARPS and Eta clusters appear to be compact and isolated.

Both similarity- and dissimilarity-based analyses for the ABSVORT500 field have the same result especially for the second half of the time period.

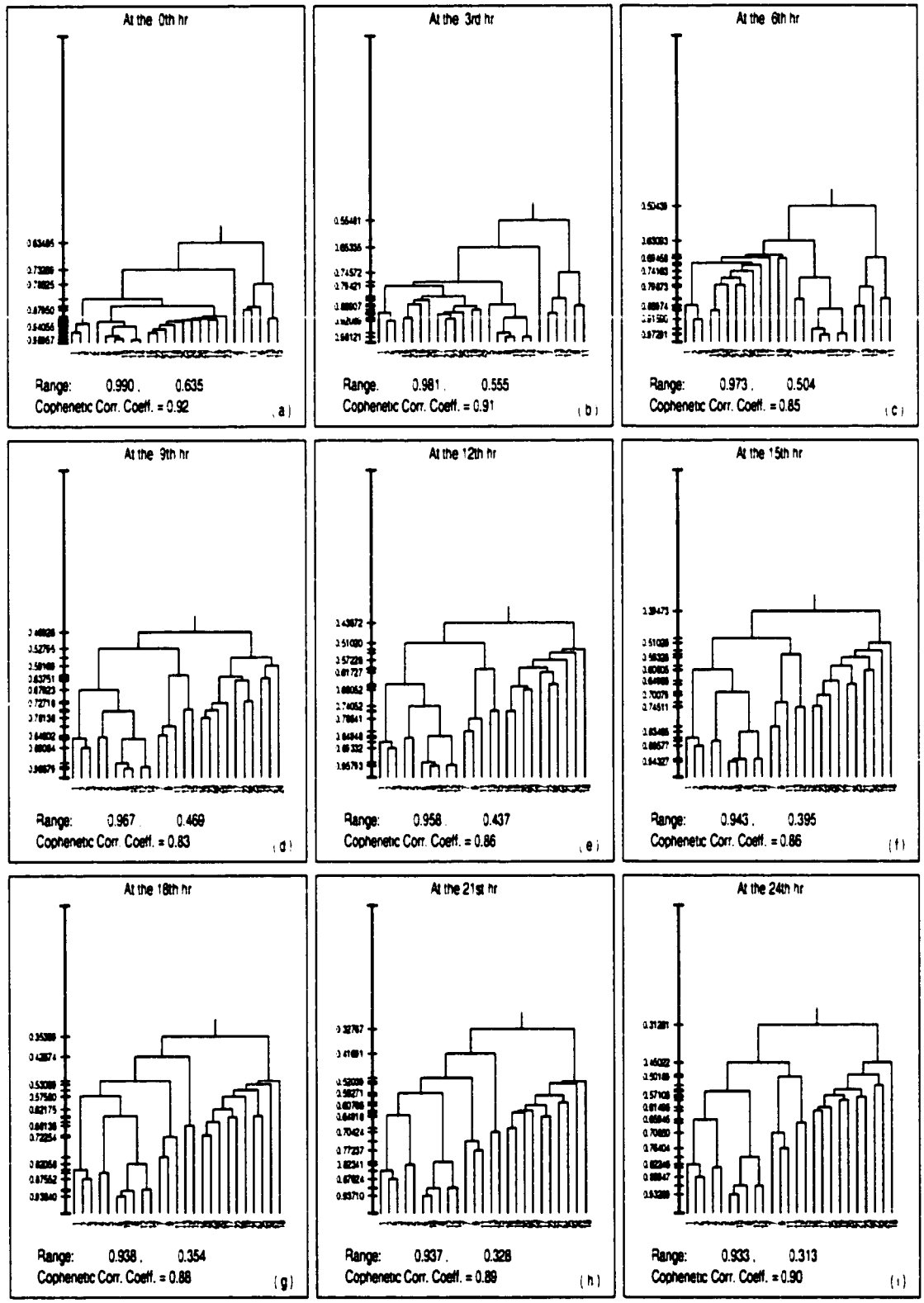
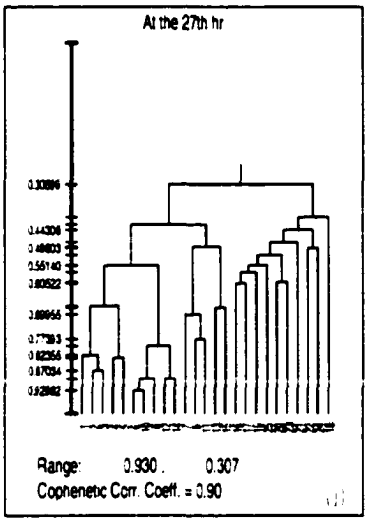
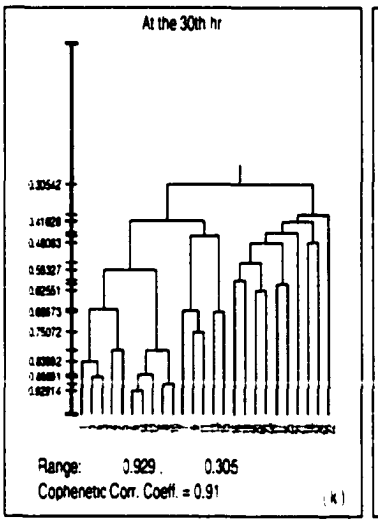
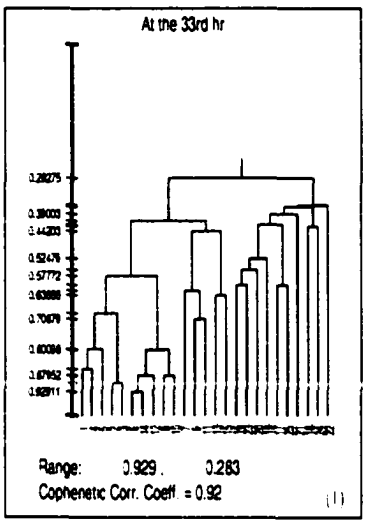


Figure 4-16a: UPGMA dendrograms for the Abs. Vorticity 500mb field based on the Correlation.







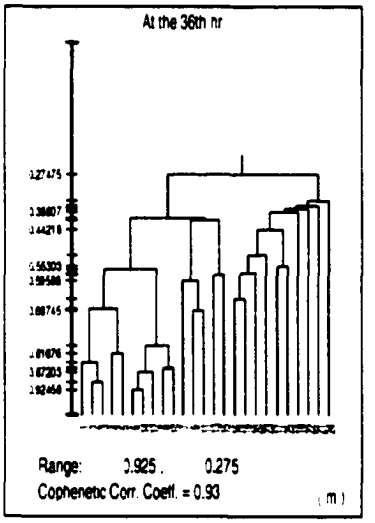


Figure 4-16a: Continued

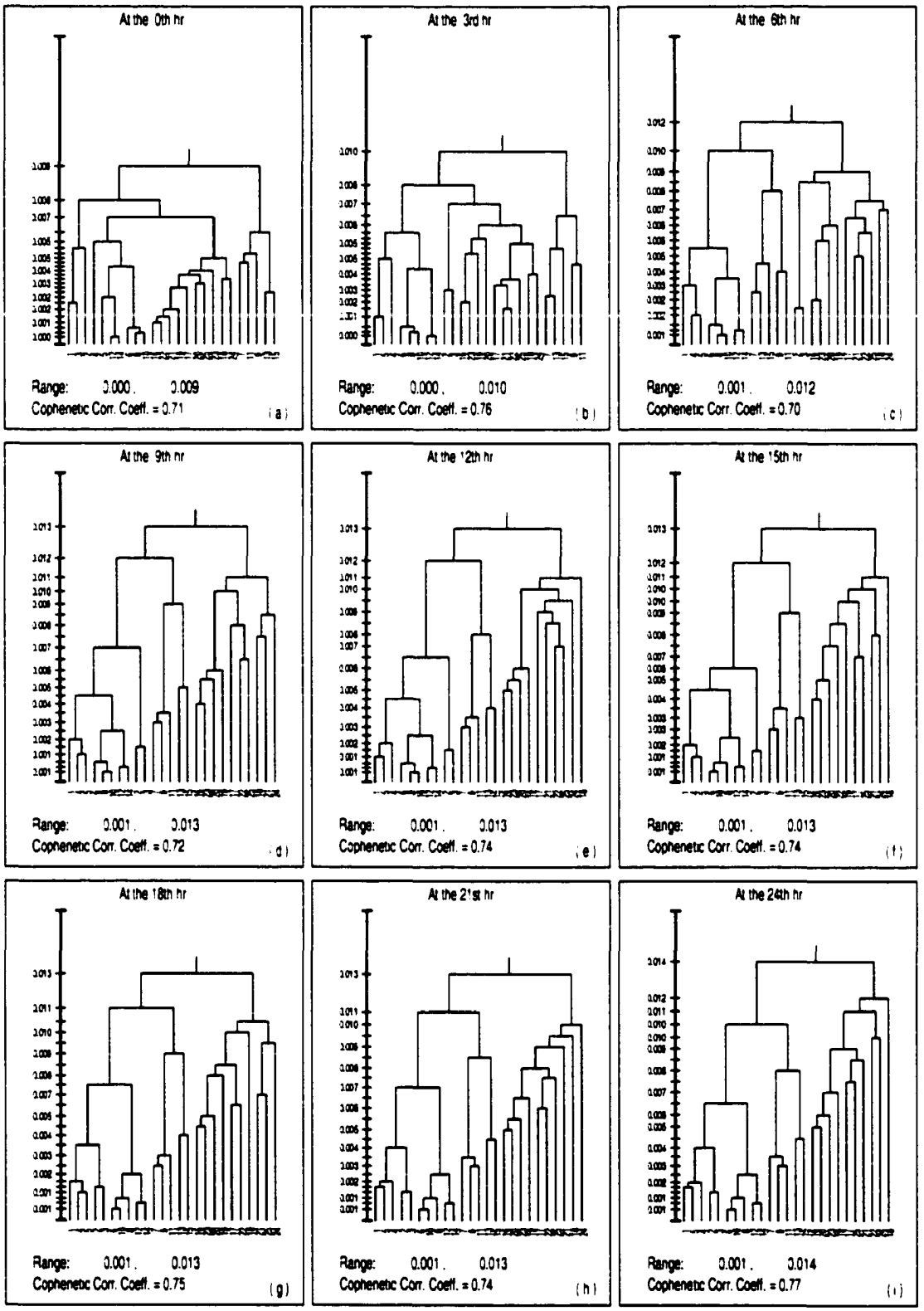


Figure 4-16b: Ward dendrograms for the Abs. Vorticity 500mb field based on the Euclidean distance; data is centered using (2.3).

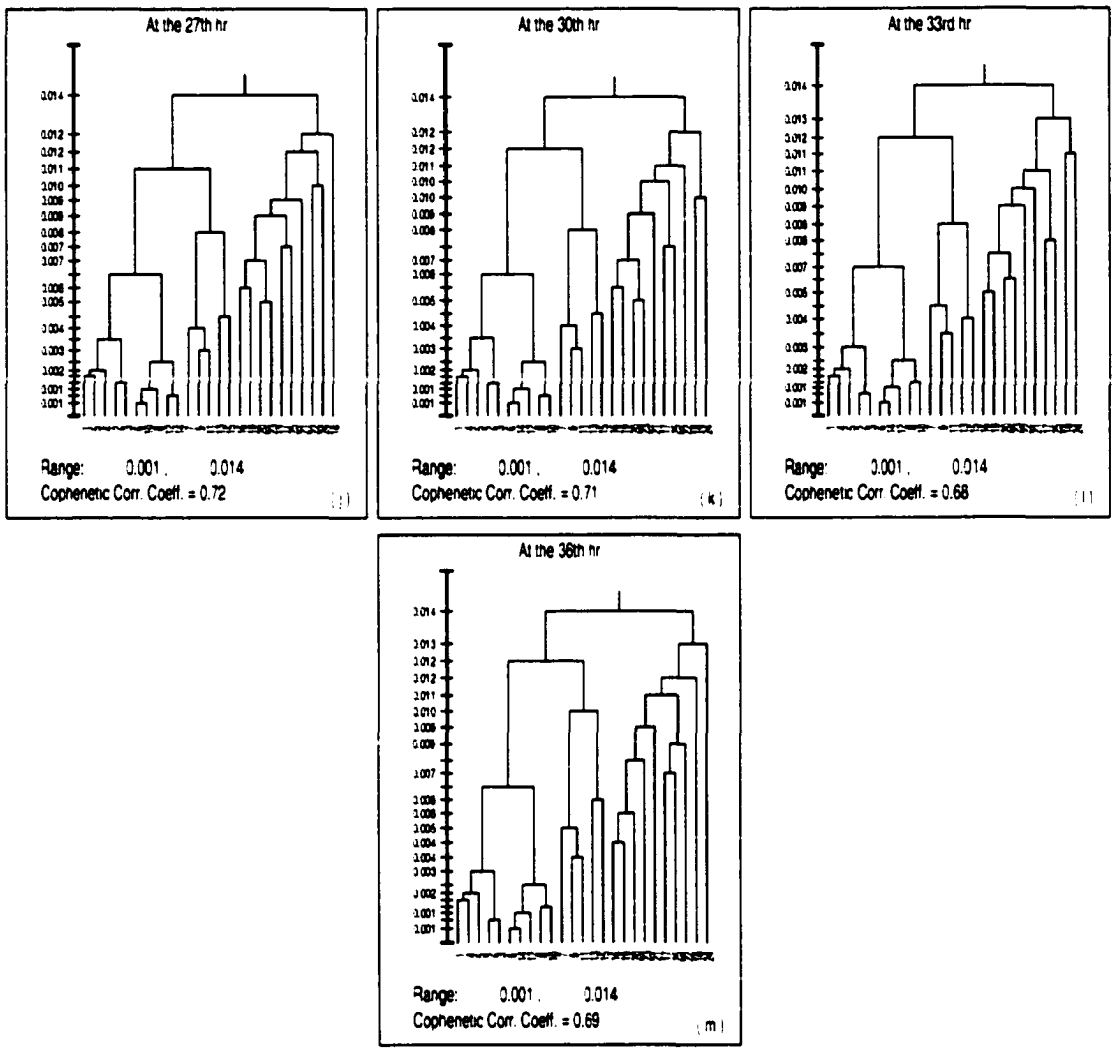


Figure 4-16b: Continued

G. 850-mb Absolute Vorticity (ABSVORT850) Field

Figure 4-17a shows the clustering dendrograms resulting from applying UPGMA method on the ABSVORT850 data set based on the correlation matrices. Starting from the 6th hour and until the end, the members follow the 4-model clustering structure. The Eta cluster is the most compact cluster but the members of each one of the 4 clusters diverge within the cluster itself when approaching the finishing time.

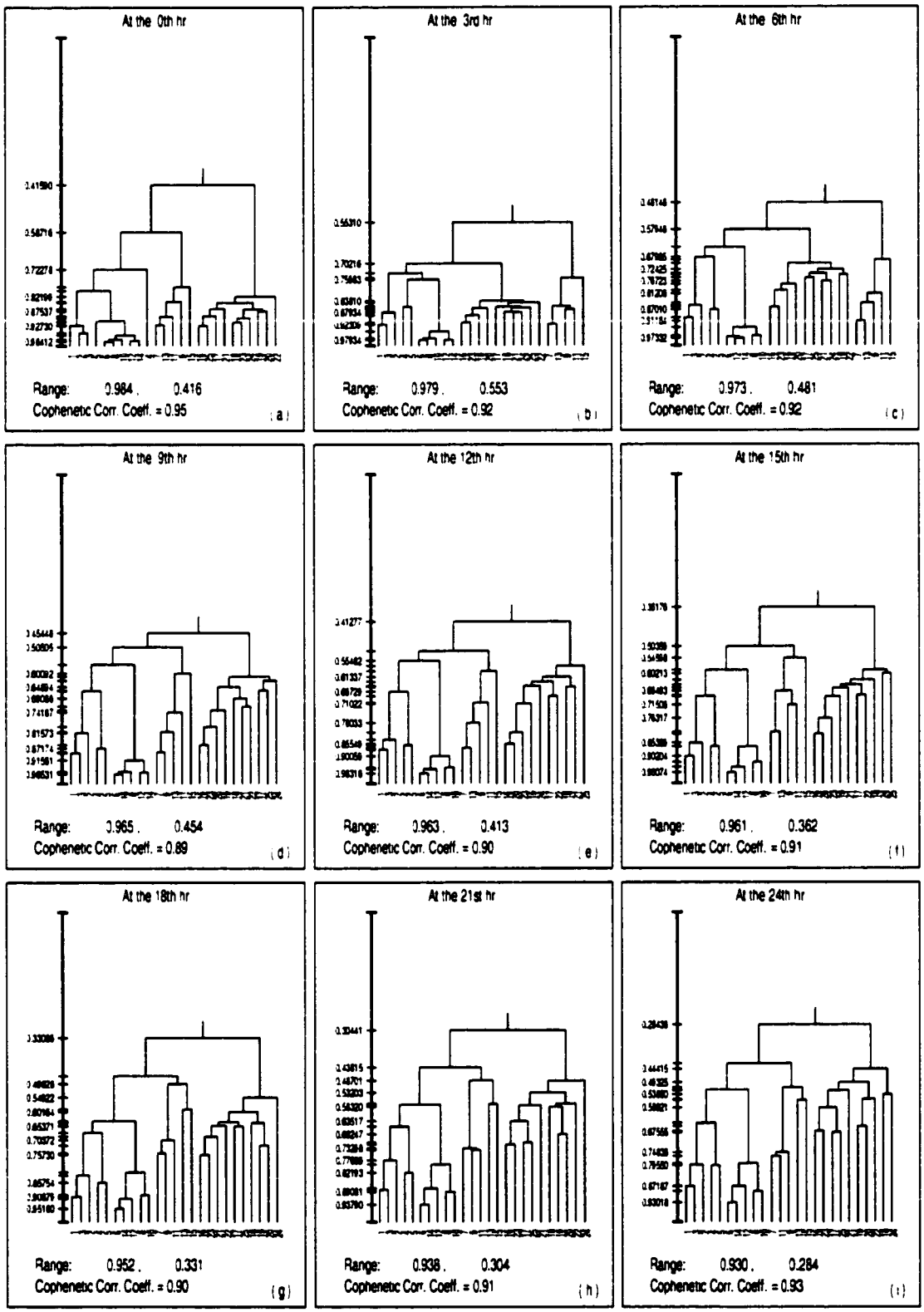


Figure 4-17a: UPGMA dendrograms for the Abs. Vorticity 850mb field based on the Correlation.

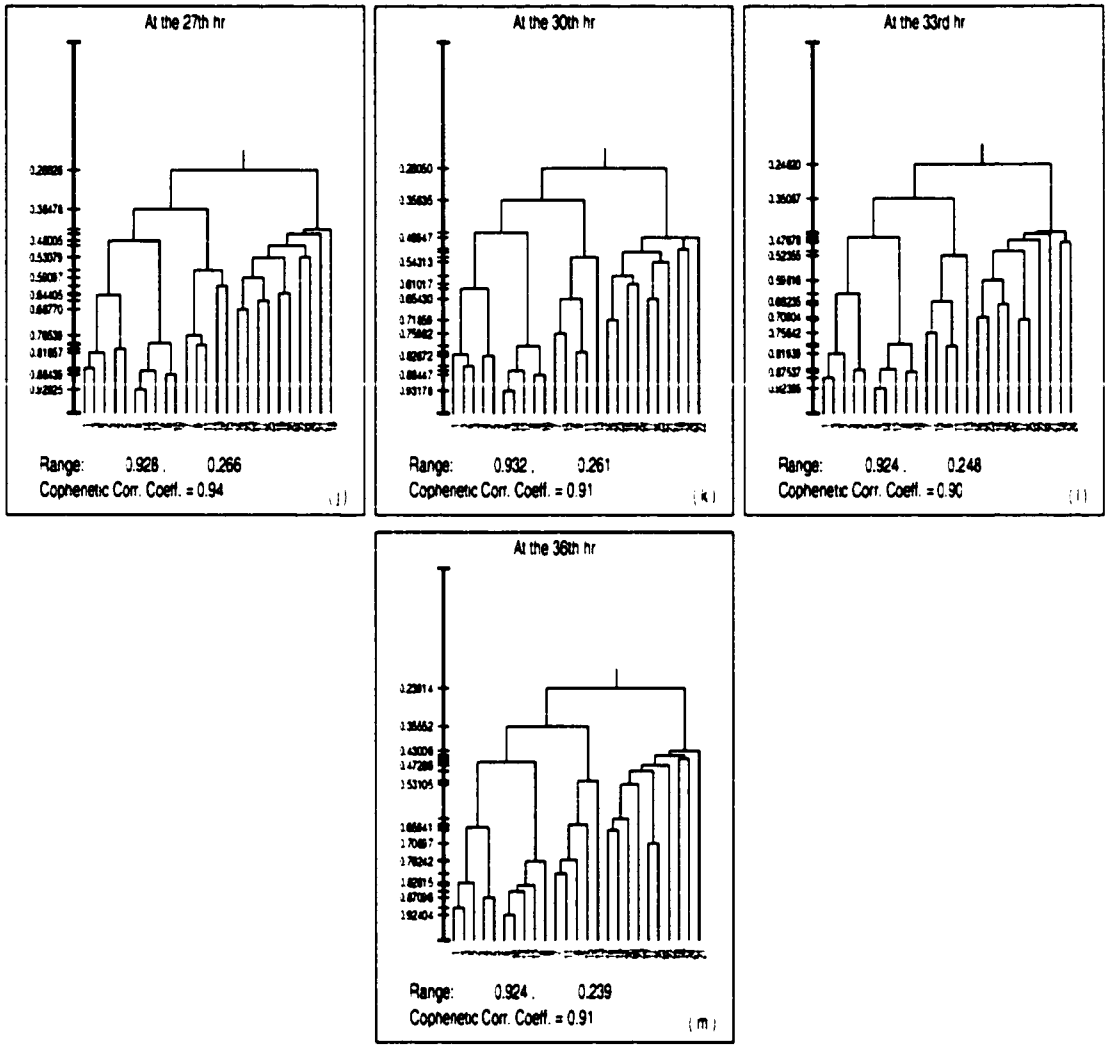


Figure 4-17a: Continued

The clustering dendrograms resulting from applying Ward's method on the ABSVORT850 data set based on the Euclidean distance are shown in Figure 4-17b. Starting from the 6th hour and until the end, the members follow the 4-model clustering structure. The 4 clusters are observed to be isolated but the Eta cluster is the compact one. The results of the two analyses are the same.

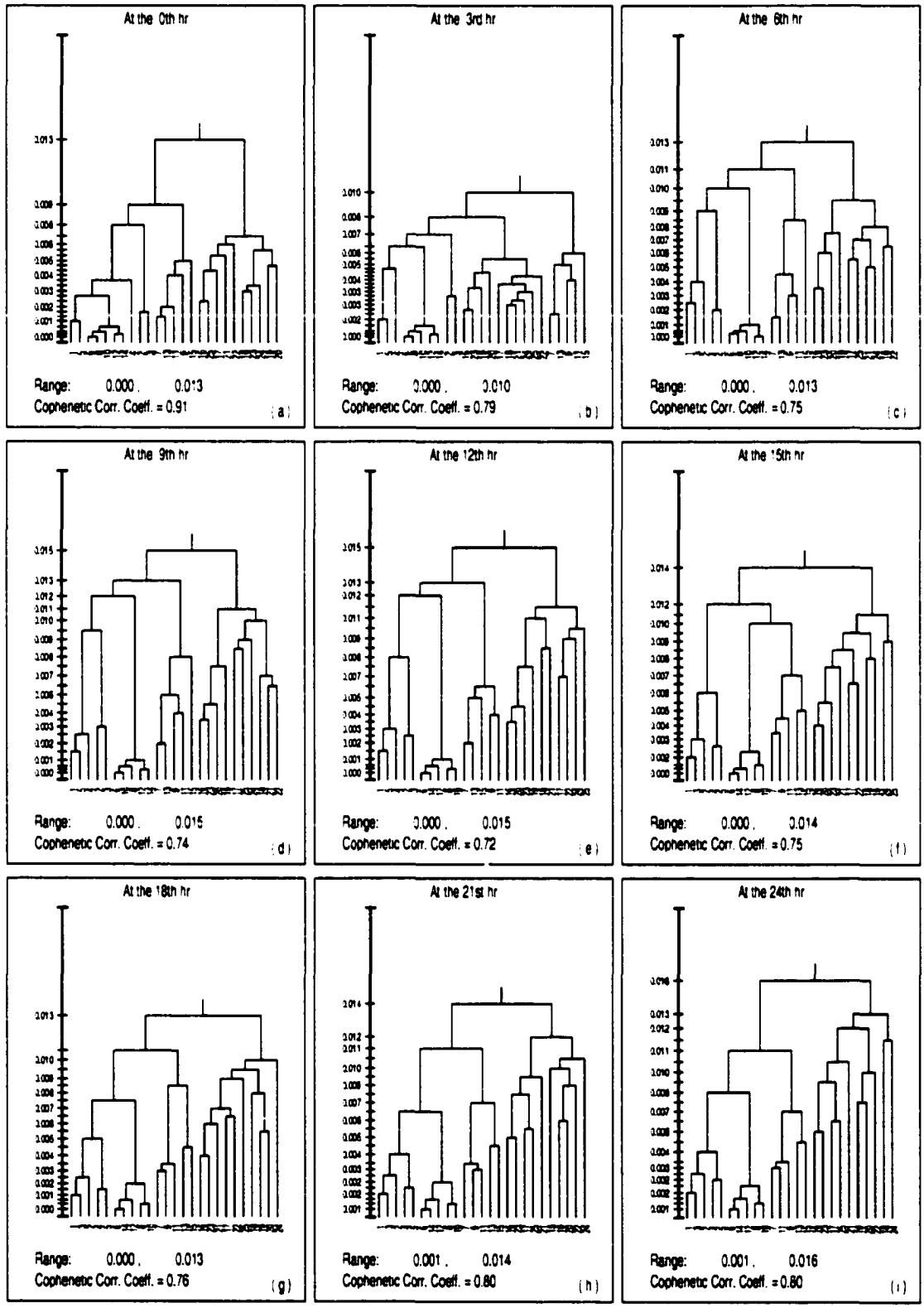


Figure 4-17b: Ward dendrograms for the Abs. Vorticity 850mb field based on the Euclidean distance; data is centered using (2.3).

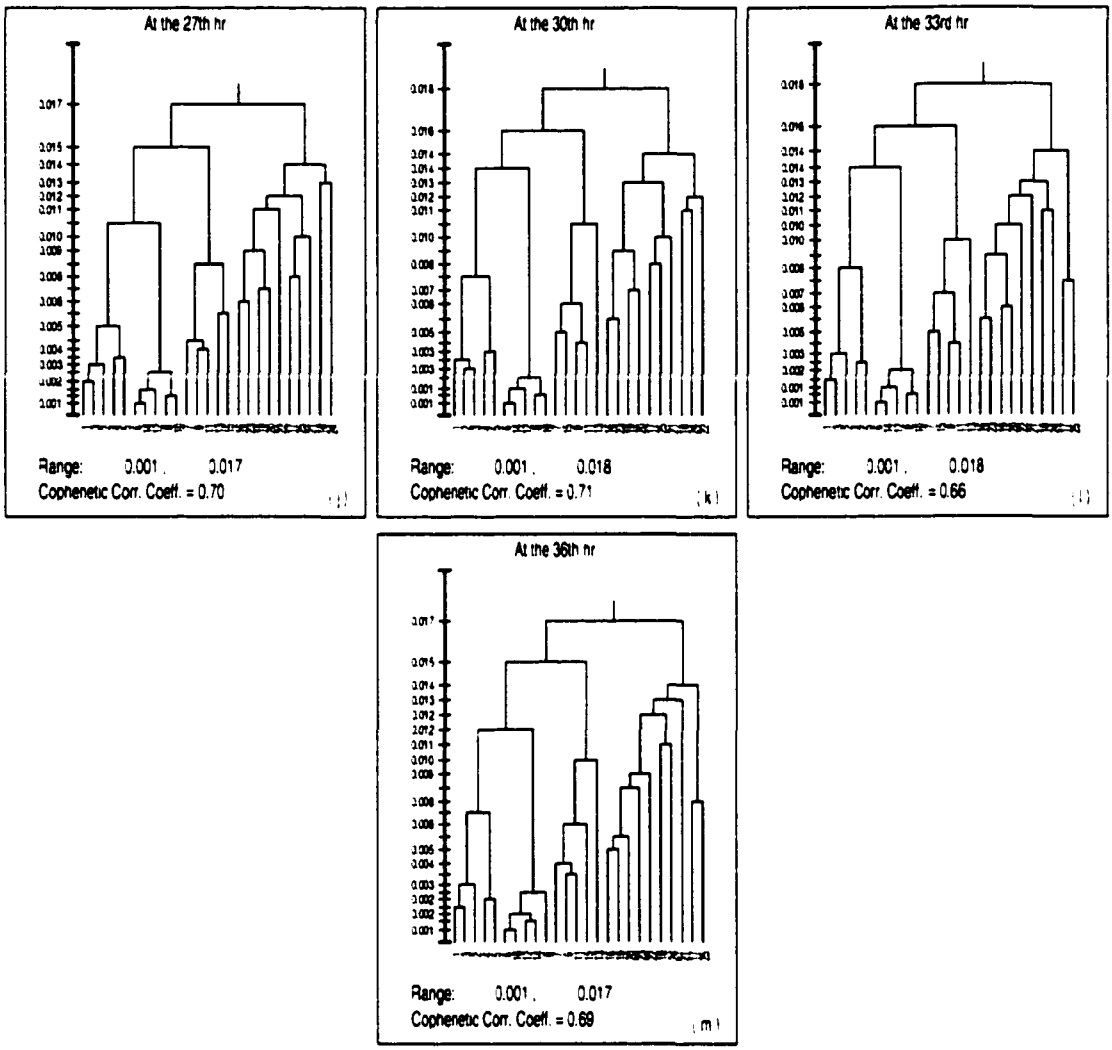


Figure 4-17b: Continued

H. 2-m Dewpoint Field

Figure 4-18a shows the clustering dendrograms resulting from applying UPGMA method on the dewpoint data set based on the correlation matrices. The members of the ensemble are largely correlated. They are all placed in one large cluster at 0.82 degree of correlation by the finishing time. At all times, members follow the 3 clusters pattern (i.e.,[ARPS], [Eta], [MM5]). Note that the members of RSM model are not available for this field.

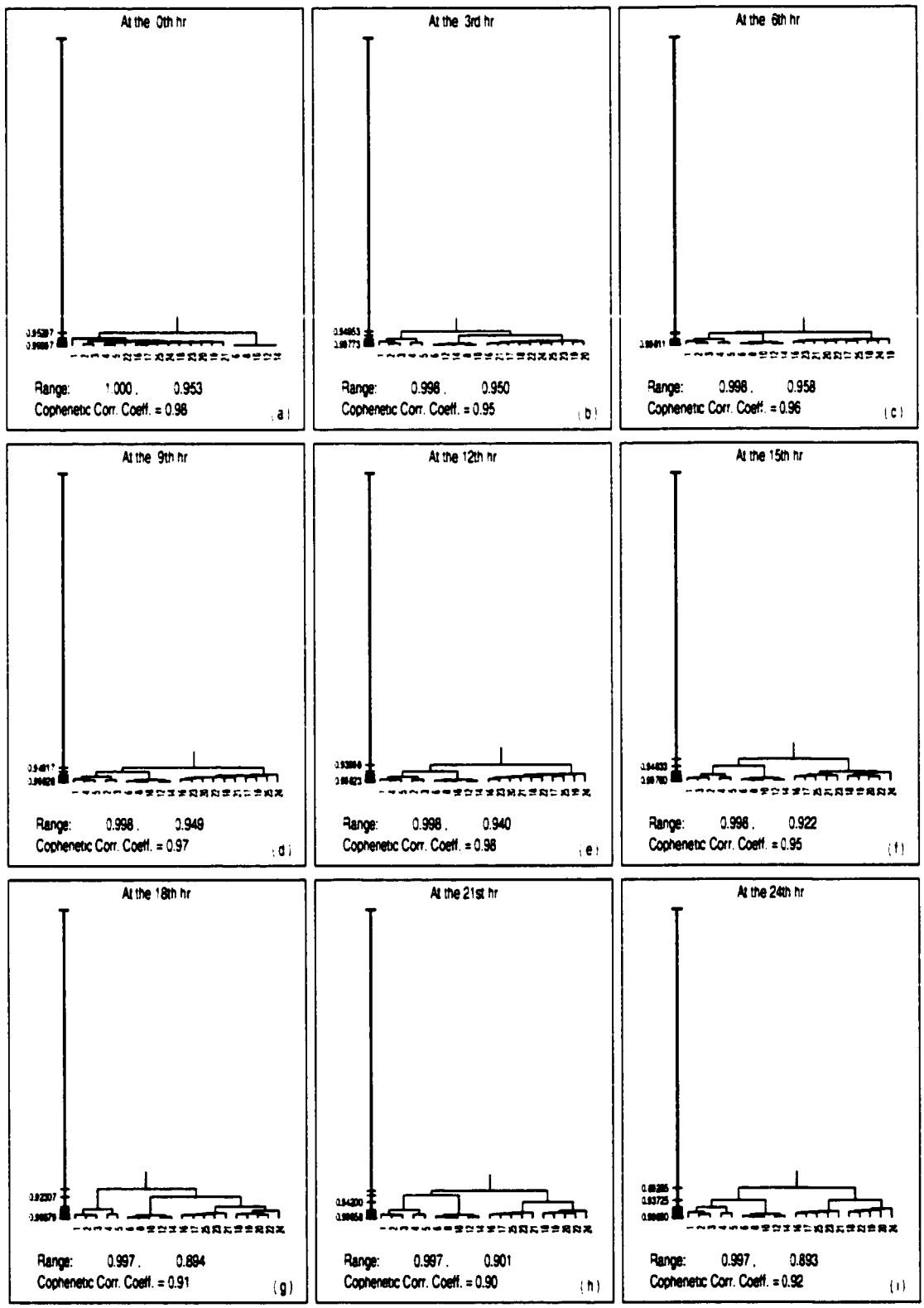


Figure 4-18a: UPGMA dendrograms for the Dewpoint field based on the Correlation.

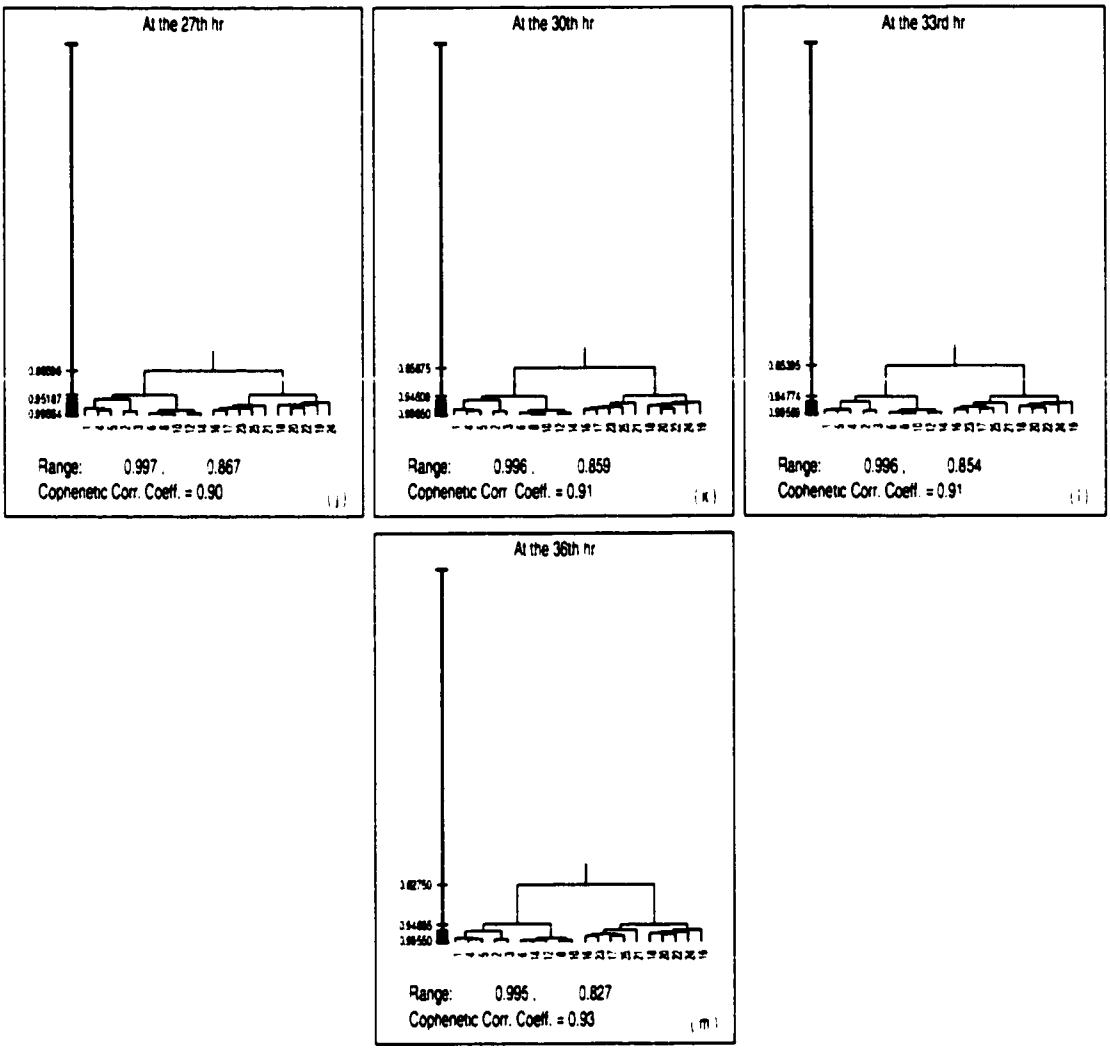


Figure 4-18a: Continued

The clustering dendrograms resulting from applying Ward's method on the dewpoint data set based on the Euclidean distance are shown in Figure 4-18b. Starting from the 3rd hour and until the end, the members follow the 3 clusters pattern (i.e.,[ARPS], [Eta], [MM5]). Note that the members of RSM model are not available for this field. The Eta cluster is always built first and all 3 clusters seem to be isolated. Both analyses have the same results and both find no inter-model cluster.

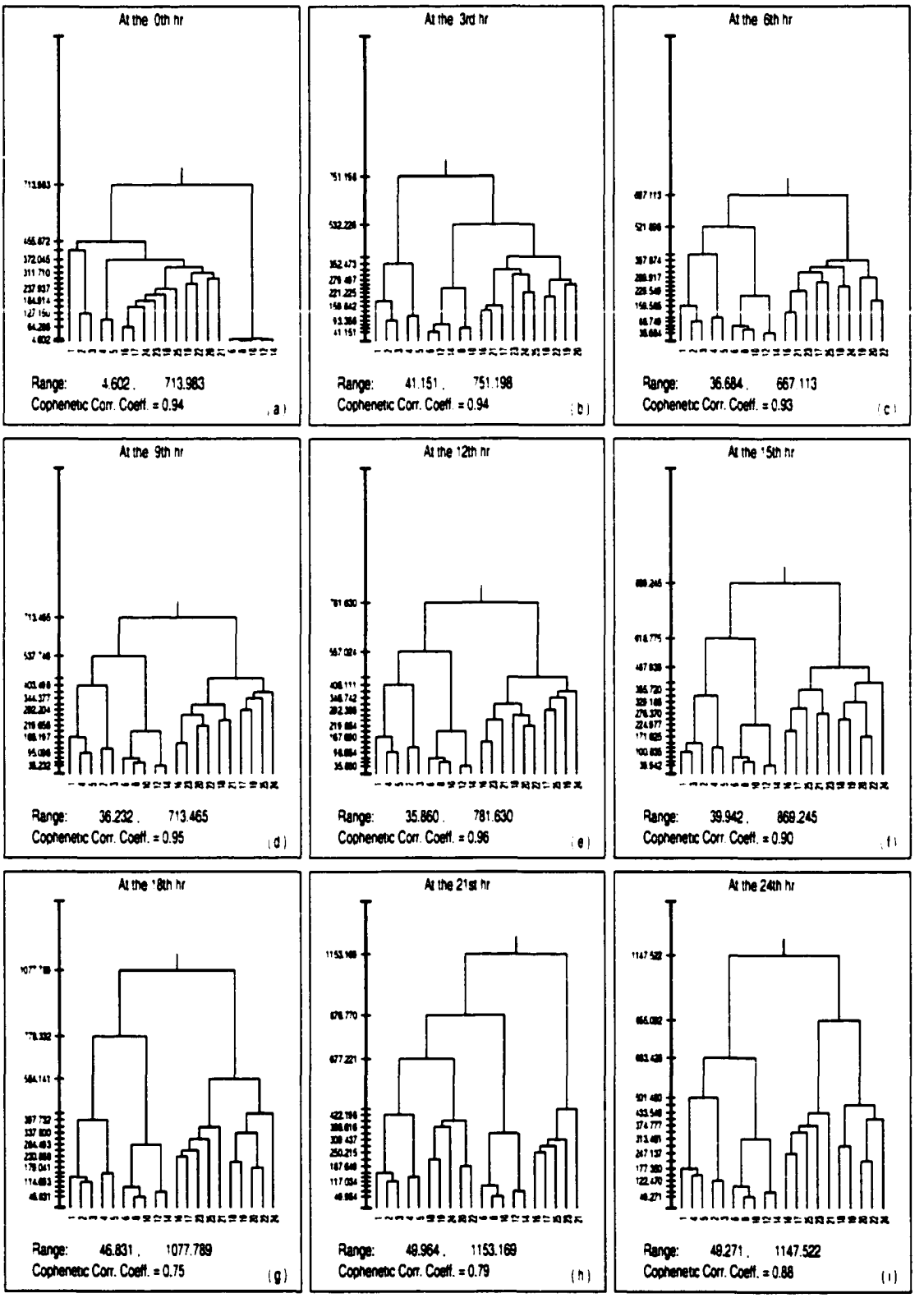


Figure 4-18b: Ward dendrograms for the Dewpoint field based on the Euclidean distance; data is centered using (2.3).

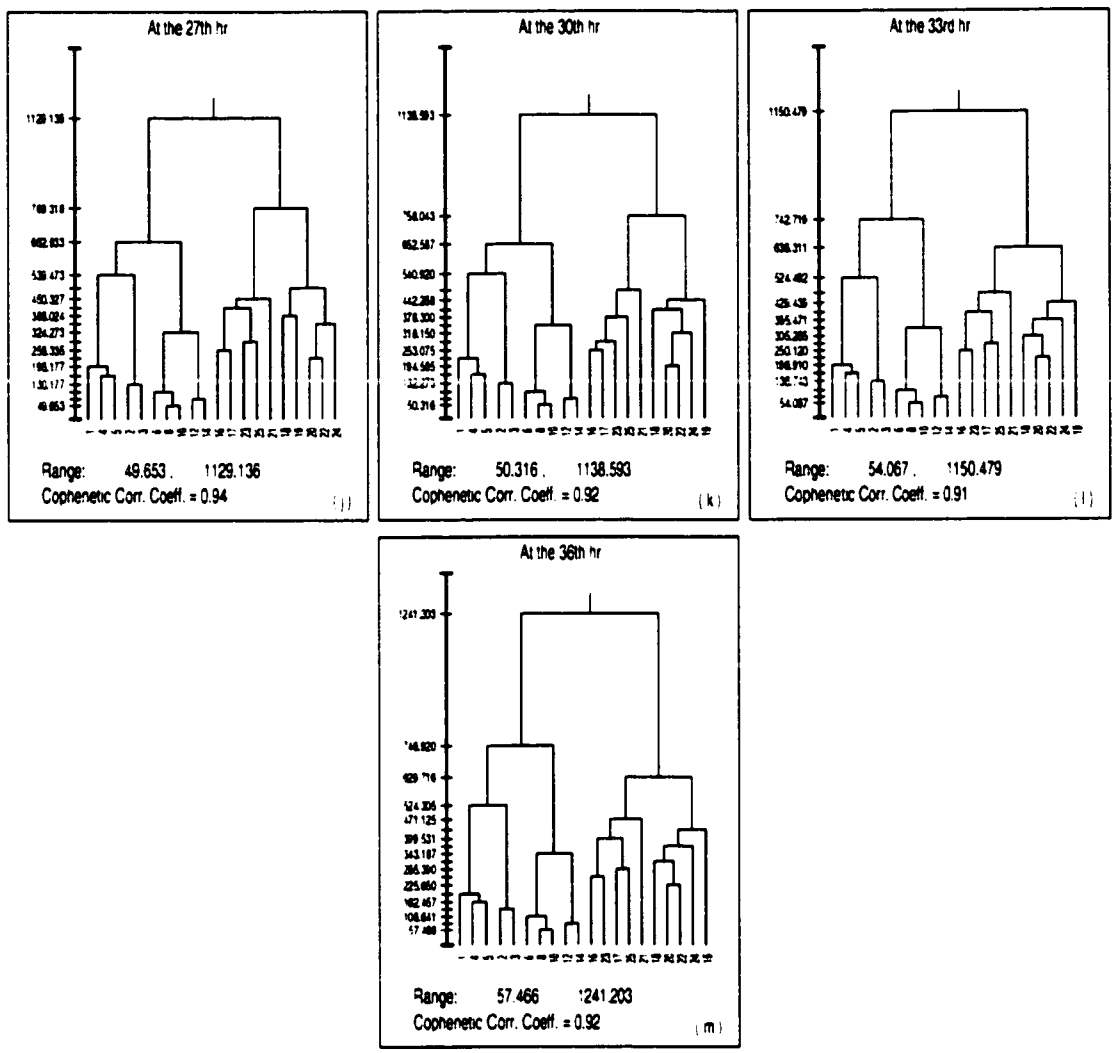


Figure 4-18b: Continued

4.5.2 PCA Based On the Correlation and Euclidean Similarity

In addition to being a tool for clustering, the correlation-based PCA provides a preliminary evaluation of the divergence/condensation among the variables. This is done through the analysis of the eigenvalues and the variances they explain. In this section we present a cross comparison of the variance explained by the first two eigenvalues of the correlation matrices for all fields we study in the SAMEX data sets,

Table 4-12. The results of the similarity- and dissimilarity-based PCA analyses for the 850 mb absolute vorticity are then discussed.

Table 4-12: Comparison of the variance explained by the first two eigenvalues of the correlation matrices for various fields at the initial and finishing time. (var(λ_i) stands for the variance explained by λ_i).

Field	Time	var(λ_1)	Var(λ_2)	var(λ_1)+var(λ_2)
Precipitation	Initial Time	39.26	11.23	50.49
	Finishing Time	20.94	10.41	31.35
Pressure	Initial Time	88.39	7.91	96.30
	Finishing Time	67.14	16.26	83.40
CAPE	Initial Time	80.29	8.24	88.53
	Finishing Time	51.47	13.73	65.20
Height (500 mb)	Initial Time	98.67	0.43	99.10
	Finishing Time	97.70	1.78	99.48
Temperature (2 m)	Initial Time	88.59	6.32	94.91
	Finishing Time	91.21	5.33	96.54
Wind (250 mb)	Initial Time	97.43	1.52	98.95
	Finishing Time	89.64	7.13	96.77
Absolute Vorticity (250 mb)	Initial Time	76.03	7.59	83.62
	Finishing Time	46.22	14.08	60.30
Absolute Vorticity (500 mb)	Initial Time	80.19	7.76	87.95
	Finishing Time	42.00	11.90	53.90
Absolute Vorticity (850 mb)	Initial Time	60.87	18.75	79.62
	Finishing Time	39.66	13.97	53.63
Dewpoint (2 m)	Initial Time	80.19	7.76	87.95
	Finishing Time	42.00	11.90	53.90

This table agrees with some of our CA results in the previous section especially for the height, wind and the temperature fields. As for the height and wind fields, the first eigenvalue at every output time accounts for nearly all of the variance in the data. The correlation-based CA fails to recover the clustering structures for these fields, due to the fact that members are highly correlated. Therefore, all members are placed into one cluster with a very high correlation value. As for the temperature field, the correlation-based CA shows that the height of the clustering trees is decreasing when

approaching the finishing time. Here the first eigenvalue for the temperature field at the finishing time is larger than its counterpart at the initial time. This means that the members get more concentrated on the first dimension as the time passes.

A. 850-mb Absolute Vorticity (ABSVORT850) Field

Using the correlation matrix of the finishing time, the similarity-based PCA is applied for this field. The promax rotation (section 2.3.2 B) of the first 4 PCs leads to a strong simple structure, Table 4-13a. A 69.2% of the total variance is extracted by retaining 4 PCs. By examining the simple structure in Table 4-13a, the clustering structure of the 25 members at the 36th hour is as follow: PC1 clusters 7, 9, 11, 13, and 15; PC2 clusters 16-25; PC3 clusters 1-5; and PC4 clusters 6, 8, 10, 12, and 14. That's, the members are assigned to four clusters each of which represents one model (i.e., [ARPS], [Eta], [RSM], [MM5]). The correlation matrix of the rotated 4 PCs at the 36th hour, Table 4-13b, shows that the 4 clusters are well separated. This structure is the same as the one obtained from the correlation-based CA analysis of this field.

The dissimilarity-based PCA is, then, applied using the Euclidean similarity matrix of the finishing time. The result of this analysis is slightly different. Rotating the first 3 PCs using promax leads to a strong simple structure, Table 4-14a. By examining the simple structure in Table 4-14a, the clustering structure of the 25 members at the 36th hour is as follow: PC1 clusters 1-6, 8, 10, 12 and 14; PC2 clusters 16-25; and PC3 clusters 7, 9, 11, 13 and 15. That is, the members of both ARPS and Eta models are placed into the first cluster, members of RSM model are placed into the second cluster, and members of MM5 formed the third cluster (i.e., [ARPS, Eta], [RSM], [MM5]).

These 3 clusters are well separated as shown by the correlation matrix of the 3 rotated PCs, Table 4-14b. The overall results of PCA and CA analyses using both correlation and Euclidean Similarity measures are very consistent for absolute vorticity (850mb) data.

Table 4-13a : Rotated loadings of the first 4 PCs using promax for the Abs. Vorticity 850mb data at the 36th hour based on the correlation.

n	PC 1	PC 2	PC 3	PC 4
1	0.00	-0.03	**0.95**	0.07
2	0.03	0.00	**0.71**	0.17
3	0.04	-0.06	**0.80**	0.16
4	-0.04	0.02	**0.95**	-0.11
5	-0.06	0.00	**0.96**	-0.08
6	0.01	0.01	0.00	**0.95**
7	**0.92**	0.02	0.00	-0.01
8	0.01	0.01	-0.06	**0.91**
9	**0.79**	-0.05	-0.16	0.20
10	0.02	-0.02	-0.04	**0.97**
11	**0.86**	-0.01	-0.06	0.13
12	-0.05	0.06	0.11	**0.83**
13	**0.70**	0.02	0.18	-0.12
14	0.03	-0.01	0.05	**0.88**
15	**0.89**	0.01	0.03	-0.11
16	0.01	**0.85**	-0.05	0.03
17	0.03	**0.78**	-0.05	-0.06
18	0.05	**0.66**	0.05	-0.07
19	-0.10	**0.79**	-0.03	0.01
20	-0.02	**0.73**	0.01	0.11
21	0.04	**0.65**	-0.03	0.03
22	-0.12	**0.67**	-0.03	0.11
23	0.07	**0.63**	0.10	0.06
24	0.04	**0.69**	0.08	-0.12
25	0.03	**0.79**	-0.04	-0.06

Table 4-13b: the correlation matrix of the 4 rotated PCs (correlation-based promax) for the Abs. Vorticity 850mb data set at the 36th hour.

	1	2	3	4
1	1.00	0.26	0.38	0.52
2	0.26	1.00	0.40	0.42
3	0.38	0.40	1.00	0.54
4	0.52	0.42	0.54	1.00

Table 4-14a: Rotated loadings of the first 3 PCs using promax for the Abs. Vorticity 850mb data at the 36th hour based on the Euclidean Similarity.

n	PC 1	PC 2	PC 3
1	**0.98**	-0.09	0.13
2	**0.82**	-0.04	0.03
3	**0.90**	-0.11	0.07
4	**0.94**	-0.15	0.23
5	**0.95**	-0.13	0.22
6	**0.60**	0.19	-0.28
7	0.15	-0.07	**-0.72**
8	**0.54**	0.19	-0.28
9	0.15	-0.05	**-0.70**
10	**0.58**	0.18	-0.30
11	0.19	-0.03	**-0.71**
12	**0.61**	0.20	-0.22
13	0.26	-0.20	**-0.45**
14	**0.60**	0.17	-0.28
15	0.17	-0.11	**-0.65**
16	0.06	**0.70**	0.01
17	-0.05	**0.60**	0.07
18	0.10	**0.40**	0.10
19	0.12	**0.61**	0.07
20	0.18	**0.58**	0.00
21	0.03	**0.39**	0.09
22	0.15	**0.49**	0.08
23	0.20	**0.46**	0.00
24	0.08	**0.42**	0.14
25	-0.02	**0.62**	0.06

Table 4-14b: the correlation matrix of the 3 rotated PCs (EucSimi-based promax) for the Abs. Vorticity 850mb data set at the 36th hour.

	1	2	3
1	1.00	0.47	-0.43
2	0.47	1.00	-0.27
3	-0.43	-0.27	1.00

4.6 Summary and Concluding remarks

The goal of the work in this chapter is to explore the algorithmis clustering of multi-model, short-term ensemble forecasting. The multi-model ensemble data are from SAMEX, an experiment coordinated by CAPS at the University of Oklahoma and involving scientists at NCEP and NSSL. The full ensemble consists of 25 members

made from 4 different numerical weather prediction models: 5 members from the ARPS, 5 members from the NCEP Eta model, 5 members from the NCEP RSM, and 10 members from MM5 run at NSSL. Each model member extends out to 36 hour beginning from 0000 UTC 29 May 1998. The aim is to analyze the trajectories from these members and see how they cluster during evolution for the various fields (Alhamed and Lakshmivarahan 2000).

Since two of the models (Eta, RSM) use the breeding of growing modes technique to initialize their forecasts, and one of the models (MM5) is configured with varying model physical parameterization schemes, the cluster analyses allow us to investigate the roles played by both model initial condition and model physics uncertainties. Two statistical techniques are used for these analyses, CA and PCA, and three different measures are used to define the similarity between the forecast fields, namely correlation, Euclidean similarity, and Euclidean distance. Based on the analyses of the 3-hourly accumulated precipitation (ACCPPT) field, it is observed that the members of each model have tendencies to stay together and build their own cluster. The correlation-based analyses support this finding very strongly. According to these analyses, the members of one model are shown to be coherent and members from different models are isolated. On the other hand, the distance based cluster analysis agrees with this tendency but to a lesser extent. Admittedly, the distance based CA is able to reveal inter-models clusters, indicating similarity between members generated by different models. The existence of similarity across models is also found when using a nonhierarchical clustering algorithm such as the *K*mean and nucleated agglomerative (NA) algorithms on these precipitation data.

When other forecast fields are examined, such as 250-mb wind speed, 500-mb geopotential height, CAPE, and mean sea-level pressure, the tendency for the forecast members to cluster by model is found. Neither similarity-based nor dissimilarity-based CA or PCA find any inter-model cluster to exist over more than a few model output times. The one strong tendency seen across all forecast times and fields is for the forecasts to cluster by the model used to produce the forecasts. This indicates that the models are more similar to themselves than to the initialization methods used. It is observed that by the 36th hour the model results cluster with the model first, and then with the institution (i.e., Eta and RSM cluster next with each other, before clustering with ARPS and MM5). Even though the Eta and RSM are using the breeding of growing modes technique to generate the perturbed initial and boundary conditions, very little indication of the bred modes clustering together is seen (mode 1 from Eta, RSM clustering together, then mode 2 from Eta, RSM clustering together, etc.). The models all cluster together first, showing the very different model climates that appear quite quickly in the model forecasts (David Stensrud, 2000, personal communication).

The height of the dendrograms can be used to evaluate how much the forecasts are alike, with taller connection points on the dendrograms indicating that the forecasts are less alike. The subjective analysis of all clustering dendrograms indicates that the MM5 forecasts, that include model physics differences and use less rapidly growing initial conditions than the bred modes, are the most dissimilar of all the forecasts. This is particularly true of the precipitation forecasts, but also is evident in the other fields. Model physics differences, even when used within the context of a single model, provide for a substantial divergence in the solutions. However, even more important is

the use of different models in order to produce divergence in the forecast solutions. Even when using bred modes, both the Eta model and RSM cluster with themselves first, and then they cluster together before clustering with either the ARPS or MM5. The bred mode forecasts are not outliers in the sense that they do not span the range of the forecasts on this day. Indeed, the Eta and RSM often are the first models to cluster together. The results in this chapter strongly suggest that model differences provide much of the divergence in the solutions from this case and that the techniques used to vary the initial conditions are much less important. This conclusion is supported by the subjective analyses of the model forecast fields from selected time periods, which also indicates that the models are very similar to themselves and that it is the fields from different models that have very different structures (David Stensrud, 2000, personal communication).

These results further highlight the important role played by model physics in determining the resulting forecasts. While perturbing a single model does improve the forecast divergence, as shown here and by Buizza et al. (1999) and Stensrud et al. (2000), it is clear that using totally different models is likely a more fruitful approach. We hope that the use of completely different models in short-range ensemble forecasting will continue to be explored vigorously as it enhances the ensemble variability, which is one of the remaining concerns with present operational ensemble forecast systems.

CHAPTER 5

VALIDATION: THEORY AND ALGORITHMS

5.1 Introduction

Jain and Dubes (1988) define cluster validation as follows:

> "Cluster validation refers to procedures that evaluate the results of cluster analysis in a quantitative and objective fashion".

Numerous techniques and approaches have been proposed in the literature of cluster analysis to address the problem of validating the results of the clustering methods. Many years ago Hubert and Arabie (1985) wrote a paper related to this topic and began this paper by saying " We will not try to review this literature comprehensively since that task would require the length of a monograph". Since it is not possible to comprehensively review this area of clustering, we have instead chosen to discuss some of the common approaches that likely to be applicable to our problem. We need to bring to the reader's attention that these techniques were developed in context other than meteorology so our presentation in this chapter is from mathematical and algorithmic perspective. In the next chapter (chapter 6) the applicability and suitability of these techniques to the problem in meteorology are discussed.

In fact, the topic of cluster validation is visited in some earlier chapters. In chapter 2, the notion of Cophenetic Correlation is introduced. Cophenetic correlation is one approach of validating clustering results using the data itself. It measures how well the clustering tree fits the data. This technique is used throughout chapter 3 and 4. The Cophenetic correlation coefficient is computed for every clustering dendrogram. The

results of using this coefficient are discussed in the following chapter. Subjective validation from a meteorological perspective is introduced and used in chapter 4. In this validation technique, an expert meteorologist validates the results through comparing the horizontal contour plots of the individual members of the ensemble.

Three different strategies are identified for validating clustering solutions (Jain and Dubes 1988):

- External: compare a clustering solution to an a priori structure.
- Internal: evaluate the match between the clustering solution and the data internally. That's using only the data.
- Relative: compare several clustering solutions of the same set of objects.

Hypothesis testing is one approach for validating the clustering structure found by the clustering algorithm. This approach follows the standard statistical framework for hypothesis testing. In this framework, a null hypothesis is first defined, which expresses the randomness in the data set. The randomness refers to the absence of clustering structure. Second, an index, or statistic is selected to test the null hypothesis. This index must be sensitive to the existence of the clustering structure in the data. The distribution of the index under the null hypothesis is then computed or estimated. The distribution is used to find a threshold that decides the computed value of the index against the null hypothesis. The rejection of the null hypothesis indicates the presence of the clustering structure in the data. Hence, the partition found by the algorithm is valid.

The most common null hypotheses in cluster validation are described as follows (Jain and Dubes 1988).

a) Random graph hypothesis:

H_0: All nxn rank order resemblance matrices are equally likely.

b) Random label hypothesis:

H_0: All permutations of the label on n objects are equally likely.

c) Random position hypothesis:

H_0. All sets of n locations in some region of d-dimensional space are equally likely.

The knowledge of the scale of the data is significant when selecting a null hypothesis. Not all hypotheses are appropriate for all data scales. The random graph hypothesis is for data on ordinal scale whereas the random position hypothesis is for ratio scale. The random label hypothesis can be applied for all scales when a clustering structure is imposed a priori. The main difficulty in applying this test is the computation of the distribution of the statistic under the chosen null hypothesis (Milligan 1996). One way for computing the distribution is through direct calculation. For example, the distribution for the random label hypothesis is the set of all $n!$ permutations of labels on the n objects under study. That's for 10 objects data set, 3,628,800 values must be computed and for 25 objects data set, over 155×10^{23} values need to be computed in order to establish the distribution. Another way is to estimate the distribution through the use of Monte Carlo analysis (Jain and Dubes 1988). The complexity and the expense of computing the distribution are so high that this approach of validation has not received great attention from applied researchers (Milligan 1996).

Other approaches for clustering validation include determining the correct number of clusters, replication analysis, and measuring the agreement between different

partitions of the same data which are discussed in the following sections. Applications of these techniques are conducted on SAMEX data and the results of these applications are discussed in the next chapter (Chapter 6)

5.2 Determining the number of clusters

The problem of determining the "correct" number of clusters in the clustering solution is a significant problem in the cluster analysis. The clustering algorithms do not determine the correct number of clusters as a part of solution. They are not designed to do so. Instead, the hierarchical algorithms produce a nested hierarchy of clusters or the user must provide the number of clusters for the partitional algorithms (Milligan 1996). For this reason many algorithms and indexes have been proposed for computing the correct number of clusters (Milligan 1981, Milligan and Cooper 1985, Pal and Biswas 1997, and Bezdek and Pal 1998). We now present some of the most common techniques and highlight some recent developments on some of these indexes.

5.2.1 Davies-Bouldin Index (*DB*)

The rationale of the Davies-Bouldin (*DB*) index is that for a good partition the within-cluster scatter should be small and the between-cluster spread should be large. *DB* index is the function of the ratio of the sum of the within-cluster scatter to the between-cluster scatter (Davies and Bouldin 1979).

Let $W_i = \left(\frac{1}{n_i} \sum_{x_j \in C_i} \left\| x_j - m_i \right\|_2^2 \right)^{\frac{1}{2}}$ be the within-cluster scatter for the i^{th} cluster,

where n_i is the number of objects in the cluster and m_i is the center of the cluster (2.23).

Let $B_{ij} = \left\| m_i - m_j \right\|_2$ be the between-cluster scatter between the i^{th} and the j^{th} clusters.

For a given partition of n objects into k clusters, the within-to-between cluster spread for pairs of clusters i and j is defined as

$$R_{ij} = \frac{w_i - w_j}{B_{ij}} . \tag{5.1}$$

The index for the i^{th} cluster is

$$R_i = \max_{i \neq j} \{R_{ij}\} \tag{5.2}$$

The Davies-Bouldin (DB) index for the k-cluster partition is defined as follow.

$$DB = \frac{1}{k} \sum_{i=1}^{k} R_i , \qquad \text{for } k>1 \tag{5.3}$$

The index is 0 when each object is in an individual cluster and it is not defined when all objects are placed in one cluster.

Geometrically, one seeks clustering solution that minimizes the within-cluster scatter and maximizes the between-cluster spread. Thus, the smaller value of the DB index indicates better solution. Therefore, the number of clusters k that minimizes the DB index is taken as the optimal number of clusters. For well-clustered data, the index is expected to decrease monotonically as the number of clusters increases until the optimal number of clusters is obtained (Bezdek and Pal 1998).

Example 5.1:

This example demonstrates the use of DB index and other indexes later in this section to determine the correct number of clusters for 10 points in 2-dimensional space. The data matrix X for this example is given as follows:

$$X = \begin{bmatrix} 82 & 15 & 79 & 40 & 13 & 46 & 77 & 18 & 42 & 16 \\ 33 & 45 & 40 & 92 & 39 & 89 & 35 & 42 & 87 & 41 \end{bmatrix}$$

Figure 5-1 shows 2-dimensional representation of the 10 points in X, and Figure 5-2 shows the clustering dendrogram of the 10 points using Ward's method based on the Euclidean distance.

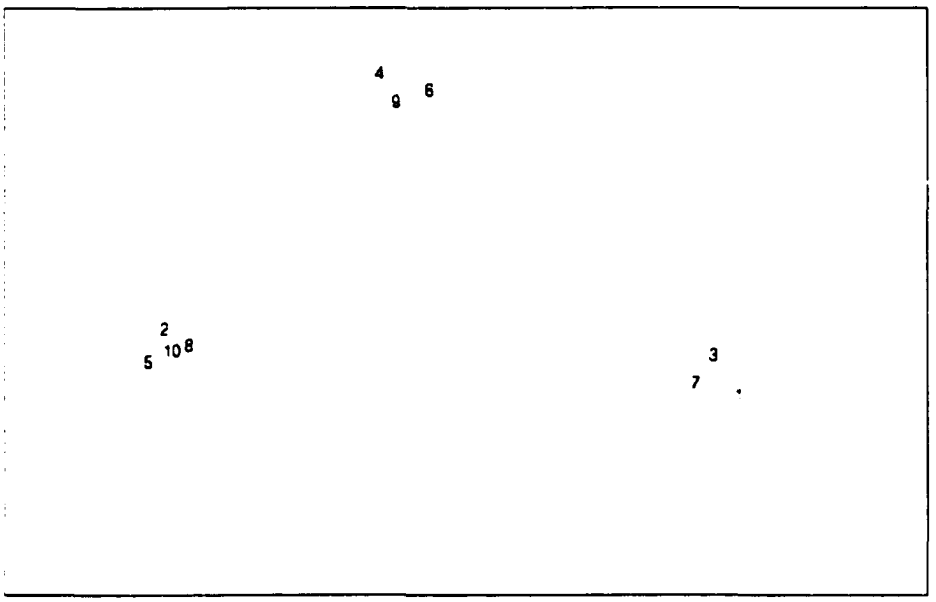


Figure 5-1: 2-dimensional representation of the 10 points in example 5.1.

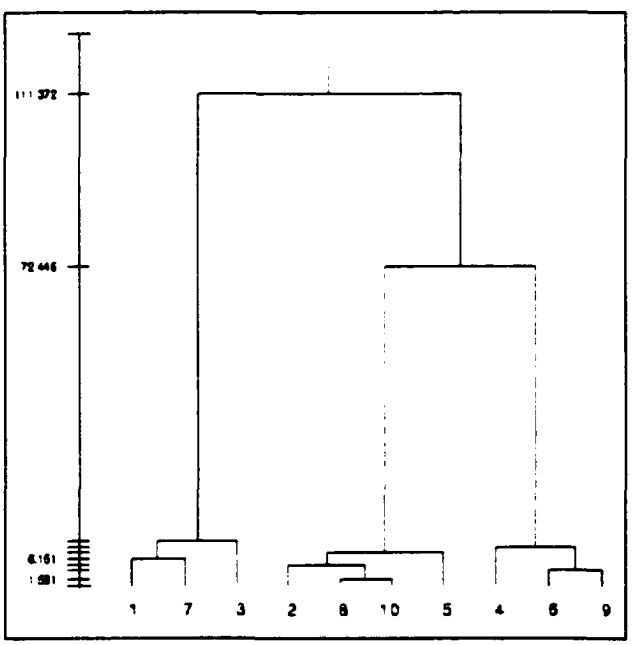


Figure 5-2: Ward's dendrogram for the 10 points in example 5.1.

Once the clustering solution is obtained, the *DB* index is computed for different number of clusters as follows:

k	2	3	4	5	6
DB	0.53	0.11	0.28	0.36	0.42

Recall that the number of clusters *k* that minimizes the *DB* index is taken as the optimal number of clusters. Since the minimum value for the *DB* index is found when k=3 as shown above, then the optimal number of clusters for the 10 points in this example is 3.

5.2.2 Modified Hubert's Index ($MH\Gamma$)

The modified Hubert's index $MH\Gamma$ is a modification of the Hubert's Γ statistic that used to assess the fit between data and a priori structure (Jain and Dubes 1988). In its raw form, Hubert's index is defined as the point serial correlation between two matrices. The first matrix is the *nxn* observed dissimilarity matrix, P_{nxn} (assumed to be the Euclidean distance matrix). The second matrix Q_{nxn} is defined in terms of any partition imposed by the clustering algorithm as follow.

$$q_{ij} = \begin{cases} 0 & \text{if objects } i \text{ and } j \text{ are in the same cluster.} \\ 1 & \text{otherwise.} \end{cases}$$

This matrix is called the "model" matrix (Jain and Dubes 1988). When both P and Q matrices are symmetric, Hubert's index can be written in its raw form as follow.

$$\Gamma = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_{ij} q_{ij} \tag{5.4}$$

The normalized version of the Hubert's index $\hat{\Gamma}$ is the sample correlation coefficient between the entries of P and Q matrices.

$$\hat{\Gamma} = \frac{\frac{1}{M} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (p_{ij} - \bar{p})(q_{ij} - \bar{q})}{\sigma_p \quad \sigma_q}, \qquad -1 \le \hat{\Gamma} \le 1 \tag{5.5}$$

where $M = \frac{n(n-1)}{2}$ and $\bar{p}$ and $\bar{q}$ are the sample means, and σ_p and σ_q are the sample standard deviations:

$$\bar{p} = \frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} p_{ij}, \qquad \bar{q} = \frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} q_{ij},$$

$$\sigma_p = \left[\frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} p_{ij}^2 - \bar{p}^2\right]^{\frac{1}{2}}, \qquad \sigma_q = \left[\frac{1}{M}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} q_{ij}^2 - \bar{q}^2\right]^{\frac{1}{2}}.$$

The Hubert's index is usually used to compute the degree of linear correspondence between the entries of the observed dissimilarity matrix P and the model matrix Q. A large positive value of Γ (or close to 1 in the case of $\hat{\Gamma}$) indicates that the two matrices are linearly correlated. In the cluster validation, this index is used to test whether the association between P and Q is unusually large under the random label hypothesis.

In such situation, the distribution of Γ or $\hat{\Gamma}$ under the null hypothesis must be known. One way to find the distribution is to compute the index for all $n!$ permutations and then use its histogram. Another way is to use Monte Carlo method to estimate the distribution. To avoid the complexities associated with computing the distribution of Γ or $\hat{\Gamma}$, the modified Hubert's index $MH\Gamma$ has been proposed for cluster validation. The difference between the Hubert's index and its modification is in the definition of the model matrix. In the modified form, the model matrix is obtained by using the Euclidean distance between the centers of the clusters to which the objects belong.

Let $L(i) = c$ if the i^{th} object is in the c^{th} cluster.

Let $d_{j,k} = \|m_j - m_k\|_2$ be the Euclidean distance between the centers of j^{th} and k^{th}

cluster.

Define the model matrix Q as follow.

$$q_{ij} = d_{L(i),L(j)}, \quad 1 \le i, j \le n \tag{5.6}$$

Using this new definition of the model matrix Q, the raw $MH\Gamma$ is computed by the formula (5.4) and formula (5.5) is used for the normalized index $MH\hat{\Gamma}$. When each object is in an individual cluster, the index will be 1 and it is not defined when there is only one cluster.

From computational experience (Jain and Dubes 1988, and Bezdek and Pal 1998), it is known that this index tends to increase with the increase of the number of clusters until the optimal number of clusters is found. Therefore, for well-clustered data a significant knee is found in a plot of the values of the index against the number of clusters *k*. The sharp knee in the plot identifies the correct number of clusters for the clustering solution.

Example 5.2:

This example demonstrates the use of $MH\hat{\Gamma}$ index to determine the correct number of clusters for the clustering solution of example 5.1. The values of this index for different number of clusters are as follows:

K	2	3	4	5	6
$MH\hat{\Gamma}$	0.68	0.99	0.99	1.00	1.00

The idea of $MH\hat{\Gamma}$ index is to plot the values of the index against the number of clusters (*k*) and look for a significant knee in the plot which identifies the correct number of clusters for the clustering solution. The plot of this index against *k* (Figure 5-3) indicates that the correct number of clusters for the 10 points is again 3.

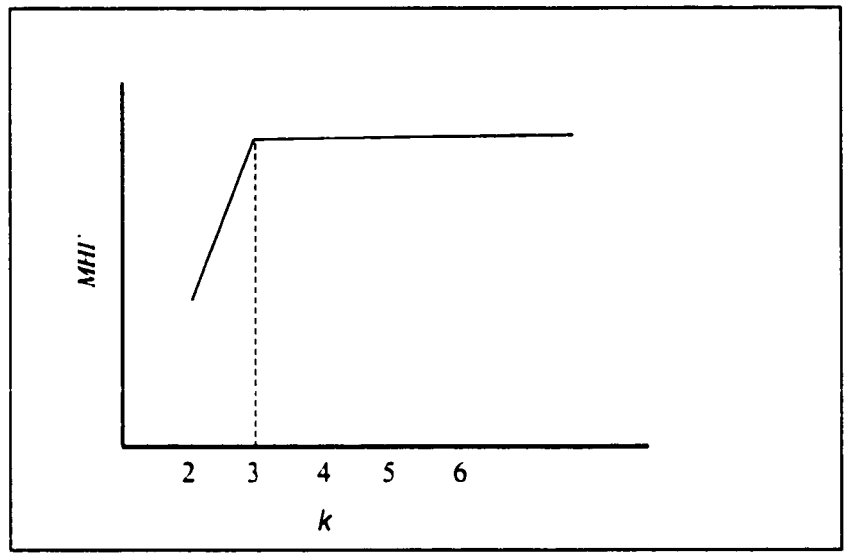


Figure 5-3: Plot of $MH\hat{I}$ against the number of cluster for example 5.2.

5.2.3 Dunn's Index (*DU*) and its Generalizations

Dunn's index (*DU*) is based on the same geometrical rationale as the *DB* index (section 5.2.1). That's the good partition is the one that has large between-cluster separation and small within-cluster scatter. This index uses the notions of *diameter* Δ and *set distance* δ to define the compactness and separation of clusters (Bezdek and Pal 1998).

Let $\Delta(C_i) = \max_{r,s \in C_i} \{d_{r,s}\}$ be the diameter of the i^{th} cluster.

Let $\delta(C_i, C_j) = \min_{\substack{r \in C_i \\ s \in C_j}} \{d_{r,s}\}$ be the set distance between the i^{th} and j^{th} clusters.

For a given partition $C_1 \cup \ldots C_i \cup \ldots C_k = X$, the Dunn's index (*DU*) is defined as follow.

$$DU = \min_{1 \le i \le k} \left\{ \min_{\substack{1 \le j \le k \\ j \ne i}} \left\{ \frac{\delta(C_i, C_j)}{\max_{1 \le l \le k} \{\Delta(C_l)\}} \right\} \right\} \tag{5.7}$$

The index is not defined when there is only one cluster or when each object is placed in an individual cluster ($k=n$).

Since the set distance represents the between-cluster separation and the diameter represents the within-cluster scatter, then the larger value of the *DU* index corresponds to good clustering solution. Therefore, the number of clusters k that maximizes the index is taken as the optimal number of clusters.

Dunn's index has received a comprehensive review and development by Bezdek and Pal (1998) and Pal and Biswas (1997). It was shown that both diameter and set distance are severely sensitive to changes in the clustering structure. These quantities may be affected by one or two noisy points (Bezdek and Pal 1998). In there paper, the authors provide generalizations for Dunn's index which are not as sensitive to the noisy points as the original index. They used the formula (5.7) as the general paradigm and proposed several new definitions for the diameter and set distance for the purpose of defining new cluster validity indexes. One index that performs very well in their study is the one that uses the average distance as the set distance between two clusters.

$$\delta_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{\substack{x \in C_i \\ y \in C_j}} d_{x,y} \tag{5.8}$$

where n_i is the number of objects in the i^{th} cluster.

The Bezdek and Pal's generalized Dunn's index (*BPDU*) can be written using (5.7) as follow.

$$BPDU = \min_{1 \le i \le k} \left\{ \min_{\substack{1 \le j \le k \\ j \ne i}} \left\{ \frac{\delta_{avg}(C_i, C_j)}{\max_{1 \le l \le k} \{\Delta(C_l)\}} \right\} \right\} \tag{5.9}$$

Since this new index is based on the same geometrical rationale as the original index, the large value of *BPDU* corresponds to good partition. Hence, the number of clusters k that maximizes the index is taken as the optimal number of clusters.

All indexes discussed so far are suitable for hyperspherical type clusters only. For chain type of clusters, other set of indexes is proposed by Pal and Biswas (1997). These new indexes are generalized form of Dunn's index and they used its original formula (5.7) as the general paradigm. Pal and Biswas (1997) used concepts from graph theory, such as minimal spanning tree and Gabriel graph, to define the diameter of clusters. Using the new graph theoretic definitions of diameter, the authors generalized the Davies-Bouldin index and Dunn's index and proposed new set of indexes. Their paper shows that the Gabriel graph-based Dunn's index (DU_{GG}) outperforms all other indexes, original and the generalized indexes as well. Moreover, this index is shown to be suitable for both chain and spherical types clusters. The Pal and Biswas's generalized Dunn's index based on the Gabriel graph DU_{GG} is now introduced.

Let $GG_i=(V_i,E_i)$ be the Gabriel Graph (GG_i), the points p_i and p_j are connected if

$$\|p_i - p_j\|_2^2 < \|p_i - p_k\|_2^2 + \|p_k - p_j\|_2^2 \qquad \forall k, k \neq i, k \neq j.$$

where $\|p_i - p_j\|_2$ is the Euclidean distance between p_i and p_j. Given the partition $C_1 \cup \ldots C_i \cup \ldots C_k = X$, the diameter of a cluster is defined in terms of edges of its Gabriel graph.

Let $E_i=\{e_{ij} \mid 1 \leq j \leq |E_i|\}$ be the set of edges of GG_i computed on cluster C_i. The diameter Δ_i^{GG} of the cluster C_i is defined as follow.

$$\Delta_i^{GG} = \max_j \{ e_{ij}, j = 1,2,\ldots,|E_i| \} \tag{5.10}$$

The set distance (or separation) between two clusters C_i and C_j is defined as the Euclidean distance between the centers of the two clusters.

$$\delta_{GG}(C_i, C_j) = \|m_i - m_j\|_2 \tag{5.11}$$

The DU_{GG} index is written using (5.7) as follow.

$$DU_{GG} = \min_{1\le i\le k}\left\{\min_{\substack{1\le j\le k\\ j\neq i}}\left\{\frac{\delta_{GG}(C_i, C_j)}{\max_{1\le l\le k}\{\Delta_l^{GG}\}}\right\}\right\} \tag{5.12}$$

For a good partition, the separation (numerator in 5.12) should be large and the maximum diameter (denominator) should be smaller, hence the larger value of the DU_{GG} index is the better cluster solution. Therefore, the number of clusters k that maximizes the DU_{GG} index is taken as the correct number of clusters.

Example 5.3:

This example demonstrates the use of three indexes discussed in this subsection, namely *DU*, *BPDU*, and DU_{GG} indexes to determine the correct number of clusters for the clustering solution of example 5.1. The values of these three indexes are computed for different number of clusters as follows:

k	2	3	4	5	6
DU	0.99	6.56	0.80	0.85	0.67
BPDU	1.08	7.21	0.97	0.96	0.98
DU_{GG}	1.17	8.17	0.90	1.05	0.92

Recall that the number of clusters k that maximizes the values of these three indexes is taken as the optimal number of clusters. As shown on the table above, the three indexes have their maximum values when k=3. Hence, the optimal number of clusters for the 10 points in example 5.1 is again 3.

5.3 Replication analysis

Replication analysis is one approach of validation that measures the stability of the resultant clustering structure of a clustering method. It was developed by McIntyre and Blashfield (1980) and by Morey, Blashfield, and Skinner (1983). McIntyre and Blashfield (1980) defined this approach as the process of estimating the degree of

replicability (or stability) of a clustering solution across a series of data set. The rationale behind this technique is that a good clustering structure is stable across multiple random samples. Milligan (1996) describes the major steps of replication analysis as follows.

1. Divide the data set into two random samples, A and B.
2. Perform a cluster analysis on sample A and determining its clustering solution.
3. Compute the center of each cluster obtained in step 2.
4. Compute the Euclidean distance between each object in B and all cluster centers computed in step 3. Assign each object in sample B to the nearest center. This assignment known as the Nearest Centroid Assignment rule (NC Rule). This step when finished generates a clustering solution to sample B based on NC Rule.
5. Perform cluster analysis on sample B and determine its clustering solution. Sample B is subjected to the same cluster analysis as used for sample A. That is, the decision regarding the normalization, resemblance measure, clustering method, and the number of clusters must be the same.
6. Compute the agreement between the two clustering solutions of the sample B. That's the agreement between the clustering based on NC Rule and the cluster analysis of the sample B.

The agreement between two clustering structures is computed using a Measure of Agreement such as Rand index or Jaccard index, which will be discussed in a subsequent section.

One of the recent developments of the replication analysis was done by Breckenridge (1989). He performed a study on this technique using three different assignment rules to implement step 4. The rules he used are: nearest centroid (NC), nearest neighbor (NN), and quadratic discriminate analysis (NQ). He shows that replication using nearest neighbor (NN) results in superior goodness-of-fir and it outperforms other rules. The idea of nearest neighbor assignment rule is to compute the Euclidean distance between every object in sample B and all objects in sample A and then assign each object in sample B to the cluster where in which its nearest neighbor from sample A resides.

5.4 Measures of Agreement between Partitions

The problem of assessing the degree for agreement between two different partitions of n objects has attracted a great attention and interest in the clustering literature. Hubert and Arabie (1985) and Milligan and Cooper (1985, 1986) cited many works in this area. They also added significant development in defining and studying the performance of several indexes for comparing partitions. All such indexes are derived in same manner using the contingency table.

Given two different partitions U and V for the same n objects:

$$U=\{u_1, u_2, \ldots, u_R\}$$

$$V=\{v_1, v_2, \ldots, v_C\}$$

where R and C represent the number of clusters in U and V respectively.

Let n_{ij} be the number of objects in both u_i and v_j, the RxC contingency table, Table 5-1, can be constructed using the information on cluster overlap between the U and V partitions.

Table 5-1: Contingency Table.

	v_1	v_2	.....	v_C	
u_1	n_{11}	n_{12}	.....	n_{1C}	$n_{1.}$
u_2	n_{21}	n_{22}	.....	n_{2C}	$n_{2.}$
.	.				.
.	.				.
.	.				.
u_R	n_{R1}	n_{R2}	.....	n_{RC}	$n_{R.}$
	$n_{.1}$	$n_{.2}$	.....	$n_{.C}$	n

The term $n_{i.}$ in Tabe 5-1 refers to the sum of the i^{th} row or the number of objects in the u_i, and $\sum_{i=1}^{R} n_{i.} = n$. The term $n_{.j}$ is the number of objects in v_j and $\sum_{j=1}^{C} n_{.j} = n$.

The agreement indexes assess the correspondence between the two partitions U and V based on the types of object pairs in the contingency table. In fact there are four different types among the $\frac{1}{2}n(n-1)$ distinct pairs.

Type 1 (T_1): number of pairs of objects that are placed in the same cluster in both partitions.

Type 2 (T_2): number of pairs that are placed in different clusters in partition U but in the same cluster in V.

Type 3 (T_3): number of pairs that are placed in the same cluster in partition U but in different clusters in V.

Type 4 (T_4): number of pairs that are placed in different clusters in both partitions.

Hubert and Arabie (1985) provided explicit formula for each type as a function of n_{ij}, $n_{i.}$, $n_{.j}$, and n.

$$T_1 = \sum_{i=1}^{R}\sum_{j=1}^{C}\binom{n_{ij}}{2} = \frac{1}{2}\sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}(n_{ij}-1) \qquad (5.13)$$

$$T_2 = \sum_{j=1}^{C}\binom{n_{\cdot j}}{2} - \sum_{i=1}^{R}\sum_{j=1}^{C}\binom{n_{ij}}{2} = \frac{1}{2}\left[\sum_{j=1}^{C} n_{\cdot j}^2 - \sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2\right] \tag{5.14}$$

$$T_3 = \sum_{i=1}^{R}\binom{n_{i\cdot}}{2} - \sum_{i=1}^{R}\sum_{j=1}^{C}\binom{n_{ij}}{2} = \frac{1}{2}\left[\sum_{i=1}^{R} n_{i\cdot}^2 - \sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2\right] \tag{5.15}$$

$$\left.\begin{aligned} T_4 &= \binom{n}{2} - [T_1 + T_2 + T_3] \\ &= \frac{1}{2}\left[n^2 + \sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2 - \sum_{i=1}^{R} n_{i\cdot}^2 - \sum_{j=1}^{C} n_{\cdot j}^2\right] \end{aligned}\right\} \tag{5.16}$$

(T_1+T_4) represents the agreement between U and V partitions and (T_2+T_3) represents the disagreement between the two partitions.

Using T_1, T_2, T_3, and T_4 types, several indexes are proposed in the clustering literature. Two of the most well-known agreement measures are Rand and Jaccard. The formulas for these two indexes are given below (Hubert and Arabie 1985 and Jain and Dubes 1988).

Rand Index:

$$(T_1 + T_4)\Big/\binom{n}{2} = 1 + \left[\sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2 - \frac{1}{2}\left(\sum_{i=1}^{R} n_{i\cdot}^2 + \sum_{j=1}^{C} n_{\cdot j}^2\right)\right]\Big/\binom{n}{2} \tag{5.17}$$

Jaccard Index:

$$\frac{T_1}{T_1 + T_2 + T_3} = \frac{\sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2 - \sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}}{\sum_{i=1}^{R} n_{i\cdot}^2 + \sum_{j=1}^{C} n_{\cdot j}^2 - \sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2 - \sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}} \tag{5.18}$$

Note that $\sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij} = n$, so the Jaccard index can be rewritten in simplified form as follows.

$$\frac{T_1}{T_1+T_2+T_3}=\frac{\sum_{i=1}^{R}\sum_{j=1}^{C}n_{ij}^2-n}{\sum_{i=1}^{R}n_{i.}^2+\sum_{j=1}^{C}n_{.j}^2-\sum_{i=1}^{R}\sum_{j=1}^{C}n_{ij}^2-n} \tag{5.19}$$

The Rand and Jaccard indexes have value between 0 and 1 (Jain and Dubes 1988). The maximum value of 1 indicates that the two compared clusters are identical.

5.5 Summary and Concluding remarks

In this chapter, several approaches for validating clustering solution are reviewed. First approach is concerned with finding the correct number of clusters in the data under study. Determining the correct number of clusters is significant problem in cluster analysis since the clustering algorithms do not compute it. This number is either computed independently from the clustering algorithm using validation index or given by the user. Five of the common indexes to determining the correct number of clusters are reviewed. Two of which are recently developed. The second approach discussed in this chapter is the replication analysis which is used to measure the stability of the clustering solution across multiple samples of the data being clustered. Another validation technique concerns with comparing the clustering solution produced by different clustering algorithms for the same data. Two well-known indexes for comparing partitions are also reviewed. The applications of these techniques for validation are conducted for the SAMEX data in chapter 6.

CHAPTER 6

APPLICATIONS OF CLUSTER VALIDATION ON SAMEX DATA

6.1 Introduction

Once a clustering solution is found, the analysts are faced with several questions. Can we believe the found solution and use it? How many clusters are in the data? Are the clusters stable? Are the solutions found by different algorithms the same? Cluster validation attempts to answer these questions. In chapter 5, the theory and some of common algorithms for validating the resultant clustering solution are discussed. This chapter presents the application of several approaches of validation on the SAMEX data. In chapter 4, ten field variables on the SAMEX data are cluster analyzed. For each field, more than one clustering solutions are found using different algorithms. So this chapter is concerned with validating these solutions.

Two validation methods have been applied in chapter 4. First, the match between the clustering tree and the data themselves is evaluated using cophenetic correlation (section 2.2.1). The cophenetic correlation coefficient (2.22) is computed for every clustering tree in this research and printed out along with the tree. The closer value of this coefficient to 1.0, the better match between the clustering trees and the data themselves. Therefore, the clustering tree fit the data clusters very well. Second, an expert meteorologist subjectively validated the clustering solutions in chapter 4. The horizontal contour plots of the individual members for various fields at different times were constructed. The expert was asked to validate the clustering solution found by

Ward's method by examining these plots. After intense examination, the expert concluded that in almost all cases clustering solutions are meteorologically meaningful.

First approach we apply for validation in this chapter is the determination of the correct number of clusters. The hierarchical cluster analysis algorithm produces a nested structure or a hierarchy of clusters. Further information is needed in order to identify the number of clusters in the data. The nonhierarchical cluster analysis on the other hand require the number of clusters to be specified by the user, hence, specifying incorrect number may lead to incorrect partition. Recall from chapter 5 that enormous number of indexes for determing the number of clusters has been proposed in the literature and each of which is based on different rationale. It was found by experience that no single index is likely to be consistent across different clustering algorithms and data sets. The popular approach to overcome this problem is to apply many indexes and use the majority rule. If the majority of applied indexes agree on certain value for the number of clusters, k, then this values is chosen as the correct number of clusters for the data under study (Bezdek and Pal 1998). Our approach in determining the correct number of clusters is to apply the five indexes discussed in section 5.2 and then use the majority rule. These indexes are: Davies-Bouldin DB (5.3), Modified Hubert's $MH\hat{\Gamma}$ (5.5), Dunn's DU (5.7), Bezdek and Pal's generalized Dunn's $BPDU$ (5.9), and Gabriel Graph-based Dunn's DU_{GG} (5.12).

Second approach used to validate the clustering results is the replication analysis (section 5.3). This approach validates the stability of the clustering solution found by the clustering algorithm. This analysis is implemented for each data set as described in section 5.3. Both nearest centroid (NC) and nearest neighbor (NN) are used as the

assignment rules (section 5.3), and Rand index (section 5.4) is used as the agreement measures.

We extended this approach to study the sensitivity of the resultant clustering solution across multiple samples of the same data. The two independent solutions obtained from cluster analyzing the two samples are compared to each other and to the Ward's solutions obtained for the full data (chapter 4). If the three partitions agree well with each other then the solution being found is insensitive across the data themselves.

The third approach to validate the clustering results is to compare the various clustering solutions of the same data found by different clustering techniques. The Rand index is used to compare the solutions found by rPCA, HCA, and nHCA techniques. If the results across these techniques are consistent, it is assumed that meaningful structure in the data is being found (Bezdek and Pal 1998).

In this chapter, we conduct validation analysis for the accumulated precipitation (ACCPPT) field and the mean sea-level pressure (MSLP) field for all output times. As for the other fields, the validation is analyzed for the finishing time only (36^{th} hour).

6.2 Clustering Validation for the Accumulated Precipitation (ACCPPT) field

6.2.1 Number of Clusters

We apply the five indexes *DB*, $MH\hat{I}$, *DU*, *BPDU*, and DU_{GG} (section 5.2) on the clustering structures obtained using Ward's method based on the Euclidean distance (Figure 4-6). Table 6-1 shows the computed values for these indexes and the number of clusters suggested by each one for the accumulated precipitation (ACCPPT).

Table 6-1: Determining the correct number of clusters for the ACCPPT data.

Time	Index	Number of clusters							k
		2	3	4	5	6	7	8	
3rd hour	DB	1.28	1.46	1.34	1.33	1.18	1.06	**1.04**	6
	$MH\hat{\Gamma}$	0.51	0.53	**0.76**	0.81	0.83	0.86	0.87	
	DU	0.66	0.5	0.61	0.67	**0.76**	0.76	0.62	
	$BPDU$	0.78	0.71	0.82	0.87	**0.99**	0.88	0.82	
	DU_{GG}	0.57	0.53	0.55	0.59	**0.71**	0.71	0.68	
6th hour	DB	1.38	1.46	1.68	1.7	1.48	1.31	**1.19**	3
	$MH\hat{\Gamma}$	0.51	**0.69**	0.69	0.72	0.72	0.75	0.76	
	DU	0.68	**0.7**	0.56	0.64	0.64	0.64	0.69	
	$BPDU$	0.86	**0.88**	0.75	0.87	0.84	0.85	0.88	
	DU_{GG}	0.61	**0.63**	0.46	0.49	0.49	0.51	0.59	
9th hour	DB	1.88	1.05	1.35	1.27	1.08	0.98	**0.82**	3
	$MH\hat{\Gamma}$	0.69	**0.79**	0.78	0.82	0.87	0.9	0.91	
	DU	0.56	**0.8**	0.49	0.49	0.54	0.55	0.55	
	$BPDU$	0.73	**0.98**	0.71	0.63	0.69	0.7	0.7	
	DU_{GG}	0.44	**0.71**	0.44	0.49	0.54	0.58	0.68	
12th hour	DB	1.44	1.33	1.34	1.08	1.23	1.08	**1.06**	3
	$MH\hat{\Gamma}$	0.81	0.88	0.84	0.85	0.86	0.9	0.91	
	DU	0.63	**0.73**	0.48	0.56	0.56	0.59	0.51	
	$BPDU$	0.87	**0.95**	0.63	0.73	0.7	0.73	0.68	
	DU_{GG}	0.61	**0.69**	0.45	0.53	0.44	0.47	0.47	
15th hour	DB	1.93	1.74	1.49	1.7	1.39	1.34	**1.19**	4
	$MH\hat{\Gamma}$	0.66	0.64	**0.78**	0.78	0.82	0.82	0.84	
	DU	0.64	0.53	**0.66**	0.57	0.64	0.61	0.61	
	$BPDU$	0.81	0.66	**0.82**	0.72	0.81	0.76	0.76	
	DU_{GG}	0.48	0.45	**0.56**	0.4	0.45	0.5	0.5	
18th hour	DB	1.67	1.87	1.51	1.3	1.51	1.42	**1.27**	8
	$MH\hat{\Gamma}$	0.42	0.35	0.6	**0.72**	0.73	0.77	0.8	
	DU	0.66	0.53	0.54	0.61	0.65	0.68	**0.7**	
	$BPDU$	0.83	0.71	0.71	0.79	0.83	0.84	**0.86**	
	DU_{GG}	**0.54**	0.41	0.42	0.47	0.43	0.46	0.47	
21st hour	DB	1.35	**0.45**	1.16	0.85	1.13	1.18	1.03	3
	$MH\hat{\Gamma}$	0.78	0.8	**0.89**	0.9	0.91	0.92	0.92	
	DU	0.8	**1.01**	0.59	0.76	0.7	0.66	0.72	
	$BPDU$	0.88	**1.07**	0.77	0.95	0.83	0.79	0.86	
	DU_{GG}	0.63	**0.98**	0.47	0.65	0.47	0.48	0.52	

Table 6-1 *Continued*

Time	Index	Number of clusters							*k*
		2	3	4	5	6	7	8	
24th hour	DB	2.36	1.71	1.4	1.26	1.28	1.13	1.01	5
	$MH\hat{\Gamma}$	0.43	0.63	0.75	0.81	0.83	0.88	0.9	
	DU	0.46	0.52	0.55	0.58	0.5	0.57	0.58	
	BPDU	0.67	0.73	0.75	0.77	0.64	0.71	0.73	
	DU_{GG}	0.34	0.39	0.41	0.47	0.41	0.46	0.47	
27th hour	DB	0.38	1.45	1.17	0.94	0.76	0.71	0.67	2
	$MH\hat{\Gamma}$	0.73	0.84	0.87	0.89	0.88	0.91	0.92	
	DU	1.18	0.53	0.56	0.6	0.6	0.7	0.81	
	BPDU	1.26	0.79	0.81	0.82	0.8	0.92	1.02	
	DU_{GG}	1.18	0.46	0.49	0.53	0.63	0.73	0.84	
30th hour	DB	0.44	1.61	1.47	1.18	1.04	0.89	0.83	2
	$MH\hat{\Gamma}$	0.6	0.7	0.73	0.8	0.85	0.88	0.9	
	DU	0.88	0.46	0.46	0.52	0.54	0.63	0.63	
	BPDU	1.01	0.75	0.68	0.77	0.8	0.87	0.86	
	DU_{GG}	0.97	0.41	0.41	0.47	0.5	0.57	0.57	
33rd hour	DB	1.31	1.21	0.74	0.75	0.87	0.78	0.55	8
	$MH\hat{\Gamma}$	0.67	0.72	0.76	0.83	0.9	0.94	0.94	
	DU	0.76	0.51	0.63	0.65	0.71	0.78	0.89	
	BPDU	0.89	0.63	0.77	0.78	0.85	0.91	0.96	
	DU_{GG}	0.65	0.5	0.61	0.63	0.62	0.68	0.87	
36th hour	DB	1.11	0.85	0.73	1.02	0.71	0.62	0.53	8
	$MH\hat{\Gamma}$	0.38	0.61	0.72	0.82	0.85	0.91	0.97	
	DU	0.54	0.61	0.65	0.57	0.57	0.73	0.98	
	BPDU	0.66	0.72	0.77	0.72	0.66	0.82	1.08	
	DU_{GG}	0.52	0.58	0.63	0.48	0.53	0.66	0.9	

The suggested value of *k* is shaded in the Table 6-1 and the corrected number of clustering determined using the majority rule is shown in the last column (column *k*). This table shows dramatic change in the number of clusters throughout the time. It goes up from 3 (at 12th hour) to 8 (at 18th hour). Then it drops down to 2 (at 27th hour) and ends with 8 at the finishing time. This dramatic change is not logical in the ensemble forecasting. Also the number of clusters determined by Table 6-1 is not correct at all

times. For example, the determined number of clusters for the 30th hour is 2 whereas the subjective analysis of this time (section 4.3) indicates that the ensemble members group into 4 clusters. If the number of cluster is limited to fewer number ($\leq$5) then 2 is the suggested number of clusters for the 18th and 33rd hours, and 4 is the suggested number of clusters for the 36th hour. This in fact removes the dramatic changes in the number of clusters. The trade off, however, is the loss of structure in the resultant solutions, since one of the obtained clusters account for less than 15% of the other with respects to the size of each cluster. Cutting the clustering dendrograms (Figure 4-6) at the suggested number of clusters by Table 6-1 results in partitions that have one common observation. In these partitions, one large cluster and some other small clusters are recovered. The small clusters account for small percentile of the size of the large one. In fact some of these clusters are atomic (i.e., contains only one member). Such partitions (chain-like type clusters), which appear in this data set especially when approaching the finishing time, are not desired in cluster analysis. Recall from section 5.2.3 that for the chain type cluster, only DU_{GG} index is applicable among the five indexes discussed in section 5.2. Thus, when determining the number of clusters in chain type clustering structures, only the value suggested by the DU_{GG} index should be considered. However, this particular index changes dramatically from time to time and fails to determine the correct number of clusters at the 30th hour which is found by expert to be 4 (section 4.3). These observations suggest that these indexes are inapplicable to the precipitation data analyzed in this section.

6.2.2 Replication and Sensitivity analyses

The replication analysis (section 5.3) is conducted for each output time of the ACCPPT data set. Each data matrix is divided into two random samples of equal size. The two samples are clustered using Ward's method based on the Euclidean distance. Both nearest centroid (NC) and nearest neighbor (NN) assignment rules (section 5.3) are applied. The clustering structures resulted from the assignment rules and the one resulting from cluster analyzing the second samples are compared using Rand agreement measure (5.17). Table 6-2 shows the replication analyses for the ACCPPT data set. As the table indicates, in more than 80% of times, neither NC rule nor NN rule are able to recover the clustering structure in the data. These rules assign all members to only one cluster that indicates that the data do not replicate very well. *In other words, the clustering solutions are not stable across multiple samples for precipitation.* However, before such conclusion can be drawn about this data, some observations need to be stated.

Table 6-2: Replication analyses for the ACCPPT data set

Time	k	NC Rule		NN Rule	
		k	Rand	*k*	Rand
3rd	6	2	0.22	1	-
6th	3	1	-	1	-
9th	3	1	-	1	-
12th	3	1	-	1	-
15th	4	1	-	1	-
18th	8	2	0.20	2	0.44
21st	3	1	-	1	-
24th	5	1	-	1	-
27th	2	1	-	1	-
30th	2	1	-	1	-
33rd	8	1	-	1	-
36th	8	1	-	1	-

First, the Rand agreement measure can be computed, in principle, between two clustering solutions even if one of them has only one cluster. For example, the agreement values for Rand index are 0.60 at the 21st hour and 0.92 at the 30th hour. Reporting these values in Table 6-2 is intentionally avoided because this produces misleading results. Second, is the chain-like type clustering structures observed in section 6.2.1. For example, although 8 clusters are determined for the 36th hour, these clusters are as follow: one large cluster (15 members), two small clusters (2 and 3 members) and five atomic clusters (1 member in each). These observations led to examining the data themselves. The Euclidean distance matrix of the data matrix at the 36th hour is examined. With respect to the least value in the Euclidean distance matrix, the members of ARPS, Eta models and some members from MM5 (16, 19, 20, and 23) are found to have reasonable distance among themselves. Other members have great distance values with all other members. Figure 6-1 shows 2D representation of these members using the first and second principal components of the Euclidean similarity matrix derived from the Euclidean distance matrix for the 36th hour data.

This observation, as illustrated by Figure 6-1, is consistent with the comments in section 4.3 about the variability among the members of MM5 model. Moreover, it agrees with clustering tree for this time. The most significant observation is about constructing the two samples required in the replication analysis. The two samples must be equally well representative of the data. They both must have the same characteristics of the data themselves. *This is unlikely to happen in sum meteorological fields, such as precipitation.* Since the precipitation data are usually discontinuous, it is not unusual, for instance, to rain in some areas of the domain while dry in others. In fact, the data

matrices for the ACCPPT contain so many zeros (no rain) that the two samples may lead to incorrect results. These observations indicate that the replication analysis is inapplicable to such data and the conclusion about the stability of the clustering structures lacks evidence.

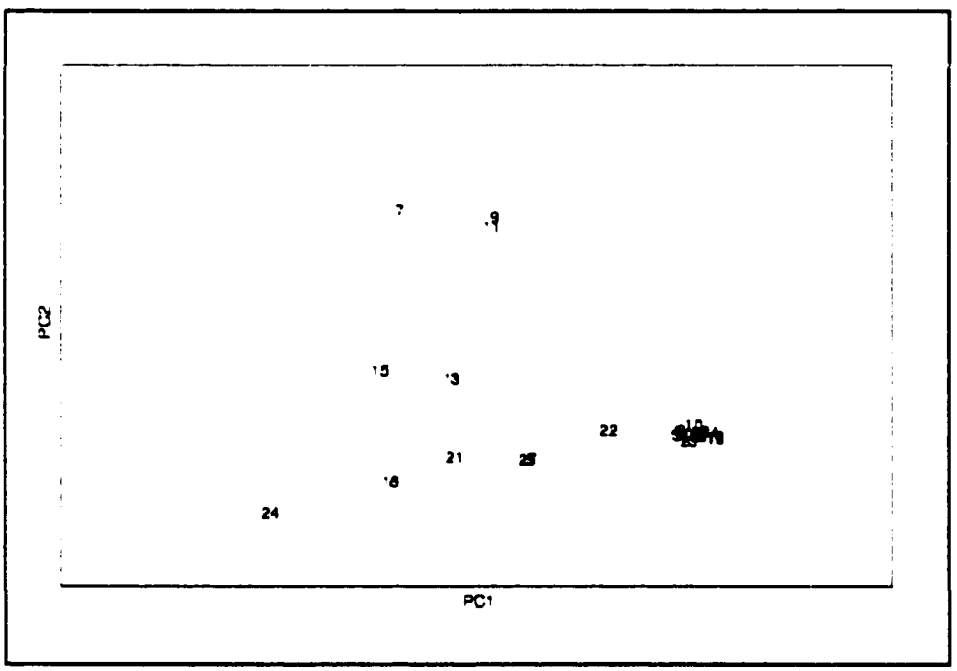


Figure 6-1: 2D representation the ensemble members at the 36th hour data for ACCPPT field.

The sensitivity of the recovered clustering solutions across multiple samples is then analyzed for the ACCPPT. Recall from section 5.3 that in the replication analysis algorithm the two halves (or samples) of the data matrix are clustered using the same clustering procedure. However, the clustering solutions of the two samples are not used directly in this analysis. Instead, an assignment rule is used to assess the stability of the resulting clustering solution (section 5.3). The sensitivity of the clustering solution is now validated by comparing the clustering solutions obtained for the two samples to the

solution obtained for the full data in Figure 4-6 and for each other. This enables us to test whether the clustering solution for a given data set remains the same when only portion of the data used in the clustering procedure. Table 6-3 shows the computed agreement between the clustering solutions obtained in Figure 4-6 and the ones obtained using either portion of the data (samples A and B) for each output time. As the table shows, the recovered clustering solutions in most cases are relatively insensitive whether the data used in full or when only portion of data is used to cluster the ensemble members. However, in some cases (e.g., 27th and 30th hours) the recovered clustering solution slightly differs when only portion of the data is used.

Table 6-3: The sensitivity of the clustering solutions for the ACCPPT across multiple samples computed using Rand index.

Time	k	Full and Sample A	Full and Sample B	Sample A and Sample B
3rd	6	1.0	1.0	1.0
6th	3	1.0	1.0	1.0
9th	3	0.85	1.0	0.85
12th	3	1.0	1.0	1.0
15th	4	1.0	0.99	0.99
18th	8	0.98	0.94	0.92
21st	3	1.0	0.73	0.73
24th	5	0.88	1.0	0.88
27th	2	0.62	1.0	0.62
30th	2	0.62	1.0	0.62
33rd	8	1.0	0.95	0.95
36th	8	1.0	1.0	1.0

6.2.3 Comparing Partitions

Clustering solutions obtained for the same data using different algorithms are compared using objective agreement measures. The clustering solution is valid when different algorithms lead to the same solution. Various clustering solutions produced by different

clustering methods for the same data set are compared between each other. The comparison is limited for the finishing time data (36^{th} hour) and can be done for any other time in the same manner. First, the rotated principal component analysis (rPCA) results are compared to those of the hierarchical cluster analysis (HCA). The comparison is done between the similarity-based solutions (Figure 4-2 and Table 4-3) and between the dissimilarity-based solutions (Figure 4-6 and Table 4-4). The HCA solutions are obtained by cutting the clustering tree at the number of clusters suggested by rPCA. The agreement between the different solutions is computed using Rand index (section 5.4) and shown in Table 6-4. As the table shows, the agreement is moderate for the dissimilarity-based and reasonable for the similarity-based solutions.

Table 6-4: Agreement between rPCA and HCA solutions for the ACCPPT at the 36^{th} hour.

	K	Rand
Based on Correlation	4	0.82
Based on Euclidean	4	0.77

Second, the clustering solutions recovered by the hierarchical cluster analysis (HCA) and the nonhierarchical cluster analysis (nHCA) are compared to each other. The partitions obtained form Ward's method (Figure 4-6), *K*mean algorithm (Table 4-5), and nucleated agglomerative (NA) algorithm (Table 4-6) are compared using Rand index. Since the indexes for determining the number of clusters for the precipitation data (section 6.2.1) found to be inapplicable to such data; so, in order to compare the partitions, the number of clusters is chosen to be 4 as determined by the expert (section 4-3). The seed points for *K*mean and NA are specified subjectively based on Ward's result. The computed agreements between the 3 solutions are shown in Table 6-5. As

the table shows, there is a very good agreement between Ward's and NA algorithms. Also, the table indicates that the solution found by Kmean is different than those recovered by Ward's and NA methods.

Table 6-5: Agreement between HCA and nHCA solutions for the ACCPPT at the 36th hour.

	k	Rand
Ward's and *K*mean	4	0.77
Ward's and NA	4	0.93
*K*mean and NA	4	0.73

6.3 Clustering Validation for the Mean Sea Level Pressure (MSLP) field

6.3.1 Number of Clusters

In this section, the five indexes *DB*, $MH\hat{\Gamma}$, *DU*, *BPDU*, and DU_{GG} (section 5.2) are applied on the clustering structures obtained using Ward's method based on the Euclidean distance (Figure 4-10). Table 6-6 shows the computed values for these indexes and the number of clusters suggested by each one for the MSLP data at 6 hours interval. The suggested value of *k* is shaded in the table and the corrected number of clusters determined using the majority rule is shown in the last column (column *k*). As the table shows, the clustering structures are consistent throughout the time. Cutting the clustering dendrograms of Figure 4-10 at the number of clusters suggested by Table 6-6 results in partitions that support our finding in chapter 4 about the tendency of the members of each model. That's the members are clustered first by models and then by institutions (i.e., Eta and RSM in one cluster). However, the number of clusters determined by Table 6-6 for the 36th hour is 3 whereas the number of clusters found by

the expert is 4 (section 4.4). The only difference is that Table 6-6 suggests that the members of both Eta and RSM are more similar to each other than to the others. This, in fact, does not contradict with the expert finding.

Table 6-6: Determining the correct number of clusters for the MSLP.

Time	Index	Number of clusters							K
		2	3	4	5	6	7	8	
0^{th} hour	DB	0.61	0.84	1.19	0.74	0.69	0.73	1.08	2
	$MH\hat{I}$	0.87	0.94	0.95	0.96	0.96	0.96	0.97	
	DU	0.9	0.56	0.4	0.63	0.66	0.69	0.55	
	$BPDU$	1.24	0.76	0.57	0.8	0.91	0.92	0.8	
	DU_{GG}	1.64	1.26	0.78	0.9	0.9	0.78	0.45	
6^{th} hour	DB	1.02	1.07	**0.76**	0.79	1.17	1.06	1	4
	$MH\hat{I}$	0.44	0.72	**0.95**	0.95	0.95	0.95	0.95	
	DU	0.32	0.29	**0.66**	0.36	0.36	0.36	0.36	
	$BPDU$	0.64	0.6	**0.94**	0.67	0.55	0.55	0.54	
	DU_{GG}	1.19	0.92	1.13	0.89	0.51	0.51	0.65	
12^{th} hour	DB	1.38	1.14	1.23	1.22	1.11	1.09	**1.03**	3
	$MH\hat{I}$	0.27	**0.8**	0.85	0.87	0.9	0.9	0.9	
	DU	0.25	0.3	0.21	0.21	**0.32**	0.27	0.27	
	$BPDU$	0.47	0.47	0.59	0.52	**0.78**	0.51	0.51	
	DU_{GG}	0.77	**0.8**	0.7	0.53	0.56	0.47	0.46	
18^{th} hour	DB	0.88	**0.82**	1.07	0.97	1.18	1.28	1.2	3
	$MH\hat{I}$	0.62	**0.85**	0.9	0.9	0.9	0.92	0.92	
	DU	**0.43**	0.4	0.33	0.17	0.17	0.27	0.27	
	$BPDU$	0.67	**0.73**	0.6	0.44	0.41	0.63	0.51	
	DU_{GG}	1.37	1.22	0.84	0.77	0.5	0.43	0.43	
24^{th} hour	DB	1.28	0.96	**0.66**	0.86	0.85	0.95	1.03	4
	$MH\hat{I}$	0.43	0.77	**0.9**	0.91	0.91	0.92	0.92	
	DU	0.4	0.53	**0.83**	0.48	0.33	0.34	0.34	
	$BPDU$	0.63	0.76	**1.13**	0.77	0.53	0.53	0.53	
	DU_{GG}	1.12	1.06	**1.14**	0.68	0.58	0.58	0.55	

Table 6-6: *Continued.*

Time	Index	Number of clusters							K
		2	3	4	5	6	7	8	
30th hour	DB	1.47	0.78	1.71	1.49	1.53	0.98	1.02	3
	$MH\hat{\Gamma}$	0.32	0.77	0.79	0.78	0.78	0.87	0.88	
	DU	0.36	0.63	0.19	0.19	0.19	0.19	0.19	
	$BPDU$	0.58	0.98	0.66	0.43	0.43	0.43	0.43	
	DU_{GG}	0.77	1.25	0.4	0.4	0.4	0.55	0.53	
36th hour	DB	1.44	0.7	1.01	0.92	0.79	0.63	0.97	3
	$MH\hat{\Gamma}$	0.34	0.9	0.92	0.91	0.92	0.92	0.94	
	DU	0.36	0.75	0.36	0.36	0.36	0.36	0.37	
	$BPDU$	0.59	1.15	0.83	0.69	0.69	0.62	0.64	
	DU_{GG}	0.62	1.52	0.75	0.75	0.81	0.73	0.53	

6.3.2 Replication and Sensitivity analyses

The replication analysis (section 5.3) is conducted for the MSLP data set at 6 hours interval. The two samples are clustered using Ward's method based on the Euclidean distance. Both nearest centroid (NC) and nearest neighbor (NN) assignment rules (section 5.3) are used and the results of these analyses are shown in Table 6-7. Overall, the clustering solutions replicate moderately well and the NN rule results in better agreement. At the finishing time (36th h), the clustering structures for the MSLP data replicates very good, hence it is stable. Note that the pressure data are continuous, hence the two samples are expected to be equally well representative and capture the same characteristics of the data. Thus, the replication analysis can be applicable to some meteorological fields, but not for all.

Table 6-7: Replication analyses for the MSLP data set.

Time	k	NC Rule		NN Rule	
		k	Rand	k	Rand
0^{th}	2	1	-	1	-
6^{th}	4	3	0.80	3	0.78
12^{th}	3	3	0.57	2	0.64
18^{th}	3	2	0.65	2	0.67
24^{th}	4	3	0.75	3	0.80
30^{th}	3	3	0.67	2	0.67
36^{th}	3	3	0.94	3	1.0

The sensitivity of the recovered clustering solutions across multiple samples is then analyzed for the MSLP data set. Recall from section 5.3 that in the replication analysis algorithm the two samples of the data matrix are clustered using the same clustering procedure. However, the clustering solutions of the two samples are not used directly in this analysis. Instead, an assignment rule is used to assess the stability of the resulting clustering solution (section 5.3). The sensitivity of the clustering solution is now validated by comparing the clustering solutions obtained for the two samples to the solution obtained for the full data in Figure 4-10 and for each other. This enables us to test whether the clustering solution for a given data set remains the same when only portion of the data used in the clustering procedure. Table 6-8 shows the computed agreement between the clustering solutions obtained in Figure 4-10 and the ones obtained using either portion of the data (samples A and B) for each output time. As the table shows, the recovered clustering solutions in almost all cases are identical whether the data used in full or when only portion of data is used to cluster the ensemble members. This indicates the consistency of the clustering solutions of the ensemble members for the MSLP data, except at 12 hours.

Table 6-8: The sensitivity of the clustering solutions for the MSLP across multiple samples computed using Rand index.

Time	*k*	Full and Sample A	Full and Sample B	Sample A and Sample B
0^{th}	2	1.0	1.0	1.0
6^{th}	4	1.0	1.0	1.0
12^{th}	3	0.75	0.75	1.0
18^{th}	3	1.0	1.0	1.0
24^{th}	4	1.0	1.0	1.0
30^{th}	3	1.0	1.0	1.0
36^{th}	3	1.0	1.0	1.0

6.3.3 Comparing Partitions

The clustering solutions obtained for the MSLP data at the 36^{th} hour using different methods are compared. First, we compare the rotated principal component analysis (rPCA) results to those of the hierarchical cluster analysis (HCA). The comparison is done between the similarity-based solutions (Figure 4-7 and Table 4-8) and between the dissimilarity-based solutions (Figure 4-10 and Table 4-9). The HCA solutions are obtained by cutting the clustering tree at the number of clusters suggested by rPCA. The agreement between the different solutions is computed using Rand index (section 5.4) and shown in Table 6-9. As shown by the table, the clustering solutions obtained using rPCA and HCA are identical.

Table 6-9: Agreement between PCA and HCA solutions for the MSLP at the 36^{th} hour.

	k	Rand
Based on Correlation	3	1.0
Based on Euclidean	3	1.0

Second, we compare the clustering solutions recovered by the hierarchical cluster analysis (HCA) and the nonhierarchical cluster analysis (nHCA). The partitions obtained form Ward's method (Figure 4-10), *K*mean algorithm (Table 4-10), and nucleated agglomerative (NA) algorithm (Table 4-11) are compared using Rand indexes. The number of clusters is chosen to be 3 as determined in section 6.3.1 and the seed points for *K*mean and NA are specified subjectively based on Ward's result. The computed agreements between the 3 solutions are shown in Table 6-10. As the table shows, the clustering solutions recovered by HCA and nHCA are identical. Tables 6-9 and 6-10 indicate that all clustering methods applied to the pressure (MSLP) data lead to the same clustering solutions with respect to 3 clusters.

Table 6-10: Agreement between HCA and nHCA solutions for the MSLP at the 36^{th} hour.

	k	Rand
Ward's and *K*mean	3	1.0
Ward's and NA	3	1.0
*K*mean and NA	3	1.0

6.4 Cluster Validation for the other fields

In this section, we apply the validation techniques for the other fields at the finishing time only.

6.4.1 Number of Clusters

In this section, the five indexes *DB*, $MH\hat{\Gamma}$, *DU*, *BPDU*, and DU_{GG} (section 5.2) are applied on the clustering structures obtained using Ward's method based on the Euclidean distance (section 4.5.1). Table 6-11 shows the computed values for these indexes and the number of clusters suggested by each one for the other fields at the 36^{th}

hour. The suggested value of k is shaded in the table and the corrected number of clustering determined using the majority rule is shown in the last column (column k). If the clustering trees for each field (36th hour) are cut at the level suggested by Table 6-11, then the resultant solutions have one thing in common. That is, the ensemble members are grouped according their models. The three fields of Absolute Vorticity have some similarity with the Accumulated Precipitation with respect to the variability among the members. However, contrary to the Accumulated Precipitation, their members strictly cluster according to their models.

Table 6-11: Determining the correct number of clusters for the other fields at the finishing time (36th hour).

Field	Index	Number of clusters							k
		2	3	4	5	6	7	8	
CAPE	DB	1.03	1.18	1.13	**0.96**	1.06	1.04	1.01	5
	$MH\hat{\Gamma}$	0.59	0.77	**0.89**	0.9	0.9	0.92	0.95	
	DU	0.66	0.49	0.64	0.69	0.58	0.58	**0.75**	
	$BPDU$	0.85	0.84	0.92	0.89	0.71	0.71	**0.93**	
	DU_{GG}	0.75	0.73	0.7	**0.77**	0.55	0.63	0.67	
Height 500mb	DB	0.72	**0.64**	0.83	1.06	0.93	0.83	0.82	3
	$MH\hat{\Gamma}$	0.77	**0.88**	0.88	0.91	0.92	0.95	0.96	
	DU	0.54	**0.66**	0.38	0.37	0.37	0.43	0.51	
	$BPDU$	**1.08**	1.01	0.62	0.67	0.67	0.77	0.86	
	DU_{GG}	1.11	**1.43**	0.92	0.74	0.74	0.79	0.68	
Temperature 2m	DB	1.02	0.8	**0.65**	0.72	0.72	0.74	0.67	5
	$MH\hat{\Gamma}$	0.74	0.85	0.89	**0.96**	0.96	0.96	0.97	
	DU	0.83	**0.9**	0.74	0.84	0.63	0.49	0.54	
	$BPDU$	1.03	1.01	0.8	**1.14**	0.92	0.73	0.8	
	DU_{GG}	0.97	1.04	0.94	**1.13**	0.82	0.66	0.72	

Table 6-11 *Continued*

Field	Index	Number of clusters							k
		2	3	4	5	6	7	8	
Wind 250mb	DB	**1.03**	1.15	1.18	1.04	1.25	1.2	1.07	2
	$MH\hat{\Gamma}$	**0.8**	0.83	0.86	0.87	0.89	0.92	0.93	
	DU	0.71	0.61	0.66	**0.72**	0.72	0.66	0.67	
	BPDU	**1.03**	0.81	0.8	0.91	0.86	0.81	0.93	
	DU_{GG}	**0.97**	0.68	0.68	0.76	0.63	0.73	0.73	
Absolute Vorticity 250mb	DB	1.74	1.62	1.74	1.37	1.43	1.27	1.11	8
	$MH\hat{\Gamma}$	0.58	0.6	0.76	0.78	0.86	0.88	0.91	
	DU	0.63	0.56	0.62	0.63	0.64	0.64	**0.69**	
	BPDU	**0.86**	0.67	0.74	0.75	0.76	0.76	0.82	
	DU_{GG}	0.55	0.43	0.47	0.49	0.49	0.58	**0.63**	
Absolute Vorticity 500mb	DB	2.08	1.56	1.48	1.28	1.36	1.24	**1.09**	6
	$MH\hat{\Gamma}$	0.5	0.63	0.62	0.73	**0.82**	0.85	0.9	
	DU	0.56	0.6	0.58	0.63	**0.69**	0.69	0.69	
	BPDU	0.76	**0.79**	0.64	0.68	0.75	0.72	0.72	
	DU_{GG}	0.43	0.45	0.41	0.44	0.48	**0.55**	0.55	
Absolute Vorticity 850mb	DB	1.79	1.53	1.51	1.32	1.29	1.2	**0.97**	2
	$MH\hat{\Gamma}$	0.57	0.58	0.68	0.76	0.85	0.81	0.83	
	DU	**0.64**	0.56	0.56	0.6	0.6	0.47	0.53	
	BPDU	**0.87**	0.71	0.72	0.77	0.77	0.56	0.62	
	DU_{GG}	0.54	0.52	0.52	**0.56**	0.56	0.45	0.51	
Dewpoint 2m	DB	0.81	0.76	0.87	0.76	0.69	**0.66**	0.74	2
	$MH\hat{\Gamma}$	0.89	0.88	**0.95**	0.95	0.96	0.96	0.97	
	DU	**1.07**	0.64	0.9	0.57	0.59	0.64	0.73	
	BPDU	**1.26**	0.73	1.03	0.85	0.89	0.95	0.98	
	DU_{GG}	1.13	0.65	0.97	0.79	0.82	0.81	0.77	

6.4.2 Replication and Sensitivity analyses.

The replication analysis (section 5.3) is conducted for several other fields in the SAMEX data at the finishing time, and the results are shown in Table 6-12. As the table shows, the absolute vorticity fields and the dewpoint do not replicate. *Knowledge of the*

data for these fields is very important to decide about the applicability of the replication analysis to these fields.

Table 6-12: Replication analyses for the other fields at the finishing time.

Field	k	NC Rule		NN Rule	
		k	Rand	k	Rand
CAPE	5	4	0.61	3	0.50
Height 500mb	3	3	0.69	3	0.75
Temperature 2m	5	2	0.67	2	0.67
Wind 250mb	2	2	0.52	2	0.92
Absolute Vorticity 250mb	8	1	-	1	-
Absolute Vorticity 500mb	6	1	-	1	-
Absolute Vorticity 850mb	2	1	-	1	-
Dewpoint 2m	2	1	-	1	-

The sensitivity of the recovered clustering solutions across multiple samples is then analyzed for the other fields at the 36th hour. Recall, from section 5.3, that in the replication analysis algorithm the two samples of the data matrix are clustered using the same clustering procedure. However, the clustering solutions of the two samples are not used directly in this analysis. Instead, an assignment rule is used to assess the stability of the resulting clustering solution (section 5.3). The sensitivity of the clustering solution is now validated by comparing the clustering solutions obtained for the two samples to the solution obtained for the full data in section 4.5.1 using Ward's method and for each other. This enables us to test whether the clustering solution for a given data set remains the same when only portion of the data used in the clustering procedure. Table 6-13 shows the computed agreement between the clustering solutions obtained in section 4.5.1 and the ones obtained using either portion of the data (samples A and B) for each output time.

As the table shows, the recovered clustering solutions are identical for all fields, except absolute vorticity at 250 and 500mb, whether the data used in full or when only portion of data is used to cluster the ensemble members. For these two fields, the agreements are very high. This indicates that the clustering solutions for these fields are consistent.

Table 6-13: The sensitivity of the clustering solutions for the other fields at the 36th hour across multiple samples computed using Rand index.

Field	*k*	Full and Sample A	Full and Sample B	Sample A and Sample B
CAPE	5	1.0	1.0	1.0
Height 500mb	3	1.0	1.0	1.0
Temperature 2m	5	1.0	1.0	1.0
Wind 250mb	2	1.0	1.0	1.0
Absolute Vorticity 250mb	8	0.98	1.0	0.98
Absolute Vorticity 500mb	6	0.96	0.98	0.95
Absolute Vorticity 850mb	2	1.0	1.0	1.0
Dewpoint 2m	2	1.0	1.0	1.0

6.5 Summary and concluding remarks

This chapter investigates the applicability of the formal methods of cluster validation to meteorological data. Several approaches for clustering validation are applied on SAMEX data. Five different indexes are used to determine the correct number of clusters in the data. Also, the stability of the resultant clustering solutions is analyzed using the replication analysis. In addition, the validity of clustering solutions is assessed by comparing the results of clustering the same data using different methods. These validation techniques have been developed in different contexts other than meteorology (e.g., computer science, biology, psychology, ... etc). The typical problems in these areas are usually consists of large number of objects in relatively small dimension,

whereas problems in ensemble forecasting consists of small number of objects but in high dimension.

Form the results in this chapter; the techniques used to determine the number of clusters work miserably on the meteorological data. Their suggested number of clusters, in most cases, is not logical and undependable. Also, the replication analysis is found inapplicable to meteorological data in two cases. First, when the members preserve high variability among themselves, which leads to chain-like type clusters. Second, when constructing two samples that capture the same characteristics of the data is not possible. These two cases arise in some fields such as the accumulated precipitation. As for some other fields, such as MSLP, the stability is moderate to strong.

The formal methods of validation are found inapplicable for meteorological data. Therefore, we believe that meteorologist still has to rely on the use of subjective intuition of an expert (or multiple experts).

CHAPTER 7

CONCLUSIONS

This dissertation aims to develop the tools needed to automate the decision making process in the ensemble forecasting system. A fundamental methodology in data mining, namely clustering, is used to detect and identify the clustering structures of the ensemble members during their evolution. Two multivariate data analysis tools are examined: cluster analysis and principal component analysis. Different algorithms from these tools are applied on two different short-term ensembles forecasting. The first is a uni-model consisting of 10 members and runs for 48 hours. The data are available at 6-hour interval for each field. The second is a multi-model having 25 members and runs for 36 hours where the data is available at 3-hour interval. Ten fields are analyzed and their clustering structures are identified for each output time during the time period of the ensemble. In fact, 85 data sets from the uni-model ensemble and 129 data sets from the multi-model ensemble (total of 214 data sets) are analyzed in our research. The clustering solutions are then validated using different techniques form cluster validity. This research demonstrates the use of clustering algorithms for automating the decision process involved in finding similarity in ensemble members.

In chapter 2, the literature related to cluster analysis and principal component analysis is reviewed. A description of the properties of the data matrices used in this research and how they are constructed is given in this chapter. This chapter presents the general framework for the hierarchical cluster analysis and reviews the common and new developed resemblance measures. Four hierarchical clustering algorithms, single

linkage, complete linkage, average linkage, and Ward's are reviewed. Also, two none hierarchical clustering algorithms, *K*mean and nucleated agglomerative are discussed. In addition, the basic idea of the principal component analysis is discussed and two rotation algorithms, quartimax and promax, are reviewed.

The application of clustering is first conducted on a small uni-model short-term ensemble in chapter 3. The data of this ensemble are the output of the Eta model and obtained from the Storm Prediction Center (SPC), Norman, Oklahoma. Ten different meteorological fields are analyzed from this ensemble. Various combinations of clustering algorithms and techniques are applied on the data sets of these fields. Based on the analyses of several meteorological fields, it is observed that the similarity-based principal component analysis and similarity-based cluster analysis do not provide useful clustering tendencies in all cases. They are very useful only when variance in the data set is not concentrated in one dimension. However, when the variables are highly correlated the rotated principal component analysis gives more useful results than those given by the cluster analysis. Using PCA, the degree of concentration of the data can be realized by analyzing the variance explained by the first eigenvalue across the time. Also when oblique solution is found by PCA, the separation between the PCs (or clusters) can be measured using the correlation matrix of the rotated PCs which is computed by the promax algorithm. On the other hand, dissimilarity or distance based cluster analysis and principal component analysis provide useful information in all cases.

For the distant-based hierarchical cluster analysis, a method has been developed for drawing the clustering trees for each field that takes into consideration the different

states of the ensemble members during their evolution. The clustering trees for each field are drawn using the same scale. Therefore, the tightness or divergence among the ensemble members for one field can be realized visually by examining the trees height and the level of mergers as the time goes from the initial time to the finishing time.

A heuristic, rule of thumb, stopping rule for correlation-based PCA is derived to help analyst to decide whether rotation of PC is needed. The rule determines the likelihood that there is only one cluster in the data by examining the ratio of the first eigenvalue to the second one. When there is only 1 cluster then the rotation using quartimax is not needed.

Moreover, the clustering algorithms are shown to be sensitive in the context of ensemble forecasting system. The sensitivity of the hierarchical cluster analysis is studied with respect to the changes of several parameters, such as scaling method, resemblance coefficient, clustering method. Also, the nonhierarchical clustering algorithms are found sensitive to the change of the seed point selection.

In chapter 4, the clustering techniques are applied on a multi-model ensemble known as SAMEX, which is a project coordinated by CAPS at the University of Oklahoma and involving scientists at NCEP and NSSL. The full ensemble consists of 25 members made from 4 different numerical weather prediction models: 5 members from the ARPS, 5 members from the NCEP Eta model, 5 members from the NCEP RSM, and 10 members from MM5 run at NSSL. Each model member extends out to 36 hour beginning from 0000 UTC 29 May 1998. Ten different fields are analyzed from this ensemble. The results of these analyses in this chapter continue to support our previous finding about the performance of the dissimilarity-based versus the similarity-

based clustering techniques. As the opposite of the similarity-based clustering, the dissimilarity-based clustering techniques are found to be useful tools for clustering in all cases.

The results of clustering the members of the SAMEX data show that the ensemble members have strong tendency to build the clusters according to their models. Based on the analyses of the 3-hourly accumulated precipitation (ACCPPT) field, it is observed that the members of each model have tendencies to stay together and build their own cluster. The correlation-based analyses support this finding very strongly. According to these analyses, the members of one model are shown to be coherent and members from different models are isolated. On the other hand, the distance based cluster analysis agrees with this tendency but to a lesser extent. When other forecast fields are examined, such as mean sea-level pressure, CAPE, 500-mb geopotential height, and 250-mb wind speed, the tendency for the forecast members to cluster by model is found. Neither similarity-based nor dissimilarity-based CA or PCA find any inter-model cluster to exist over more than a few model output times. The one strong tendency seen across all forecast times and fields is for the forecasts to cluster by the model used to produce the forecasts. This indicates that the models are more similar to themselves than to the initialization methods used. It is observed that by the 36^{th} hour the model results cluster with the model first, and then with the institution (i.e., Eta and RSM cluster next with each other, before clustering with ARPS and MM5). Even though the Eta and RSM are using the breeding of growing modes technique to generate the perturbed initial and boundary conditions, very little indication of the bred modes clustering together is seen (mode 1 from Eta, RSM clustering together, then mode 2

from Eta, RSM clustering together, etc.). The models all cluster together first, showing the very different model climates that appear quite quickly in the model forecasts (David Stensrud, 2000, personal communication).

The subjective analysis of all clustering dendrograms in this chapter indicates that the MM5 forecasts, that include model physics differences and use less rapidly growing initial conditions than the bred modes, are the most dissimilar of all the forecasts. This is particularly true of the precipitation forecasts, but also is evident in the other fields. Model physics differences, even when used within the context of a single model, provide for a substantial divergence in the solutions. However, even more important is the use of different models in order to produce divergence in the forecast solutions. Even when using bred modes, both the Eta model and RSM cluster with themselves first, and then they cluster together before clustering with either the ARPS or MM5. The bred mode forecasts are not outliers in the sense that they do not span the range of the forecasts on this day. Indeed, the Eta and RSM often are the first models to cluster together. The results strongly suggest that model differences provide much of the divergence in the clustering solutions and that the techniques used to vary the initial conditions are much less important. This conclusion is supported by the subjective analyses of the model forecast fields from selected time periods, which also indicates that the models are very similar to themselves and that it is the forecasts from different models that have very different structures (David Stensrud, 2000, personal communication).

These results further highlight the important role played by model physics in determining the resulting forecasts. While perturbing a single model does improve the

forecast divergence, as shown here and by Buizza et al. (1999) and Stensrud et al. (2000), it is clear that using totally different models is likely a more fruitful approach. We hope that the use of completely different models in short-range ensemble forecasting will continue to be explored vigorously as it enhances the ensemble variability, which is one of the remaining concerns with present operational ensemble forecast systems.

In chapter 5, several approaches for validating clustering solution are reviewed. First approach is concerned with finding the correct number of clusters in the data under study. Determining the correct number of clusters is significant problem in cluster analysis since the clustering algorithms do not compute it. This number is either computed independently from the clustering algorithm using validation index or given by the user. Five of the common indexes to determining the correct number of clusters are reviewed. Two of which are recently developed. The second approach discussed in this chapter is the replication analysis which is used to measure the stability of the clustering solution across multiple samples of the data being clustered. Another validation technique concerns with comparing the clustering solution produced by different clustering algorithms for the same data. Two well-known indexes for comparing partitions are reviewed. Applications of these techniques for validation are conducted in chapter 6.

In chapter 6, the applicability of the formal methods of cluster validation to meteorological data is investigated. Several approaches for clustering validation are applied on SAMEX data. Five different indexes are used to determine the correct number of clusters in the data. Also, the stability of the resultant clustering solutions is

analyzed using the replication analysis. In addition, the validity of clustering solutions is assessed by comparing the results of clustering the same data using different methods. These validation techniques have been developed in different contexts other than meteorology (e.g., computer science, biology, psychology, ... etc). The typical problems in these areas are usually consists of large number of objects in relatively small dimension, whereas problems in ensemble forecasting consists of small number of objects but in high dimension.

The results in this chapter show that the formal techniques of validation cannot be applied blindly in the context of meteorology. Different types of clustering are obtained for different fields. For example, the clustering for accumulated precipitation field was more like chain type whereas the clustering obtained for pressure field was spherical type. Even for the same field, the clustering type may differ from time to another as shown by the accumulated precipitation field. Therefore, when determining the number of clusters only one index among the five indexes discussed in chapter 5 is applicable to chain type clustering. In other words, the type of resultant clustering must be considered when choosing the index. The general conclusion is that the techniques used to determine the number of clusters work miserably on the meteorological data. Their suggested number of clusters, in most cases, is not logical and undependable.

Also, the replication analysis is found to be inapplicable for all meteorological fields. In addition to chaining problem that arises in some field, there is the problem of constructing the two samples. In order for replication analysis to be useful in assessing the stability of clustering solution, the two samples must be equally well representative of the data. This is unlikely to happen in all fields in meteorology. For example, it is not

unusual to rain in some areas while dry in other areas of the domain. Thus, the two samples may lead to incorrect results. For this technique to be useful, the user must be careful and aware of the data.

The results of applying the formal methods of validation on the ensemble data from SAMEX indicate that these methods are inapplicable for meteorological data. Therefore, meteorologist still has to rely on the use of subjective intuition of an expert (or multiple experts) to validate the clustering solution for the ensemble members.

As stated earlier the ultimate goal of this research is to automate the process of making decision about determining the best forecast for the atmosphere. Such automation can be achieved by developing an expert system that utilizes the techniques of clustering and make the decision based on a set of rules. We developed a suite of programs (Appendix B) that constitute the infrastructure for such expert system. All computations related to this research are done on IBM RS/6000 Power Server 590 with one 66.7 MHz processor, 0.5 GB main memory, and 63GB RAID 5 hard disk. The computation time of clustering the 25 ensemble members (SAMEX) is calculated for different clustering procedures carried out on this system, Table 7-1. The time given in this table is for the entire clustering procedure for a given 9477x25 data matrix. That is, it includes the time for normalizing the data matrix (if any) and for the computation of the similarity/dissimilarity matrix in addition to the clustering algorithm. The table shows the user time, system time, and the real time in seconds for each one of the clustering procedures. The *Real Time* is the total amount of time elapsed to run the program (in seconds), the *User Time* is the time the user part of the program took to run (in seconds), and the *System Time* is the time the kernel spent running the program (in

seconds). The computer system used to carry out the clustering programs is not devoted for this research. Hence, the computation times shown in Table 7-1 vary depending on the number of the users of the computer system at the same time. The results shown in Table 7-1 points out the feasibility of automating the process of clustering the ensemble members.

Table 7-1: The amount of time elapsed to run clustering procedure to cluster 25 ensemble members for a given 9477x25 data matrix. The *Real Time* is the total amount of time elapsed to run the program (in seconds), the *User Time* is the time the user part of the program took to run (in seconds), and the *System Time* is the time the kernel spent running the program (in seconds).

Clustering Procedure	User Time	System Time	Real Time
HCA using UPGMA method based on the correlation	9.0 *sec.*	0.3 *sec.*	11.0 *sec.*
HCA using Ward's method based on the Euclidean distance. Data matrix is centered first.	8.9	0.1	9.0
Kmean algorithm	12.0	0.1	12.0
Nucleated agglomerative algorithm	21.7	0.0	22.0
rPCA using quartimax rotation	8.8	0.0	9.0
rPCA using promax rotation	8.8	0.0	9.0

The automation of the process of clustering the ensemble members can be implemented by an expert system. Developing such system has not been done, but this work can now be taken to the next level in creating the expert system for use by forecasters. The expert system should utilize the clustering methodology and draw a conclusion based on a set of stored rules, called the knowledge base, and other information supplied by user. Building the knowledge base is very important but it is out of the scope of this research since it requires a meteorological experiences. As for the clustering methods, we recommend the use of different techniques for the same

data. First, the ensemble members are to be clustered based on the correlation measure. Before using the correlation measure, the heuristic rule (chapter 3) can be used to determine the likelihood that there is more than one cluster in the data. If the rule indicates that only one cluster exists in the data, then the correlation-based clustering is not needed. Second, the ensemble members are to be clustered using Ward's method based on the Euclidean distance. The result of Ward's clustering is used to provide the initial information for the *K*mean and nucleated agglomerative (NA) algorithms. Then, the agreement between the clustering solutions obtained by all clustering methods applied on the same data need to be computed. If partitions agree well then a meaningful clustering solution for the data is being found. If all methods do not agree on one solution, then the clustering solution found by the majority of the methods is to be chosen as the final partition of the members of the ensemble.

Future plans for this research include developing the software for the expert system. The expert system that will make the decision regarding the clustering solution in a short-term ensemble forecasting. Another aim of future work is to test the applicability and usability of cluster tendency. Cluster tendency is the process of deciding whether the data tend to cluster into natural groups without implementing clustering algorithm to identify the groups themselves (Jain and Dubes 1988). Cluster tendency is a preliminary process of clustering and could lead to tremendous effect on the performance of the expert system. Since knowing whether data contains cluster or not without actually clustering them could lead to better utilization of the computing resources especially in meteorology where the amount of data analyzed are usually huge.

REFERENCES

Agrawal, R.; J. Gehrke; D. Gunopulos; and P. Raghavan, [1998], *Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications*, Proceeding of the 1998 ACM SIGMOD International Conference on Management of Data, Seattle, WA.

Alhamed, A. and S. Lakshmivarahan, [2000], *Clustering Methodologies Applied to Short-Term Ensemble Forecasting*, Proceeding of the 2nd Conference on Artificial Intelligence, American Meteorological Society, 80th Annual Meeting, Longbeach, CA., pp. 49-55.

Alhamed, A. and S. Lakshmivarahan, [2000], *A Clustering Approach To Multi-Model Ensemble Analysis Using SAMEX Data: Preliminary Results*, Proceeding of the ACM SAC'2000, March 19-21 2000, Como, Italy.

Alhamed, A., S. Lakshmivarahan, and D. Stensrud, [2000], *Cluster Analysis of Multi-Model Ensemble Data from SAMEX*, Monthly Weather Review, Submitted.

Anderberg, M. R., [1973], **Cluster Analysis For Applications**, Academic Press, NY.

Backer, E., [1995], **Computer-assisted Reasoning in Cluster Analysis**, Prentice Hall, UK.

Basilevsky, A., [1983], **Applied Matrix Algebra in the Statistical Sciences**, North-Holland.

Bezdek, J. C. and N. R. Pal, [1998], *Some New Indexes of Cluster Validity*, IEEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, **28**, pp. 301-15.

Black, T. L., [1994], *The New NMC Mesoscale Eta Model: Description and Forecast Examples*, Weather and Forecasting, **9**, pp. 265-78.

Berson, A.; S. Smith; and K. Thearling, [2000], **Building Data Mining Applications for CRM**, McGraw-Hill, New York, NY.

Betts, A.K., and M.J. Miller, [1986], *A new convective adjustment scheme. Part II: Single column tests using GATE wave, BOMEX, and arctic air-mass data sets.* Quart. J. Roy. Meteor. Soc., **112**, pp. 693-709.

Bock, H.-H., [1996], *Probability Models and Hypotheses Testing in Partitioning Cluster Analysis*, in Clustering and Classification (eds P. Arabie, L. J. Hubert and G. De Soete), World Scientific, Singapore, pp. 377-453

Bock, H.-H., [1997], *Simultaneius Visulaization and Clustering Methods as an Alternative to Kohonen Maps*, in Learning, Networks and Statistics (eds G. D. Riccia, H.-J. Lenz and R. Kruse), Springer-Verlag Wien, NY, pp. 67-85.

Breckenridge, J. N., [1989], *Replicating Cluster Analysis: Method, Consistency, and Validity*, Multivariate Behavioral Research, **24**, pp. 147-161.

Brooks, H. E.; M. S. Tracton; D. J. Stensrud; G. DiMego; and Z. Toth, [1995], *Short-Range Ensemble Forecasting: Report from a Workshop*, Bulletin of the American Meteorological Society, **76**, pp. 1617-34.

Buizza, R., [1997], *Potential forecast skill of ensemble prediction and spread and skill distributions of the ECMWF ensemble prediction system*, Monthly Weather Review, **125**, pp. 99-119.

Buizza, R. and T.N. Palmer, [1995], *The singular-vector structure of the atmospheric general circulation*. Journal of Atmospheric Science, **52**, pp. 1434-1456.

Buizza, M. Miller, and T.N. Palmer, [1999], *Stochastic representation of model uncertainties in the ECMWF ensemble prediction system.* Quart. J. Roy. Meteor. Soc., submitted.

Burk, S.D., and W.T. Thompson, [1989], *A vertically nested regional numerical weather prediction model with second-order closure physics.* Monthly Weather Review, **117**, pp. 2305-2324.

Cios, K.; W. Pedrycz; and R. Swiniarski, [1998], **Data Mining Methods for Knowledge Discovery**, Kluwer Academic Pub., Boston, MA.

Davies, D. L. and D. W. Bouldin, [1979], *A cluster separation measure*, IEEE Trans. Pattern Analysis Mach. Intell., **1**, pp.224-227.

Dey, C. H., [1996], *GRIB: The WMO Format for the Storage of Weather Product Information and the Exchange of Weather Product Messages in Gridded Binary Form*, Edition 1, NCEP, Office Note 388, source: ftp://nic.fb4.noaa.gov/pub/nws/nmc/docs/gribed1

Diehr, G., [1985], *Evaluation of a Branch and Bound Algorithm for Clustering*, Journal of Scientific and Statistical Computing, **6**, pp. 268-84.

Du, J.; S.L. Mullen; and F. Sanders, [1997], *Short-range ensemble forecasting of quantitative precipitation*, Monthly Weather Review, **125**, pp. 2427-2459.

Dudhia, J., [1993], *A nonhydrostatic version of the Penn State-NCAR mesoscale model: Validation tests and simulation of an Atlantic cyclone and cold front.* Monthly Weather Review, **121**, pp. 1493-1513.

Ellis, T. M.; I. R. Philips; and T. M. Lahey, [1994], **FORTRAN 90 programming,** Addison-Wesley, 825 pp.

Elmore, K. L. and M. B. Richman, [2001], "*Euclidean distance as a similarity metric for principal component analysis*", Monthly Weather Review, **129**.

Errico, R., and D.P. Baumhefner, [1987], *Predictability experiments using a high-resolution limited-area model.* Monthly Weather Review, **115**, pp. 488-504.

Etter, D. M., [1993], **Structured FORTRAN 77 for engineers and scientists**, 4th edition, The Benjamin/Cummings Pub. Co., Inc., 616 pp.

Everitt, B. S., [1993], **Cluster Analysis**, 3rd edition, Arnold, London, UK.

Ganti, V.; R. Ramakrishnan; and J. Gehrke, [1999], *Clustering Large Datasets in Arbitrary Metric Spaces*, Proceeding of the 15th International Conference on Data Engineering, IEEE CS Press, Los Alamitos, CA, pp. 502-511.

Ganti, V.; J. Gehrke; and R. Ramakrishnan, [1999], *Mining Very Large Database*, Computer, **32**, pp. 38-45.

Gong, X. and M. B. Richman, [1995], *On the Application of Cluster Analysis to Growing Season Precipitation Data in North America East of the Rockies*, Journal of Climate, **8**, pp. 897-931.

Gordon, A. D., [1996], *Hierarchical Classification*, in Clustering and Classification (eds P. Arabie, L. J. Hubert and G. De Soete), World Scientific, Singapore, pp. 65-121.

Gordon, A. D., [1999], **Classification**, 2nd edition, Chapman & Hall/CRC, NY.

Grell, G.A., [1993], *Prognostic evaluation of assumptions used by cumulus parameterizations.* Monthly Weather Review, **121**, pp. 764-787.

Grell, G.; J. Dudhia; and D. Stauffer [1994], *A Description of the Fifth-Generation Penn State/NCAR Mesoscale Model (MM5)*, NCAR/TN-398+STR, 121 pp. [Available from MMM Division, NCAR, P.O. Box 3000, Boulder, CO 80307].

Guha, S.; R. Rastogi; and K. Shim, [1998], *CURE: An Efficient Clustering Algorithm for Large Databases*, Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, ACM Press, New York, 73-84.

Guha, S.; R. Rastogi; and K. Shim, [1999], *ROCK: A Robust Clustering Algorithm for Categorical Attributes*, Proceedings of the 15th International Conference on Data Engineering, IEEE CS Press, Los Alamitos, CA, pp. 512-521.

Hamill, T. M. and S.J. Colucci, [1997], *Verification of Eta-RSM short-range ensemble forecasts*, Monthly Weather Review, **125**, pp. 1312-1327.

Hamill, T. M. and Colucci, [1998], *Evaluation of Eta-RSM ensemble probabilistic precipitation forecasts*, Monthly Weather Review, **126**, pp. 711-724.

Harman, H. H., [1976], **Modern Factor Analysis**, 3rd edition revised, University of Chicago Press, Chicago.

Harris, C. W. and H. F. Kaiser, [1964], *Oblique Factor Analytic Solutions by Orthogonal Transformations*, Psychometrika, **29**, pp.347-62.

Harrison, M.S.J., T.N. Palmer, D.S. Richardson, and R. Buizza, [1999], *Analysis and model dependencies in medium-range ensembles: Two transplant case studies.* Quart. J. Roy. Meteor. Soc., in press.

Hartigan, J. A., [1975], **Clustering Algorithms**, John Wiley & Sons, Inc. (Wiley series in probability and mathematical statistics), NY.

Hendrickson, A. E. and P. O. White, [1964], *Promax: A Quick Method for Rotation to Oblique Simple Structure*, The British Journal of Statistical Psychology, **17**, pp. 65-70.

Hoffman, R. N., and E. Kalnay, [1983], *Lagged average forecasting, an alternative to Monte Carlo forecasting*, Tellus, **35A**, pp. 100-118.

Hotelling, H., [1933], *Analysis of a Complex of Statistical Variables into Principal Components*, The Journal of Educational Psychology, **24**, Sep., pp. 417-441, Oct., pp. 498-520.

Houtekamer, P. L., and J. Derome, [1995], *Methods for ensemble prediction*, Monthly Weather Review, **123**, pp. 2181-2196.

Hou, D.; E. Kalnay; and K. Drogemeier, [2000], *Objective verification of the SAMEX98 ensemble forecasts*, Monthly Weather Review, [In Press].

Houtekamer, P. L., and L. Lefaivre, [1997], *Using ensemble forecasts for model verification*, Monthly Weather Review, **125**, pp. 2416-2426.

Hubert, L. and P. Arabie, [1985], *Comparing Partitions*, Journal of Classification, **2**, pp. 193-218.

IMSL, [1997], Fortran Subroutines for Statistical Applications, [Available form Visual Numerics, Inc. 9990 Richmond Avenue, Suite 400, Houston, TX 77042, URL: http://www.vni.com].

Jackson, E. J., [1991], **A User's Guide To Principal Components**, Wiley-Interscience Publication, NY.

Jain, A. J. and R. C. Dubes, [1988], **Algorithms For Clustering Data**, Prentice Hall, NJ.

Jennrich, R. I. and P. F. Sampson, [1966], *Rotation for Simple Loadings*, Psychometrika, **31**, pp. 313-23.

Jolliffe, I. T., [1986], **Principal Component Analysis**, Springer-Verlag, New York.

Juang, H.-M. and M. Kanamitsu, [1994], *The NMC Nested Regional Spectral Model*, Monthly Weather Review, **122**, pp. 3-26.

Kain, J.S., and J.M. Fritsch, [1990], *A one-dimensional entraining/detraining plume model and its application in convective parameterization.* Journal of Atmospheric Sciences, **47**, pp. 2784-2802.

Kaiser, H. F., [1958], *The Varimax Criterion for Analytic Rotation in Factor Analysis*, Psychometrika, **23**, pp. 187-200.

Karypis, G.; E.-H. Han; and V. Kumar, [1999], *Chameleon: Hierarchical Clustering Using Dynamic Modeling*, Computer, **32**, pp. 68-75.

Kaufman, L. and P. Rousseeuw, [1990], **Finding Groups in Data: An Introduction to Cluster Analysis**, A Wiley-Interscience Publication, NY.

Key, J. and R. G. Crane, [1986], *A comparison of synoptic classification schemes based on objective procedures*, Journal Climatology, **6**, pp. 375-88.

Komarov, V. S.; V. I. Akselevich; and A. V. Kerminskii, [1994], *Applying the Modified Method of Clustering of Arguments (MMCA) to Forecasting of Parameters of Mean Wind*, Atmos. Oceanic Opt., **7**, pp. 125-28.

Krishnaiah, P. R. and L. N. Kanal, [1982], **Classification Pattern Recongnition and Reduction of Dimensionality (Handbook of Statistics 2)**, North-Holand Pub., Netherlands.

Lance, G. N., and W. T. Williams, [1967], *A General Theory of Classification Sorting Strategies: 1. Hierarchical Systems*, The computer Journal, **9**, pp. 373-380.

Lorr, M., [1983], **Cluster Analysis for social scientists**, Jossey-Bass Inc., Publishers, San Francisco, CA.

Mather, P. M., [1976], **Computational Methods of Multivariate Analysis in Physical Geography**, John Wiley & Sons, Inc., London, UK.

McIntyre, R. M. and R. K. Blashfield, [1980], *A Nearest-Centroid Technique for Evaluating the Minimum-variance Clustering Procedure*, Multivariate Behavioral Research, **2**, pp. 225-238.

Mesinger, F. and R. E. Treadon, [1995], *Horizontal" Reduction of Pressure to Sea Level: Comparison against the NMC's Shell Method*, **123**, Monthly Weather Review, pp. 59-68.

Milligan, G. W., [1979], *Ultrametric Hierarchical Clustering Algorithms*, Psychometrika, **44**, pp. 343-46.

Milligan, G. W., [1981], *A Monte Carlo Study of Thirty Internal Criterion Measures for Cluster Analysis*, Psychometrika, **46**, pp. 187-199

Milligan, G. W., [1996], *Clustering Validation: Results and Implications for Applied Analysis*, in Clustering and Classification (eds P. Arabie, L. J. Hubert and G. De Soete), World Scientific, Singapore, pp. 341-75.

Milligan, G. W. and M. C. Cooper, [1985], *An Examination of Procedures for Determining the Number of Clusters in a Data Set*, Psychometrika, **50**, pp. 159-179.

Milligan, G. W. and M. C. Cooper, [1986], *A Study of the Comparability of External Criteria for Hierarchical Cluster Analysis*, Multivariate Behavioral Research, **21**, pp. 441-458.

Mirkin, B., [1996], **Mathematical Classification and Clustering**, Kluwer Academic Publication, Dordrecht, Netherlands.

Mirkin, B. and I. Muchnik, [1996], *Clustering and Multidimensional Scaling in Russia (1960-1990): A Review*, in Clustering and Classification (eds P. Arabie, L. J. Hubert and G. De Soete), World Scientific, Singapore, pp. 295-339.

Mittelstadt, J., [1995], *Introduction to Ensemble Forecasting*, Western Region Technical Attachement, No. 95-29, Salt Lake City, UT.

Mo, K. and M. Ghil, [1988], *Cluster Analysis of Multiple Planetary Flow Regimes*, Journal o f Geophysical Research, **93**, pp. 10927-52.

Molteni, R.; R. Buizza; T.N. Palmer; and T. Petroliagis, [1996], *The ECMWF ensemble prediction system: Methodology and validation*, Quart. J. Roy. Meteor. Soc., **122**, pp. 73-119.

Morey, L. C.; R. K. Blashfeild; and H. A. Skinner, [1983], *A comparison of Cluster Analysis Techniques Within a Sequential Validation Framework*, Multivariate Behavioral Research, **18**, pp. 309-329.

Mosier, C. I., [1939], *Determining A Simple Structure When Loadings for Certain Tests are Known*, Psychometrika, **4**, pp.149-62.

Mullen, S.L., and D.P. Baumhefner, [1988], *Sensitivity to numerical simulations of explosive oceanic cyclogenesis to changes in physical parameterizations*. Monthly Weather Review, **116**, pp. 2289-2329.

Murtagh, F., [1996], *Neural Networks for Clustering*, in Clustering and Classification (eds P. Arabie, L. J. Hubert and G. De Soete), World Scientific, Singapore, pp. 235-69.

NCAR, [1998], NCAR Graphics version 4.1.1 [Available on-line from http://ngwww.ucar.edu/ngdoc/ng/.]

Okada, A., [1996], *A Review of Cluster Analysis Research in Japan*, in Clustering and Classification (eds P. Arabie, L. J. Hubert and G. De Soete), World Scientific, Singapore, pp. 371-94.

Pal, N. R. and J. Biswas, [1997], *Cluster Validation using Graph Theoretic Concepts*, Pattern Recognition, *30*, pp. 847-857.

Preisendorfer, R. W., [1988], **Principal Component Analysis in Meteorology and Oceanography**, Elsevier Science Pub., Amsterdam, Netherlands.

Ramakrishnan, N. and A. Y. Grama, [1999], *Data Mining: From Serendipity to Science*, Computer, **32**, pp. 34-37.

Richman, M. B., [1986], *Rotation of Principal Components*, Journal of Climatology, **6**, pp. 293-335.

Richman, M. B. and X. Gong, [1999], "*Relationships between the definition of the hyperplane width to the fidelity of principal component loading patterns*", Journal of climate, 12, pp. 1557-1576.

Romesburg, C. H., [1984], **Cluster Analysis For Researchers**, LifeTime Learning Pub. , Belmont, CA.

Ronberg, B. and W. C. Wang, [1987], *Climate patterns deriverd from Chinese proxy precipitation records: An evaluation of the station networks and statistical techniques*, Journal Climatology, **7**, pp. 391-416.

Rosenberg, S.; I. Mechelen; and P. De Boeck, [1996], *A Hierarchical Classes Model: Theory and Method with Applications in Psychology and Psychopathology*, in Clustering and Classification (eds P. Arabie, L. J. Hubert and G. De Soete), World Scientific, Singapore, pp. 123-55.

Stensrud, D. J., J.-W. Bao, and T. T. Warner, [2000], *Using initial condition and model physics perturbations in short-range ensembles simulations of mesoscale convective systems*, Monthly Weather Review, **128**, pp. 2077-2107.

Toth, Z., and E. Kalnay, [1993], *Ensemble forecasting at NMC: The generation of perturbations*, Bull. Amer. Meteor. Soc., **74**, pp. 2317-2330.

Toth, Z., and E. Kalnay, [1997], *Ensemble forecasting at NCEP and the breeding method*, Monthly Weather Review, **125**, pp. 3297-3319.

Tracton, M. S. and E. Kalnay, [1993], *Operational Ensemble Prediction at the National Meteorological Center: Practical Aspects*, NMC Notes, **8**, pp. 379-398

Tryon, R. and D. E. Bailey, [1970], **Cluster Analysis**, McGraw-Hill Book Company, NY.

Wilks, D. S., [1995], **Statistical Methods in the Atmospheric Sciences**, International Geophysics Series.

Wishart, D., [1969], *An Algorithm for Hierarchical Classifications*, Biometrics, **25**, pp. 165-70.

Witten, I. and E. Frank, [2000], **Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations**, Morgan Kaufmann Publishers, San Francisco, CA.

WMO, [1988], World Meteorological Organization publication No. 309, Manual on Codes, Vol. 1, Part B, Secretariat of the WMO, Geneva, Switzerland, 1988, plus Supplements No. 1, 2, and 3.

Xue, M.; K. Droegemeier; and V. Wong; A. Shapiro, [2000a], *The Advanced Regional Prediction System (ARPS) - A multiscale nonhydrostatic atmospheric simulation and prediction tool. Part I: Model dynamics and verification.* Accepted by Meteorology and Atmospheric Physics.

Xue, M.; K. Droegemeier; and V. Wong; A. Shapiro, [2000b], *The Advanced Regional Prediction System (ARPS) - A multiscale nonhydrostatic atmospheric simulation and prediction tool. Part II: Model physics and applications.* Accepted by Meteorology and Atmospheric Physics.

Zhang, D.-L., and R.A. Anthes, [1982], *A high-resolution model of the planetary boundary layer - sensitivity tests and comparisons with SESAME-79 data*, J. Appl. Meteor., **21**, pp. 1594-1609.

Appendix A

GRIB Format

GRIB Format

GRIB (GRIdded Binary) is a general purpose, bit-oriented data exchange format approved by the World Meteorological Organization (WMO) Commission for Basic Systems (CBS). It is an efficient method for transmitting large amount of gridded data over high speed telecommunication lines (Dey 1996). The data are packed into the standard GRIB code; hence message can be made compact, which will produce faster transmissions between computers. GRIB can be used for either transmission or storage.

Typically, GRIB data consists of a sequence of 2-D (lon, lat) chunks of 4-D field variables. For example, (lon, lat, level, time) for the v component on the wind. The sequence is usually organized in files containing all field variables at particular time. That's 3-D (lon, lat, level) volume for each field variable.

GRIB message, also called record, is encoded as a continuous bit stream. GRIB record is logically divided into six sections. The six section provide control information and/or data and these are:

- Indicator Section
- Product Definition Section (PDS)
- Grid Description Section (GDS)
- Bit Map Section (BMS)
- Binary Data Section (BDS)
- '7777' (ASCII Characters)

The GDS and BMS sections optional.

GRIB bit stream is encoded/decoded in term of octets (eight consecutive binary bits, or bytes). The octets codes are described in the WMO Manual on Codes (WMO 1988).

The first section in the GRIB record, Indication Section, is eight octets long and serves to identify the start of the record, provides the total length of the record, and the indicates edition number of GRIB. The PDS section is usually 28 octets long. It contains information about the GRIB message such as, parameter (or field) table version, the originating center, the numerical model, the geographical area covered by the data, the field itself, level, and date/time. The GDS section is an optional and provides a grid description for grids not defined in PDS. The BMS section is an optional and provides either a bit map or a reference to a bit map that defined by the center. The bit map consists of contiguous bits that correspond to the data point as defined in the grid description. A bit with value of 1 indicates the presence of a datum for the corresponding grid point in the BDS, and 0 otherwise. The BDS section provides the packed data and the information required to unpack the original data. The last section contains the ASCII characters '7777' and serves as verification for the completion of the GRIB record.

Computer programs that decode the GRIB files are available for various platforms. The GRIB decoders are available from ftp:. nic.fb4.noaa.gov.pub/info.. In our implementation, we use the appropriate decoder for our machine (IBM RS6000 running AIX 4.x) and we wrote our program (Appendix B) that constructs one data matrix (chapter 3) for each field at one time.

Appendix B

Programs Listing

```
! ***************************************************************************************************
! File Name: ConstructDataMatrix.f
! Author: Ahmad Alhamed
! Construct X[mxn] data matrix from grib files. This program reads n Grib files and extracts the  entire grid values for one
! forecast field at one time and then enlist these values as one column in the data matrix X. Note that this program does
! not actually decode/unpack the GRIB files. When compiles, it should be linked to the appropriate unpacker which can
! be obtained form ftp://nic.fb4.noaa.gov/pub/.
! ***************************************************************************************************
MODULE Global
    IMPLICIT NONE

    CHARACTER(LEN=40) :: FileOfFiles
        LOGICAL :: FilesNamesFromFile
        INTEGER :: ParmID, LevelType,LevelValue, Time1,Time2
END MODULE Global
! ========================================================================
PROGRAM GRIBreader
        USE Global
        IMPLICIT NONE

        INTEGER :: M, N, AllocatStatus, I, J
        REAL, ALLOCATABLE, DIMENSION(:,:) :: DataMatrix
        INTEGER IERR

        CALL GETARG(1, FileOfFiles)
        IF ( LEN(TRIM(FileOfFiles)) /= 0 ) THEN
                FilesNamesFromFile = .TRUE.
        ELSE
                FilesNamesFromFile = .FALSE.
        END IF
        PRINT '($,A)','Enter No. of runs OR No. of column in Data Matrix: '
        READ *, N
        PRINT '($,A)','Enter No. of rows in Data Matrix: '
        READ *, M
        ALLOCATE( DataMatrix(M,N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(DataMatrix) ** No Enough Memory ***"
        !CONSTRUCT DATA MATRIX FROM GRIB FILES
        CALL ConstructDataMatrix(M, N, DataMatrix)
        DEALLOCATE( DataMatrix, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(DataMatrix)** Deallocation Error***"
END PROGRAM GRIBreader
! ========================================================================
SUBROUTINE ConstructDataMatrix(Row, Col, DataMatrix)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: Row, Col
        REAL, DIMENSION(Row,Col), INTENT(OUT) :: DataMatrix
        REAL, ALLOCATABLE, DIMENSION(:) :: OneRun
        INTEGER:: J, AllocatStatus, K
        CHARACTER(LEN=44) :: FileName, String
        INTEGER I, IERR, JERR
        CHARACTER :: Ch

        ALLOCATE( OneRun(Row), STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(OneRun)** No Enough Memory ***"
        PRINT '($,A)','Enter Parameter ID to lookup: '
        READ *, ParmID
        PRINT '($,A)','Enter the Level Type: '
        READ *, LevelType
        PRINT '($,A)','Enter the Level Value: '
        READ *, LevelValue
        PRINT '($,A)','Enter the Time Range: '
        READ *, Time1, Time2
        IF(FilesNamesFromFile) THEN
                OPEN (UNIT=10,FILE=FileOfFiles,STATUS='OLD',&
                & ACCESS='SEQUENTIAL',FORM='FORMATTED',IOSTAT=IERR)
                IF (IERR.NE.0) THEN
```

```
                    PRINT 8 , FileOfFiles, IERR
5                   FORMAT ( ' ERROR OPENING UNIT=10, FILE NAME = ',A10,&
                    &  ', IOSTAT = ',I8)
                    STOP 5
                ENDIF
            ENDIF
            DO J=1, Col
              IF(FilesNamesFromFile) THEN
                READ (10, *) FileName
                PRINT 150 , J, FileName
150             FORMAT('Reading GRIB File for Run ',I2,' : ', A30)
              ELSE
                PRINT 200 , J
200             FORMAT('Enter GRIB File Name for Run ',I2,' : ',$)
                READ *, FileName
              ENDIF
                CALL Decode(FileName, Row, OneRun)
                DataMatrix(1:Row,J) = OneRun
            END DO
            PRINT *,"Data Matrix has been created successfully"
            PRINT '($,A)',"Do you want to save on file (Y/N)? "
            READ *, Ch
            IF ( Ch == 'Y' .OR. Ch == 'y')  THEN
                PRINT '($,A)','Enter File Name To Save Data Matrix:  '
                READ *, FileName
                OPEN (UNIT=3,FILE=FileName,STATUS='NEW',IOSTAT=IERR)
                IF (IERR /= 0) THEN
                    PRINT 8 , FileName, IERR
8                   FORMAT ( ' ERROR OPENING UNIT=3, FILE NAME = ',A15,&
                    &  ', IOSTAT = ',I8)
                    STOP 3
                ENDIF
                DO I=1, Row
                    WRITE (3,*) (DataMatrix(I,J),J=1,Col)
                END DO
                CLOSE (UNIT=3)
            END IF
            DEALLOCATE( OneRun, STAT = AllocatStatus )
            IF ( AllocatStatus /= 0 )  STOP  "(OneRun)** Deallocation Error ***"
END SUBROUTINE ConstructDataMatrix
! ========================================================================
SUBROUTINE Decode(NFILE, M, Run)
      USE Global
      IMPLICIT NONE
! ***********************************************************************
! Construct one column of the Data Matrix from the GRIB file
! Input:
!           GribFile : GRIB File name
!           PARM_ID : Parameter ID (see Table    , in GRIB manual )
!           M           : No. of rows in Data Matrix
!     Time    : Time 0,6th hour, 12th hour, ... , 48th hour
!           Flag        :
!             (1)       : Sum of the grid corresponded to the parameter
!             (2)       : Min. value of the grid
!             (3)       : Max. value of the grid
! Output
!     Run  : 1D array contains the values for one column of the
!                         Data Matrix
! ***********************************************************************
            INTEGER, INTENT(IN) :: M
            CHARACTER(LEN=44), INTENT(IN) :: NFILE
            REAL, DIMENSION(M), INTENT(OUT) :: Run

! local variables
            REAL :: FindSum, FindMin, FindMax  ! functions
            INTEGER, PARAMETER :: IMAXIN=200,JJMAXIN=200,JMAXIN=IMAXIN*JJMAXIN
            INTEGER, PARAMETER :: IBUFSIZE=15000000
            INTEGER :: KPDS(100),KGDS(200), KPTR(20), JSTAT(13)
            LOGICAL :: LIN(JMAXIN)
```

```
      REAL :: FLD(JMAXIN)
      INTEGER :: NDX, LUGBIN, IRCBYTE, IPARM, NDATA, KRET
      INTEGER :: KBYTES, MERR, IERR, JERR, JSTART, I1, JERR2, I
      CHARACTER IBUF(IBUFSIZE)*1,MSGA(200+17*JMAXIN/8)
      CHARACTER * 80  COMND
      CHARACTER * 10  JFILE
      CHARACTER * 80  LS
      CHARACTER :: Ch

      DATA  COMND /'ls -l                                '/
      DATA  JFILE /'unpkls.dat'/
      LUGBIN=11
! READ IN ENTIRE GRIB FILE
      COMND(7:50)  = NFILE(1:44)
      COMND(51:57) = ' | awk '
      COMND(58:58) = CHAR(39)
      COMND(59:68) = '{print $5}'
      COMND(69:69) = CHAR(39)
      COMND(70:70) = '>'
      COMND(71:80) = JFILE(1:10)
      CALL SYSTEM(COMND)

      OPEN (UNIT=9,FILE=JFILE,STATUS='OLD',&
      &  ACCESS='SEQUENTIAL',FORM='FORMATTED',IOSTAT=IERR)
      IF (IERR.NE.0) THEN
            PRINT 8 , JFILE, IERR
8           FORMAT ( ' ERROR OPENING UNIT=9, FILE NAME = ',A10,&
            & ', IOSTAT = ',I8)
            CALL W3TAGE('UNPKGRB1')
            STOP 8
      ENDIF

!*** To read the size of input file in bytes
      READ (9,'(A)',IOSTAT=JERR) ls
      IF (JERR.NE.0) THEN
            PRINT 9 , JFILE, JERR
9           FORMAT ( ' ERROR READING UNIT=9, FILE NAME = ',A10,&
            & ', IOSTAT = ',I8)
            CLOSE (UNIT=9)
            CALL W3TAGE('UNPKGRB1')
            STOP 9
      ENDIF
      CLOSE (UNIT=9)
      READ (ls(1:10),'(I10)') kbytes

! TEST TO SEE IF INPUT GRIB FILE IS TOO BIG
      IF (IBUFSIZE < KBYTES) THEN
            PRINT *,'GRIB INPUT FILE IS TOO BIG '
            PRINT *,'CHANGE PROGRAM SO PARAMETER IBUFSIZE IS LARGER'
            PRINT *,'THAN THE NUMBER OF BYTES IN THE FILE'
            CALL W3TAGE('UNPKGRB1')
            STOP 40
      END IF

! OPEN GRIB FILE

      OPEN (UNIT=LUGBIN,FILE=NFILE,STATUS='OLD',ACCESS='DIRECT',&
            &   FORM='UNFORMATTED',IOSTAT=MERR,RECL=KBYTES)

      IF (MERR /= 0) THEN
            PRINT *,'OPEN INPUT FILE ERROR ON FILE = ', NFILE
            PRINT *,'ERROR = ',MERR
            CALL W3TAGE('UNPKGRB1')
            STOP 20
      END IF
!   READ GRIB FILE
      READ(LUGBIN, REC=1, IOSTAT=JERR2) (IBUF(I),I=1,KBYTES)
      CLOSE (LUGBIN) ! Input GRIB File
! FIND  all 'GRIB' IN FILE, THEN DECODE
```

```
        JSTART = 1
        I=JSTART

        DO WHILE (JSTART.LT.KBYTES)

                DO WHILE (ICHAR(IBUF(I)) /= 71.AND.ICHAR(IBUF(I+1)) /= 82&
                &.AND.ICHAR(IBUF(I+2)) /= 73.AND.ICHAR(IBUF(I+3)) /= 66)
                        I=I+1
                ENDDO
                IF (ICHAR(IBUF(I+7)) == 1) THEN
                        I1=I+7
! PARAMETER (GRIB doc: table 2)
                        IPARM=ICHAR(IBUF(I1+9))
!  LENGTH OF GRIB MESSAGE
                        IRCBYTE = ICHAR(IBUF(I+4))*65536+ICHAR(IBUF(I+5))&
                                &   *256+ ICHAR(IBUF(I+6))
                        JSTART = I + IRCBYTE
! Decode if needed parameter is found
                        IF (IPARM == ParmID)  THEN
                          CALL XMOVEX(MSGA,IBUF(I),IRCBYTE)
! DECODE GRIB MESSAGE
                          CALL W3FI63(MSGA,KPDS,KGDS,LIN,FLD,KPTR,KRET)
                          NDATA=KPTR(10)
                          IF(KRET /= 0) THEN
                            PRINT *,'W3FI63 GRIB UNPACKER ERROR =',KRET
                            STOP 30
                          END IF
                          IF(KPDS(5)==ParmID .AND. KPDS(6)==LevelType .AND. &
                                & KPDS(7)==LevelValue .AND. KPDS(14)==Time1.AND. &
                                & KPDS(15)==Time2) THEN
                                PRINT *,'Record found at time range:',KPDS(14),KPDS(15)
                                DO NDX=1, 6045
!  ***************************************For Percipitation PRCP ************
                                        IF(ParmID == 61) THEN
                                                IF ( FLD(NDX) > 0.0 ) THEN
                                                        Run(NDX) = FLD(NDX)
                                                ELSE
                                                        Run(NDX) = 0.0
                                                END IF
                                        ELSE
!***************************************For others**************************
                                                Run(NDX) = FLD(NDX)
                                        END IF
                                END DO
                          END IF
                        END IF
                        I = JSTART
                ENDIF
        END DO
END SUBROUTINE Decode
! ======================================================================
REAL FUNCTION FindSum(I, J, Grid)
    IMPLICIT NONE
! ***********************************************************************
! Returns sum of the values of the grid points
! INPUTS:
!       I, J : Dimensions of grid
!       Grid :
! ***********************************************************************

        INTEGER, INTENT(IN) :: I, J
        REAL, DIMENSION(40000), INTENT(IN) :: Grid
        REAL Sum
        INTEGER K, P

        K = I * J
        Sum = 0.0
        DO P=1, K
                IF ( Grid(P) > 0.0 )  Sum = Sum + Grid(P)
```

```
        END DO
        FindSum = Sum
END FUNCTION FindSum
! ======================================================================
REAL FUNCTION FindMin(I, J, Grid)
    IMPLICIT NONE
! ***********************************************************************
! Returns min. value of the grid points
! INPUTS:
!       I, J : Dimensions of grid
!       Grid :
! ***********************************************************************

    INTEGER, INTENT(IN) :: I, J
    REAL, DIMENSION(40000), INTENT(IN) :: Grid
    REAL Min
    INTEGER K, P

    K = I * J
    Min = 1000000000
    DO P=1, K
        IF (Grid(P)>0.0 .AND. Grid(P)<Min) Min = Grid(P)
    END DO
    FindMin = Min
END FUNCTION FindMin
! ======================================================================
REAL FUNCTION FindMax(I, J, Grid)
    IMPLICIT NONE
! ***********************************************************************
! Returns max. value of the grid points
! INPUTS:
!       I, J : Dimensions of grid
!       Grid :
! ***********************************************************************

    INTEGER, INTENT(IN) :: I, J
    REAL, DIMENSION(40000), INTENT(IN) :: Grid
    REAL Max
    INTEGER K, P

    K = I * J
    Max = -1000000000
    DO P=1, K
        IF (Grid(P)>0.0 .AND. Grid(P)>Max) Max = Grid(P)
    END DO
    FindMax = Max
END FUNCTION FindMax
! ======================================================================
```

```
! ***************************************************************************************************************
! File Name: hca.f
! Author: Ahmad Alhamed
! This program performs Hierarchical Cluster Analysis. It calls other subroutines and functions from other files,
! resemblance.f, agglomerative.f, cophenetic.f, tree.f, and CAlib.f. When compiling this program, it should be linked with
! the NCAR graphics library which is used to draw the clustering trees. Also, it should be linked with IMSL library which
! is used to compute the correlation.
! ***************************************************************************************************************
MODULE Global
        IMPLICIT NONE

        TYPE Merge
         INTEGER :: MergedCluster1, MergedCluster2, NewLable
         REAL :: ResDegree
        END TYPE Merge
        LOGICAL :: DataWritten, ResWritten
        INTEGER :: OutputFormat, TPP
!                                       TPP : Trees Per Page
        INTEGER, PARAMETER :: OTOMOP=1, OTAMSP=2, OTAMOP=3, ATOMOP=4
                ! OTOMOP : One Time One Method One Page
                ! OTAMSP : One Time All Methods Seperate Pages
                ! OTAMOP : One Time All Methods One Page
                ! ATOMOP : All Time One Method One Page

        INTEGER, PARAMETER :: NoOfMethods=7
        INTEGER :: StartingTime, NoOfTimes, Interval
        TYPE(Merge), ALLOCATABLE, DIMENSION(:) :: AllHierarchy
        REAL, ALLOCATABLE, DIMENSION(:)  :: AllCophCoef
        INTEGER :: AllHierarchyCntr, AllCophCoefCntr
        INTEGER :: ReadLable
        CHARACTER(LEN=3), ALLOCATABLE, DIMENSION(:)  : LABLES
END MODULE Global
! ===========================================================================
PROGRAM HCA
    IMPLICIT NONE

    CHARACTER(LEN=20) :: argv

        CALL GETARG(1, argv)
        IF ( LEN(TRIM(argv)) == 0 ) THEN
                CALL UserInterface
        ELSE
                CALL ControlFile(argv)
        END IF
END PROGRAM HCA
! ===========================================================================
SUBROUTINE UserInterface
    USE Global
        IMPLICIT NONE
! ***************************************************************
!  Show the user interface, and obtain all required input
! ***************************************************************

        INTEGER :: MainMenu, NormalizeMenu, ResCoeffMenu, &
                & ClusteringMethodMenu ! functions
        INTEGER :: M, N, Choice, Coef, Method, Normalize, IERR, AllocatStatus, I
        CHARACTER(LEN=50) :: OutputFile, InputFile
        CHARACTER(LEN=25) :: Prefix
        CHARACTER(LEN=70) :: Title

        PRINT '($,A)','Enter Title: '
        READ (*, FMT='(A)') Title
        ! main menu
        Choice = MainMenu()
        PRINT *
        PRINT '($,A)','Enter No. of runs OR No. of column in Data Matrix: '
        READ *, N
        PRINT '($,A)','Enter No. of attributes OR No. of rows in Data Matrix: '
```

```
        READ *, M
        IF (Choice == 1 .OR. Choice == 3)  THEN
                PRINT '($,A)','Enter Input File Name:  '
                READ *, InputFile
        ELSE
                InputFile = '*******'
        END IF
        ! normalization menu
        Normalize = NormalizeMenu()
        ! Res. coeff. menu
        Coef = ResCoeffMenu()
        ! clustering method menu
        Method = ClusteringMethodMenu()
        PRINT '($,A)','Enter prefix for File Name To Save Tree on PS format:'
        READ *, Prefix
        ALLOCATE( LABLES(N), STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 )  STOP  "(LABLES) ** No EnoughMemory **"
        DO I=1,N
                WRITE(LABLES(I), '(I3)' ) I
        END DO
! ***************************
        IF(Method == 8) THEN
                OutputFormat = OTAMOP
        ELSE
                OutputFormat = OTOMOP
        END IF
! ***************************
        OutputFile = TRIM(Prefix)//'CLUST.out'
        OPEN (UNIT=7,FILE=OutputFile,STATUS='NEW',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 7 , OutputFile, IERR
7               FORMAT (' ERROR OPENING UNIT=7, FILE NAME ='&
                & ,A13,', IOSTAT = ',I8)
                STOP 7
        END IF

        ! initialize counters to 1
        AllHierarchyCntr = 1
        AllCophCoefCntr = 1
        SELECT CASE(OutputFormat)
        CASE (OTOMOP)
                ALLOCATE( AllHierarchy(N-1), STAT = AllocatStatus )
                IF ( AllocatStatus /= 0 )  STOP  "(AllHierarchy) ** No EnoughMemory **"
                ALLOCATE( AllCophCoef(1), STAT = AllocatStatus )
                IF ( AllocatStatus /= 0 )  STOP  "(AllCophCoef) ** No EnoughMemory **"

                Call Start(Title, Choice, N, M, Normalize, Coef, Method, InputFile)
        CASE (OTAMOP)
                ALLOCATE( AllHierarchy( (N-1)*NoOfMethods ), STAT = AllocatStatus )
                IF ( AllocatStatus /= 0 )  STOP  "(AllHierarchy) ** No EnoughMemory **"
                ALLOCATE( AllCophCoef(NoOfMethods), STAT = AllocatStatus )
                IF ( AllocatStatus /= 0 )  STOP  "(AllCophCoef) ** No EnoughMemory **"
                Call Start(Title, Choice, N, M, Normalize, Coef, Method, InputFile)
        END SELECT
        ! draw tree or trees from AllHierarchy
        CALL DrawTrees(N, Coef, Method, Normalize, Title, Prefix)
        DEALLOCATE( AllHierarchy, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(AllHierarchy) **DeallocationError**"
        DEALLOCATE( AllCophCoef, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(AllCophCoef) **DeallocationError**"
        CLOSE (UNIT=7)
END SUBROUTINE UserInterface
! ========================================================================
SUBROUTINE ControlFile(ControlFileName)
        USE Global
        IMPLICIT NONE
! ***************************************************************
!     Read the required input from input file called 'Control File'
! ***************************************************************
```

```
      CHARACTER(LEN=*), INTENT(IN) :: ControlFileName
      INTEGER :: M, N, Choice, Coef, Method, Normalize,IERR, I, J, Time, AllocatStatus
      CHARACTER(LEN=50) :: OutputFile, InputFile, FileName
      CHARACTER(LEN=25) :: Prefix
      CHARACTER(LEN=70) :: Title, NewTitle
      REAL :: X, Y

      OPEN (UNIT=10,FILE=ControlFileName,STATUS='OLD',IOSTAT=IERR)
      IF (IERR /= 0) THEN
            PRINT 15 , ControlFileName, IERR
15          FORMAT ( ' ERROR OPENING UNIT=10, FILE NAME = ',A15,&
            & ', IOSTAT = ',I8)
            STOP 15
      ENDIF
      READ (10, *) OutputFormat, TPP
      IF (OutputFormat == 4) THEN
            READ (10, *) StartingTime, NoOfTimes, Interval
      END IF
      READ (10, FMT='(A)') Title
      READ (10, *) Choice
      READ (10, *) N, M
      READ (10, *) InputFile
      READ (10, *) Normalize
      READ (10, *) Coef
      READ (10, *) Method
      READ (10, *) Prefix
      READ (10, *) ReadLable
      ALLOCATE( LABLES(N), STAT = AllocatStatus )
      IF ( AllocatStatus /= 0 ) STOP "(LABLES) ** No EnoughMemory ***
      IF (ReadLable == 1) THEN
            READ (10, *) (LABLES(I), I=1,N)
      ELSE
            DO I=1,N
                  WRITE(LABLES(I), '(I3)' ) I
            END DO
      END IF
      OutputFile = TRIM(Prefix)//'CLUST.out'
      OPEN (UNIT=7,FILE=OutputFile,STATUS='NEW',IOSTAT=IERR)
      IF (IERR /= 0) THEN
            PRINT 7 , OutputFile, IERR
7           FORMAT (' ERROR OPENING UNIT=7, FILE NAME ='&
            & ,A13,', IOSTAT = ',I8)
            STOP 7
      END IF
      ! initialize counters to 1
      AllHierarchyCntr = 1
      AllCophCoefCntr = 1
      SELECT CASE(OutputFormat)
      CASE (OTOMOP)
      ALLOCATE( AllHierarchy(N-1), STAT = AllocatStatus )
      IF ( AllocatStatus /= 0 ) STOP "(AllHierarchy) ** No EnoughMemory ***
      ALLOCATE( AllCophCoef(1), STAT = AllocatStatus )
      IF ( AllocatStatus /= 0 ) STOP "(AllCophCoef) ** No EnoughMemory ***
      Call Start(Title, Choice, N, M, Normalize, Coef, Method, InputFile)
      CASE (OTAMSP, OTAMOP)
            ALLOCATE( AllHierarchy( (N-1)*NoOfMethods ), STAT = AllocatStatus )
            IF ( AllocatStatus /= 0 ) STOP "(AllHierarchy) ** No EnoughMemory ***
            ALLOCATE( AllCophCoef(NoOfMethods), STAT = AllocatStatus )
            IF ( AllocatStatus /= 0 ) STOP "(AllCophCoef) ** No EnoughMemory ***
            Call Start(Title, Choice, N, M, Normalize, Coef, Method, InputFile)
      CASE (ATOMOP)
            ALLOCATE( AllHierarchy( (N-1)*NoOfTimes ), STAT = AllocatStatus )
            IF ( AllocatStatus /= 0 ) STOP "(AllHierarchy) ** No EnoughMemory ***
            ALLOCATE( AllCophCoef(NoOfTimes), STAT = AllocatStatus )
            IF ( AllocatStatus /= 0 ) STOP "(AllCophCoef) ** No EnoughMemory ***
            OPEN(UNIT=50,FILE=InputFile,STATUS='OLD',IOSTAT=IERR)
            IF (IERR /= 0) THEN
                  PRINT 16 , InputFile, IERR
```

```
16                      FORMAT ( ' ERROR OPENING UNIT=50, FILE NAME = ',A15, ', IOSTAT = ',I8)
                        STOP 16
                ENDIF
                Time = StartingTime
                DO I=1, NoOfTimes
                        READ (50, *) FileName
                        SELECT CASE (MOD(Time, 10))
                        CASE (1)
                          WRITE(NewTitle, 11) TRIM(Title), Time
                        CASE (2)
                          IF (Time == 12) THEN
                            WRITE(NewTitle, 14) TRIM(Title), Time
                          ELSE
                            WRITE(NewTitle, 12) TRIM(Title), Time
                          END IF
                        CASE (3)
                          WRITE(NewTitle, 13) TRIM(Title), Time
                        CASE DEFAULT
                          WRITE(NewTitle, 14) TRIM(Title), Time
                        END SELECT
                        Time = Time + Interval
                        CALL Start(NewTitle, Choice, N, M, &
                        & Normalize, Coef, Method, FileName)
                END DO
11              FORMAT ( A,' at the ',I2,'st hr' )
12              FORMAT ( A,' at the ',I2,'nd hr' )
13              FORMAT ( A,' at the ',I2,'rd hr' )
14              FORMAT ( A,' at the ',I2,'th hr' )
        END SELECT
        ! draw tree or trees from AllHierarchy
        CALL DrawTrees(N, Coef, Method, Normalize, Title, Prefix)
        CLOSE (UNIT=7)
        DEALLOCATE( AllHierarchy, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(AllHierarchy) **DeallocationError***
        DEALLOCATE( AllCophCoef, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(AllCophCoef) **DeallocationError***
END SUBROUTINE ControlFile
! ======================================================================
SUBROUTINE Start(Title, Choice, N, M, Normalize, Coef, Method, InputFileName)
        USE Global
        IMPLICIT NONE

        CHARACTER(LEN=*), INTENT(IN) :: InputFileName
        CHARACTER(LEN=*), INTENT(IN) :: Title
        INTEGER, INTENT(IN) :: N, M, Choice, Normalize, Coef, Method
        INTEGER :: AllocatStatus, I, J, NUM
        REAL, ALLOCATABLE, DIMENSION(:,:) :: DataMatrix
        REAL, ALLOCATABLE, DIMENSION(:)   :: ResemblanceMatrix
        CHARACTER(LEN=10) :: CoefAbbr

        DataWritten = .FALSE.
        ResWritten = .FALSE.
        WRITE (7, *) '    ', Title
        WRITE (7, *) '  ----------------------------------------'
        WRITE (7, *)
        IF (Choice == 1 .OR. Choice == 2) THEN
                ALLOCATE( DataMatrix(M,N), STAT = AllocatStatus )
                IF (AllocatStatus/= 0 ) STOP  "(DataMatrix) ** No Enough Memory ***
        END IF
    ALLOCATE( ResemblanceMatrix(N*(N-1)/2), STAT = AllocatStatus )
    IF (AllocatStatus/= 0 ) STOP  "(ResemblanceMatrix) ** No Enough Memory ***
        SELECT CASE(Choice)
        CASE (1)
                !READ DATA MATRIX FROM INPUT FILE
                CALL ReadDataMatrixFromFile(M, N, DataMatrix, InputFileName)
        CASE (2)
                !READ DATA MATRIX FROM KEYBOARD
                PRINT *,'Enter Data Matrix elements in rowwise order:'
                READ *, ((DataMatrix(I,J), J=1,N),I=1,M)
```

```
        CASE (3)
                !READ RESEMBLANCE MATRIX FROM INPUT FILE
                CALL ReadResemblanceMatrixFromFile(N, ResemblanceMatrix, InputFileName)
        CASE (4)
                !READ RESEMBLANCE MATRIX FROM KEYBOARD
                PRINT *,'Enter Lower Triangular (Off diagnal) of Resemblance'
                PRINT *,'Matrix elements in columnwise order:'
                READ *, (ResemblanceMatrix(I),I=1, N*(N-1)/2 )
        CASE (5)
                STOP
        END SELECT
        IF (Choice==1.OR.Choice==2) THEN
                ! print data matrix, and calculate resemblance matrix
                IF(M<=20.AND.N<=10.AND.(.NOT.DataWritten))  THEN
                        CALL PrintDataMatrix(M, N, DataMatrix)
                        DataWritten = .TRUE.
                END IF
                ! normalize if requested
                SELECT CASE(Normalize)
                CASE (1)
                        CALL CenterByColMean(M, N, DataMatrix)
                CASE (2)
                        CALL NormalizeByColMax(M, N, DataMatrix)
                CASE (3)
                        CALL NormalizeByColMinMax(M, N, DataMatrix)
                CASE (4)
                        CALL NormalizeByMax(M, N, DataMatrix)
                END SELECT
                ! calculating res matrix
                CALL CalculateResemblanceMatrix(Coef, M, N, DataMatrix, ResemblanceMatrix)
        END IF
    ! print resemblance matrix
        IF(.NOT.ResWritten) THEN
                CALL PrintResemblanceMatrix(N, ResemblanceMatrix)
                ResWritten = .TRUE.
        END IF
! ************************************************************************
! if coefficient is correlation, we need the absolute values only. That's because
! the sign does not matter with respect to the significance of similarity.
        IF (Coef == 4)  THEN
                ResemblanceMatrix = ABS(ResemblanceMatrix)
        END IF
! ************************************************************************
        SELECT CASE (OutputFormat)
        CASE (OTOMOP)
                CALL DoClustering(N, Coef, Method, ResemblanceMatrix)
        CASE (OTAMSP, OTAMOP)
                DO I=1, 7
                        CALL DoClustering(N, Coef, I, ResemblanceMatrix)
                END DO
        CASE (ATOMOP)
                CALL DoClustering(N, Coef, Method, ResemblanceMatrix)
        END SELECT
        IF (Choice == 1 .OR. Choice == 2)  THEN
                DEALLOCATE( DataMatrix, STAT = AllocatStatus )
                IF ( AllocatStatus /= 0 )  STOP  "(DataMatrix)" Deallocation Error"
        END IF
        DEALLOCATE( ResemblanceMatrix, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 )  STOP  "(ResemblanceMatrix)" Deallocation Error"
END SUBROUTINE Start
! ======================================================================
INTEGER FUNCTION MainMenu()
        IMPLICIT NONE

        INTEGER :: Choice

        PRINT *,'-----------------------------------------'
        PRINT *,'              Main Menu'
        PRINT *,'(1) Read Data Matrix from Input file'
```

```
        PRINT *,'(2) Read Data Matrix from Keyboard'
        PRINT *,'(3) Read Resemblance Matrix from Input file'
        PRINT *,'(4) Read Resemblance Matrix from Keyboard'
        PRINT *,'(5) Exit'
        PRINT *,'-------------------------------------'
        PRINT '($,A)',' Enter Choice ===> '
        DO
                READ *, Choice
                IF (Choice >= 1 .AND. Choice <= 5) EXIT
        END DO
        IF (Choice == 5) STOP
        MainMenu = choice
END FUNCTION MainMenu
! ============================================================================
INTEGER FUNCTION ResCoeffMenu()
        IMPLICIT NONE

        INTEGER :: Coef

        PRINT *
        PRINT *,'-------------------------------'
        PRINT *,'Resemblance Coefficint:'
        PRINT *,' (1) Euclidean Distance'
        PRINT *,' (2) Square Euclidean Distance'
        PRINT *,' (3) Average Euclidean Distance'
        PRINT *,' (4) Correlation'
        PRINT *,' (5) Euclidean Similarity'
        PRINT *,' (6) Manhattan Distance'
        PRINT *,' (7) Mahalanobis Distance'
        PRINT *,' (8) Sup Distance'
        PRINT *,'-------------------------------'
        PRINT '($,A)',' Enter Choice ===> '
        DO
                READ *, Coef
                IF (Coef >= 1 .AND. Coef <= 8) EXIT
        END DO
        ResCoeffMenu = Coef
END FUNCTION ResCoeffMenu
! ============================================================================
INTEGER FUNCTION ClustenngMethodMenu()
        IMPLICIT NONE

        INTEGER :: Method

        PRINT *,'-------------------------------'
        PRINT *,'Clustenng Method:'
        PRINT *,' (1) SLINK (Single Linkage)'
        PRINT *,' (2) CLINK (Complete Linkage)'
        PRINT *,' (3) UPGMA (group average)'
        PRINT *,' (4) WPGMA (weighted average)'
        PRINT *,' (5) UPGMC (unweighted centroid)'
        PRINT *,' (6) WPGMC (weighted centroid)'
        PRINT *," (7) Ward's method (minimum variance)"
        PRINT *," (8) All above methods"
        PRINT *,'-------------------------------'
        PRINT '($,A)',' Enter Choice ===> '
        DO
                READ *, Method
                IF (Method >= 1 .AND. Method <= 8) EXIT
        END DO
        ClusteringMethodMenu = Method
END FUNCTION ClusteringMethodMenu
! ============================================================================
SUBROUTINE DoClustering(N, Coef, Method, ResemblanceMatrix)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: N, Coef, Method
        REAL, DIMENSION(N*(N-1)/2), INTENT(IN) :: ResemblanceMatrix
```

```
        TYPE(Merge), ALLOCATABLE, DIMENSION(:) :: Hierarchy
        INTEGER :: I, AllocatStatus
        LOGICAL :: Dissim
        REAL :: CutAt, CophCorrCoef
        CHARACTER(LEN=10) :: CoefAbbr, MethodAbbr

        ALLOCATE( Hierarchy(N-1), STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Hierarchy) "" No EnoughMemory """
        CALL WhatCoefficient(Coef,CoefAbbr,Dissim)
        CALL WhatClusteringMethod( Method, MethodAbbr)
        WRITE (7, *) MethodAbbr
        CALL Agglomerative(N, Method, Coef, Dissim, ResemblanceMatrix, Hierarchy)
        ! add this Hierarchy to AllHierarchy
        DO I = 1, N-1
                AllHierarchy(AllHierarchyCntr) = Hierarchy(I)
                AllHierarchyCntr = AllHierarchyCntr + 1
        END DO
        ! compute cophenetic correlation
        CALL Cophenetic(N, Hierarchy, ResemblanceMatrix, CophCorrCoef)
        ! add this CophCorrCoef to AllCophCoef
        AllCophCoef(AllCophCoefCntr) = CophCorrCoef
        AllCophCoefCntr = AllCophCoefCntr + 1

        DEALLOCATE( Hierarchy, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Hierarchy) "" DeallocationError"""
END SUBROUTINE DoClustering
! ========================================================================
SUBROUTINE OpenGKS(Prefix)
        USE Global
        IMPLICIT NONE

        CHARACTER(LEN=*), INTENT(IN) :: Prefix
        CHARACTER(LEN=50) :: FileName

        ! Open GKS.
        CALL GOPKS(6,0)
        ! to overide the default ps name
        WRITE(FileName, * ) TRIM(prefix), '.ps'
        CALL NGSETC('ME',FileName)
        IF (OutputFormat==ATOMOP .OR. OutputFormat==OTAMOP) THEN
                ! Specify the output position for the next PostScript
                ! workstation to be opened. These calls must appear before
                ! the call to open the workstation.
                CALL NGSETI('LX', 123)
                CALL NGSETI('LY', 87)
                CALL NGSETI('UX',525)
                CALL NGSETI('UY',705)
        END IF
        ! open workstation of type PS in portrait mode
        CALL GOPWK(1,2,23)
        ! activate opened workstation
        CALL GACWK(1)
END SUBROUTINE OpenGKS
! ========================================================================
SUBROUTINE CloseGKS
        IMPLICIT NONE

        ! Deactivate and close workstation
        CALL GDAWK(1)
        CALL GCLWK(1)
        ! close GKS
        CALL GCLKS
END SUBROUTINE CloseGKS
! ========================================================================
SUBROUTINE BOX(X,Y,szX,szY)
    IMPLICIT NONE
! ************************************************************************
! Draw a square box with lower left corner at (X,Y) and size SZ x SZ
! and put the label LAB in the center.
```

```
! ***********************************************************************

      REAL, INTENT(IN) :: X, Y, szX, szY
      REAL, DIMENSION(5) :: A,B

            ! Draw box.
      A(1) = X
      B(1) = Y
      A(2) = X+szX
      B(2) = Y
      A(3) = A(2)
      B(3) = Y+szY
      A(4) = X
      B(4) = B(3)
      A(5) = X
      B(5) = Y
      CALL GPL(5,A,B)
END SUBROUTINE BOX
! ======================================================================
SUBROUTINE WhatCoefficient(CoefIndex,CoefAbb,Dis)
      IMPLICIT NONE

      INTEGER, INTENT(IN) :: CoefIndex
      CHARACTER(LEN=*), INTENT(OUT) :: CoefAbb
      LOGICAL, INTENT(OUT) :: Dis

      SELECT CASE(CoefIndex)
      CASE (1)
                  CoefAbb = 'Euc'
                  Dis = .TRUE.
      CASE (2)
                  CoefAbb = 'sqEuc'
                  Dis = .TRUE.
      CASE (3)
                  CoefAbb = 'aveEuc'
                  Dis = .TRUE.
      CASE (4)
                  CoefAbb = 'Corr'
                  Dis = .FALSE.
      CASE (5)
                  CoefAbb = 'EucSimi'
                  Dis =  FALSE.
      CASE (6)
                  CoefAbb = 'Manhattan'
                  Dis = .TRUE.
      CASE (7)
                  CoefAbb = 'Mahalanobis'
                  Dis = .TRUE.
      CASE (8)
                  CoefAbb = 'Sup Distance'
                  Dis = .TRUE.
      END SELECT
END SUBROUTINE WhatCoefficient
! ======================================================================
SUBROUTINE WhatClusteringMethod( Method, ClusteringMethod)
      IMPLICIT NONE

      INTEGER, INTENT(IN) :: Method
      CHARACTER(LEN=*), INTENT(OUT) :: ClusteringMethod

      SELECT CASE(Method)
      CASE (1)
            ClusteringMethod = "SLINK"
      CASE (2)
            ClusteringMethod = "CLINK"
      CASE (3)
            ClusteringMethod = "UPGMA"
      CASE (4)
            ClusteringMethod = "WPGMA"
```

```
    CASE (5)
        ClusteringMethod = "UPGMC"
    CASE (6)
        ClusteringMethod = "WPGMC"
    CASE (7)
        ClusteringMethod = "Ward's"
    END SELECT
END SUBROUTINE WhatClusteringMethod
! ========================================================================
SUBROUTINE DrawTrees(N, Coef, Method, Normalize, Title, Prefix)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: N, Coef, Method, Normalize
        CHARACTER(LEN=*), INTENT(IN) :: Title, Prefix
        TYPE(Merge), ALLOCATABLE, DIMENSION(:) :: OneHierarchy
        INTEGER :: AllocatStatus, I, J, K, Cntr, NUM, Time, Cntr2, TLine
        CHARACTER(LEN=50) :: TreeTitle, SubTitle
        REAL :: FindMaxOfAllHierarch  ! function
        REAL :: X, Y, CophCoef, Scale, Hight, Width, BoxH, BoxW
        LOGICAL :: Dissim

        ALLOCATE( OneHierarchy(N-1), STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP  "(OneHierarchy)""No Enough Memory"""
        ! open and setup GKS
        CALL OpenGKS(Prefix)
        SELECT CASE (OutputFormat)
        CASE (OTOMOP)
                SubTitle = ''
                CALL ConstructSubTitle(Coef, Method, Normalize, SubTitle, Dissim)
                CALL PLCHLQ( 0.5, 0.98,TRIM(Title), 18.0,0.,0.)
                CALL PLCHLQ( 0.5, 0.95,TRIM(SubTitle),13.0,0.,0.)
                OneHierarchy = AllHierarchy
                ! draw clustering tree on PS file
                CALL Tree(N, OneHierarchy, '', Dissim, 0.0, 0.0, 0.0, AllCophCoef(1) )
                CALL FRAME
        CASE (OTAMSP)
                Cntr = 1
                DO I=1, NoOfMethods
                        DO J=1, N-1
                                OneHierarchy(J) = AllHierarchy(cntr)
                                Cntr = Cntr + 1
                        END DO
                        SubTitle = ''
                        CALL ConstructSubTitle(Coef, I, Normalize, SubTitle, Dissim)
                        CALL PLCHLQ( 0.5, 0.98,TRIM(Title), 18.0,0.,0.)
                        CALL PLCHLQ( 0.5, 0.95,TRIM(SubTitle),13.0,0.,0.)
                        CALL Tree(N, OneHierarchy, '', Dissim, 0.0, 0.0, 0.0, AllCophCoef(I) )
                        CALL FRAME
                END DO
        CASE (OTAMOP)
                !Draw three rows and three columns of square boxes.
                Cntr = 1
                Cntr2 = 1
                NUM = 0
                CALL ConstructSubTitle(Coef, Method, Normalize, SubTitle,Dissim)
                CALL PLCHLQ( 0.5, 0.98,TRIM(Title), 18.0,0.,0.)
                CALL PLCHLQ( 0.5, 0.95,TRIM(SubTitle),13.0,0.,0.)
                DO J=3,1,-1
                        Y = 0.310*REAL(J-1)
                        DO  I=1,3
                                X = 0.3333*REAL(I-1)
                                NUM = NUM+1
                                IF (NUM > NoOfMethods) THEN
                                        EXIT
                                END IF
                                CALL BOX(X, Y, 0.32, 0.30)
                                SubTitle = ''
                                CALL WhatClusteringMethod(NUM,TreeTitle)
```

```
                        DO K=1, N-1
                                OneHierarchy(K) = AllHierarchy(cntr)
                                Cntr = Cntr + 1
                        END DO
                        CophCoef = AllCophCoef(Cntr2)
                        CALL Tree(N, OneHierarchy, TreeTitle, Dissim, 0.0, X, Y, CophCoef)
                        Cntr2 = Cntr2 + 1
                END DO
        END DO
        CALL FRAME
CASE (ATOMOP)
        ! find max degree in AllHierarchy
        Scale = FindMaxOfAllHierarch( (N-1)*NoOfTimes )
        SubTitle = ''
        CALL ConstructSubTitle(Coef, Method, Normalize, SubTitle, Dissim)
        IF (TPP == 4) THEN
          TLine = 2
          Hight = 0.460
          Width = 0.490
          BoxH = 0.450
          BoxW = 0.480
        END IF
        IF (TPP == 9) THEN
          TLine = 3
          Hight = 0.310
          Width = 0.330
          BoxH = 0.30
          BoxW = 0.320
        END IF
        Cntr = 1
        Cntr2 = 1
        Time = StartingTime
        NUM = 0
        DO WHILE (.NOT.(NUM > NoOfTimes))
          ! display title and subtitle
          CALL BOX(0.0, 0.0, 1.0, 1.0)
          CALL PLCHLQ( 0.5, 0.98,TRIM(Title), 18.0,0.,0.)
          CALL PLCHLQ( 0.5, 0.95,TRIM(SubTitle),13.0,0.,0.)
          DO J=TLine,1,-1
                Y = 0.01 + Hight*REAL(J-1)
                DO  I=1,TLine
                        X = 0.01 + Width*REAL(I-1)
                        NUM = NUM+1
                        IF (NUM > NoOfTimes) THEN
                                EXIT
                        END IF
                        CALL BOX(X, Y, BoxW, BoxH)
                        SELECT CASE (MOD(Time, 10))
                        CASE (1)
                                WRITE(TreeTitle, 11) Time
                        CASE (2)
                          IF (Time == 12) THEN
                                WRITE(TreeTitle, 14) Time
                          ELSE
                                WRITE(TreeTitle, 12) Time
                          END IF
                        CASE (3)
                                WRITE(TreeTitle, 13) Time
                        CASE DEFAULT
                                WRITE(TreeTitle, 14) Time
                        END SELECT
                        DO K=1, N-1
                                OneHierarchy(K) = AllHierarchy(cntr)
                                Cntr = Cntr + 1
                        END DO
                        CophCoef = AllCophCoef(Cntr2)
                        CALL Tree(N, OneHierarchy, TreeTitle, Dissim,Scale,X,Y,CophCoef)
                        Time = Time + Interval
                        Cntr2 = Cntr2 + 1
```

```
                        END DO
                    END DO
                    CALL FRAME
                END DO
11              FORMAT ( 'At the ',I2,'st hr' )
12              FORMAT ( 'At the ',I2,'nd hr' )
13              FORMAT ( 'At the ',I2,'rd hr' )
14              FORMAT ( 'At the ',I2,'th hr' )
        END SELECT
        ! close GKS
        CALL CloseGKS
        DEALLOCATE( OneHierarchy, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 )  STOP  "(OneHierarchy) ""DeallocationError"""
END SUBROUTINE DrawTrees
! ========================================================================
SUBROUTINE ConstructSubTitle(Coef, Method, Normalize, SubTitle, Dissim)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: Coef, Method, Normalize
        CHARACTER(LEN=*), INTENT(OUT) :: SubTitle
        LOGICAL, INTENT(OUT) :: Dissim
        CHARACTER(LEN=10) :: CoefAbbr, MethodAbbr
        CHARACTER(LEN=30) :: SubTitle0

        SELECT CASE(Normalize)
        CASE(0)
                SubTitle0 = ''
        CASE (1)
                SubTitle0 = 'Centered (col) - '
        CASE (2)
                SubTitle0 = 'Normalized (col max) - '
        CASE (3)
                SubTitle0 = 'Normalized (col min,max) - '
        CASE (4)
                SubTitle0 = 'Normalized (max) - '
        END SELECT
        CALL WhatCoefficient(Coef,CoefAbbr,Dissim)
        CALL WhatClusteringMethod( Method, MethodAbbr)
        SubTitle = ''
        IF (OutputFormat == OTAMOP)  THEN
                WRITE(SubTitle, 10) TRIM(SubTitle0), TRIM(CoefAbbr)
        ELSE
                WRITE(SubTitle, 20) TRIM(SubTitle0), TRIM(CoefAbbr), TRIM(MethodAbbr)
        END IF
10      FORMAT (A, A)
20      FORMAT (A, A, ' - ', A)
END SUBROUTINE ConstructSubTitle
! ========================================================================
REAL FUNCTION FindMaxOfAllHierarch(Size)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: Size
        REAL :: MaxDegree
        INTEGER :: I

        MaxDegree = AllHierarchy(1)%ResDegree
        DO I=2, Size
                IF( AllHierarchy(I)%ResDegree > MaxDegree ) THEN
                        MaxDegree = AllHierarchy(I)%ResDegree
                END IF
        END DO
        FindMaxOfAllHierarch = MaxDegree
END FUNCTION FindMaxOfAllHierarch
! ========================================================================
```

```
! ********************************************************************************************************************
! File Name: resemblance.f
! Author: Ahmad Alhamed
!
! Calculates resemblance matrix using different coefficints
! Input:
!         Coef  : Coefficint of resemblance
!            (1) Euclidean Distance
!            (2) SquareEuclidean Distance
!            (4) Correlation
!            (5) Euclidean Similanty
!            (6) Manhattan Distance
!            (7) Mahalanobis Distance
!            (8) Sup Distance
!         M     : No. of attributes (rows) in the data matrix
!         N     : No. of objects (columns) in the data matrix
!         Data  : MxN data matrix (2D array)
! Output
!         Resemblance : 1D array contains the lower traingle of resemblance matrix in a col-major order
! ********************************************************************************************************************
SUBROUTINE CalculateResemblanceMatrix(Coef, M, N, Data, Resemblance)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: Coef, M, N
        REAL, DIMENSION(M,N), INTENT(IN) :: Data
        REAL, DIMENSION(N*(N-1)/2), INTENT(OUT) :: Resemblance
        CHARACTER(LEN=50) :: CoefName

        SELECT CASE(Coef)
        CASE (1)
                CALL Euclidean(M, N, Data, Resemblance)
                CoefName = 'Euclidean Distance'
        CASE (2)
                CALL SquareEuclidean(M, N, Data, Resemblance)
                CoefName = 'SquareEuclidean Distance'
        CASE (4)
                CALL CorrelationCoef(M, N, Data, Resemblance)
                CoefName = 'Correlation'
        CASE (5)
                CALL EuclideanSimiCoef(M, N, Data, Resemblance)
                CoefName = 'Euclidean Similarity'
        CASE (6)
                CALL Manhattan(M, N, Data, Resemblance)
                CoefName = 'Manhattan Distance'
        CASE (7)
                CALL Mahalanobis(M, N, Data, Resemblance)
                CoefName = 'Mahalanobis Distance'
        CASE (8)
                CALL SupDistance(M, N, Data, Resemblance)
                CoefName = 'Sup Distance'
        END SELECT
        IF (.NOT.ResWritten) THEN
                WRITE (7, 71) CoefName
                WRITE (7, *) '-------------------------------&
                &-------------------------'
        END IF
71      FORMAT ('  Resemblance Matrix - computed using ', A )
END SUBROUTINE CalculateResemblanceMatrix
! ================================================
SUBROUTINE Euclidean(M, N, Data, Resemblance)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N
    REAL, DIMENSION(M,N), INTENT(IN) :: Data
    REAL, DIMENSION(N*(N-1)/2), INTENT(OUT) :: Resemblance
    REAL, ALLOCATABLE, DIMENSION(:) :: Vector
    INTEGER:: I, J, K, AllocatStatus
```

```
    ALLOCATE( Vector(M), STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 )  STOP "(Vector)** No Enough Memory **"
    I = 1
    DO J=1, N-1
        DO K=J+1, N
            Vector = Data(1:M, J) - Data(1:M, K)
            Resemblance(I) = SQRT(DOT_PRODUCT(Vector,Vector))
            I = I + 1
        END DO
    END DO
    DEALLOCATE( Vector, STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 )  STOP "(Vector)** Deallocation Error **"
END SUBROUTINE Euclidean
! ============================================
SUBROUTINE SquareEuclidean(M, N, Data, Resemblance)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N
    REAL, DIMENSION(M,N), INTENT(IN) :: Data
    REAL, DIMENSION(N*(N-1)/2), INTENT(OUT) :: Resemblance
    REAL, ALLOCATABLE, DIMENSION(:) :: Vector
    INTEGER:: I, J, K, AllocatStatus

    ALLOCATE( Vector(M), STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 )  STOP "(Vector)** No Enough Memory **"
    I = 1
    DO J=1, N-1
        DO K=J+1, N
            Vector = Data(1:M, J) - Data(1:M, K)
            Resemblance(I) = DOT_PRODUCT(Vector,Vector)
            I = I + 1
        END DO
    END DO
    DEALLOCATE( Vector, STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 )  STOP "(Vector)** Deallocation Error **"
END SUBROUTINE SquareEuclidean
! ============================================
SUBROUTINE CorrelationCoef(M, N, Data, Resemblance)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N
    REAL, DIMENSION(M,N), INTENT(IN) :: Data
    REAL, DIMENSION(N*(N-1)/2), INTENT(OUT) :: Resemblance
    REAL, ALLOCATABLE, DIMENSION(:,:) :: C, incd
    REAL, ALLOCATABLE, DIMENSION(:) :: xmean
    INTEGER:: I, J, K, AllocatStatus,nmiss
    REAL :: sumwt

    ALLOCATE( C(N,N), STAT = AllocatStatus )
    IF (AllocatStatus/= 0 )  STOP "( C ) ** No Enough Memory **"
    ALLOCATE( xmean(N), STAT = AllocatStatus )
    IF (AllocatStatus/= 0 )  STOP "( xmean ) ** No Enough Memory **"
    ALLOCATE( incd(1,1), STAT = AllocatStatus )
    IF (AllocatStatus/= 0 )  STOP "( incd ) ** No Enough Memory **"
        ! use IMSL routine to calculate correlation
    CALL CORVC(0,M,N,Data,M,0,0,0,2,xmean,C,N,incd,1, M,nmiss,sumwt)
        ! extract only the upper-right triangle of the correlation matrix
    I = 1
    DO J=1, N-1
        DO K=J+1, N
            Resemblance(I) = C(J,K)
            I = I + 1
        END DO
    END DO
    DEALLOCATE( C, STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 )  STOP "( C )** Deallocation Error**"
    DEALLOCATE( xmean, STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 )  STOP "(xmean)** Deallocation Error **"
    DEALLOCATE( incd, STAT = AllocatStatus )
```

```
    IF ( AllocatStatus /= 0 ) STOP "(incd)" Deallocation Error"
END SUBROUTINE CorrelationCoef
! ============================================
SUBROUTINE EuclideanSimiCoef(M, N, Data, Resemblance)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N
    REAL, DIMENSION(M,N), INTENT(IN) :: Data
    REAL, DIMENSION(N*(N-1)/2), INTENT(OUT) :: Resemblance
    REAL, ALLOCATABLE, DIMENSION(:) :: Vector
    INTEGER:: I, J, K, AllocatStatus
    REAL :: Mx

    ALLOCATE( Vector(M), STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 ) STOP "(Vector)" No Enough Memory "
    I = 1
    DO J=1, N-1
        DO K=J+1, N
            Vector = Data(1:M, J) - Data(1:M, K)
            Resemblance(I) = SQRT(DOT_PRODUCT(Vector,Vector))
            I = I + 1
        END DO
    END DO
    Mx = MAXVAL(Resemblance)
    Resemblance = Resemblance / Mx
        DO I=1,N*(N-1)/2
                Resemblance(I) = 1.0 - Resemblance(I)
        END DO
    DEALLOCATE( Vector, STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 ) STOP "(Vector)" Deallocation Error "
END SUBROUTINE EuclideanSimiCoef
! ============================================
SUBROUTINE Manhattan(M, N, Data, Resemblance)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N
    REAL, DIMENSION(M,N), INTENT(IN) :: Data
    REAL, DIMENSION(N*(N-1)/2), INTENT(OUT) :: Resemblance
    REAL, ALLOCATABLE, DIMENSION(:) :: Vector
    INTEGER:: I, J, K, AllocatStatus

    ALLOCATE( Vector(M), STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 ) STOP "(Vector)" No Enough Memory "
    I = 1
    DO J=1, N-1
        DO K=J+1, N
            Vector = Data(1:M, J) - Data(1:M, K)
            Resemblance(I) = SUM(ABS(Vector))
            I = I + 1
        END DO
    END DO
    DEALLOCATE( Vector, STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 ) STOP "(Vector)" Deallocation Error "
END SUBROUTINE Manhattan
! ============================================
SUBROUTINE SupDistance(M, N, Data, Resemblance)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N
    REAL, DIMENSION(M,N), INTENT(IN) :: Data
    REAL, DIMENSION(N*(N-1)/2), INTENT(OUT) :: Resemblance
    REAL, ALLOCATABLE, DIMENSION(:) :: Vector
    INTEGER:: I, J, K, AllocatStatus

    ALLOCATE( Vector(M), STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 ) STOP "(Vector)" No Enough Memory "
    I = 1
    DO J=1, N-1
        DO K=J+1, N
```

```
            Vector = Data(1:M, J) - Data(1:M, K)
            Resemblance(I) = MAXVAL(ABS(Vector))
            I = I + 1
        END DO
    END DO
    DEALLOCATE( Vector, STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 ) STOP "(Vector)** Deallocation Error ***
END SUBROUTINE SupDistance
! ========================================================================


! ************************************************************************************************************************
! File Name: agglomerative.f
! Author: Ahmad Alhamed
!
! Execute clustering method
! Input:
!       N     : no. of initial clusters (col. in data matrix)
!    Method : used method, it can be one of the following
!      (1) : Single-link
!      (2) : Complete-link
!      (3) : UPGMA (group average)
!      (4) : WPGMA (weighted average)
!      (5) : UPGMC (unweighted centroid)
!      (6) : WPGMC (weighted centroid)
!      (7) : Ward's method (minimum variance)
!       Coeff  : resemblance coefficient
!       Dissim : Dissimilarity coefficient or similarity coefficient
!       Res    : Resemblance matrix
! Ooutput:
!       Hierarchy : result of clustering to be used to draw tree
! ************************************************************************************************************************
SUBROUTINE Agglomerative(N, Method, Coeff, Dissim, Res, Hierarchy)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: N, Method, Coeff
        LOGICAL, INTENT(IN) :: Dissim
        REAL, DIMENSION(N*(N-1)/2), INTENT(IN) :: Res
        TYPE(Merge), DIMENSION(N-1), INTENT(OUT) :: Hierarchy
        REAL :: LargeDissimilarity, SmallSimilarity
        INTEGER :: Offset   ! Func. returns index in Resemblance Matrix
        REAL   :: UpdateResDeg ! Func. returns new degree of resemblance
        INTEGER :: Index, I, J, K, L, AllocatStatus, Len
        REAL   :: Degree
        REAL, ALLOCATABLE, DIMENSION(:) :: Temp, Resemblance
        INTEGER, ALLOCATABLE, DIMENSION(:) :: Members
        LOGICAL, ALLOCATABLE, DIMENSION(:) :: Exist

        Len = N*(N-1) / 2
        LargeDissimilarity = 32.0E+23
        SmallSimilarity   = -32.0E+23
        ! Allocate memory space for arrays
        ALLOCATE( Members(N), STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Members)** No Enough Memory ***
        ALLOCATE( Exist(N), STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Exist)** No Enough Memory ***
        ALLOCATE( Resemblance(N*(N-1)/2), STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Temp)** No Enough Memory ***
        ALLOCATE( Temp(N*(N-1)/2), STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Resemblance)** No Enough Memory ***
        Resemblance = Res
        ! Initialize Members and Exist
        Members = 1
        Exist = .TRUE.
        ! Execute agglomerative clustering
        DO I = 1, N-1
                Temp = Resemblance
```

```
! Find the most similar pair
IF (Dissim) THEN
        Degree = MINVAL(Resemblance)
        DO L=1, Len
                IF (Resemblance(L) == Degree) THEN
                        Index = L
                        EXIT
                END IF
        END DO
ELSE
        Degree = MAXVAL(Resemblance)
        DO L=1, Len
                IF (Resemblance(L) == Degree) THEN
                        Index = L
                        EXIT
                END IF
        END DO
END IF
! Find corresponding cluster for this Index
CALL ClustersOfIndex(N, Index, J, K)
! Update Hierarchy
Hierarchy(I)%MergedCluster1 = J
Hierarchy(I)%MergedCluster2 = K
Hierarchy(I)%NewLable = J
IF (Method /= 7 ) THEN  ! for methods other than Ward's
  Hierarchy(I)%ResDegree = Degree
ELSE   ! for Ward's
  IF (I>1) THEN
        IF (Coeff == 2) THEN  ! for square Euc
          Hierarchy(I)%ResDegree = SQRT(Hierarchy(I-1)%ResDegree**2.0  + Degree/2.0)
        ELSE
          Hierarchy(I)%ResDegree = Hierarchy(I-1)%ResDegree + Degree
        END IF
  ELSE
        IF (Coeff == 2) THEN  ! for square Euc
          Hierarchy(I)%ResDegree = SQRT(Degree/2.0)
        ELSE
          Hierarchy(I)%ResDegree = Degree
        END IF
  END IF
END IF
! Set the corresponding index of K in Exist array to False
Exist(K) = .FALSE.
! Update Resemblance matrix
DO L = 1, N
        IF ( L/=J .AND. L/=K .AND. Exist(L) )  THEN
                Index = Offset(N, L, J)
                Resemblance(Index) = UpdateResDeg(N, L, J, K,  Method, Members, Temp)
                Index = Offset(N, L, K)
                IF (Dissim)  THEN
                        Resemblance(Index) = LargeDissimilarity
                ELSE
                        Resemblance(Index) = SmallSimilarity
                END IF
        END IF
END DO
! Cancel the value of the larger index of the newly merged cluster in the Resemblance matrix
Index = Offset(N, J, K)
IF (Dissim)  THEN
        Resemblance(Index) = LargeDissimilarity
ELSE
        Resemblance(Index) = SmallSimilarity
END IF
! Change no. of members with respect to the newly
! merged clusters
Members(J) = Members(J) + Members(K)
Members(K) = 0
WRITE (7, *) '-------------------------'
WRITE (7, *) 'Resemblance at step:', I
```

```
                CALL PrintResemblanceMatrix(N, Resemblance)
                WRITE (7, 10) Hierarchy(I)%MergedCluster1, Hierarchy(I)%MergedCluster2, &
                & Hierarchy(I)%NewLable, Hierarchy(I)%ResDegree
10      FORMAT ('Merge',I2,' and ',I2,' to form ',I2,' at ==> ',F10.2)
        END DO

        ! Deallocate arrays
        DEALLOCATE( Members, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP  "(Members)** Deallocation Error ***
        DEALLOCATE( Exist, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP  "(Exist)** Deallocation Error ***
        DEALLOCATE( Resemblance, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP  "(Resemblance)** Deallocation Error ***
        DEALLOCATE( Temp, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP  "(Temp)** Deallocation Error ***
END SUBROUTINE Agglomerative
! ========================================================================
SUBROUTINE ClustersOfIndex(N, I, J, K)
        IMPLICIT NONE
! ************************************************************************
! Finds the corresponding clusters of an index in the Resemblance matrix
! INPUTS:
!       N  : No. of clusters in the Resemblance matrix
!       I  : Index in the Resemblance matrix
! OUTPUTS
!       J, K : Clusters correspond to I
! ************************************************************************
        INTEGER, INTENT(IN)  :: N, I
        INTEGER, INTENT(OUT) :: J, K
        INTEGER :: Prev, Loc, R

        Prev = 0
        Loc = 0
        DO R =1, N-1
                Prev = Loc
                Loc = Loc + (N-R)
                IF (I <= Loc)  THEN
                        J = R
                        EXIT
                END IF
        END DO
        K = J + (I-Prev)
END SUBROUTINE ClustersOfIndex
! ========================================================================
REAL FUNCTION UpdateResDeg(N, K, R, S, Method, Members, Res)
        IMPLICIT NONE
! ************************************************************************
! Returns new resemblnace using Lance and Williams' formula which
! given in :
!  A. Jain, R. Dubes, "Algonthms for Clustering Data", Prentice Hall,
!  1988, PP. 79-80
! INPUTS:
!       N : Original no. of clusters = no. of col in Data matrix
!       K : Existing cluster
!       R, S : newly formed cluster
!       Method : used method, it can be one of the following
!         (1) : Single-link
!         (2) : Complete-link
!         (3) : UPGMA (group average)
!         (4) : WPGMA (weighted average)
!         (5) : UPGMC (unweighted centroid)
!         (6) : WPGMC (weighted centroid)
!         (7) : Ward's method (minimum variance)
!       Members: no. of members in each cluster
!       Res    : Resemblance matrix
! OUTPUT:
!       returns the new resemblance degree
! ************************************************************************
        INTEGER, INTENT(IN) :: N, K, R, S, Method
```

```
        INTEGER, DIMENSION(N), INTENT(IN) :: Members
        REAL, DIMENSION(N*(N-1)/2), INTENT(IN) :: Res
        REAL :: AlphaR, AlphaS, Beta, Gamma
        INTEGER :: Offset   ! Func. returns index in Resemblance Matrix

        SELECT CASE(Method)
        CASE (1)
                AlphaR = 0.50
                AlphaS = 0.50
                Beta   = 0.0
                Gamma  = -0.50
        CASE (2)
                AlphaR = 0.50
                AlphaS = 0.50
                Beta   = 0.0
                Gamma  = 0.50
        CASE (3)
                AlphaR = REAL(Members(R)) / ( REAL(Members(R)) + REAL(Members(S)) )
                AlphaS = REAL(Members(S)) / ( REAL(Members(R)) + REAL(Members(S)) )
                Beta   = 0.0
                Gamma  = 0.0
        CASE (4)
                AlphaR = 0.50
                AlphaS = 0.50
                Beta   = 0.0
                Gamma  = 0.0
        CASE (5)
                AlphaR = REAL(Members(R)) / ( REAL(Members(R))  + REAL(Members(S)) )
                AlphaS = REAL(Members(S)) / ( REAL(Members(R))  + REAL(Members(S)) )
                Beta   = -1.0*REAL(Members(R))*REAL(Members(S)) &
                        & / ( REAL(Members(R)) + REAL(Members(S)) )**2.0
                Gamma  = 0.0
        CASE (6)
                AlphaR = 0.50
                AlphaS = 0.50
                Beta   = -0.250
                Gamma  = 0.0
        CASE (7)
                AlphaR =( REAL(Members(R))+REAL(Members(K)) ) &
                        & / ( REAL(Members(R))+REAL(Members(S)) +REAL(Members(K)) )
                AlphaS =( REAL(Members(S))+REAL(Members(K)) ) &
                        & / ( REAL(Members(R))+REAL(Members(S))  +REAL(Members(K)) )
                Beta   = ( -1.0*REAL(Members(K)) )/( REAL(Members(R)) &
                        & +REAL(Members(S))+REAL(Members(K)) )
                Gamma  = 0.0
        END SELECT
        UpdateResDeg = AlphaR*Res(Offset(N,K,R)) + AlphaS*Res(Offset(N,K,S)) &
                & + Beta*Res(Offset(N,R,S)) + Gamma*ABS(Res(Offset(N,K,R)) - Res(Offset(N,K,S)) )
END FUNCTION UpdateResDeg
! ======================================================================
INTEGER FUNCTION Offset(N, J, K)
    IMPLICIT NONE
! **********************************************************************
! Return index of clusters J, and K  in the Resemblance matrix
! INPUTS:
!     N : No. of clusters in the Resemblance matrix
!     J, K : Clusters to find their index in the Resemblance matrix
! OUTPUT:
!     Returns index of J, and K in the Resemblance matrix
! **********************************************************************
    INTEGER, INTENT(IN) :: N, J, K
    INTEGER :: P, Q

    P = MIN(J, K)
    Q = MAX(J, K)
    Offset = Q + (P-1) * N - (P*(P+1)) / 2
END FUNCTION Offset
! ======================================================================
```

```
! ***********************************************************************
! File Name: cophenetic.f
! Author: Ahmad Alhamed
! Compute the cophenetic correlation coefficient
! Ref: Romesburg, H., "Cluster Analysis for Researchers",
!    Lifetime Learning Pub., 1984, PP 24-27
! Input:
!          N       : no. of clusters (col. in data matrix)
!          Hierarchy : hierarch of clusters as produced by agglomerative
!    Res   : Resemblance matrix
!          Method : string containing the method names
! Output:
!    CopheneticCoeff: Cophenetic Correlation Coeff.
! ***********************************************************************
SUBROUTINE Cophenetic(N, Hierarchy, Res, CopheneticCoeff)
    USE Global
    IMPLICIT NONE
    INTEGER, INTENT(IN) :: N
    TYPE(Merge), DIMENSION(N-1), INTENT(IN) .: Hierarchy
        REAL, DIMENSION(N*(N-1)/2), INTENT(IN) :: Res
        REAL, INTENT(OUT) :: CopheneticCoeff
        INTEGER :: Offset   ! Func. returns index in CopheneticMatrix
    INTEGER, ALLOCATABLE, DIMENSION(:,:) :: Clusters
    INTEGER, ALLOCATABLE, DIMENSION(:) :: Members
    REAL, ALLOCATABLE, DIMENSION(:) :: CopheneticMatrix
    LOGICAL, ALLOCATABLE, DIMENSION(:) :: Exist
    REAL :: SUMxy, SUMx2, SUMy2, SUMx, SUMy
    INTEGER :: AllocatStatus, I, J, K, P, Q, NDX, L

    L = N*(N-1)/2
    ALLOCATE( Clusters(N,N), STAT = AllocatStatus )
    IF (AllocatStatus/= 0 ) STOP "(Clusters) ** No Enough Memory ***
    ALLOCATE( Members(N), STAT = AllocatStatus )
    IF (AllocatStatus/= 0 ) STOP "(Members) ** No Enough Memory ***
    ALLOCATE( CopheneticMatrix(L), STAT = AllocatStatus )
    IF (AllocatStatus/= 0 ) STOP "(CopheneticMatrix) ** No Enough Memory ***
    ALLOCATE( Exist(L), STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 ) STOP "(Exist)** No Enough Memory ***
    DO I=1, N
        Clusters(I, 1) = I
        Members(I) = 1
    END DO
        Exist = .FALSE.
    DO I=1, N-1
        J = Hierarchy(I)%MergedCluster1
        K = Hierarchy(I)%MergedCluster2
                ! STEP 1: merge clusters
        DO Q=1,   Members(K)
            Members(J) = Members(J) + 1
            NDX = Members(J)
            Clusters(J, NDX) = Clusters(K, Q)
        END DO
                ! STEP 2: compute cophenetic matrix
        DO P=1,  Members(J)
                  DO Q=1, Members(K)
                          IF ( Clusters(J,P) /= Clusters(K,Q) ) THEN
              NDX = Offset(N, Clusters(J,P), Clusters(K,Q) )
                            IF ( EXIST(NDX) .EQV. .FALSE. ) THEN
                              CopheneticMatrix(NDX) = Hierarchy(I)%ResDegree
                              EXIST(NDX) = .TRUE.
                            END IF
                          END IF
                  END DO
        END DO
    END DO
        ! compute Cophenetic correlation Coeff
        SUMxy = 0.0
        SUMx2 = 0.0
        SUMy2 = 0.0
```

```
        SUMx  = 0.0
        SUMy  = 0.0
        DO I=1, L
                SUMxy = SUMxy + ( Res(I) * CopheneticMatrix(I) )
                SUMx2 = SUMx2 + Res(I)**2.0
                SUMy2 = SUMy2 + CopheneticMatrix(I)**2.0
                SUMx  = SUMx + Res(I)
                SUMy  = SUMy + CopheneticMatrix(I)
        END DO
        CopheneticCoeff = ( SUMxy - (1/REAL(L))*SUMx*SUMy ) / &
                          & ( SQRT( (SUMx2-(1/REAL(L))*SUMx**2.0 ) * &
                          & (SUMy2-(1/REAL(L))*SUMy**2.0 ) ))
END SUBROUTINE Cophenetic
!=========================================================================


! ***************************************************************
! File Name: tree.f
! Author: Ahmad Alhamed
!
! Draw clustering tree
! ***************************************************************
SUBROUTINE Tree(N, Hierarchy, Title, Dissim, Scale, OriginX, OriginY, CophCorrCoef)
    USE Global
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: N
    LOGICAL, INTENT(IN) :: Dissim
    TYPE(Merge), DIMENSION(N-1), INTENT(IN) :: Hierarchy
    CHARACTER(LEN=*), INTENT(IN) :: Title
    REAL, INTENT(IN) :: Scale, OriginX, OriginY, CophCorrCoef
    INTEGER :: appid, wid, gkswid, ierr
    REAL, DIMENSION(N) :: XLoc, YLoc
    REAL :: HUnit, VUnit, Height, Width, Offset, X, Y, Dot
    REAL :: Range1, Range2, FontScale
    INTEGER :: I, J, K
    REAL :: MaxDegree, DegreeScale, prevY, prevDegree
    INTEGER, DIMENSION(N) :: OrderedList
    CHARACTER(LEN=3) :: Lable
    CHARACTER(LEN=15) :: Degree
    CHARACTER(LEN=80) :: Buff

! order cluster on X axis
    CALL OrderClusters(N, Hierarchy, OrderedList)

! INITIALIZE
    IF (OutputFormat==ATOMOP .OR. OutputFormat==OTAMOP) THEN
     IF (TPP == 4) THEN
            Height = 0.45
            Width = 0.48
            Offset = 0.03
            Dot = 0.0016
            FontScale = 1.0
        END IF
     IF (TPP == 9) THEN
            Height = 0.30
            Width = 0.32
            Offset = 0.03
            Dot = 0.0016
            FontScale = 1.0
        END IF
    ELSE
        ! OutputFormat==OTOMOP .OR. OutputFormat==OTAMSP
        Height = 1.0
        Width = 1.0
        Offset = 0.1
        Dot = 0.005
        FontScale = 2.0
```

```
    END IF

    IF (Dissim) THEN
          SELECT CASE(OutputFormat)
          CASE(OTOMOP, OTAMSP, OTAMOP)
                MaxDegree = Hierarchy(N-1)%ResDegree
          CASE(ATOMOP)
                MaxDegree = Scale
          END SELECT
    ELSE
      MaxDegree = 1.0
    END IF
    prevDegree = MaxDegree
    HUnit = (Width-(3.0*Offset)) / REAL(N)
    IF (HUnit == 0.0)  HUnit = Dot
    VUnit = (Height-(3.5*Offset)) / MaxDegree

! initialixe X locations for N clusters
    X = OriginX + (2.0*Offset)
    DO I=1, N
          X = X + HUnit
          XLoc(OrderedList(I)) = X
    END DO
    YLoc = OriginY + (2.0*Offset)   ! for all clusters

    ! display title
    CALL PLCHLQ(OriginX+(Width/2.0), OriginY+(Height-0.3*Offset),TRIM(Title), 9.0*FontScale,0.,0.)
    ! Set the line width
    CALL GSLWSC(2.)
    ! draw scale to the left of tree
    CALL LINE(OriginX+(2.0*Offset), OriginY+2.0*Offset, OriginX+(2.0*Offset), OriginY+(Height-0.7*Offset) )
    CALL LINE(OriginX+(2.0*Offset-(3.0*Dot)),OriginY+2.0*Offset, OriginX+(2.0*Offset+(3.0*Dot)),OriginY+2.0*Offset)
    CALL LINE(OriginX+(2.0*Offset-(3.0*Dot)),OriginY+(Height-0.7*Offset),&
    &  OriginX+(2.0*Offset+(3.0*Dot)), OriginY+(Height-0.7*Offset))
    ! lable X axis
    DO I=1, N
          WRITE(Lable, '(A3)' ) TRIM(LABLES(OrderedList(I)))
          CALL PLCHLQ(XLoc(OrderedList(I))+(2.0*Dot), OriginY+(1.7*Offset),Lable ,8.0*FontScale,0.,1.)
    END DO
    WRITE(Buff, 31)  Hierarchy(1)%ResDegree, Hierarchy(N-1)%ResDegree
31  FORMAT ('Range: ',F15.3,' , ',F15.3)
    CALL PLCHLQ(OriginX+Offset, OriginY+(1.0*Offset),TRIM(Buff),9.0*FontScale,0.,-1.)
    WRITE(Buff, 32) CophCorrCoef
32  FORMAT ('Cophenetic Corr. Coeff. = ',F4.2)
    CALL PLCHLQ(OriginX+Offset, OriginY+(.5*Offset),TRIM(Buff),9.*FontScale,0.,-1.)
    ! set the line width to smaller
    CALL GSLWSC(1.)
    ! draw the tree
    prevY = OriginY
    Y = OriginY + (2.0*Offset)
    DO I=1, N-1
          ! in case of tie, do not update Y
          IF(Hierarchy(I)%ResDegree /= prevDegree)  THEN
               IF (Dissim)  THEN
                    Y = OriginY + (2.0*Offset) + Hierarchy(I)%ResDegree*VUnit
               ELSE
                    Y = OriginY+(2.0*Offset)+(1.0 - Hierarchy(I)%ResDegree)*VUnit
               END IF
          END IF
          J = Hierarchy(I)%MergedCluster1
          K = Hierarchy(I)%MergedCluster2
          ! NCARG code to draw line
          CALL LINE(XLoc(J), YLoc(J), XLoc(J), Y)
          CALL LINE(XLoc(K), YLoc(K), XLoc(K), Y)
          CALL LINE(XLoc(J), Y, XLoc(K), Y)
          IF(Hierarchy(I)%ResDegree /= prevDegree)  THEN
                    ! draw tick on the scale, and display degree
                    CALL LINE(OriginX+(2.0*Offset-(3.0*Dot)), Y, OriginX+(2.0*Offset+(3.0*Dot)), Y)
                    ! if degree close to each other, they will be
```

```
                ! overlaped on the plot. Skip if overlap
                IF (Dissim)  THEN
                        WRITE(Degree, '(F15.3)' ) Hierarchy(I)%ResDegree
                ELSE
                        WRITE(Degree, '(F15.5)' ) Hierarchy(I)%ResDegree
                END IF
                IF ( Y-prevY > (5.0*Dot) ) THEN
                        CALL PLCHLQ(OriginX+(2.0*Offset-(6.0*Dot)), Y, Degree,6.0*FontScale,0.,1.)
                        prevY = Y
                END IF
        END IF
        ! update X, Y locations
        XLoc(J) = XLoc(J) + (XLoc(K)-XLoc(J))/2.0
        YLoc(J) = Y
        prevDegree = Hierarchy(I)%ResDegree
    END DO
    ! draw line from last merge to the top
    CALL LINE(XLoc(J), Y, XLoc(J), Y+0.4*Offset )
END SUBROUTINE Tree
! ============================================================
SUBROUTINE OrderClusters(N, Hierarchy, OrderedList)
    USE Global
    IMPLICIT NONE
! ************************************************************
! Order the clusters to appear nice on the tree
! ************************************************************
    INTEGER, INTENT(IN) :: N
    TYPE(Merge), DIMENSION(N-1), INTENT(IN) :: Hierarchy
    INTEGER, DIMENSION(N), INTENT(OUT) :: OrderedList
    INTEGER, ALLOCATABLE  DIMENSION(:,:) :: Clusters
    INTEGER, ALLOCATABLE, DIMENSION(:) :: Members
    INTEGER :: AllocatStatus, I, J, K, P, Q, NDX

    ALLOCATE( Clusters(N,N), STAT = AllocatStatus )
    IF (AllocatStatus/= 0 )  STOP  "(Clusters) ** No Enough Memory **"
    ALLOCATE( Members(N), STAT = AllocatStatus )
    IF (AllocatStatus/= 0 )  STOP  "(Members) ** No Enough Memory **"
    DO I=1, N
        Clusters(I, 1) = I
        Members(I) = 1
    END DO
    DO P=1, N-1
        J = Hierarchy(P)%MergedCluster1
        K = Hierarchy(P)%MergedCluster2
        DO Q=1,   Members(K)
            Members(J) = Members(J) + 1
            NDX = Members(J)
            Clusters(J, NDX) = Clusters(K, Q)
        END DO
    END DO
    DO I=1, N
        OrderedList(I) = Clusters(1, I)
    END DO
END SUBROUTINE OrderClusters
! ============================================================
```

Nonhierarchical (partitional) Cluster Analysis
Kmean Algorithm

```
! ************************************************************************************************************
! File Name: kmean.f
! Author: Ahmad Alhamed
! This program performs kmean algorithm, a nonhierarchical Cluster Analysis method. It calls other subroutines and
! functions from CAlib.f. When compiling this program, it should be linked with with IMSL library which is used for
! computation of the kmean algorithm.
! ************************************************************************************************************
MODULE Global
    IMPLICIT NONE

        INTEGER :: InputMode
        INTEGER, PARAMETER :: OneTime=1, AllTimes=2
                ! OneTime : Apply Kmean for one time only
                ! AllTimes : Apply Kmean for all time instances
        INTEGER :: StartingTime, NoOfTimes, Interval, NDX
        INTEGER, PARAMETER :: CntrlFP=10, OutFP=7 ! File Pointers
        CHARACTER(LEN=100), ALLOCATABLE, DIMENSION(:,:) :: ClustersTable
END MODULE Global
! ========================================================================
PROGRAM kmeanAlg
        IMPLICIT NONE

        CHARACTER(LEN=20) :: argv

        CALL GETARG(1, argv)
        IF ( LEN(TRIM(argv)) == 0 ) THEN
                CALL UserInterface
        ELSE
                CALL ControlFile(argv)
        END IF
END PROGRAM kmeanAlg
! ========================================================================
SUBROUTINE UserInterface
        USE Global
        IMPLICIT NONE
! ************************************************************************
!   Show the user interface, and obtain all required input
! ************************************************************************

        INTEGER :: NormalizeMenu, SeedPointMenu       ! functions
        INTEGER :: M, N, K, Normalize, IERR, AllocatStatus, SeedPointChoice
        CHARACTER(LEN=50) :: OutputFile, InputFile
        CHARACTER(LEN=25) :: Prefix
        CHARACTER(LEN=70) :: Title

        PRINT '($,A)','Enter Title: '
        READ (*, FMT='(A)') Title
        PRINT '($,A)','Enter prefix for Output File Name:'
        READ *, Prefix
        PRINT '($,A)','Enter No. of observations (rows) in Data Matrix: '
        READ *, M
        PRINT '($,A)','Enter No. of variables (column) in Data Matrix: '
        READ *, N
        PRINT '($,A)','Enter Input File Name:  '
        READ *, InputFile
        ! normalization menu
        Normalize = NormalizeMenu()
        PRINT '($,A)','Enter Number of clusters:  '
        READ *, K
        ! Seed Point menu
        SeedPointChoice = SeedPointMenu()
        OutputFile = TRIM(Prefix)//'Kmean.out'
        ! this statment append to existing file or create new one
        OPEN (UNIT=OutFP,FILE=OutputFile,POSITION='APPEND',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 7 , OutFP,OutputFile, IERR
7               FORMAT (' ERROR OPENING UNIT=',I2,' FILE NAME =',A20,', IOSTAT = ',I8)
```

```
                STOP 7
        END IF
        WRITE (OutFP, '(A)') Title
        WRITE (OutFP, '(A)') "***** Kmean Clustering Method *****"
        CALL WriteSubTitle1(Normalize)
        WRITE (OutFP, 9) K
9       FORMAT ('Number of Clusters, K, =',I3)
        CALL WriteSeedPointsSubTitle(K, SeedPointChoice)
        WRITE (OutFP, *)
        InputMode = OneTime
        Call Start(M, N, InputFile, Normalize, K, SeedPointChoice)
        CLOSE (UNIT=OutFP)
END SUBROUTINE UserInterface
! ======================================================================
SUBROUTINE ControlFile(ControlFileName)
        USE Global
        IMPLICIT NONE
! **********************************************************************
!    Read the required input from input file called 'Control File'
! **********************************************************************

        CHARACTER(LEN=*), INTENT(IN) :: ControlFileName
        INTEGER :: M, N, K, Normalize, IERR, AllocatStatus, SeedPointChoice, I, Time
        CHARACTER(LEN=50) :: OutputFile, InputFile, FileName
        CHARACTER(LEN=25) :: Prefix
        CHARACTER(LEN=70) :: Title

        OPEN (UNIT=CntrlFP,FILE=ControlFileName,STATUS='OLD',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 3 , CntrlFP,ControlFileName, IERR
3               FORMAT ( ' ERROR OPENING UNIT=',I2,' FILE NAME = ',A15, ', IOSTAT = ',I8)
                STOP 3
        ENDIF
        READ (CntrlFP, *) InputMode
        IF (InputMode == AllTimes) THEN
                READ (CntrlFP, *) StartingTime, NoOfTimes, Interval
        END IF
        READ (CntrlFP, FMT='(A)') Title
        READ (CntrlFP, *) Prefix
        READ (CntrlFP, *) M, N
        READ (CntrlFP, *) InputFile
        READ (CntrlFP, *) Normalize
        READ (CntrlFP, *) K
        READ (CntrlFP, *) SeedPointChoice
        OutputFile = TRIM(Prefix)//'Kmean.out'
        ! this statment append to existing file or create new one
        OPEN (UNIT=OutFP,FILE=OutputFile,POSITION='APPEND',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 7 , OutFP, OutputFile, IERR
7               FORMAT (' ERROR OPENING UNIT=',I2,' FILE NAME =',A13,', IOSTAT = ',I8)
                STOP 7
        END IF
        WRITE (OutFP, '(A)') Title
        WRITE (OutFP, '(A)') "***** Kmean Clustering Method *****"
        CALL WriteSubTitle1(Normalize)
        WRITE (OutFP, 9) K
9       FORMAT ('Number of Clusters, K, =',I3)
        CALL WriteSeedPointsSubTitle(K, SeedPointChoice)
        WRITE (OutFP, *)
        SELECT CASE(InputMode)
        CASE (OneTime)
                Call Start(M, N, InputFile, Normalize, K, SeedPointChoice)
        CASE (AllTimes)
                OPEN(UNIT=50,FILE=InputFile,STATUS='OLD',IOSTAT=IERR)
                IF (IERR /= 0) THEN
                        PRINT 10 , InputFile, IERR
10                      FORMAT ( ' ERROR OPENING UNIT=50, FILE NAME = ',A15, ', IOSTAT = ',I8)
                        STOP 10
                ENDIF
```

```
                ALLOCATE( ClustersTable(NoOfTimes,K), STAT = AllocatStatus )
                IF (AllocatStatus/= 0 ) STOP "(ClustersTable) ""No Enough Memory"""
                Time = StartingTime
                DO I=1, NoOfTimes
                        NDX = I
                        READ (50, *) FileName
                        SELECT CASE (MOD(Time, 10))
                        CASE (1)
                                WRITE(OutFP, 11) Time
                        CASE (2)
                                IF (Time == 12) THEN
                                        WRITE(OutFP, 14) Time
                                ELSE
                                        WRITE(OutFP, 12) Time
                                END IF
                        CASE (3)
                                WRITE(OutFP, 13) Time
                        CASE DEFAULT
                                WRITE(OutFP, 14) Time
                        END SELECT
                        Time = Time + Interval
                        Call Start(M, N, FileName, Normalize, K, SeedPointChoice)
                END DO
                CALL WriteClustersTable(K)
                DEALLOCATE( ClustersTable, STAT = AllocatStatus )
                IF ( AllocatStatus /= 0 ) STOP "(ClustersTable)""Deallocation Error"""
11              FORMAT ('At the ',I2,'st hr' )
12              FORMAT ('At the ',I2,'nd hr' )
13              FORMAT ('At the ',I2,'rd hr' )
14              FORMAT ('At the ',I2,'th hr' )
        END SELECT
        CLOSE (UNIT=50)
        CLOSE (UNIT=OutFP)
END SUBROUTINE ControlFile
! ======================================================================
SUBROUTINE Start(M, N, InputFileName, Normalize, K, SeedPointChoice)
        USE Global
        IMPLICIT NONE

        CHARACTER(LEN=*), INTENT(IN) :: InputFileName
        INTEGER, INTENT(IN) :: M, N, Normalize, K, SeedPointChoice
        INTEGER :: AllocatStatus, I, J, NUM
        REAL, ALLOCATABLE, DIMENSION(:,:) :: DataMatrix, X, Seeds

        ALLOCATE( DataMatrix(M,N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(DataMatrix) "" No Enough Memory """
        !READ DATA MATRIX FROM INPUT FILE
        CALL ReadDataMatrixFromFile(M, N, DataMatrix, InputFileName)
        ! normalize if requested
        SELECT CASE(Normalize)
        CASE (1)
                CALL CenterByColMean(M, N, DataMatrix)
        CASE (2)
                CALL NormalizeByColMax(M, N, DataMatrix)
        CASE (3)
                CALL NormalizeByColMinMax(M, N, DataMatrix)
        CASE (4)
                CALL NormalizeByMax(M, N, DataMatrix)
        END SELECT
        ! transpose datamatrix to be suitable for kmean algorithm
        ALLOCATE( X(N,M), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(X) "" No Enough Memory """
        X = TRANSPOSE(DataMatrix)
        DEALLOCATE( DataMatrix, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(DataMatrix)"" Deallocation Error"""
        ! obtain Seed Points
        ALLOCATE( Seeds(K,M), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(Seeds) "" No Enough Memory """
        CALL ObtainSeedPoints(M, N, X, K, SeedPointChoice, Seeds)
```

```
        ! implement kmean algorithm
        CALL DoClustering(M, N, X, K, Seeds)
        DEALLOCATE( X, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(X)"" Deallocation Error"""
        DEALLOCATE( Seeds, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Seeds)"" Deallocation Error"""
END SUBROUTINE Start
! ========================================================================
SUBROUTINE DoClustering(M, N, X, K, Seeds)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: M, N, K
        REAL, DIMENSION(N,M), INTENT(IN) :: X
        REAL, DIMENSION(K,M), INTENT(IN) :: Seeds
        INTEGER :: IC(N), IND(M), NC(K), Allocatstatus, I, J
        REAL :: SWT(K,M), WSS(K)
        REAL, ALLOCATABLE, DIMENSION(:,:) :: CM
        CHARACTER(LEN=100) :: Str1,Str2, Str3
        LOGICAL :: First

        ALLOCATE( CM(K,M), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(CM) "" No Enough Memory """
        CM = Seeds
        DO I=1,M
                IND(I) = I
        END DO
        CALL KMEAN(N,M,M,X,N,0,0,IND,K,30,CM,K,SWT,K,IC,NC,WSS)
        WRITE(OutFP,'(A)') '  Cluster Membership'
        WRITE(OutFP,'(15I6)') (I, I=1,N)
        WRITE(OutFP,'(15I6)') (IC(I), I=1,N)
        WRITE(OutFP,'(A)') '  No of Members In Each Cluster'
        WRITE(OutFP,'(10I8)') (I, I=1,K)
        WRITE(OutFP,'(10I8)') (NC(I), I=1,K)
        WRITE(OutFP,'(A)') '  The Within sum of squares'
        WRITE(OutFP,'(10I20)') (I, I=1,K)
        WRITE(OutFP,'(10F20.2)') (WSS(I), I=1,K)
        WRITE(OutFP,*)
        WRITE(OutFP,*)
        ! add to clusters table
        IF (InputMode == AllTimes) THEN
          DO I=1, K
                First = .TRUE.
                WRITE(Str1, '(A1,I1)') '['
                DO J=1, N
                        IF ( IC(J) == I) THEN
                                IF (First) THEN
                                        WRITE(Str2, '(I2)') J
                                        First = .FALSE.
                                ELSE
                                        WRITE(Str2, '(A1,I2)') ',', J
                                END IF
                                WRITE(Str3, '(A,A)') TRIM(Str1),TRIM(Str2)
                                Str1 = Str3
                        END IF
                END DO
                Str2 = ']'
                WRITE(Str3, '(A,A)') TRIM(Str1), TRIM(Str2)
                ClustersTable(NDX,I) = Str3
          END DO
        END IF
END SUBROUTINE DoClustering
! ========================================================================
SUBROUTINE WriteClustersTable(K)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: K
        INTEGER :: Time, I, J
```

```
        WRITE(OutFP, *) 'Clusters Table'
        WRITE(OutFP, '(A,T10,A)') 'Time', 'Clusters'
        Time = StartingTime
        DO I=1, NoOfTimes
                WRITE(OutFP,'($,I2)') Time
                DO J=1, K
                        WRITE(OutFP,'($,A,3X)') TRIM( ClustersTable(I,J) )
                END DO
                WRITE(OutFP, *)
                Time = Time + Interval
        END DO
END SUBROUTINE WriteClustersTable
! ================================================================================
```

Nonhierarchical (partitional) Cluster Analysis
Nucleated Agglomerative (NA) Algorithm

```
! ************************************************************************************************************
! File Name: nucleated.f
! Author: Ahmad Alhamed
! This program performs kmean algorithm, a nonhierarchical Cluster Analysis method. It calls other subroutines and
! functions from CAlib.f. When compiling this program, it should be linked with with IMSL library which is used for
! computation of the kmean algorithm.
! ************************************************************************************************************
MODULE Global
        IMPLICIT NONE

        INTEGER, PARAMETER :: CntrlFP=10, OutFP=7 ! File Pointers
        LOGICAL, ALLOCATABLE, DIMENSION(:) :: Exist
END MODULE Global
! ========================================================================
PROGRAM NAalg
        IMPLICIT NONE

        CHARACTER(LEN=20) :: argv

        CALL GETARG(1, argv)
        IF ( LEN(TRIM(argv)) == 0 )  THEN
                CALL UserInterface
        ELSE
                CALL ControlFile(argv)
        END IF
END PROGRAM NAalg
! ========================================================================
SUBROUTINE UserInterface
        USE Global
        IMPLICIT NONE
! ************************************************************
!   Show the user interface, and obtain all required input
! ************************************************************

        INTEGER :: NormalizeMenu, SeedPointMenu       ! functions
        INTEGER :: M, N, Kmax, Kmin, Normalize, IERR, AllocatStatus, SeedPointChoice
        CHARACTER(LEN=50) :: OutputFile, InputFile
        CHARACTER(LEN=25) :: Prefix
        CHARACTER(LEN=70) :: Title

        PRINT '($,A)','Enter Title: '
        READ (*, FMT='(A)') Title
        PRINT '($,A)','Enter prefix for Output File Name:'
        READ *, Prefix
        PRINT '($,A)','Enter No. of observations (rows) in Data Matrix: '
        READ *, M
        PRINT '($,A)','Enter No. of variables (column) in Data Matrix: '
        READ *, N
        PRINT '($,A)','Enter Input File Name:  '
        READ *, InputFile
        ! normalization menu
        Normalize = NormalizeMenu()
        PRINT '($,A)','Enter Max. Number of clusters:  '
        READ *, Kmax
        PRINT '($,A)','Enter Min. Number of clusters:  '
        READ *, Kmin
        ! Seed Point menu
        SeedPointChoice = SeedPointMenu()
        OutputFile = TRIM(Prefix)//'Nucleated.out'
        OPEN (UNIT=OutFP,FILE=OutputFile,STATUS='NEW',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 7 , OutFP,OutputFile, IERR
7               FORMAT (' ERROR OPENING UNIT=',I2,' FILE NAME =' ,A20,', IOSTAT = ',I8)
                STOP 7
        END IF
        WRITE (OutFP, '(A)') Title
        WRITE (OutFP, '(A)') "**** Nucleated Agglomerative Clustering Method ****"
```

```
        CALL WriteSubTitle1(Normalize)
        WRITE (OutFP, 9) Kmax, Kmin
9       FORMAT ('Number of Clusters: ',I3, ' Down to ',I3)
        CALL WriteSeedPointsSubTitle(Kmax, SeedPointChoice)
        WRITE (OutFP, *)
        Call Start(M, N, InputFile, Normalize, Kmax, Kmin, SeedPointChoice)
        CLOSE (UNIT=OutFP)
END SUBROUTINE UserInterface
! ============================================================================
SUBROUTINE ControlFile(ControlFileName)
        USE Global
        IMPLICIT NONE
! ***************************************************************
!    Read the required input from input file called 'Control File'
! ***************************************************************

        CHARACTER(LEN=*), INTENT(IN) :: ControlFileName
        INTEGER :: M, N, Kmax, Kmin, Normalize, IERR, AllocatStatus, SeedPointChoice, I, Time
        CHARACTER(LEN=50) :: OutputFile, InputFile, FileName
        CHARACTER(LEN=25) :: Prefix
        CHARACTER(LEN=70) :: Title

        OPEN (UNIT=CntrlFP,FILE=ControlFileName,STATUS='OLD',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 3 , CntrlFP,ControlFileName, IERR
3               FORMAT ( ' ERROR OPENING UNIT=',I2,' FILE NAME = ',A15, ', IOSTAT = ',I8)
                STOP 3
        ENDIF
        READ (CntrlFP, FMT='(A)') Title
        READ (CntrlFP, *) Prefix
        READ (CntrlFP, *) M, N
        READ (CntrlFP, *) InputFile
        READ (CntrlFP, *) Normalize
        READ (CntrlFP, *) Kmax, Kmin
        READ (CntrlFP, *) SeedPointChoice
        OutputFile = TRIM(Prefix)//'Nucleated.out'
        OPEN (UNIT=OutFP,FILE=OutputFile,STATUS='NEW',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 7 , OutFP, OutputFile, IERR
7               FORMAT (' ERROR OPENING UNIT=',I2,' FILE NAME =' ,A13,', IOSTAT = ',I8)
                STOP 7
        END IF
        WRITE (OutFP, '(A)') Title
        WRITE (OutFP, '(A)') "**** Nucleated Agglomerative Clustering Method ****"
        CALL WriteSubTitle1(Normalize)
        WRITE (OutFP, 9) Kmax, Kmin
9       FORMAT ('Number of Clusters: ',I3, ' Down to ',I3)
        CALL WriteSeedPointsSubTitle(Kmax, SeedPointChoice)
        WRITE (OutFP, *)
        Call Start(M, N, InputFile, Normalize, Kmax, Kmin, SeedPointChoice)
        CLOSE (UNIT=OutFP)
END SUBROUTINE ControlFile
! ============================================================================
SUBROUTINE Start(M, N, InputFileName, Normalize, Kmax, Kmin, SeedPointChoice)
        USE Global
        IMPLICIT NONE

        CHARACTER(LEN=*), INTENT(IN) :: InputFileName
        INTEGER, INTENT(IN) :: M, N, Normalize, Kmax, Kmin, SeedPointChoice
        INTEGER :: AllocatStatus, I, J, NUM
        REAL, ALLOCATABLE, DIMENSION(:,:) :: DataMatrix, Seeds

        ALLOCATE( DataMatrix(M,N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(DataMatrix) ** No Enough Memory ***"
        !READ DATA MATRIX FROM INPUT FILE
        CALL ReadDataMatrixFromFile(M, N, DataMatrix, InputFileName)
        ! normalize if requested
        SELECT CASE(Normalize)
        CASE (1)
```

```
                CALL CenterByColMean(M, N, DataMatrix)
        CASE (2)
                CALL NormalizeByColMax(M, N, DataMatrix)
        CASE (3)
                CALL NormalizeByColMinMax(M, N, DataMatrix)
        CASE (4)
                CALL NormalizeByMax(M, N, DataMatrix)
        END SELECT
        ! obtain Seed Points
        ALLOCATE( Seeds(M,Kmax), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(Seeds) ** No Enough Memory ***"
        CALL ObtainSeedPoints(M, N, DataMatrix, Kmax, SeedPointChoice, Seeds)
        ! implement nucleated agglomerative alg.
        CALL DcClustering(M, N, DataMatrix, Kmax, Kmin, Seeds)
        DEALLOCATE( DataMatrix, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(DataMatrix)** Deallocation Error***"
        DEALLOCATE( Seeds, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Seeds)** Deallocation Error***"
END SUBROUTINE Start
! ========================================================================
SUBROUTINE DoClustering(M, N, X, Kmax, Kmin, Centers)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: M, N, Kmax, Kmin
        REAL, DIMENSION(M,N), INTENT(IN) :: X
        REAL, DIMENSION(M,Kmax), INTENT(IN) :: Centers
        INTEGER :: Membership(N), NC(Kmax) !NC: no of members in each cluster
        INTEGER :: I, J, AllocatStatus

        ALLOCATE( Exist(Kmax), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(Exist) ** No Enough Memory ***"
        Exist = .TRUE.
        CALL InitialPartition(M, N, X, Centers, Kmax, Membership)
        CALL UpdateCentersAndNC(M, N, X, Kmax, Membership, Centers, NC)
        CALL OptimizeClustering(M, N, X, Kmax, Membership, Centers, NC)
        CALL WriteKSolution(M, N, X, Centers, Kmax, Kmax, Membership, NC)
        IF ( Kmin < Kmax ) THEN
          CALL Agglomerate(M, N, X, Centers, Kmax, Kmin, Membership, NC)
        END IF
        DEALLOCATE( Exist, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Exist)** Deallocation Error***"
END SUBROUTINE DoClustering
! ========================================================================
SUBROUTINE InitialPartition(M, N, X, Centers, Kmax, Membership)
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: M, N, Kmax
        REAL, DIMENSION(M,N), INTENT(IN) :: X
        REAL, DIMENSION(M,Kmax), INTENT(IN) :: Centers
        INTEGER, DIMENSION(N), INTENT(OUT) :: Membership
        INTEGER :: I, K
        REAL :: Nearest, Dist

        DO I=1, N
          Nearest = SQRT( DOT_PRODUCT( X(1:M,I)-Centers(1:M,1), X(1:M,I)-Centers(1:M,1) ) )
          Membership(I) = 1
                DO K=2, Kmax
                  Dist = SQRT( DOT_PRODUCT( X(1:M,I)-Centers(1:M,K), X(1:M,I)-Centers(1:M,K) ) )
                  IF ( Dist < Nearest ) THEN
                        Nearest = Dist
                        Membership(I) = K
                  END IF
                END DO
        END DO
END SUBROUTINE InitialPartition
! ========================================================================
SUBROUTINE UpdateCentersAndNC(M, N, X, K, Membership, Centers, NC)
    USE Global
```

```
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N, K
    REAL, DIMENSION(M,N), INTENT(IN) :: X
    INTEGER, DIMENSION(N), INTENT(IN) :: Membership
    REAL, DIMENSION(M,K), INTENT(OUT) :: Centers
    INTEGER, DIMENSION(K), INTENT(OUT) :: NC
    INTEGER :: I, J

        Centers = 0.0
        DO I=1, N
          Centers(1:M, Membership(I)) = Centers(1:M, Membership(I)) + &
                & X(1:M,I)
        END DO
        NC = 0
        DO I=1, K
          DO J=1, N
                IF ( Membership(J) == I ) THEN
                        NC(I) = NC(I) + 1
                END IF
          END DO
        END DO
        DO I=1, K
          IF ( Exist(I) ) THEN
                Centers(1:M, I) = Centers(1:M, I) / REAL( NC(I) )
          END IF
        END DO
END SUBROUTINE UpdateCentersAndNC
! ========================================================================
SUBROUTINE OptimizeClustering(M, N, X, Kmax, Membership, Centers, NC)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N, Kmax
    REAL, DIMENSION(M,N), INTENT(IN) :: X
    INTEGER, DIMENSION(N), INTENT(INOUT) :: Membership
    REAL, DIMENSION(M,Kmax), INTENT(INOUT) :: Centers
    INTEGER, DIMENSION(Kmax), INTENT(INOUT) :: NC
    INTEGER :: I, J, K, MoveTo, Cycle
    LOGICAL :: MovedInThisCycle, MoveObj, First
    REAL :: LHS, RHS, Min

    Cycle = 0
    DO
      MovedInThisCycle = .FALSE.
      Cycle = Cycle + 1
      DO I=1, N    ! for all objects
         MoveObj = .FALSE.
         J = Membership(I)
         DO K=1, Kmax    ! for all clusters
           First = .TRUE.
           IF ( J /= K ) THEN
              LHS = REAL(NC(K))/(REAL(NC(K))+1.0) * DOT_PRODUCT(&
              & X(1:M,I)-Centers(1:M,K), X(1:M,I)-Centers(1:M,K))

              IF (NC(J) > 1) THEN
                RHS = REAL(NC(J))/(REAL(NC(J))-1.0)*DOT_PRODUCT( &
                & X(1:M,I)-Centers(1:M,J),X(1:M,I)-Centers(1:M,J))
              ELSE
                RHS = 0.0
              END IF
              IF ( LHS < RHS ) THEN
                MoveObj = .TRUE.
                IF (First) THEN
                  Min = LHS
                  Moveto = K
                  First = .FALSE.
                ELSE
                  IF (LHS < Min) THEN
                    Min = LHS
```

```
                    Moveto = K
                  END IF
                END IF
              END IF
            END IF
          END DO
          IF (MoveObj) THEN
            Membership(I) = MoveTo
            MovedInThisCycle = .TRUE.
            CALL UpdateCentersAndNC(M, N, X, Kmax, Membership, &
                  & Centers, NC)
          END IF
        END DO
        IF (.NOT. MovedInThisCycle ) EXIT   ! if one full cycle has no obj to move
      END DO
END SUBROUTINE OptimizeClustering
! ========================================================================
SUBROUTINE Agglomerate(M, N, X, Centers, Kmax, Kmin, Membership, NC)
      USE Global
      IMPLICIT NONE

      INTEGER, INTENT(IN) :: M, N, Kmax, Kmin
      REAL, DIMENSION(M,N), INTENT(IN) :: X
      INTEGER, DIMENSION(N), INTENT(INOUT) :: Membership
      REAL, DIMENSION(M,Kmax), INTENT(INOUT) :: Centers
      INTEGER, DIMENSION(Kmax), INTENT(INOUT) :: NC
      INTEGER :: I, J, K, Mi, Mj
      LOGICAL :: First
      REAL :: Ex, Min

          K = Kmax
          DO
            First = .TRUE.
            DO I=1, Kmax-1
              DO J=I+1, Kmax
                    IF (Exist(I) .AND. Exist(J) ) THEN
                          Ex = ( REAL(NC(I))*REAL(NC(J)) ) / ( REAL(NC(I))+REAL(NC(J)) ) * &
                          & DOT_PRODUCT( Centers(1:M,I)-Centers(1:M,J), Centers(1:M,I)-Centers(1:M,J) )
                          IF (First) THEN
                            Min = Ex
                            Mi = I
                            Mj = J
                            First = .FALSE.
                          ELSE
                            IF (Ex < Min) THEN
                                  Min = Ex
                                  Mi = I
                                  Mj = J
                            END IF
                          END IF
                    END IF
              END DO
            END DO
            WRITE(OutFP, 20) Mi, Mj, Mi
20          FORMAT ('Merging Clusters ',I3, ' and ',I3,' into cluster ',I3)
            ! update the membership to reflect the merge and the reduction
            ! of the number of cluters
            DO I=1, N
              ! transfer members of Mj cluster to cluster Mi
              IF (Membership(I) == Mj ) THEN
                    Membership(I) = Mi
              END IF
            END DO
            ! reduce number of clusters by one after merging the nearest two
            K = K-1
            ! flag Mj cluster not exist
            Exist(Mj) = .FALSE.
            CALL UpdateCentersAndNC(M, N, X, Kmax, Membership, Centers, NC)
            CALL WriteKSolution(M, N, X, Centers, K, Kmax, Membership, NC)
```

```
          IF (K == Kmin) EXIT
        END DO
END SUBROUTINE Agglomerate
! ========================================================================
SUBROUTINE WriteKSolution(M, N, X, Centers, K, Kmax, Membership, NC)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: M, N, K, Kmax
        REAL, DIMENSION(M,N), INTENT(IN) :: X
        REAL, DIMENSION(M,Kmax), INTENT(IN) :: Centers
        INTEGER, DIMENSION(N), INTENT(IN) :: Membership
        INTEGER, DIMENSION(Kmax), INTENT(IN) :: NC
        INTEGER :: I
        REAL :: WSS(Kmax)  ! Within Sums of Squares
        REAL :: Sc         ! deviation of the points from the
              ! cluster centers =  SQRT( 1/(N-K) * Sum WSS )
        REAL :: RSS        ! Residual Sums of Squares of the solution
              ! involving Kmax clusters, RSS = Sc^2 / m(n-K)
        CHARACTER(LEN=100), ALLOCATABLE, DIMENSION(:) :: ClusteringStructure
        CHARACTER(LEN=100) :: Str1,Str2, Str3

        WRITE(OutFP,'(A)') '   Cluster Membership'
        WRITE(OutFP,'(15I6)') (I, I=1,N)
        WRITE(OutFP,'(15I6)') (Membership(I), I=1,N)
        WRITE(OutFP,'(A)') '   No of Members In Each Cluster'
        WRITE(OutFP '(10I8)') (I, I=1,Kmax)
        WRITE(OutFP,'(10I8)') (NC(I), I=1,Kmax)
        CALL ComputeWSS(M, N, X, Centers, Kmax, Membership, WSS)
        WRITE(OutFP,'(A)') '   The Within sum of squares'
        WRITE(OutFP,'(10I20)') (I, I=1,Kmax)
        WRITE(OutFP,'(10F20.2)') (WSS(I), I=1,Kmax)
        WRITE(OutFP,'(A,F10.2)') '       Total within sum of squares=',SUM(WSS)
        Sc = SQRT( SUM(WSS) / (N-K) )
        WRITE(OutFP,'(A)') '   The deviation of the points from the cluster centers (Sc)'
        WRITE(OutFP,'(A,F10.2)') '      Sc= ', Sc
        RSS =  Sc**2 / (M*(N-K))
        WRITE(OutFP,25) K
25      FORMAT ('   The Residual Sums of Squares of the solution involving',I4,' clusters (RSS)')
        WRITE(OutFP,'(A,F10.2)') '      RSS= ', RSS
        WRITE(OutFP,'(A,I4)') '   The Clustering Structure for k= ',K
        CALL WriteClusteringStructure(N, Kmax, Membership)
        WRITE(OutFP,*)
        WRITE(OutFP,*) '----------------------------------------------------'
        WRITE(OutFP,*)
END SUBROUTINE WriteKSolution
! ========================================================================
SUBROUTINE ComputeWSS(M, N, X, Centers, K, Membership, WSS)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N, K
    REAL, DIMENSION(M,N), INTENT(IN) :: X
    REAL, DIMENSION(M,K), INTENT(IN) :: Centers
    INTEGER, DIMENSION(N), INTENT(IN) :: Membership
    REAL, DIMENSION(K), INTENT(OUT) :: WSS
    INTEGER :: I, J

        WSS = 0.0
        DO I=1, N
           J = Membership(I)
           WSS(J) = WSS(J) + DOT_PRODUCT( X(1:M,I)-Centers(1:M,J), X(1:M,I)-Centers(1:M,J) )
        END DO
END SUBROUTINE ComputeWSS
! ========================================================================
SUBROUTINE WriteClusteringStructure(N, K, Membership)
    USE Global
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: N, K
```

```
    INTEGER, DIMENSION(N), INTENT(IN) :: Membership
    INTEGER :: I, J, Allocatstatus
    CHARACTER(LEN=100), ALLOCATABLE, DIMENSION(:) :: ClusteringStructure
    CHARACTER(LEN=100) :: Str1,Str2, Str3
    LOGICAL :: First

        ALLOCATE( ClusteringStructure(K), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 )  STOP  "(ClusteringStructure) ""No Enough Memory"""
        DO I=1, K
          First = .TRUE.
          WRITE(Str1, '(A1,I1)') '['
                DO J=1, N
                  IF ( Membership(J) == I) THEN
                        IF (First) THEN
                          WRITE(Str2, '(I2)') J
                          First = .FALSE.
                        ELSE
                          WRITE(Str2, '(A1,I2)') ',', J
                        END IF
                        WRITE(Str3, '(A,A)') TRIM(Str1),TRIM(Str2)
                        Str1 = Str3
                  END IF
                END DO
          Str2 = ']'
          WRITE(Str3, '(A,A)') TRIM(Str1), TRIM(Str2)
          ClusteringStructure(I) = Str3
        END DO
        WRITE(OutFP, *) 'Clustering Structure'
        DO J=1, K
          IF ( Exist(J) ) THEN
                WRITE(OutFP,'($,A,3X)') TRIM( ClusteringStructure(J) )
          END IF
        END DO
        DEALLOCATE( ClusteringStructure, STAT = AllocatStatus )
        IF (AllocatStatus/= 0 )  STOP  "(ClusteringStructure) ""Deallocation Error"""
END SUBROUTINE WriteClusteringStructure
! ================================================================================
```

Cluster Validation
Davies-Bouldin Index

```
! ************************************************************************************************************************
! File Name: DBIndex.f
! Author: Ahmad Alhamed
! This program computes the Davies-Bouldin Index. It calls other subroutines and functions from CAlib.f.
! ************************************************************************************************************************
MODULE Global
    IMPLICIT NONE

        INTEGER, PARAMETER :: CntrlFP=10, OutFP=7  ! File Pointers
END MODULE Global
! ======================================================================
PROGRAM DB
    IMPLICIT NONE

    CHARACTER(LEN=20) :: argv

        CALL GETARG(1, argv)
        IF ( LEN(TRIM(argv)) == 0 )  THEN
                PRINT *, 'Usage: db ControlFileName'
        ELSE
                CALL ControlFile(argv)
        END IF
END PROGRAM DB
! ======================================================================
SUBROUTINE ControlFile(ControlFileName)
        USE Global
        IMPLICIT NONE
! ***************************************************************
!    Read the required input from input file called 'Control File'
! ***************************************************************

        CHARACTER(LEN=*), INTENT(IN) :: ControlFileName
        INTEGER :: M, N, IERR
        CHARACTER(LEN=50) :: OutputFile, InputFile
        CHARACTER(LEN=25) :: Prefix
        CHARACTER(LEN=70) :: Title

        OPEN (UNIT=CntrlFP,FILE=ControlFileName,STATUS='OLD',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 3 , CntrlFP,ControlFileName, IERR
3               FORMAT ( ' ERROR OPENING UNIT=',I2,' FILE NAME = ',A15,', IOSTAT = ',I8)
                STOP 3
        ENDIF
        READ (CntrlFP, FMT='(A)') Title
        READ (CntrlFP, *) Prefix
        READ (CntrlFP, *) M, N
        READ (CntrlFP, *) InputFile
        OPEN (UNIT=OutFP,FILE=Prefix,POSITION='APPEND',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 7 , OutFP, OutputFile, IERR
7               FORMAT (' ERROR OPENING UNIT=',I2,' FILE NAME =' ,A13,', IOSTAT = ',I8)
                STOP 7
        END IF
        WRITE (OutFP, '(A)') Title
        WRITE (OutFP, '(A)') "*** Davies-Bouldin index ***"
        WRITE (OutFP, *)
        WRITE (OutFP, *) 'K          DB'
        Call Start(M, N, InputFile)
        WRITE (OutFP, *)
        CLOSE (UNIT=OutFP)
END SUBROUTINE ControlFile
! ======================================================================
SUBROUTINE Start(M, N, InputFileName)
        USE Global
        IMPLICIT NONE

        CHARACTER(LEN=*), INTENT(IN) :: InputFileName
```

```
        INTEGER, INTENT(IN) :: M, N
        INTEGER :: K, AllocatStatus, I, J
        REAL, ALLOCATABLE, DIMENSION(:,:) :: DataMatrix
        INTEGER, ALLOCATABLE, DIMENSION(:) :: Membership, NC, temp

        ALLOCATE( DataMatrix(M,N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(DataMatrix) ** No Enough Memory ***
        !READ DATA MATRIX FROM INPUT FILE
        CALL ReadDataMatrixFromFile(M, N, DataMatrix, InputFileName)
        ALLOCATE(Membership(N))
        DO
          READ (CntrlFP, *) K
          IF (K == 0)          EXIT
          ALLOCATE(NC(K))
          READ(CntrlFP,*) (NC(I), I=1,K)
          DO I=1, K
                ALLOCATE( temp( NC(I) ) )
                READ(CntrlFP,*) (temp(J), J=1, NC(I))
                DO J=1, NC(I)
                   Membership(temp(j)) = I
                END DO
                DEALLOCATE(temp)
          END DO
          ! Implement Davies-Bouldin index.
          CALL CalculateDB(M, N, DataMatrix, K, NC, Membership)
          DEALLOCATE(NC)
        END DO
        DEALLOCATE(Membership)
        DEALLOCATE( DataMatrix, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(DataMatrix)** Deallocation Error***
END SUBROUTINE Start
! ========================================================================
SUBROUTINE CalculateDB(M, N, X, K, NC, Membership)
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: M, N, K
        REAL, DIMENSION(M,N), INTENT(IN) :: X
        INTEGER, DIMENSION(N), INTENT(IN) :: Membership
        INTEGER, DIMENSION(K), INTENT(IN) :: NC
        INTEGER :: I, J, AllocatStatus
        REAL, ALLOCATABLE, DIMENSION(:,:) :: Centers
        REAL, ALLOCATABLE, DIMENSION(:) :: WS  ! Within Scatter

        ALLOCATE( Centers(M,K), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(Centers) ** No Enough Memory ***
        ALLOCATE( WS(K) )
        CALL CalculateCenters(M, N, X, K, Membership, NC, Centers)
        CALL ComputeWScatter(M, N, X, K, Centers, Membership, NC, WS)
        CALL DBindex(M, N, K, Centers, WS)
        DEALLOCATE( Centers, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Centers)** Deallocation Error***
        DEALLOCATE( WS )
END SUBROUTINE CalculateDB
! ========================================================================
SUBROUTINE ComputeWScatter(M, N, X, K, Centers, Membership, NC, WS)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N, K
    REAL, DIMENSION(M,N), INTENT(IN) :: X
    REAL, DIMENSION(M,K), INTENT(IN) :: Centers
    INTEGER, DIMENSION(N), INTENT(IN) :: Membership
    INTEGER, DIMENSION(K), INTENT(IN) :: NC
    REAL, DIMENSION(K), INTENT(OUT) :: WS
    INTEGER :: I, J

        WS = 0.0
        DO I=1, N
           J = Membership(I)
           WS(J) = WS(J) + DOT_PRODUCT( X(1:M,I)-Centers(1:M,J),X(1:M,I)-Centers(1:M,J) )
```

```
        END DO
        DO J=1, K
           WS(J) = SQRT( WS(J) / REAL(NC(J)) )
        END DO
END SUBROUTINE ComputeWScatter
! ================================================================================
SUBROUTINE DBindex(M, N, K, Centers, WS)
    USE Global
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N, K
    REAL, DIMENSION(M,K), INTENT(IN) :: Centers
    REAL, DIMENSION(K), INTENT(IN) :: WS
    INTEGER :: I, J
    LOGICAL :: First
    REAL :: Rij, Max, DB
    REAL, ALLOCATABLE, DIMENSION(:) :: Ri

        ALLOCATE( Ri(K) )
        DO I=1, K
           First = .TRUE.
           DO J=1, K
                IF ( I /=J ) THEN
                        Rij = ( WS(I) + WS(J) ) / &
                        & SQRT( DOT_PRODUCT( &
                        &       Centers(1:M,I)-Centers(1:M,J), &
                        &       Centers(1:M,I)-Centers(1:M,J) ) )
                        IF (First) THEN
                          Max = Rij
                          First = .FALSE.
                        ELSE
                          IF (Rij > Max) THEN
                                Max = Rij
                          END IF
                        END IF
                END IF
           END DO
           Ri(I) = Max
        END DO
        DB = SUM(Ri) / REAL(K)
        WRITE(OutFP,'(I2,10X,F7.2)') K, DB
END SUBROUTINE DBindex
! ================================================================================
```

Cluster Validation
Modified Hubert Index

```
! ************************************************************************************************************************
! File Name: MHindex.f
! Author: Ahmad Alhamed
! This program computes the Modified Hubert Index. It calls other subroutines and functions from CAlib.f.
! ************************************************************************************************************************
MODULE Global
    IMPLICIT NONE

        INTEGER, PARAMETER :: CntrlFP=10, OutFP=7 ! File Pointers
END MODULE Global
! ======================================================================
PROGRAM MH
    IMPLICIT NONE

    CHARACTER(LEN=20) :: argv

        CALL GETARG(1, argv)
        IF ( LEN(TRIM(argv)) == 0 )  THEN
                PRINT *, 'Usage: mh ControlFileName'
        ELSE
                CALL ControlFile(argv)
        END IF
END PROGRAM MH
! ======================================================================
SUBROUTINE ControlFile(ControlFileName)
        USE Global
        IMPLICIT NONE
! ***************************************************************
!    Read the required input from input file called 'Control File'
! ***************************************************************

        CHARACTER(LEN=*), INTENT(IN) :: ControlFileName
        INTEGER :: M, N, IERR
        CHARACTER(LEN=50) :: OutputFile, InputDataFile
        CHARACTER(LEN=25) :: Prefix
        CHARACTER(LEN=70) :: Title

        OPEN (UNIT=CntrlFP,FILE=ControlFileName,STATUS='OLD',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 3 , CntrlFP,ControlFileName, IERR
3               FORMAT ( ' ERROR OPENING UNIT=',I2,' FILE NAME = ',A15, ', IOSTAT = ',I8)
                STOP 3
        ENDIF
        READ (CntrlFP, FMT='(A)') Title
        READ (CntrlFP, *) Prefix
        READ (CntrlFP, *) M, N
        READ (CntrlFP, *) InputDataFile
        OPEN (UNIT=OutFP,FILE=Prefix,POSITION='APPEND',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 7 , OutFP, OutputFile, IERR
7               FORMAT (' ERROR OPENING UNIT=',I2,' FILE NAME =' ,A13,', IOSTAT = ',I8)
                STOP 7
        END IF
        WRITE (OutFP, '(A)') Title
        WRITE (OutFP, '(A)') '***** Modified Hubert index *****'
        WRITE (OutFP, *)
        WRITE (OutFP, *) 'K          MH'
        Call Start(M, N, InputDataFile)
        WRITE (OutFP, *)
        CLOSE (UNIT=OutFP)
END SUBROUTINE ControlFile
! ======================================================================
SUBROUTINE Start(M, N, DataFile)
        USE Global
        IMPLICIT NONE

        CHARACTER(LEN=*), INTENT(IN) :: DataFile
```

```
        INTEGER, INTENT(IN) :: M, N
        INTEGER :: K, AllocatStatus, I, J
        REAL, ALLOCATABLE, DIMENSION(:,:) :: DataMatrix, DissMatrix
        INTEGER, ALLOCATABLE, DIMENSION(:) :: Membership, NC, temp

        ALLOCATE( DataMatrix(M,N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(DataMatrix) ** No Enough Memory ***
        ALLOCATE( DissMatrix(N,N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(DissMatrix) ** No Enough Memory ***
        !READ DATA MATRIX FROM INPUT FILE
        CALL ReadMatrixFromFile(M, N, DataMatrix, DataFile)
        !calculate Dissimilarity MATRIX
        CALL Euclidean(M, N, DataMatrix, DissMatrix)
        ALLOCATE(Membership(N))
        DO
          READ (CntrlFP, *) K
          IF (K == 0)        EXIT
          ALLOCATE(NC(K))
          READ(CntrlFP,*) (NC(I), I=1,K)
          DO I=1, K
                ALLOCATE( temp( NC(I) ) )
                READ(CntrlFP,*) (temp(J), J=1, NC(I))
                DO J=1, NC(I)
                   Membership(temp(j)) = i
                END DO
                DEALLOCATE(temp)
          END DO
          ! implement Modified Hubert index.
          CALL CalculateMH(M, N, DataMatrix, DissMatrix, K, NC, Membership)
          DEALLOCATE(NC)
        END DO
        DEALLOCATE(Membership)
        DEALLOCATE( DataMatrix, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(DataMatrix)** Deallocation Error***
        DEALLOCATE( DissMatrix, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(DissMatrix)** Deallocation Error***
END SUBROUTINE Start
! ======================================================================
SUBROUTINE CalculateMH(M, N, X, D1, K, NC, Membership)
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: M, N, K
        REAL, DIMENSION(M,N), INTENT(IN) :: X
        REAL, DIMENSION(N,N), INTENT(IN) :: D1
        INTEGER, DIMENSION(N), INTENT(IN) :: Membership
        INTEGER, DIMENSION(K), INTENT(IN) :: NC
        INTEGER :: I, J, AllocatStatus
        REAL, ALLOCATABLE, DIMENSION(:,:) :: Centers
        REAL, ALLOCATABLE, DIMENSION(:,:) :: D2  ! model matrix

        ALLOCATE( Centers(M,K), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(Centers) ** No Enough Memory ***
        ALLOCATE( D2(N,N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(D2) ** No Enough Memory ***
        CALL CalculateCenters(M, N, X, K, Membership, NC, Centers)
        CALL ComputeModelMatrix(M, N, K, Centers, Membership, D2)
        CALL normalizedMHindex(N, K, D1, D2)
        DEALLOCATE( Centers, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Centers)** Deallocation Error***
        DEALLOCATE( D2, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(D2)** Deallocation Error***
END SUBROUTINE CalculateMH
! ======================================================================
SUBROUTINE ComputeModelMatrix(M, N, K, Centers, Membership, D2)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N, K
    REAL, DIMENSION(M,K), INTENT(IN) :: Centers
    INTEGER, DIMENSION(N), INTENT(IN) :: Membership
```

```
    REAL, DIMENSION(N,N), INTENT(OUT) :: D2
    INTEGER :: I, J, P, Q

        D2 = 0.0
        DO I=1, N-1
           DO J=I+1, N
                IF ( Membership(I) == Membership(J) ) THEN
                   D2(I,J) = 0.0
                   D2(J,I) = 0.0
                ELSE
                   P = Membership(I)
                   Q = Membership(J)
                   D2(I,J) = SQRT( DOT_PRODUCT( &
                   &  Centers(1:M,P)-Centers(1:M,Q), &
                   &       Centers(1:M,P)-Centers(1:M,Q) ) )
                   D2(J,I) = D2(I,J)
                END IF
          END DO
        END DO
END SUBROUTINE ComputeModelMatrix
! ========================================================================
SUBROUTINE normalizedMHindex(N, K, D1, D2)
    USE Global
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: N, K
    REAL, DIMENSION(N,N), INTENT(IN) :: D1, D2
    INTEGER :: I, J, M  ! M is the num of entries
    REAL :: normalizedMH
    REAL :: M1, M2   ! means
    REAL :: s1, s2   ! Standard deviations

        M = N*(N-1) / 2
        M1 = 0.0
        M2 = 0.0
        DO I=1, N-1
           DO J=I+1, N
                M1 = M1 + D1(I,J)
                M2 = M2 + D2(I,J)
           END DO
        END DO
        M1 = M1 / REAL(M)
        M2 = M2 / REAL(M)
        S1 = 0.0
        S2 = 0.0
        DO I=1, N-1
           DO J=I+1, N
                S1 = S1 + (D1(I,J)**2 - M1**2 )
                S2 = S2 + (D2(I,J)**2 - M2**2 )
           END DO
        END DO
        S1 = SQRT( S1 / REAL(M) )
        S2 = SQRT( S2 / REAL(M) )
        normalizedMH = 0.0
        DO I=1, N-1
           DO J=I+1, N
                normalizedMH = normalizedMH + ( &
                & (D1(I,J)-M1)*(D2(I,J)-M2) / (S1*S2) )
           END DO
        END DO
        normalizedMH = normalizedMH / REAL(M)
        WRITE(OutFP,'(I2,10X,F7.2)') K, normalizedMH
END SUBROUTINE normalizedMHindex
! ========================================================================
```

Cluster Validation
Dunn's Index
and its generalized versions

```
! ************************************************************************************************************
! File Name: DunnIndex.f
! Author: Ahmad Alhamed
! This program computes the Dunn's index Index, and its generalized versions. It calls other subroutines and functions
! from CAlib.f.
!
! The definitions and choices of the diameters and the set distances are based of the paper by Bezdek and Pal:
! Bezdek, J. C. and Pal N. R. "Some New Indexes of Cluster Validity", IEEE Trans. on Systems, Man, and Cybernetics –
! Part B: Cybernetics, V. 28, No. 3, June 1998, pp. 301-15.
!
! The definition of diameter 4 is based on the paper by Pal and Biswas:
! Pal, N. R. and Biswas, J. "Cluster Validation using graph theoretic concepts". Pattern Recognition, V. 30, N. 6, 1997.
! pp. 847-857.
! ************************************************************************************************************
MODULE Global
        IMPLICIT NONE

        INTEGER, PARAMETER :: CntrlFP=10, OutFP=7  ! File Pointers
        INTEGER :: DiameterChoice, SetDistanceChoice
END MODULE Global
! ========================================================================
PROGRAM Dunn
    IMPLICIT NONE

    CHARACTER(LEN=20) :: argv

        CALL GETARG(1, argv)
        IF ( LEN(TRIM(argv)) == 0 )  THEN
                PRINT *, 'Usage: dn ControlFileName'
        ELSE
                CALL ControlFile(argv)
        END IF
END PROGRAM Dunn
! ========================================================================
SUBROUTINE ControlFile(ControlFileName)
        USE Global
        IMPLICIT NONE
! ********************************************************************
!    Read the required input from input file called 'Control File'
! ********************************************************************

        CHARACTER(LEN=*), INTENT(IN) :: ControlFileName
        INTEGER :: M, N, IERR
        CHARACTER(LEN=50) :: OutputFile, InputDataFile
        CHARACTER(LEN=25) :: Prefix
        CHARACTER(LEN=2) :: t1, t2
        CHARACTER(LEN=70) :: Title

        OPEN (UNIT=CntrlFP,FILE=ControlFileName,STATUS='OLD',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 3 , CntrlFP,ControlFileName, IERR
3               FORMAT ( ' ERROR OPENING UNIT=',I2,' FILE NAME = ',A15, ', IOSTAT = ',I8)
                STOP 3
        ENDIF
        READ (CntrlFP, FMT='(A)') Title
        READ (CntrlFP, *) Prefix
        READ (CntrlFP, *) M, N
        READ (CntrlFP, *) InputDataFile
        READ (CntrlFP, *) DiameterChoice
        READ (CntrlFP, *) SetDistanceChoice
        WRITE(t1, '(I1)') DiameterChoice
        WRITE(t2, '(I1)') SetDistanceChoice
        OPEN (UNIT=OutFP,FILE=Prefix,POSITION='APPEND',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 7 , OutFP, OutputFile, IERR
```

```
7               FORMAT (' ERROR OPENING UNIT=',I2,' FILE NAME =' ,A13,', IOSTAT = ',I8)
                STOP 7
        END IF
        WRITE (OutFP, '(A)') Title
        IF (DiameterChoice==4 .AND. SetDistanceChoice==4) THEN
          WRITE (OutFP, '(A)') "***** Dunn's index based on Gabriel Graph (DunnGG) *****
        ELSE
          WRITE (OutFP, '(A)') "***** Dunn's index *****
          WRITE (OutFP, 10) DiameterChoice, SetDistanceChoice
        END IF
10      FORMAT ('---- Diameter:',I2,',   Set Distance:',I2,' ----')
        WRITE (OutFP, *)
        IF (DiameterChoice==4 .AND. SetDistanceChoice==4) THEN
          WRITE (OutFP, *) "K          DunnGG"
        ELSE
          WRITE (OutFP, *) "K          GDunn's"
        END IF
        Call Start(M, N, InputDataFile)
        WRITE (OutFP, *)
        CLOSE (UNIT=OutFP)
END SUBROUTINE ControlFile
! ======================================================================
SUBROUTINE Start(M, N, DataFile)
        USE Global
        IMPLICIT NONE

        CHARACTER(LEN=*), INTENT(IN) :: DataFile
        INTEGER, INTENT(IN) :: M, N
        INTEGER :: K, AllocatStatus, I, J
        REAL, ALLOCATABLE, DIMENSION(:,:) :: DataMatrix, DistMatrix
        INTEGER, ALLOCATABLE, DIMENSION(:) :: Membership, NC, temp

        ALLOCATE( DataMatrix(M,N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 )  STOP "(DataMatrix) ** No Enough Memory ***
        ALLOCATE( DistMatrix(N,N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 )  STOP "(DistMatrix) ** No Enough Memory ***
        !READ DATA MATRIX FROM INPUT FILE
        CALL ReadMatrixFromFile(M, N, DataMatrix, DataFile)
        !calculate Dissimilarity MATRIX
        CALL Euclidean(M, N, DataMatrix, DistMatrix)
        ALLOCATE(Membership(N))
        DO
          READ (CntrlFP, *) K
          IF (K == 0)          EXIT
          ALLOCATE(NC(K))
          READ(CntrlFP,*) (NC(I), I=1,K)
          DO I=1, K
                ALLOCATE( temp( NC(I) ) )
                READ(CntrlFP,*) (temp(J), J=1, NC(I))
                DO J=1, NC(I)
                   Membership(temp(j)) = I
                END DO
                DEALLOCATE(temp)
          END DO
          ! implement Dunn's index.
          CALL ToCalculateDunn(M, N, DataMatrix, DistMatrix, K, NC, Membership)

          DEALLOCATE(NC)
        END DO
        DEALLOCATE(Membership)
        DEALLOCATE( DataMatrix, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 )  STOP "(DataMatrix)** Deallocation Error***
        DEALLOCATE( DistMatrix, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 )  STOP "(DistMatrix)** Deallocation Error***
END SUBROUTINE Start
! ======================================================================
SUBROUTINE ToCalculateDunn(M, N, X, D, K, NC, Membership)
        IMPLICIT NONE
```

```
        INTEGER, INTENT(IN) :: M, N, K
        REAL, DIMENSION(M,N), INTENT(IN) :: X          ! data matrix
        REAL, DIMENSION(N,N), INTENT(IN) :: D          ! distance matrix
        INTEGER, DIMENSION(N), INTENT(IN) :: Membership
        INTEGER, DIMENSION(K), INTENT(IN) :: NC
        INTEGER :: I, J, AllocatStatus
        REAL, ALLOCATABLE, DIMENSION(:,:) :: Centers
        REAL :: Diameter
        REAL, ALLOCATABLE, DIMENSION(:,:) :: SD      ! Set Distances btw the K clusters

        ALLOCATE( SD(K,K), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 )  STOP  "(SD) ** No Enough Memory ***
        ALLOCATE( Centers(M,K), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 )  STOP  "(Centers) ** No Enough Memory ***
        CALL CalculateCenters(M, N, X, K, Membership, NC, Centers)
        CALL ComputeDiameter(M, N, X, K, Centers, Membership, NC, D, Diameter)
        CALL ComputeSD(M, N, X, K, Centers, Membership, NC, D, SD)
        CALL DunnIndex(K, Diameter, SD)
        DEALLOCATE( Centers, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 )  STOP  "(Centers)** Deallocation Error***
        DEALLOCATE( SD, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 )  STOP  "(SD)** Deallocation Error***
END SUBROUTINE ToCalculateDunn
! ======================================================================
SUBROUTINE ComputeDiameter(M, N, X, K, Centers, Membership, NC, D, Diameter)
    USE Global
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N, K
    REAL, DIMENSION(M,N), INTENT(IN) :: X
    REAL, DIMENSION(M,K), INTENT(IN) :: Centers
    INTEGER, DIMENSION(N), INTENT(IN) :: Membership
    INTEGER, DIMENSION(K), INTENT(IN) :: NC
    REAL, DIMENSION(N,N), INTENT(IN) :: D
    REAL, INTENT(OUT) :: Diameter
    INTEGER :: I, J, P, Q, AllocatStatus
    REAL, ALLOCATABLE, DIMENSION(:) :: DiamK
    INTEGER, ALLOCATABLE, DIMENSION(:) :: inClusterI
    REAL :: Diameter1, Diameter2, Diameter4 !functions
    REAL :: Diameter3
                ! refer to the paper for definition of each one

        ALLOCATE(DiamK(K))
        DO I=1, K
           ! 1. extract the members of the current cluster
           ALLOCATE(inClusterI( NC(I) ) )
           P = 1
           DO J=1, N
                IF ( Membership(J) == I ) THEN
                   inClusterI(P) = J
                   P = P + 1
                END IF
           END DO
          ! 2. Find the diameter for the current cluster
          SELECT CASE(DiameterChoice)       ! it's global var
          CASE (1)
                DiamK(I) = Diameter1( N, NC(I), inClusterI, D)
          CASE (2)
                DiamK(I) = Diameter2( N, NC(I), inClusterI, D)
          CASE (3)
                Diameter3 = 0.0
                DO P=1, NC(I)
                  Diameter3 = Diameter3 + SQRT( DOT_PRODUCT( &
                  & X(1:M,inClusterI(P))-Centers(1:M,I), &
                  & X(1:M,inClusterI(P))-Centers(1:M,I)   ) )
                END DO
                DiamK(I) = 2.0 * ( Diameter3 / REAL(NC(I)) )
          CASE (4)
                DiamK(I) = Diameter4( N, NC(I), inClusterI, D)
```

```
            END SELECT

            ! 3. deallocate current cluster
            DEALLOCATE(inClusterI)
          END DO
          Diameter = MAXVAL(DiamK)
          DEALLOCATE(DiamK)
END SUBROUTINE ComputeDiameter
! ============================================================================
REAL FUNCTION Diameter1( N, NCi, inClusterI, D)
      IMPLICIT NONE

      INTEGER, INTENT(IN) :: N, NCi
      INTEGER, DIMENSION(NCi), INTENT(IN) :: inClusterI
      REAL, DIMENSION(N,N), INTENT(IN) :: D
      INTEGER :: I, J
      REAL :: Max

          IF (NCi > 1) THEN
            Max = D(inClusterI(1), inClusterI(2) )
            DO I=1, NCi-1
                    DO J=I+1, NCi
                      IF ( D(inClusterI(I), inClusterI(J)) > Max ) THEN
                              Max = D(inClusterI(I), inClusterI(J) )
                      END IF
                    END DO
            END DO
            Diameter1 = Max
          ELSE
            Diameter1 = 0.0
          END IF
END FUNCTION Diameter1
! ============================================================================
REAL FUNCTION Diameter2( N, NCi, inClusterI, D)
      IMPLICIT NONE

      INTEGER, INTENT(IN) :: N, NCi
      INTEGER, DIMENSION(NCi), INTENT(IN) :: inClusterI
      REAL, DIMENSION(N,N), INTENT(IN) :: D
      INTEGER :: I, J
      REAL :: SumDist

          IF (NCi > 1) THEN
            SumDist = 0.0
            DO I=1, NCi
                    DO J=1, NCi
                      IF ( I /= J ) THEN
                              SumDist = SumDist + D(inClusterI(I), inClusterI(J) )
                      END IF
                    END DO
            END DO
            Diameter2 = (1.0/ (REAL(NCi)*REAL(NCi-1)) ) * SumDist
          ELSE
            Diameter2 = 0.0
          END IF
END FUNCTION Diameter2
! ============================================================================
REAL FUNCTION Diameter4( N, NCi, inClusterI, D)
      IMPLICIT NONE

      INTEGER, INTENT(IN) :: N, NCi
      INTEGER, DIMENSION(NCi), INTENT(IN) :: inClusterI
      REAL, DIMENSION(N,N), INTENT(IN) :: D
      INTEGER :: I, J, K, IND
      REAL :: SDist  ! Square Distance
      REAL :: temp
      REAL, DIMENSION(N*(N-1)/2) :: GG          ! Gabriel Graph
      LOGICAL :: Edge
```

```
        IF (NCi > 1) THEN
          IND = 0
          GG = 0.0
          DO I=1, NCi-1
                  DO J=I+1, NCi
                    SDist = D(inClusterI(I), inClusterI(J))**2.0
                    Edge = .TRUE.
                    DO K=1, NCi
                      IF ( K/=I .AND. K/=J ) THEN
                              temp = D(inClusterI(I),inClusterI(K))**2.0 &
                              &    + D(inClusterI(J),inClusterI(K))**2.0
                              IF (SDist >= temp ) THEN
                                Edge = .FALSE.
                                EXIT
                              END IF
                      END IF
                    END DO          ! DO K
                    IF (Edge) THEN
                              IND = IND + 1
                              GG(IND) = D(inClusterI(I), inClusterI(J))
                    END IF
                  END DO  ! DO J
          END DO          ! DO I
          Diameter4 = MAXVAL(GG)
        ELSE
          Diameter4 = 0.0
        END IF
END FUNCTION Diameter4
! ========================================================================
SUBROUTINE ComputeSD(M, N, X, K, Centers, Membership, NC, D, SD)
    USE Global
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N, K
    REAL, DIMENSION(M,N), INTENT(IN) :: X
    REAL, DIMENSION(M,K), INTENT(IN) :: Centers
    INTEGER, DIMENSION(N), INTENT(IN) :: Membership
    INTEGER, DIMENSION(K), INTENT(IN) :: NC
    REAL, DIMENSION(N,N), INTENT(IN) :: D
    REAL, DIMENSION(K,K), INTENT(OUT) :: SD
    INTEGER :: I, J, P, Q, AllocatStatus
    INTEGER, ALLOCATABLE, DIMENSION(:) :: inClusterI, inClusterJ
    REAL :: SD1, SD2, SD3, SD4 !functions
                    ! refer to the paper for definition of each one

        SD = 0.0  ! initialize SD for the elements in the diagonal
        DO I=1, K-1
           ! 1. extract the members of the cluster I
           ALLOCATE(inClusterI( NC(I) ) )
           P = 1
           DO Q=1, N
                   IF ( Membership(Q) == I ) THEN
                     inClusterI(P) = Q
                     P = P + 1
                   END IF
           END DO
           DO J=I+1, K
                   ! 2. extract the members of the cluster J
              ALLOCATE(inClusterJ( NC(J) ) )
              P = 1
              DO Q=1, N
                           IF ( Membership(Q) == J ) THEN
                             inClusterJ(P) = Q
                        P = P + 1
                      END IF
                    END DO
                    ! 3. Find the set distance btw cluster I and J
                    SELECT CASE(SetDistanceChoice)   ! it's global var
                    CASE (1)
```

```
                    SD(I,J) = SD1(N,NC(I),NC(J),inClusterI,inClusterJ,D)
                CASE (2)
                    SD(I,J) = SD2(N,NC(I),NC(J),inClusterI,inClusterJ,D)
                CASE (3)
                    SD(I,J) = SD3(N,NC(I),NC(J),inClusterI,inClusterJ,D)
                CASE (4)
                   SD(I,J) = SQRT( DOT_PRODUCT( Centers(1:M,I)-Centers(1:M,J), &
                   & Centers(1:M,I)-Centers(1:M,J) ) )
                END SELECT
                SD(J,I) = SD(I,J)
            ! 4. deallocate cluster J
              DEALLOCATE(inClusterJ)
          END DO
          ! 5. deallocate cluster I
          DEALLOCATE(inClusterI)
        END DO
END SUBROUTINE ComputeSD
! ============================================================================
REAL FUNCTION SD1(N, NCi, NCj, inClusterI, inClusterJ, D)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: N, NCi, NCj
    INTEGER, DIMENSION(NCi), INTENT(IN) :: inClusterI
    INTEGER, DIMENSION(NCj), INTENT(IN) :: inClusterJ
    REAL, DIMENSION(N,N), INTENT(IN) :: D
    INTEGER :: I, J
    REAL :: Min

        Min = D(inClusterI(1), inClusterJ(1) )
        DO I=1, NCi
          DO J=1, NCj
                IF ( D(inClusterI(I), inClusterJ(J)) < Min ) THEN
                    Min = D(inClusterI(I), inClusterJ(J) )
                END IF
          END DO
        END DO
        SD1 = Min
END FUNCTION SD1
! ============================================================================
REAL FUNCTION SD2(N, NCi, NCj, inClusterI, inClusterJ, D)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: N, NCi, NCj
    INTEGER, DIMENSION(NCi), INTENT(IN) :: inClusterI
    INTEGER, DIMENSION(NCj), INTENT(IN) :: inClusterJ
    REAL, DIMENSION(N,N), INTENT(IN) :: D
    INTEGER :: I, J
    REAL :: Max

        Max = D(inClusterI(1), inClusterJ(1) )
        DO I=1, NCi
          DO J=1, NCj
                IF ( D(inClusterI(I), inClusterJ(J)) > Max ) THEN
                    Max = D(inClusterI(I), inClusterJ(J) )
                END IF
          END DO
        END DO
        SD2 = Max
END FUNCTION SD2
! ============================================================================
REAL FUNCTION SD3(N, NCi, NCj, inClusterI, inClusterJ, D)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: N, NCi, NCj
    INTEGER, DIMENSION(NCi), INTENT(IN) :: inClusterI
    INTEGER, DIMENSION(NCj), INTENT(IN) :: inClusterJ
    REAL, DIMENSION(N,N), INTENT(IN) :: D
    INTEGER :: I, J
    REAL :: SumDist
```

```
        SumDist = 0.0
        DO I=1, NCi
          DO J=1, NCj
                SumDist = SumDist + D(inClusterI(I), inClusterJ(J) )
          END DO
        END DO
        SD3 = SumDist / ( REAL(NCi)*REAL(NCj) )
END FUNCTION SD3
! ========================================================================
SUBROUTINE DunnIndex(K, Diameter, SD)
    USE Global
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: K
    REAL, INTENT(IN) :: Diameter
    REAL, DIMENSION(K,K), INTENT(IN) :: SD
    INTEGER :: I, J
    REAL, ALLOCATABLE, DIMENSION(:) :: Dunn
    REAL :: Min, Exp
    LOGICAL :: First

        ALLOCATE(Dunn(K))
        DO I=1, K
          First = .TRUE.
          DO J=1, K
                IF ( J /= I ) THEN
                  Exp = SD(I,J) / Diameter
                  IF (First) THEN
                        Min = Exp
                        First = .FALSE.
                  ELSE
                        IF ( Exp < Min ) THEN
                          Min = Exp
                        END IF
                  END IF
                END IF
          END DO
          Dunn(I) = Min
        END DO
        WRITE(OutFP,'(I2,10X,F7.2)') K, MINVAL(Dunn)
        DEALLOCATE(Dunn)
END SUBROUTINE DunnIndex
! ========================================================================
```

```
! ***************************************************************************************************
! File Name: MakeSamples.f
! Author: Ahmad Alhamed
! This program divide the data matrix randomly into two samples of equal size to be used in the replication analysis.
! It calls other subroutines and functions from CAlib.f.
! ***************************************************************************************************
PROGRAM MakeSamples
        IMPLICIT NONE

        INTEGER, PARAMETER :: M=9477, N=25
        INTEGER :: AllocatStatus, I, J, R, Size2
        REAL, ALLOCATABLE, DIMENSION(:,:) :: X
        INTEGER :: IND(M)  ! indexes for sample 1 and 2
        REAL :: r1
        CHARACTER(LEN=50) :: InputFile, OutputFile
        INTEGER :: cnt1, cnt2, cnt0, i

        CALL GETARG(1,InputFile)
        IF(LEN(TRIM(InputFile)) == 0) THEN
                PRINT '($,A)', 'Enter data matrix file:'
                READ *, InputFile
        END IF
        ALLOCATE( X(M,N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 )  STOP  "(X) ** No Enough Memory **"
        CALL ReadDataMatrixFromFile(M, N, X, InputFile)
        IND = 1                 ! Initialize all as Sample 1
        Size2 = 0
        CALL RANDOM_SEED
        ! Construct Sample2
        DO
                CALL RANDOM_NUMBER(r1)
                R = INT(r1*10000.0)
                IF ( R<=M .AND. IND(R)==1 ) THEN
                        IND(R) = 2
                        Size2 = Size2 + 1
                END IF
                IF ( Size2 == 4738 )  EXIT
        END DO
        ! Determine the extra row
        DO
                CALL RANDOM_NUMBER(r1)
                R = INT(r1*10000.0)
                IF ( R<=M .AND. IND(R)==1 ) THEN
                        IND(R) = 0
                        EXIT
                END IF
        END DO
        ! 2: writing samples with the extra row
        OutputFile = TRIM(InputFile)//'.S1'
        OPEN(3,file=OutputFile,form='formatted')
        DO I=1,M
          IF ( IND(I)==1 .OR. IND(I)==0 ) THEN
                WRITE(3,*) (X(I,J),J=1,N)
          END IF
        END DO
        CLOSE(3)
        OutputFile = TRIM(InputFile)//'.S2'
        OPEN(3,file=OutputFile,form='formatted')
        DO I=1,M
          IF ( IND(I)==2 .OR. IND(I)==0 ) THEN
                WRITE(3,*) (X(I,J),J=1,N)
          END IF
        END DO
        CLOSE(3)
        DEALLOCATE( X, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(X) **DeallocationError**"
END PROGRAM MakeSamples
! =========================================================================
```

```
! ***************************************************************************************************************
! File Name: replicate.f
! Author: Ahmad Alhamed
! This program performs the replication analysis. It calls other subroutines and functions from CAlib.f.
! ***************************************************************************************************************
MODULE Global
    IMPLICIT NONE

        INTEGER, PARAMETER :: CntrlFP=10, OutFP=7  ! File Pointers
END MODULE Global
! ===========================================================================
PROGRAM Replicate
    IMPLICIT NONE

    CHARACTER(LEN=20) :: argv

        CALL GETARG(1, argv)
        IF ( LEN(TRIM(argv)) == 0 )  THEN
                PRINT *, 'Usage: hr ControlFileName'
        ELSE
                CALL ControlFile(argv)
        END IF
END PROGRAM Replicate
! ===========================================================================
SUBROUTINE ControlFile(ControlFileName)
        USE Global
        IMPLICIT NONE
! ***************************************************************
!    Read the required input from input file called 'Control File'
! ***************************************************************

        CHARACTER(LEN=*), INTENT(IN) :: ControlFileName
        INTEGER :: M, N, IERR
        CHARACTER(LEN=50) :: OutputFile, Sample1, Sample2
        CHARACTER(LEN=25) :: Prefix
        CHARACTER(LEN=70) :: Title
        INTEGER :: K, AllocatStatus, I, J
        REAL, ALLOCATABLE, DIMENSION(:,:) :: X, Y
        INTEGER, ALLOCATABLE, DIMENSION(:) :: MembershipX, ncX, temp

        OPEN (UNIT=CntrlFP,FILE=ControlFileName,STATUS='OLD',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 3 , CntrlFP,ControlFileName, IERR
3               FORMAT ( ' ERROR OPENING UNIT=',I2,' FILE NAME = ',A20, ', IOSTAT = ',I8)
                STOP 3
        ENDIF
        READ (CntrlFP, FMT='(A)') Title
        READ (CntrlFP, *) Prefix
        READ (CntrlFP, *) Sample1
        READ (CntrlFP, *) Sample2
        READ (CntrlFP, *) M, N
        OPEN (UNIT=OutFP,FILE=Prefix,POSITION='APPEND',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 7 , OutFP, OutputFile, IERR
7               FORMAT (' ERROR OPENING UNIT=',I2,' FILE NAME =' ,A13,', IOSTAT = ',I8)
                STOP 7
        END IF
        WRITE (OutFP, '(A)') Title
        WRITE (OutFP, '(A)') "**** Replication Analysis ****"
        WRITE (OutFP, *)
        ALLOCATE(MembershipX(N))
        READ (CntrlFP, *) K
        ALLOCATE(ncX(K))
        READ(CntrlFP,*) (ncX(I), I=1,K)
        DO I=1, K
                ALLOCATE( temp( ncX(I) ) )
                READ(CntrlFP,*) (temp(J), J=1, ncX(I))
                DO J=1, ncX(I)
```

```
                    MembershipX(temp(J)) = I
                  END DO
                  DEALLOCATE(temp)
        END DO
        WRITE (OutFP, *)
        WRITE (OutFP, 20) K
        CALL WriteClusteringStructure(N, K, MembershipX)
20      FORMAT('Sample 1: ',I2,' Clusters')
        ALLOCATE( X(M,N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(X) ** No Enough Memory ***"
        ALLOCATE( Y(M,N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(Y) ** No Enough Memory ***"
        CALL ReadDataMatrixFromFile(M, N, X, Sample1)
        CALL ReadDataMatrixFromFile(M, N, Y, Sample2)
        CALL NearestCentroid(M, N, X, Y, K, MembershipX, ncX)
        WRITE (OutFP, *)
        CALL NearestNeighbor(M, N, X, Y, K, MembershipX)
        WRITE (OutFP, *)
        DEALLOCATE(X, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(X)** Deallocation Error***"
        DEALLOCATE(Y, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Y)** Deallocation Error***"
        DEALLOCATE(MembershipX)
        DEALLOCATE(ncX)
        CLOSE (UNIT=OutFP)
END SUBROUTINE ControlFile
! ========================================================================
SUBROUTINE NearestCentroid(M, N, X, Y, K, MembershipX, ncX)
    USE Global
    IMPLICIT NONE

        INTEGER, INTENT(IN) :: M, N, K
        REAL, DIMENSION(M,N), INTENT(IN) :: X, Y
        INTEGER, DIMENSION(N), INTENT(IN) :: MembershipX
        INTEGER, DIMENSION(K), INTENT(IN) :: ncX
        INTEGER :: I, J, AllocatStatus, Nearest
        REAL, ALLOCATABLE, DIMENSION(:,:) :: Centers
        INTEGER, ALLOCATABLE, DIMENSION(:) :: MembershipY
        REAL :: Min, Dist

        ALLOCATE( Centers(M,K), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "(Centers) ** No Enough Memory ***"
        CALL CalculateCenters(M, N, X, K, MembershipX, ncX, Centers)
        ALLOCATE(MembershipY(N))
        MembershipY = 0
        DO I=1, N
           Min = SQRT( DOT_PRODUCT( Y(1:M,I)-Centers(1:M,1),Y(1:M,I)-Centers(1:M,1) ) )
           Nearest = 1
           DO J=2, K
                 Dist = SQRT( DOT_PRODUCT(  Y(1:M,I)-Centers(1:M,J),Y(1:M,I)-Centers(1:M,J) ) )
                 IF (Dist < Min) THEN
                   Min = Dist
                   Nearest = J
                 END IF
           END DO
           MembershipY(I) = Nearest
        END DO
        WRITE(OutFP,*)
        WRITE(OutFP,*) 'Nearest Centroid Replica'
        CALL WriteClusteringStructure(N, K, MembershipY)
        WRITE(OutFP,*) '-------------------------------------------------'
        WRITE(OutFP,*)
        DEALLOCATE( Centers, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "(Centers)** Deallocation Error***"
        DEALLOCATE(MembershipY)
END SUBROUTINE NearestCentroid
! ========================================================================
SUBROUTINE NearestNeighbor(M, N, X, Y, K, MembershipX)
    USE Global
```

```
    IMPLICIT NONE

        INTEGER, INTENT(IN) :: M, N, K
        REAL, DIMENSION(M,N), INTENT(IN) :: X, Y
        INTEGER, DIMENSION(N), INTENT(IN) :: MembershipX
        INTEGER :: I, J, AllocatStatus, Nearest
        INTEGER, ALLOCATABLE, DIMENSION(:) :: MembershipY
        REAL :: Min, Dist

        ALLOCATE(MembershipY(N))
        MembershipY = 0
        DO I=1, N
           Min = SQRT( DOT_PRODUCT( Y(1:M,I)-X(1:M,1),Y(1:M,I)-X(1:M,1) ) )
           Nearest = 1
           DO J=2, N
                 Dist = SQRT( DOT_PRODUCT(  Y(1:M,I)-X(1:M,J),Y(1:M,I)-X(1:M,J) ) )
                 IF (Dist < Min) THEN
                   Min = Dist
                   Nearest = J
                 END IF
           END DO
           MembershipY(I) = MembershipX(Nearest)
        END DO
        WRITE(OutFP,*) 'Nearest Neighbor Replica'
        CALL WriteClusteringStructure(N, K, MembershipY)
        WRITE(OutFP,*) '----------'
        WRITE(OutFP,*)
        DEALLOCATE(MembershipY)
END SUBROUTINE NearestNeighbor
! ========================================================================
SUBROUTINE WriteClusteringStructure(N, K, Membership)
    USE Global
    IMPLICIT NONE

        INTEGER, INTENT(IN) :: N, K
        INTEGER, DIMENSION(N), INTENT(IN) :: Membership
        INTEGER :: I, J, Allocatstatus
        CHARACTER(LEN=100), ALLOCATABLE, DIMENSION(:) :: ClusteringStructure
        CHARACTER(LEN=100) :: Str1,Str2, Str3
        LOGICAL :: First

        ALLOCATE( ClusteringStructure(K), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 )  STOP  "(ClusteringStructure) ""No Enough Memory"""
        DO I=1, K
           First = .TRUE.
           WRITE(Str1, '(A1,I1)') '['
           DO J=1, N
                 IF ( Membership(J) == I) THEN
                       IF (First) THEN
                         WRITE(Str2, '(I2)') J
                         First = .FALSE.
                       ELSE
                         WRITE(Str2, '(A1,I2)') ',', J
                       END IF
                       WRITE(Str3, '(A,A)') TRIM(Str1),TRIM(Str2)
                       Str1 = Str3
                 END IF
           END DO
           Str2 = ']'
           WRITE(Str3, '(A,A)') TRIM(Str1), TRIM(Str2)
           ClusteringStructure(I) = Str3
        END DO
        DO J=1, K
                 WRITE(OutFP, *) TRIM( ClusteringStructure(J) )
        END DO
        DEALLOCATE( ClusteringStructure, STAT = AllocatStatus )
        IF (AllocatStatus/= 0 )  STOP  "(ClusteringStructure) ""Deallocation Error"""
END SUBROUTINE WriteClusteringStructure
! ========================================================================
```

Measures of Agreement between Partitions

```
! ************************************************************************************************************
! File Name: agreement.f
! Author: Ahmad Alhamed
! This program computes the agreement between two partitions using Rand and Jaccard indexes.
! ************************************************************************************************************
MODULE Global
    IMPLICIT NONE

        INTEGER, PARAMETER :: CntrlFP=10, OutFP=7  ! File Pointers
END MODULE Global
! ======================================================================
PROGRAM Main
    IMPLICIT NONE

    CHARACTER(LEN=20) :: argv

        CALL GETARG(1, argv)
        IF ( LEN(TRIM(argv)) == 0 )  THEN
                PRINT *, 'Usage: hr ControlFileName'
        ELSE
                CALL ControlFile(argv)
        END IF
END PROGRAM Main
! ======================================================================
SUBROUTINE ControlFile(ControlFileName)
    USE Global
        IMPLICIT NONE
! ***************************************************************
!    Read the required input from input file called 'Control File'
! ***************************************************************

        CHARACTER(LEN=*), INTENT(IN) :: ControlFileName

        INTEGER :: N, IERR
        CHARACTER(LEN=50) :: OutputFile
        CHARACTER(LEN=25) :: Prefix
        CHARACTER(LEN=70) :: Title
        INTEGER :: R, C, AllocatStatus, I, J
        INTEGER, ALLOCATABLE, DIMENSION(:) :: U, V, NU, NV, temp
                ! U and V have the membership of their partitions,
                ! R and C are the number of clusters in U and V respectively

        OPEN (UNIT=CntrlFP,FILE=ControlFileName,STATUS='OLD',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 3 , CntrlFP,ControlFileName, IERR
3               FORMAT ( ' ERROR OPENING UNIT=',I2,' FILE NAME = ',A15, ', IOSTAT = ',I8)
                STOP 3
        ENDIF
        READ (CntrlFP, FMT='(A)') Title
        READ (CntrlFP, *) Prefix
        READ (CntrlFP, *) N
        OPEN (UNIT=OutFP,FILE=Prefix,POSITION='APPEND',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 7 , OutFP, OutputFile, IERR
7               FORMAT (' ERROR OPENING UNIT=',I2,' FILE NAME =' ,A13,', IOSTAT = ',I8)
                STOP 7
        END IF
        WRITE (OutFP, '(A)') Title
        WRITE (OutFP, '(A)') '**** Agreement indexes ****'
        WRITE (OutFP, *)

        ! read U partition
        ALLOCATE(U(N))
        READ (CntrlFP, *) R
        WRITE (OutFP, 20) 'U', R
        ALLOCATE(NU(R))
        READ(CntrlFP,*) (NU(I), I=1,R)
        DO I=1, R
```

```
                ALLOCATE( temp( NU(I) ) )
                READ(CntrlFP,*) (temp(J), J=1, NU(I))
                WRITE (OutFP, *) (temp(J), J=1, NU(I))
                DO J=1, NU(I)
                   U(temp(J)) = I
                END DO
                DEALLOCATE(temp)
        END DO
        WRITE (OutFP, *)
        ! read V partition
        ALLOCATE(V(N))
        READ (CntrlFP, *) C
        WRITE (OutFP, 20) 'V', C
        ALLOCATE(NV(C))
        READ(CntrlFP,*) (NV(I), I=1,C)
        DO I=1, C
                ALLOCATE( temp( NV(I) ) )
                READ(CntrlFP,*) (temp(J), J=1, NV(I))
                WRITE (OutFP, *) (temp(J), J=1, NV(I))
                DO J=1, NV(I)
                   V(temp(J)) = I
                END DO
                DEALLOCATE(temp)
        END DO
        WRITE (OutFP, *)
20      FORMAT('Partition ',A1,': ',I2,' Clusters')
        ! implement Agreement indexes.
        CALL CalculateAgreementIndexes(N, U, V, R, C)
        WRITE (OutFP, *)
        DEALLOCATE(U)
        DEALLOCATE(V)
        DEALLOCATE(NU)
        DEALLOCATE(NV)
        CLOSE (UNIT=OutFP)
END SUBROUTINE ControlFile
! ========================================================================
SUBROUTINE CalculateAgreementIndexes(N, U, V, R, C)
    USE Global
    IMPLICIT NONE

        INTEGER, INTENT(IN) :: N, R, C
        INTEGER, DIMENSION(N), INTENT(IN) :: U, V
        INTEGER :: I, J, AllocatStatus
        INTEGER, ALLOCATABLE, DIMENSION(:,:) :: CT  ! Contingency Table
        INTEGER, ALLOCATABLE, DIMENSION(:) :: Ni_, N_j  ! Sum rows and cols in CT
        REAL :: Exp1, Exp2, Exp3, Exp4, Rand
        REAL :: JExp1, JExp2, JExp3, Jaccard

        ALLOCATE( CT(R,C), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 )  STOP  "(CT) ** No Enough Memory **"
        ALLOCATE( Ni_(R) )
        ALLOCATE( N_j(C) )

        ! construct Contingency Table
        CT = 0
        DO I=1, N
          CT( U(I), V(I) ) = CT( U(I), V(I) ) + 1
        END DO

        WRITE (OutFP, *) 'Contingency Table:'
        DO I=1, R
                WRITE (OutFP, '(10I6)') (CT(I,J), J=1,C)
        END DO
        WRITE (OutFP, *)
        DO I=1, R! to sum rows
          Ni_(I) = SUM( CT(I,1:C) )
        END DO
        WRITE (OutFP, *) 'Sum rows (Ni.)'
        WRITE (OutFP,'(10I6)') (Ni_(I), I=1,R)
```

```
        WRITE (OutFP, *)
        DO J=1, C              ! to sum cols
          N_j(J) = SUM( CT(1:R,J) )
        END DO
        WRITE (OutFP, *) 'Sum cols (N.j)'
        WRITE (OutFP, '(10I6)') (N_j(J), J=1,C)
        WRITE (OutFP, *)
        Exp1 = 0.0
        JExp1 = 0.0
        DO I=1, R
          DO J=1, C
                Exp1 = Exp1 + REAL(CT(I,J))*REAL(CT(I,J)-1) / 2.0
                JExp1 = JExp1 + REAL(CT(I,J))**2.0)
          END DO
        END DO
        Exp2 = 0.0
        JExp2 = 0.0
        DO I=1, R
                Exp2 = Exp2 + REAL(Ni_(I))*REAL(Ni_(I)-1) / 2.0
                JExp2 = JExp2 + REAL(Ni_(I))**2.0)
        END DO
        Exp3 = 0.0
        JExp3 = 0.0
        DO J=1, C
                Exp3 = Exp3 + REAL(N_j(J))*REAL(N_j(J)-1) / 2.0
                JExp3 = JExp3 + REAL(N_j(J))**2.0)
        END DO
        Exp4 = REAL(N*(N-1)) / 2.0
        Rand = (Exp4 + 2.0*Exp1 - (Exp2+Exp3) ) / Exp4
        WRITE (OutFP, 20) Rand
        WRITE (OutFP, *)
20      FORMAT('The Rand Agreement Index= ',F6.3)
        Jaccard = (JExp1-N) / (JExp2+JExp3-JExp1-N)
        WRITE (OutFP, 30) Jaccard
        WRITE (OutFP, *)
30      FORMAT('The Jaccard Agreement Index= ',F6.3)
        DEALLOCATE( CT, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 )  STOP  "(CT)** Deallocation Error***
        DEALLOCATE( Ni_ )
        DEALLOCATE( N_j )
END SUBROUTINE CalculateAgreementIndexes
! ========================================================================
```

PCA and Quartimax rotation

```
! ************************************************************************************************
! File Name: quartimax.f
! Author: Ahmad Alhamed
! This program computes the PC loadings (matrix A) and then rotates using Quartimax (matrix B).
! It calls other subroutines and functions from CAlib.f.
! ************************************************************************************************

MODULE Global
    IMPLICIT NONE

END MODULE Global

PROGRAM Main
    IMPLICIT NONE

    CHARACTER(LEN=20) :: argv

    CALL GETARG(1, argv)

    IF ( LEN(TRIM(argv)) == 0 )  THEN
        CALL UserInterface
    ELSE
        CALL ControlFile(argv)
    END IF

END PROGRAM Main
! ========================================================================
SUBROUTINE UserInterface
    USE Global
    IMPLICIT NONE
! ***************************************************************
!   Show the user interface, and obtain all required input
! ***************************************************************
    CHARACTER(LEN=30) :: InputFile, OutputFile
    CHARACTER(LEN=80) :: Title
    CHARACTER(LEN=25) :: init
        INTEGER :: NormalizeMenu, GrammianMenu      ! function
    INTEGER :: nobs, nvar, Normalize, Grammian, IERR

        PRINT '($,A)','Enter Title:  '
        READ (*, FMT='(A)') Title
    PRINT '($,A)','Enter No. of vanables (col) in the Data Matrix: '
    READ *, nvar
    PRINT '($,A)','Enter No. of observation (rows) in the Data Matrix: '
    READ *, nobs
        PRINT '($,A)','Enter Input File Name:  '
        READ *, InputFile
        PRINT '($,A)','Enter Initial for Output Files Names:  '
        READ *, init
        ! show normalize menu
        Normalize = NormalizeMenu()
        ! show grammian menu
        Grammian = GrammianMenu()
    OutputFile = TRIM(init)//'Qmax.out'
    ! open output file name
    OPEN (UNIT=7,FILE=OutputFile,STATUS='NEW',IOSTAT=IERR)
    IF (IERR /= 0) THEN
        PRINT 7 , OutputFile, IERR
7       FORMAT (' ERROR OPENING UNIT=7, FILE NAME ='&
        & ,A13,', IOSTAT = ',I8)
        STOP 7
    END IF

        WRITE(7,*)  TRIM(Title)
        SELECT CASE(Grammian)
        CASE (1)
                WRITE(7,*) 'Based on Second Moment Matrix'
        CASE (2)
```

```
                WRITE(7,*) 'Based on Covariance Matrix'
        CASE (3)
                WRITE(7,*) 'Based on Correlation Matrix'
        CASE (4)
                WRITE(7,*) 'Based on Euclidean similarity Matrix'
        END SELECT

    SELECT CASE(Normalize)
    CASE (1)
        WRITE(7,*) 'Centered (by col mean)'
    CASE (2)
        WRITE(7,*) 'Normalized (by col max)'
    CASE (3)
        WRITE(7,*) 'Normalized (by col min, max)'
    CASE (4)
        WRITE(7,*) 'Normalized (by max)'
    END SELECT
        WRITE (7,*)

        CALL pca(nobs, nvar, InputFile, Normalize, Grammian)

        CLOSE(7)

END SUBROUTINE UserInterface
! ========================================================================
SUBROUTINE ControlFile(ControlFileName)
    USE Global
    IMPLICIT NONE
! ***********************************************************************
!   Read the required input from input file called 'Control File'
! ***********************************************************************
        CHARACTER(LEN=*), INTENT(IN) :: ControlFileName
    CHARACTER(LEN=30) :: InputFile, OutputFile
    CHARACTER(LEN=80) :: Title
    CHARACTER(LEN=25) :: init, Newinit
    INTEGER :: nvar, nobs, Grammian, Normalize, IERR, I


    OPEN (UNIT=5,FILE=ControlFileName,STATUS='OLD',IOSTAT=IERR)
    IF (IERR /= 0) THEN
        PRINT 5 , ControlFileName, IERR
5       FORMAT ( ' ERROR OPENING UNIT=5, FILE NAME = ',A15,&
        &  ', IOSTAT = ',I8)
        STOP 5
    ENDIF

        READ (5, FMT='(A)') Title
    READ (5, *) nobs, nvar
        READ (5, *) InputFile
    READ (5, *) Normalize
    READ (5, *) Grammian
        READ (5, *) init

        OutputFile = TRIM(init)//'Qmax.out'
        ! open output file name
    OPEN (UNIT=7,FILE=OutputFile,STATUS='NEW',IOSTAT=IERR)
    IF (IERR /= 0) THEN
        PRINT 7 , OutputFile, IERR
7       FORMAT (' ERROR OPENING UNIT=7, FILE NAME ='&
        & ,A13,', IOSTAT = ',I8)
        STOP 7
    END IF

        WRITE(7,*)  TRIM(Title)
        SELECT CASE(Grammian)
        CASE (1)
                WRITE(7,*) 'Based on Second Moment Matrix'
        CASE (2)
                WRITE(7,*) 'Based on Covariance Matrix'
```

```
        CASE (3)
                WRITE(7,*) 'Based on Correlation Matrix'
        CASE (4)
                WRITE(7,*) 'Based on Euclidean similarity Matrix'
        END SELECT

    SELECT CASE(Normalize)
    CASE (1)
        WRITE(7,*) 'Centered (by col mean)'
    CASE (2)
        WRITE(7,*) 'Normalized (by col max)'
    CASE (3)
        WRITE(7,*) 'Normalized (by col min, max)'
    CASE (4)
        WRITE(7,*) 'Normalized (by max)'
    END SELECT
        WRITE (7, *)
        CALL pca(nobs, nvar, InputFile, Normalize, Grammian)

    CLOSE(5)
    CLOSE(7)
END SUBROUTINE ControlFile
! ========================================================================
SUBROUTINE Rotate(nvar, A)
        USE Global
    IMPLICIT NONE

! ************************************************************************
! * This subroutine program rotate PC loadings (A) using Quartimax *
! ************************************************************************
        INTEGER, INTENT(IN) :: nvar
        REAL, DIMENSION(nvar,nvar), INTENT(IN) :: A  ! unrotated PCs
        INTEGER :: nf, lda, norm, maxit, ldb, ldt
        REAL :: w, eps
        INTEGER :: i, j, r, AllocatStatus
!       nfA unrotated PCs truncated to nf,      B rotated pc loadings, T rotation matrix
        REAL, ALLOCATABLE, DIMENSION(:,:) :: nfA, B, T

        IF (nvar==10) THEN
                r = 5
        ELSE
                r = 7
        END IF
        ! setup the parameters of IMSL's frota subroutine
        lda=nvar
        norm=0
        maxit=100
        w=0.0
        eps=0.0001
        ldb=nvar

        DO i=2, r
                nf=i
                ldt=nf
                ALLOCATE( nfA(nvar,nf))
                DO j=1, nf
                        nfA(1:nvar,j) = A(1:nvar,j)
                END DO
                ALLOCATE( B(nvar,nf))
                ALLOCATE( T(nf,nf))
                CALL FROTA(nvar,nf,nfA,lda,norm,maxit,w,eps,B,ldb,T,ldt)
                WRITE(7,*) 'Quartimax rotated solution for', nf, 'PCs'
                CALL PrintMatrix(nvar, nf, B)
                DEALLOCATE( nfA )
                DEALLOCATE( B )
                DEALLOCATE( T )
        END DO
END SUBROUTINE Rotate
! ========================================================================
```

```
! ************************************************************************************************************
! File Name: promax.f
! Author: Ahmad Alhamed
! This program computes the PC loadings (matrix A) and then rotates using Promax.
! It calls other subroutines and functions from CAlib.f.
! ************************************************************************************************************

MODULE Global
    IMPLICIT NONE

END MODULE Global

PROGRAM Main
    IMPLICIT NONE

    CHARACTER(LEN=20) :: argv

    CALL GETARG(1, argv)

    IF ( LEN(TRIM(argv)) == 0 )  THEN
        CALL UserInterface
    ELSE
        CALL ControlFile(argv)
    END IF

END PROGRAM Main
! ========================================================================
SUBROUTINE UserInterface
    USE Global
    IMPLICIT NONE
! ********************************************************************
!   Show the user interface, and obtain all required input
! ********************************************************************
    CHARACTER(LEN=30) :: InputFile, OutputFile
    CHARACTER(LEN=80) :: Title
    CHARACTER(LEN=25) :: init
        INTEGER :: NormalizeMenu, GrammianMenu      ! function
    INTEGER :: nobs, nvar, Normalize, Grammian, IERR

        PRINT '($,A)','Enter Title: '
        READ (*, FMT='(A)') Title
    PRINT '($,A)','Enter No. of vanables (col) in the Data Matrix: '
    READ *, nvar
    PRINT '($,A)','Enter No. of observation (rows) in the Data Matrix: '
    READ *, nobs
        PRINT '($,A)','Enter Input File Name: '
        READ *, InputFile
        PRINT '($,A)','Enter Initial for Output Files Names: '
        READ *, init
        ! show normalize menu
        Normalize = NormalizeMenu()
        ! show grammian menu
        Grammian = GrammianMenu()
    OutputFile = TRIM(init)//'Pmax.out'
    ! open output file name
    OPEN (UNIT=7,FILE=OutputFile,STATUS='NEW',IOSTAT=IERR)
    IF (IERR /= 0) THEN
        PRINT 7 , OutputFile, IERR
7       FORMAT (' ERROR OPENING UNIT=7, FILE NAME ='&
        & ,A13,', IOSTAT = ',I8)
        STOP 7
    END IF

        WRITE(7,*)  TRIM(Title)
        SELECT CASE(Grammian)
        CASE (1)
                WRITE(7,*) 'Based on Second Moment Matrix'
        CASE (2)
```

```
                WRITE(7,*) 'Based on Covariance Matrix'
        CASE (3)
                WRITE(7,*) 'Based on Correlation Matrix'
        CASE (4)
                WRITE(7,*) 'Based on Euclidean similarity Matrix'
        END SELECT

    SELECT CASE(Normalize)
    CASE (1)
        WRITE(7,*) 'Centered (by col mean)'
    CASE (2)
        WRITE(7,*) 'Normalized (by col max)'
    CASE (3)
        WRITE(7,*) 'Normalized (by col min, max)'
    CASE (4)
        WRITE(7,*) 'Normalized (by max)'
    END SELECT
        WRITE (7,*)

        CALL pca(nobs, nvar, InputFile, Normalize, Grammian)

        CLOSE(7)

END SUBROUTINE UserInterface
! ========================================================================
SUBROUTINE ControlFile(ControlFileName)
    USE Global
    IMPLICIT NONE
! ***********************************************************************
!   Read the required input from input file called 'Control File'
! ***********************************************************************
        CHARACTER(LEN=*), INTENT(IN) :: ControlFileName
    CHARACTER(LEN=30) :: InputFile, OutputFile
    CHARACTER(LEN=80) :: Title
    CHARACTER(LEN=25) :: init, Newinit
    INTEGER :: nvar, nobs, Grammian, Normalize, IERR, I


    OPEN (UNIT=5,FILE=ControlFileName,STATUS='OLD',IOSTAT=IERR)
    IF (IERR /= 0) THEN
        PRINT 5 , ControlFileName, IERR
5       FORMAT ( ' ERROR OPENING UNIT=5, FILE NAME = ',A15,&
        &  ', IOSTAT = ',I8)
        STOP 5
    ENDIF

        READ (5, FMT='(A)') Title
    READ (5, *) nobs, nvar
        READ (5, *) InputFile
    READ (5, *) Normalize
    READ (5, *) Grammian
        READ (5, *) init

        OutputFile = TRIM(init)//'Pmax.out'
        ! open output file name
    OPEN (UNIT=7,FILE=OutputFile,STATUS='NEW',IOSTAT=IERR)
    IF (IERR /= 0) THEN
        PRINT 7 , OutputFile, IERR
7       FORMAT (' ERROR OPENING UNIT=7, FILE NAME ='&
        & ,A13,', IOSTAT = ',I8)
        STOP 7
    END IF

        WRITE(7,*)  TRIM(Title)
        SELECT CASE(Grammian)
        CASE (1)
                WRITE(7,*) 'Based on Second Moment Matrix'
        CASE (2)
                WRITE(7,*) 'Based on Covariance Matrix'
```

```
        CASE (3)
                WRITE(7,*) 'Based on Correlation Matrix'
        CASE (4)
                WRITE(7,*) 'Based on Euclidean similarity Matrix'
        END SELECT

    SELECT CASE(Normalize)
    CASE (1)
        WRITE(7,*) 'Centered (by col mean)'
    CASE (2)
        WRITE(7,*) 'Normalized (by col max)'
    CASE (3)
        WRITE(7,*) 'Normalized (by col min, max)'
    CASE (4)
        WRITE(7,*) 'Normalized (by max)'
    END SELECT
        WRITE (7, *)


        CALL pca(nobs, nvar, InputFile, Normalize, Grammian)

    CLOSE(5)
    CLOSE(7)

END SUBROUTINE ControlFile
! ========================================================================
SUBROUTINE Rotate(nvar, A)
        USE Global
    IMPLICIT NONE

! ***************************************************************
! *  This subroutine rotate PC loadings (A) using Promax        *
! *                                                             *
! ***************************************************************
        INTEGER, INTENT(IN) :: nvar
        REAL, DIMENSION(nvar,nvar), INTENT(IN) :: A  ! unrotated PCs

        INTEGER :: nf, lda, norm, maxit, ldb, ldt, imth
        REAL :: w, eps
        INTEGER :: i, j, r, AllocatStatus
!       nfA unrotated PCs truncated to nf
!   B rotated pc loadings, T rotation matrix, TRGT target matrix
!       FCOR components correlation
        REAL, ALLOCATABLE, DIMENSION(:,:) :: nfA, B, T, TRGT, FCOR
        REAL, ALLOCATABLE, DIMENSION(:) :: F

        IF (nvar==10) THEN
                r = 5
        ELSE
                r = 7
        END IF
        ! setup the parameters of IMSL's frota subroutine
        lda=nvar
        imth=1    ! Promax method
        norm=0
        maxit=100
        w=0.0     ! Quartimax
        eps=0.0001
        ldb=nvar

        DO i=2, r
                nf=i
                ldt=nf
                ALLOCATE( nfA(nvar,nf))
                DO j=1, nf
                        nfA(1:nvar,j) = A(1:nvar,j)
                END DO
                ALLOCATE( B(nvar,nf))
                ALLOCATE( T(nf,nf))
```

```
                ALLOCATE( F(nf))
                ALLOCATE( TRGT(nvar,nf))
                ALLOCATE( FCOR(nf,nf))

                F = 4.0    ! power for promax
                CALL FPRMX(nvar,nf,nfA,lda,imth,norm,w,maxit,eps,F,TRGT, &
                & nvar,B,ldb,T,ldt,FCOR,nf)

                WRITE(7,*) 'Promax rotated solution (based on Quartimax) &
                & for', nf, 'PCs'
                CALL PrintMatrix(nvar, nf, B)
                WRITE(7,*) 'Correlation matrix for the',nf,'PCs'
                CALL PrintMatrix(nf, nf, FCOR)

                DEALLOCATE( nfA )
                DEALLOCATE( B )
                DEALLOCATE( T )
                DEALLOCATE( F )
                DEALLOCATE( TRGT )
                DEALLOCATE( FCOR )
        END DO

END SUBROUTINE Rotate
! ========================================================================
```

```
! ************************************************************************************************
! File Name: CAlib.f
! Author: Ahmad Alhamed
! This files contains severl subroutines and functions that are used by different programs
! ************************************************************************************************
SUBROUTINE ReadDataMatrixFromFile(Row, Col, DataMatrix, InputFile)
    IMPLICIT NONE

        INTEGER, INTENT(IN) :: Row, Col
        CHARACTER(LEN=*), INTENT(IN) :: InputFile
        REAL, DIMENSION(Row,Col), INTENT(OUT) :: DataMatrix
        INTEGER I, J, IERR, JERR

        OPEN (UNIT=9,FILE=InputFile,STATUS='OLD',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 8 , InputFile, IERR
8               FORMAT ( ' ERROR OPENING UNIT=9, FILE NAME = ',A15,  ', IOSTAT = ',I8)
                STOP 8
        ENDIF
        DO I=1, Row
                READ (9, *) (DataMatrix(I,J), J=1,Col)
        END DO
        CLOSE (UNIT=9)
END SUBROUTINE ReadDataMatrixFromFile
! ========================================================================
SUBROUTINE ReadResemblanceMatrixFromFile(N, ResemblanceMatrix, InputFile)
    IMPLICIT NONE

        INTEGER, INTENT(IN) :: N
        CHARACTER(LEN=*), INTENT(IN) :: InputFile
        REAL, DIMENSION(N*(N-1)/2), INTENT(OUT) :: ResemblanceMatrix
        INTEGER I, IERR, JERR

        OPEN (UNIT=9,FILE=InputFile,STATUS='OLD',IOSTAT=IERR)
        IF (IERR /= 0) THEN
                PRINT 8 , InputFile, IERR
8               FORMAT ( ' ERROR OPENING UNIT=9, FILE NAME = ',A15, ', IOSTAT = ',I8)
                STOP 8
        ENDIF
        READ (9, *) (ResemblanceMatrix(I), I=1, N*(N-1)/2 )
        CLOSE (UNIT=9)
END SUBROUTINE ReadResemblanceMatrixFromFile
! ========================================================================
SUBROUTINE PrintDataMatrix(Row, Col, DataMatrix)
    IMPLICIT NONE

        INTEGER, INTENT(IN) :: Row, Col
        REAL, DIMENSION(Row,Col), INTENT(IN) :: DataMatrix
        INTEGER :: I, J

        WRITE (7, 10) Row, Col
        WRITE (7, *) '-------------------------'
10      FORMAT ('   DATA MATRIX (',I3,' x ',I3,')' )
        ! print col and row number
        WRITE (7, '($,A)' ) '    '
        WRITE (7, 20) (I, I=1, Col)
20      FORMAT ('    [',I2,'] ',$)
        WRITE (7, *)
        DO I=1, Row
                WRITE (7, 25) I
                WRITE (7,FMT='(50F10.2)') (DataMatrix(I,J),J=1,Col)
        END DO
25      FORMAT ('[',I2,'] ',$)
        WRITE (7, *)
END SUBROUTINE PrintDataMatrix
! ========================================================================
SUBROUTINE PrintResemblanceMatrix(N, ResemblanceMatrix)
```

```
    IMPLICIT NONE

        INTEGER, INTENT(IN) :: N
        REAL, DIMENSION(N*(N-1)/2), INTENT(IN) :: ResemblanceMatrix
        INTEGER :: Offset   ! Func. returns index in Resemblance Matrix
        INTEGER :: I, J

        WRITE (7, '($,A)' ) '    '
        WRITE (7, 30) (I, I=1, N-1)
30      FORMAT ('      [',I2,'] ',$)
        WRITE (7, *)
        DO I=2, N
                WRITE (7, 35) I
                WRITE (7,FMT='(30F12.3)')  ( ResemblanceMatrix(Offset(N,I,J)), J=1, I-1)
        END DO
35      FORMAT ('[',I2,'] ',$)
        WRITE (7, *)
END SUBROUTINE PrintResemblanceMatrix
! ========================================================================
INTEGER FUNCTION NormalizeMenu()
    IMPLICIT NONE

    INTEGER :: Norm

    PRINT *
    PRINT *,'-------------------------------------------------------'
    PRINT *,'Centering or Normalization:'
    PRINT *,'   (0) Do Not Normalize'
    PRINT *,'   (1) By the Mean of each Column (Col. by Col.)'
    PRINT *,'   (2) By the Max. of each Column (Col. by Col.)'
    PRINT *,'   (3) By the Min. and Max. of each Column (Col. by Col.)'
    PRINT *,'   (4) By the Max. of the data matrix'
    PRINT *,'-------------------------------------------------------'
    PRINT '($,A)','   Enter Choice ===>  '
    DO
        READ *, Norm
        IF (Norm >= 0 .AND. Norm <= 4)  EXIT
    END DO
    NormalizeMenu = Norm
END FUNCTION NormalizeMenu
! ========================================================================
SUBROUTINE CenterByColMean(M, N, X)
    IMPLICIT NONE
        ! ********************************************
        ! Center matrix X by the mean of each col
        ! Zij = Xij - MEANj   , 1<=I<=M  , 1<=J<=N
        ! ********************************************

        INTEGER, INTENT(IN) :: M, N
        REAL, DIMENSION(M, N), INTENT(INOUT) :: X
        INTEGER :: I, J, K, AllocatStatus
        REAL, ALLOCATABLE, DIMENSION(:)   :: Mean

        ALLOCATE( Mean(N) , STAT = AllocatStatus )
        IF (AllocatStatus/= 0 )  STOP  "( Mean ) "" No Enough Memory """
        ! comput Mean of each col.
        DO J=1, N
                Mean(J) = SUM( X(1:M,J) ) / REAL(M)
        END DO
        ! subtract Mean of each col from every element in this col.
        DO J=1, N
                X(1:M,J) = X(1:M,J) - Mean(J)
        END DO
        DEALLOCATE( Mean, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 )  STOP  "( Mean )"" Deallocation Error"""
END SUBROUTINE CenterByColMean
! ========================================================================
SUBROUTINE NormalizeByColMax(M, N, X)
    IMPLICIT NONE
```

```
        ! ********************************************
        ! Normalize matrix X by the max of each col
        ! Zij = Xij/MAXj  , 1<=I<=M , 1<=J<=N
        ! ********************************************

        INTEGER, INTENT(IN) :: M, N
        REAL, DIMENSION(M, N), INTENT(INOUT) :: X
        INTEGER :: I, J, K, AllocatStatus
        REAL, ALLOCATABLE, DIMENSION(:)  :: Mx

        ALLOCATE( Mx(N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( Mx ) ** No Enough Memory **"
        ! comput Max of each col.
        DO J=1, N
                Mx(J) = MAXVAL( X(1:M,J) )
        END DO
        ! divede every element of each col by its Max.
        DO J=1, N
                X(1:M,J) = X(1:M,J) / Mx(J)
        END DO
        DEALLOCATE( Mx, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "( Mx )** Deallocation Error**"
END SUBROUTINE NormalizeByColMax
! ========================================================================
SUBROUTINE NormalizeByColMinMax(M, N, X)
    IMPLICIT NONE
        ! ********************************************
        ! Normalize matrix X by the min and max of
        ! each col Zij=(Xij-MINj)/(MAXj-MINj) ,1<=I<=M
        ! ********************************************

        INTEGER, INTENT(IN) :: M, N
        REAL, DIMENSION(M, N), INTENT(INOUT) :: X
        INTEGER :: I, J, K, AllocatStatus
        REAL, ALLOCATABLE, DIMENSION(:)  :: Mn, Mx, Av

        ALLOCATE( Mn(N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( Mn ) ** No Enough Memory **"
        ALLOCATE( Mx(N), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( Mx ) ** No Enough Memory **"
        ! comput Min, Max of each col.
        DO J=1, N
                Mn(J) = MINVAL( X(1:M,J) )
                Mx(J) = MAXVAL( X(1:M,J) )
        END DO
        ! normalize by Min, Max of each col.
        DO J=1, N
                X(1:M,J) = ( X(1:M,J)-Mn(J) ) / ( Mx(J)-Mn(J) )
        END DO
        DEALLOCATE( Mn, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "( Mn )** Deallocation Error**"
        DEALLOCATE( Mx, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "( Mx )** Deallocation Error**"
END SUBROUTINE NormalizeByColMinMax
! ========================================================================
SUBROUTINE NormalizeByMax(M, N, X)
    IMPLICIT NONE
        ! ********************************************
        ! Normalize matrix X by the max of X
        !  Zij = Xij/MAX , 1<=I<=M, 1<=J<=N
        ! ********************************************

        INTEGER, INTENT(IN) :: M, N
        REAL, DIMENSION(M, N), INTENT(INOUT) :: X
        INTEGER :: I, J, K, AllocatStatus
        REAL :: Mx

        Mx = MAXVAL(X)
        X = X / Mx
```

```
END SUBROUTINE NormalizeByMax
! ========================================================================
INTEGER FUNCTION SeedPointMenu()
    IMPLICIT NONE

        INTEGER :: Choice

        PRINT *,'-------------------------------------'
        PRINT *,'            SeedPoints Menu'
        PRINT *,'(1) Choose the first K observations in the data set'
        PRINT *,'(2) Choose observations correspond to n/k, 2n/k, (k-1)n/k, n'
        PRINT *,'(3) Subjectively chooes any K observations from the data set'
        PRINT *,'(4) Choose observations correspond to K random numbers '
        PRINT *,'(5) Choose any desired partition of the observations into'
        PRINT *,'   K mutually exclusive clusters and compute the cluster'
        PRINT *,'   centroids as seed points'
        PRINT *,'-------------------------------------'
        PRINT '($,A)','   Enter Choice ===> '
        DO
                READ *, Choice
                IF (Choice >= 1 .AND. Choice <= 5) EXIT
        END DO
        SeedPointMenu = choice
END FUNCTION SeedPointMenu
! ========================================================================
SUBROUTINE ObtainSeedPoints(M, N, X, K, SeedPointChoice, Seeds)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: M, N, K, SeedPointChoice
        REAL, DIMENSION(N,M), INTENT(IN) :: X
        REAL, DIMENSION(K,M), INTENT(OUT) :: Seeds
        INTEGER, ALLOCATABLE, DIMENSION(:) :: CNTR, INDX
        REAL, ALLOCATABLE, DIMENSION(:) :: OneSeed
        INTEGER :: I, J, R
        REAL :: r1
        LOGICAL :: IsIn                 ! function

        WRITE(OutFP,'(A)') '   Seed Points'
        SELECT CASE(SeedPointChoice)
        CASE (1)
                DO I=1, K
                        Seeds(I,1:M) = X(I,1:M)
                END DO
                WRITE(OutFP,'(4X,30I4)') (I, I=1,K)
        CASE (2)
                ALLOCATE(INDX(K))
                DO I=1, K
                        INDX(I) = INT( REAL(I*N)/REAL(K) )
                END DO
                WRITE(OutFP,'(4X,30I4)') (INDX(I), I=1,K)
                DO I=1, K
                        Seeds(I,1:M) = X(INDX(I),1:M)
                END DO
                DEALLOCATE(INDX)
        CASE (3)
                ALLOCATE(INDX(K))
                READ(CntrlFP,*) (INDX(I), I=1,K)
                WRITE(OutFP,'(4X,30I4)') (INDX(I), I=1,K)
                DO I=1, K
                        Seeds(I,1:M) = X(INDX(I),1:M)
                END DO
                DEALLOCATE(INDX)
        CASE (4)
                ALLOCATE(INDX(K))
                INDX = 0
                CALL RANDOM_SEED
                DO I=1, K
                        DO
```

```
                                CALL RANDOM_NUMBER(r1)
                                R = INT(r1*10.0)
                                IF(.NOT. IsIn(R,K,INDX) ) THEN
                                        EXIT
                                END IF
                        END DO
                        INDX(I) = R
                        Seeds(I,1:M) = X(INDX(I),1:M)
                END DO
                WRITE(OutFP,'(4X,30I4)') (INDX(I), I=1,K)
                DEALLOCATE(INDX)
        CASE (5)
                WRITE(OutFP,'($,A)') '   <centroids> '
                ALLOCATE(CNTR(K))
                READ(CntrlFP,*) (CNTR(I), I=1,K)
                DO I=1, K
                        ALLOCATE( INDX( CNTR(I) ) )
                        ALLOCATE(OneSeed(M))
                        READ(CntrlFP,*) (INDX(J), J=1,CNTR(I))
                        CALL CalcualteCentroid(M,N,CNTR(I),INDX,X,OneSeed)
                        Seeds(I,1:M) = OneSeed
                        ! to print seeds
                        WRITE(OutFP,'($,A)') '{'
                        DO J=1, CNTR(I)
                                WRITE(OutFP,'($,I3)')  INDX(J)
                        END DO
                        WRITE(OutFP,'($,A)') '} '
                        DEALLOCATE(INDX, OneSeed)
                END DO
                DEALLOCATE(CNTR)
                WRITE(OutFP,*)
        END SELECT
END SUBROUTINE ObtainSeedPoints
! ============================================================================
SUBROUTINE WriteSubTitle1(Normalize)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: Normalize

        SELECT CASE(Normalize)
        CASE(0)
        CASE (1)
                WRITE(OutFP,'(A)') 'Centered (col)'
        CASE (2)
                WRITE(OutFP,'(A)') 'Normalized (col max)'
        CASE (3)
                WRITE(OutFP,'(A)') 'Normalized (col min,max)'
        CASE (4)
                WRITE(OutFP,'(A)') 'Normalized (max)'
        END SELECT
END SUBROUTINE WriteSubTitle1
! ============================================================================
SUBROUTINE WriteSeedPointsSubTitle(K, SeedPointChoice)
        USE Global
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: K, SeedPointChoice

        SELECT CASE(SeedPointChoice)
        CASE (1)
                WRITE(OutFP,'(A)') 'Seeds: First K'
        CASE (2)
                WRITE(OutFP,'(A)') 'Seeds: n/k, 2n/k, (k-1)n/k, n'
        CASE (3)
                WRITE(OutFP,'(A)') 'Seeds: Subjectively chosen'
        CASE (4)
                WRITE(OutFP,'(A)') 'Seeds: Randomly chosen'
        CASE (5)
```

```
                WRITE(OutFP,'(A)') 'Seeds: Centroid of subjectively partitioning'
        END SELECT
END SUBROUTINE WnteSeedPointsSubTitle
! ========================================================================
SUBROUTINE CalcualteCentroid(M,N,NoOfVectors,Indecis,X, Centroid)
        IMPLICIT NONE

        INTEGER, INTENT(IN) :: M, N, NoOfVectors
        INTEGER, DIMENSION(NoOfVectors), INTENT(IN) :: Indecis
        REAL, DIMENSION(N,M), INTENT(IN) :: X
        REAL, DIMENSION(M), INTENT(OUT) :: Centroid
        INTEGER :: I

        Centroid = 0.0
        DO I=1,NoOfVectors
                Centroid = Centroid + X( Indecis(I), 1:M)
        END DO
        Centroid = Centroid / REAL(NoOfVectors)
END SUBROUTINE CalcualteCentroid
! ========================================================================
LOGICAL FUNCTION IsIn(Num,Size,Array)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: Num, Size
    INTEGER, DIMENSION(Size), INTENT(IN) :: Array
    INTEGER :: I
    LOGICAL :: Found

    Found = .FALSE.
    DO I=1, Size
        IF(Array(I) == Num) THEN
            Found = .TRUE.
            EXIT
        END IF
    END DO
    IsIn = Found
END FUNCTION IsIn
! ========================================================================
SUBROUTINE CalculateCenters(M, N, X, K, Membership, NC, Centers)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N, K
    REAL, DIMENSION(M,N), INTENT(IN) :: X
    INTEGER, DIMENSION(N), INTENT(IN) :: Membership
    INTEGER, DIMENSION(K), INTENT(IN) :: NC
    REAL, DIMENSION(M,K), INTENT(OUT) :: Centers
    INTEGER :: I, J

        Centers = 0.0
        DO I=1, N
          Centers(1:M, Membership(I)) = Centers(1:M, Membership(I)) + &
                & X(1:M,I)
        END DO
        DO I=1, K
          Centers(1:M, I) = Centers(1:M, I) / REAL( NC(I) )
        END DO
END SUBROUTINE CalculateCenters
! ========================================================================
SUBROUTINE Euclidean(M, N, Data, Resemblance)
    IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N
    REAL, DIMENSION(M,N), INTENT(IN) :: Data
    REAL, DIMENSION(N,N), INTENT(OUT) :: Resemblance
    REAL, ALLOCATABLE, DIMENSION(:) :: Vector
    INTEGER:: J, K, AllocatStatus

    ALLOCATE( Vector(M), STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 ) STOP "(Vector)" No Enough Memory ""
```

```
    DO J=1, N-1
        DO K=J+1, N
            Vector = Data(1:M, J) - Data(1:M, K)
            Resemblance(K,J) = SQRT(DOT_PRODUCT(Vector,Vector))
            Resemblance(J,K) = Resemblance(K,J)
        END DO
    END DO
    DO J=1, N
        Resemblance(J,J) = 0.0
    END DO
    DEALLOCATE( Vector, STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 ) STOP "(Vector)" Deallocation Error ""
END SUBROUTINE Euclidean
! ========================================================================

! The following subroutines are used by PCA programs: quartimax.f and promax.f

! ========================================================================
SUBROUTINE pca(nobs, nvar, InputFileName, Normalize, Grammian)
        USE Global
    IMPLICIT NONE

! ************************************************************
! *  This subroutine computes unrotated PC loadings and eignvalues*
! *  from computed correlation/covariance matrix.                 *
! *                                                               *
! ************************************************************
        INTEGER, INTENT(IN) :: nobs, nvar, Grammian, Normalize
        CHARACTER(LEN=*), INTENT(IN) :: InputFileName

        INTEGER :: icov,lda,ldcov,ldevec,ndf,ldb,maxit
        INTEGER :: i, j, r, AllocatStatus, icopt, nmiss
!       X data matrix, G Grammian(i.e., 2nd moment/covariance/correlation)
!   V eigenvectors, A unrotated pc loadings, B rotated pc loadings
        REAL, ALLOCATABLE, DIMENSION(:,:) :: X, G, V, A, incd
        REAL, ALLOCATABLE, DIMENSION(:)   :: eval, pct, std &
        & , sum_square_pc, sum_square_evec, xmean
        REAL :: eps,w, sum_eval, sumwt


        ALLOCATE( X(nobs,nvar), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( X ) "" No Enough Memory ""
        ALLOCATE( G(nvar,nvar), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( G ) "" No Enough Memory ""
        ALLOCATE( V(nvar,nvar), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( V ) "" No Enough Memory ""
        ALLOCATE( A(nvar,nvar), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( A ) "" No Enough Memory ""
        ALLOCATE( incd(1,1), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( incd ) "" No Enough Memory ""
        ALLOCATE( eval(nvar), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( eval ) "" No Enough Memory ""
        ALLOCATE( pct(nvar), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( pct ) "" No Enough Memory ""
        ALLOCATE( std(nvar), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( std ) "" No Enough Memory ""
        ALLOCATE( sum_square_pc(nvar), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( sum_square_pc ) "" No Enough Memory ""
        ALLOCATE( sum_square_evec(nvar), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( sum_square_evec ) "" No Enough Memory ""
        ALLOCATE( xmean(nvar), STAT = AllocatStatus )
        IF (AllocatStatus/= 0 ) STOP "( xmean ) "" No Enough Memory ""


        CALL ReadDataMatrixFromFile(nobs, nvar, X, InputFileName)
        ! normalize if requested
        SELECT CASE(Normalize)
        CASE (1)
                CALL CenterByColMean(nobs, nvar, X)
```

```
CASE (2)
        CALL NormalizeByColMax(nobs, nvar, X)
CASE (3)
        CALL NormalizeByColMinMax(nobs, nvar, X)
CASE (4)
        CALL NormalizeByMax(nobs, nvar, X)
END SELECT

! compute grammian (2nd moment, cov, or corr) matrix
SELECT CASE(Grammian)
CASE (1) ! second moment: 1/M(XtX)
        G = (1.0/REAL(nobs))*MATMUL(TRANSPOSE(X),X)
CASE (2)
        icopt = 0   ! for var-cov
        CALL CORVC(0,nobs,nvar,X,nobs,0,0,0,icopt,xmean,G,nvar,&
        & incd,1, nobs,nmiss,sumwt)
        icov = 0   ! to be used by PRINC
CASE (3)
        icopt = 2   ! for correlation
        CALL CORVC(0,nobs,nvar,X,nobs,0,0,0,icopt,xmean,G,nvar,&
        & incd,1, nobs,nmiss,sumwt)
        icov = 1   ! to be used by PRINC
CASE (4)
        icopt = 2   ! for correlation (EucSimi ~=~ Corr)
        CALL EuclideanSimilarity(nobs, nvar, X, G)
        icov = 1   ! to be used by PRINC
END SELECT


! setup the parameters of IMSL's pca subroutine
lda=nvar
ldcov=nvar
ldevec=nvar
ndf=nvar
ldb=nvar
maxit=100
eps=0.0001

CALL PRINC(ndf,nvar,G,ldcov,icov,eval,pct,std,V,ldevec,A,lda)

sum_eval=0.
DO i=1,nvar
        sum_eval=sum_eval+eval(i)
END DO

WRITE(7,*) '        Eigenvalues      var explained       cumulative'
WRITE(7,*) 'i         (Ei)             by Ei          explained by Ei'
WRITE(7,*) '---   ---------------   ---------------   ---------------'
DO i=1,nvar
        WRITE(7,'(I4,3F20.3)')i,eval(i),eval(i)/sum_eval,pct(i)
END DO
WRITE(7,*)
WRITE(7,*) 'Sum of the eigenvalues : '
WRITE(7,'(F20.3)')sum_eval
WRITE(7,*)


DEALLOCATE( X, STAT = AllocatStatus )
IF ( AllocatStatus /= 0 ) STOP "( X )** Deallocation Error***
DEALLOCATE( G, STAT = AllocatStatus )
IF ( AllocatStatus /= 0 ) STOP "( G )** Deallocation Error***
DEALLOCATE( V, STAT = AllocatStatus )
IF ( AllocatStatus /= 0 ) STOP "( V )** Deallocation Error***
DEALLOCATE( eval, STAT = AllocatStatus )
IF ( AllocatStatus /= 0 ) STOP "( eval )** Deallocation Error***
DEALLOCATE( pct, STAT = AllocatStatus )
IF ( AllocatStatus /= 0 ) STOP "( pct )** Deallocation Error***
DEALLOCATE( std, STAT = AllocatStatus )
IF ( AllocatStatus /= 0 ) STOP "( std )** Deallocation Error***
```

```
        DEALLOCATE( sum_square_pc, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "( sum_square_pc )** Deallocation Error***
        DEALLOCATE( sum_square_evec, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "( sum_square_evec )** Deallocation Error***

        CALL Rotate(nvar, A)

        DEALLOCATE( A, STAT = AllocatStatus )
        IF ( AllocatStatus /= 0 ) STOP "( A )** Deallocation Error***

END SUBROUTINE pca
! ========================================================================
INTEGER FUNCTION GrammianMenu()
    IMPLICIT NONE

    INTEGER :: G

    PRINT *
    PRINT *,'-------------------------------'
    PRINT *,'Grammian:'
    PRINT *,'   (1) Second Moment'
    PRINT *,'   (2) Covariance'
    PRINT *,'   (3) Correlation'
    PRINT *,'   (4) Euclidean Similarity'
    PRINT *,'-------------------------------'
    PRINT '($,A)','  Enter Choice ===>  '
    DO
        READ *, G
        IF (G >= 1 .AND. G <= 4) EXIT
    END DO
    GrammianMenu = G
END FUNCTION GrammianMenu
! ========================================================================
SUBROUTINE EuclideanSimilarity(M, N, Data, ES)
        IMPLICIT NONE

    INTEGER, INTENT(IN) :: M, N
    REAL, DIMENSION(M,N), INTENT(IN) :: Data
    REAL, DIMENSION(N,N), INTENT(OUT) :: ES
    REAL, ALLOCATABLE, DIMENSION(:) :: Vector
    INTEGER:: J, K, AllocatStatus
    REAL :: Mx

    ALLOCATE( Vector(M), STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 ) STOP "(Vector)** No Enough Memory ***

    DO J=1, N-1
        DO K=J+1, N
            Vector = Data(1:M, J) - Data(1:M, K)
            ES(K,J) = SQRT(DOT_PRODUCT(Vector,Vector))
                        ES(J,K) = ES(K,J)
        END DO
    END DO
    DO J=1, N
                ES(J,J) = 0.0
    END DO
    Mx = MAXVAL(ES)
    ES = ES / Mx
    DO J=1, N
        DO K=1, N
            ES(J,K) = 1.0 - ES(J,K)
        END DO
    END DO

    DEALLOCATE( Vector, STAT = AllocatStatus )
    IF ( AllocatStatus /= 0 ) STOP "(Vector)** Deallocation Error ***

END SUBROUTINE EuclideanSimilarity
! ========================================================================
```